EARLY DETECTION OF CHRONIC KIDNEY DISEASE USING MACHINE LEARNING

TEAM ID: PNT2022TMID25581

IBM-PROJECT-53532-1661414687

NALAYATHIRAN PROJECT REPROT BY

PRITHICA G -210919104034

ANUJHA R -210919104003

ANUPRIYA S -210919104004

NITHIYA N - 210919104031

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

LOYOLA INSTITUTE OF TECHNOLOGY, CHENNAI-125

ANNA UNIVERSITY,CHENNAI

CONTENTS

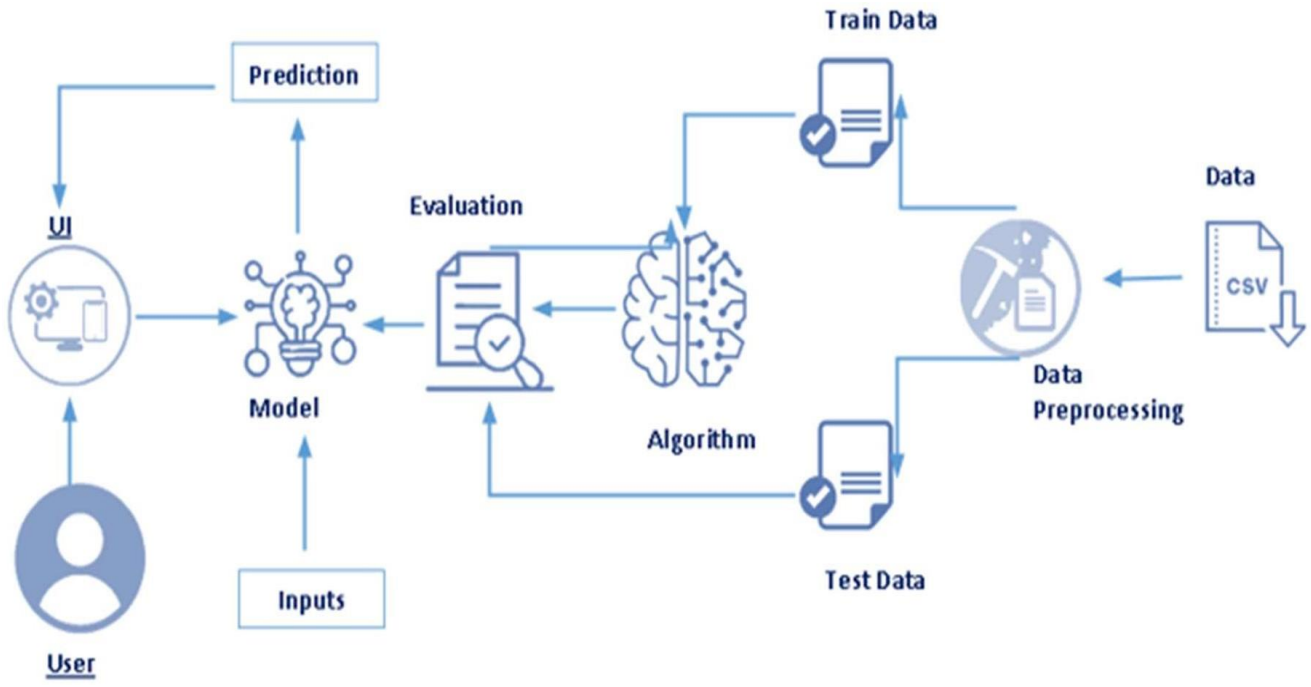
1.0 Introduction:

Chronic kidney disease prediction is one of the most important issues in health care-analytics. The most interesting and challenging tasks in day-to-day lives as one third of adult population is affected by chronic kidney disease (CKD), and millions die each year because they do not have access to affordable treatment. Chronic Kidney Disease can be cured, if treated in the early stages. The main aim of the project is to predict whether the patient have chronic kidney disease of not in a painless, accurate and faster way based on certain diagnostic measurement like Blood Pressure(BP), Albumin(Al) etc., and then appropriate treatment can be given based on the details provided by the model.

2.0 Skills Required:

Python, Python Web Frame works, Python For data Visualization, Data Preprocessing Techniques, Machine Learning, IBM Cloud, IBM Watson Studio, Python-Flask

3.0 Architecture:



In this píoject we have followed the entiíe softwaíe development life cycle píocess using agile methodology.

4.0 LITERATURE SURVEY:

INTRODUCTION:-

There are a large number of people in the world who gets affected to kidney related diseases. Our country India ranks first in terms of peoplegetting affected by kidney related disease. Though population plays a major factorin it, the fact that China which has higher population than us is actually ranking ina much lower place than us in the term of people getting affected by kidney relateddisease. Generally, the kidney diseases are caused by the abnormal physiological functionalities of human kidney. Therefore, the characteristic symptoms are generated based on the differentiation between normal physiological functionalitiesand abnormal physiological functionalities of the kidney. The difference in the functionality of kidney is mainly due to the lifestyle and the food preference peoplehave in their life. So the reason we were doing the is project is to find the symptoms of kidney related diseases for the users and alert them before the disease becomesfatal for the patient. The main motive of this project is to find the symptom and itscure as soon as possible.

