**EARLY DETECTION OF CHRONIC KIDNEY DISEASE USING MACHINE LEARNING**

**PROJECT REPORT**

***Submitted* *by***

Sudharsana Suresh

Naveen U

Geethanjali B

Nithik N

Bharath B

of

**INFORMATION TECHNOLOGY**

**MADRAS INSTITUTE OF TECHNOLOGY**

|  |  |  |
| --- | --- | --- |
| **CHAPTER NO** | **TITTLE** | **PAGE NO** |
| **1** | **INTRODUCTION** |  |
| **1.1** | Project Overview | **4** |
| **1.2** | Purpose | **4** |
| **2** | **LITERATURE SURVEY** |  |
| **2.1** | Existing problem | **5** |
| **2.2** | References | **5** |
| **2.3** | Problem Statement Definition | **7** |
| **3** | **IDEATION & PROPOSED SOLUTION** |  |
| **3.1** | Empathy Map Canvas | **8** |
| **3.2** | Ideation & Brainstorming | **9** |
| **3.3** | Proposed Solution | **9** |
| **3.4** | Problem Solution fit | **12** |
| **4** | **REQUIREMENT ANALYSIS** |  |
| **4.1** | Functional requirement | **14** |
| **4.2** | Non-Functional requirements | **14** |
| **5** | **PROJECT DESIGN** |  |
| **5.1** | Data Flow Diagram | **15** |
| **5.2** | Solution & Technical Architecture | **16** |
| **5.3** | User Stories | **17** |
| **6** | **PROJECT PLANNING & SCHEDULING** |  |
| **6.1** | Sprint Planning & Estimation | **18** |
| **6.2** | Sprint Delivery Schedule | **18** |
| **6.3** | Reports from JIRA | **19** |
| **7** | **CODING & SOLUTIONING** |  |
| **7.1** | Feature 1 | **20** |
| **7.2** | Feature 2 | **29** |
| **7.3** | Database Schema | **29** |
| **8** | **RESULTS** |  |
| **8.1** | Deployment in IBM Cloud | **30** |
| **8.2** | Prediction Results | **31** |
| **8.3** | Performance Metrics | **33** |
| **9** | **CONCLUSION** | **34** |
| **10** | **FUTURE SCOPE** | **34** |
| **11** | **APPENDIX** | **35** |
|  | Source Code  GitHub & Project Demo Link |  |

**CHAPTER 1**

**INTRODUCTION**

**1.1 Project Overview**

Prediction of chronic kidney disease is one of the most crucial challenges in healthcare analytics. The most fascinating and difficult jobs in daily life because millions of people die each year due to lack of access to inexpensive treatment for chronic kidney disease (CKD), which affects one third of the adult population. If chronic kidney disease is addressed early on, it may be cured. The major goal of the project is to use diagnostic measurements like albumin and blood pressure to quickly, accurately, and painlessly determine if a patient has chronic kidney disease or not. Based on the information provided by the model, suitable treatment can then be administered.

**1.2 Purpose of the Project**

The project's goal is to warn medical professionals of kidney illness early on, ensuring a quick recovery or kidney disease prevention. The goal of this project is to use machine learning to develop a model for the early diagnosis of chronic kidney disease. Integration of the output into Flask is present. The user input on the numerous factors required to decide on the early identification of renal illness is collected through the front end designed in HTML. The IBM cloud is using the similar model

**CHAPTER 2**

**LITREATURE SURVEY**

**2.1 Existing problem**

In many countries today, kidney disease is discovered in its late stages, resulting in the loss of precious lives. There aren't many ways to spot them in the beginning. The majority of user information is still unconfirmed, making it challenging to identify bogus users. The application's user interface is not intuitive, and in order to engage with it, a user needs a device running the Android operating system and a working internet connection.

**2.2 References**

[1] B. Gudeti, S. Mishra, S. Malik, T. F. Fernandez, A. K. Tyagi and S. Kumari, "A Novel Approach to Predict Chronic Kidney Disease using Machine Learning Algorithms," 2020 4th International Conference on Electronics, Communication and Aerospace Technology (ICECA), 2020, pp. 1630-1635, doi: 10.1109/ICECA49313.2020.9297392.

[2] A. Charleonnan, T. Fufaung, T. Niyomwong, W. Chokchueypattanakit, S. Suwannawach and N. Ninchawee, "Predictive analytics for chronic kidney disease using machine learning techniques," 2016 Management and Innovation Technology International Conference (MITicon), 2016, pp. MIT-80-MIT-83, doi: 10.1109/MITICON.2016.8025242.

[3] A. J. Aljaaf et al., "Early Prediction of Chronic Kidney Disease Using Machine Learning Supported by Predictive Analytics," 2018 IEEE Congress on Evolutionary Computation (CEC), 2018, pp. 1-9, doi: 10.1109/CEC.2018.8477876.

[4] R. A. Alassaf et al., "Pre-emptive Diagnosis of Chronic Kidney Disease Using Machine Learning Techniques," 2018 International Conference on Innovations in Information Technology (IIT), 2018, pp. 99-104, doi: 10.1109/INNOVATIONS.2018.8606040.

[5] A. Sobrinho, A. C. M. D. S. Queiroz, L. Dias Da Silva, E. De Barros Costa, M. Eliete Pinheiro and A. Perkusich, "Computer-Aided Diagnosis of Chronic Kidney Disease in Developing Countries: A Comparative Analysis of Machine Learning Techniques," in IEEE Access, vol. 8, pp. 25407-25419, 2020, doi: 10.1109/ACCESS.2020.2971208.

[6] Y. Amirgaliyev, S. Shamiluulu and A. Serek, "Analysis of Chronic Kidney Disease Dataset by Applying Machine Learning Methods," 2018 IEEE 12th International Conference on Application of Information and Communication Technologies (AICT), 2018, pp. 1-4, doi: 10.1109/ICAICT.2018.8747140.

[7] J. Qin, L. Chen, Y. Liu, C. Liu, C. Feng and B. Chen, "A Machine Learning Methodology for Diagnosing Chronic Kidney Disease," in IEEE Access, vol. 8, pp. 20991-21002, 2020, doi: 10.1109/ACCESS.2019.2963053.

[8] M. U. Emon, A. M. Imran, R. Islam, M. S. Keya, R. Zannat and Ohidujjaman, "Performance Analysis of Chronic Kidney Disease through Machine Learning Approaches," 2021 6th International Conference on Inventive Computation Technologies (ICICT), 2021, pp. 713-719, doi: 10.1109/ICICT50816.2021.9358491.

[9] T. M. Rahman, S. Siddiqua, S. E. Rabby, N. Hasan and M. H. Imam, "Early Detection of Kidney Disease Using ECG Signals Through Machine Learning Based Modelling," 2019 International Conference on Robotics,Electrical and Signal Processing Techniques (ICREST), 2019, pp. 319-323, doi: 10.1109/ICREST.2019.8644354.

[10] W. H. S. D. Gunarathne, K. D. M. Perera and K. A. D. C. P. Kahandawaarachchi, "Performance Evaluation on Machine Learning Classification Techniques for Disease Classification and Forecasting through Data Analytics for Chronic Kidney Disease (CKD)," 2017 IEEE 17th International Conference on Bioinformatics and Bioengineering (BIBE), 2017, pp. 291-296, doi: 10.1109/BIBE.2017.00-39.

**2.3 Problem Statement**

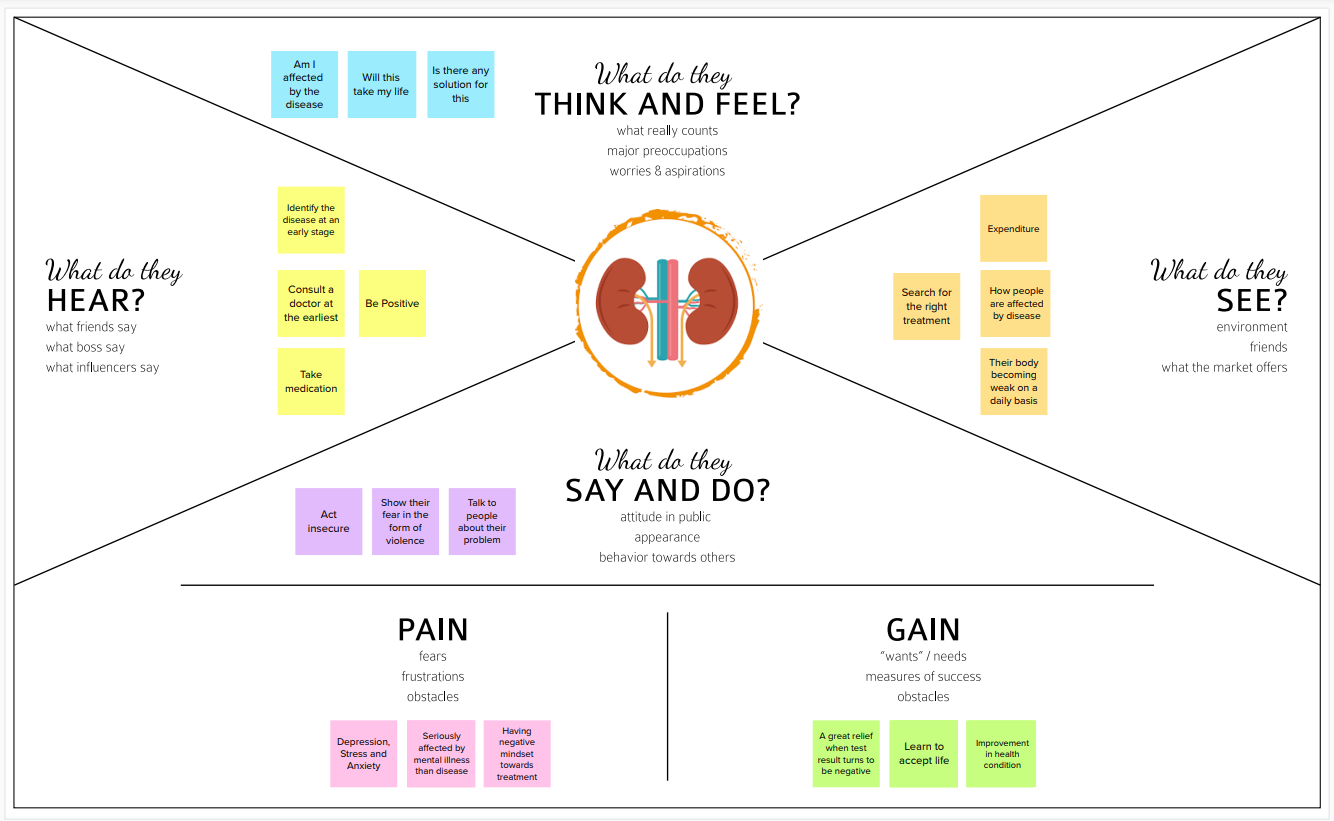
Chronic Kidney Disease (CKD) is a dangerous illness that, if detected in time, can be cured. Most people aren't aware that the many medical tests we have for different reasons could tell us crucial things about kidney diseases. In order to determine whether aspects of a variety of medical tests might contain information that is relevant to the disease, they are examined. The research suggests that doing so allows us to determine the severity of the issue, and we use this knowledge to develop a machine learning model that predicts chronic kidney disease. Early treatment of chronic kidney disease may result in a cure. With the aid of diagnostic data such as Blood Pressure (Bp), Albumin, and other factors, the main objective of this study is to more rapidly and reliably determine whether a patient has chronic kidney disease (Al).

**CHAPTER 3**

**IDEATION & PROPOSED SOLUTION**

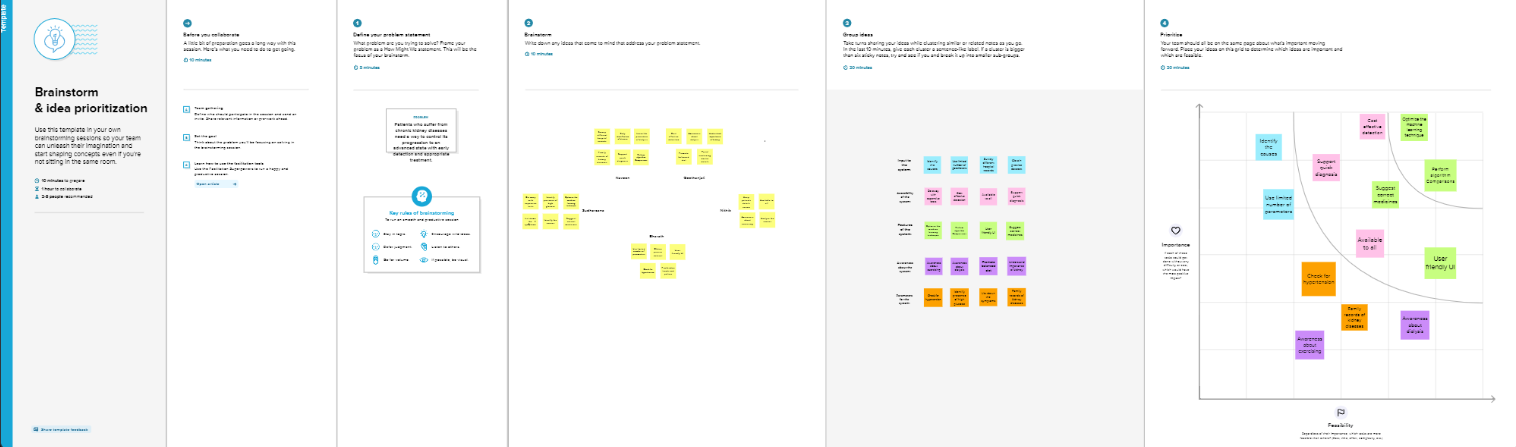
**3.1 Empathy map**

Empathy maps are a helpful tool used by designers to not only evaluate user behaviour but also to visually communicate their findings to colleagues, uniting the team around a shared understanding of the user. Empathy maps work best when utilised early in the design process in user-centered design.



