IBM NALAIYA THIRAN – PROJECT REPORT

EARLY DETECTION OF CHRONIC KIDNEY DISEASE USING MACHINE LEARNING

TEAM ID: PNT2022TMID12917

ARCHANA A (19L201)

ABARNA S (19L202)

BHUVANESH G (19L208)

SRI RAM R (19L255)

Dissertation submitted in partial fulfilment of the requirements for the degree of

BACHELOR OF ENGINEERING

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

Of Anna University

NOVEMBER 2022

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

PSG COLLEGE OF TECHNOLOGY

(Autonomous Institution)

COIMBATORE – 641 004

CONTENTS

1: INTRODUCTION

1.1 Project Overview

Early detection of chronic kidney disease using machine learning is to test the presence of kidney disease quickly and diagnose with appropriate medications. Since late detection of such diseases might lead to fatal reactions and medications might be expensive. Because to reduce such impact of kidney disease with detecting the disease in early stages. So that by utilizing ML algorithms, detection of kidney diseases can be done earlier and the risks due to the kidney diseases can also be reduced if it is detected earlier.

1.2 Purpose

Chronic kidney disease is A severe medical problem that should be treated earlier. Several medical tests taken for some purposes may also contain valuable information related to kidney disease. As a result the attributes of tests are investigated to distinguish which attribute may contain helpful information about the disease. Further on this information helps to measure the severity of the problem earlier and helps to make use of those information to build a machine learning model to predict the data. This 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.

2: LITERATURE SURVEY

2.1: Existing Problem

Chronic kidney disease, also called chronic kidney failure, involves a gradual loss of kidney function. Kidneys filter wastes and excess fluids from your blood, which are then removed in urine. Advanced chronic kidney disease can cause dangerous levels of fluid, electrolytes and wastes to build up in your body. Initially there are generally no symptoms; later, symptoms may include leg swelling, feeling tired, vomiting, loss of appetite, and confusion. In the early stages of chronic kidney disease, few signs or symptoms are noticeable which cannot make to realize that we have kidney disease until the condition is advanced.

2.2: References

S.No	Paper Title	Author	Journal Name	Publication Year	Description
1	Prediction of kidney disease-A machine learning perspective.	Pankaj Chittora	IEEE Access	2021	The paper explains the use of Machine Learning to act as a tool to detect diseases on time. Seven classifier algorithms have been applied in this research such as Artificial Neural Network, C5.0, Chi-square Automatic interaction detector, logistic regression, linear support vector machine with penalty L1 & with penalty L2 and random tree. The dataset used here consists of 400 instances and 24 attributes. The attributes are labelled in two classes as CKD (chronic kidney disease) and non-CKD. Out of all classifiers used, SVM was found to

					generate consistent results with higher accuracy.
2	XGBoost Model for chronic kidney disease	Adeola Ogunleye, Qing-Guo Wang	IEEE	2019	The paper explains the use of XGBoost Algorithm for detecting Chronic Kidney Disease. XGBoost is an extendible and cutting-edge application of gradient boosting machines and it has proven to push the limits of computing power for boosted trees algorithms. In this algorithm, decision trees are created in sequential form. It was developed for the sole purpose of model performance and computational speed. The model was found to starkly identify CKD apart from other disease with the attributes in the data set. The proposed model was applied and found to have achieved an accuracy, sensitivity and specificity of 1.000, 1.000 and 1.000, respectively.
3	Early detection of kidney disease using advanced machine learning models.	A.Vaishnovi Anuhya, Ayyala Ganesh, Nallabathini Poojitha, Amandeep Singh, Amitha S K, Dr. Dhananjaya. V	International Research Journal of Engineering and Technology (IRJET)	2022	Every year, an increasing number of patients are diagnosed with late stages of renal disease. Chronic Kidney Disease, also known as Chronic Renal Disease, is characterized by abnormal kidney function or a breakdown of renal function that progresses over months or years. Chronic kidney disease is often found during screening of persons who are known to be at risk for

					kidney issues, such as those with high blood pressure or diabetes, and those with a blood family who has chronic kidney disease (CKD). As a result, early prognosis is critical in battling the disease and providing effective therapy. Only early identification and continuous monitoring can avoid serious kidney damage or renal failure. Machine Learning (ML) plays a significant part in the healthcare system, and it may efficiently aid and help with decision support in medical institutions. The primary goals of this research are to design and suggest a in medical institutions. The primary goals of this research are to design and suggest a machine learning method for predicting CKD. Support Vector Machine (SVR), Random Forest (LR), Artificial Neural Network (ANN), and Decision Tree are four master teaching methodologies investigated (DT). The components are built using chronic kidney disease datasets, and the outcomes of these models are compared to select the optimal model for prediction.
4	Machine learning algorithm for early	Zvi Segal, Dan Kalifa, Kira Radinsky,	BMC	2017	End stage renal disease (ESRD) describes the most severe stage of chronic kidney disease

	detection of end-stage renal disease	Bar Ehrenberg, Guy Elad, Gal Maor, Maor Lewis, Muhammad Tibi, Liat Korn & Gideon Koren			(CKD), when patients need dialysis or renal transplant. There is often a delay in recognizing, diagnosing, and treating the various etiologies of CKD. The objective of the present study was to employ machine learning algorithms to develop a prediction model for progression to ESRD based on a large-scale multidimensional database.
5	Chronic kidney disease prediction using machine learning techniques	G. Nandhini, J Aravinth		2021	Early diagnosis and characterization are the important components in determining the treatment of chronic kidney disease (CKD). CKD is an ailment which tends to damage the kidney and affect their effective functioning of excreting waste and balancing body fluids. Some of the complications included are hypertension, anemia (low blood count), mineral bone disorder, poor nutritional health, acid base abnormalities, and neurological complications. Early and error-free detection of CKD can be helpful in averting further deterioration of patient's health. These chronic diseases are prognosticated using various types of data mining classification approaches and machine learning (ML) algorithms. This Prediction is

					performed using Random Forest (RF) Classifier, Logistic Regression (LR) and K-Nearest Neighbor (K-NN) algorithm and Support Vector Machine (SVM). The data used is collected from the UCI Repository with 400 data sets with 25 attributes. This data has been fed into Classification algorithms. The experimental results show that K-NN, LR, SVM hands out. Repository with 400 data sets with 25 attributes. This data has been fed into Classification algorithms. The experimental results show that K-NN, LR, SVM hands out an accuracy of 94%, 98% and 93.75% respectively. The RF classifier gives out a maximum accuracy of 100%.
6	Detection of Chronic Kidney Disease using Machine Learning Algorithms with Least Number of Predictors	Marwa Almasoud, Tomas E Ward	International Journal of Advanced Computer Science and Applications	2018	Chronic kidney disease (CKD) is one of the most critical health problems due to its increasing prevalence. In this paper, we aim to test the ability of machine learning algorithms for the prediction of chronic kidney disease using the smallest subset of features. Several statistical tests have been done to remove redundant features such as the ANOVA test, the Pearson's correlation, and the Cramer's V test.

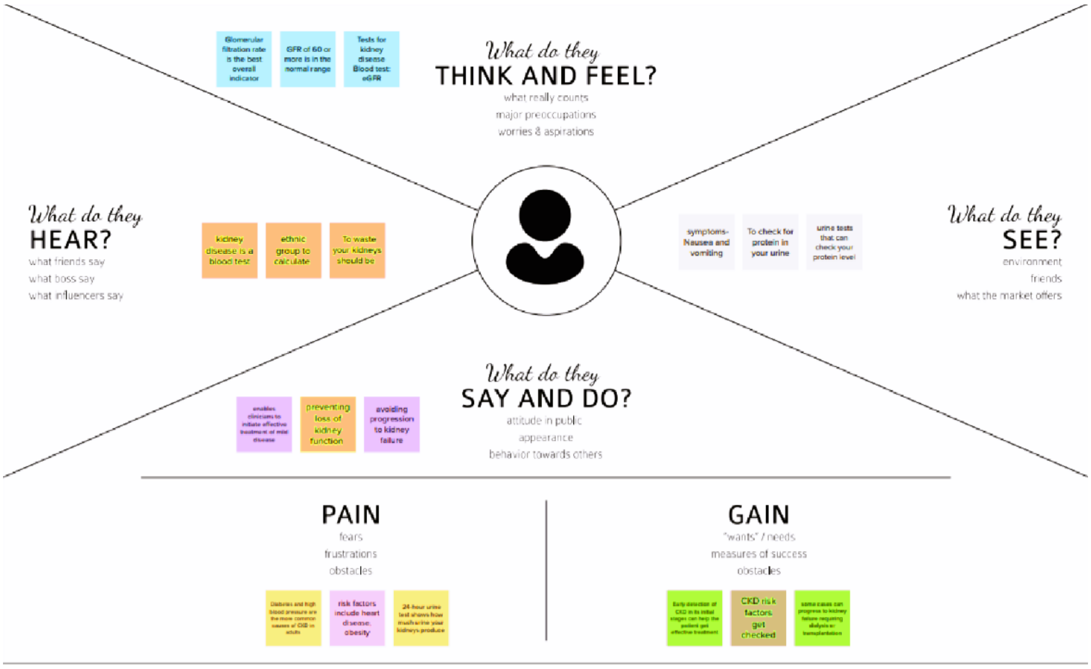
					Logistic regression, support vector machines, random forest, and gradient boosting. earning algorithms for the prediction of chronic kidney disease using the smallest subset of features. Several statistical tests have been done to remove redundant features such as the ANOVA test, the Pearson's correlation, and the Cramer's V test. Logistic regression, support vector machines, random forest, and gradient boosting algorithms have been trained and tested using 10-fold cross-validation. We achieve an accuracy of 99.1 according to F1-measure from Gradient Boosting classifier. Also, we found that hemoglobin has higher importance for both random forest and Gradient boosting in detecting CKD. Finally, our results are among the highest compared to previous studies but with a smaller number of features reached so far.

2.3: Problem Statement Definition

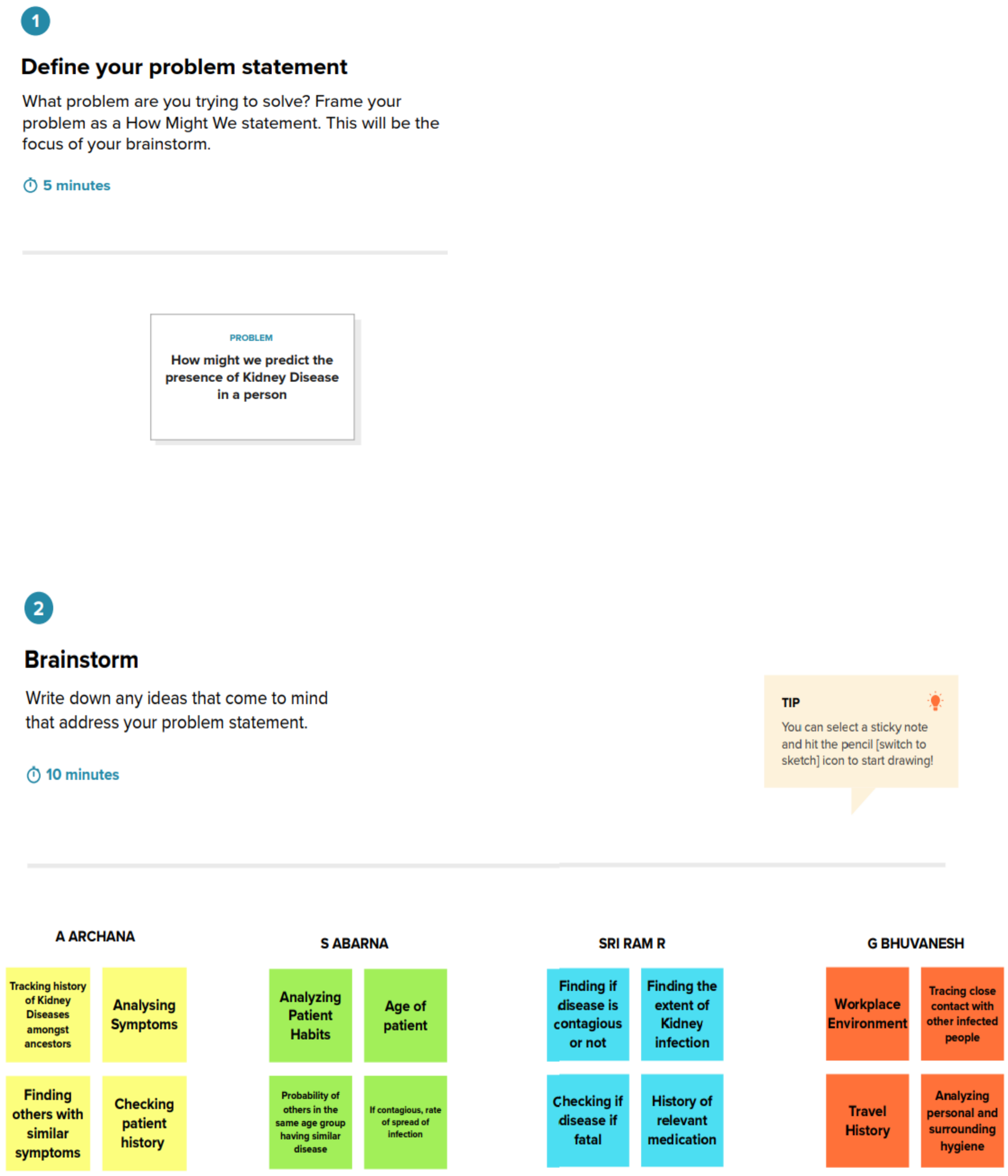
Chronic Kidney Disease (CKD) is a major medical problem and can be cured if treated in the early stages. Usually, people are not aware that medical tests we take for different purposes could contain valuable information concerning kidney diseases. Consequently, attributes of various medical tests are investigated to distinguish which attributes may contain helpful information about the disease. The information says that it helps us to measure the severity of the problem and we make use of such information to build a machine learning model that predicts chronic kidney disease.

3: IDEATION AND PROPOSED SOLUTION

3.1: Empathy Map Canvas


Glomerular filtration rate is the best overall indicator
GFR of 60 or more is in the normal range
Tests for kidney disease Blood test: eGFR
What do they
THINK AND FEEL?
what really counts
major preoccupations
worries & aspirations
What do they
HEAR?
what friends say
what boss say
what influencers say
kidney disease is a blood test
ethnic group to calculate
To waste your kidneys should be
symptoms- Nausea and vomiting
To check for protein in your urine
urine tests that can check your protein level
What do they
SEE?
environment
friends
what the market offers
enables clinicians to initiate effective treatment of mild disease
preventing loss of kidney function
avoiding progression to kidney failure
What do they
SAY AND DO?
attitude in public
appearance
behavior towards others
PAIN
fears
frustrations
obstacles
Diabetes and high blood pressure are the more common causes of CKD in adults
risk factors include heart disease, obesity
24-hour urine test shows how much urine your kidneys produce
GAIN
"wants" / needs
measures of success
obstacles
Early detection of CKD in its initial stages can help the patient get effective treatment
CKD risk factors get checked
some cases can progress to kidney failure requiring dialysis or transplantation

3.2: Ideation and Brainstorming



1

Define your problem statement

What problem are you trying to solve? Frame your problem as a How Might We statement. This will be the focus of your brainstorm.

5 minutes

PROBLEM

How might we predict the presence of Kidney Disease in a person

2

Brainstorm

Write down any ideas that come to mind that address your problem statement.

10 minutes

TIP

You can select a sticky note and hit the pencil [switch to sketch] icon to start drawing!

A ARCHANA

- Tracking history of Kidney Diseases amongst ancestors
- Analysing Symptoms
- Finding others with similar symptoms
- Checking patient history

S ABARNA

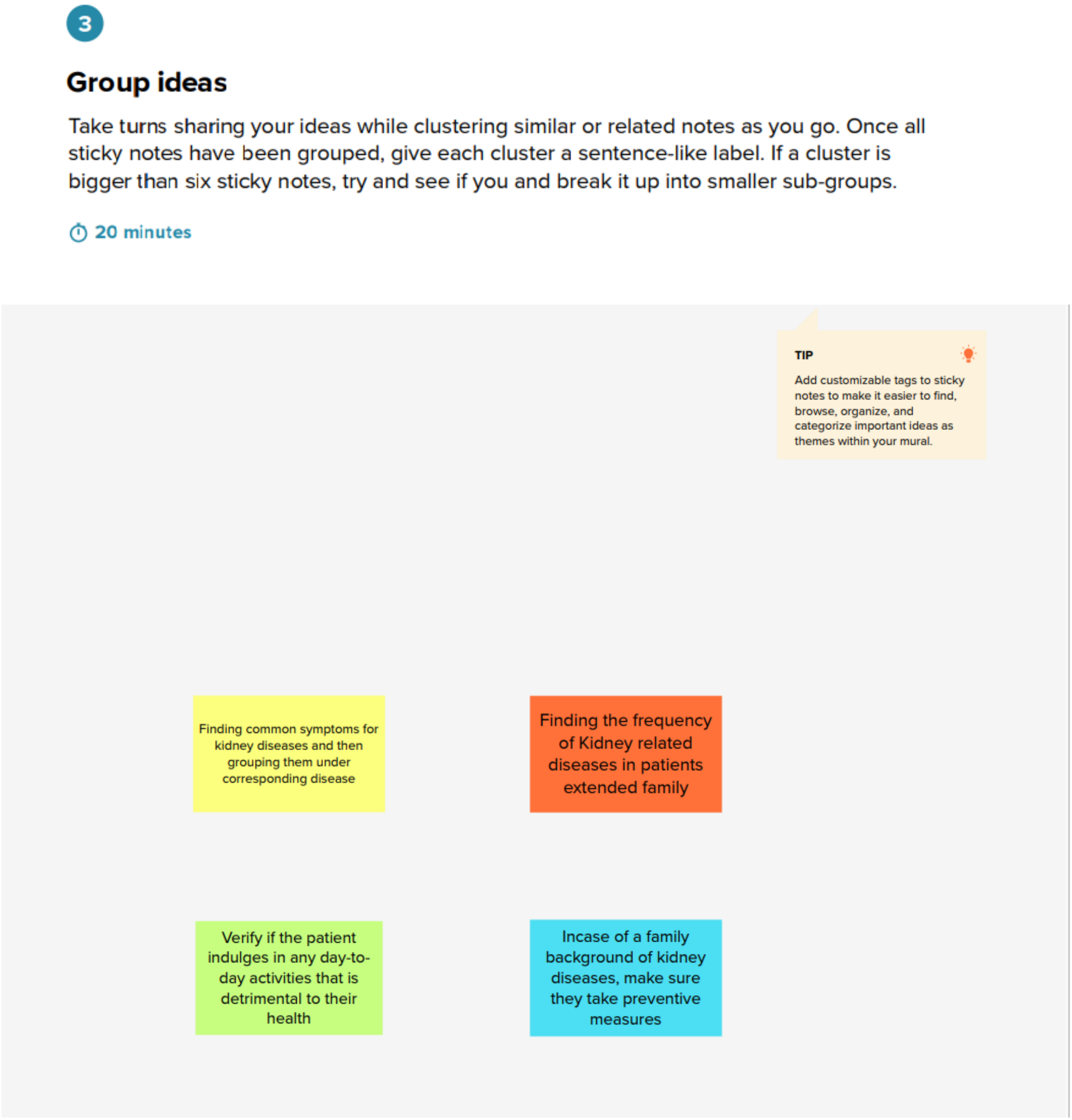
- Analyzing Patient Habits
- Age of patient
- Probability of others in the same age group having similar disease
- If contagious, rate of spread of infection

SRI RAM R

- Finding if disease is contagious or not
- Finding the extent of Kidney infection
- Checking if disease if fatal
- History of relevant medication

G BHUVANESH

- Workplace Environment
- Tracing close contact with other infected people
- Travel History
- Analyzing personal and surrounding hygiene



3

Group ideas

Take turns sharing your ideas while clustering similar or related notes as you go. Once all sticky notes have been grouped, give each cluster a sentence-like label. If a cluster is bigger than six sticky notes, try and see if you and break it up into smaller sub-groups.

⏱ 20 minutes

TIP

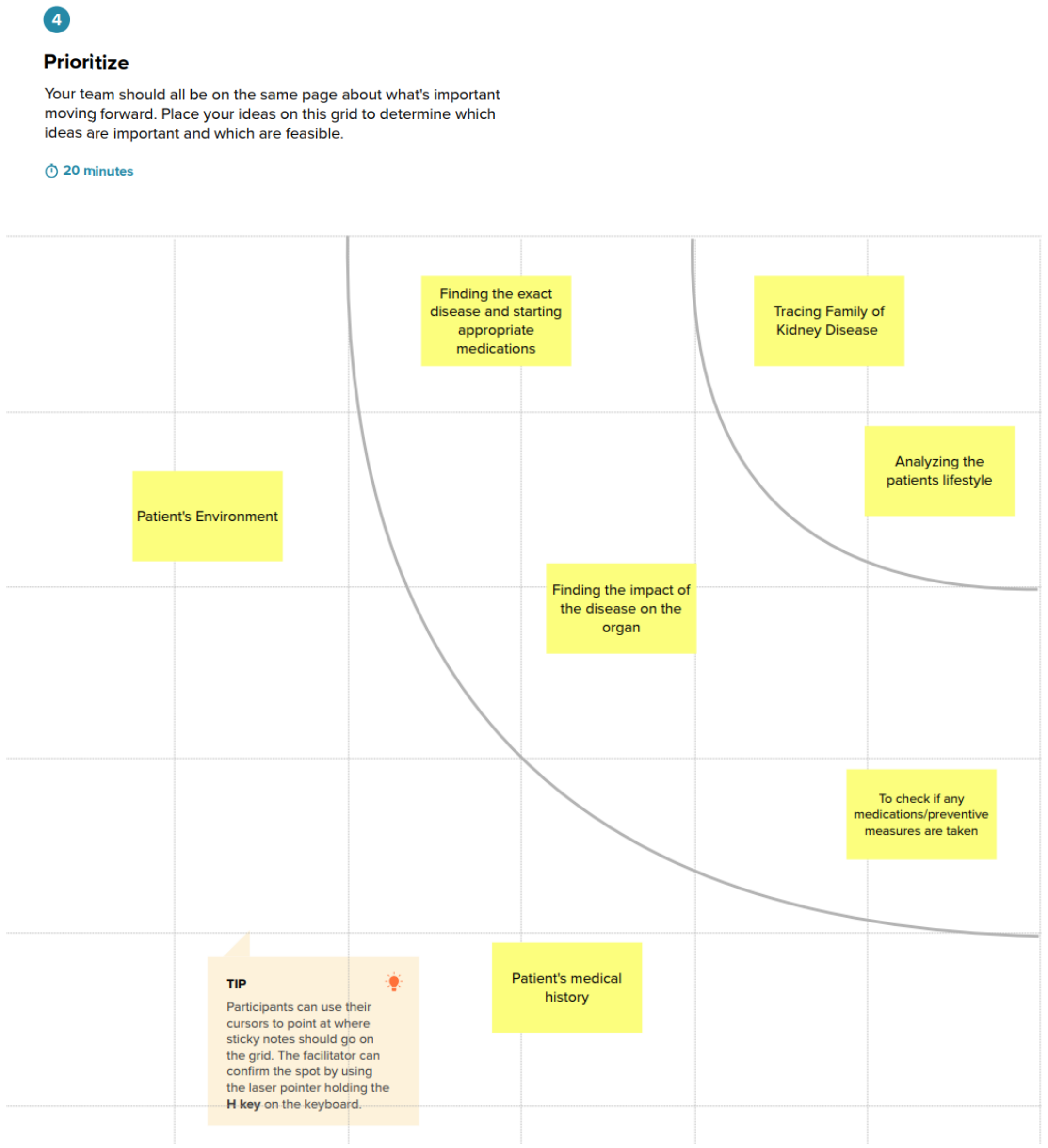
Add customizable tags to sticky notes to make it easier to find, browse, organize, and categorize important ideas as themes within your mural.