Literature Review:-

[1]

DATE OF PUBLICATION	PAPER NAME	AUTHOR NAME	LINK
February 12, 2019	Detailed Review of Chronic Kidney Disease	Yesubabu Kakitapallia, Janakiram Ampolua, Satya Dinesh Madasa, M.L.S. Sai Kumar	https://www.karger.com /Article/Pdf/504622

The proposed method is used to explain about the effects of kidney diseases. In this method Kidney damage can be assessed by albumin creatinine rate (ACR); albuminuria is one of the identifiers of kidney function in a timed urine collection. It was stated that one of the reasons for CKD, i.e., excretion of protein uria, which is due to the intake of cooked meat or increased intake of protein or any kidney infection.

[2]

DATE OF PUBLICATION	PAPER NAME	AUTHOR NAME	LINK
October 1, 2019	Chronic Kidney Disease Diagnosis and Management	Teresa K. Chen, MD, MHS, Daphne H. Knicely, MD, and Morgan E Grams, MD, PhD	https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7015670/

The proposed system uses CKD method for diagnosis. In this method once adiagnosis of CKD has been made, the next step is to determine staging, whichis based on GFR, albuminuria, and cause of CKD . Staging of GFR is classifiedas G1 (GFR ≥90 ml/min/1.73 m^2), G2 (GFR 60–89 ml/min/1.73 m^2), G3a (45–59 ml/min/1.73 m^2), G3b (30–44 ml/min/1.73 m^2), G4 (15–29 ml/min/1.73 m^2), and G5 (<15 ml/min/1.73 m^2).

[3]

DATE OF PUBLICATION	PAPER NAME	AUTHOR NAME	LINK
27 November 2013	Risk factors for chronic kidney disease: an update	Rumeyza Kazancioglu	https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4089662/

This system explains about inheritance in the kidney disease such as Uromodulin. It is another identified mutation is related to APOL1. An autosomal recessive pattern of inheritance is demonstrated and associated witha substantially higher risk of ESRD (10- fold higher risk of ESRD due to focal glomerulosclerosis and 7-fold higher risk of ESRD due to (hypertension).

[4]

DATE OF PUBLICATION	PAPER NAME	AUTHOR NAME	LINK
31 December2014	Systems biology towards novel chronic kidney disease diagnosis and treatment.	Dr. Bernd Mayer	https://cordis.europa.eu/docs/results/241/241544/final1-syskid-final-report-2015-03-13.pdf

This system explains about the hemodynamic in our glomerulor. Alterations in glomerular hemodynamic were considered of upmost importance. Both afferent arteriolar glomerular vasodilatation and efferent vasoconstriction increase intra-glomerular filtration pressure thus leading to hyper filtration, which on the short term stabilizes GFR but on the long term leads to progressing glomerular sclerosis thereby initiating a vicious cycle

[5]

DATE OF PUBLICATION	PAPER NAME	AUTHOR NAME	LINK
June 2020	Chronic kidney disease: prevalence and risk factors.	A.M. Aringazina, O.Zh.Narmanova, G.O. Nuskabaeva, Zh.A. Tagaevaq	www.researchgate.net/publication/342930212_Chronic_kidney_disease_prevalence_and_risk_factors_literature_review

This system explains about the disease risks and affects that has on us and our surrounding. It also explains about the ill factor it might have and chances of spreading to anyone in our surrounding.

[6]

DATE OF PUBLICATION	PAPER NAME	AUTHOR NAME	LINK
July 2011	The Burden of Chronic Kidney Disease on Developing Nations: A 21st Century Challengein Global Health	Nugent R.A, Fathima S.F., Feigl A.B, Chyung D	https://www.karger.com/Article/Fulltext/321382

The proposed system explains about the ill effects these kidney diseases have on developing countries. In developing nations, the growing prevalence of chronic diseases such as chronic kidney disease has severe implications on health and economic output. The rapid rise of common risk factors such as diabetes, hypertension, and obesity, especially among the poor, will result in even greater and more profound burdens that developing nations are not equipped to handle.

[7]

DATE OF PUBLICATION	PAPER NAME	AUTHORNAME	LINK
FEBRAURY ,2020	A Comparative Analysis of Machine Learning Techniques	Alvaro Sobrinho, Andessa C.M. Da Silveira, Leandro Dias da Silva, Evandro de B. Costa	https://www.researchgate.net/publication/339014686_ComputerAided_Diagnosis_of_Chronic_Kidney_Disease_in_Developing_Countries_A_Comparative_Analysis_of_Machine_Learning_Techniques

This system explains about the use of diagnosing with the help of computer. Software systems have been developed to assist physicians during CKD monitoring and diagnosis. For example, CKD-Go1is a web application (app) tohelp users verify their kidney function by inputting their ACR and GF

[8]

DATE OF PUBLICATION	PAPERNAME	AUTHORNAME	LINK
January 2018	Diagnosis of Kidney Conditions Using Low-Cost Paper Diagnostics	Md. Nazibul Islam	https://www.researchgate.net/publication/325499924_Diagnosis_of_Kidney_Conditions_Using_Low-Cost_Paper_Diagnostics

This proposed system uses Low cost paper diagnosis method. The methodPaper diagnostic devices (PADs) can play a vital role in low-cost and rapid diagnosis of kidney condition, resulting in early detection of kidney complications. Paper diagnostics are paper and cellulose based analytical devices capable of qualitative or quantitative detection of biomarker.

