CHAPTER 1

INTRODUCTION

1.1 PROJECT OVERVIEW

Chronic kidney disease, also called chronic kidney failure, involves a gradual loss of kidney function. Your kidneys filter wastes and excess fluids from your blood, which are then removed in your urine. Advanced chronic kidney disease can cause dangerous levels of fluid, electrolytes and wastes to build up in your body. In the early stages of chronic kidney disease, you might have few signs or symptoms. You might not realize that you have kidney disease until the condition is advanced. Treatment for chronic kidney disease focuses on slowing the progression of kidney damage, usually by controlling the cause. But even controlling the cause might not keep kidney damage from progressing. Chronic kidney disease can progress to end-stage kidney failure, which is fatal without artificial filtering (dialysis) or a kidney transplant. Without leading to vital stage of Kidney disease, Machine Learning algorithms can be used to predict the earlier detection. Early detection and cure of chronic kidney disease (CKD) is extremely desirable as it can lead to the prevention of unwanted consequences. CKD is often diagnosed in later stages when dialysis or kidney transplant are the only options left to save the patient's life. Whereas an early diagnosis can lead to the prevention of kidney failure. The best way to measure the kidney function or to predict the stages of kidney disease is to monitor the Glomerular Filtration Rate (GFR) on regular basis.

1.2 PURPOSE

A combination of estimated glomerular filtration rate (GFR), age, diet, existing medical conditions, and albuminuria can be used to assess the severity of kidney disease but requires more accurate information about the risk to the kidney is required to make clinical decisions about diagnosis, treatment, and referral. The purpose of this model is to develop and validate predictive models for chronic kidney disease. The main goal will be to evaluate kidney failure, which means the need for kidney dialysis or kidney transplant first. These models also teach the patient how to live a healthy life and help the doctor see the risk and severity of the disease, as well as how to proceed with the treatment in the future. It may be possible to identify patterns of data collection using ANN, mining methods, and the future occurrence of certain diseases that may cause harm can be predicted in advance. The purpose of the proposed model is to predict whether the patient will suffer or develop chronic kidney disease in the future if he continues their lifestyle. This information can be used to determine whether the kidney disease is using eGFR (glomerular filtration rate), which helps the doctor plan the appropriate treatment.

Estimated glomerular filtration rate (eGFR) defines the degree of kidney disease and measures kidney function.

The main function of the kidney is to filter the blood in the body. Kidney disease is a silent killer because it can cause kidney failure without causing any symptoms or concern. Chronic kidney disease is defined as a decline in kidney function over a period of months or years. Kidney disease is often caused by diabetes and high blood pressure. Chronic kidney disease is a major health problem that affects people worldwide. Not getting the right treatment for chronic kidney disease can have serious consequences, affecting people who cannot afford it. Glomerular filtration rate (GFR) is the most accurate test to determine your kidney function and the degree of chronic kidney disease. Blood creatinine level, age, gender, and other characteristics can be used to calculate it. In most cases it is better to get sick early. Therefore, it is possible to prevent serious illness. Chronic Kidney Disease is detected and analyzed with the help of Random Forest Algorithm. Random Forest is an algorithm that is used for supervised classification. It creates a forest of many trees to calculate the accuracy efficiently. The accuracy for this classifier is directly proportional to the number of trees. The results produced by Random Forest, even without hyper-parameter tuning, are more reliable because of its flexibility. It is simple and works very efficiently especially when the size of the data set is large. It retains the accuracy rate by recognizing outliers and anomalies. However, it is not direct to implement and is computationally expensive.

CHAPTER 2

LITERATURE SURVEY

2.1 EXISTING PROBLEM

[1] Chronic Kidney Disease Prediction Using Machine Learning Techniques.

This paper deals with the prediction of CKD in people. A wrapper method used here for feature selection is ACO. ACO is a meta-heuristic optimization algorithm. Out of the 24 attributes present 12 best attributes are taken for prediction. Prediction is done using the machine learning technique, SVM. In this classification problem SVM classifies the output into two class with CKD and without CKD. The main objective of this study was to predict patients with CKD using less number attributes while maintaining a higher accuracy. Here we obtain an accuracy of about 96 percentage.

[2] Comparative Analysis for Prediction of Kidney Diseases using ML models.

The proposed system utilizes the CKD prediction dataset. After pre-processing and feature selection, the DT, KNN, and logistic regression algorithms have been used. J. Hussain and the team have obtained an accuracy of 0.995 in predicting CKD in early stages using multilayer perception while including pre-processing of data set with neural networks to fill the missing values. The workflow includes discarding the outliers, selecting the optimal seven attributes with statistical analysis, and discarding the attributes which have a higher inter co-relation by principal component analysis (PCA).

[3] Prior Stage Kidney Disease Prediction Using AI & Supervised Machine Learning Techniques.

In this a system is proposed that uses various data mining techniques like KNN, DT, NB, and SB classifiers. The metrics provided below gives us information on the quality of the outcomes that we get in this study. A confusion matrix helps us with this by describing the performance of the classifier. This paper deals with the prediction of CKD in people. A wrapper method used here. SBC is a meta heuristic algorithm. Out of the 24 attributes present 12 best attributes are taken for prediction. Prediction is done using the machine learning technique, SBC. In this classification problem SBC classifies the output into two class with CKD and without CKD.

[4] Comparison and development of machine learning tools in the prediction of chronic kidney disease progression.

The clinical and blood biochemical results from 551 patients with proteinuria were collected. Thirteen blood-derived tests and 5 demographic features were used as non-urinary clinical variables to predict the 24-h urinary protein outcome response. Nine predictive models were established and compared, including logistic regression, Elastic Net, lasso regression, ridge regression, support vector machine, random forest, XG Boost, neural network and k-nearest neighbour. The AU-ROC, sensitivity (recall), specificity, accuracy, log-loss and precision of each of the models were evaluated. The effect sizes of each variable were analysed and ranked. The linear models including Elastic Net, lasso regression, ridge regression and logistic regression showed the highest overall predictive power, with an average AUC and a precision above 0.87 and 0.8, respectively. Logistic regression ranked first, reaching an AUC of 0.873, with a sensitivity and specificity of 0.83 and 0.82, respectively.

[5] Prediction of Chronic Kidney Disease Using Machine Learning Algorithm.

The proposed system makes a comparison of the performance of the different classification methods and ensemble algorithms that are used for detection of chronic kidney disease. This study involves six different basic classifiers namely: k nearest neighbour (KNN), naive bayes, support vector machines (SVM), random trees, J48, decision tables and three different ensemble algorithms namely: bagging, AdaBoost, random subspace are used in the study. Classification results were derived using three different performance evaluation criteria (kappa,accuracy and the area under the ROC curve (AUC)). The result says that J48 basis algorithm for use with random subspace and bagging ensemble algorithms and random tree basis algorithm for use with bagging ensemble algorithm has provided 100% classification success. Data mining is vastly being investigated in diagnostic results. Huge amount of un-mined data derived from healthcare industry is used to discover sensible information in order to diagnose and help in decision making. Data mining helps in extracting hidden information from huge dataset. It also helps in categorizing data, validate them and derive unique patterns in them. Several datamining techniques like classification, clustering, regression, association analysis,etc are used in healthcare data mining.The below table contains the list of articles that includes the existing problem.

S.No	YEAR	AUTHOR NAME	TITLE	ALGORITHM	DRAWBACKS
01	2022	Saurabh Pal	Chronic Kidney Disease Prediction Using Machine Learning.	Decision Tree, GFR, SVM, Machine Learning	It has been a challenging task to discover the correct set of attributes.
02	2021	Vineeta Gulatiand Neeraj Raheja	Comparative Analysis for prediction of kidney diseases using Intelligent Machine Learning.	K-Nearest Neighbors, Logistic Regression, Decision Tree method Random Forest and, Naïve Bayes, Support Vector Machine and Multi-Layer Perceptron Algorithm	Sometimes setting a biasing values and activation function may consume more time for processing the inputs.
03	2021	Barot mitisha1, prof. Barkha bhavsa	Comparative Analysis for Prediction of Kidney Disease Using Intelligent Machine Learning.	KNN, DT, NB, and SBclassifiers	Individual F1-scores are 95% for non-CKD and 97% for CKD.In this, there lies a problem in the selection of best suitable attributes.
04	2019	Jing Xiao and Ruifeng Ding et.al	Comparison and development of machine learning tools in the prediction of chronic kidney disease.	Logistic regression, k Nearest Neighbors(KNN) regression, SVC, Gaussian NB, decision tree classifier, Random Forest classifier.	This requires the huge amount of dataset collection. Those data should be pre-processed by an efficient method.
05	2018	Siddheshwar Tekale	Prediction of Chronic Kidney Disease Using Machine Learning	Logistic regression, Elastic Net, lasso regression, ridge regression, support vector machine, random forest, XGBoost, neura	It has the problem of conducting the performance analysis criteria as there is need for specifying a optimum criteria value.

Table 2.1 Literature Survey

2.2 REFERENCES

1. Saurabh Pal - Chronic Kidney Disease Prediction Using Machine Learning Methods.Issue:16,August 2022
2. Vineeta Gulatiand Neeraj Raheja - Comparative Analysis for prediction of kidney Diseases using Intelligent Machine Learning Models.Issue:2021
3. Barot mitishal - Prior Stage Kidney Disease Prediction Using AI & Supervised Machine Learning.Issue:12,Dec 2021
4. Jing Xiao and Ruifeng Ding et.al - Comparison and development of Machine learning tools in the prediction of chronic kidney disease progression.Issue:2019
5. Siddheshwar Tekale,Pranjal Shingavi et.al - Prediction of Chronic Kidney Disease Using Machine Learning Algorithm.Issue:10.Oct 2018

2.3 PROBLEM STATEMENT DEFINITION

The kidney is one of the most important body organs that filtrates all the wastes and water from human body to make urine. Chronic Kidney Disease (CKD), also commonly known as chronic renal disease or chronic kidney failure, is a life-threatening disease that is attributed to the failure of the kidney in performing its routine functionality. It leads to the continuous decrease of Glomerular Filtration Rate (GFR) for a period of 3 months or more and is a universal health problem. Some common symptoms of the disease include hypertension, irregular foamy urine, vomiting, shortness of breath, itching and cramps, whereas high blood pressure and diabetes are the main cause of this disorder's is often diagnosed in later stages when dialysis or kidney transplant are the only options left to save the patient's life.