Finding common symptoms for kidney diseases and then grouping them under corresponding disease

Finding the frequency of Kidney related diseases in patients extended family

Verify if the patient indulges in any day-to-day activities that is detrimental to their health

Incase of a family background of kidney diseases, make sure they take preventive measures



4

Prioritize

Your team should all be on the same page about what's important moving forward. Place your ideas on this grid to determine which ideas are important and which are feasible.

⏱ 20 minutes

Finding the exact disease and starting appropriate medications

Tracing Family of Kidney Disease

Analyzing the patients lifestyle

Patient's Environment

Finding the impact of the disease on the organ

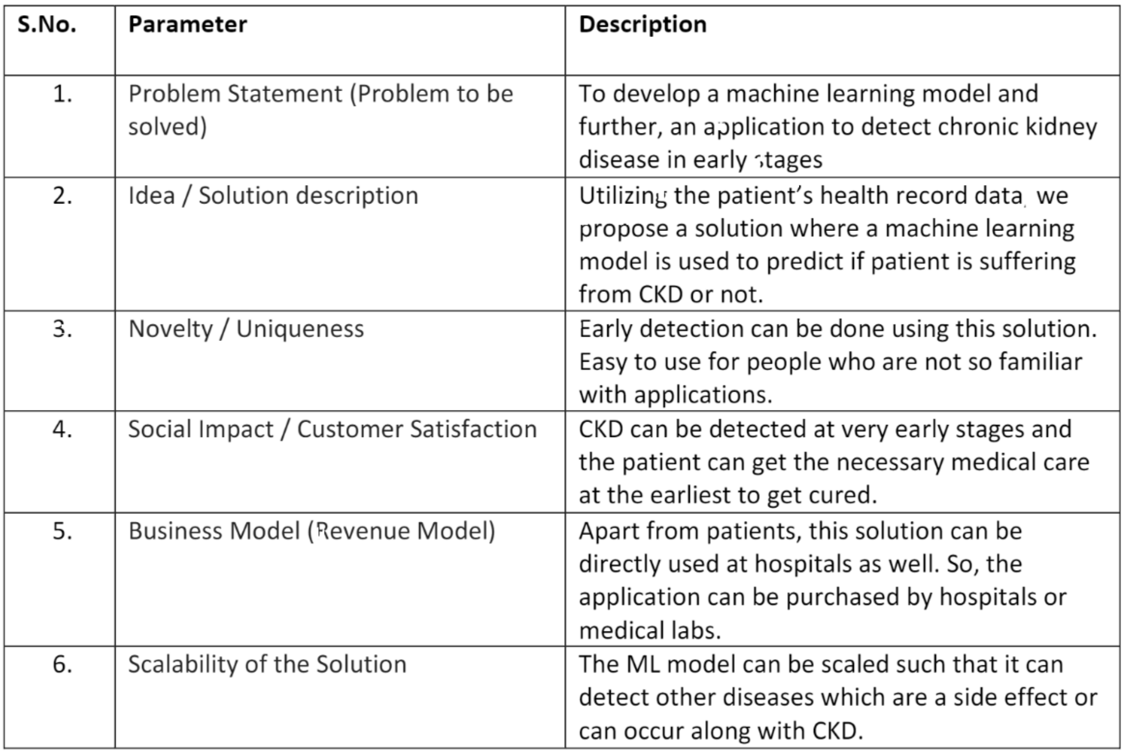
To check if any medications/preventive measures are taken

Patient's medical history

TIP

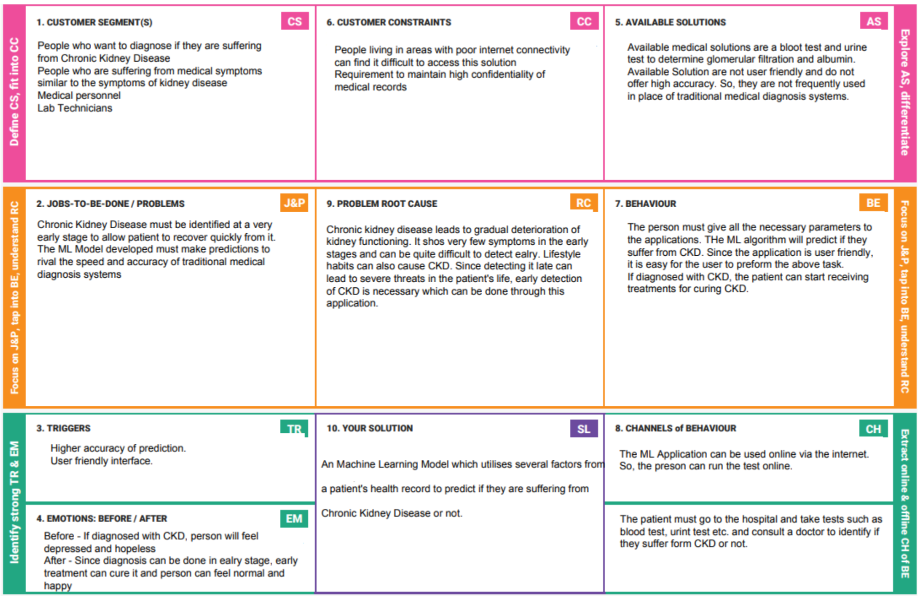
Participants can use their cursors to point at where sticky notes should go on the grid. The facilitator can confirm the spot by using the laser pointer holding the **H key** on the keyboard.

3.3: Proposed Solution



S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	To develop a machine learning model and further, an application to detect chronic kidney disease in early stages
2.	Idea / Solution description	Utilizing the patient's health record data, we propose a solution where a machine learning model is used to predict if patient is suffering from CKD or not.
3.	Novelty / Uniqueness	Early detection can be done using this solution. Easy to use for people who are not so familiar with applications.
4.	Social Impact / Customer Satisfaction	CKD can be detected at very early stages and the patient can get the necessary medical care at the earliest to get cured.
5.	Business Model (Revenue Model)	Apart from patients, this solution can be directly used at hospitals as well. So, the application can be purchased by hospitals or medical labs.
6.	Scalability of the Solution	The ML model can be scaled such that it can detect other diseases which are a side effect or can occur along with CKD.

3.4: Problem Solution Fit



Define CS, fit into CC
Explore AS, differentiate
Focus on J&P, tap into BE, understand RC
Focus on J&P, tap into BE, understand RC
Identify strong TR & EM
Extract online & offline CH of BE
CS
CC
AS
J&P
RC
BE
TR
SL
CH
EM

1. CUSTOMER SEGMENT(S)

People who want to diagnose if they are suffering from Chronic Kidney Disease
People who are suffering from medical symptoms similar to the symptoms of kidney disease
Medical personnel
Lab Technicians

6. CUSTOMER CONSTRAINTS

People living in areas with poor internet connectivity can find it difficult to access this solution
Requirement to maintain high confidentiality of medical records

5. AVAILABLE SOLUTIONS

Available medical solutions are a bloot test and urine test to determine glomerular filtration and albumin. Available Solution are not user friendly and do not offer high accuracy. So, they are not frequently used in place of traditional medical diagnosis systems.

2. JOBS-TO-BE-DONE / PROBLEMS

Chronic Kidney Disease must be identified at a very early stage to allow patient to recover quickly from it. The ML Model developed must make predictions to rival the speed and accuracy of traditional medical diagnosis systems

9. PROBLEM ROOT CAUSE

Chronic kidney disease leads to gradual deterioration of kidney functioning. It shos very few symptoms in the early stages and can be quite difficult to detect ealry. Lifestyle habits can also cause CKD. Since detecting it late can lead to severe threats in the patient's life, early detection of CKD is necessary which can be done through this application.

7. BEHAVIOUR

The person must give all the necessary parameters to the applications. THe ML algorithm will predict if they suffer from CKD. Since the application is user friendly, it is easy for the user to preform the above task.
If diagnosed with CKD, the patient can start receiving treatments for curing CKD.

3. TRIGGERS

Higher accuracy of prediction.
User friendly interface.

4. EMOTIONS: BEFORE / AFTER

Before - If diagnosed with CKD, person will feel depressed and hopeless
After - Since diagnosis can be done in ealry stage, early treatment can cure it and person can feel normal and happy

10. YOUR SOLUTION

An Machine Learning Model which utilises several factors from a patient's health record to predict if they are suffering from Chronic Kidney Disease or not.

8. CHANNELS of BEHAVIOUR

The ML Application can be used online via the internet. So, the preson can run the test online.

The patient must go to the hospital and take tests such as blood test, urint test etc. and consult a doctor to identify if they suffer form CKD or not.

4: REQUIREMENT ANALYSIS

4.1: Functional Requirements

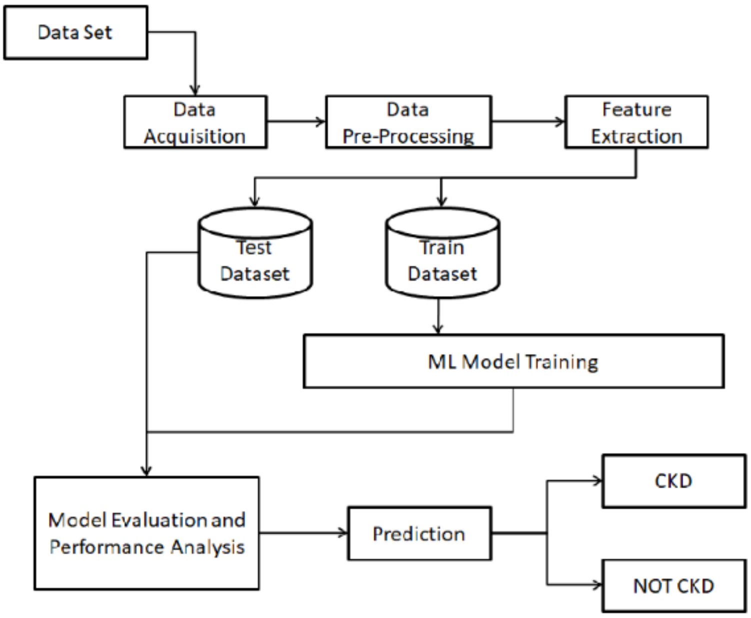
FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Registration	Registration through Google Form Registration through Hospitals Registration through Medical Camps
FR-2	User Confirmation	Confirmation via Message Confirmation via Mail
FR-3	User Requirements	Knowledge in application to submit required data Inferring the output and acting accordingly
FR-4	User Infrastructure	A system to support ML data modelling A Suitable GPU and CPU
FR-5	User Network	Network infrastructure to connect the User Interface and the Data Storage
FR-6	User Cost	No Expenditure will be necessary

4.2: Non – Functional Requirements

FR No.	Non-Functional Requirement	Description
NFR-1	Usability	It can be used for various Datasets and trained for specific models and prediction methods.
NFR-2	Security	Secure Cloud Storage can be used to store data
NFR-3	Reliability	The Reliability of the predicted outcomes should be sufficient
NFR-4	Performance	The Computer's Hardware will play a major role in determining the performance of the System
NFR-5	Availability	It is available as a software package.
NFR-6	Scalability	It can be scaled up and interconnected with other applications to provide various medical services.

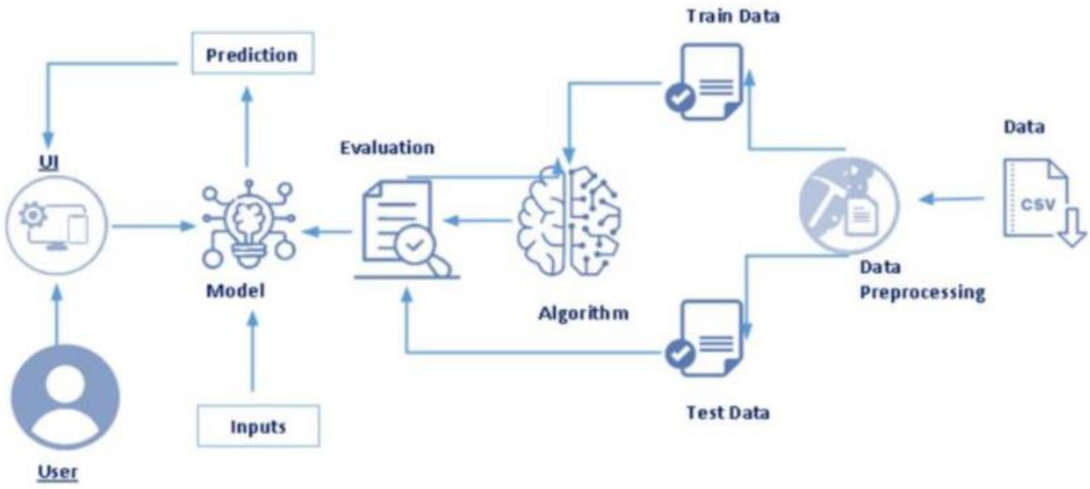
5: PROJECT DESIGN

5.1: Data Flow Diagrams



1. The Collected Data is pre-processes and is fed as input to the ML model.
2. Individual Feature Extraction is carried out before it is fed as input to the ML model.
3. The Dataset is divided into Test and Train Datasets.
4. The Model is trained using the Train Dataset.
5. The Model is then analysed using the Test Dataset and the predictions are carried out.
6. The error in prediction is calculated and used as feedback to train the model.
7. The model accuracy increases over the course.

5.2: Solution and Technical Architecture



5.3: User Stories

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria
Administrator	Data Collection	USN-1	As an Admin, I collect the appropriate dataset for predicting the chronic kidney disease.	I can submit images to the dataset under the guidelines and requirements mentioned
		USN-2	As an Admin, I split the dataset as train and test datasets.	From the Dataset images are split into Train and Test Data by an 80-20 ratio
	Model Building	USN-3	As an Admin, I split the Model into Training and Testing from the overall dataset.	I can feed the Separate Datasets to the ML Model for Testing and Training
Customer		USN-4	As a User, I submit my blood pressure and sugar level and other required information	I can view the required details that are to be used for prediction for specific individuals
Administrator	Training and Testing	USN-5	As an Admin, I Train the Model using	I can train the model for multiple epochs,

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria
			Regression algorithm and Test the Performance of the model.	improving efficiency of the prediction
	Implementation of the Application	USN-6	As an Admin, I work on predicting the spread of chronic kidney diseaseand predict the possibility of kidney failure	The Final System is deployed and the prediction is made
Customer		USN-7	As a User, I collect information and inferences from the system and analyse my lifestyle	I can view the out-come of the prediction and the analysis of the predicted outcome
Customer Care Executive		USN -8	As a Customer Care Executive, I shall attend the calls and guide the user.	I must clearly know the details of the model and the UI
	Maintenance	USN-9	I must be sure that the ML Model is up to date and is working in proper condition	I can check the outcome and update the model when required

6: PROJECT PLANNING AND SCHEDULING

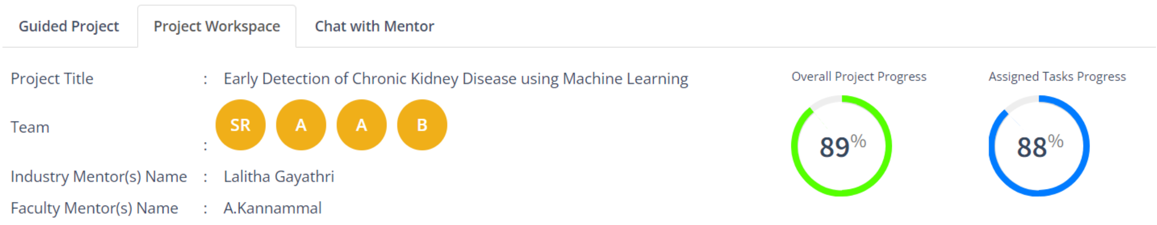
6.1: Sprint Planning and Estimation

Sprint	Milestone
Sprint 1	1. User Registers into the application 2. User inputs data of past illness, habits and other information relating to chronic kidney diseases 3. User submits the details regarding Blood Sugar Levels, Blood Pressure and other required information relating to his current health conditions
Sprint 2	1. User can access the prediction model 2. After collecting the required information from the User, the ML model can now be accessed and used to predicted the presence of kidney diseases
Sprint 3	1. Application stores the predictions, that can be used for future analysis. 2. The data stored has to be maintained securely.
Sprint 4	1. Administrator should properly maintain the data and the prediction algorithm should be updated whenever required.

6.2 Sprint Delivery Schedule

Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint-1	20	6 Days	24 Oct 2022	29 Oct 2022	20	29 Oct 2022
Sprint-2	20	6 Days	31 Oct 2022	05 Nov 2022	20	04 Nov 2022
Sprint-3	20	6 Days	07 Nov 2022	12 Nov 2022	20	09 Nov 2022
Sprint-4	20	6 Days	14 Nov 2022	19 Nov 2022	20	16 Nov 2022

6.3: Reports from JIRA



Project Title : Early Detection of Chronic Kidney Disease using Machine Learning

Team : SR A A B

Industry Mentor(s) Name : Lalitha Gayathri

Faculty Mentor(s) Name : A.Kannammal

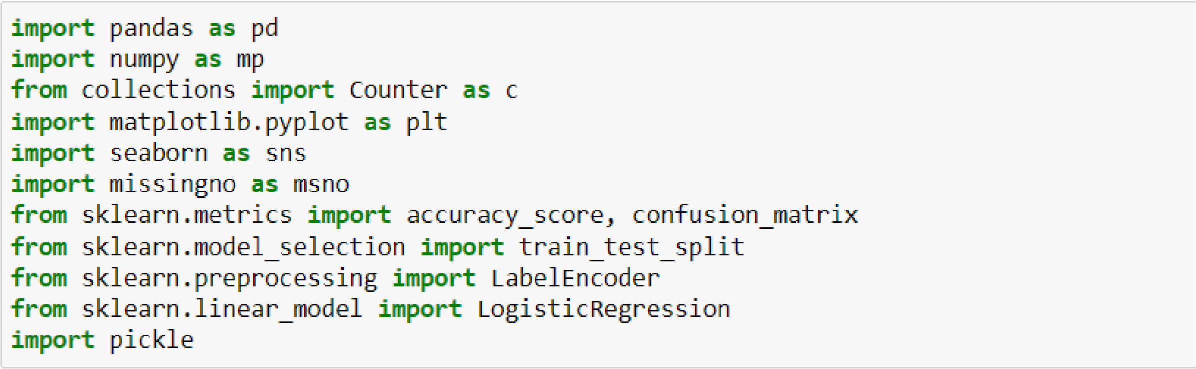
Overall Project Progress 89%

Assigned Tasks Progress 88%

7. CODING AND SOLUTION

7.1 FEATURE 1 - DATA PREPROCESSING

Importing the necessary libraries for processing the dataset and building the model

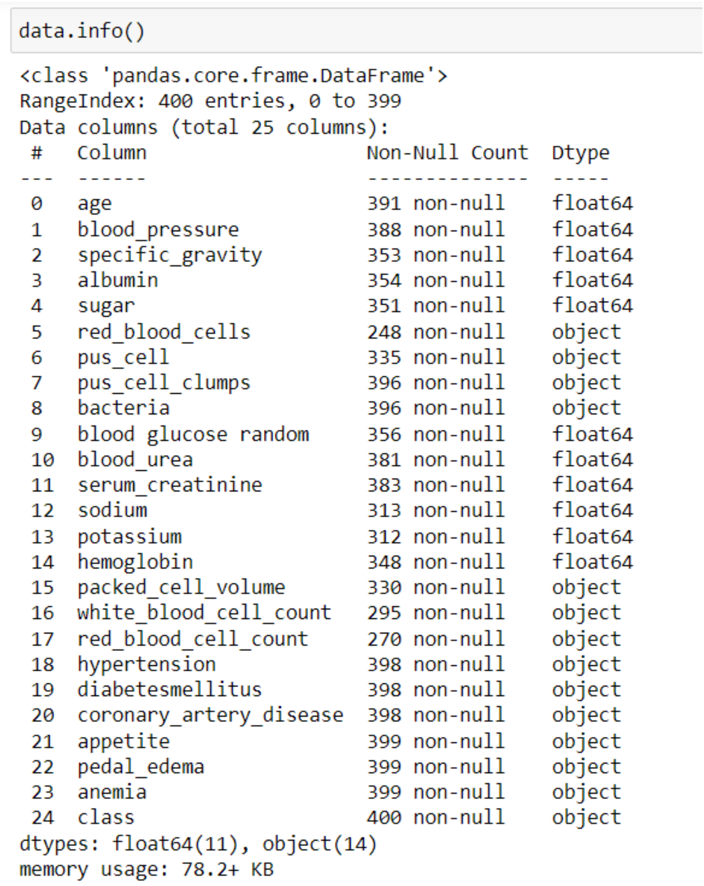


```
import pandas as pd
import numpy as mp
from collections import Counter as c
import matplotlib.pyplot as plt
import seaborn as sns
import missingno as msno
from sklearn.metrics import accuracy_score, confusion_matrix
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.linear_model import LogisticRegression
import pickle
```