[9]

Date OF PUBLICATION	PAPERNAME	AUTHOR NAME	LINK
December 2019	Machine Learning Applied to Kidney Disease Prediction: Comparison Study	Akm Shahariar Azad Rabby, Rezwana Mamata, Monira Akter Laboni, Ohidujjaman	https://www.researchgate.net/publication/338356158_Machine_Learning_Applied_to_Kidney_Disease_Prediction_Comparison_Study

The proposed system uses applied machine language for early detection of kidney disease. The main aim is to find an optimized and efficient machine learning (ML) technique that can effectively recognize and predict the condition of chronic kidney disease. the data has been divided into two sections. In one section train dataset got trained and another section got evaluated by test dataset. The analysis results show that the Decision Tree Classifier and Gaussian Naive Bayes achieved the highest performance than the other classifiers, obtaining the accuracy score of 100% and 1 recall(Sensitivity) score.

[10]

Date OF PUBLICATION	PAPER NAME	AUTHORNAME	LINK
January 2022	Development of a Graphical User Interface Software for The Predictionof Chronic Kidney Disease	S.C. Nwaneri and H.C. Ugo	https://www.ajol.info/index.php/njt/article/view/225270/212516

The proposed system explains about the training and development ofmodel for kidney diagnosing. This process involves building the neural network architecture and determining the activation functions to be used at the output of each layer of the network.The ANN model is built using the Keas libraries in python. Dense is used to assign the number of layers for the network. The ANN model consists of 25 neurons in the input layer and one neuron at the output layer.

As per agile methodology there are various phases in our project.

1. Project Design and Planning Phase
2. Project Development Phase
3. Train the Machine learning model on IBM.

5.0 Project Design and Planning Phase:

In the Píoject Design and planning phase we have

5.0 Ideation phase

5.1 Design Phase I

5.2 Design phase II

5.3 Píoject planning phase

5.1 IDEATION PHASE

During the ideation phase we have done problem definition, Empathy map canvas and Brainstorming and ideation process. Let us see in detail each activity.

5.1.1 PROBLEM DEFINITION

Team ID : PNT2022TMID25581

PROJECT NAME : EARLY DETECTION OF CHRONIC KIDNEY DISEASE

PROBLEM STATEMENT

EARLY DETECTION OF CHRONIC KIDNEY DISEASE USING MACHINE LEARNING

- Irregular lifestyle of current days leads to many disease chronic diseases. One of them being Chronic Kidney Disease(CKD), it has been of a growing concern, kidney is one of the most important organs in the body required for filtering blood, once a person has lost their kidneys, they could survive only for 18 days without their kidneys, it would take a fortune to just keep the person alive, with treatments like dialysis, transplant etc.

- 10% of the population worldwide is affected by chronic kidney disease (CKD), and millions die each year because they do not have access to affordable treatment.

- People usually don't realize that the medical tests we perform for various purposes can contain valuable information related to kidney disease. Subsequently, the attributes of various medical tests are examined to distinguish which attributes may contain useful information about the disease. The information, they say, helps us gauge the severity of the problem, and we use that information to build a machine learning model that predicts chronic kidney disease.

- Early detection of kidney disease can help in treatment which could save lives. Analyzing various medical tests, would give us an idea about which attributes help us distinguish the disease.

- The main aim of this project is to predict whether the patient have chronic kidney disease or not, in more accurate and faster way based on certain diagnostic measurements like Blood Pressure (BP), Albumin(Al).