I am	The person to predict the Chronic Kidney Disease using Machine Learning Techniques.
I am trying to	Use the recent technologies to predict the human kidney disease.
But	I am unaware of the existing technology that can help me a lot to predict the disease and I don't know to use the current technology.
Because	I don't want to waste the cost and time.
Which makes me feel	I want a best accuracy which can predict the disease so that the people can move with their necessary treatments.

Table 2.2-Problem Statement Definition

CHAPTER 3

IDEATION AND PROPOSED SOLUTION

3.1 EMPATHY MAP CANVAS

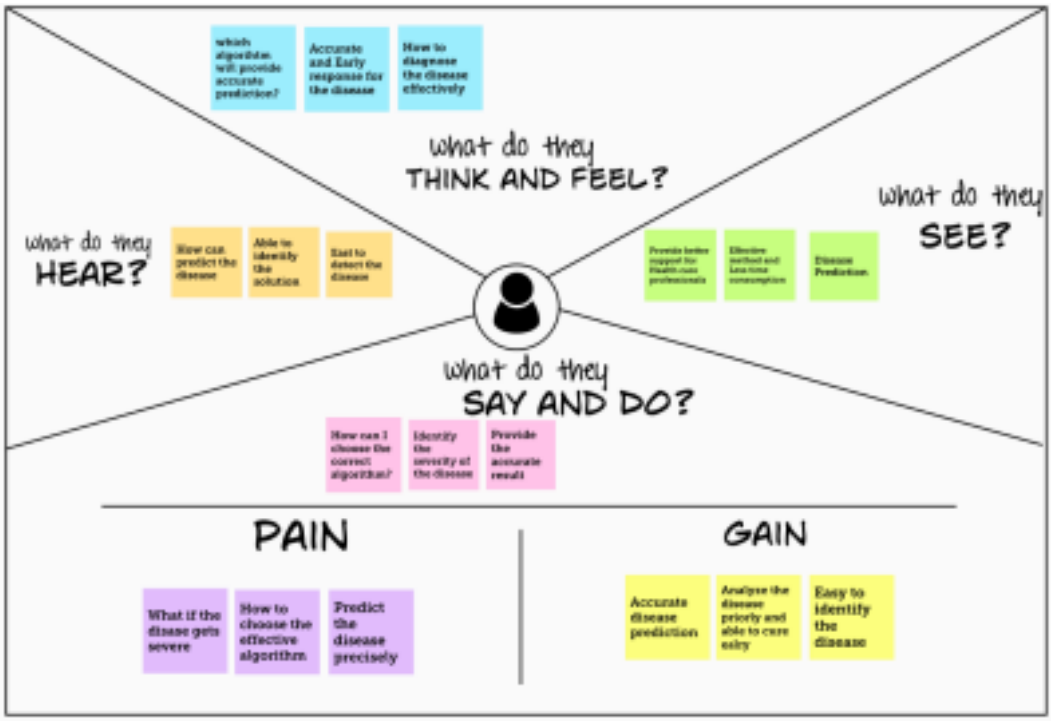

Figure 3.1 Empathy Map
A collaborative visualization used to articulate user insights

An empathy map is a widely-used visualization tool within the field .In relation to empathetic design, the primary purpose of an empathy map is to bridge the understanding of the end user.

3.2 IDEATON AND BRAINSTORMING

Brainstorm and Idea Prioritization Template

Step-1: Team Gathering, Collaboration and Select the Problem Statement

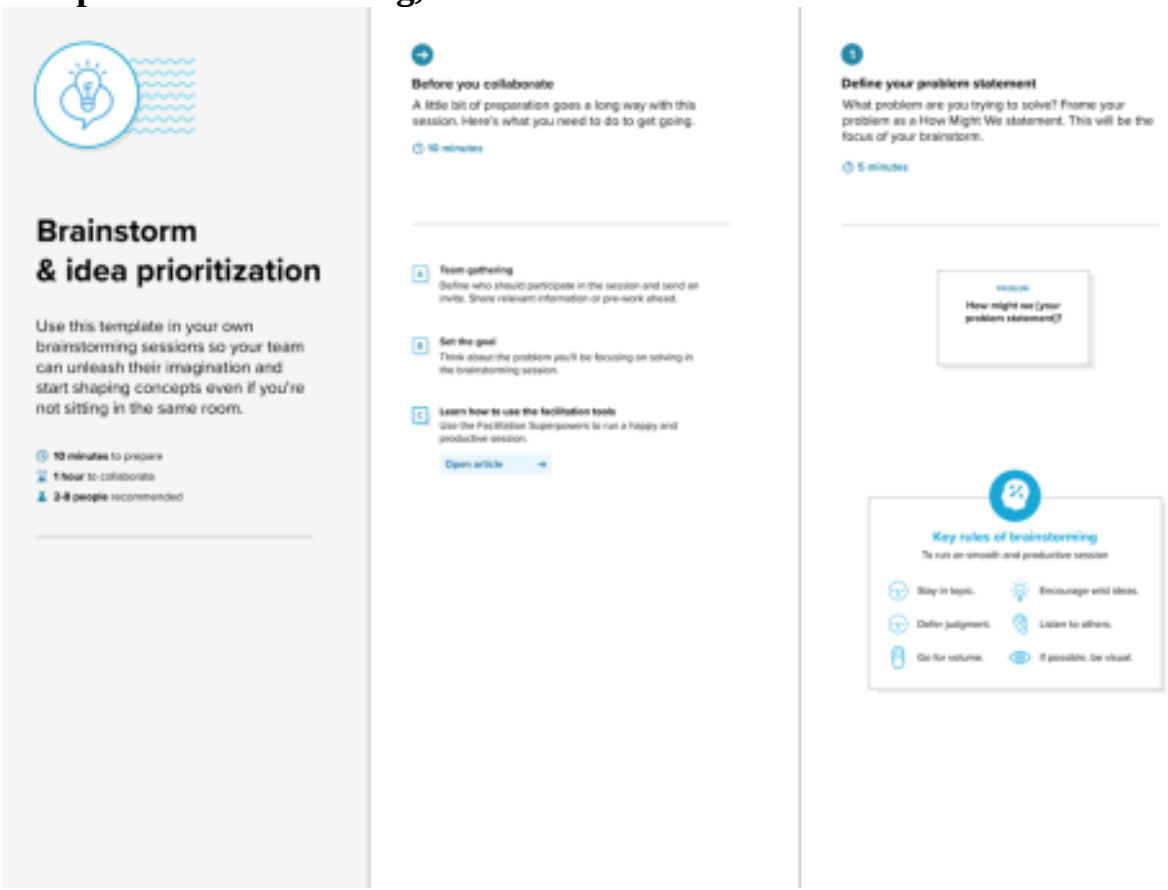

Figure 3.2 Ideation and Brainstorming

Team Gathering, Collaboration and Select the Problem Statement

A principal difference between ideation and brainstorming is that ideation is commonly more thought of as being an individual pursuit, while brainstorming is almost always a group activity.

Step-2: Brainstorm, Idea Listing and Grouping

Figure 3.3 Brainstorm, Idea Listing and Grouping

The idea listing and grouping is used to organize and analyse large numbers of ideas by categorising them. By organising and reorganising ideas, students gain a better appreciation of, and dialogue about, their ideas. As students create idea clusters, new contexts and connections among themes emerge.