The dataset is loaded into the jupyter notebook file.

```
data=pd.read_csv(r"C:\Users\archa\Desktop\IBM Datasets\chronickidneydisease.csv")
```

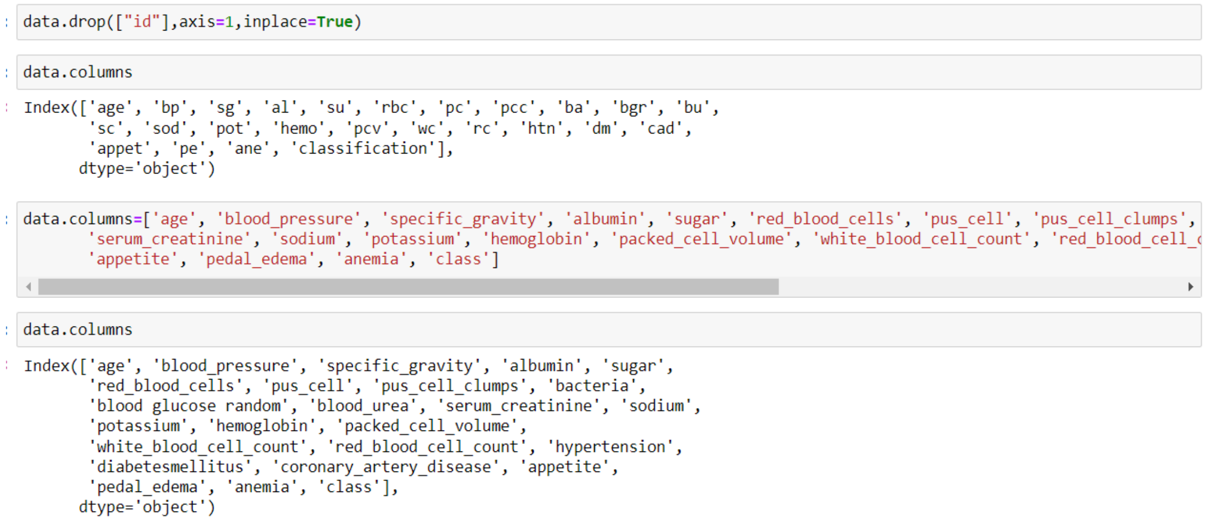
Information of the dataset is observed.



```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 400 entries, 0 to 399
Data columns (total 25 columns):
 #   Column                   Non-Null Count  Dtype  
---  ------                   --------------  -----  
 0   age                      391 non-null    float64
 1   blood_pressure           388 non-null    float64
 2   specific_gravity         353 non-null    float64
 3   albumin                  354 non-null    float64
 4   sugar                    351 non-null    float64
 5   red_blood_cells          248 non-null    object 
 6   pus_cell                 335 non-null    object 
 7   pus_cell_clumps          396 non-null    object 
 8   bacteria                 396 non-null    object 
 9   blood glucose random     356 non-null    float64
 10  blood_urea               381 non-null    float64
 11  serum_creatinine         383 non-null    float64
 12  sodium                   313 non-null    float64
 13  potassium                312 non-null    float64
 14  hemoglobin               348 non-null    float64
 15  packed_cell_volume       330 non-null    object 
 16  white_blood_cell_count   295 non-null    object 
 17  red_blood_cell_count     270 non-null    object 
 18  hypertension             398 non-null    object 
 19  diabetesmellitus         398 non-null    object 
 20  coronary_artery_disease  398 non-null    object 
 21  appetite                 399 non-null    object 
 22  pedal_edema              399 non-null    object 
 23  anemia                   399 non-null    object 
 24  class                    400 non-null    object 
dtypes: float64(11), object(14)
memory usage: 78.2+ KB
```

The id column in the dataset is dropped and the columns are renamed for easy understanding.



```
data.drop(["id"],axis=1,inplace=True)
```

```
data.columns
```

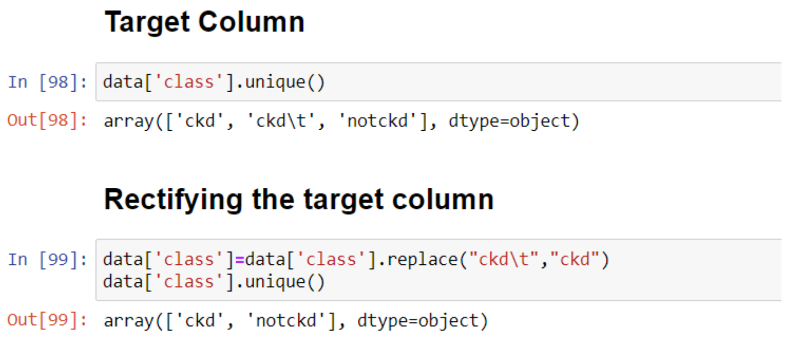
```
Index(['age', 'bp', 'sg', 'al', 'su', 'rbc', 'pc', 'pcc', 'ba', 'bgr', 'bu',
       'sc', 'sod', 'pot', 'hemo', 'pcv', 'wc', 'rc', 'htn', 'dm', 'cad',
       'appet', 'pe', 'ane', 'classification'],
      dtype='object')
```

```
data.columns=['age', 'blood_pressure', 'specific_gravity', 'albumin', 'sugar', 'red_blood_cells', 'pus_cell', 'pus_cell_clumps',
        'serum_creatinine', 'sodium', 'potassium', 'hemoglobin', 'packed_cell_volume', 'white_blood_cell_count', 'red_blood_cell_c
        'appetite', 'pedal_edema', 'anemia', 'class']
```

```
data.columns
```

```
Index(['age', 'blood_pressure', 'specific_gravity', 'albumin', 'sugar',
       'red_blood_cells', 'pus_cell', 'pus_cell_clumps', 'bacteria',
       'blood glucose random', 'blood_urea', 'serum_creatinine', 'sodium',
       'potassium', 'hemoglobin', 'packed_cell_volume',
       'white_blood_cell_count', 'red_blood_cell_count', 'hypertension',
       'diabetesmellitus', 'coronary_artery_disease', 'appetite',
       'pedal_edema', 'anemia', 'class'],
      dtype='object')
```

The class, which is the output column is observed. The classes are changed so that there can be only two output – ‘ckd’ or ‘notckd’.



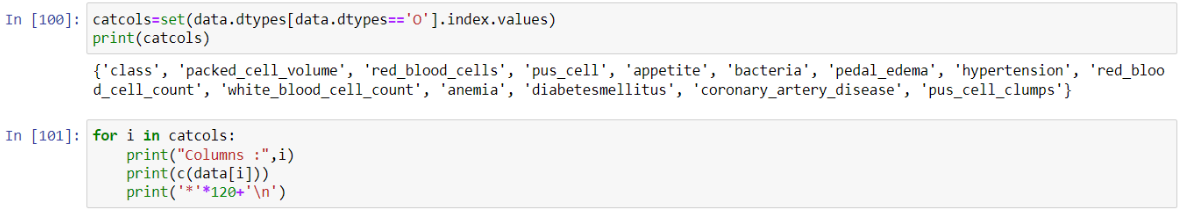
Target Column

```
In [98]: data['class'].unique()

Out[98]: array(['ckd', 'ckd\t', 'notckd'], dtype=object)
```

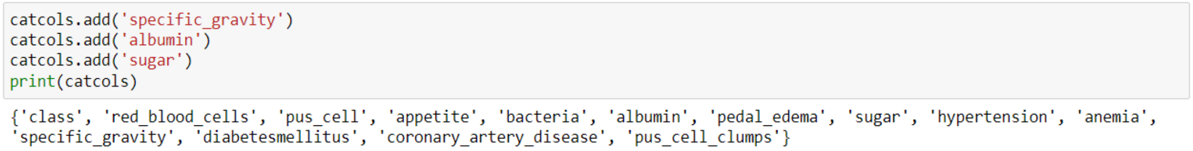
Rectifying the target column

```
In [99]: data['class']=data['class'].replace("ckd\t","ckd")
         data['class'].unique()

Out[99]: array(['ckd', 'notckd'], dtype=object)
```

The categorical columns are analysed. The fields which are not categorical are removed from the set 'catcols' and the fields(columns) which are categorical are added to the set.

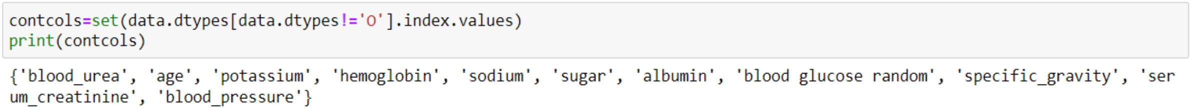


```
In [100]: catcols=set(data.dtypes[data.dtypes=='O'].index.values)
          print(catcols)

          {'class', 'packed_cell_volume', 'red_blood_cells', 'pus_cell', 'appetite', 'bacteria', 'pedal_edema', 'hypertension', 'red_bloo
          d_cell_count', 'white_blood_cell_count', 'anemia', 'diabetesmellitus', 'coronary_artery_disease', 'pus_cell_clumps'}

In [101]: for i in catcols:
              print("Columns :",i)
              print(c(data[i]))
              print('*'*120+'\n')
```

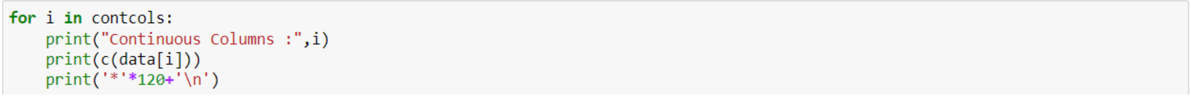


```
catcols.add('specific_gravity')
catcols.add('albumin')
catcols.add('sugar')
print(catcols)

{'class', 'red_blood_cells', 'pus_cell', 'appetite', 'bacteria', 'albumin', 'pedal_edema', 'sugar', 'hypertension', 'anemia',
'specific_gravity', 'diabetesmellitus', 'coronary_artery_disease', 'pus_cell_clumps'}
```

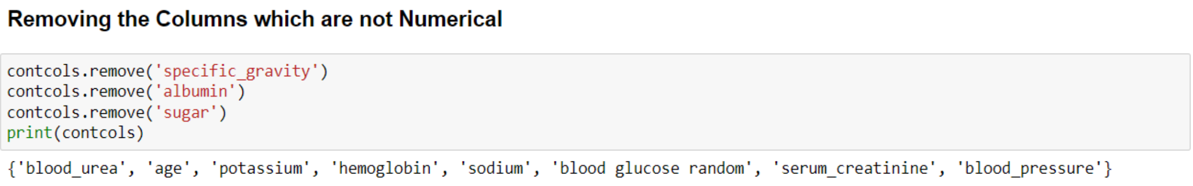
Similarly, this is done for continuous columns. The continuous columns are analysed. The fields which are not continuous are removed from the set 'contcols' and the fields(columns) which are continuous are added to the set.



```
contcols=set(data.dtypes[data.dtypes!='O'].index.values)
print(contcols)

{'blood_urea', 'age', 'potassium', 'hemoglobin', 'sodium', 'sugar', 'albumin', 'blood glucose random', 'specific_gravity', 'ser
um_creatinine', 'blood_pressure'}
```

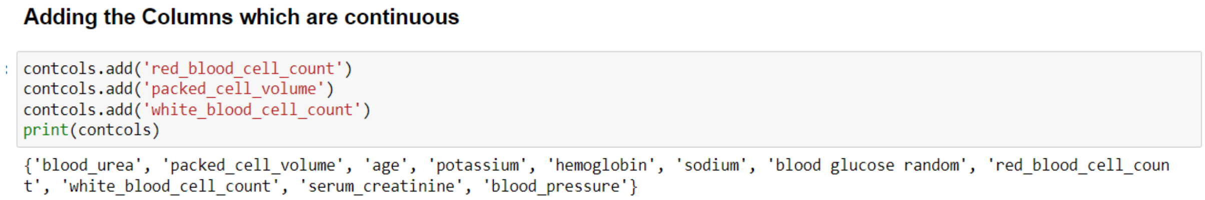


```
for i in contcols:
    print("Continuous Columns :",i)
    print(c(data[i]))
    print('*'*120+'\n')
```



Removing the Columns which are not Numerical

```
contcols.remove('specific_gravity')
contcols.remove('albumin')
contcols.remove('sugar')
print(contcols)

{'blood_urea', 'age', 'potassium', 'hemoglobin', 'sodium', 'blood glucose random', 'serum_creatinine', 'blood_pressure'}
```

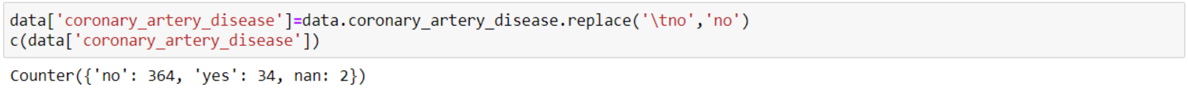


Adding the Columns which are continuous

```
contcols.add('red_blood_cell_count')
contcols.add('packed_cell_volume')
contcols.add('white_blood_cell_count')
print(contcols)
```

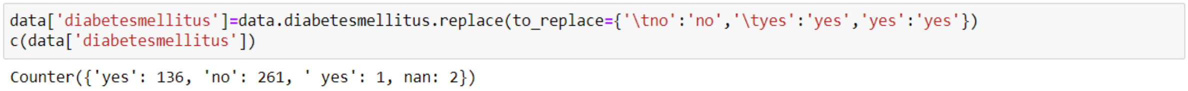
```
{'blood_urea', 'packed_cell_volume', 'age', 'potassium', 'hemoglobin', 'sodium', 'blood glucose random', 'red_blood_cell_coun
t', 'white_blood_cell_count', 'serum_creatinine', 'blood_pressure'}
```

The columns coronary artery disease and diabetes mellitus have ambiguous values. So this is rectified by either assigning the values to yes or no.



```
data['coronary_artery_disease']=data.coronary_artery_disease.replace('\tno','no')
c(data['coronary_artery_disease'])
```

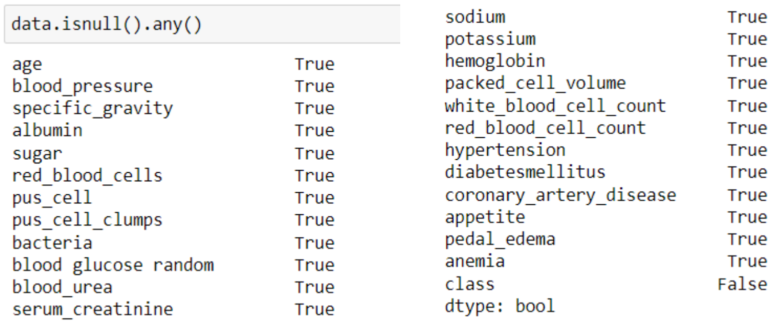
```
Counter({'no': 364, 'yes': 34, nan: 2})
```



```
data['diabetesmellitus']=data.diabetesmellitus.replace(to_replace={'\tno':'no','\tyes':'yes','yes':'yes'})
c(data['diabetesmellitus'])
```

```
Counter({'yes': 136, 'no': 261, ' yes': 1, nan: 2})
```

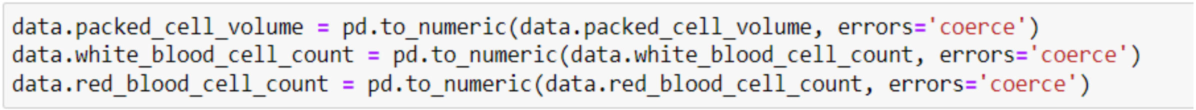
The nest step is to verify if there are any null values present in the dataset.



```
data.isnull().any()
```

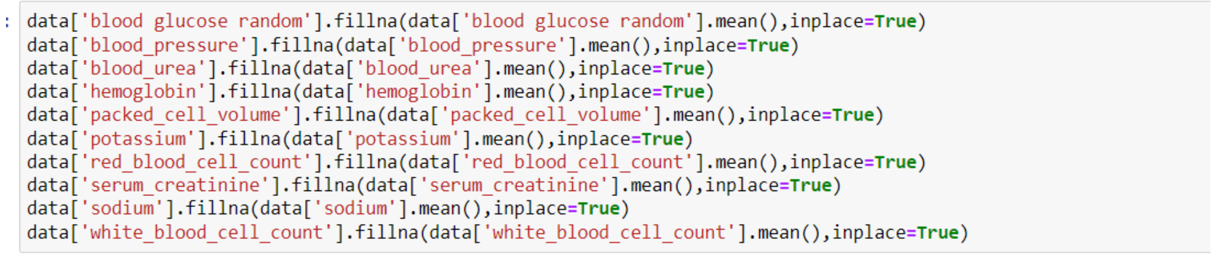
```
age                        True
blood_pressure             True
specific_gravity           True
albumin                    True
sugar                      True
red_blood_cells            True
pus_cell                   True
pus_cell_clumps            True
bacteria                   True
blood glucose random       True
blood_urea                 True
serum_creatinine           True
sodium                     True
potassium                  True
hemoglobin                 True
packed_cell_volume         True
white_blood_cell_count     True
red_blood_cell_count       True
hypertension               True
diabetesmellitus           True
coronary_artery_disease    True
appetite                   True
pedal_edema                True
anemia                     True
class                     False
dtype: bool
```

The null values must be removed. But before that some continuous data are represented as a string. These are converted to numeric using to_numeric() function.

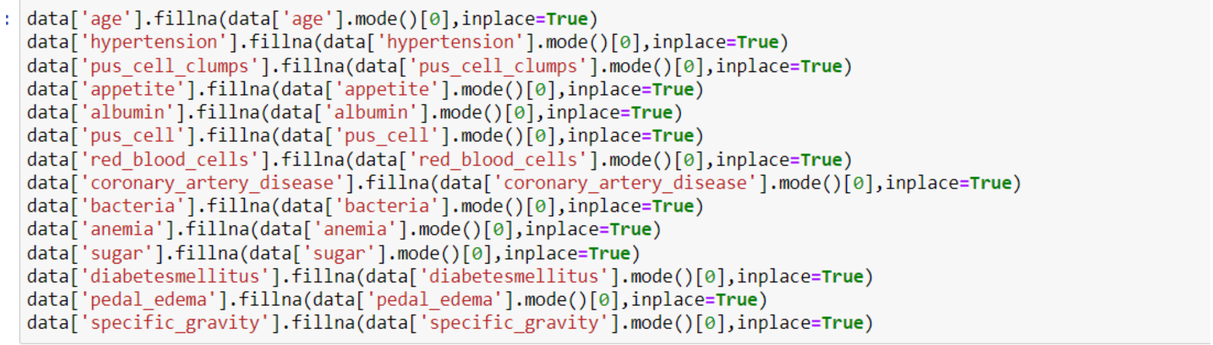


```
data.packed_cell_volume = pd.to_numeric(data.packed_cell_volume, errors='coerce')
data.white_blood_cell_count = pd.to_numeric(data.white_blood_cell_count, errors='coerce')
data.red_blood_cell_count = pd.to_numeric(data.red_blood_cell_count, errors='coerce')
```

Now, the null values must be removed. This can be done by replacing the null values using either mean, median or mode using fillna() function.