**3.2 Ideation & Brain Storming**

Ideation is a general term that refers to the process of coming up with and expressing new ideas. It is an imaginative thought that seeks to resolve a dilemma or offer a more effective means of carrying out an action. It includes creating fresh concepts, upgrading existing ones, and figuring out how to put fresh concepts into action. Brainstorming is the most frequently practiced form of ideation. The intention of brainstorming is to leverage the collective thinking of the group, by engaging with each other, listening, and building on other ideas.

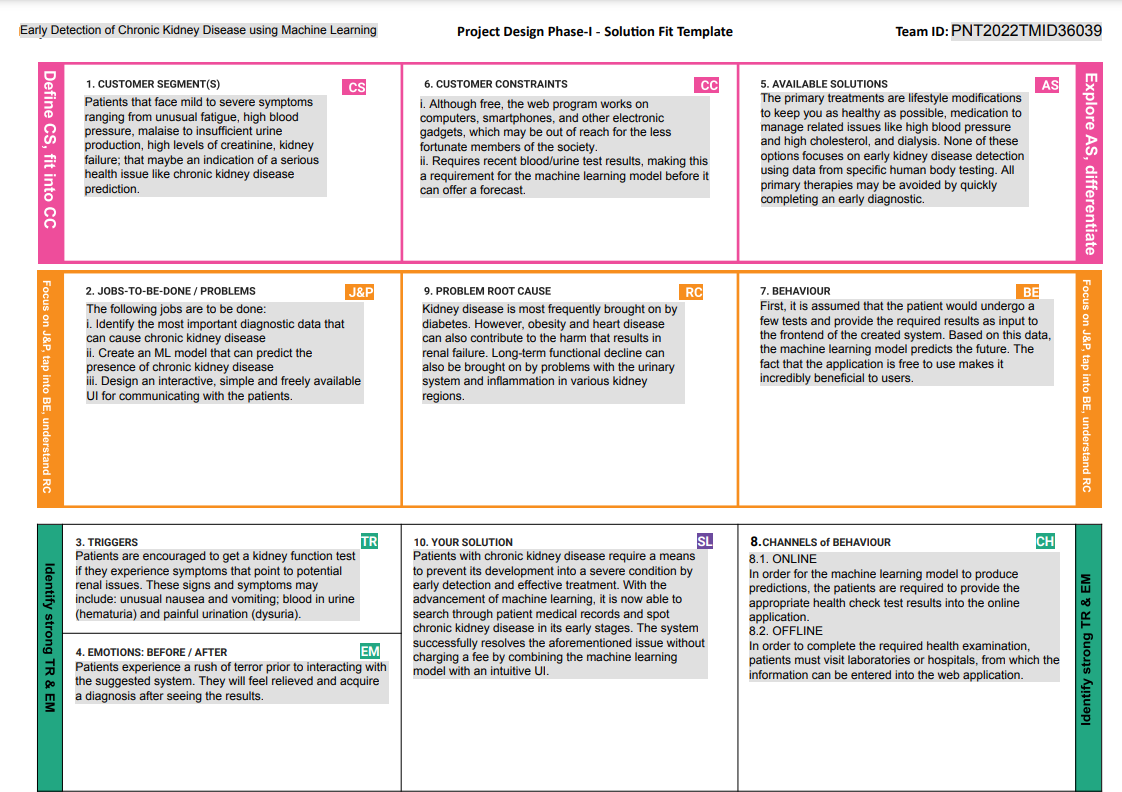


**3.3 Proposed Solution**

|  |  |  |
| --- | --- | --- |
| **S.No** | **Parameter** | **Description** |
| 1 | Problem Statement (Problem to be solved) | Patients who suffer from chronic kidney diseases need a way to control its progression to an advanced state with early detection and appropriate treatment. Machine learning has advanced to the point that it is now possible to look through patient medical information and identify chronic kidney disease in its early stages. |
| 2 | Idea / Solution description | Since certain data are missing, the initial step is to perform pre-processing by cleaning the dataset, along with scaling and normalisation of values. The next step is to use dimensionality reduction to identify the key features in the dataset and to remove any irrelevant ones. To accomplish early detection of chronic kidney disease utilising the indicated key traits, a decision tree model must be fitted. |
| 3 | Novelty / Uniqueness | * An indicator of how well the kidneys is working is the amount of a waste product called creatinine in the blood. By examining this data, early kidney disease can be identified by detecting deviations from the norm. * In the case of healthcare management products, it is especially important to have a UI that is very user-friendly and open to everyone. |
| 4 | Social Impact / Customer Satisfaction | The primary goal of this application is early prediction, and appropriate treatments may be able to prevent or delay the disease's progression to an advanced state. |
| 5 | Business Model (Revenue Model) | * The suggested strategy has the potential to generate income from direct patients as payment for the development of immediate outcomes. * It can also collaborate with the healthcare sector to generate revenue from patients who come in for kidney disease diagnosis. |
| 6 | Scalability of the Solution | * The dimensionality reduction process can be adjusted to produce precise predictions with an increase in the features taken into account. * The accuracy of many models can be compared in order to determine which is best. * It can be used for a variety of illnesses in addition to chronic disorders. |

**3.4 Problem Solution Fit**

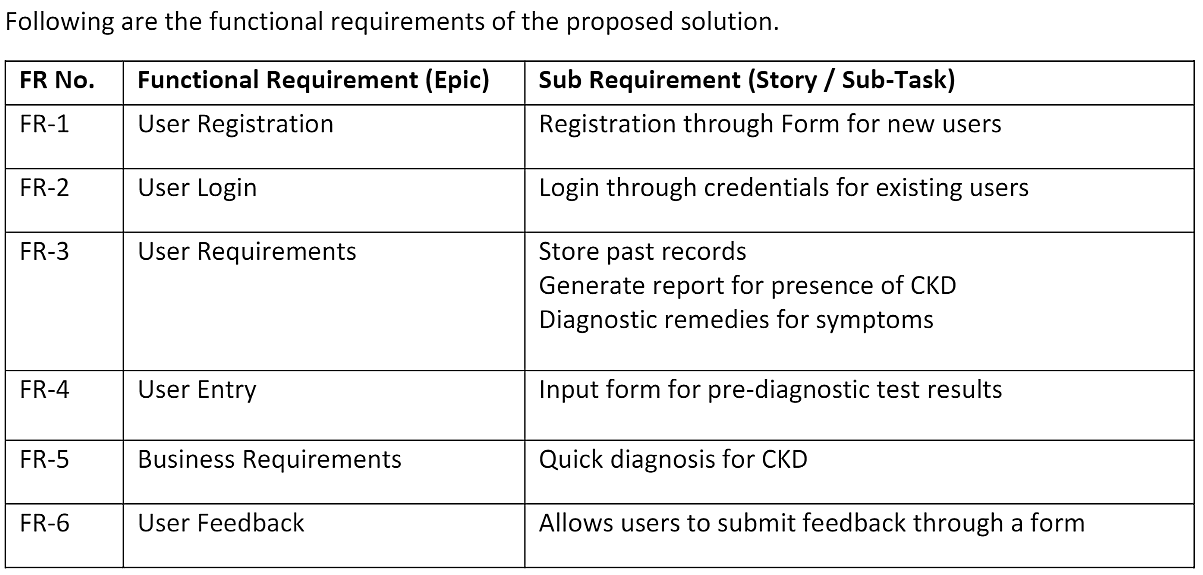
Problem-Solution Fit happens when there is proof that customers are interested in particular tasks, challenges, and benefits. You've established that a problem exists and created a value offer that takes into account the tasks, challenges, and gains of your clients at this point. A problem-solution-fit occurs when such a solution is discovered and a business develops a strategy that, from a variety of angles, offers a game changer for customers. However, if businesses miss evaluating the Problem-Solution Fit they developed, they face a risk of finding that no one wants their solution, which is unfortunate considering the effort and money invested.