Step-3: Idea Prioritization

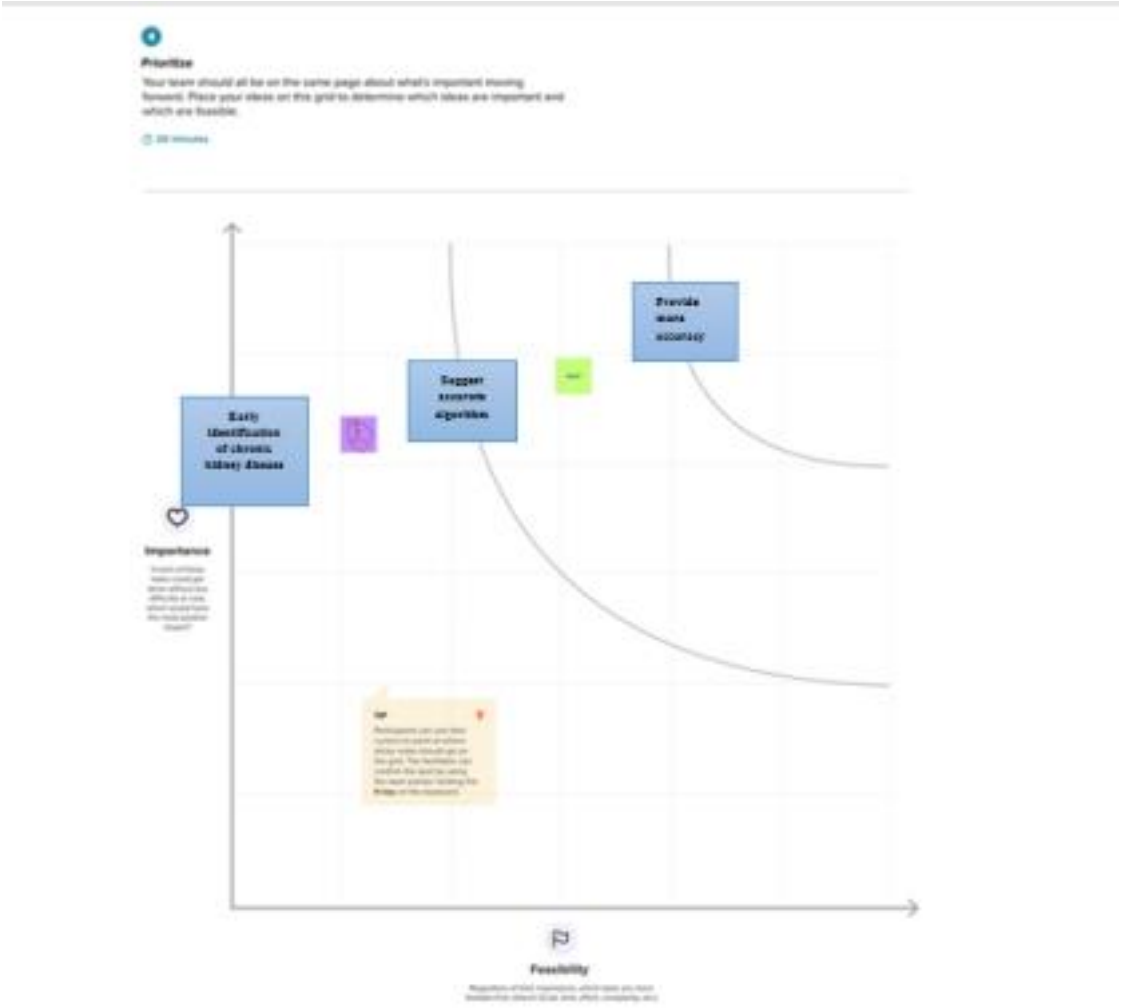

Figure 3.4 Idea Prioritization

Idea prioritization is just a part of the idea management process. Having a structured idea management process and a systematic way of gathering, evaluating and prioritizing new ideas takes time. To make it work, the entire idea management process should be integrated to the everyday ways of working.

3.3 PROPOSED SOLUTION

Project team shall fill the following information in proposed solution template.

S. No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	Early Detection of Chronic Kidney Disease using Machine Learning.
2.	Idea / Solution description	Early detection and cure of Chronic Kidney Disease (CKD) is extremely desirable as it can lead to the prevention of unwanted consequences. Machine learning methods are used to predict the various stages of CKD using the dataset obtained from the medical records of affected people.
3.	Novelty / Uniqueness	Specifically, we have used the Random Forest and J48 algorithms to obtain a sustainable and practicable model to detect various stages of CKD with comprehensive medical accuracy.
4.	Social Impact / Customer Satisfaction	As the Chronic Kidney Diseases (CKD) is a silent disease, as most sufferers have no symptoms until kidney function drops. Our project can be a huge game changer in a medical field by helping doctors to predict the CKD in a early stage and to the lives of thousands of people.
5.	Business Model (Revenue Model)	This application is recommended to patients in low cost with subscription basis.
6.	Scalability of the Solution	Machine-learning methods can be employed to analyze nanomaterials better and nanoscale biological materials, and aid in the effort to find new materials and the best routes to design nanomaterials optimally.

Table 3.5 Proposed Solution

3.4 PROPOSED SOLUTION FIT

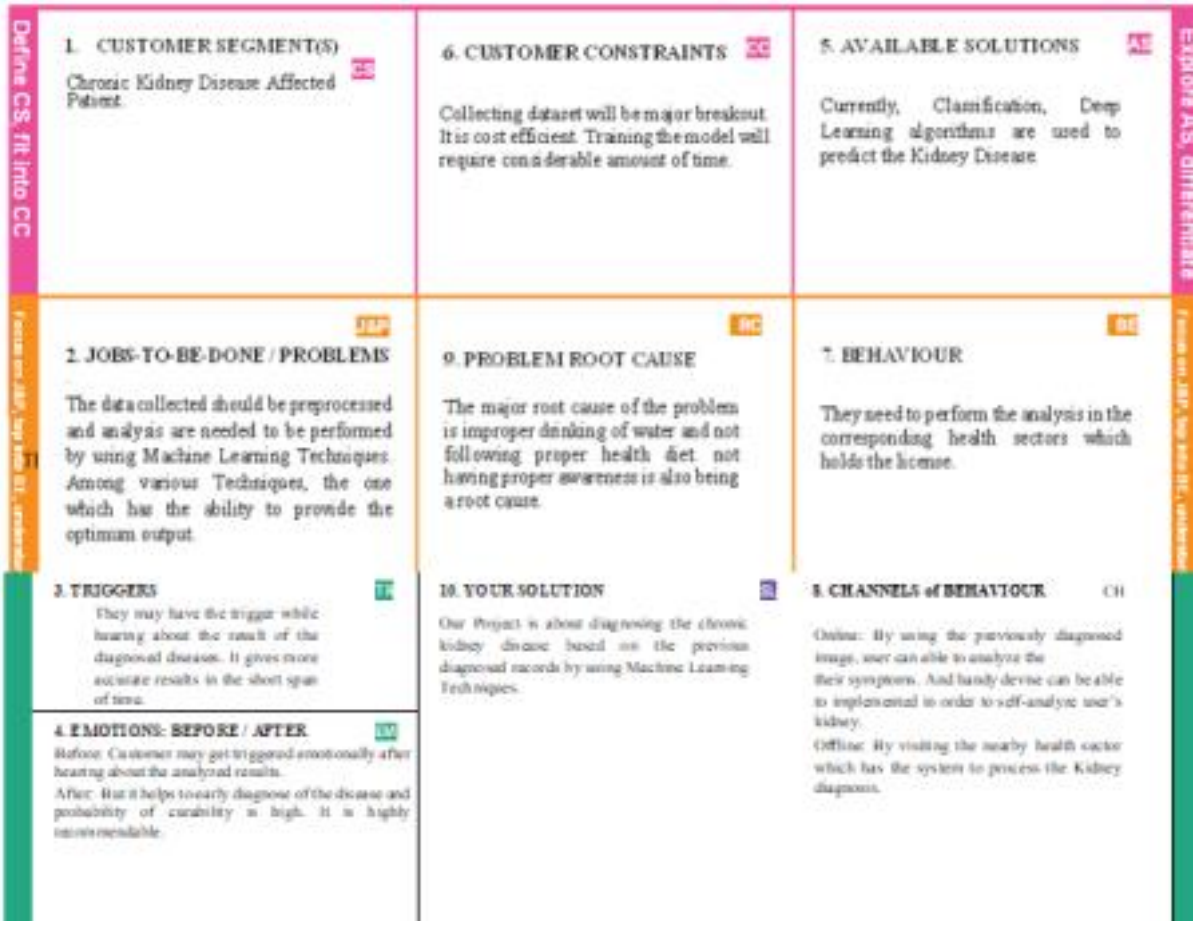

Figure 3.6 Solution fit of design wiht user requirements

This occurs when the user have evidence that customers care about certain jobs, pains, and gains. At this stage the user proved the existence of a problem and have designed a value proposition that addresses customers' jobs, pains and gains.

CHAPTER 4

REQUIREMENT ANALYSIS

4.1 FUNCTIONAL REQUIREMENTS

Following are the functional requirements of the proposed solution.

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Registration	Registration through Form Registration through Gmail.
FR-2	User Confirmation	Confirmation via Email Confirmation via OTP Confirmation via Phone number.
FR-3	Capturing image	Capture the image of the kidney by using Radioactive material and check the parameter of the scanned image.
FR-4	Capturing image	Upload the image for the prediction of the disease in the kidney.
FR-5	Kidney Identification	Identify the kidney and predict the disease in the kidney.
FR-6	Image Description	Suggestion the best method for diagnosing the disease.

Table 4.1 Functional Requirements

4.2 NON-FUNCTIONAL REQUIREMENTS

Following are the non-functional requirements of the proposed solution.

FR No.	Non-Functional Requirement	Description
NFR-1	Usability	Datasets of all the kidney is used to find the disease that present in the kidney.
NFR-2	Security	The information belongs to the user and kidney are secured highly without vulnerable to the malicious users.
NFR-3	Reliability	The dataset collected on the kidney should be important for predicting the disease in the kidney.
NFR-4	Performance	Performance is based on the collected dataset which is used for disease prediction.
NFR-5	Availability	It is available for all the user who tend to predict the disease in the kidney.
NFR-6	Scalability	Increasing the analysis range for the prediction of disease in the kidney.