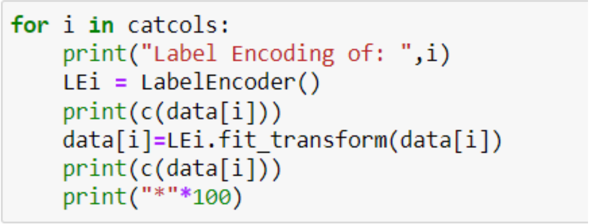


```
data['blood glucose random'].fillna(data['blood glucose random'].mean(),inplace=True)
data['blood_pressure'].fillna(data['blood_pressure'].mean(),inplace=True)
data['blood_urea'].fillna(data['blood_urea'].mean(),inplace=True)
data['hemoglobin'].fillna(data['hemoglobin'].mean(),inplace=True)
data['packed_cell_volume'].fillna(data['packed_cell_volume'].mean(),inplace=True)
data['potassium'].fillna(data['potassium'].mean(),inplace=True)
data['red_blood_cell_count'].fillna(data['red_blood_cell_count'].mean(),inplace=True)
data['serum_creatinine'].fillna(data['serum_creatinine'].mean(),inplace=True)
data['sodium'].fillna(data['sodium'].mean(),inplace=True)
data['white_blood_cell_count'].fillna(data['white_blood_cell_count'].mean(),inplace=True)
```



```
data['age'].fillna(data['age'].mode()[0],inplace=True)
data['hypertension'].fillna(data['hypertension'].mode()[0],inplace=True)
data['pus_cell_clumps'].fillna(data['pus_cell_clumps'].mode()[0],inplace=True)
data['appetite'].fillna(data['appetite'].mode()[0],inplace=True)
data['albumin'].fillna(data['albumin'].mode()[0],inplace=True)
data['pus_cell'].fillna(data['pus_cell'].mode()[0],inplace=True)
data['red_blood_cells'].fillna(data['red_blood_cells'].mode()[0],inplace=True)
data['coronary_artery_disease'].fillna(data['coronary_artery_disease'].mode()[0],inplace=True)
data['bacteria'].fillna(data['bacteria'].mode()[0],inplace=True)
data['anemia'].fillna(data['anemia'].mode()[0],inplace=True)
data['sugar'].fillna(data['sugar'].mode()[0],inplace=True)
data['diabetesmellitus'].fillna(data['diabetesmellitus'].mode()[0],inplace=True)
data['pedal_edema'].fillna(data['pedal_edema'].mode()[0],inplace=True)
data['specific_gravity'].fillna(data['specific_gravity'].mode()[0],inplace=True)
```

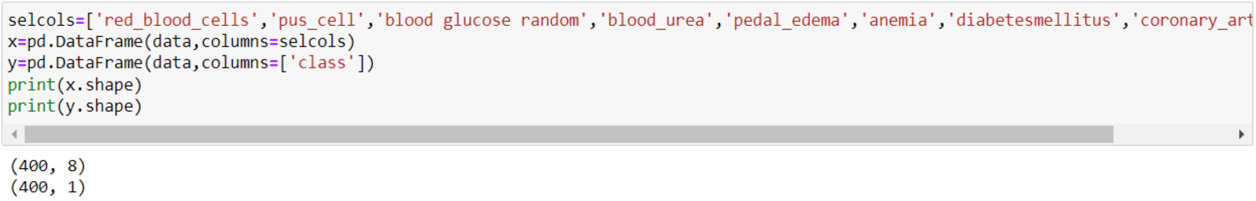
Both numerical and categorical values are present in the dataset. Only numerical can be processed by the computer. So, the categorical values must be encoded. This is done Label encoding in the sklearn library. Here, each of the integer is assigned a unique integer based on alphabetical ordering.



```
for i in catcols:
    print("Label Encoding of: ",i)
    LEi = LabelEncoder()
    print(c(data[i]))
    data[i]=LEi.fit_transform(data[i])
    print(c(data[i]))
    print("*"*100)
```

7.2 – MODEL BUILDING

In machine learning, the concept of the dependent variable (y) and independent variables(x) is important to understand. Here, the dependent variable is nothing but output in the dataset and the independent variable is all inputs in the dataset. We can denote any symbol (alphabets). In our dataset, we can say that class is the dependent variable and all other columns are independent. But in order to select the independent columns, we will be selecting only those columns which are highly correlated and some value to our dependent column.



```
selcols=['red_blood_cells','pus_cell','blood glucose random','blood_urea','pedal_edema','anemia','diabetesmellitus','coronary_art
x=pd.DataFrame(data,columns=selcols)
y=pd.DataFrame(data,columns=['class'])
print(x.shape)
print(y.shape)
```

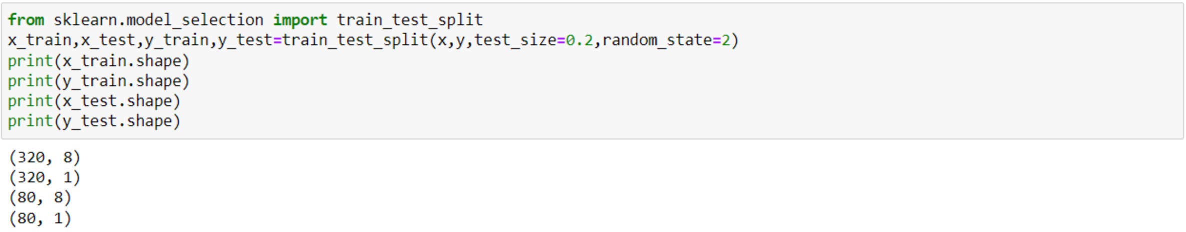
```
(400, 8)
(400, 1)
```

The next step is to split the data set into training data and testing data. This is done using train_test_split. The train-test split is a technique for evaluating the performance of a machine learning algorithm.

Train Dataset: Used to fit the machine learning model.

Test Dataset: Used to evaluate the fit machine learning model.

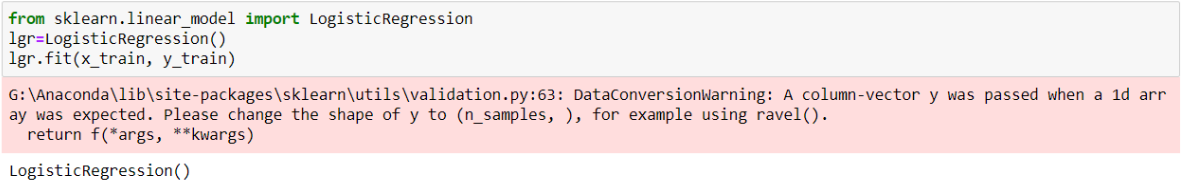
About 80% of the data is used for training the machine learning model and 20% is used for testing.



```
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=2)
print(x_train.shape)
print(y_train.shape)
print(x_test.shape)
print(y_test.shape)
```

```
(320, 8)
(320, 1)
(80, 8)
(80, 1)
```

The dataset is trained using the Logistic Regression Model as it was found to have more accuracy over other models such as Decision Tree Classifier.

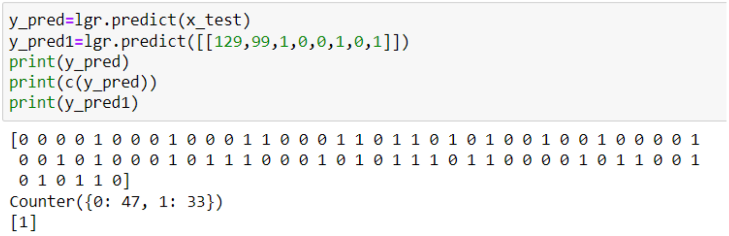


```
from sklearn.linear_model import LogisticRegression
lgr=LogisticRegression()
lgr.fit(x_train, y_train)
```

```
G:\Anaconda\lib\site-packages\sklearn\utils\validation.py:63: DataConversionWarning: A column-vector y was passed when a 1d arr
ay was expected. Please change the shape of y to (n_samples, ), for example using ravel().
  return f(*args, **kwargs)
```

```
LogisticRegression()
```

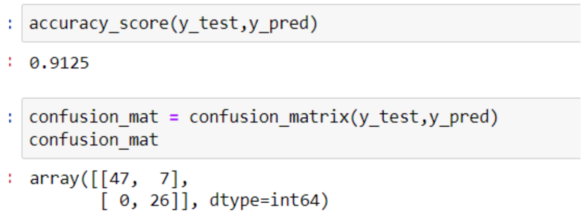
To the trained model, both the test input and an external input is given and the output is verified.



```
y_pred=lgr.predict(x_test)
y_pred1=lgr.predict([[129,99,1,0,0,1,0,1]])
print(y_pred)
print(c(y_pred))
print(y_pred1)
```

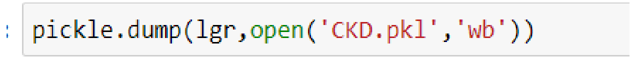
```
[0 0 0 0 1 0 0 0 1 0 0 0 1 1 0 0 0 1 1 0 1 1 0 1 0 1 0 0 1 0 0 1 0 0 0 0 1
 0 0 1 0 1 0 0 0 1 0 1 1 1 0 0 0 1 0 1 0 1 1 1 0 1 1 0 0 0 0 1 0 1 1 0 0 1
 0 1 0 1 1 0]
Counter({0: 47, 1: 33})
[1]
```

For the trained ML model. The accuracy and confusion matrix is plotted.



```
: accuracy_score(y_test,y_pred)
: 0.9125

: confusion_mat = confusion_matrix(y_test,y_pred)
  confusion_mat
: array([[47,  7],
         [ 0, 26]], dtype=int64)
```

Finally, the model is saved as a pickle file.

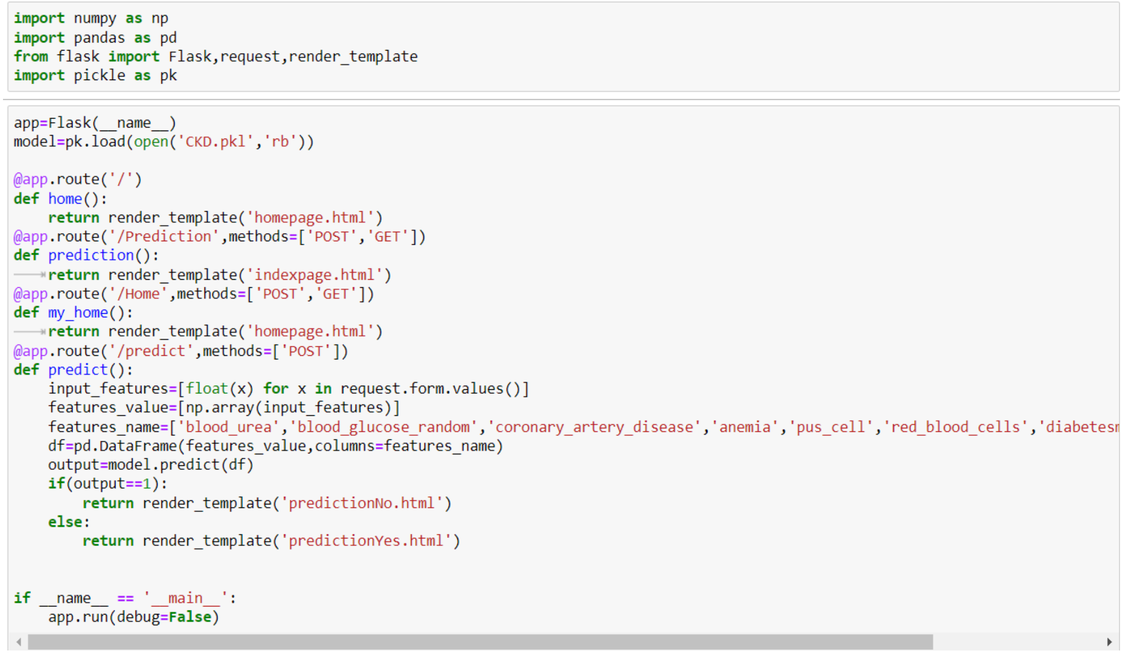


```
: pickle.dump(lgr,open('CKD.pkl','wb'))
```

7.3 – Local Deployment

The HTML code for the home page, index page and the two prediction pages are written.

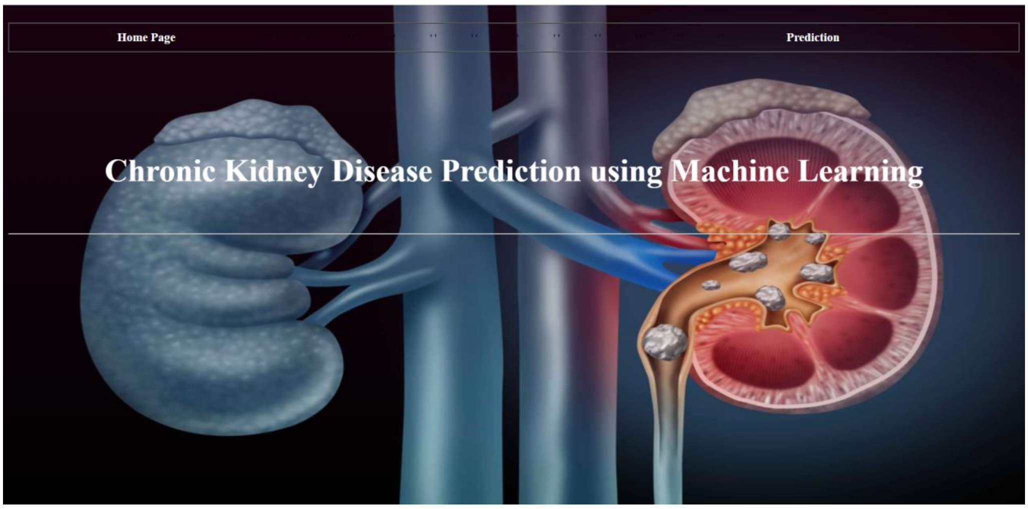
The app.py flask file which is a web framework written in python for server-side scripting is coded and run on jupyter notebook.



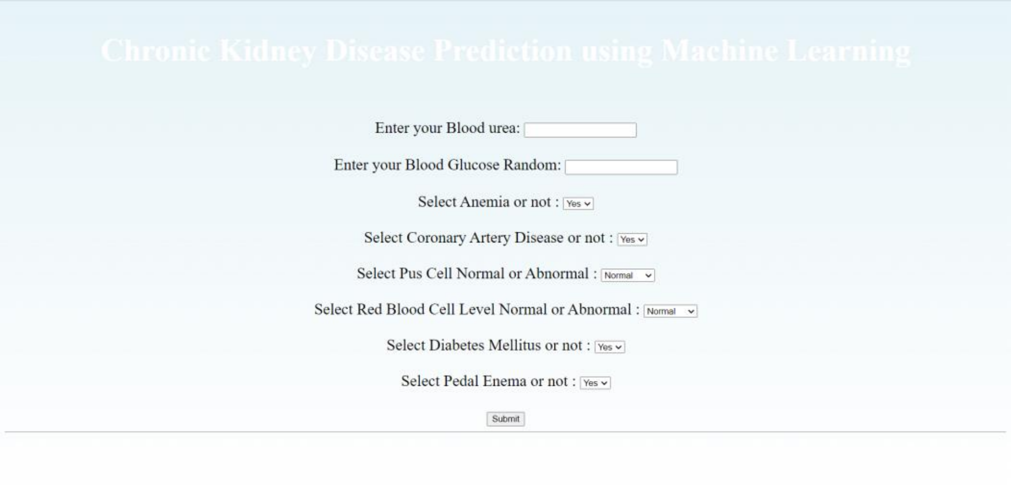
```
import numpy as np
import pandas as pd
from flask import Flask,request,render_template
import pickle as pk
```

```
app=Flask(__name__)
model=pk.load(open('CKD.pkl','rb'))

@app.route('/')
def home():
    return render_template('homepage.html')
@app.route('/Prediction',methods=['POST','GET'])
def prediction():
    return render_template('indexpage.html')
@app.route('/Home',methods=['POST','GET'])
def my_home():
    return render_template('homepage.html')
@app.route('/predict',methods=['POST'])
def predict():
    input_features=[float(x) for x in request.form.values()]
    features_value=[np.array(input_features)]
    features_name=['blood_urea','blood_glucose_random','coronary_artery_disease','anemia','pus_cell','red_blood_cells','diabetesm
    df=pd.DataFrame(features_value,columns=features_name)
    output=model.predict(df)
    if(output==1):
        return render_template('predictionNo.html')
    else:
        return render_template('predictionYes.html')


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

The app.py runs on the local host: 5000 and the web page is viewed.



Chronic Kidney Disease Prediction using Machine Learning



Chronic Kidney Disease Prediction using Machine Learning

Enter your Blood urea:

Enter your Blood Glucose Random:

Select Anemia or not : Yes

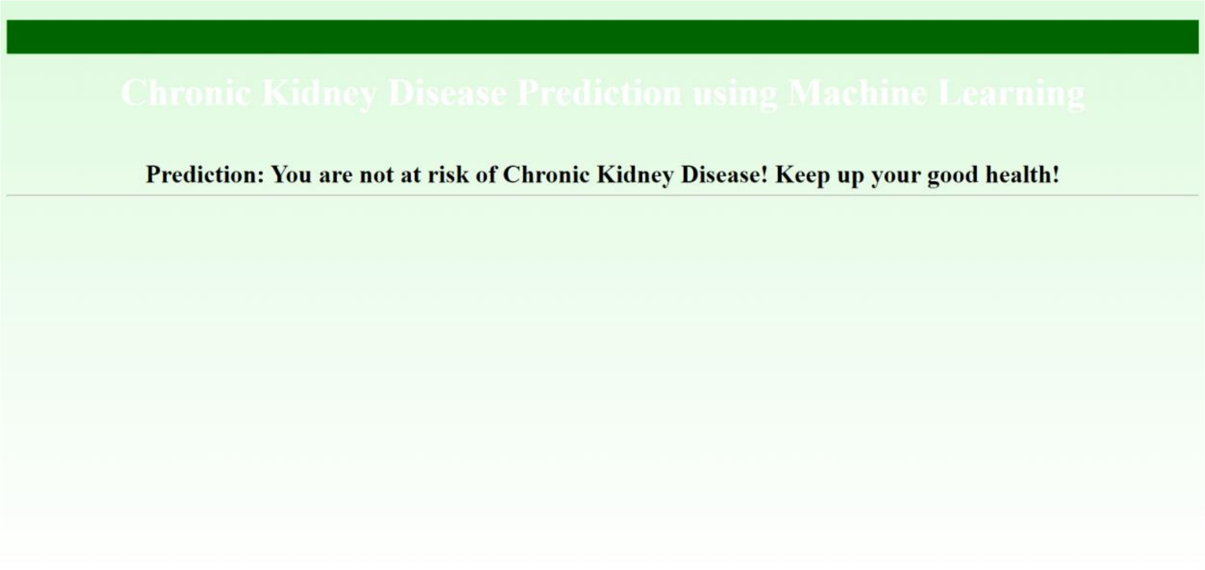
Select Coronary Artery Disease or not : Yes

Select Pus Cell Normal or Abnormal : Normal

Select Red Blood Cell Level Normal or Abnormal : Normal

Select Diabetes Mellitus or not : Yes

Select Pedal Enema or not : Yes



Chronic Kidney Disease Prediction using Machine Learning

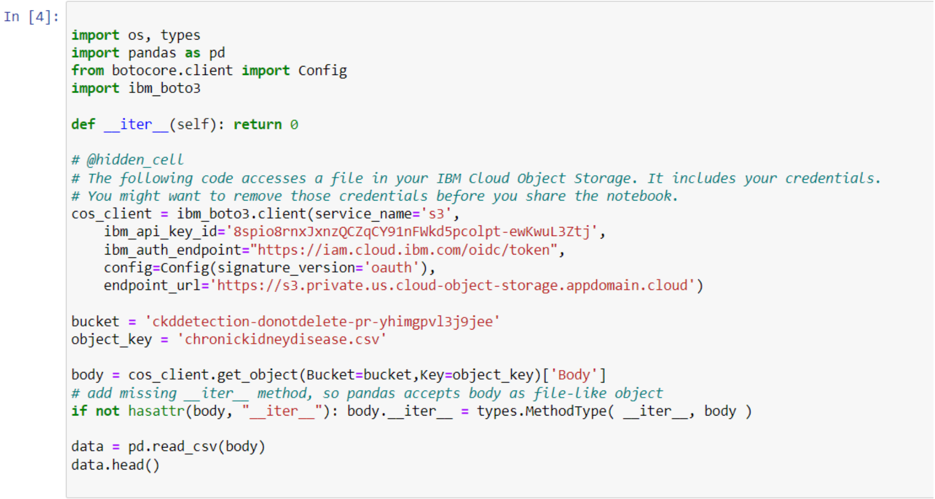
Prediction: You are not at risk of Chronic Kidney Disease! Keep up your good health!

FEATURE 7.4 – IBM CLOUD DEPLOYMENT

Once the local deployment is completed, the next step is to deploy the ML model from the cloud. Here, we utilise the IBM cloud platform. In the IBM cloud platform, Watson studio, Machine learning and Object Storage are used.

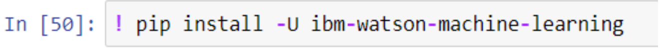
The same code that was used for training the model is loaded in the Watson studio. Few minor corrections are made in the code.

The pd.read_csv is replaced using a code which utilises API key to load the dataset stored in the IBM cloud space.



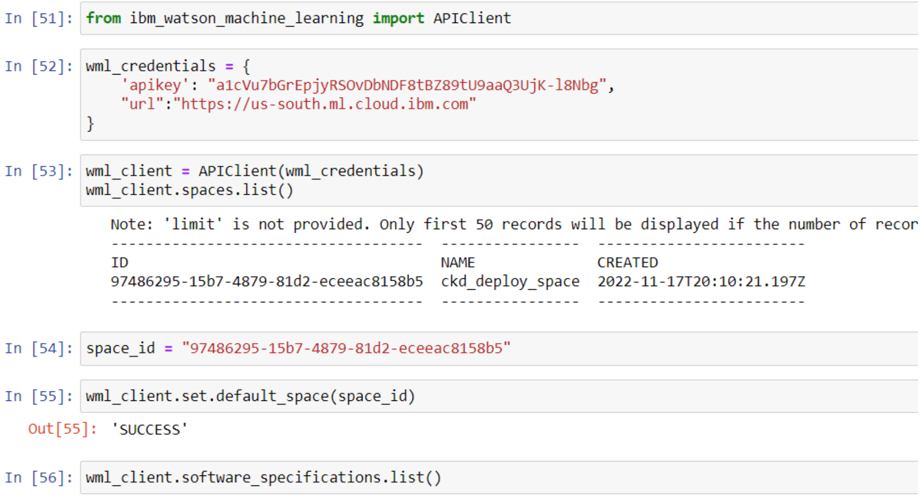
```
In [4]:
import os, types
import pandas as pd
from botocore.client import Config
import ibm_boto3