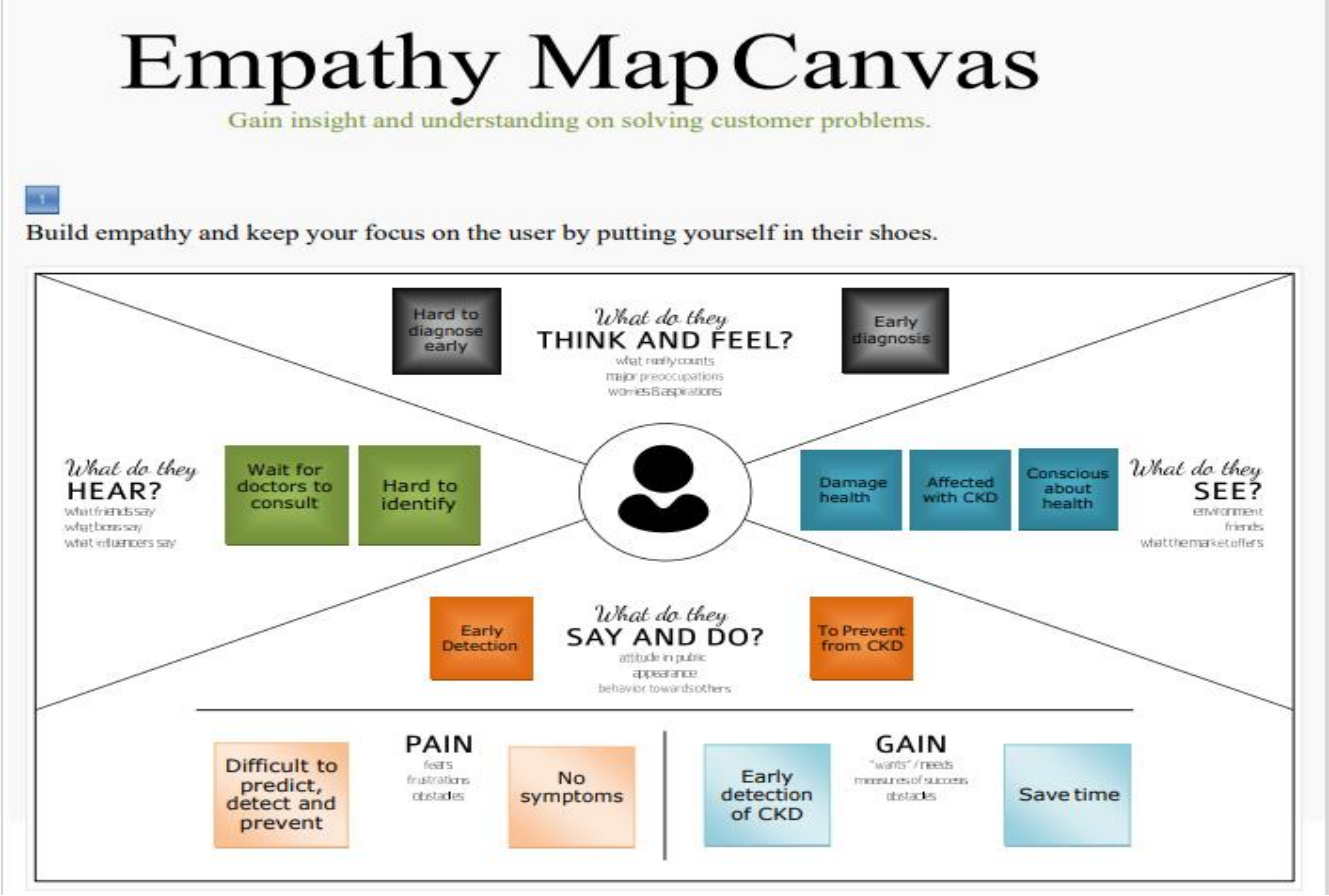
5.1.2 EMPATHY MAP CANVAS

An empathy map is a collaborative visualization used to articulate what we know about a particular type of user. It externalizes knowledge about users in older to

1) Create a shared understanding of user needs, and

2) Aid in decision making.

An empathy map helps to map what a design team knows about the potential audience. This tool helps to understand the reason behind some actions a user takes deeply. This tool helps build Empathy towards users and helps design teams shift focus from the product to the users who aíe going to use the product.



5.1.3 SOLUTION ARCHITECTURE

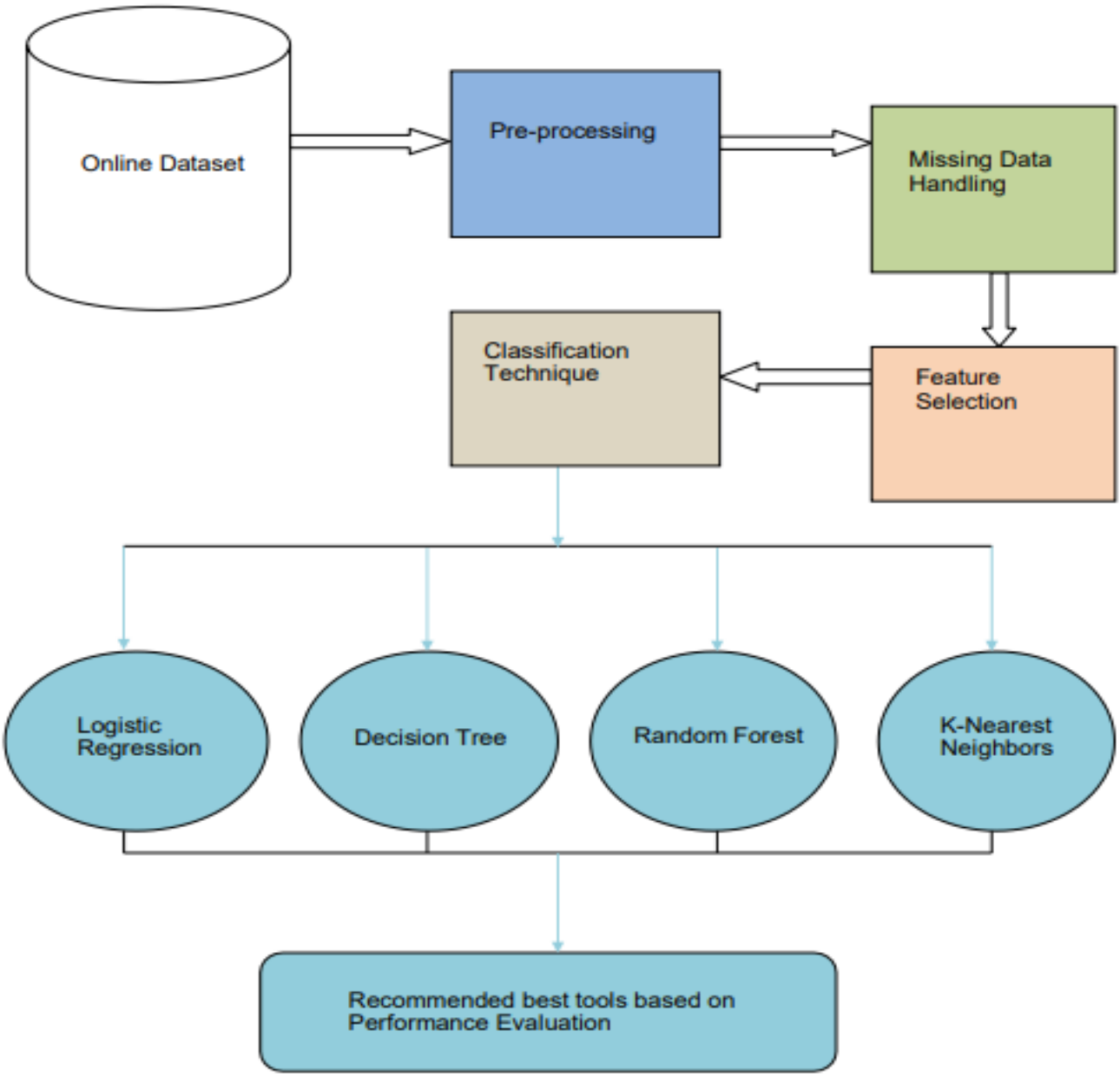
Solution Architecture:

Solution architecture is a complex process with many sub processes that bridges the gap between business problems and technology solutions.

Its goals are to:

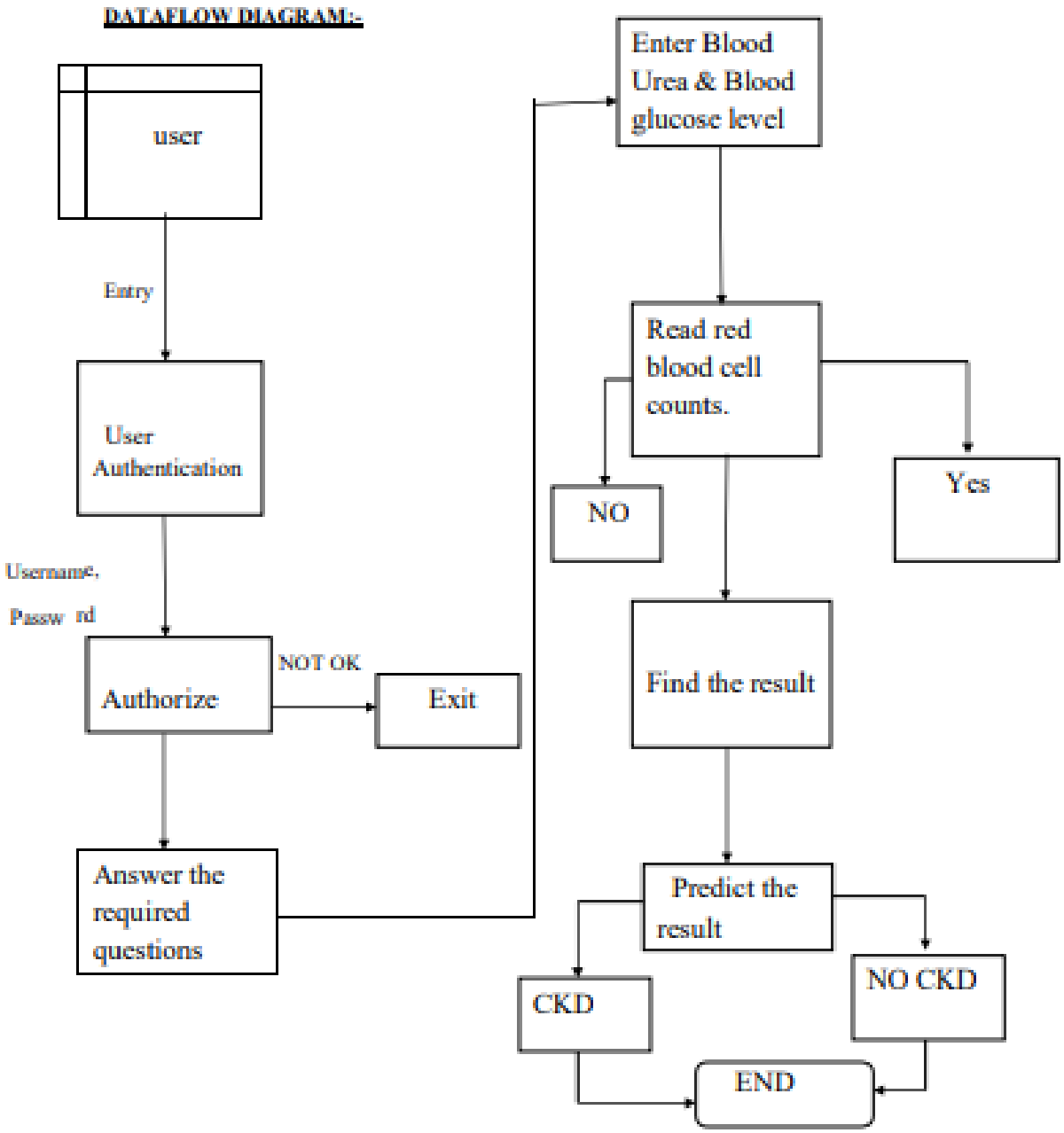
- Find the best tech solution to solve existing business problems.
- Describe the structure, characteristics, behavior, and other aspects of the software to project stakeholders.
- Define features, development phases, and solution requirements.
- Provide specifications according to which the solution is defined, managed, and delivered.