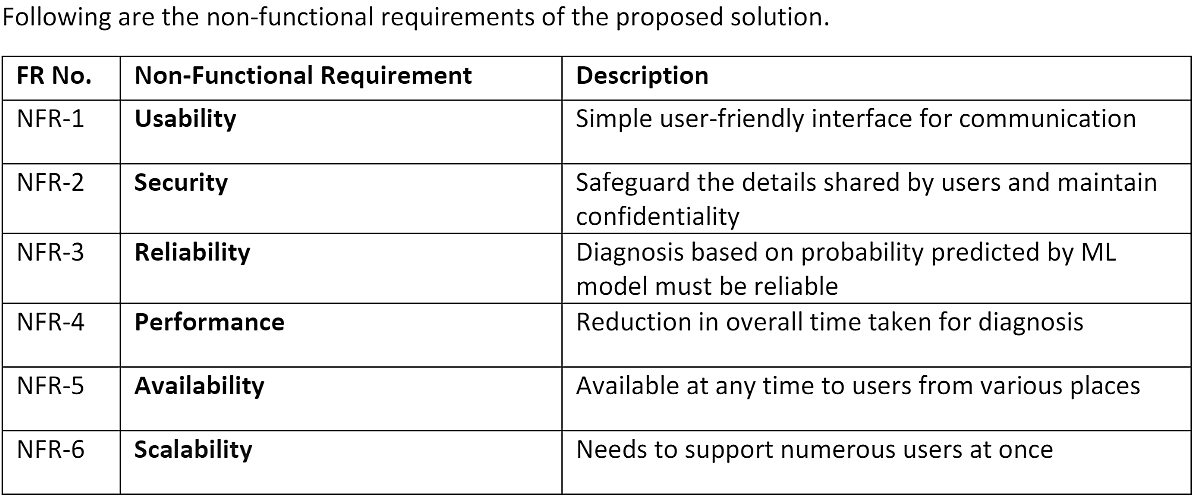
**CHAPTER 4**

**REQUIREMENT ANALYSIS**

**4.1 Functional Requirements**



**4.2 Non Functional Requirements**

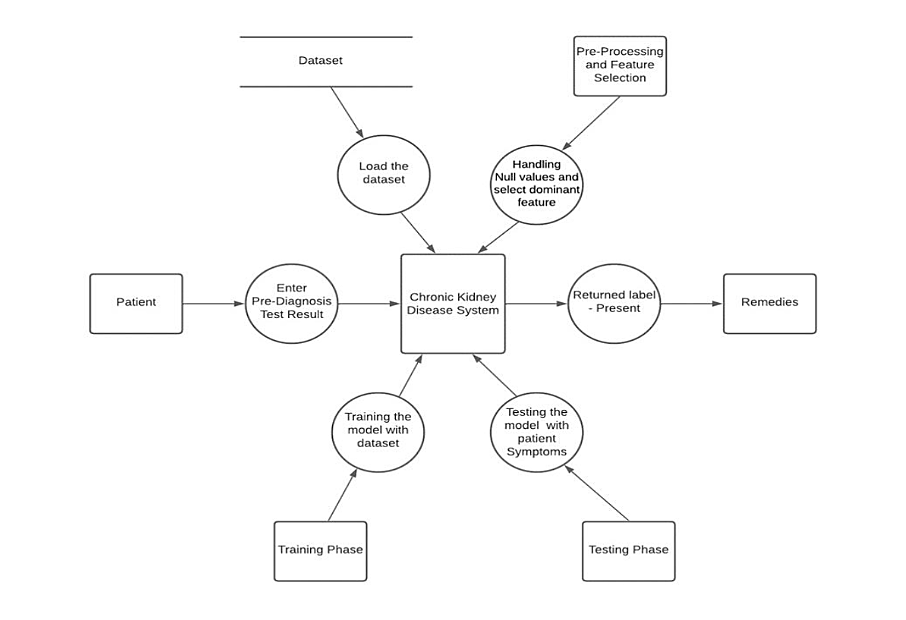


**CHAPTER 5**

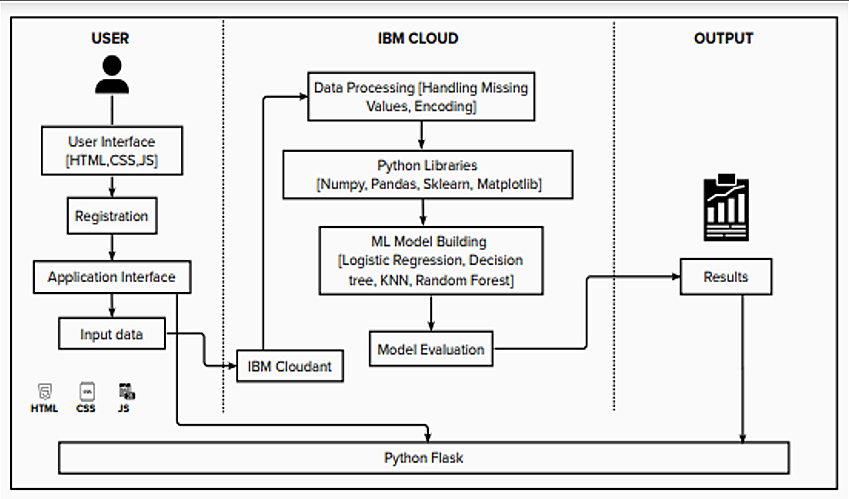
**PROJECT DESIGN**

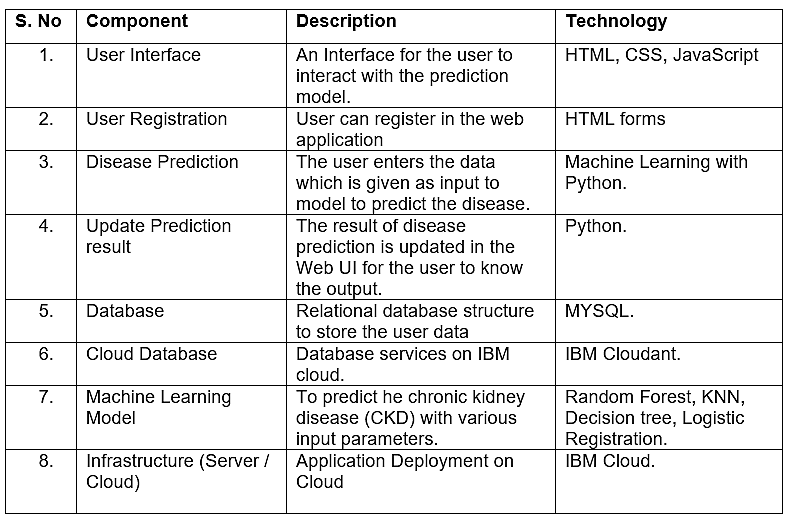
**5.1 Data Flow Diagram**

A data flow diagram is a visualization tool used to illustrate the flow of processes in a company or a specific project within it. It highlights the movement of information as well as the sequence of steps or events required to complete a work task.