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='8spio8rnxJxnzQCZqCY91nFWkd5pcolpt-ewKwuL3Ztj',
    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 = 'ckddetection-donotdelete-pr-yhimgpvl3j9jee'
object_key = 'chronickidneydisease.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 )

data = pd.read_csv(body)
data.head()
```

The ibm Watson machine learning library is installed.



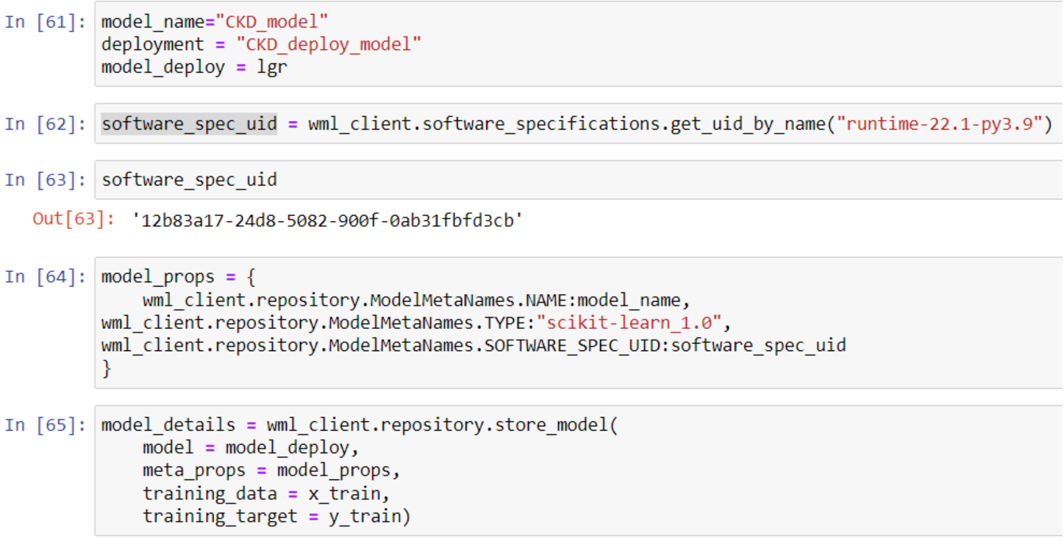
```
In [50]: ! pip install -U ibm-watson-machine-learning
```

Using the unique API key generated in IBM Cloud and mentioning our server location. Using the API credentials a new space is created in IBM Watson. The space has its unique Space id. This space id is set as default.



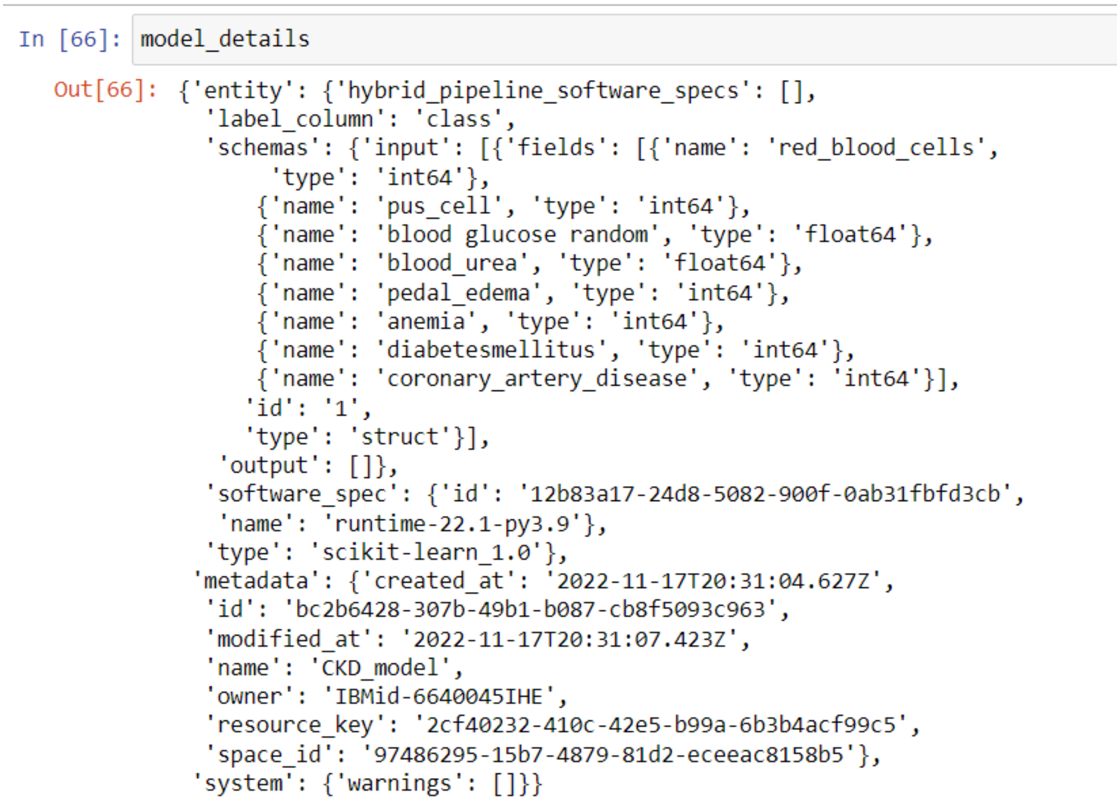
```
In [51]: from ibm_watson_machine_learning import APIClient

In [52]: wml_credentials = {
             'apikey': "a1cVu7bGrEpjyRSOvDbNDF8tBZ89tU9aaQ3UjK-l8Nbg",
             "url":"https://us-south.ml.cloud.ibm.com"
         }

In [53]: wml_client = APIClient(wml_credentials)
         wml_client.spaces.list()

            Note: 'limit' is not provided. Only first 50 records will be displayed if the number of recor
            ------------------------------------  ----------------  ------------------------
            ID                                    NAME              CREATED
            97486295-15b7-4879-81d2-eceeac8158b5  ckd_deploy_space  2022-11-17T20:10:21.197Z
            ------------------------------------  ----------------  ------------------------

In [54]: space_id = "97486295-15b7-4879-81d2-eceeac8158b5"

In [55]: wml_client.set.default_space(space_id)

Out[55]: 'SUCCESS'

In [56]: wml_client.software_specifications.list()
```

The required ML model is downloaded. Then, we look for the version that is being supported by IBM and downloading the correct version. This is followed by creating a new deployment space for the model. The next step is to set up the model requirements and link it to the deployment space followed by saving the model to the space by mentioning the attributes of the model.



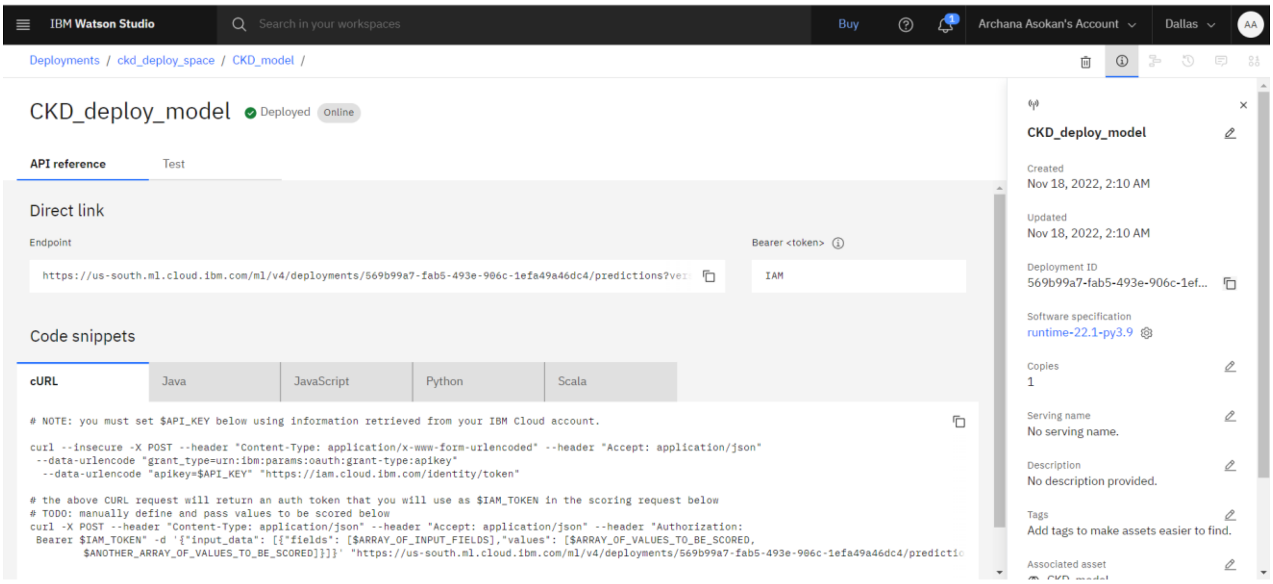
```
In [61]: model_name="CKD_model"
         deployment = "CKD_deploy_model"
         model_deploy = lgr

In [62]: software_spec_uid = wml_client.software_specifications.get_uid_by_name("runtime-22.1-py3.9")

In [63]: software_spec_uid

Out[63]: '12b83a17-24d8-5082-900f-0ab31fbfd3cb'

In [64]: model_props = {
             wml_client.repository.ModelMetaNames.NAME:model_name,
         wml_client.repository.ModelMetaNames.TYPE:"scikit-learn_1.0",
         wml_client.repository.ModelMetaNames.SOFTWARE_SPEC_UID:software_spec_uid
         }

In [65]: model_details = wml_client.repository.store_model(
             model = model_deploy,
             meta_props = model_props,
             training_data = x_train,
             training_target = y_train)
```

The details of the saved model are generated. This includes the API key required for saving the model.



```
In [66]: model_details

Out[66]: {'entity': {'hybrid_pipeline_software_specs': [],
           'label_column': 'class',
           'schemas': {'input': [{'fields': [{'name': 'red_blood_cells',
                 'type': 'int64'},
                {'name': 'pus_cell', 'type': 'int64'},
                {'name': 'blood glucose random', 'type': 'float64'},
                {'name': 'blood_urea', 'type': 'float64'},
                {'name': 'pedal_edema', 'type': 'int64'},
                {'name': 'anemia', 'type': 'int64'},
                {'name': 'diabetesmellitus', 'type': 'int64'},
                {'name': 'coronary_artery_disease', 'type': 'int64'}],
               'id': '1',
               'type': 'struct'}],
            'output': []},
           'software_spec': {'id': '12b83a17-24d8-5082-900f-0ab31fbfd3cb',
            'name': 'runtime-22.1-py3.9'},
           'type': 'scikit-learn_1.0'},
          'metadata': {'created_at': '2022-11-17T20:31:04.627Z',
           'id': 'bc2b6428-307b-49b1-b087-cb8f5093c963',
           'modified_at': '2022-11-17T20:31:07.423Z',
           'name': 'CKD_model',
           'owner': 'IBMid-6640045IHE',
           'resource_key': '2cf40232-410c-42e5-b99a-6b3b4acf99c5',
           'space_id': '97486295-15b7-4879-81d2-eceeac8158b5'},
          'system': {'warnings': []}}
```

Once the model has been created, saved, and deployed in the IBM cloud, we can see the below window. This consists of the Key required for deploying the model and also the header file and end scoring point which is to be integrated with the flask file.



CKD_deploy_model

Direct link

Endpoint

https://us-south.ml.cloud.ibm.com/ml/v4/deployments/569b99a7-fab5-493e-906c-1efa49a46dc4/predictions?ver

Bearer <token>

IAM

Code snippets

```
# NOTE: you must set $API_KEY below using information retrieved from your IBM Cloud account.

curl --insecure -X POST --header "Content-Type: application/x-www-form-urlencoded" --header "Accept: application/json"
 --data-urlencode "grant_type=urn:ibm:params:oauth:grant-type:apikey"
  --data-urlencode "apikey=$API_KEY" "https://iam.cloud.ibm.com/identity/token"

# the above CURL request will return an auth token that you will use as $IAM_TOKEN in the scoring request below
# TODO: manually define and pass values to be scored below
curl -X POST --header "Content-Type: application/json" --header "Accept: application/json" --header "Authorization:
 Bearer $IAM_TOKEN" -d '{"input_data": [{"fields": [$ARRAY_OF_INPUT_FIELDS],"values": [$ARRAY_OF_VALUES_TO_BE_SCORED,
        $ANOTHER_ARRAY_OF_VALUES_TO_BE_SCORED]}]}' "https://us-south.ml.cloud.ibm.com/ml/v4/deployments/569b99a7-fab5-493e-906c-1efa49a46dc4/predictic
```

CKD_deploy_model

Created
Nov 18, 2022, 2:10 AM

Updated
Nov 18, 2022, 2:10 AM

Deployment ID
569b99a7-fab5-493e-906c-1ef...

Software specification
runtime-22.1-py3.9

Copies
1

Serving name
No serving name.

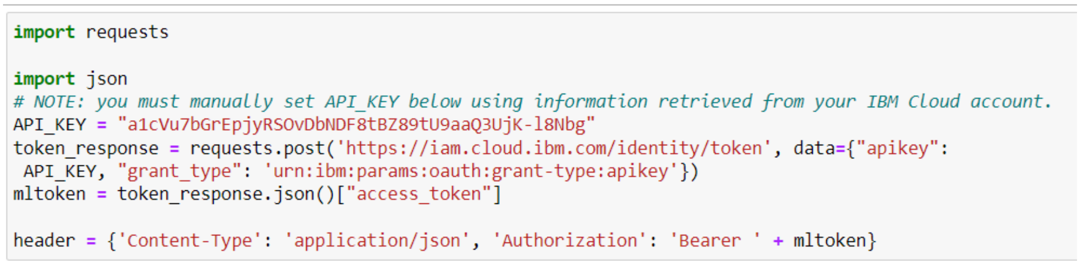
Description
No description provided.

Tags
Add tags to make assets easier to find.

Associated asset

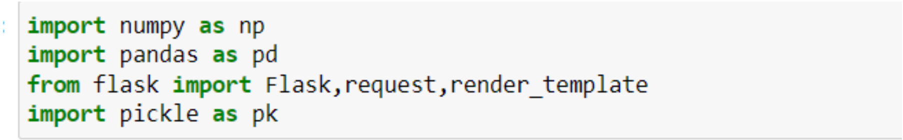
INTEGRATING FLASK WITH SCORING END POINT:

The below code is used to link the code with the model in IBM cloud. This consists of the user specific API key.



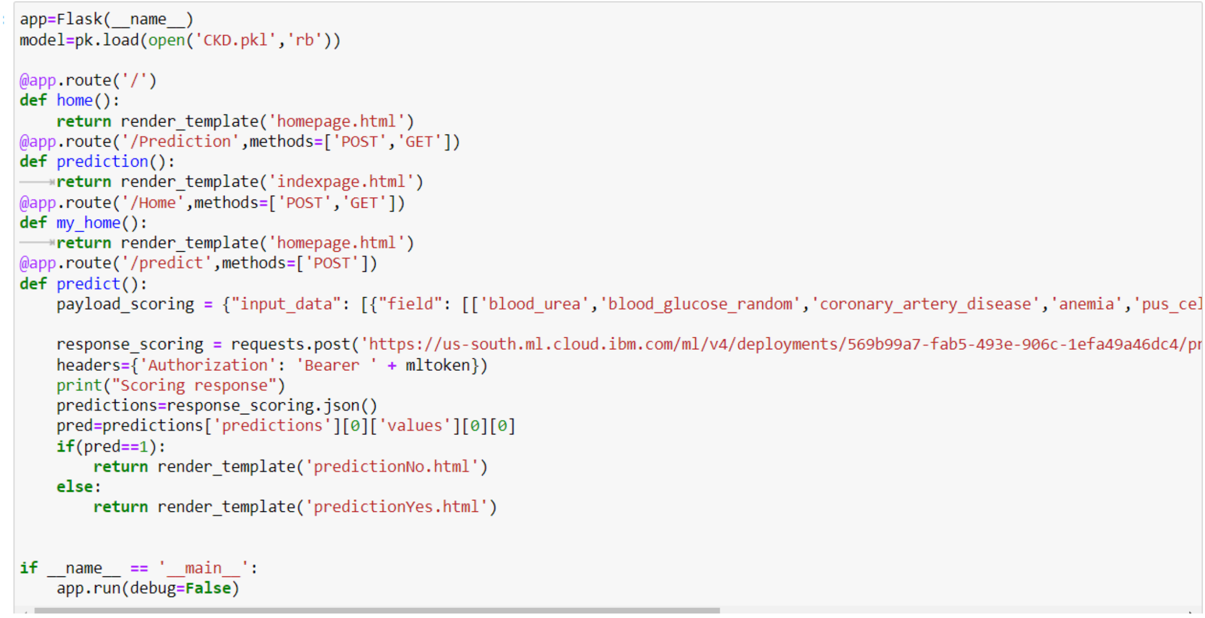
```
import requests

import json
# NOTE: you must manually set API_KEY below using information retrieved from your IBM Cloud account.
API_KEY = "a1cVu7bGrEpjyRSOvDbNDF8tBZ89tU9aaQ3UjK-l8Nbg"
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}
```

The necessary libraries are imported.



```
import numpy as np
import pandas as pd
from flask import Flask,request,render_template
import pickle as pk
```

The program given below serves as the backend for our Web page API and linking our Machine Learning model with it. We have a home page. From that you will be directed to the index page where you can give the inputs. This input received from that page is then sent to out ML model to do the prediction and the output will be displayed at the next web page, which can be either a prediction yes page (CKD positive) or prediction no page (CKD negative).

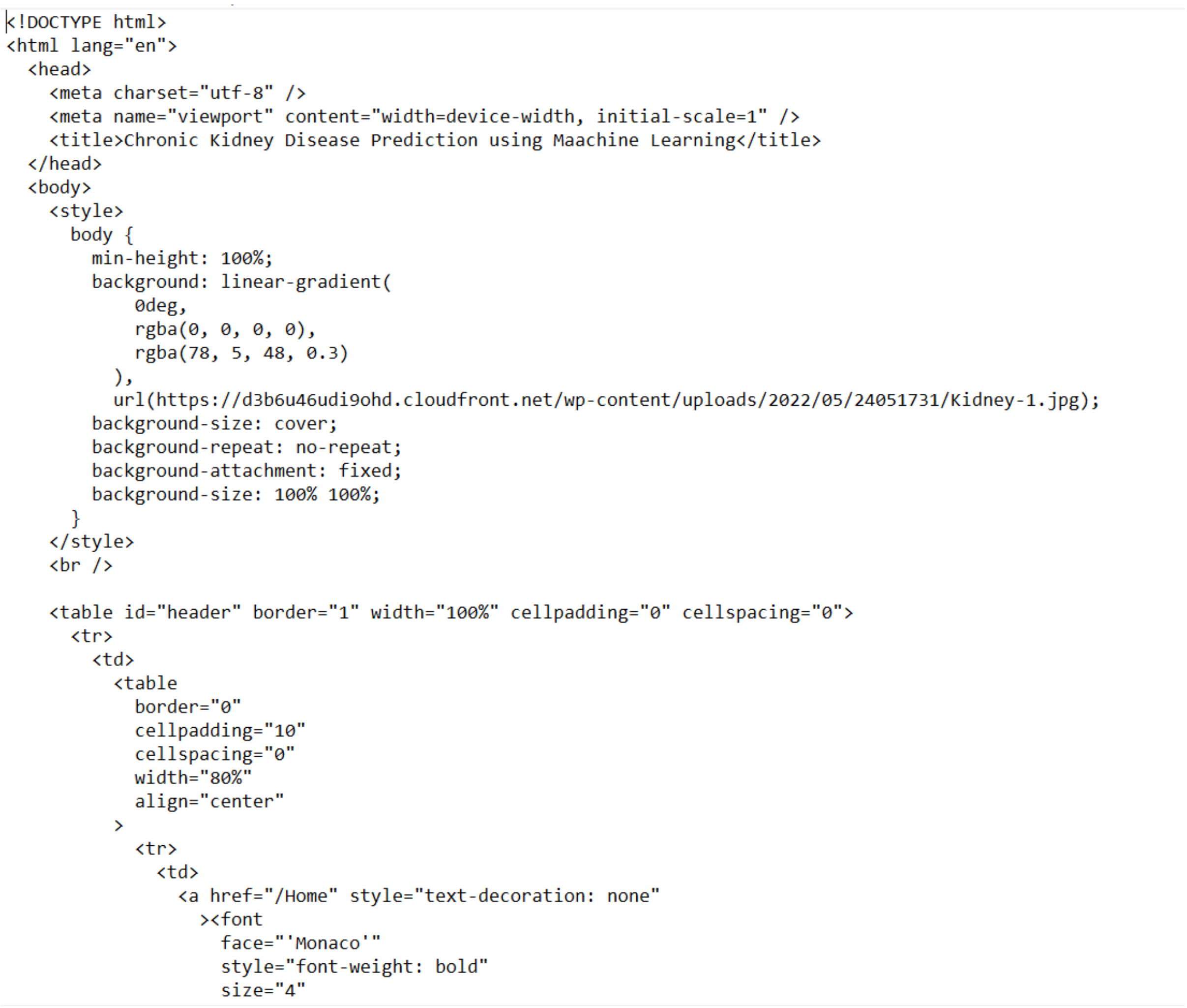


```
app=Flask(__name__)
model=pk.load(open('CKD.pkl','rb'))

@app.route('/')
def home():
    return render_template('homepage.html')
@app.route('/Prediction',methods=['POST','GET'])
def prediction():
    return render_template('indexpage.html')
@app.route('/Home',methods=['POST','GET'])
def my_home():
    return render_template('homepage.html')
@app.route('/predict',methods=['POST'])
def predict():
    payload_scoring = {"input_data": [{"field": [['blood_urea','blood_glucose_random','coronary_artery_disease','anemia','pus_cel

    response_scoring = requests.post('https://us-south.ml.cloud.ibm.com/ml/v4/deployments/569b99a7-fab5-493e-906c-1efa49a46dc4/pr
    headers={'Authorization': 'Bearer ' + mltoken})
    print("Scoring response")
    predictions=response_scoring.json()
    pred=predictions['predictions'][0]['values'][0][0]
    if(pred==1):
        return render_template('predictionNo.html')
    else:
        return render_template('predictionYes.html')


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

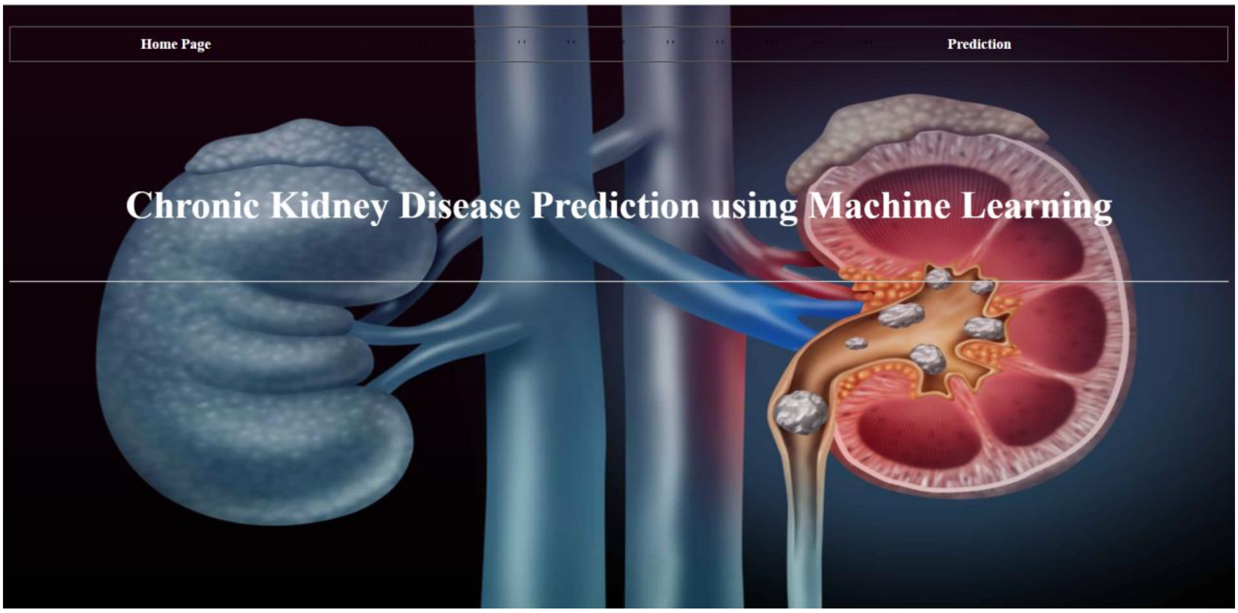
HTML PAGES:

Given below are the HTML codes for the four webpages and its corresponding result.