Figure 4.2 Non-Functional Requirements

CHAPTER 5

PROJECT DESIGN

5.1 DATA FLOW DIAGRAMS

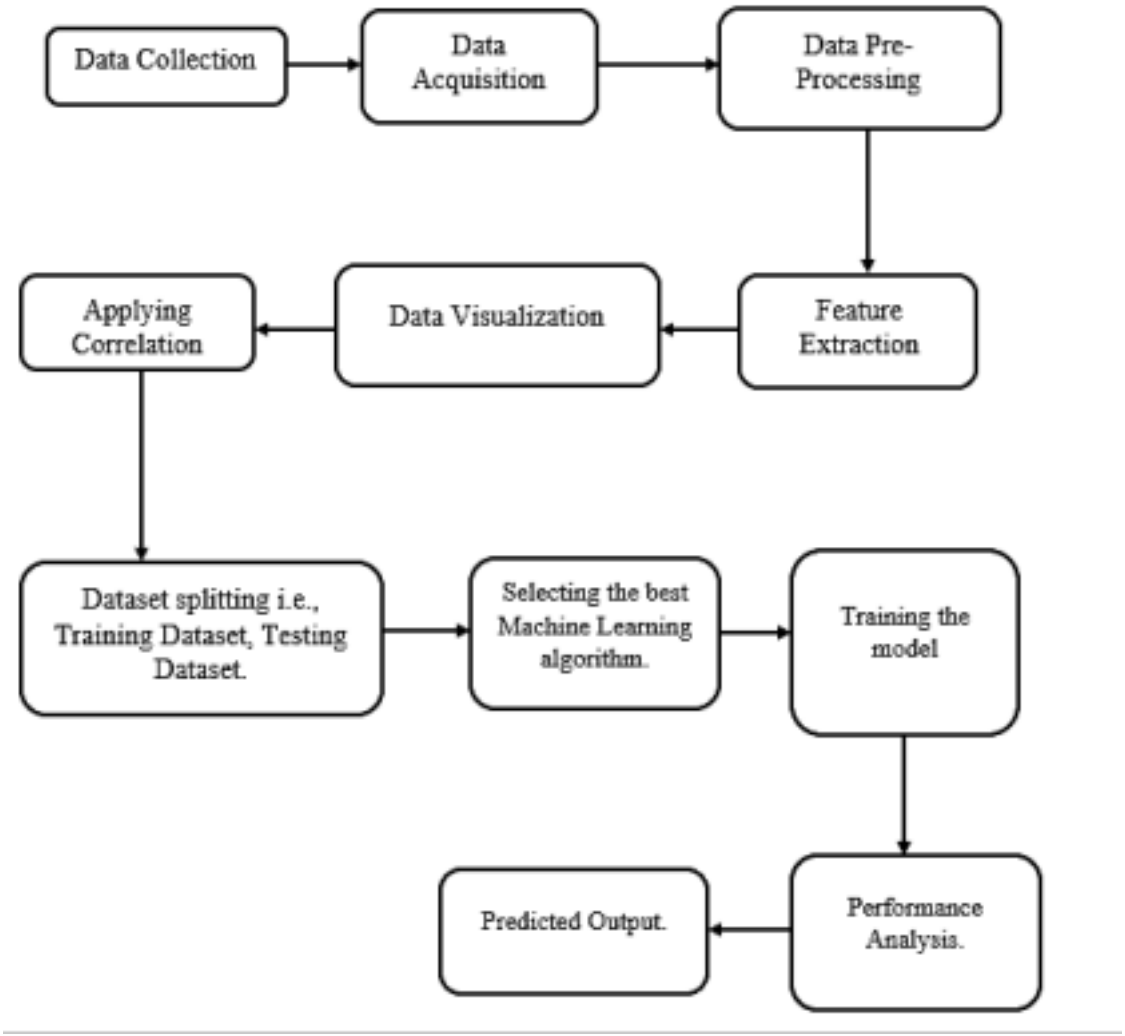

Figure 5.1 Data Flow Diagrams

A data flow diagram is a graphical or visual representation using a standardized set of symbols and notations to describe a business's operations through data movement. They are often elements of a formal methodology such as Structured Systems Analysis and Design Method.

5.2 SOLUTION AND TECHNICAL ARCHITECTURE

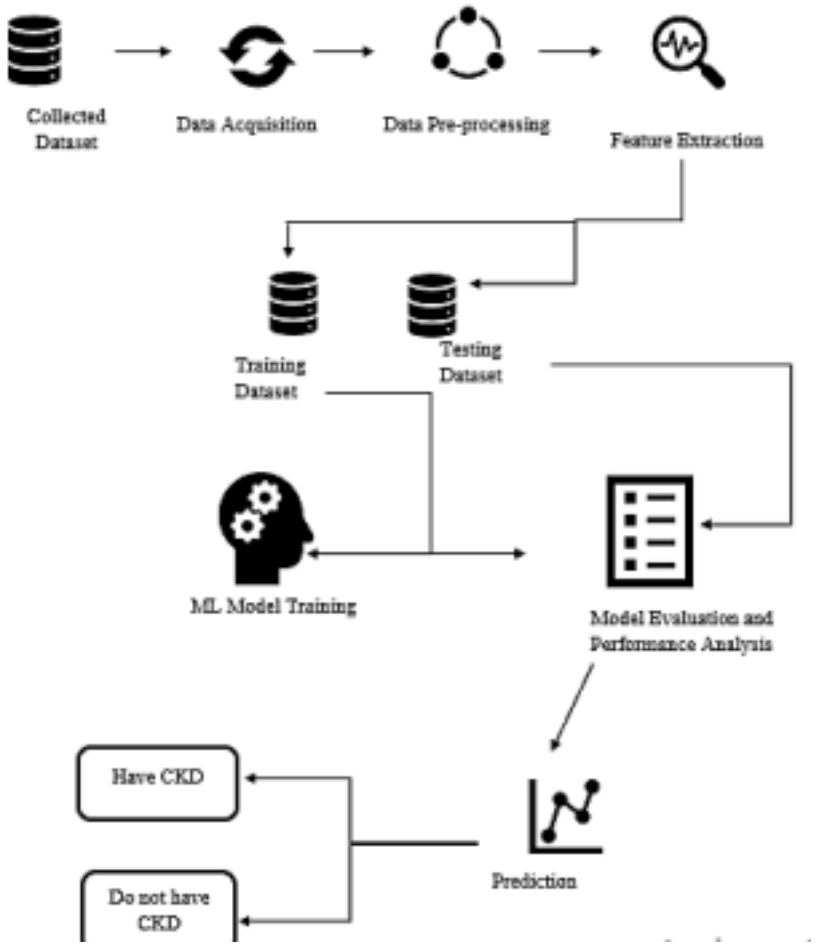

Figure 5.2 Project flow deployment

Solution Architects are most similar to project managers, ensuring that all parties, including stakeholders, are on the same page and moving in the right direction at all stages. Technical architects manage all activities leading to the successful implementation of a new application. A solution architect must have a technical background with at least eight years of work experience in one or more IT areas including but not limited to: IT architecture, infrastructure, and cloud development.

5.3 USER STORIES

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer (Web user) Step 1	Registration	USN-1	As a user, I can register for the site by entering my email, password, and confirming my email id.	I can access my account and view the details.	High	Sprint-1
Customer Step 2	Confirmation	USN-2	As a user, I will receive confirmation email once I have registered for the site.	I can receive confirmation email and click confirm.	High	Sprint-1
Customer Step 3	Login	USN-3	As a user, I can login into the site by clicking onto the login link.	I can successfully login to the page and my details will be shown.	Low	Sprint-2
Customer Step 4	Dashboard	USN-4	As a user, I can access my dashboard.	I can modify the details in the dashboard.	Medium	
Customer Step 5	Homepage	USN-5	As a user, I can view the contact details and required information.	Based on user requirements ,Contents are categorized.	High	Sprint-3
Customer Care Executive	Help	USN-6	As a user, I could contact the site owner if I faced any issues.	Report issues option will be provided.	High	Sprint-3

Table 5.3 User Stories

CHAPTER 6

PROJECT PLANNING AND SCHEDULING

6.1 SPRINT PLANNING AND ESTIMATION

Sprint	Functional Requirement (Epic)	User Story Number	User Story/Task	Story Points	Priority	Team Members
Sprint-1	Data Collection	USN-1	Collect the suitable dataset for predicting the chronic kidney disease.	10	High	Kaviya.N
Sprint-1	Data Pre Processing	USN-2	Datasets are transformed into useful format.	7	Medium	Kaviya.N
Sprint-2	Model Building	USN-3	Calculate the Index values	10	High	Abirami.V
Sprint-2		USN-4	Splitting the Model into Training and Testing from the overall dataset.	7	Medium	Abirami.V
Sprint-3	Training and Testing	USN-5	Train the Model using Regression algorithm and testing the performance of the model.	10	High	Bhava Dharani.G
Sprint-3	Application Building	USN-6	Build the HTML and python code	7	Medium	Bhava Dharani.G
Sprint-4		USN-7	Run Flask App	10	High	Pradeep.M .M
Sprint-4	Implementation of the Application	USN-8	Deploy the model on IBM cloud.	7	Medium	Pradeep.M .M

Table 6.1 Sprint Planning and Estimation

6.2 SPRINT DELIVERY SCHEDULE

Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)
Sprint-1	10	6 Days	24 Oct 2022	29 Oct 2022
Sprint-2	10	6 Days	31 Oct 2022	05 Nov 2022
Sprint-3	10	6 Days	06 Oct 2022	12 Nov 2022
Sprint-4	10	6 Days	14 Nov 2022	19 Nov 2022

Table 6.2 Sprint Delivery Schedule

6.3 REPORTS FROM JIRA

BURNDOWN CHART

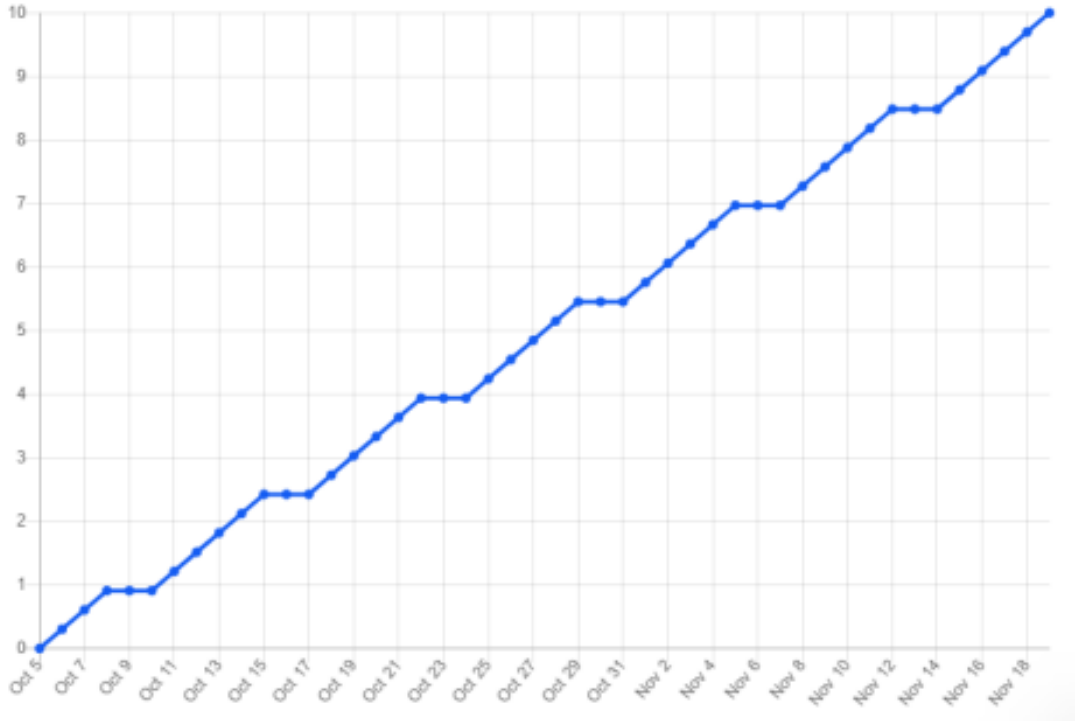

Figure 6.3 Burndown Chart

A burndown chart shows the amount of work that has been completed in an epic or sprint, and the total work remaining. Burndown charts are used to predict your

team's likelihood of completing their work in the time available.It displays the scope of a project and the work completed.