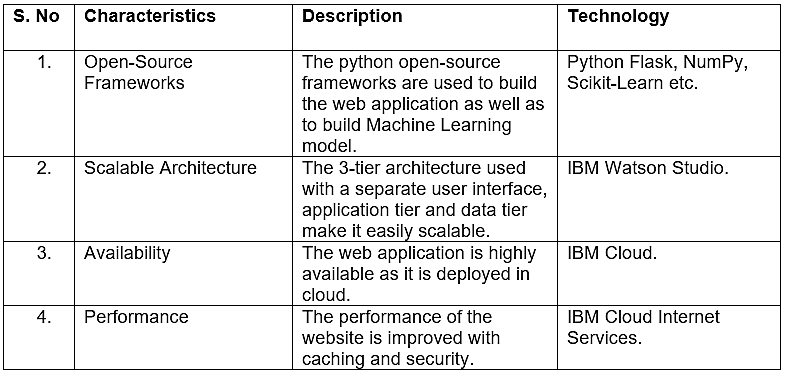


**5.2 Solution & Technical Architecture**

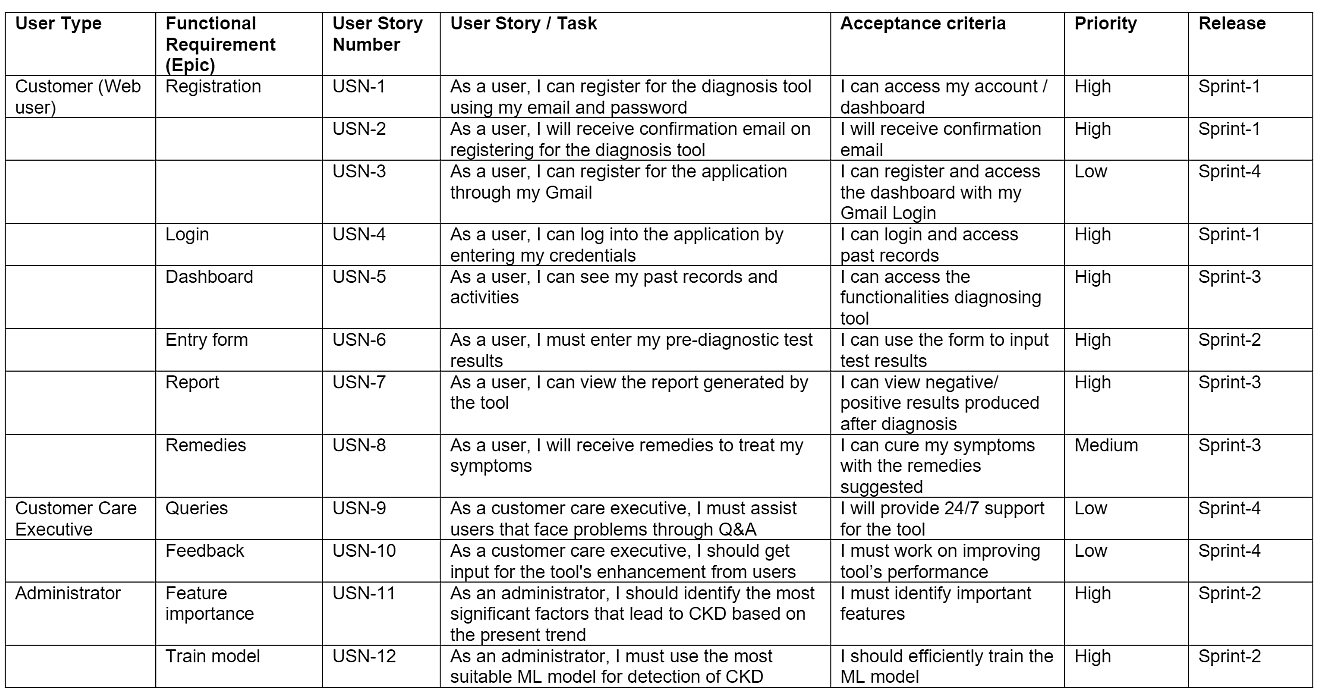




**Application Characteristics**



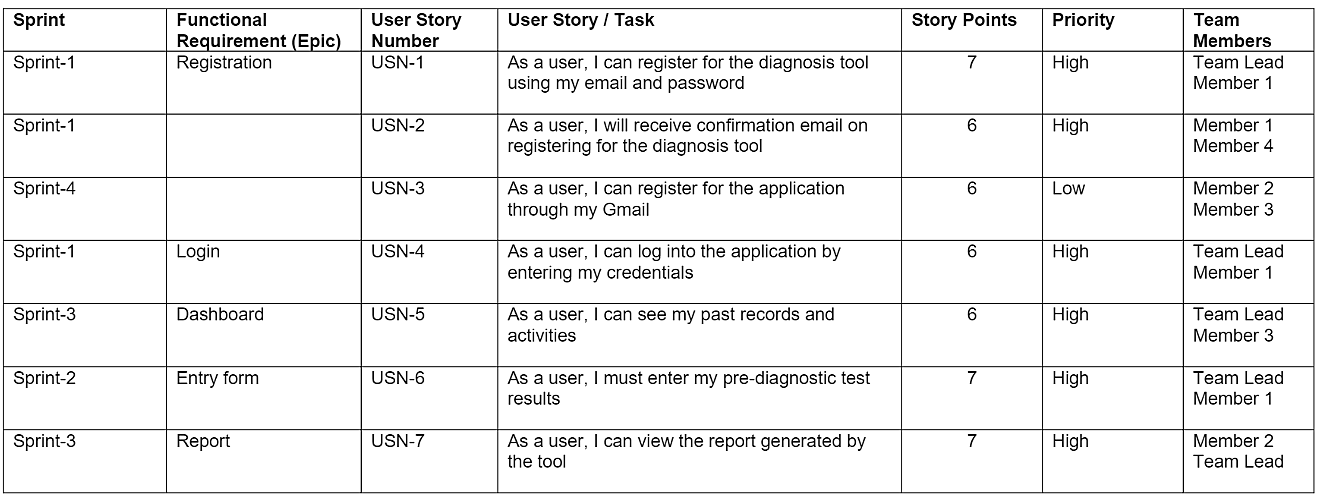
**5.3 User Stories**

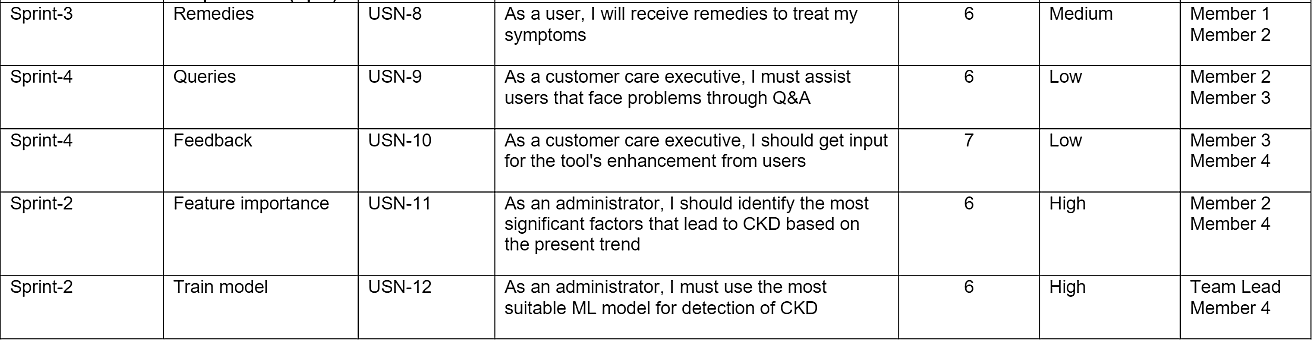


**CHAPTER 6**

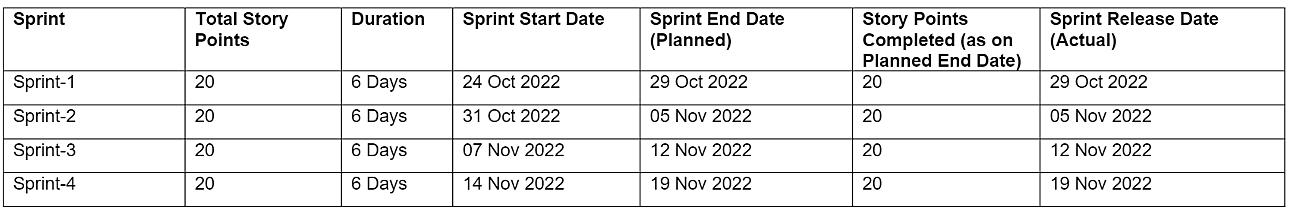
**PROJECT PLANNING AND SCHEDULING**

**6.1 Sprint Planning and Estimation**

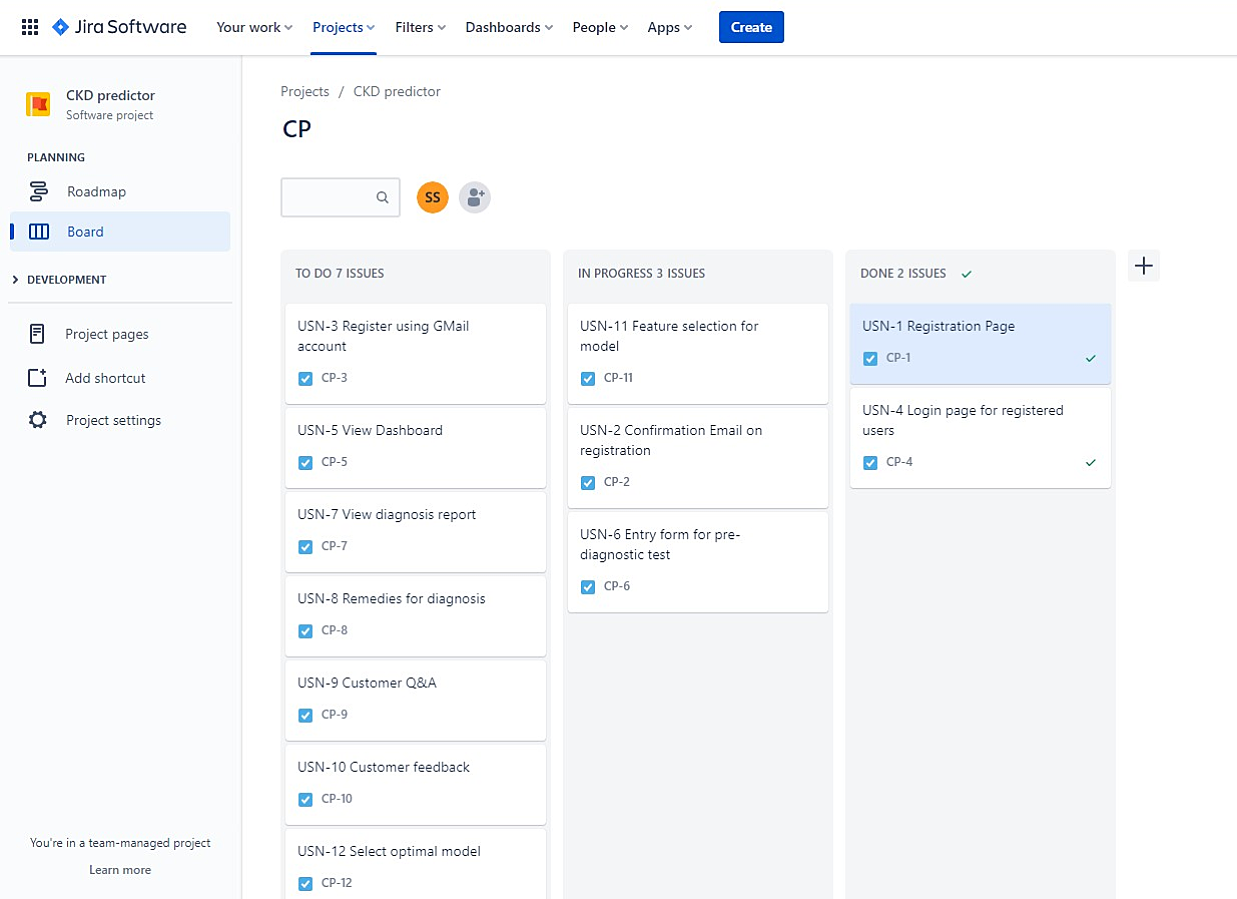




**6.2 Sprint Delivery Schedule**



**6.3 Reports From JIRA**

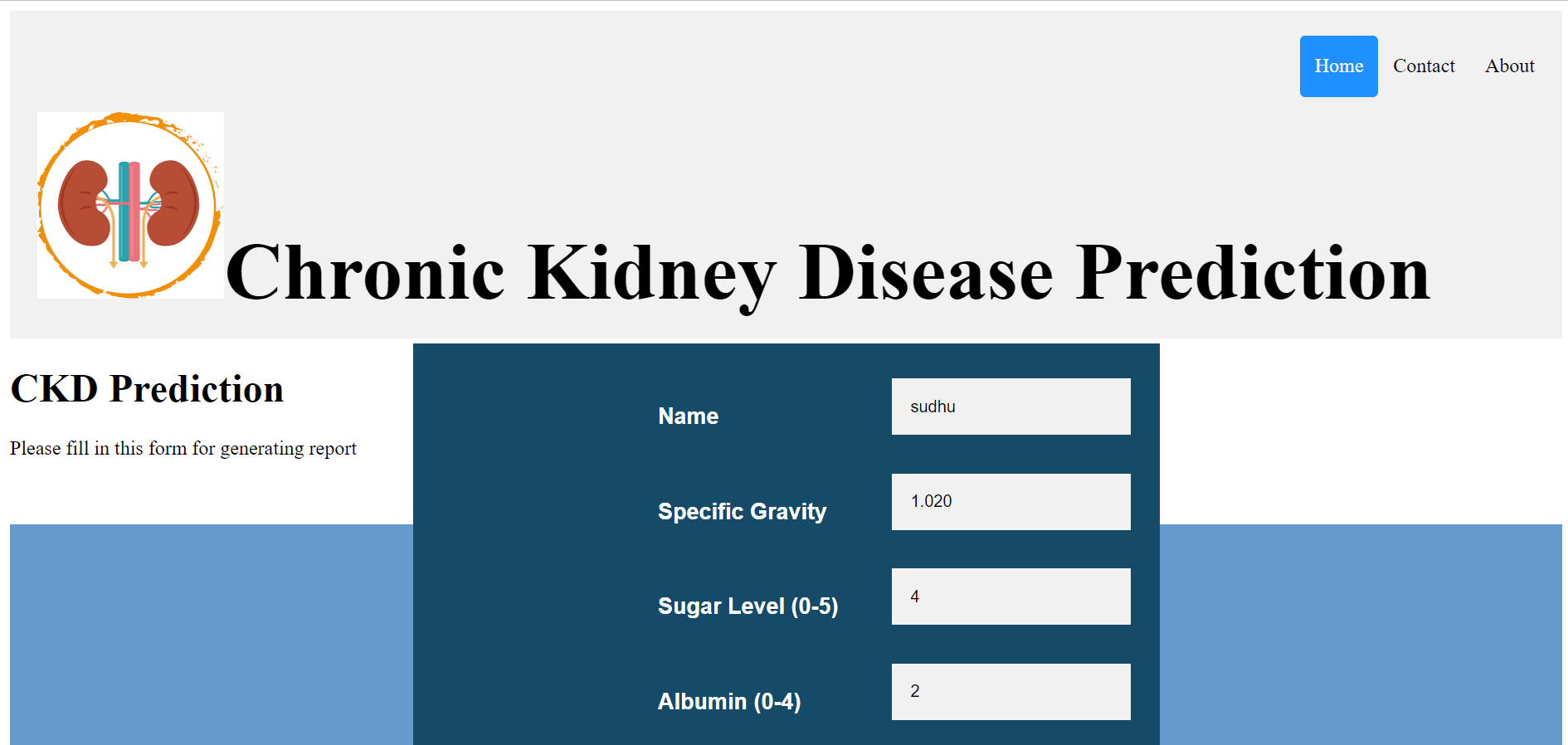


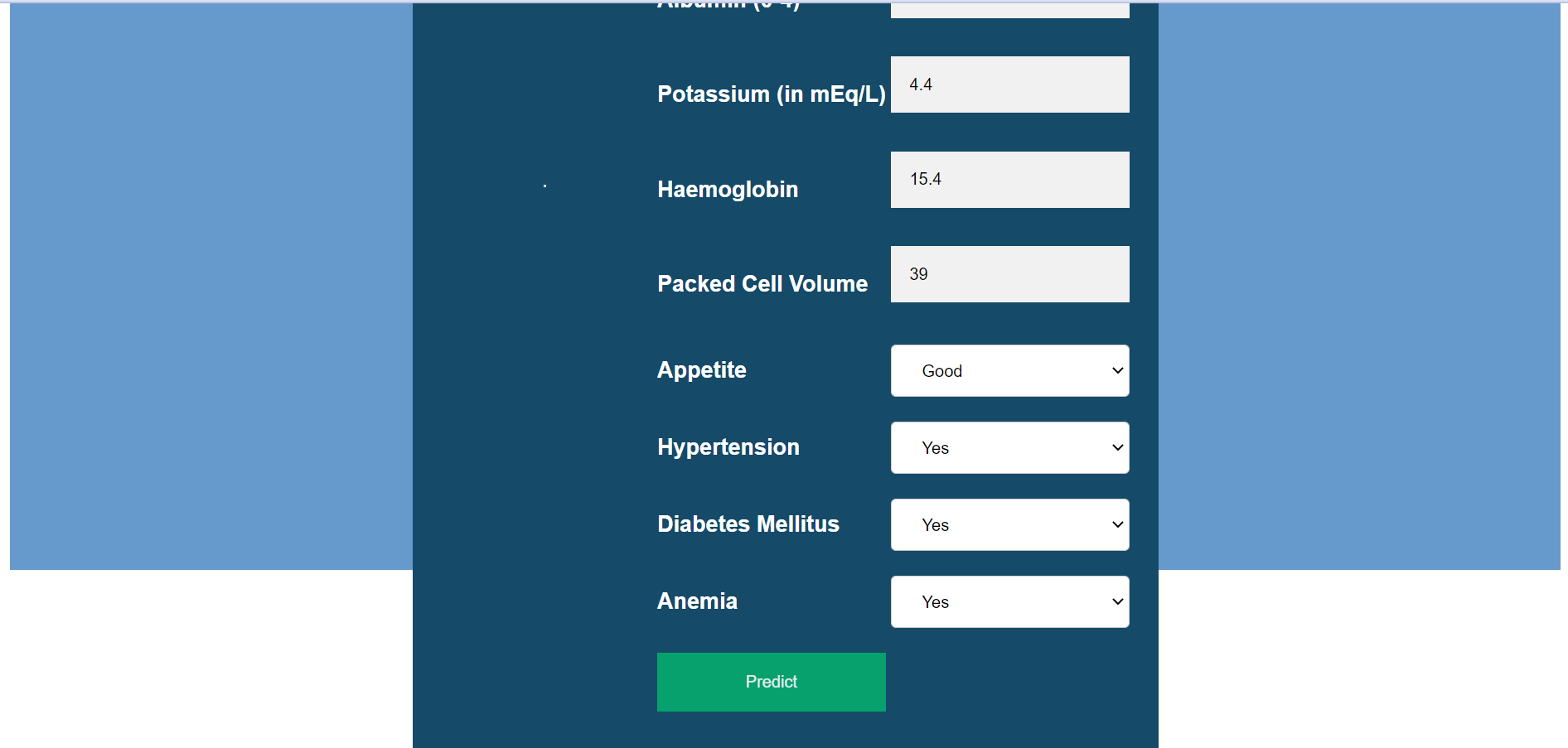
**CHAPTER 7**

**CODING AND SOLUTIONING**

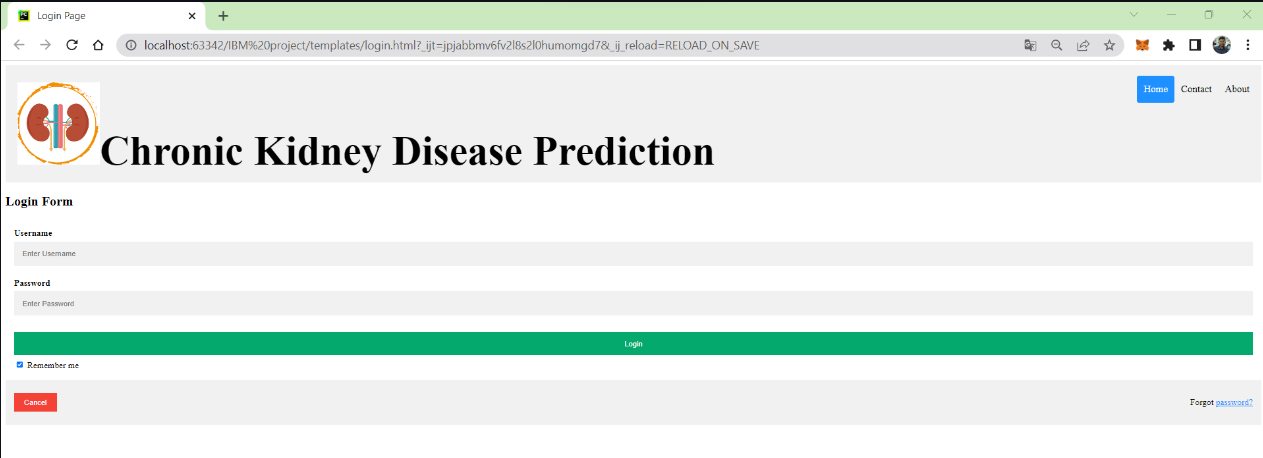
**7.1 Feature 1: Pre-diagnostic Form**

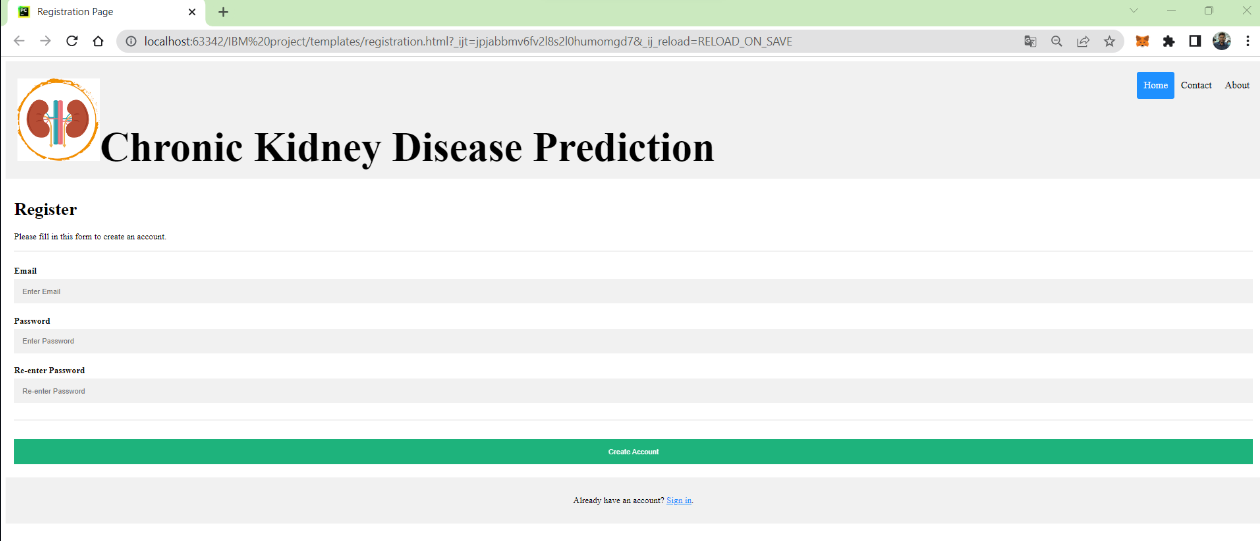
The user can enter the results of the pre-diagnostic tests that have previously been performed on this page. This pre-diagnosis form contains the top ten characteristics determined by feature selection using Extra Tree Classifier on the dataset including 24 different features. The model predicts the likelihood of chronic kidney disease based on the values input.