Home page:

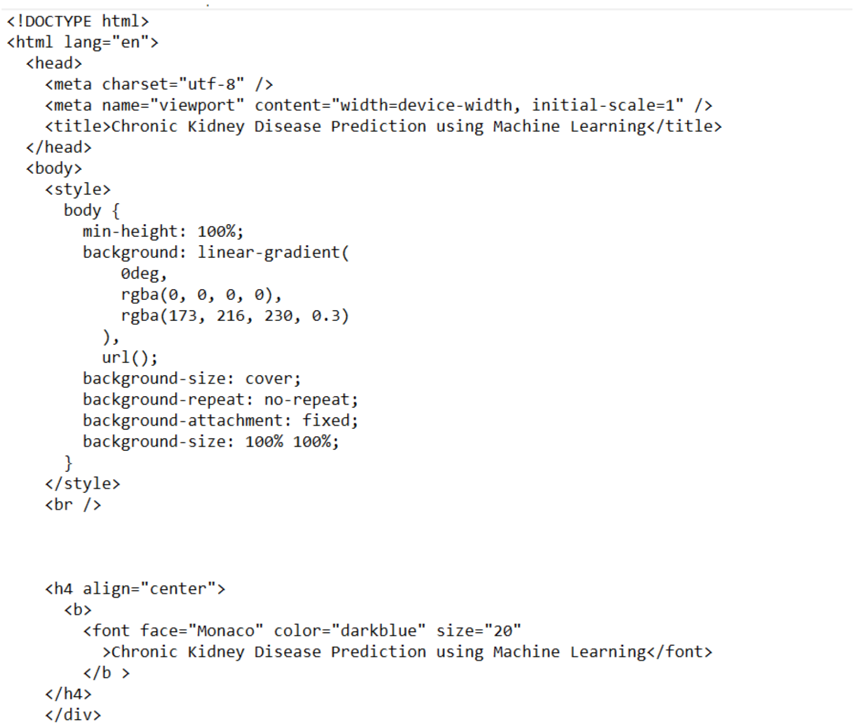


```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <title>Chronic Kidney Disease Prediction using Maachine Learning</title>
  </head>
  <body>
    <style>
      body {
        min-height: 100%;
        background: linear-gradient(
            0deg,
            rgba(0, 0, 0, 0),
            rgba(78, 5, 48, 0.3)
          ),
          url(https://d3b6u46udi9ohd.cloudfront.net/wp-content/uploads/2022/05/24051731/Kidney-1.jpg);
        background-size: cover;
        background-repeat: no-repeat;
        background-attachment: fixed;
        background-size: 100% 100%;
      }
    </style>
    <br />

    <table id="header" border="1" width="100%" cellpadding="0" cellspacing="0">
      <tr>
        <td>
          <table
            border="0"
            cellpadding="10"
            cellspacing="0"
            width="80%"
            align="center"
          >
            <tr>
              <td>
                <a href="/Home" style="text-decoration: none"
                  ><font
                    face="'Monaco'"
                    style="font-weight: bold"
                    size="4"
                  >
                  </font
                ></a>
              </td>
              <td>
                <a href="/Prediction" style="text-decoration: none"
                  ><font
                    face="'Monaco'"
                    style="font-weight: bold"
                    size="4"
                    color="white"
                    >Prediction</font
                  ></a
                >
              </td>
            </tr>
          </table>
        </td>
      </tr>
    </table>

    <br />
    <br />
    <br />
    <br />
    <br />
    <br />
    <br />

    <h1 align="center">
      <b>
        <font face="Monaco" color="white" size="35"
          >Chronic Kidney Disease Prediction using Machine Learning</font
        ></b
      >
    </h1>

    <br /><br />

    <hr />
  </body>
</html>
```

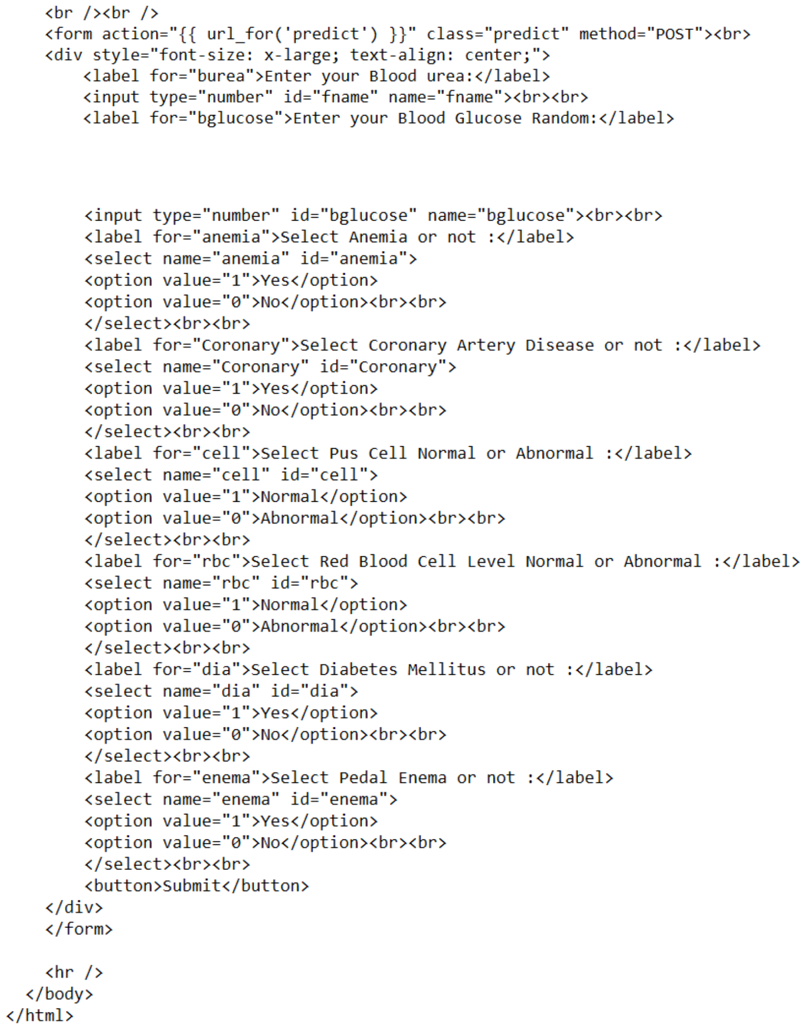


Chronic Kidney Disease Prediction using Machine Learning

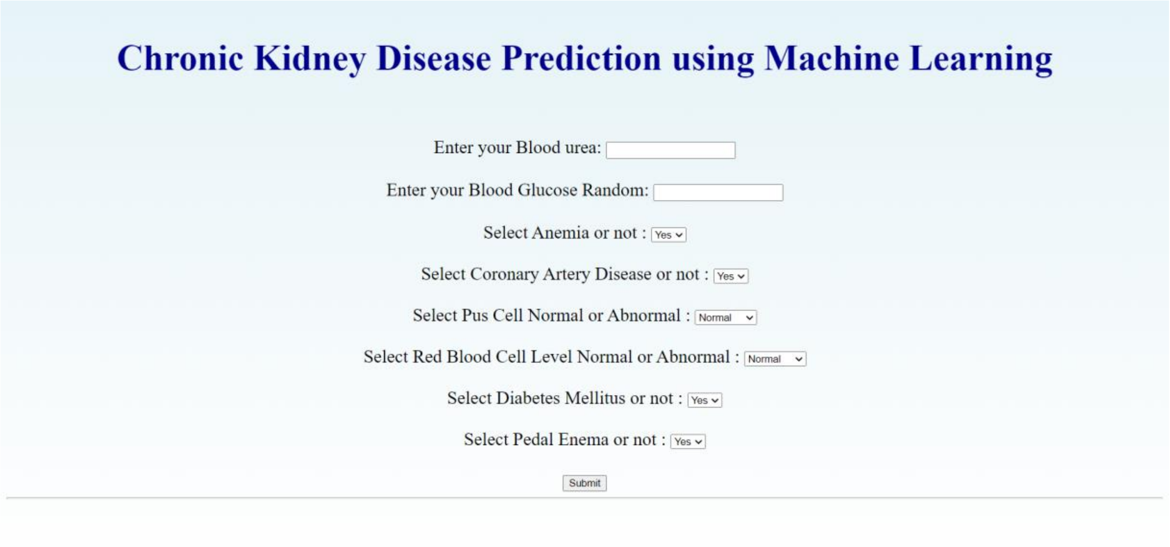
Index page:



```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <title>Chronic Kidney Disease Prediction using Machine Learning</title>
  </head>
  <body>
    <style>
      body {
        min-height: 100%;
        background: linear-gradient(
            0deg,
            rgba(0, 0, 0, 0),
            rgba(173, 216, 230, 0.3)
          ),
          url();
        background-size: cover;
        background-repeat: no-repeat;
        background-attachment: fixed;
        background-size: 100% 100%;
      }
    </style>
    <br />



    <h4 align="center">
      <b>
        <font face="Monaco" color="darkblue" size="20"
          >Chronic Kidney Disease Prediction using Machine Learning</font>
        </b >
    </h4>
    </div>
```



```
    <br /><br />
    <form action="{{ url_for('predict') }}" class="predict" method="POST"><br>
    <div style="font-size: x-large; text-align: center;">
        <label for="burea">Enter your Blood urea:</label>
        <input type="number" id="fname" name="fname"><br><br>
        <label for="bglucose">Enter your Blood Glucose Random:</label>



        <input type="number" id="bglucose" name="bglucose"><br><br>
        <label for="anemia">Select Anemia or not :</label>
        <select name="anemia" id="anemia">
        <option value="1">Yes</option>
        <option value="0">No</option><br><br>
        </select><br><br>
        <label for="Coronary">Select Coronary Artery Disease or not :</label>
        <select name="Coronary" id="Coronary">
        <option value="1">Yes</option>
        <option value="0">No</option><br><br>
        </select><br><br>
        <label for="cell">Select Pus Cell Normal or Abnormal :</label>
        <select name="cell" id="cell">
        <option value="1">Normal</option>
        <option value="0">Abnormal</option><br><br>
        </select><br><br>
        <label for="rbc">Select Red Blood Cell Level Normal or Abnormal :</label>
        <select name="rbc" id="rbc">
        <option value="1">Normal</option>
        <option value="0">Abnormal</option><br><br>
        </select><br><br>
        <label for="dia">Select Diabetes Mellitus or not :</label>
        <select name="dia" id="dia">
        <option value="1">Yes</option>
        <option value="0">No</option><br><br>
        </select><br><br>
        <label for="enema">Select Pedal Enema or not :</label>
        <select name="enema" id="enema">
        <option value="1">Yes</option>
        <option value="0">No</option><br><br>
        </select><br><br>
        <button>Submit</button>
    </div>
    </form>

    <hr />
  </body>
</html>
```



Chronic Kidney Disease Prediction using Machine Learning

Enter your Blood urea:

Enter your Blood Glucose Random:

Select Anemia or not : Yes

Select Coronary Artery Disease or not : Yes

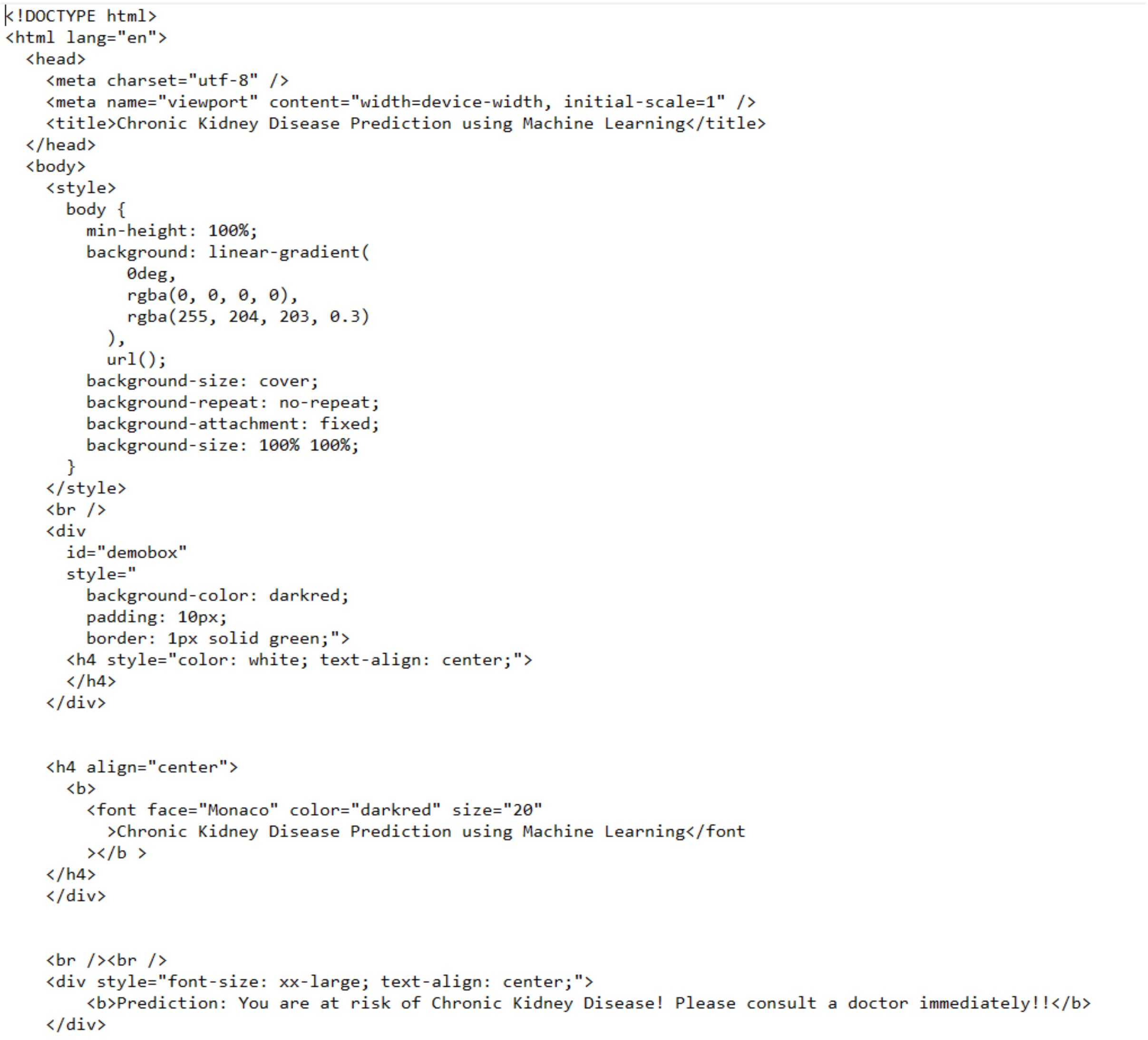
Select Pus Cell Normal or Abnormal : Normal

Select Red Blood Cell Level Normal or Abnormal : Normal

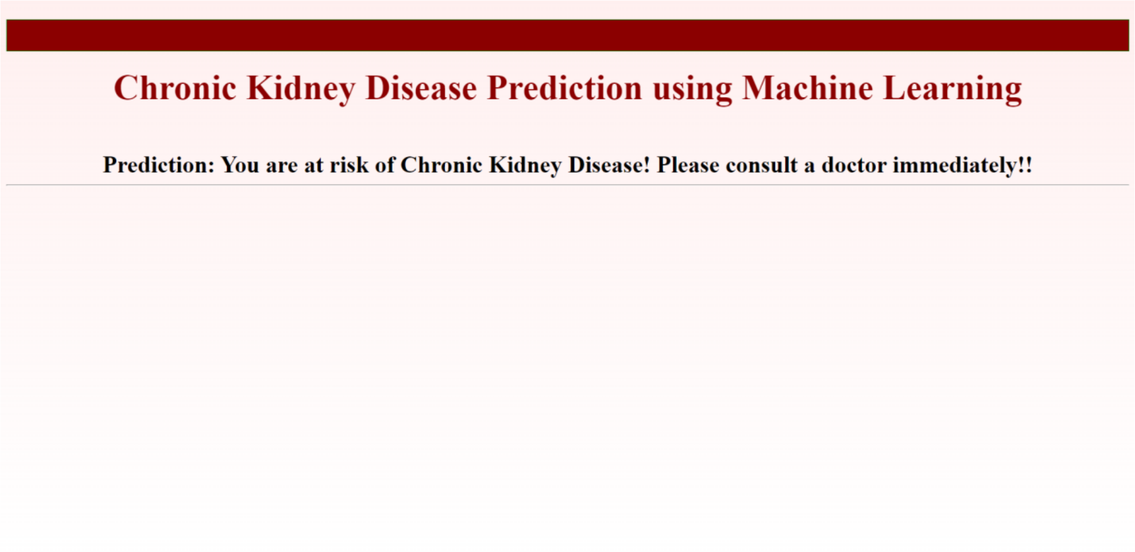
Select Diabetes Mellitus or not : Yes

Select Pedal Enema or not : Yes

Chronic Kidney Disease Positive Page:



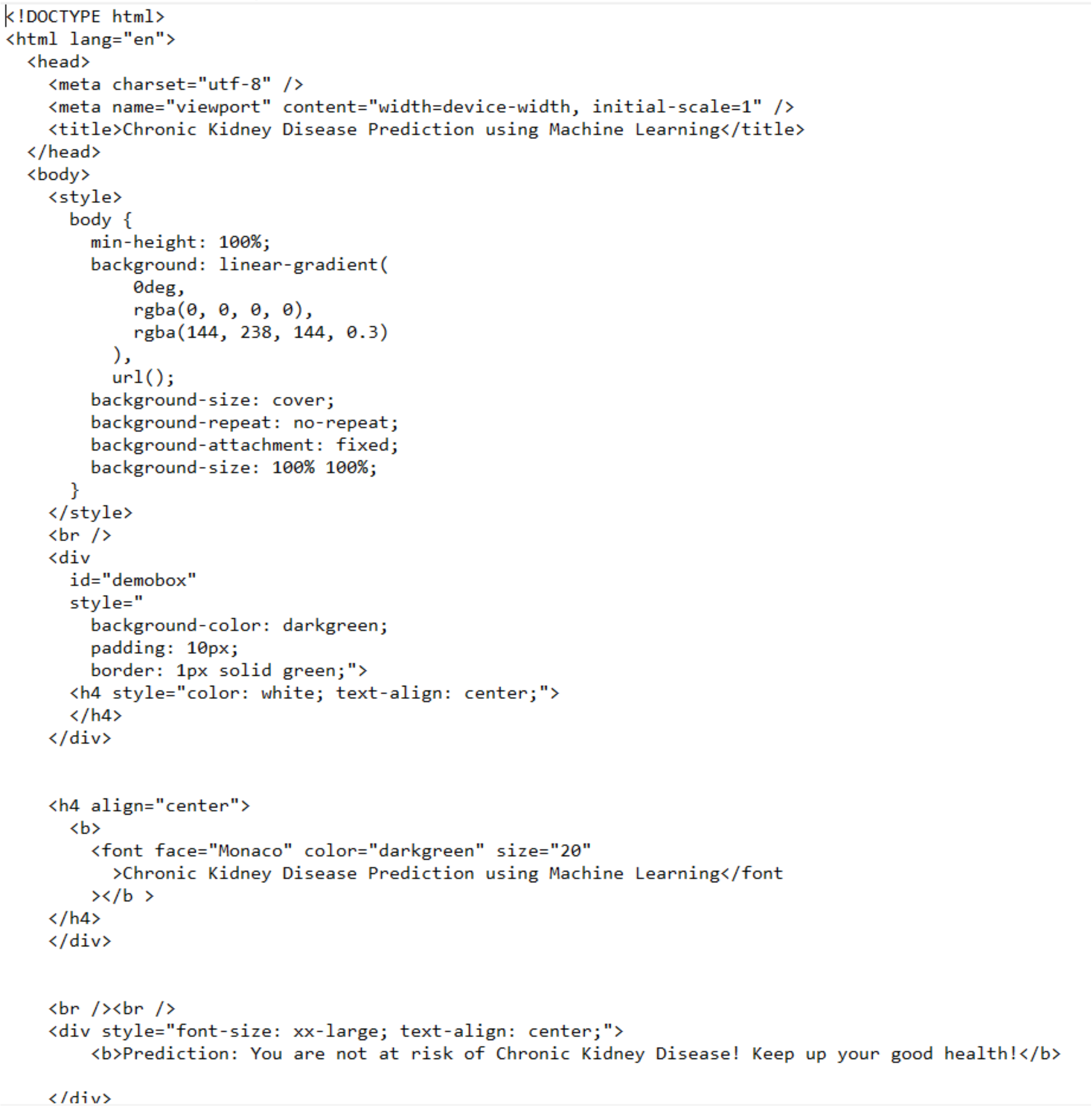
```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <title>Chronic Kidney Disease Prediction using Machine Learning</title>
  </head>
  <body>
    <style>
      body {
        min-height: 100%;
        background: linear-gradient(
            0deg,
            rgba(0, 0, 0, 0),
            rgba(255, 204, 203, 0.3)
          ),
          url();
        background-size: cover;
        background-repeat: no-repeat;
        background-attachment: fixed;
        background-size: 100% 100%;
      }
    </style>
    <br />
    <div
      id="demobox"
      style="
        background-color: darkred;
        padding: 10px;
        border: 1px solid green;">
      <h4 style="color: white; text-align: center;">
      </h4>
    </div>


    <h4 align="center">
      <b>
        <font face="Monaco" color="darkred" size="20"
          >Chronic Kidney Disease Prediction using Machine Learning</font
        ></b >
    </h4>
    </div>


    <br /><br />
    <div style="font-size: xx-large; text-align: center;">
        <b>Prediction: You are at risk of Chronic Kidney Disease! Please consult a doctor immediately!!</b>
    </div>
```