BURNUP CHART

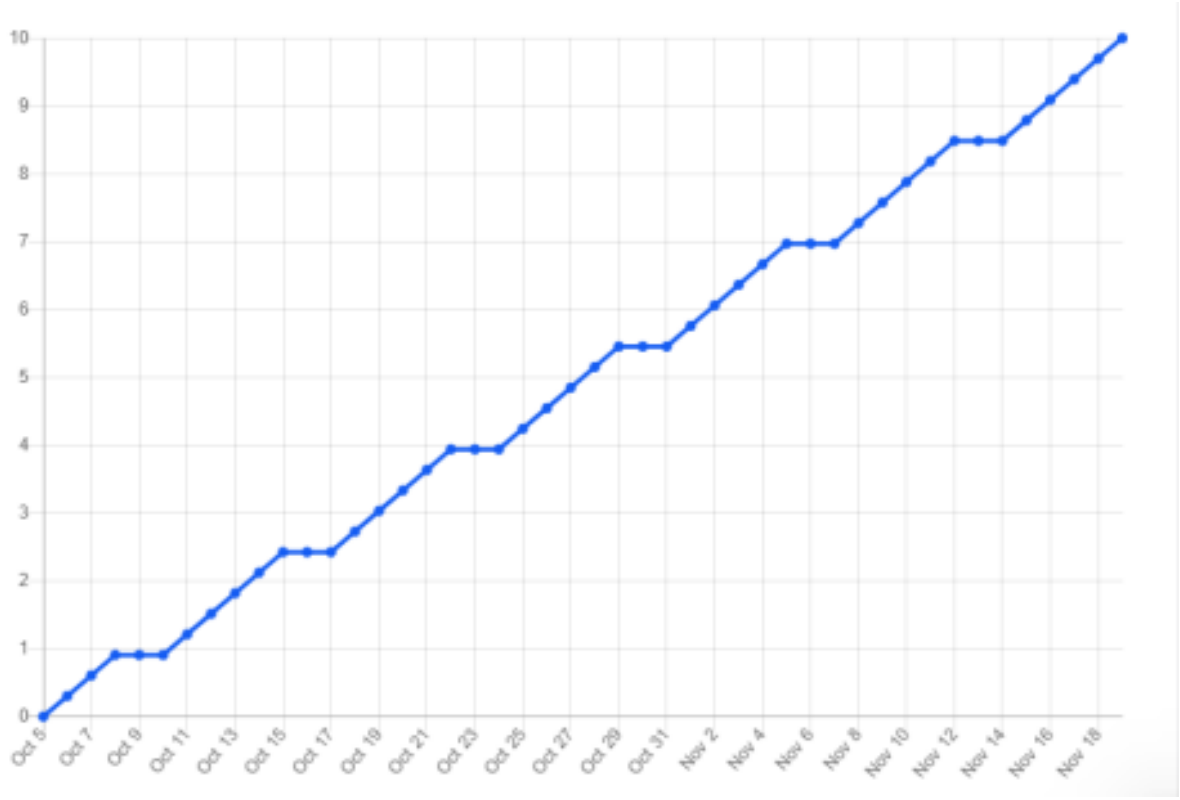

Fig 6.4 Burnup Chart

A burnup chart highlights the work you've completed against your total project scope while a burn down chart highlights the amount of work remaining in a project. A burnup chart contains a work completed line and a project scope line. It displays the scope of a project and the work completed.

CHAPTER 7

CODING AND SOLUTIONING

7.1 FEATURE 1(RANDOM FOREST ALGORITHM MODEL)

The first feature of the deployement is the process of Random Forest Classifier is used to train and test the model for detecting the Liver Disease with the help of collected and pre-processed dataset collections.

 # Train Test Split

X_train, X_test, y_train, y_test = train_test_split(df2.iloc[:,:-1], df2['classification'], test_size = 0.33, random_state=44,stratify= df2['classification'])

#Correlation Dataset

df_complete_corr=complete_correlation['corr']

df_complete_corr.dropna(axis=1,how='all').dropna(axis=0,how='all').style.background_gradient(cmap='nipy_spectral_r', axis=None).set_precision(2)

#Tuned Parameters

tuned_parameters =
[{'n_estimators':[7,8,9,10,11,12,13,14,15,16],'max_depth':[2,3,4,5,6,None],

'class_weight':[None,{0:0.33,1:0.67},'balanced'],'random_state':[42]}]

clf = GridSearchCV(RandomForestClassifier(), tuned_parameters,
cv=10,scoring='f1')

clf.fit(X_train, y_train)

Random forest is a Supervised Machine Learning Algorithm that is used widely in Classification and Regression problems. It builds decision trees on different samples and takes their majority vote for classification and average in case of regression.

 # RandomForestRegressor

from sklearn.ensemble import RandomForestRegressor

reg=RandomForestRegressor()

reg.fit(X_train,y_train)

7.2 FEATURE 2(FLASK CONNECTIVITY)

Python flask is the first feature that helps to complete this project. It allows the user to create local server and host the website in a local machine.

#Connecting index.html

```
from flask import Flask, request, redirect, render_template
app = Flask(__name__)
@app.route("/",methods=['GET', 'POST'])
def index():
    return render_template('index.html')
@app.route("/val",methods=['POST'])
```

#Connecting rename.html and rename2.html

```
test_df=pd.DataFrame(test)
test_df=np.array(test_df).reshape(1, -1)

ans1=loaded_class.predict(test_df)
ans2=loaded_reg.predict(test_df)
if int(ans1)>=0.5:
    return render_template('rename.html')
else:
    return render_template('rename2.html')
```

#Debugging

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

7.3 DATABASE SCHEMA

In the recent decades, the evolution of omics technologies has led to advances in all biological fields, creating a demand for effective storage, management and exchange of rapidly generated data and research discoveries. To address this need, the development of databases of experimental outputs has become a common part of scientific practice in order to serve as knowledge sources and data-sharing platforms, providing information about genes, transcripts, proteins or metabolites. In this review, we present omics databases available currently, with a special focus on their application in kidney research and possibly in clinical practice. Databases are divided into two categories: general databases with a broad information scope and kidney-specific databases distinctively concentrated on kidney pathologies. In research, databases can be used as a rich source of information about pathophysiological mechanisms and molecular targets. In the future, databases will support clinicians with their decisions, providing better and faster diagnoses and setting the direction towards more preventive, personalized medicine. We also provide a test case demonstrating the potential of biological databases in comparing multi-omics datasets and generating new hypotheses to answer a critical and common diagnostic problem in nephrology practice. In the future, employment of databases combined with data integration and data mining should provide powerful insights into unlocking the mysteries of kidney disease, leading to a potential impact on pharmacological intervention and therapeutic disease management.

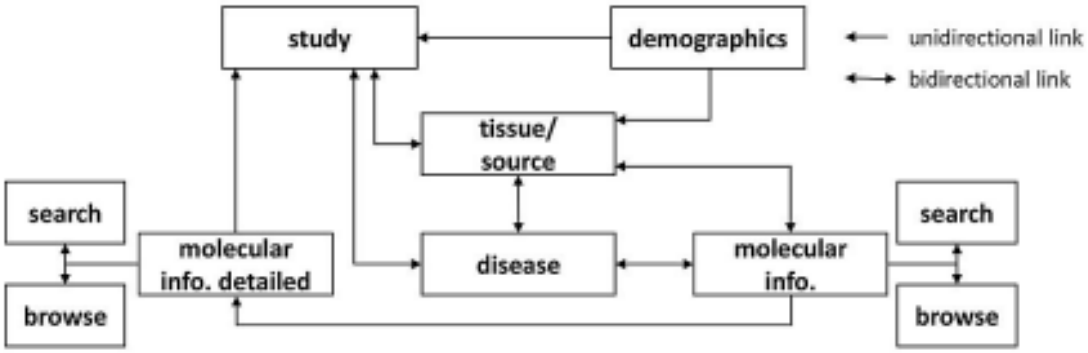

Figure 7.1 Database Schema

CHAPTER 8

TESTING

8.1 TEST CASES

Test Case ID	15455	**Test Case Description**	Test the Chronic Kidney Disease Prediction Functionality	
S #	**Prerequisites:**		1	By Clicking the website link
1	Access to Chrome Browser		2	Details should be in a integer format
2	Entering the details required		3	Data should be filled
3	check for correct values		4	Provide the datasets for model training
4	Application to train the model			

Test Scenario Verify whether the deployed project predicts as per expected

Step #	**Step Details**	**Expected Results**	**Actual Results**	**Pass / Fail / Not executed / Suspended**
1	Navigate to corresponding website link	Site should open	As Expected	Pass
2	Enter the details	Details should be entered	As Expected	Pass
3	Click Submit	Check the result	As Expected	Pass
4	Output results	Results are generated	As Expected	Pass