**7.1.1 Login & Registration Form**





**7.1.2 Methodologies**

**7.1.2.1 Dataset Information**

The dataset contains 26 features and 401 records. The data columns are of the type float,int and String. The features that are present in the data set include:

id

age

bp

sg

al

su

rbc

pc

pcc

ba

bgr

bu

sc

sod

pot

hemo

pcv

wc

rc

htn

dm

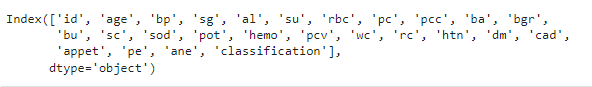
cad

appet

pe

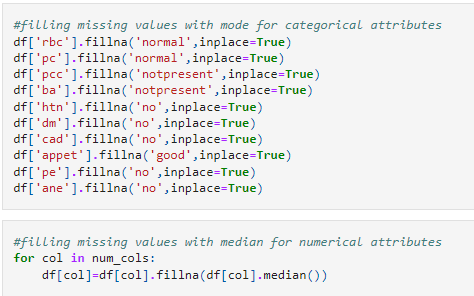
ane

classification



**7.1.2.2 Data Cleaning**

The selected dataset was examined. By collecting the dataset's feature summary, it was possible to comprehend the various data kinds that were there. Floating values were specifically converted from numerical columns. The most commonly present value, or mode, was used to fill in both the numerical and category columns with missing data. The category columns were then subjected to label encoding, which made it easier for the machine learning model to function.



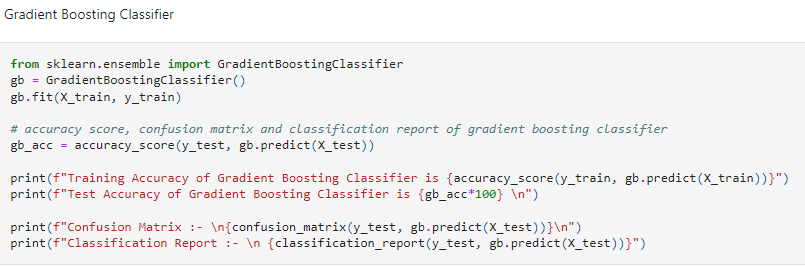


The dataset is divided into 80:20 ratio where 80% is for training data and 20% for testing data.



**7.1.2.3 Model Building**

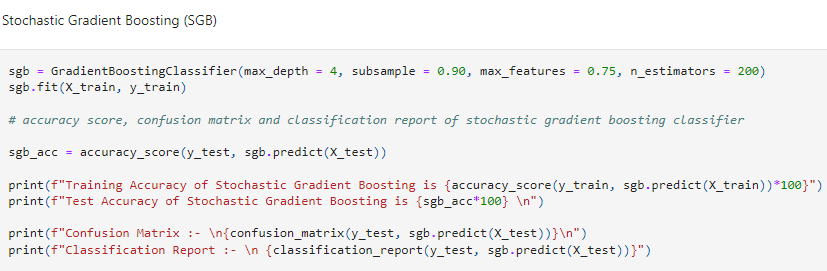
**Gradient Boosting Classifier**



**Accuracy**



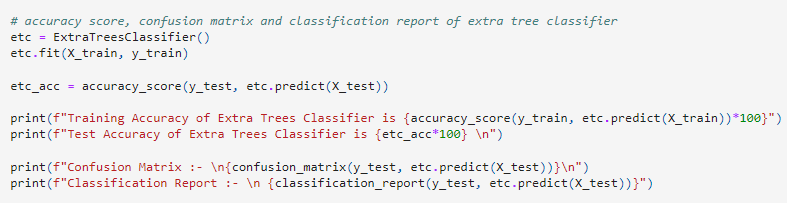
**Stochastic Gradient Boosting**



**Accuracy**



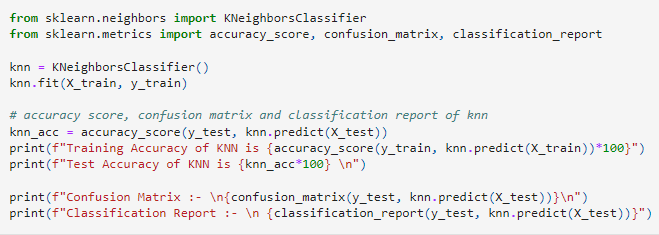
**Extra Tree Classifier**



**Accuracy**



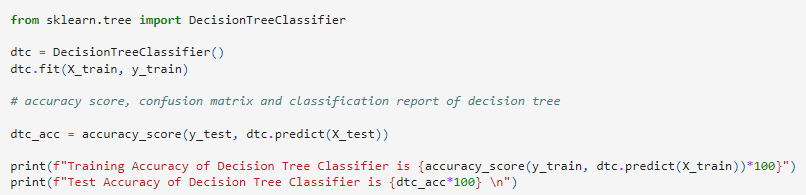
**K-Nearest Neighbour**



**Accuracy**



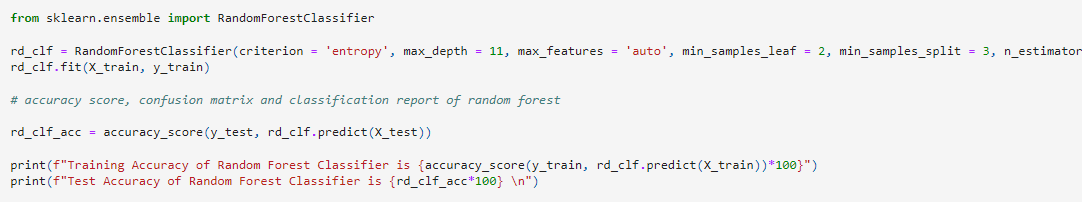
**Decision Tree Classifier**



**Accuracy**



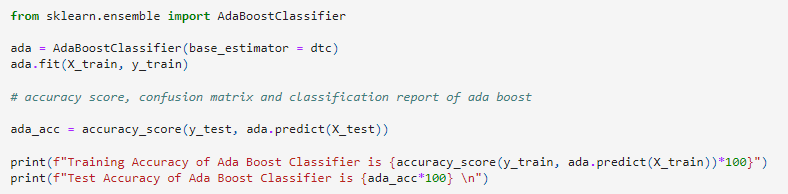
**Random Forest Classifier**



**Accuracy**



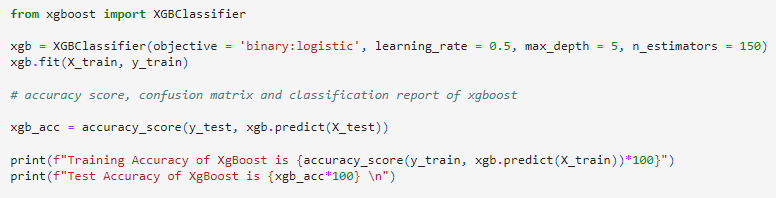
**Ada Boost Classifier**



**Accuracy**



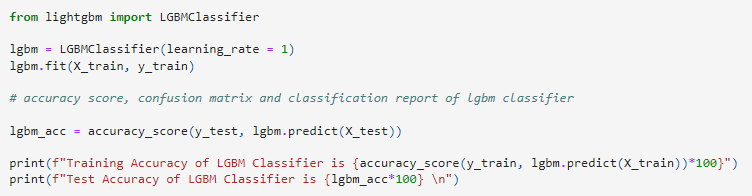
**XGBoost**



**Accuracy**



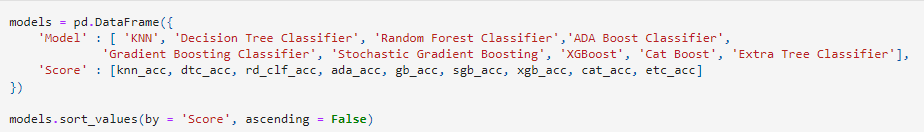
**LGBM Classifiers**

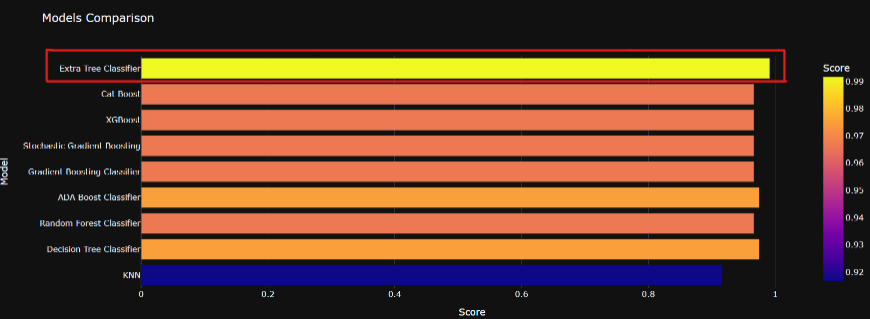


**Accuracy**



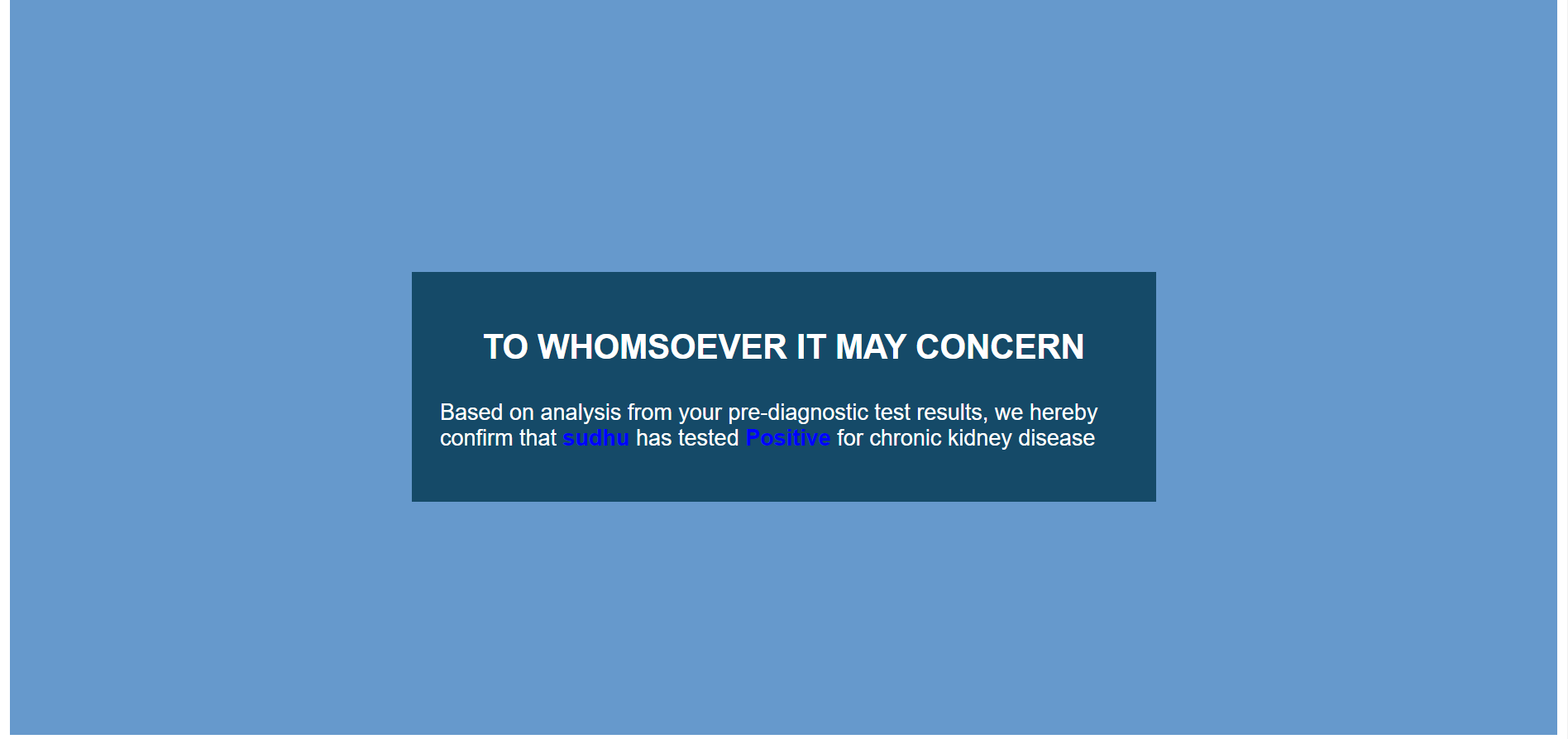
**Model Comparison**





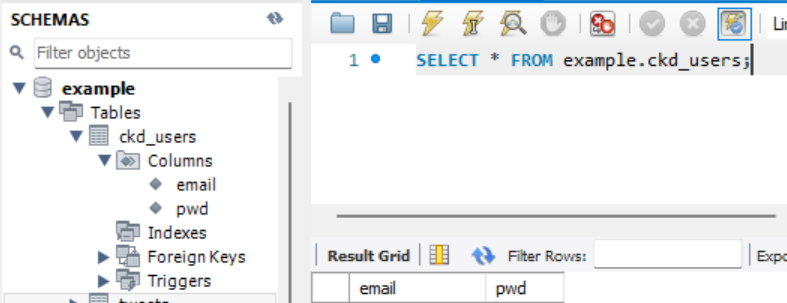
**7.2 Feature 2: Report Generation**

The forecast outcome is shown on this page. The classifier model receives the user-inputted values from the previous page as input. The user is shown the forecast result in the form of a report.