ARCHITECTURE DIAGRAM:



5.2 DATA FLOW DIAGRAM AND USER STORIES

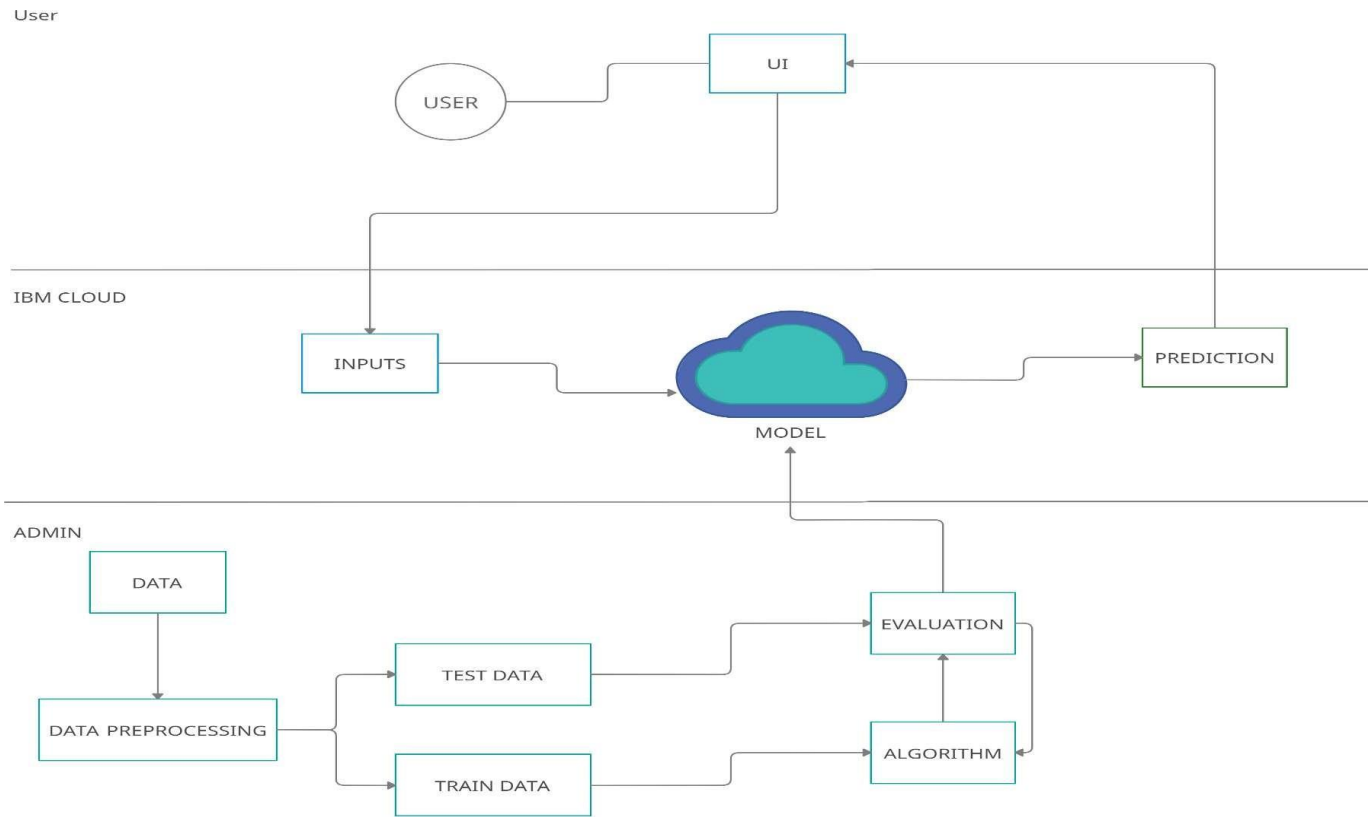
TEAM ID: PNT2022TMID25581



5.3TECHNICAL ARCHICTECTURE

Technology Architecture describe the logical software and hardware capabilities that are required to support the deployment of business, data, and application services. This includes IP infrastructure, middleware, networks, communications, processing, standards, etc.

Technology architecture deals with the deployment of application components on technology components. A standard set of predefined technology components is provided in order to represent servers, network, workstations.



6.0 CODING

In [1]:
```
import pandas as pd
```

In [2]:
```
import numpy as np
```

In [3]:
```
from collections import Counter as c
```

In [4]:
```
import matplotlib.pyplot as plt
```

In [5]:
```
import seaborn as sns
```

In [6]:
```
import missingno as msno
```

In [7]:
```
from sklearn.metrics import accuracy_score,confusion_matrix
```

In [8]:
```
from sklearn.model_selection import train_test_split
```

In [9]:
```
from sklearn.preprocessing import LabelEncoder
```

In [10]:
```
from sklearn.linear_model import LogisticRegression
```

In [11]:
```
import pickle
```

In [12]:
```
data=pd.read_csv("F:\SEM 7\IBM Project\kidney_disease.csv")
```

In [13]:
```
data.head()
```

Out[13]:

	id	age	bp	sg	al	su	rbc	pc	pcc	ba	...	pcv	wc	rc	htn	dm	cad	appet	pe	ane	classification
0	0	48.0	80.0	1.020	1.0	0.0	NaN	normal	notpresent	notpresent	...	44	7800	5.2	yes	yes	no	good	no	no	ckd
1	1	7.0	50.0	1.020	4.0	0.0	NaN	normal	notpresent	notpresent	...	38	6000	NaN	no	no	no	good	no	no	ckd
2	2	62.0	80.0	1.010	2.0	3.0	normal	normal	notpresent	notpresent	...	31	7500	NaN	no	yes	no	poor	no	yes	ckd
3	3	48.0	70.0	1.005	4.0	0.0	normal	abnormal	present	notpresent	...	32	6700	3.9	yes	no	no	poor	yes	yes	ckd
4	4	51.0	80.0	1.010	2.0	0.0	normal	normal	notpresent	notpresent	...	35	7300	4.6	no	no	no	good	no	no	ckd