Table 8.1 Test Cases

8.2 USER ACCEPTANCE TESTING

USER ACCEPTANCE TEST-1

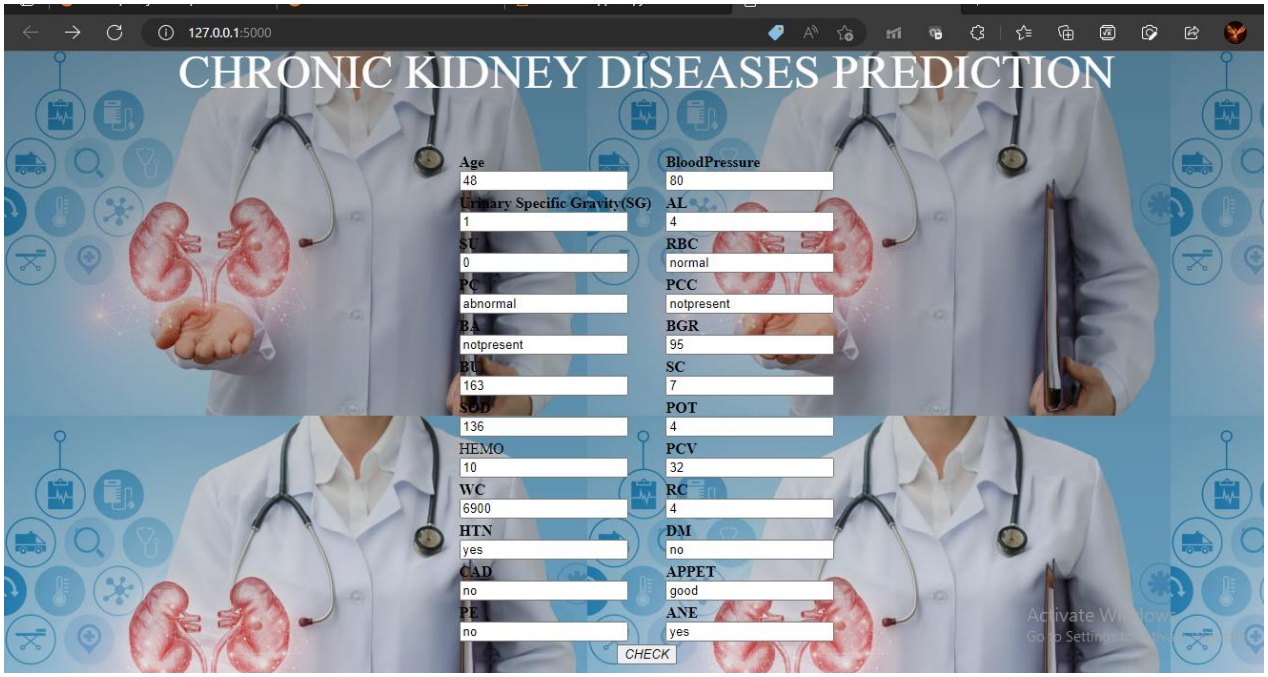

Figure 8.2 User Interface

In this page, the user needs to provide the corresponding values which was provided during the medical analysis of the kidney.

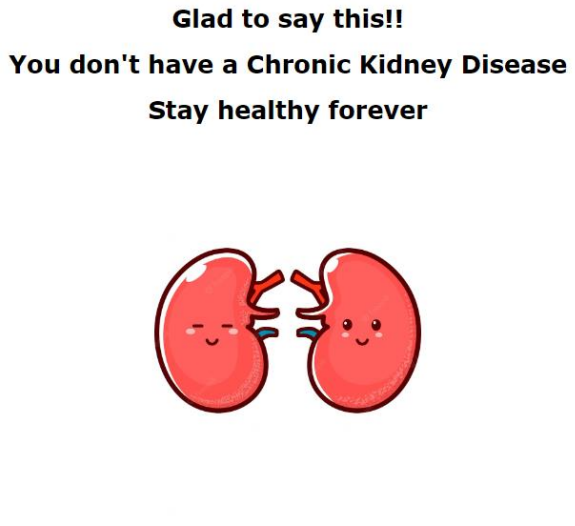

Figure 8.3 Predicted Output Test Case for Non-Chronic Kidney Disease

This page will pop out after diagnosing the provided input values as there is no chronic kidney disease predicted.

USER ACCEPTANCE TEST-2

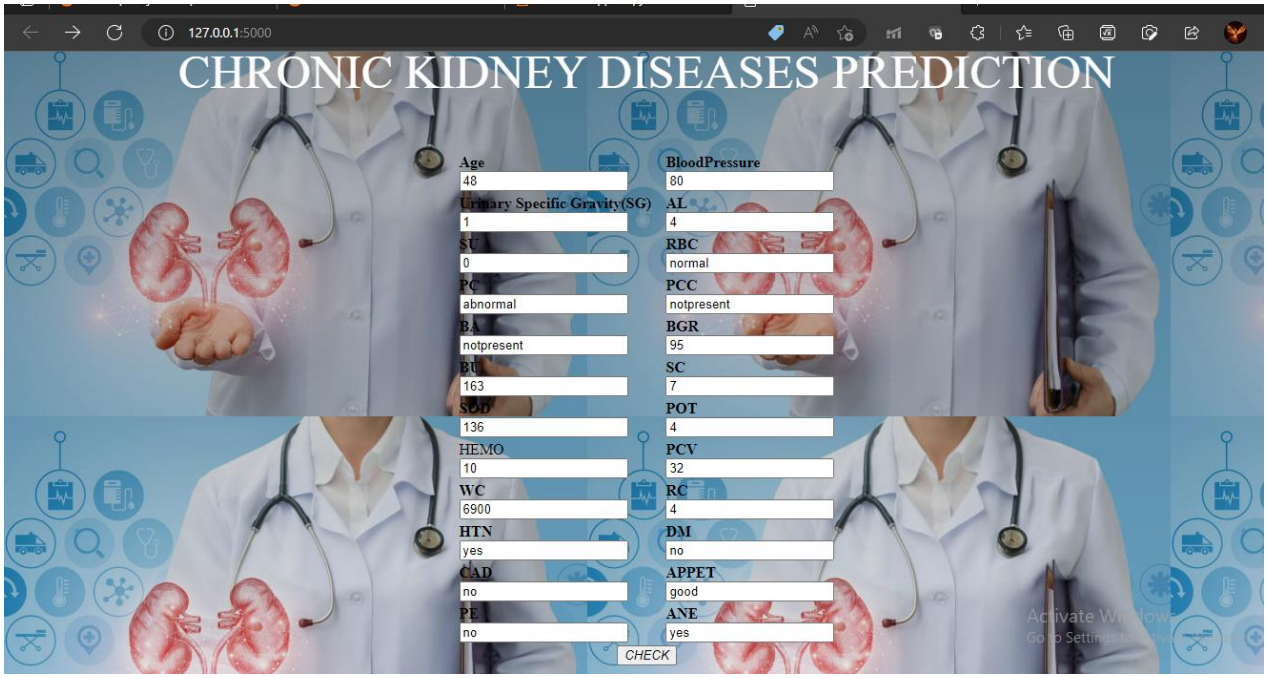

Figure 8.4 User Interface

In this page, the user needs to provide the corresponding values which was provided during the medical analysis of the kidney.

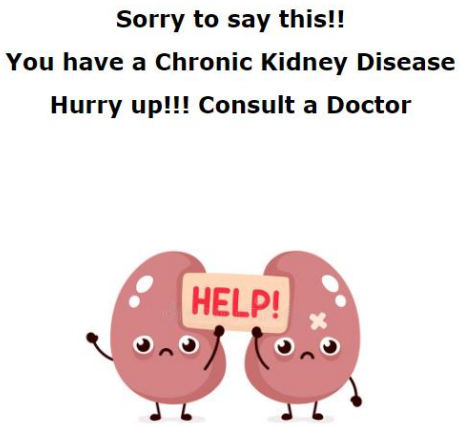

Figure 8.5 Predicted Output Test Case for Chronic Kidney Disease

This page will pop out after diagnosing the provided input values as there is no chronic kidney disease predicted.

CHAPTER 9

RESULTS

9.1 PERFORMANCE METRICS

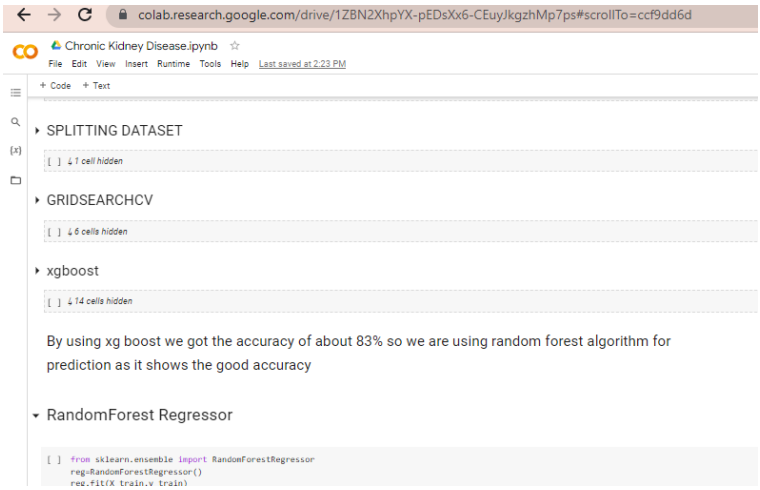

Figure 9.1 Accuracy of XGBoost Algorithm

This algorithm achieved about 83% of accuracy. In order to improve that accuracy Random Forest Algorithm is used. Through that algorithm about 89% accuracy is achieved.