**7.3 Database Schema**

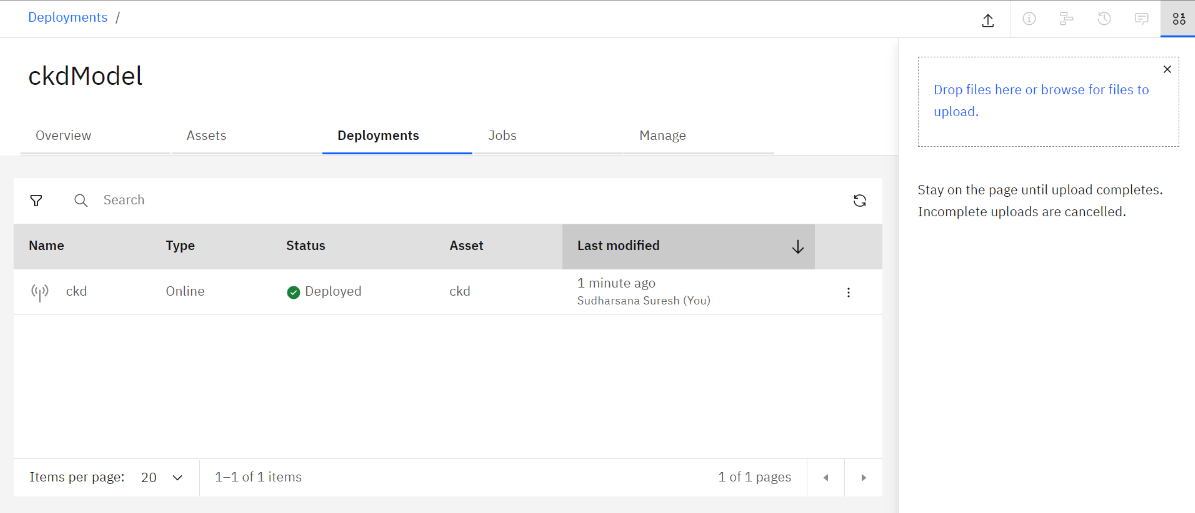
The inputs from the website's login page are stored in a database that is made using MySQL. It creates a table called "ckd users" with the columns "email" and "password" in it. When a new user logs in, they can create an account and have their information saved in the database.



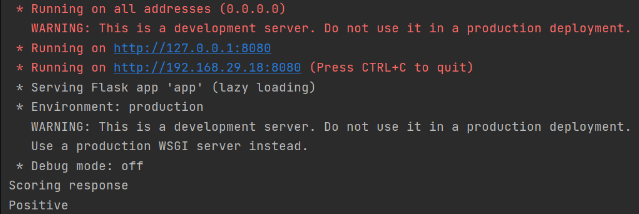
**CHAPTER 8**

**RESULTS**

**8.1 Deployment in IBM Cloud**

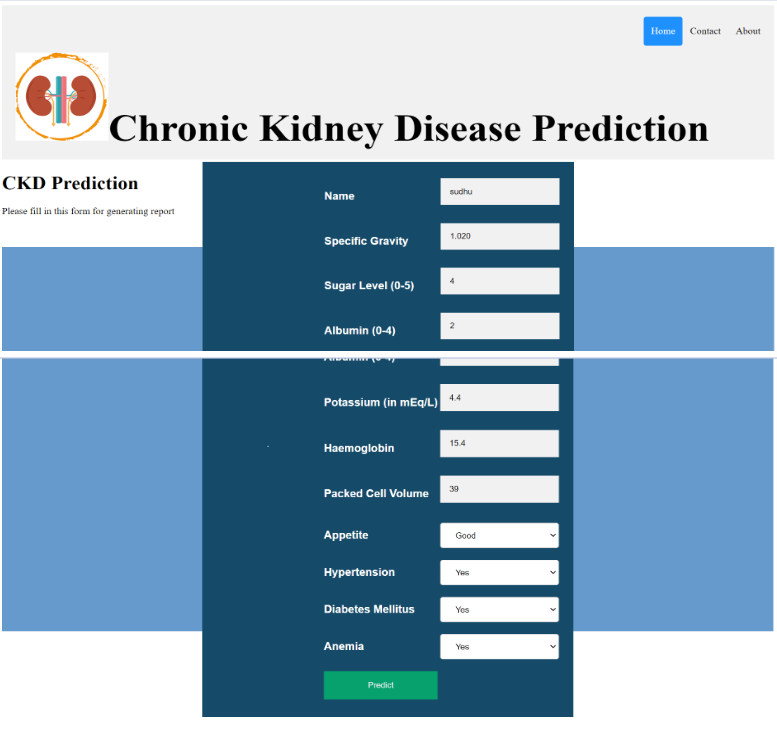


**Integrating Flask with Scoring EndPoint**

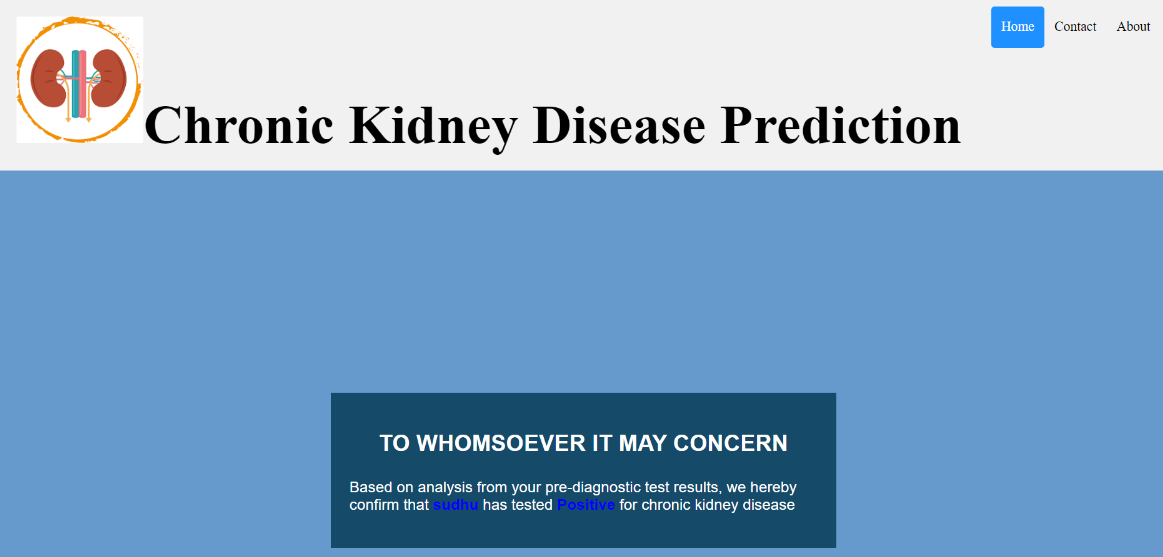


**8.2 Prediction Results**

**8.2.1 Data Entry Form**



**8.2.2 Output**



**8.3 Performance Metrics**

|  |  |
| --- | --- |
| **Model** | **Accuracy** |
| Gradient Boosting Classifier | Training Accuracy: 1.0  Testing Accuracy: 96.66 |
| Stochastic Gradient Boosting | Training Accuracy: 100.0  Testing Accuracy: 96.66 |
| Extra Tree Classifier | Training Accuracy: 100.0  Testing Accuracy: 99.16 |
| K Neighest Neighbour | Training Accuracy: 94.64  Testing Accuracy: 91.66 |
| Decision Tree Classifier | Training Accuracy: 96.78  Testing Accuracy: 97.5 |
| Random Forest Classifier | Training Accuracy: 99.28  Testing Accuracy: 96.66 |
| Ada Boost Classifier | Training Accuracy: 100.0  Testing Accuracy: 97.5 |
| XGBoost | Training Accuracy: 100.0  Testing Accuracy: 96.66 |
| Cat Boost Classifier | Training Accuracy: 98.92  Testing Accuracy: 96.66 |
| LGBM Classifier | Training Accuracy: 100.0  Testing Accuracy: 96.66 |

**CHAPTER 9**

**CONCLUSION**

The benefit of using this approach is that because the prediction process goes much more quickly, doctors may start treating patients with CKD as soon as possible and classify bigger patient populations in a shorter amount of time. We would want to work with larger datasets in the future or compare the results of this dataset with another dataset that contains the same information because the dataset utilized in this paper only has 400 cases. Additionally, using the proper information, we attempt to determine whether a person with this condition has a higher likelihood of developing chronic risk factors like hypertension, a family history of renal failure, and diabetes in order to reduce the incidence of CKD. For both professionals and patients, making an early prognosis is essential to preventing and delaying the progression of chronic renal disease to kidney failure.

**CHAPTER 10**

**FUTURE SCOPE**

In such chronic disorders, data miming algorithms can be employed to pinpoint crucial indicators of disease suffering. When people are struggling with the cost of doing multiple tests, data prediction can be used as an affordable and precise option. The algorithm assists in choosing a subset of tests that would provide comparable accuracy rather than doing all of them, saving a significant amount of money. Because it is free, Boruta Analysis aids in medical diagnosis, which might be costly otherwise. It expedites and makes the diagnostic more affordable for the patients. The problems are stopped from getting worse by early detection. It can be used for a variety of ailments as well as countless other things and is not only restricted to treating chronic illnesses.

**CHAPTER 11**

**APPENDIX**

**11.1. Source code**

**login.html**

<!DOCTYPE html>

<html>

<head>

<meta charset="UTF-8">

<title>Login Page</title>

<style><!DOCTYPE html>

<html>

<head>

<meta charset="UTF-8">

<title>Login Page</title>

<style>

input[type=text], input[type=password] {

width: 100%;

padding: 12px 20px;

margin: 8px 0;

display: inline-block;

border: 1px solid #ccc;

box-sizing: border-box;

}

button {

background-color: #04AA6D;

color: white;

padding: 14px 20px;

margin: 8px 0;

border: none;

cursor: pointer;

width: 100%;

}

button:hover {

opacity: 0.8;

}

.cancelbtn {

width: auto;

padding: 10px 18px;

background-color: #f44336;

}

.imgcontainer {

text-align: center;

margin: 24px 0 12px 0;

}

img.avatar {

width: 40%;

border-radius: 50%;

}

.container {

padding: 16px;

}

span.psw {

float: right;

padding-top: 16px;

}

@media screen and (max-width: 300px) {

span.psw {

display: block;

float: none;

}

.cancelbtn {

width: 100%;

}

}

.header {

overflow: hidden;

background-color: #f1f1f1;

padding: 20px 10px;

}

.header a {

float: left;

color: black;

text-align: center;

padding: 12px;

text-decoration: none;

font-size: 18px;

line-height: 25px;

border-radius: 4px;

}

.header a.logo {

font-size: 75px;

font-weight: bold;

align: center;

}

.header a:hover {

background-color: #ddd;

color: black;

}

.header a.active {

background-color: dodgerblue;

color: white;

}

.header-right {

float: right;

}

@media screen and (max-width: 500px) {

.header a {

float: none;

display: block;

text-align: left;

}

.header-right {

float: none;

}

}

\* {box-sizing: border-box}

.container {

padding: 16px;

}

input[type=text], input[type=password] {

width: 100%;

padding: 15px;

margin: 5px 0 22px 0;

display: inline-block;

border: none;

background: #f1f1f1;

}

input[type=text]:focus, input[type=password]:focus {

background-color: #ddd;

outline: none;

}

hr {

border: 1px solid #f1f1f1;

margin-bottom: 25px;

}

.registerbtn {

background-color: #04AA6D;

color: white;

padding: 16px 20px;

margin: 8px 0;

border: none;

cursor: pointer;

width: 100%;

opacity: 0.9;

}

.registerbtn:hover {

opacity:1;

}

a {

color: dodgerblue;