5 rows × 26 columns

In [14]:
```
data.tail()
```

Out[14]:

	id	age	bp	sg	al	su	rbc	pc	pcc	ba	...	pcv	wc	rc	htn	dm	cad	appet	pe	ane	classification
395	395	55.0	80.0	1.020	0.0	0.0	normal	normal	notpresent	notpresent	...	47	6700	4.9	no	no	no	good	no	no	notckd
396	396	42.0	70.0	1.025	0.0	0.0	normal	normal	notpresent	notpresent	...	54	7800	6.2	no	no	no	good	no	no	notckd
397	397	12.0	80.0	1.020	0.0	0.0	normal	normal	notpresent	notpresent	...	49	6600	5.4	no	no	no	good	no	no	notckd
398	398	17.0	60.0	1.025	0.0	0.0	normal	normal	notpresent	notpresent	...	51	7200	5.9	no	no	no	good	no	no	notckd
399	399	58.0	80.0	1.025	0.0	0.0	normal	normal	notpresent	notpresent	...	53	6800	6.1	no	no	no	good	no	no	notckd

5 rows × 26 columns

In [15]:
```
data.head(10)
```

Out[15]:

	id	age	bp	sg	al	su	rbc	pc	pcc	ba	...	pcv	wc	rc	htn	dm	cad	appet	pe	ane	classification
0	0	48.0	80.0	1.020	1.0	0.0	NaN	normal	notpresent	notpresent	...	44	7800	5.2	yes	yes	no	good	no	no	ckd
1	1	7.0	50.0	1.020	4.0	0.0	NaN	normal	notpresent	notpresent	...	38	6000	NaN	no	no	no	good	no	no	ckd
2	2	62.0	80.0	1.010	2.0	3.0	normal	normal	notpresent	notpresent	...	31	7500	NaN	no	yes	no	poor	no	yes	ckd
3	3	48.0	70.0	1.005	4.0	0.0	normal	abnormal	present	notpresent	...	32	6700	3.9	yes	no	no	poor	yes	yes	ckd
4	4	51.0	80.0	1.010	2.0	0.0	normal	normal	notpresent	notpresent	...	35	7300	4.6	no	no	no	good	no	no	ckd
5	5	60.0	90.0	1.015	3.0	0.0	NaN	NaN	notpresent	notpresent	...	39	7800	4.4	yes	yes	no	good	yes	no	ckd
6	6	68.0	70.0	1.010	0.0	0.0	NaN	normal	notpresent	notpresent	...	36	NaN	NaN	no	no	no	good	no	no	ckd
7	7	24.0	NaN	1.015	2.0	4.0	normal	abnormal	notpresent	notpresent	...	44	6900	5	no	yes	no	good	yes	no	ckd
8	8	52.0	100.0	1.015	3.0	0.0	normal	abnormal	present	notpresent	...	33	9600	4	yes	yes	no	good	no	yes	ckd
9	9	53.0	90.0	1.020	2.0	0.0	abnormal	abnormal	present	notpresent	...	29	12100	3.7	yes	yes	no	poor	no	yes	ckd

10 rows × 26 columns

In [17]:
```
data.head(10)
```

Out[17]:

	age	bp	sg	al	su	rbc	pc	pcc	ba	bgr	...	pcv	wc	rc	htn	dm	cad	appet	pe	ane	classification
0	48.0	80.0	1.020	1.0	0.0	NaN	normal	notpresent	notpresent	121.0	...	44	7800	5.2	yes	yes	no	good	no	no	ckd
1	7.0	50.0	1.020	4.0	0.0	NaN	normal	notpresent	notpresent	NaN	...	38	6000	NaN	no	no	no	good	no	no	ckd
2	62.0	80.0	1.010	2.0	3.0	normal	normal	notpresent	notpresent	423.0	...	31	7500	NaN	no	yes	no	poor	no	yes	ckd
3	48.0	70.0	1.005	4.0	0.0	normal	abnormal	present	notpresent	117.0	...	32	6700	3.9	yes	no	no	poor	yes	yes	ckd
4	51.0	80.0	1.010	2.0	0.0	normal	normal	notpresent	notpresent	106.0	...	35	7300	4.6	no	no	no	good	no	no	ckd
5	60.0	90.0	1.015	3.0	0.0	NaN	NaN	notpresent	notpresent	74.0	...	39	7800	4.4	yes	yes	no	good	yes	no	ckd
6	68.0	70.0	1.010	0.0	0.0	NaN	normal	notpresent	notpresent	100.0	...	36	NaN	NaN	no	no	no	good	no	no	ckd
7	24.0	NaN	1.015	2.0	4.0	normal	abnormal	notpresent	notpresent	410.0	...	44	6900	5	no	yes	no	good	yes	no	ckd
8	52.0	100.0	1.015	3.0	0.0	normal	abnormal	present	notpresent	138.0	...	33	9600	4	yes	yes	no	good	no	yes	ckd
9	53.0	90.0	1.020	2.0	0.0	abnormal	abnormal	present	notpresent	70.0	...	29	12100	3.7	yes	yes	no	poor	no	yes	ckd