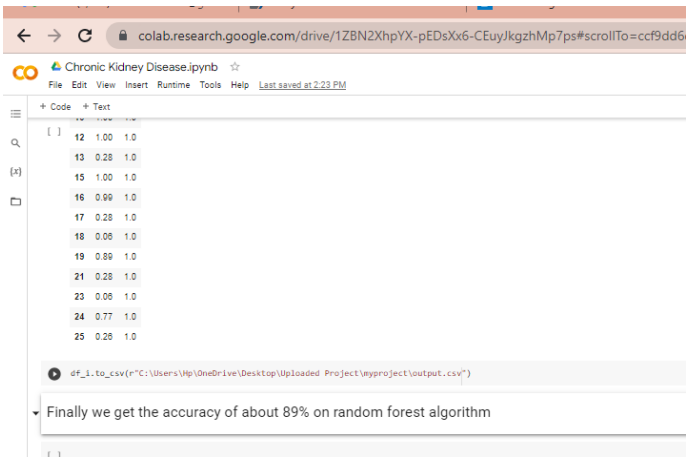

Figure 9.2 Accuracy of Random Forest Algorithm

CHAPTER 10

ADVANTAGES AND DISADVANTAGES

ADVANTAGES

- Allows patients to receive timely treatment
- Slow down the disease's progression
- Prevent cardiovascular
- Timely identification of dialysis
- Effective treatment of mild disease
- Avoid progression to kidney failure

DISADVANTAGES

- Not sure whether it always predicts correctly
- Missing values may hold the drawback
- Still not achieved 100% optimization
- Needs advanced machine learning methods

CHAPTER 11

CONCLUSION

Chronic Kidney Disease (CKD) or chronic renal disease has become a major issue with a steady growth rate. A person can only survive without kidneys for an average time of 18 days, which makes a huge demand for a kidney transplant and Dialysis. It is important to have effective methods for early prediction of CKD. Machine learning methods are effective in CKD prediction. This work proposes a workflow to predict CKD status based on clinical data, incorporating data prepossessing, a missing value handling method with collaborative filtering and attributes selection. Out of the 11 machine learning methods considered, the extra tree classifier and random forest classifier are shown to result in the highest accuracy and minimal bias to the attributes. The research also considers the practical aspects of data collection and highlights the importance of incorporating domain knowledge when using machine learning for CKD status prediction.

CHAPTER 12

FUTURE SCOPE

The increasing rate of CKD has placed a large negative impact on individuals' lives. Advanced ML technology has made the early detection of CKD easier and more accurate. Doctors and medical care professionals have used ML algorithms in the effective diagnosis of CKD. However, there is very little research on the detection of secondary infections of prolonged CKD such as albuminuria and toxin production through the ML algorithm. These secondary infections also place a negative impact, especially on diabetics and patients with high blood pressure. Therefore, Determination of the role of ML algorithms in detecting CKD associated diseases can be an effective research. Further research on effective treatment prediction and nutritional chart prediction of CKD patients through ML algorithm needs to be done in the future. Advanced technologies such as CNN, ML, random forest, and different classifiers can be used for these aspects to increase the recovery rate in CKD. By following this way, researchers and medical care professionals can enhance their service quality in accurate CKD diagnosis and treatment. Effective detection of CKD through ML algorithm is rapid and cost effective, and due to this reason, the method can gain large popularity in the future.

CHAPTER 13

APPENDIX

Source Code:

Random Forest Alogirhtm:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
#from python.nominal import associations
import warnings
warnings.filterwarnings('ignore')
from google.colab import drive
drive.mount('/content/drive')
df=pd.read_csv(r"/content/chronickidneydisease.csv")
df
df.duplicated().sum()
df.info()
def convert_dtype(df,feature):
    df[feature]=pd.to_numeric(df[feature],errors='coerce')
features=['pcv','wc','rc'] for i in features:
    convert_dtype(df,i)
plt.subplot(1,2,1)
sns.boxplot(x=df['classification'],y=df['age'])
def extract_cat_num(kidney):
    cat_col=[col for col in kidney.columns if kidney[col].dtype=='O']
    num_col=[col for col in kidney.columns if kidney[col].dtype!='O']
    return cat_col,num_col
cat_col,num_col=extract_cat_num(df)
len(num_col)
plt.figure(figsize=(30,30))
for i,feature in enumerate(num_col):
    plt.subplot(5,3,i+1)
    df[feature].hist()
    plt.title(feature)
len(cat_col)
plt.figure(figsize=(20,20))
for i,feature in enumerate(cat_col):
    plt.subplot(4,3,i+1)
```

```
    sns.countplot(df[feature])

df.groupby(['rbc','classification'])['rc'].agg(['count','mean','median','min','max'])
plt.figure(figsize=(10,10))
plt.scatter(x=df.hemo,y=df['pcv'],color="red")

plt.xlabel('Hemo')

plt.ylabel('pcv')

plt.title('Relationship between haemoglobin and packed cell volume')

Text(0.5, 1.0, 'Relationship between haemoglobin and packed cell volume')
grid=sns.FacetGrid(df,hue='classification',aspect=2)
grid.map(sns.kdeplot,'rc')

grid.add_legend()

plt.figure(figsize=(12,10))

sns.scatterplot(x=df['rc'],y=df['hemo'],hue=df['classification'])
plt.xlabel('rc')
plt.ylabel('hemo')

plt.title('Relationship between haemoglobin and red blood cell count')
Text(0.5, 1.0, 'Relationship between haemoglobin and red blood cell count')
from dython.nominal import identify_nominal_columns
categorical_features=identify_nominal_columns(df)

categorical_features

complete_correlation= associations(df, filename= 'complete_correlation')
df_complete_corr=complete_correlation['corr']
df_complete_corr.dropna(axis=1, how='all').dropna(axis=0, how='all')
df.corr().style.background_gradient(cmap="nipy_spectral_r")
df=df.drop(["id"],axis=1)
df=df.drop(["age"],axis=1)

df=df.drop(["wc"],axis=1)

df.columns

df.describe()

missing_values=df.columns[df.isnull().any()]

df[missing_values].isnull().sum()

labels = []

valuecount = []

percentcount = []

for col in missing_values:
```

```
    labels.append(col)

    valuecount.append(df[col].isnull().sum())

    percentcount.append(df[col].isnull().sum()/df.shape[0])
ind = np.arange(len(labels))
fig, (ax1, ax2) = plt.subplots(1,2,figsize=(10,5))

rects = ax1.barh(ind, np.array(valuecount), color='yellow')
ax1.set_yticks(ind)
ax1.set_yticklabels(labels, rotation='horizontal')

ax1.set_xlabel("Count of missing values")

ax1.set_title("Variables with missing values");

rects = ax2.barh(ind, np.array(percentcount), color='green')
ax2.set_yticks(ind)
ax2.set_yticklabels(labels, rotation='horizontal')
ax2.set_xlabel("Percentage of missing values")

ax2.set_title("Variables with missing values");

print("Total count of missing value in a dataset:",df.isnull().sum().
df['bp'] = df['bp'].fillna(df['bp'].mean())
df["bp"].isnull().sum()

df['sg'] = df['sg'].fillna(df['sg'].mean())

df["sg"].isnull().sum()

df['al'] = df['al'].fillna(df['al'].mean())

df["al"].isnull().sum()

df['su'] = df['su'].fillna(df['su'].mean())

df["su"].isnull().sum()

df['rbc'] = df['rbc'].fillna("not mentioned")

df["rbc"].isnull().sum()

df['pc'] = df['pc'].fillna(df['pc'].mode()[0])

df["pc"].isnull().sum()

df = df.dropna(axis=0, subset=['pcc',"htn","appet"])
df['bgr'] = df['al'].fillna(df['al'].mean())
df["bgr"].isnull().sum()

df['bu'] = df['bu'].fillna(df['bu'].mean())

df["bu"].isnull().sum()
```

```
df['sc'] = df['sc'].fillna(df['sc'].mean())

df['sod'] = df['sod'].fillna(df['sod'].mean())

df['pot'] = df['pot'].fillna(df['pot'].mean())

df['pcv'] = df['pcv'].fillna(df['pcv'].mode()[0])

df['hemo'] = df['hemo'].fillna(df['hemo'].mean())

df['rc'] = df['rc'].fillna("not mentioned")
df["rc"].isnull().sum()

df.info()

df.isnull().sum()

for column in df:

    if (df[column].dtype=="object"):

       print("\n",column)

       print(df[column].value_counts(),"\n")

or column in df:

    if (df[column].dtype=="object"):

       print("\n",column)

       print(df[column].value_counts(),"\n")

objList = df.select_dtypes(include = "object").columns

print (objList)

from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()

for feat in objList:

   df[feat] = le.fit_transform(df[feat].astype(str))

print (df.info())

df.dtypes

X = df.drop(['classification', 'sg',  'rc', 'pcv'], axis = 1)

y = df['classification']

X.columns

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 = 42)
from sklearn.ensemble import RandomForestClassifier
model = RandomForestClassifier(n_estimators = 20)
model.fit(X_train, y_train)

from sklearn.metrics import confusion_matrix, accuracy_score

confusion_matrix(y_test, model.predict(X_test))

print(f"Accuracy is {round(accuracy_score(y_test, model.predict(X_test))*100, 2)}%")
mport pickle
pickle.dump(model, open('kidney.pkl', 'wb'))

from xgboost import XGBClassifier

params={'learning-rate':[0,0.5,0.20,0.25],

     'max_depth':[5,8,10],

    'min_child_weight':[1,3,5,7],

    'gamma':[0.0,0.1,0.2,0.4],

    'colsample_bytree':[0.3,0.4,0.7]}

from sklearn.model_selection import RandomizedSearchCV

classifier=XGBClassifier()

random_search.best_params_

classifier=XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
        colsample_bynode=1, colsample_bytree=0.3, gamma=0.2, gpu_id=-1,
        importance_type='gain', interaction_constraints='', learning_rate=0.300000012,
max_delta_step=0,

        max_depth=5, min_child_weight=1,

        monotone_constraints='()', n_estimators=100, n_jobs=8,

        num_parallel_tree=1, random_state=0, reg_alpha=0, reg_lambda=1,
        scale_pos_weight=1, subsample=1, tree_method='exact',
        validate_parameters=1, verbosity=None)

y_pred=classifier.predict(X_test)

random_search=RandomizedSearchCV(classifier,param_distributions=params,n_iter=5,scoring=
'r')
random_search.fit(X_train,y_train)

random_search.best_estimator_    #Checking for best model
from sklearn.metrics import confusion_matrix,accuracy_score
confusion_matrix(y_test,y_pred)
array([[54,  0],
```

```
      [ 0, 23]], dtype=int64)

accuracy_score(y_test,y_pred)

from sklearn.ensemble import RandomForestRegressor
reg=RandomForestRegressor()
reg.fit(X_train,y_train)