}

.signin {

background-color: #f1f1f1;

text-align: center;

}

</style>

</head>

<body>

<form action="/action\_page.php" method="post">

<div class="header">

<a href="#default" class="logo"><img src="kidneylogo.png" width="150" height="150">Chronic Kidney Disease Prediction</a>

<div class="header-right">

<a class="active" href="#home">Home</a>

<a href="#contact">Contact</a>

<a href="#about">About</a>

</div>

</div>

<h2>Login Form</h2>

<div class="container">

<label for="uname"><b>Username</b></label>

<input type="text" placeholder="Enter Username" name="uname" required>

<label for="psw"><b>Password</b></label>

<input type="password" placeholder="Enter Password" name="psw" required>

<button type="submit">Login</button>

<label>

<input type="checkbox" checked="checked" name="remember"> Remember me

</label>

</div>

<div class="container" style="background-color:#f1f1f1">

<button type="button" class="cancelbtn">Cancel</button>

<span class="psw">Forgot <a href="#">password?</a></span>

</div>

</form>

</body>

</html>

**format.css**

.imgcontainer {

text-align: center;

margin: 24px 0 12px 0;

}

.header {

overflow: hidden;

background-color: #f1f1f1;

padding: 20px 10px;

}

.header a {

float: left;

color: black;

text-align: center;

padding: 12px;

text-decoration: none;

font-size: 18px;

line-height: 25px;

border-radius: 4px;

}

.header a.logo {

font-size: 75px;

font-weight: bold;

align: center;

}

.header a:hover {

background-color: #ddd;

color: black;

}

.header a.active {

background-color: dodgerblue;

color: white;

}

.header-right {

float: right;

}

@media screen and (max-width: 500px) {

.header a {

float: none;

display: block;

text-align: left;

}

.header-right {

float: none;

}

}

\* {box-sizing: border-box}

.container {

padding: 16px;

}

input[type=text], input[type=password] {

width: 100%;

padding: 15px;

margin: 5px 0 22px 0;

display: inline-block;

border: none;

background: #f1f1f1;

}

input[type=text]:focus, input[type=password]:focus {

background-color: #ddd;

outline: none;

}

hr {

border: 1px solid #f1f1f1;

margin-bottom: 25px;

}

.registerbtn {

background-color: #04AA6D;

color: white;

padding: 16px 20px;

margin: 8px 0;

border: none;

cursor: pointer;

width: 100%;

opacity: 0.9;

}

.registerbtn:hover {

opacity:1;

}

a {

color: dodgerblue;

}

.signin {

background-color: #f1f1f1;

text-align: center;

}

**registration.html**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>Registration Page</title>

</head>

<style>

.imgcontainer {

text-align: center;

margin: 24px 0 12px 0;

}

.header {

overflow: hidden;

background-color: #f1f1f1;

padding: 20px 10px;

}

.header a {

float: left;

color: black;

text-align: center;

padding: 12px;

text-decoration: none;

font-size: 18px;

line-height: 25px;

border-radius: 4px;

}

.header a.logo {

font-size: 75px;

font-weight: bold;

align: center;

}

.header a:hover {

background-color: #ddd;

color: black;

}

.header a.active {

background-color: dodgerblue;

color: white;

}

.header-right {

float: right;

}

@media screen and (max-width: 500px) {

.header a {

float: none;

display: block;

text-align: left;

}

.header-right {

float: none;

}

}

\* {box-sizing: border-box}

.container {

padding: 16px;

}

input[type=text], input[type=password] {

width: 100%;

padding: 15px;

margin: 5px 0 22px 0;

display: inline-block;

border: none;

background: #f1f1f1;

}

input[type=text]:focus, input[type=password]:focus {

background-color: #ddd;

outline: none;

}

hr {

border: 1px solid #f1f1f1;

margin-bottom: 25px;

}

.registerbtn {

background-color: #04AA6D;

color: white;

padding: 16px 20px;

margin: 8px 0;

border: none;

cursor: pointer;

width: 100%;

opacity: 0.9;

}

.registerbtn:hover {

opacity:1;

}

a {

color: dodgerblue;

}

.signin {

background-color: #f1f1f1;

text-align: center;

}

</style>

<body>

<div></div>

<div class="header">

<a href="#default" class="logo"><img src="kidneylogo.png" width="150" height="150">Chronic Kidney Disease Prediction</a>

<div class="header-right">

<a class="active" href="#home">Home</a>

<a href="#contact">Contact</a>

<a href="#about">About</a>

</div>

</div>

<form action="action\_page.php">

<div class="container">

<h1>Register</h1>

<p>Please fill in this form to create an account.</p>

<hr>

<label for="email"><b>Email</b></label>

<input type="text" placeholder="Enter Email" name="email" id="email" required>

<label for="psw"><b>Password</b></label>

<input type="password" placeholder="Enter Password" name="psw" id="psw" required>

<label for="psw-repeat"><b>Re-enter Password</b></label>

<input type="password" placeholder="Re-enter Password" name="psw-repeat" id="psw-repeat" required>

<hr>

<button type="submit" class="registerbtn">Create Account</button>

</div>

<div class="container signin">

<p>Already have an account? <a href="#">Sign in</a>.</p>

</div>

</form>

</body>

</html>

**Data cleaning.ipynb**

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

import warnings

warnings.filterwarnings('ignore')

df=pd.read\_csv(r'/content/drive/MyDrive/Dataset\_med\_rec/kidney\_disease.csv')

df.info()

df['pcv']=df['pcv'].apply(lambda x:x if type(x)==type(3.5) else x.replace('\t43','43').replace('\t?','Nan'))

# cleaning "WC"

df['wc']=df['wc'].apply(lambda x:x if type(x)==type(3.5) else x.replace('\t?','Nan').replace('\t6200','6200').replace('\t8400','8400'))

# cleaning "RC"

df['rc']=df['rc'].apply(lambda x:x if type(x)==type(3.5) else x.replace('\t?','Nan'))

# cleaning "dm"

df['dm']=df['dm'].apply(lambda x:x if type(x)==type(3.5) else x.replace('\tno','no').replace('\tyes','yes').replace(' yes','yes'))

# cleaning "CAD"

df['cad']=df['cad'].apply(lambda x:x if type(x)==type(3.5) else x.replace('\tno','no'))

# cleaning "Classification"

df['classification']=df['classification'].apply(lambda x:x if type(x)==type(3.5) else x.replace('ckd\t','ckd'))

#explicitly converting numerical columns

mistyped=[['pcv','rc','wc']]

for i in mistyped:

df[i]=df[i].astype('float')

#categorical columns

cat\_cols=list(df.select\_dtypes('object'))

cat\_cols

#numerical columns

num\_cols=list(df.select\_dtypes(['int64','float64']))

num\_cols

#handling missing data

df.isnull().sum().sort\_values(ascending=False)

#filling missing values with mode for categorical attributes

df['rbc'].fillna('normal',inplace=True)

df['pc'].fillna('normal',inplace=True)

df['pcc'].fillna('notpresent',inplace=True)

df['ba'].fillna('notpresent',inplace=True)

df['htn'].fillna('no',inplace=True)

df['dm'].fillna('no',inplace=True)

df['cad'].fillna('no',inplace=True)

df['appet'].fillna('good',inplace=True)

df['pe'].fillna('no',inplace=True)

df['ane'].fillna('no',inplace=True)

#filling missing values with median for numerical attributes

for col in num\_cols:

df[col]=df[col].fillna(df[col].median())

df.isna().sum().sort\_values(ascending=False)

df['classification'].unique()

#label encoding for target class

df['classification']=df['classification'].map({'ckd':1,'notckd':0})

#label encoding for categorical attributes

df['rbc']=df['rbc'].map({'normal':0,'abnormal':1})

df['pc']=df['pc'].map({'normal':0,'abnormal':1})

df['pcc']=df['pcc'].map({'notpresent':0,'present':1})

df['ba']=df['ba'].map({'notpresent':0,'present':1})

df['htn']=df['htn'].map({'no':0,'yes':1})

df['dm']=df['dm'].map({'no':0,'yes':1})

df['cad']=df['cad'].map({'no':0,'yes':1})

df['pe']=df['pe'].map({'no':0,'yes':1})

df['ane']=df['ane'].map({'no':0,'yes':1})

df['appet']=df['appet'].map({'good':0,'poor':1})

from sklearn.model\_selection import train\_test\_split

x=df.drop('classification',axis=1)#independent

y=df['classification']#dependent

X\_train,X\_test,y\_train,y\_test=train\_test\_split(x,y,test\_size=0.2,random\_state=0)

print("X\_train size {} , X\_test size {}".format(X\_train.shape,X\_test.shape))

**Model\_Building\_On\_IBM**

import warnings

from decimal import Decimal

import numpy as np

import seaborn as sns

import pandas as pd

from sklearn.ensemble import ExtraTreesClassifier

from sklearn.tree import DecisionTreeClassifier

from sklearn.neighbors import KNeighborsClassifier

from sklearn.svm import SVC

from sklearn.metrics import accuracy\_score,confusion\_matrix,classification\_report

from sklearn.model\_selection import train\_test\_split, cross\_val\_score

from sklearn.model\_selection import GridSearchCV

from sklearn.ensemble import RandomForestClassifier

from xgboost import XGBClassifier

import matplotlib.pyplot as plt

import os, types

from botocore.client import Config

import ibm\_boto3

def \_\_iter\_\_(self): return 0

# @hidden\_cell

# The following code accesses a file in your IBM Cloud Object Storage. It includes your credentials.

# You might want to remove those credentials before you share the notebook.

cos\_client = ibm\_boto3.client(service\_name='s3',

ibm\_api\_key\_id='\*\*\*\*',

ibm\_auth\_endpoint="https://iam.cloud.ibm.com/oidc/token",

config=Config(signature\_version='oauth'),

endpoint\_url='https://s3.private.us.cloud-object-storage.appdomain.cloud')

bucket = 'ckdprediction-donotdelete-pr-kvcxes1swicxqv'

object\_key = 'processed\_kidney.csv'

body = cos\_client.get\_object(Bucket=bucket,Key=object\_key)['Body']

# add missing \_\_iter\_\_ method, so pandas accepts body as file-like object

if not hasattr(body, "\_\_iter\_\_"): body.\_\_iter\_\_ = types.MethodType( \_\_iter\_\_, body )

df = pd.read\_csv(body)

df.head()

# Separating the dependent and independent variables

y = df['classification']

X = df.drop(['classification','id'], axis = 1)

X.head()

# Feature Selection using Extra Tree Classifier

# Building the model

extra\_tree\_forest = ExtraTreesClassifier(n\_estimators = 5,criterion ='entropy', max\_features = 2)

# Training the model

extra\_tree\_forest.fit(X, y)

# Computing the importance of each feature

feature\_importance = extra\_tree\_forest.feature\_importances\_

# Normalizing the individual importances

feature\_importance\_normalized = np.std([tree.feature\_importances\_ for tree in

extra\_tree\_forest.estimators\_],

axis = 0)

# Plotting a Bar Graph to compare the models

plt.figure(figsize=(50,20))

plt.bar(X.columns, feature\_importance\_normalized)

plt.xlabel('Feature Labels')

plt.ylabel('Feature Importances')

plt.title('Comparison of different Feature Importances')

plt.show()

# Identifying top 10 features for pre-diagnosis test

feature\_scores=pd.DataFrame(extra\_tree\_forest.feature\_importances\_,columns=['Score'],index=X.columns).sort\_values(by='Score',ascending=False)

top10\_feature = feature\_scores.nlargest(n=10, columns=['Score'])

plt.figure(figsize=(20,6))

print(top10\_feature.index)

g = sns.barplot(x=top10\_feature.index, y=top10\_feature['Score'])

p = plt.title('Top 10 Features with Extra Tree Classifier')

p = plt.xlabel('Feature name')

p = plt.ylabel('Extra Tree score')

p = g.set\_xticklabels(g.get\_xticklabels(), horizontalalignment='right')

top10\_feature.index

X.columns

X.columns

for ele in X.columns:

if ele not in top10\_feature.index:

X = X.drop(ele, axis = 1)

X.head()

from sklearn.model\_selection import train\_test\_split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 0.30, random\_state = None)

from sklearn.ensemble import GradientBoostingClassifier

gb = GradientBoostingClassifier()

gb.fit(X\_train, y\_train)

# accuracy score, confusion matrix and classification report of gradient boosting classifier

gb\_acc = accuracy\_score(y\_test, gb.predict(X\_test))

print(f"Training Accuracy of Gradient Boosting Classifier is {accuracy\_score(y\_train, gb.predict(X\_train))}")

print(f"Test Accuracy of Gradient Boosting Classifier is {gb\_acc\*100} \n")

print(f"Confusion Matrix :- \n{confusion\_matrix(y\_test, gb.predict(X\_test))}\n")

print(f"Classification Report :- \n {classification\_report(y\_test, gb.predict(X\_test))}")

# Stochastic Gradient Boosting (SGB)

sgb = GradientBoostingClassifier(max\_depth = 4, subsample = 0.90, max\_features = 0.75, n\_estimators = 200)

sgb.fit(X\_train, y\_train)