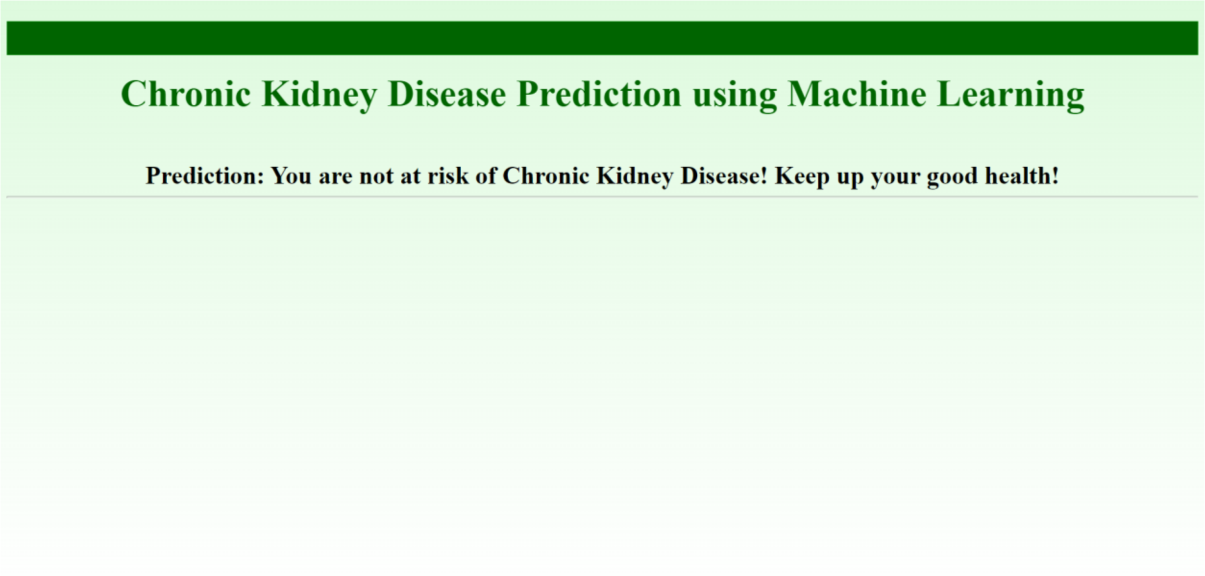
Chronic Kidney Disease Prediction using Machine Learning

Prediction: You are at risk of Chronic Kidney Disease! Please consult a doctor immediately!!

Chronic Kidney Disease Negative Page:



```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <title>Chronic Kidney Disease Prediction using Machine Learning</title>
  </head>
  <body>
    <style>
      body {
        min-height: 100%;
        background: linear-gradient(
            0deg,
            rgba(0, 0, 0, 0),
            rgba(144, 238, 144, 0.3)
          ),
          url();
        background-size: cover;
        background-repeat: no-repeat;
        background-attachment: fixed;
        background-size: 100% 100%;
      }
    </style>
    <br />
    <div
      id="demobox"
      style="
        background-color: darkgreen;
        padding: 10px;
        border: 1px solid green;">
      <h4 style="color: white; text-align: center;">
      </h4>
    </div>


    <h4 align="center">
      <b>
        <font face="Monaco" color="darkgreen" size="20"
          >Chronic Kidney Disease Prediction using Machine Learning</font
        ></b >
    </h4>
    </div>


    <br /><br />
    <div style="font-size: xx-large; text-align: center;">
        <b>Prediction: You are not at risk of Chronic Kidney Disease! Keep up your good health!</b>

    </div>
```



Chronic Kidney Disease Prediction using Machine Learning

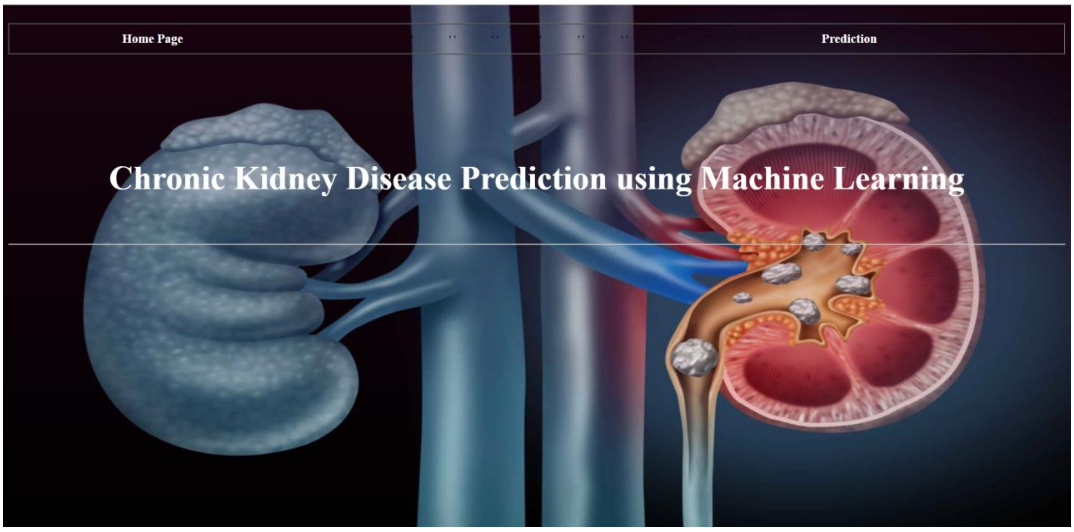
Prediction: You are not at risk of Chronic Kidney Disease! Keep up your good health!

When the flask file is run, the ML model that is saved in the IBM cloud is deployed. This results in the generation of the home page in the browser used. From the home page, we can navigate to and provide inputs in the index page. The ML model will then compute the result and display either one of the two prediction pages.

8. TESTING

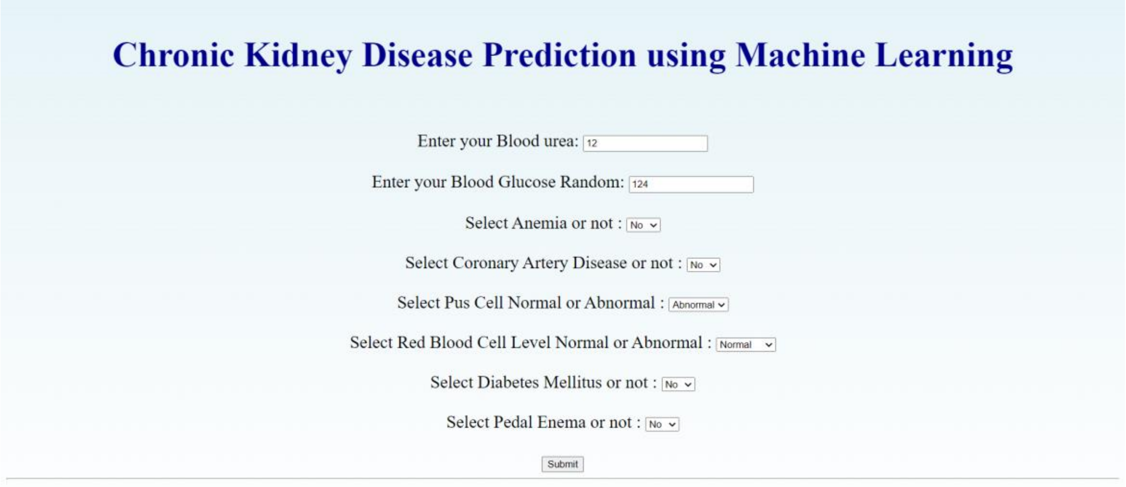
8.1 TEST CASES

Test Case Number	Blood Urea	Blood Glucose Random	Anemia	Coronary artery disease	Pus Cell	RBC Level	Diabetes Mellitus	Pedal Enema	CKD or Not
1	53	423	Yes	No	Abnormal	Abnormal	Yes	No	CKD
2	109	26	No	Yes	Normal	Normal	Yes	No	Not CKD
3	75	38	No	No	Normal	Normal	No	Yes	Not CKD
4	54	100	No	Yes	Normal	Abnormal	No	Yes	CKD
5	140	16	Yes	No	Normal	Abnormal	No	No	Not CKD



Chronic Kidney Disease Prediction using Machine Learning

TEST CASE 1:



Chronic Kidney Disease Prediction using Machine Learning

Enter your Blood urea: 12

Enter your Blood Glucose Random: 124

Select Anemia or not : No

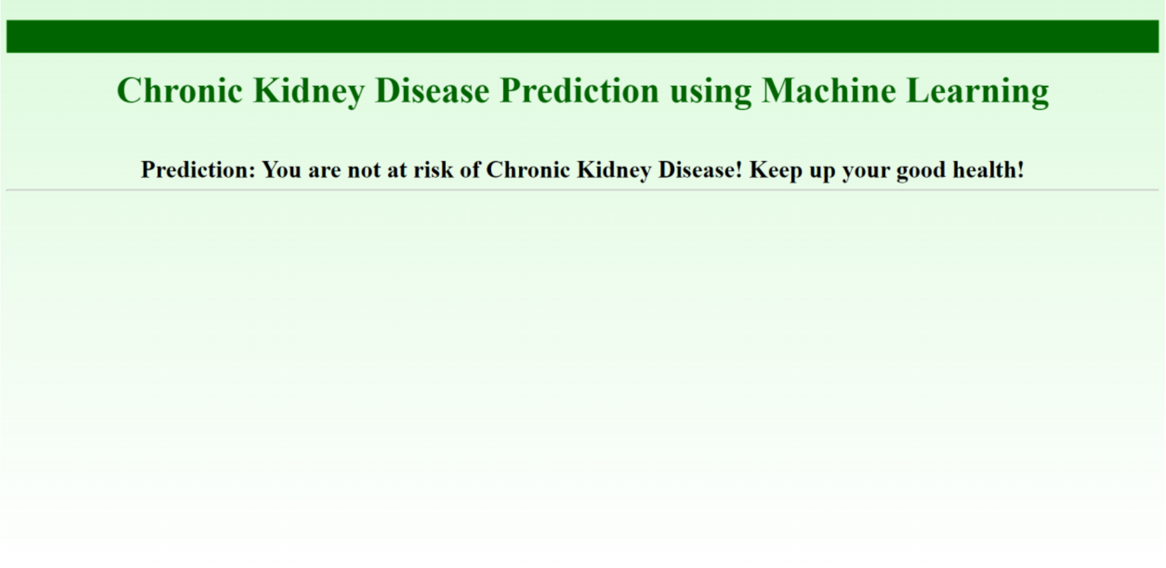
Select Coronary Artery Disease or not : No

Select Pus Cell Normal or Abnormal : Abnormal

Select Red Blood Cell Level Normal or Abnormal : Normal

Select Diabetes Mellitus or not : No

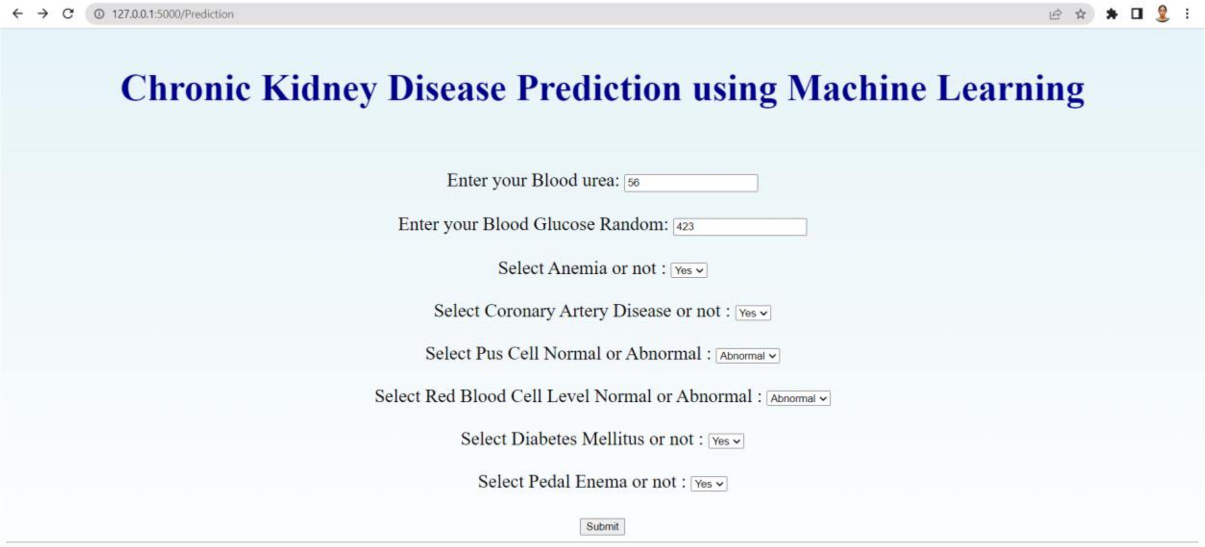
Select Pedal Enema or not : No



Chronic Kidney Disease Prediction using Machine Learning

Prediction: You are not at risk of Chronic Kidney Disease! Keep up your good health!

TEST CASE 2:



Chronic Kidney Disease Prediction using Machine Learning

Enter your Blood urea: 56

Enter your Blood Glucose Random: 423

Select Anemia or not : Yes

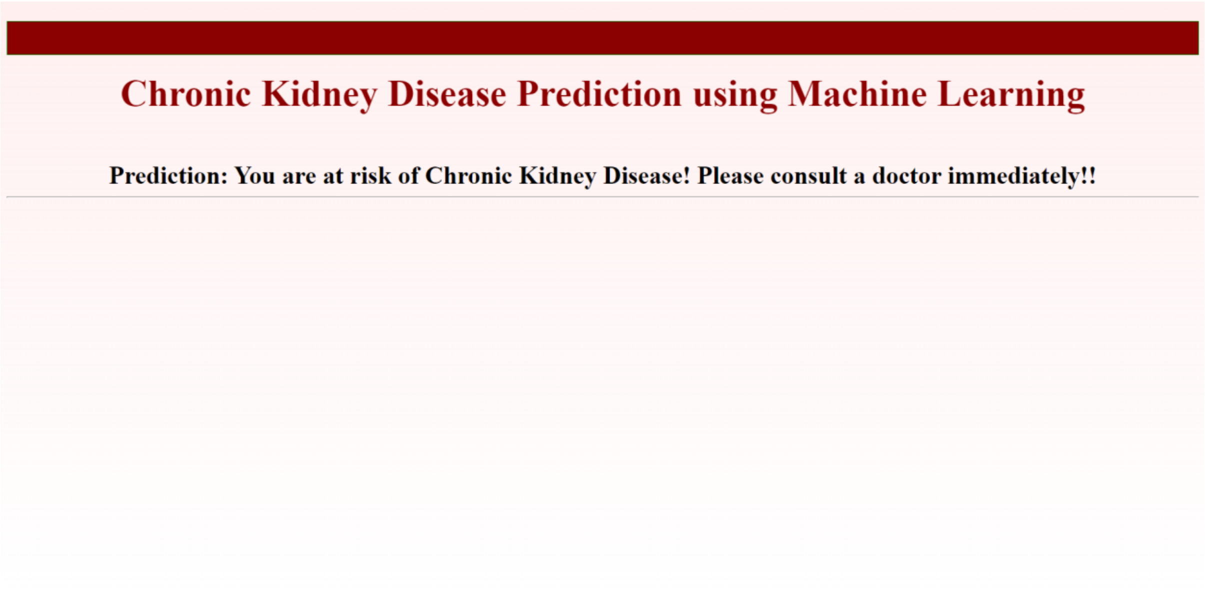
Select Coronary Artery Disease or not : Yes

Select Pus Cell Normal or Abnormal : Abnormal

Select Red Blood Cell Level Normal or Abnormal : Abnormal

Select Diabetes Mellitus or not : Yes

Select Pedal Enema or not : Yes



Chronic Kidney Disease Prediction using Machine Learning

Prediction: You are at risk of Chronic Kidney Disease! Please consult a doctor immediately!!

As observed, different inputs are given and the result is seen. Similarly, different values of inputs such as urea, blood sugar, anaemia, coronary artery disease etc are given and the result is observed and noted.

8.2 USER ACCEPTANCE TESTING

DEFECT ANALYSIS:

The number of resolved or closed bugs at each severity level, and how they were resolved

Resolution	Severity 1	Severity 2	Severity 3	Severity 4	Subtotal
By Design	10	3	1	1	18
Fixed	14	3	2	2	12
Skipped	0	0	0	0	0
Won't Fix	0	0	0	0	0
Totals	24	6	3	3	30

TEST CASE ANALYSIS:

This report shows the number of test cases that have passed, failed, and untested

Section	Total Cases	Not Tested	Fail	Pass
Print Engine	13	0	0	13
Client Application	21	0	0	21
Security	2	0	0	2
Exception Reporting	1	0	0	1
Final Report Output	8	0	0	8
Version Control	1	0	0	1

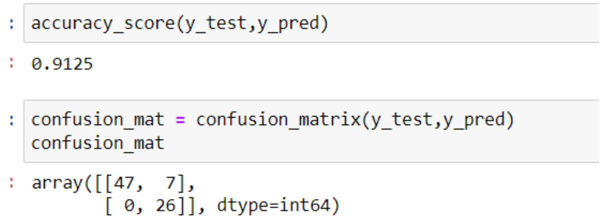
9. RESULTS

9.1 PERFORMANCE METRICS

The Logistic Regression ML model that we have used here has better performance in accuracy compared to other models. We have compared the performance metrics of 2 models and selected this as the best for the application. The model performed well for all the test cases. The API developed also performed good with no glitches or lag found during the deployment and testing phase.

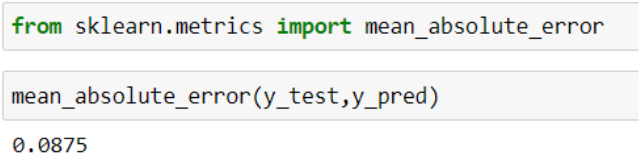
The accuracy score, confusion matrix and error values are attached below to support the above statement.

Accuracy = 91.25%



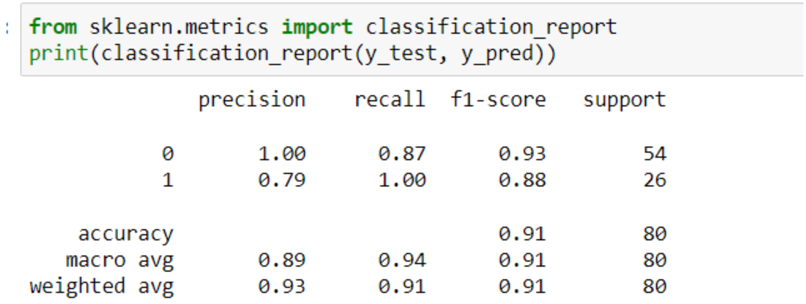
```
: accuracy_score(y_test,y_pred)

: 0.9125

: confusion_mat = confusion_matrix(y_test,y_pred)
  confusion_mat

: array([[47,  7],
         [ 0, 26]], dtype=int64)
```

Mean absolute error = 0.0875 = 8.75%



```
from sklearn.metrics import mean_absolute_error

mean_absolute_error(y_test,y_pred)

0.0875
```

Classification Report:



```
: from sklearn.metrics import classification_report
  print(classification_report(y_test, y_pred))

              precision    recall  f1-score   support

           0       1.00      0.87      0.93        54
           1       0.79      1.00      0.88        26

    accuracy                           0.91        80
   macro avg       0.89      0.94      0.91        80
weighted avg       0.93      0.91      0.91        80
```

The ML model trained is found to have good accuracy with very minimal error. So, the developed ML model for Early Detection of Chronic Kidney Disease is highly accurate.

10. ADVANTAGES AND DIS-ADVANTAGES

ADVANTAGES:

Chronic kidney disease involves a gradual loss of kidney function. 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.

Detecting the Chronic kidney disease in the early stage has more advantage to take medications in the first stage

DIS-ADVANTAGE:

The limitations of early detection include the following: issues related to the sensitivity of currently available tests; the pre-test probability of disease in the population of interest; a lack of ability to accurately predict progression in individuals; and in some parts of the world the inability to offer therapy irrespective of diagnosis.

11. CONCLUSION

The early identification of CKD should increase the amount of time that patients are exposed to therapeutic strategies of proven benefit. Such increased exposure is expected to have several beneficial effects: preventing or delaying progression to ESRD; improving patient safety through avoidance of nephrotoxic effects of drugs and/or procedures (for example, contrast-based imaging); and, possibly, preventing AKI episodes in patients identified as having CKD.

In turn, these improvements should lead to a reduction in the need for RRT and improved health for these individuals. Given the already large burden of CKD on health-care resources, which is projected to increase still further in both developed and developing countries, improvements in CKD detection, an improved ability to predict individual patients’ prognosis and better treatment strategies are necessary. As the incidence of diabetes increases in the developing world, and exposure to infections and drug therapies remain a constant threat to all, the impetus to ensure that identification of CKD occurs as early as possible must continue.