X_test.columns

y_pred=reg.predict(X_test)

pickle. dump(reg, open('randomreg_chronic', 'wb'))

y_pred

l_pred=list(y_pred)

l_test=list(y_test)

d={'prob':l_pred,'out':y_test}

df_i=pd.DataFrame(d)

df_i.head(20)

df_i.to_csv(r"/content/chronickidneydisease.csv")
```

HTML:

index.html

```
<html>

<head>

<meta http-equiv="Content-Type" content="text/html;charset=UTF-8">
<title>CKD</title>
<style>

body {

   background: linear-gradient(

                rgba(10,10,10, .35),

                rgba(10,10,10, .105)),

                url("https://reflectionbusiness.com/wp-
content/uploads/2022/10/kidney.jpg");     background-position: center;
  background-repeat: no-repeat;

  background-size: cover;

.container {

  border: 2px solid #ccc;
```

```
  padding: 10px;
  width: 20em;
height:21em;
background-color:white;
}
.hello{
opacity: 0.5;
}
</style>
</head>
<body>
<p style="font-size:30px;color:white;">
<center><p style="font-size:50px;color:WHITE;">CHRONIC KIDNEY DISEASES PREDICTION</p></center>
<form action="/val" method="post"><center>
<table width="30%" align="center">
        <tr>
            <td width="60%"></td>
            <td width="60%"></td>
        </tr>
        <tr>
            <td>
                <label for="" width="">Age</label><br />
                <input type="text" name="" id="">
            </td>
            <td>
                <label for="" width="">Blood Pressure</label><br />
                <input type="text" name="" id="">
            </td>
        </tr>
        <tr>
            <td>
                <label for="" width="">Urinary Specific Gravity(SG)</label><br />
                <input type="text" name="" id="">
            </td>
```

```
            <td>
                <label for="" width="">AL</label><br />
                <input type="text" name="" id="">
            </td>
        </tr>
        <tr>
            <td>
                <label for="" width="">SU</label><br />
                <input type="text" name="" id="">
            </td>
            <td>
                <label for="" width="">RBC</label><br />
                <input type="text" name="" id="">
            </td>
        </tr>
        <tr>
            <td>
                <label for="" width="">PC</label><br />
                <input type="text" name="" id="">
    </td>
            <td>
                <label for="" width="">PCC</label><br />
                <input type="text" name="" id="">
 </td>
        </tr>
        <tr>
            <td>
                <label for="" width="">BA</label><br />
                <input type="text" name="" id="">
   </td>
            <td>
                <label for="" width="">BGR</label><br />
                <input type="text" name="" id="">
</td>
        </tr>
          <tr>
            <td>
```

```
            <label for="" width="">BU</label><br />
            <input type="text" name="" id="">
    </td>
          <td>
            <label for="" width="">SC</label><br />
            <input type="text" name="" id="">
    </td>
        </tr>
        <tr>
          <td>
            <label for="" width="">SOD</label><br />
            <input type="text" name="" id="">
  </td>
          <td>
            <label for="" width="">POT</label><br />
            <input type="text" name="" id="">
  </td>
        </tr>
        <tr>
          <td>
            <label for="" width="">HEMO</label><br />
            <input type="text" name="" id="">
</td>
          <td>
            <label for="" width="">PCV</label><br />
            <input type="text" name="" id="">
  </td>
        </tr>
       <tr>
          <td>
            <label for="" width="">WC</label><br />
            <input type="text" name="" id="">
   </td>
          <td>
            <label for="" width="">RC</label><br />
            <input type="text" name="" id="">
    </td>
        </tr>
        <tr>
          <td>
```

```
            <label for="" width="">HTC</label><br />
            <input type="text" name="" id="">
  </td>
          <td>
            <label for="" width="">DM</label><br />
            <input type="text" name="" id="">
   </td>
        </tr>
         <tr>
          <td>
            <label for="" width="">CAD</label><br />
            <input type="text" name="" id="">
 </td>
          <td>
            <label for="" width="">APPET</label><br />
            <input type="text" name="" id="">
</td>
        </tr>
         <tr>
          <td>
            <label for="" width="">PE</label><br />
            <input type="text" name="" id="">
          </td>
          <td>
            <label for="" width="">ANE</label><br />
            <input type="text" name="" id="">
          </td>
        </tr>
      </table>
</center>
 <center><strong><button type="submit">CHECK</button></center></strong>
</form>
</body>
</html>
```

rename.html

```
<html>
<head>
<style>
body {
  background-color: white;
}
</style>
</head>
<body >
<br>
<center><h1 style="font-family:verdana">Sorry to say this!!</h1>
        <h1 style="font-family:verdana">You have a Chronic Kidney Disease</h1></center>
<center><h1 style="font-family:verdana">Hurry up!!! Consult a
Doctor</h1></center><center><img alt="Qries" src="https://thumbs.dreamstime.com/b/sad-
suffering-sick-cute-kidneys-sad-suffering-sick-cute-kidneys-asks-help-character-vector-flat-
cartoon-illustration-icon-150470345.jpg"
      width=500" height="500"></center>


</body>
</html>
```

rename2.html

```
<html>
<head>
<style>
body {
  background-color: white;
}
</style>
</head>
```

```
<body >
<br>
<br>
<br>
<center><h1 style="font-family:verdana">Glad to say this!!</h1></center>
<center><h1 style="font-family:verdana">You don't have a Chronic Kidney Disease</h1></center>
<center><h1 style="font-family:verdana">Stay healthy forever</h1></center>
<div>
<center><img alt="Qries" src="C:\Users\HP\Downloads\happyyy.jpeg"
        width=500" height="500"></center>
</div>
</body>
</html>
```

FLASK CONNECTIVITY:

```
import pickle
loaded_class = pickle. load(open('randomclass_chronic', 'rb'))
loaded_reg = pickle. load(open('randomreg_chronic', 'rb'))
import numpy as np
import pandas as pd
from flask import Flask, request, redirect, render_template, jsonify
import requests
import json
 # NOTE: you must manually set API_KEY below using information retrieved from your IBM Cloud account.
API_KEY = "RlfZtnoH9uVoRIaNZCAV0QXNrfb50Yal6ca53NDToHak"
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"]
print("mltoken",mltoken)
```

```
 header = {'Content-Type': 'application/json', 'Authorization': 'Bearer ' + mltoken}
app = Flask(__name__)
@app.route("/",methods=['GET', 'POST'])
def index():
    return render_template('index.html')
@app.route("/val",methods=['POST'])
def val():
    test=[]
    if request.method == 'POST':
        test.append(request.form.get("age"))
        test.append(request.form.get("bp"))
        test.append(request.form.get("sg"))
        test.append(request.form.get("al"))
        test.append(request.form.get("su"))
        rb=request.form.get("rbc")
        if rb=='abnormal':
            test.append(1)
        else:
            test.append(0)
        pc=request.form.get("pc")
        if pc=='abnormal':
            test.append(1)
        else:
            test.append(0)
        pcc=request.form.get("pcc")
        if pcc=='present':
            test.append(1)
        else:
            test.append(0)
```

```
ba=request.form.get("ba")
if ba=='present':
    test.append(1)
else:
    test.append(0)
test.append(request.form.get("bgr"))
test.append(request.form.get("bu"))
test.append(request.form.get("sc"))
test.append(request.form.get("sod"))
test.append(request.form.get("pot"))
test.append(request.form.get("hemo"))
test.append(request.form.get("pcv"))
test.append(request.form.get("wc"))
test.append(request.form.get("rc"))
ht=request.form.get("htn")
if ht=='yes':
    test.append(1)
else:
    test.append(0)
d=request.form.get("dm")
if d=='yes':
    test.append(1)
else:
    test.append(0)
ca=request.form.get("cad")
if ca=='yes':
    test.append(1)
else:
    test.append(0)
```

```
    ap=request.form.get("appet")
    if ap=='good':
        test.append(1)
    elif ap=='poor':
        test.append(0)
    else:
        test.append(np.nan)
    p=request.form.get("pe")
    if p=='yes':
        test.append(1)
    else:
        test.append(0)
    an=request.form.get("ane")
    if an=='yes':
        test.append(1)
    else:
        test.append(0)
print(test)
test_df=pd.DataFrame(test)
test_df=np.array(test_df).reshape(1, -1)

ans1=loaded_class.predict(test_df)
ans2=loaded_reg.predict(test_df)
if int(ans1)==1:
    answer1="Sorry to say!! You have CHRONIC DISEASE!!!"
    return render_template('rename.html',answer1=answer1,answer2=ans2)
else:
    answer1="Happy to say that you don't have CHRONIC DISEASE"
    return render_template('rename2.html',answer1=answer1,answer2=ans2)
```

```
if __name__ == "__main__":
    app.debug=True
    app.run(debug=False)
import requests
import json
# NOTE: you must manually set API_KEY below using information retrieved from your IBM Cloud account.
API_KEY = "RlfZtnoH9uVoRIaNZCAV0QXNrfb50Yal6ca53NDToHak"
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"]
print("mltoken",mltoken)
header = {'Content-Type': 'application/json', 'Authorization': 'Bearer ' + mltoken}
 # NOTE: manually define and pass the array(s) of values to be scored in the next line
payload_scoring = {"input_data": [{"field": [["classification"]],
                                  "values": [[1]]}]}
response_scoring = requests.post('https://us-south.ml.cloud.ibm.com/ml/v4/deployments/cf82ff26-7e2d-4b9b-8be0-39628c3cc319/predictions?version=2022-11-17', json=payload_scoring,
 headers={'Authorization': 'Bearer ' + mltoken})
print("Scoring response")
predictions=response_scoring.json()
print(predictions)
'"predictions =response_scoring.json()
pred=predictions['predictions'][0]['values'][0][0]'"
```

Github and Project Video Demo Link:

Github Link:

https://github.com/IBM-EPBL/IBM-Project-21212-1659775228

Project Video Demo Link:

https://drive.google.com/file/d/1m3s17yxs09BnRpnNk1RTB9rmuGAXcMBr/view?usp=sharing