# accuracy score, confusion matrix and classification report of stochastic gradient boosting classifier

sgb\_acc = accuracy\_score(y\_test, sgb.predict(X\_test))

print(f"Training Accuracy of Stochastic Gradient Boosting is {accuracy\_score(y\_train, sgb.predict(X\_train))\*100}")

print(f"Test Accuracy of Stochastic Gradient Boosting is {sgb\_acc\*100} \n")

print(f"Confusion Matrix :- \n{confusion\_matrix(y\_test, sgb.predict(X\_test))}\n")

print(f"Classification Report :- \n {classification\_report(y\_test, sgb.predict(X\_test))}")

# \*\*Extra Tree Classifier\*\*

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

etc = ExtraTreesClassifier()

etc.fit(X\_train, y\_train)

etc\_acc = accuracy\_score(y\_test, etc.predict(X\_test))

print(f"Training Accuracy of Extra Trees Classifier is {accuracy\_score(y\_train, etc.predict(X\_train))\*100}")

print(f"Test Accuracy of Extra Trees Classifier is {etc\_acc\*100} \n")

print(f"Confusion Matrix :- \n{confusion\_matrix(y\_test, etc.predict(X\_test))}\n")

print(f"Classification Report :- \n {classification\_report(y\_test, etc.predict(X\_test))}")

# \*\*KNN\*\*

from sklearn.neighbors import KNeighborsClassifier

from sklearn.metrics import accuracy\_score, confusion\_matrix, classification\_report

knn = KNeighborsClassifier()

knn.fit(X\_train, y\_train)

# accuracy score, confusion matrix and classification report of knn

knn\_acc = accuracy\_score(y\_test, knn.predict(X\_test))

print(f"Training Accuracy of KNN is {accuracy\_score(y\_train, knn.predict(X\_train))\*100}")

print(f"Test Accuracy of KNN is {knn\_acc\*100} \n")

print(f"Confusion Matrix :- \n{confusion\_matrix(y\_test, knn.predict(X\_test))}\n")

print(f"Classification Report :- \n {classification\_report(y\_test, knn.predict(X\_test))}")

# \*\*Decision Tree Classifier\*\*

from sklearn.tree import DecisionTreeClassifier

dtc = DecisionTreeClassifier()

dtc.fit(X\_train, y\_train)

# accuracy score, confusion matrix and classification report of decision tree

dtc\_acc = accuracy\_score(y\_test, dtc.predict(X\_test))

print(f"Training Accuracy of Decision Tree Classifier is {accuracy\_score(y\_train, dtc.predict(X\_train))\*100}")

print(f"Test Accuracy of Decision Tree Classifier is {dtc\_acc\*100} \n")

print(f"Confusion Matrix :- \n{confusion\_matrix(y\_test, dtc.predict(X\_test))}\n")

print(f"Classification Report :- \n {classification\_report(y\_test, dtc.predict(X\_test))}")

# hyper parameter tuning of decision tree

from sklearn.model\_selection import GridSearchCV

grid\_param = {

'criterion' : ['gini', 'entropy'],

'max\_depth' : [3, 5, 7, 10],

'splitter' : ['best', 'random'],

'min\_samples\_leaf' : [1, 2, 3, 5, 7],

'min\_samples\_split' : [1, 2, 3, 5, 7],

'max\_features' : ['auto', 'sqrt', 'log2']

}

grid\_search\_dtc = GridSearchCV(dtc, grid\_param, cv = 5, n\_jobs = -1, verbose = 1)

grid\_search\_dtc.fit(X\_train, y\_train)

# best estimator

dtc = grid\_search\_dtc.best\_estimator\_

# accuracy score, confusion matrix and classification report of decision tree

dtc\_acc = accuracy\_score(y\_test, dtc.predict(X\_test))

print(f"Training Accuracy of Decision Tree Classifier is {accuracy\_score(y\_train, dtc.predict(X\_train))\*100}")

print(f"Test Accuracy of Decision Tree Classifier is {dtc\_acc\*100} \n")

print(f"Confusion Matrix :- \n{confusion\_matrix(y\_test, dtc.predict(X\_test))}\n")

print(f"Classification Report :- \n {classification\_report(y\_test, dtc.predict(X\_test))}")

# \*\*Random Forest Classifier\*\*

from sklearn.ensemble import RandomForestClassifier

rd\_clf = RandomForestClassifier(criterion = 'entropy', max\_depth = 11, max\_features = 'auto', min\_samples\_leaf = 2, min\_samples\_split = 3, n\_estimators = 130)

rd\_clf.fit(X\_train, y\_train)

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

rd\_clf\_acc = accuracy\_score(y\_test, rd\_clf.predict(X\_test))

print(f"Training Accuracy of Random Forest Classifier is {accuracy\_score(y\_train, rd\_clf.predict(X\_train))\*100}")

print(f"Test Accuracy of Random Forest Classifier is {rd\_clf\_acc\*100} \n")

print(f"Confusion Matrix :- \n{confusion\_matrix(y\_test, rd\_clf.predict(X\_test))}\n")

print(f"Classification Report :- \n {classification\_report(y\_test, rd\_clf.predict(X\_test))}")

# ADA Boost Classifier

from sklearn.ensemble import AdaBoostClassifier

ada = AdaBoostClassifier(base\_estimator = dtc)

ada.fit(X\_train, y\_train)

# accuracy score, confusion matrix and classification report of ada boost

ada\_acc = accuracy\_score(y\_test, ada.predict(X\_test))

print(f"Training Accuracy of Ada Boost Classifier is {accuracy\_score(y\_train, ada.predict(X\_train))\*100}")

print(f"Test Accuracy of Ada Boost Classifier is {ada\_acc\*100} \n")

print(f"Confusion Matrix :- \n{confusion\_matrix(y\_test, ada.predict(X\_test))}\n")

print(f"Classification Report :- \n {classification\_report(y\_test, ada.predict(X\_test))}")

# XGBoost

from xgboost import XGBClassifier

xgb = XGBClassifier(objective = 'binary:logistic', learning\_rate = 0.5, max\_depth = 5, n\_estimators = 150)

xgb.fit(X\_train, y\_train)

# accuracy score, confusion matrix and classification report of xgboost

xgb\_acc = accuracy\_score(y\_test, xgb.predict(X\_test))

print(f"Training Accuracy of XgBoost is {accuracy\_score(y\_train, xgb.predict(X\_train))\*100}")

print(f"Test Accuracy of XgBoost is {xgb\_acc\*100} \n")

print(f"Confusion Matrix :- \n{confusion\_matrix(y\_test, xgb.predict(X\_test))}\n")

print(f"Classification Report :- \n {classification\_report(y\_test, xgb.predict(X\_test))}")

get\_ipython().system('pip install catboost')

# \*\*Cat Boost Classifier\*\*

from catboost import CatBoostClassifier

cat = CatBoostClassifier(iterations=10)

cat.fit(X\_train, y\_train)

cat\_acc = accuracy\_score(y\_test, cat.predict(X\_test))

print(f"Training Accuracy of Cat Boost Classifier is {accuracy\_score(y\_train, cat.predict(X\_train))\*100}")

print(f"Test Accuracy of Cat Boost Classifier is {cat\_acc\*100} \n")

print(f"Confusion Matrix :- \n{confusion\_matrix(y\_test, cat.predict(X\_test))}\n")

print(f"Classification Report :- \n {classification\_report(y\_test, cat.predict(X\_test))}")

# \*\*LGBM Classifier\*\*

from lightgbm import LGBMClassifier

lgbm = LGBMClassifier(learning\_rate = 1)

lgbm.fit(X\_train, y\_train)

# accuracy score, confusion matrix and classification report of lgbm classifier

lgbm\_acc = accuracy\_score(y\_test, lgbm.predict(X\_test))

print(f"Training Accuracy of LGBM Classifier is {accuracy\_score(y\_train, lgbm.predict(X\_train))\*100}")

print(f"Test Accuracy of LGBM Classifier is {lgbm\_acc\*100} \n")

print(f"{confusion\_matrix(y\_test, lgbm.predict(X\_test))}\n")

print(classification\_report(y\_test, lgbm.predict(X\_test)))

# \*\*Models Comparison\*\*

models = pd.DataFrame({

'Model' : [ 'KNN', 'Decision Tree Classifier', 'Random Forest Classifier','ADA Boost Classifier',

'Gradient Boosting Classifier', 'Stochastic Gradient Boosting', 'XGBoost', 'Cat Boost', 'Extra Tree Classifier'],

'Score' : [knn\_acc, dtc\_acc, rd\_clf\_acc, ada\_acc, gb\_acc, sgb\_acc, xgb\_acc, cat\_acc, etc\_acc]

})

models.sort\_values(by = 'Score', ascending = False)

import plotly.express as px

px.bar(data\_frame = models, x = 'Score', y = 'Model', color = 'Score', template = 'plotly\_dark',

title = 'Models Comparison')

get\_ipython().system('pip install ibm\_watson\_machine\_learning')

from ibm\_watson\_machine\_learning import APIClient

wml\_credentials = {

"url" : "https://us-south.ml.cloud.ibm.com",

"apikey" : "\*\*\*\*"

}

client = APIClient(wml\_credentials)

def guid\_from\_space\_name(client, space\_name):

space = client.spaces.get\_details()

return(next(item for item in space['resources'] if item['entity']["name"] == space\_name)['metadata']['id'])

space\_uid = guid\_from\_space\_name(client, 'ckdModel')

print("sSpace UID = "+space\_uid)

client.set.default\_space(space\_uid)

client.software\_specifications.list(limit=200)

software\_spec\_uid = client.software\_specifications.get\_uid\_by\_name("runtime-22.1-py3.9")

software\_spec\_uid

model\_details = client.repository.store\_model(model = etc, meta\_props = {

client.repository.ModelMetaNames.NAME: "ckd",

client.repository.ModelMetaNames.TYPE:"scikit-learn\_1.0",

client.repository.ModelMetaNames.SOFTWARE\_SPEC\_UID: software\_spec\_uid

}

)

model\_id = client.repository.get\_model\_uid(model\_details)

model\_id

**Integerating flask with end score**

**app.py**

from flask import Flask,render\_template,request

#import pickle

import requests

# NOTE: you must manually set API\_KEY below using information retrieved from your IBM Cloud account.

API\_KEY = "v6zXMCaTXBv8fJlMVEucGxo5uWRpDfXWGxqBpPn2P\_Xy"

token\_response = requests.post('https://iam.cloud.ibm.com/identity/token', data={"apikey":

API\_KEY, "grant\_type": 'urn:ibm:params:oauth:grant-type:apikey'})

mltoken = token\_response.json()["access\_token"]

header = {'Content-Type': 'application/json', 'Authorization': 'Bearer ' + mltoken}

#model=pickle.load(open('CKDmodel.pkl','rb'))

app=Flask(\_\_name\_\_)

@app.route('/')

def index():

return render\_template('index.html')

@app.route('/predict',methods=['POST'])

def predict\_satisfaction():

v1=float(request.form.get('sg'))

v2=int(request.form.get('su'))

v3=int(request.form.get('ab'))

v4=float(request.form.get('pot'))

v5=float(request.form.get('hemo'))

v6=int(request.form.get('pcv'))

v7=request.form.get('ap')

v7 = 0 if 'Good' else 1

v8=request.form.get('ht')

v8 = 0 if 'No' else 1

v9=request.form.get('dm')

v9=0 if 'No' else 1

v10=request.form.get('ae')

v10=0 if 'No' else 1

name=request.form.get('name')

total = [[v1,v3,v2,v4,v5,v6,v8,v9,v7,v10]]

# NOTE: manually define and pass the array(s) of values to be scored in the next line

payload\_scoring = {"input\_data": [{"field": [["sg", "ab","su", "pot", "hemo", "pcv", "ht", "dm","ap", "ae"]],

"values": total}]}

response\_scoring = requests.post(

'https://us-south.ml.cloud.ibm.com/ml/v4/deployments/2abfa46a-b72d-4d9c-8165-8a20f83abe0d/predictions?version=2022-11-20',

json=payload\_scoring,

headers={'Authorization': 'Bearer ' + mltoken})

print("Scoring response")

d = {0: "Negative", 1: "Positive"}

pred=response\_scoring.json()

res=pred['predictions'][0]['values'][0][0]

print(d[int(res)])

return render\_template('report.html', res=d[int(res)], pname=name)

'''

#prediction

result=model.predict(np.array([v1,v2,v3,v4,v5,v6,v7,v8,v9,v10]).reshape(1,10))

d={0:"Negative",1:"Positive"}

prediction\_result=d[result[0]]

return render\_template('report.html',res=prediction\_result,pname=name)

'''

if \_\_name\_\_=='\_\_main\_\_':

app.run(host='0.0.0.0',port=8080)

**11.2. Github link:**

https://github.com/IBM-EPBL/IBM-Project-2050-1658424611

**11.3. Demo link:**

https://drive.google.com/file/d/1X6udxq0Kya8h-PT1DCwZH4DR5AAhILbj/view?usp=sharing