12. FUTURE WORKS

This project may help the future to detect the Chronic Kidney Disease early and to proceed with the treatments during the initial stage itself. Also, In Future, is model can also be improvised to give even more accuracy which helps to provide finite information.

13. APPENDIX

13.1 SOURCE CODE

DATA PRE-PROCESSING AND MODEL BUILDING CODE:

```
import pandas as pd

import numpy as mp

from collections import Counter as c

import matplotlib.pyplot as plt

import seaborn as sns

import missingno as msno

from sklearn.metrics import accuracy_score, confusion_matrix

from sklearn.model_selection import train_test_split

from sklearn.preprocessing import LabelEncoder

from sklearn.linear_model import LogisticRegression

import pickle

data=pd.read_csv(r"C:\Users\archa\Desktop\IBM Datasets\chronickidneydisease.csv")

data.head(5)

data.tail(5)

data.drop(["id"],axis=1,inplace=True)

data.columns

data.columns=['age', 'blood_pressure', 'specific_gravity', 'albumin', 'sugar', 'red_blood_cells',
'pus_cell', 'pus_cell_clumps', 'bacteria', 'blood glucose random', 'blood_urea',

       'serum_creatinine', 'sodium', 'potassium', 'hemoglobin', 'packed_cell_volume',
'white_blood_cell_count', 'red_blood_cell_count', 'hypertension', 'diabetesmellitus',
'coronary_artery_disease',

       'appetite', 'pedal_edema', 'anemia', 'class']

data.columns
data.info()

data['class'].unique()

data['class']=data['class'].replace("ckd\t","ckd")

data['class'].unique()

# ## Categorical Columns

catcols=set(data.dtypes[data.dtypes=='O'].index.values)

print(catcols)

    print("Columns :",i)

    print(c(data[i]))

    print('*'*120+'\n')


catcols.remove('red_blood_cell_count')

catcols.remove('packed_cell_volume')

catcols.remove('white_blood_cell_count')

print(catcols)

# ## Numerical Columns

contcols=set(data.dtypes[data.dtypes!='O'].index.values)

print(contcols)

for i in contcols:

    print("Continuous Columns :",i)

    print(c(data[i]))

    print('*'*120+'\n')


contcols.remove('specific_gravity')

contcols.remove('albumin')
contcols.remove('sugar')

print(contcols)

contcols.add('red_blood_cell_count')

contcols.add('packed_cell_volume')

contcols.add('white_blood_cell_count')

print(contcols)

catcols.add('specific_gravity')

catcols.add('albumin')

catcols.add('sugar')

print(catcols)

data['coronary_artery_disease']=data.coronary_artery_disease.replace('\tno','no')

c(data['coronary_artery_disease'])

data['diabetesmellitus']=data.diabetesmellitus.replace(to_replace={'\tno':'no','\tyes':'yes','yes':'yes'})

c(data['diabetesmellitus'])


data.isnull().any()

data.isnull().sum()

data.packed_cell_volume = pd.to_numeric(data.packed_cell_volume, errors='coerce')

data.white_blood_cell_count = pd.to_numeric(data.white_blood_cell_count, errors='coerce')

data.red_blood_cell_count = pd.to_numeric(data.red_blood_cell_count, errors='coerce')


data['blood glucose random'].fillna(data['blood glucose random'].mean(),inplace=True)

data['blood_pressure'].fillna(data['blood_pressure'].mean(),inplace=True)

data['blood_urea'].fillna(data['blood_urea'].mean(),inplace=True)

data['hemoglobin'].fillna(data['hemoglobin'].mean(),inplace=True)
```

```
data['packed_cell_volume'].fillna(data['packed_cell_volume'].mean(),inplace=True)

data['potassium'].fillna(data['potassium'].mean(),inplace=True)

data['red_blood_cell_count'].fillna(data['red_blood_cell_count'].mean(),inplace=True)

data['serum_creatinine'].fillna(data['serum_creatinine'].mean(),inplace=True)

data['sodium'].fillna(data['sodium'].mean(),inplace=True)

data['white_blood_cell_count'].fillna(data['white_blood_cell_count'].mean(),inplace=True)



data['age'].fillna(data['age'].mode()[0],inplace=True)

data['hypertension'].fillna(data['hypertension'].mode()[0],inplace=True)

data['pus_cell_clumps'].fillna(data['pus_cell_clumps'].mode()[0],inplace=True)

data['appetite'].fillna(data['appetite'].mode()[0],inplace=True)

data['albumin'].fillna(data['albumin'].mode()[0],inplace=True)

data['pus_cell'].fillna(data['pus_cell'].mode()[0],inplace=True)

data['red_blood_cells'].fillna(data['red_blood_cells'].mode()[0],inplace=True)

data['coronary_artery_disease'].fillna(data['coronary_artery_disease'].mode()[0],inplace=True
)

data['bacteria'].fillna(data['bacteria'].mode()[0],inplace=True)

data['anemia'].fillna(data['anemia'].mode()[0],inplace=True)

data['sugar'].fillna(data['sugar'].mode()[0],inplace=True)

data['diabetesmellitus'].fillna(data['diabetesmellitus'].mode()[0],inplace=True)

data['pedal_edema'].fillna(data['pedal_edema'].mode()[0],inplace=True)

data['specific_gravity'].fillna(data['specific_gravity'].mode()[0],inplace=True)

data.isnull().any()

for i in catcols:

    print("Label Encoding of: ",i)
    LEi = LabelEncoder()

    print(c(data[i]))

    data[i]=LEi.fit_transform(data[i])

    print(c(data[i]))

    print("*"*100)


selcols=['red_blood_cells','pus_cell','blood glucose random','blood_urea','pedal_edema','anemia','diabetesmellitus','coronary_artery_disease']

x=pd.DataFrame(data,columns=selcols)

y=pd.DataFrame(data,columns=['class'])

print(x.shape)

print(y.shape)

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=2)

print(x_train.shape)

print(y_train.shape)

print(x_test.shape)

print(y_test.shape)

from sklearn.linear_model import LogisticRegression

lgr=LogisticRegression()

lgr.fit(x_train, y_train)

y_pred=lgr.predict(x_test)

y_pred1=lgr.predict([[129,99,1,0,0,1,0,1]])

print(y_pred)

print(c(y_pred))

print(y_pred1)
```

```
accuracy_score(y_test,y_pred)

confusion_mat = confusion_matrix(y_test,y_pred)

confusion_mat

pickle.dump(lgr,open('CKD.pkl','wb'))
```

TRAINING THE MODEL ON IBM:

```
import pandas as pd

import numpy as mp

from collections import Counter as c

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.metrics import accuracy_score, confusion_matrix

from sklearn.model_selection import train_test_split

from sklearn.preprocessing import LabelEncoder

from sklearn.linear_model import LogisticRegression

import pickle


import os, types

import pandas as pd

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='8spio8rnxJxnzQCZqCY91nFWkd5pcolpt-ewKwuL3Ztj',

    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 = 'ckddetection-donotdelete-pr-yhimgpvl3j9jee'

object_key = 'chronickidneydisease.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 )


data = pd.read_csv(body)

data.head()


data.head(5)

data.tail(5)


data.drop(["id"],axis=1,inplace=True)


data.columns
data.columns=['age', 'blood_pressure', 'specific_gravity', 'albumin', 'sugar', 'red_blood_cells', 'pus_cell', 'pus_cell_clumps', 'bacteria', 'blood glucose random', 'blood_urea',

       'serum_creatinine', 'sodium', 'potassium', 'hemoglobin', 'packed_cell_volume', 'white_blood_cell_count', 'red_blood_cell_count', 'hypertension', 'diabetesmellitus', 'coronary_artery_disease',

       'appetite', 'pedal_edema', 'anemia', 'class']



data.columns

data.info()



data['class'].unique()

data['class']=data['class'].replace("ckd\t","ckd")

data['class'].unique()




# ## Categorical Columns

catcols=set(data.dtypes[data.dtypes=='O'].index.values)

print(catcols)



    print("Columns :",i)

    print(c(data[i]))

    print('*'*120+'\n')



catcols.remove('red_blood_cell_count')

catcols.remove('packed_cell_volume')

catcols.remove('white_blood_cell_count')

print(catcols)
```

```
# ## Numerical Columns

contcols=set(data.dtypes[data.dtypes!='O'].index.values)

print(contcols)

for i in contcols:

    print("Continuous Columns :",i)

    print(c(data[i]))

    print('*'*120+'\n')


contcols.remove('specific_gravity')

contcols.remove('albumin')

contcols.remove('sugar')

print(contcols)


contcols.add('red_blood_cell_count')

contcols.add('packed_cell_volume')

contcols.add('white_blood_cell_count')

print(contcols)


catcols.add('specific_gravity')

catcols.add('albumin')

catcols.add('sugar')

print(catcols)


data['coronary_artery_disease']=data.coronary_artery_disease.replace('\tno','no')

c(data['coronary_artery_disease'])
```

```
data['diabetesmellitus']=data.diabetesmellitus.replace(to_replace={'\tno':'no','\tyes':'yes','yes':'
yes'})

c(data['diabetesmellitus'])


data.isnull().any()

data.isnull().sum()


data.packed_cell_volume = pd.to_numeric(data.packed_cell_volume, errors='coerce')

data.white_blood_cell_count = pd.to_numeric(data.white_blood_cell_count, errors='coerce')

data.red_blood_cell_count = pd.to_numeric(data.red_blood_cell_count, errors='coerce')



data['blood glucose random'].fillna(data['blood glucose random'].mean(),inplace=True)

data['blood_pressure'].fillna(data['blood_pressure'].mean(),inplace=True)

data['blood_urea'].fillna(data['blood_urea'].mean(),inplace=True)

data['hemoglobin'].fillna(data['hemoglobin'].mean(),inplace=True)

data['packed_cell_volume'].fillna(data['packed_cell_volume'].mean(),inplace=True)

data['potassium'].fillna(data['potassium'].mean(),inplace=True)

data['red_blood_cell_count'].fillna(data['red_blood_cell_count'].mean(),inplace=True)

data['serum_creatinine'].fillna(data['serum_creatinine'].mean(),inplace=True)

data['sodium'].fillna(data['sodium'].mean(),inplace=True)

data['white_blood_cell_count'].fillna(data['white_blood_cell_count'].mean(),inplace=True)



data['age'].fillna(data['age'].mode()[0],inplace=True)

data['hypertension'].fillna(data['hypertension'].mode()[0],inplace=True)
```

```
data['pus_cell_clumps'].fillna(data['pus_cell_clumps'].mode()[0],inplace=True)

data['appetite'].fillna(data['appetite'].mode()[0],inplace=True)

data['albumin'].fillna(data['albumin'].mode()[0],inplace=True)

data['pus_cell'].fillna(data['pus_cell'].mode()[0],inplace=True)

data['red_blood_cells'].fillna(data['red_blood_cells'].mode()[0],inplace=True)

data['coronary_artery_disease'].fillna(data['coronary_artery_disease'].mode()[0],inplace=True
)

data['bacteria'].fillna(data['bacteria'].mode()[0],inplace=True)

data['anemia'].fillna(data['anemia'].mode()[0],inplace=True)

data['sugar'].fillna(data['sugar'].mode()[0],inplace=True)

data['diabetesmellitus'].fillna(data['diabetesmellitus'].mode()[0],inplace=True)

data['pedal_edema'].fillna(data['pedal_edema'].mode()[0],inplace=True)

data['specific_gravity'].fillna(data['specific_gravity'].mode()[0],inplace=True)


data.isnull().any()


for i in catcols:

    print("Label Encoding of: ",i)

    LEi = LabelEncoder()

    print(c(data[i]))

    data[i]=LEi.fit_transform(data[i])

    print(c(data[i]))

    print("*"*100)
```

```
selcols=['red_blood_cells','pus_cell','blood glucose
random','blood_urea','pedal_edema','anemia','diabetesmellitus','coronary_artery_disease']

x=pd.DataFrame(data,columns=selcols)

y=pd.DataFrame(data,columns=['class'])

print(x.shape)

print(y.shape)

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=2)

print(x_train.shape)

print(y_train.shape)

print(x_test.shape)

print(y_test.shape)

from sklearn.linear_model import LogisticRegression

lgr=LogisticRegression()

lgr.fit(x_train, y_train

y_pred=lgr.predict(x_test)

y_pred1=lgr.predict([[129,99,1,0,0,1,0,1]])

print(y_pred)

print(c(y_pred))

print(y_pred1)

accuracy_score(y_test,y_pred)

confusion_mat = confusion_matrix(y_test,y_pred)

confusion_mat



get_ipython().system(' pip install -U ibm-watson-machine-learning')

from ibm_watson_machine_learning import APIClient
```

```
wml_credentials = {

    'apikey': "a1cVu7bGrEpjyRSOvDbNDF8tBZ89tU9aaQ3UjK-l8Nbg",

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

}

wml_client = APIClient(wml_credentials)

wml_client.spaces.list()


space_id = "97486295-15b7-4879-81d2-eceeac8158b5"

wml_client.set.default_space(space_id)

wml_client.software_specifications.list()


model_name="CKD_model"

deployment = "CKD_deploy_model"

model_deploy = lgr


software_spec_uid = wml_client.software_specifications.get_uid_by_name("runtime-22.1-
py3.9")

=software_spec_uid


=model_props = {

    wml_client.repository.ModelMetaNames.NAME:model_name,

wml_client.repository.ModelMetaNames.TYPE:"scikit-learn_1.0",

wml_client.repository.ModelMetaNames.SOFTWARE_SPEC_UID:software_spec_uid

}

model_details = wml_client.repository.store_model(

    model = model_deploy,
    meta_props = model_props,

    training_data = x_train,

    training_target = y_train)

model_details
```

HOMEPAGE HTML CODE:

```
<!DOCTYPE html>

<html lang="en">

  <head>

    <meta charset="utf-8" />

    <meta name="viewport" content="width=device-width, initial-scale=1" />

    <title>Chronic Kidney Disease Prediction using Maachine Learning</title>

  </head>

  <body>

    <style>

      body {

        min-height: 100%;

        background: linear-gradient(

            0deg,

            rgba(0, 0, 0, 0),

            rgba(78, 5, 48, 0.3)

          ),

          url(https://d3b6u46udi9ohd.cloudfront.net/wp-
content/uploads/2022/05/24051731/Kidney-1.jpg);

        background-size: cover;

        background-repeat: no-repeat;
        background-attachment: fixed;

        background-size: 100% 100%;

      }

    </style>

    <br />


    <table id="header" border="1" width="100%" cellpadding="0" cellspacing="0">

      <tr>

        <td>

          <table

            border="0"

            cellpadding="10"

            cellspacing="0"

            width="80%"

            align="center"

          >

            <tr>

              <td>

                <a href="/Home" style="text-decoration: none"

                  ><font

                    face="'Monaco'"

                    style="font-weight: bold"

                    size="4"

                    color="white"

                    >Home Page</font
        ></a
      >
    </td>
    <td>' '</td>
    <td>' '</td>
    <td>' '</td>
    <td>' '</td>
    <td>' '</td>
    <td>' '</td>
    <td>' '</td>
    <td>' '</td>
    <td>' '</td>
    <td>' '</td>
    <td>' '</td>
    <td>' '</td>
    <td>
      <a style="text-decoration: none"
        ><font
          face="'Monaco'"
          style="font-weight: bold"
          size="4"
          color="white"
        >
        </font
      ></a>
```

```
          </td>
          <td>
            <a href="/Prediction" style="text-decoration: none"
              ><font
                face="'Monaco'"
                style="font-weight: bold"
                size="4"
                color="white"
                >Prediction</font
              ></a
            >
          </td>
        </tr>
      </table>
    </td>
  </tr>
</table>
<br />
<br />
<br />
<br />
<br />
<br />
<br />
<h1 align="center">
```

```
      <b>
        <font face="Monaco" color="white" size="35"
          >Chronic Kidney Disease Prediction using Machine Learning</font
        ></b
      >
    </h1>
    <br /><br />
    <hr />
  </body>
</html>
```

INDEX PAGE HTML CODE

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <title>Chronic Kidney Disease Prediction using Machine Learning</title>
  </head>
  <body>
    <style>
      body {
        min-height: 100%;
        background: linear-gradient(
            0deg,
            rgba(0, 0, 0, 0),
        rgba(173, 216, 230, 0.3)
      ),
      url();
    background-size: cover;
    background-repeat: no-repeat;
    background-attachment: fixed;
    background-size: 100% 100%;
  }
</style>
<br />
<h4 align="center">
  <b>
    <font face="Monaco" color="darkblue" size="20"
      >Chronic Kidney Disease Prediction using Machine Learning</font>
    </b >
</h4>
</div>
<br /><br />
<form action="{{ url_for('predict') }}" class="predict" method="POST"><br>
<div style="font-size: x-large; text-align: center;">
    <label for="burea">Enter your Blood urea:</label>
    <input type="number" id="fname" name="fname"><br><br>
    <label for="bglucose">Enter your Blood Glucose Random:</label>
    <input type="number" id="bglucose" name="bglucose"><br><br>
    <label for="anemia">Select Anemia or not :</label>
```

```
<select name="anemia" id="anemia">

<option value="1">Yes</option>

<option value="0">No</option><br><br>

</select><br><br>

<label for="Coronary">Select Coronary Artery Disease or not :</label>

<select name="Coronary" id="Coronary">

<option value="1">Yes</option>

<option value="0">No</option><br><br>

</select><br><br>

<label for="cell">Select Pus Cell Normal or Abnormal :</label>

<select name="cell" id="cell">

<option value="1">Normal</option>

<option value="0">Abnormal</option><br><br>

</select><br><br>

<label for="rbc">Select Red Blood Cell Level Normal or Abnormal :</label>

<select name="rbc" id="rbc">

<option value="1">Normal</option>

<option value="0">Abnormal</option><br><br>

</select><br><br>

<label for="dia">Select Diabetes Mellitus or not :</label>

<select name="dia" id="dia">

<option value="1">Yes</option>

<option value="0">No</option><br><br>

</select><br><br>

<label for="enema">Select Pedal Enema or not :</label>
```

```
    <select name="enema" id="enema">
    <option value="1">Yes</option>
    <option value="0">No</option><br><br>
    </select><br><br>
    <button>Submit</button>
  </div>
  </form>
  <hr />
 </body>
</html>
```

PREDICTION YES PAGE HTML CODE

```
<!DOCTYPE html>
<html lang="en">
 <head>
  <meta charset="utf-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1" />
  <title>Chronic Kidney Disease Prediction using Machine Learning</title>
 </head>
 <body>
  <style>
   body {
    min-height: 100%;
    background: linear-gradient(
      0deg,
      rgba(0, 0, 0, 0),
        rgba(255, 204, 203, 0.3)
      ),
      url();
    background-size: cover;
    background-repeat: no-repeat;
    background-attachment: fixed;
    background-size: 100% 100%;
  }
</style>
<br />
<div
  id="demobox"
  style="
    background-color: darkred;
    padding: 10px;
    border: 1px solid green;">
  <h4 style="color: white; text-align: center;">
  </h4>
</div>
<h4 align="center">
  <b>
    <font face="Monaco" color="darkred" size="20"
      >Chronic Kidney Disease Prediction using Machine Learning</font
    ></b >
</h4>
```

```
    </div>
    <br /><br />
    <div style="font-size: xx-large; text-align: center;">
        <b>Prediction: You are at risk of Chronic Kidney Disease! Please consult a doctor immediately!!</b>
    </div>
    <hr />
  </body>
</html>
```

PREDICTION NO PAGE HTML CODE

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <title>Chronic Kidney Disease Prediction using Machine Learning</title>
  </head>
  <body>
    <style>
      body {
        min-height: 100%;
        background: linear-gradient(
            0deg,
            rgba(0, 0, 0, 0),
            rgba(144, 238, 144, 0.3)
      ),
      url();
    background-size: cover;
    background-repeat: no-repeat;
    background-attachment: fixed;
    background-size: 100% 100%;
  }
</style>
<br />
<div
  id="demobox"
  style="
    background-color: darkgreen;
    padding: 10px;
    border: 1px solid green;">
  <h4 style="color: white; text-align: center;">
  </h4>
</div>
<h4 align="center">
  <b>
    <font face="Monaco" color="darkgreen" size="20"
      >Chronic Kidney Disease Prediction using Machine Learning</font
    ></b >
</h4>
</div>
```

```
    <br /><br />

    <div style="font-size: xx-large; text-align: center;">

        <b>Prediction: You are not at risk of Chronic Kidney Disease! Keep up your good
health!</b>

    </div>

    <hr />

  </body>

</html>
```

FLASK (app.py) CODE:

```
import requests

import json

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

API_KEY = "a1cVu7bGrEpjyRSOvDbNDF8tBZ89tU9aaQ3UjK-l8Nbg"

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}


import numpy as np


import pandas as pd


from flask import Flask,request,render_template


import pickle as pk
app=Flask(__name__)


model=pk.load(open('CKD.pkl','rb'))


@app.route('/')
```

```
def home():

    return render_template('homepage.html')

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

def prediction():

                    return render_template('indexpage.html')

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

def my_home():

                    return render_template('homepage.html')

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

def predict():

    payload_scoring = {"input_data": [{"field":
[['blood_urea','blood_glucose_random','coronary_artery_disease','anemia','pus_cell','red_bloo
d_cells','diabetesmellitus','pedal_edema']], "values": [input_features]}]}

    response_scoring = requests.post('https://us-
south.ml.cloud.ibm.com/ml/v4/deployments/569b99a7-fab5-493e-906c-
1efa49a46dc4/predictions?version=2022-11-17', json=payload_scoring,

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

    print("Scoring response")

    predictions=response_scoring.json()

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

    if(pred==1):

        return render_template('predictionNo.html')
    else:

        return render_template('predictionYes.html')

if __name__ == '__main__':

    app.run(debug=False)
```

13.2 GIT REPO LINK AND PROJECT DEMO LINK

Git Repo: https://github.com/IBM-EPBL/IBM-Project-42119-1660650072

Project demo video link: https://youtu.be/nDAlhi5cPxM