10 rows × 25 columns

In [18]:
```
data.columns
```

Out[18]:
```
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')
```

In [19]:
```
data.columns=['age','blood_pressure','specific_gravity','albumin','sugar','red_blood_cells','pus_cell','pus_cell_clumps','bacteria','bloo
```

In [20]:
```
data.columns
```

Out[20]:
```
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')
```

In [21]:
```
data.columns
```

Out[21]:
```
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')
```

In [22]:
```
data.info()
```

```
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
```

In [23]:
```
data['class'].unique()
```

Out[23]:
```
array(['ckd', 'ckd\t', 'notckd'], dtype=object)
```

In [24]:
```
data['class']=data['class'].replace("ckd\t","ckd")
data['class'].unique()
```

Out[24]:
```
array(['ckd', 'notckd'], dtype=object)
```

In [27]:

```
catcols.remove('red_blood_cell_count')
catcols.remove('packed_cell_volume')
catcols.remove('white_blood_cell_count')
print(catcols)
```

```
{'anemia', 'appetite', 'pus_cell_clumps', 'class', 'bacteria', 'hypertension', 'pus_cell', 'pedal_edema', 'red_blood_cells', 'coronary_ar
tery_disease', 'diabetesmellitus'}
```

In [28]:

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

```
{'albumin', 'sugar', 'serum_creatinine', 'blood glucose random', 'potassium', 'sodium', 'blood_pressure', 'blood_urea', 'hemoglobin', 'sp
ecific_gravity', 'age'}
```

In [29]:

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

```
Columns: albumin
Counter({0.0: 199, 1.0: 44, 2.0: 43, 3.0: 43, 4.0: 24, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, na
n: 1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, na
n: 1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, na
n: 1, nan: 1, nan: 1})
************************************************************************************************************************

Columns: sugar
Counter({0.0: 290, 2.0: 18, 3.0: 14, 4.0: 13, 1.0: 13, 5.0: 3, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, na
n: 1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, na
n: 1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, na
n: 1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1})
************************************************************************************************************************

Columns: serum_creatinine
Counter({1.2: 40, 1.1: 24, 1.0: 23, 0.5: 23, 0.7: 22, 0.9: 22, 0.6: 18, 0.8: 17, 2.2: 10, 1.5: 9, 1.7: 9, 1.3: 8, 1.6: 8, 1.8: 7, 1.4: 7,
2.5: 7, 2.8: 7, 1.9: 6, 2.7: 5, 2.1: 5, 2.0: 5, 3.2: 5, 3.3: 5, 3.9: 4, 7.3: 4, 4.0: 3, 2.4: 3, 3.4: 3, 2.9: 3, 5.3: 3, 2.3: 3, 7.2: 2,
4.6: 2, 4.1: 2, 5.2: 2, 6.3: 2, 3.0: 2, 6.1: 2, 6.7: 2, 5.6: 2, 6.5: 2, 4.4: 2, 6.0: 2, 3.8: 1, 24.0: 1, 9.6: 1, 76.0: 1, 7.7: 1, nan: 1,
10.8: 1, 5.9: 1, 3.25: 1, nan: 1, 9.7: 1, 6.4: 1, 32.0: 1, nan: 1, nan: 1, 8.5: 1, 15.0: 1, 3.6: 1, 10.2: 1, 11.5: 1, nan: 1, 12.2: 1, 9.
2: 1, 13.8: 1, 16.9: 1, 7.1: 1, 18.0: 1, 13.0: 1, 48.1: 1, 14.2: 1, 16.4: 1, nan: 1, nan: 1, 2.6: 1, 7.5: 1, 4.3: 1, 18.1: 1, 11.8: 1, 9.
3: 1, 6.8: 1, 13.5: 1, nan: 1, 12.8: 1, 11.9: 1, nan: 1, nan: 1, nan: 1, 12.0: 1, nan: 1, 13.4: 1, 15.2: 1, 13.3: 1, nan: 1, nan: 1, nan:
1, nan: 1, nan: 1, 0.4: 1})
************************************************************************************************************************

Columns: blood glucose random
Counter({99.0: 10, 100.0: 9, 93.0: 9, 107.0: 8, 117.0: 6, 140.0: 6, 92.0: 6, 109.0: 6, 131.0: 6, 130.0: 6, 70.0: 5, 114.0: 5, 95.0: 5, 12
3.0: 5, 124.0: 5, 102.0: 5, 132.0: 5, 104.0: 5, 125.0: 5, 122.0: 5, 121.0: 4, 106.0: 4, 76.0: 4, 91.0: 4, 129.0: 4, 133.0: 4, 94.0: 4, 8
8.0: 4, 118.0: 4, 139.0: 4, 111.0: 4, 113.0: 4, 120.0: 4, 119.0: 4, 74.0: 3, 108.0: 3, 171.0: 3, 137.0: 3, 79.0: 3, 150.0: 3, 112.0: 3, 1
27.0: 3, 219.0: 3, 172.0: 3, 89.0: 3, 128.0: 3, 214.0: 3, 105.0: 3, 78.0: 3, 103.0: 3, 82.0: 3, 97.0: 3, 81.0: 3, 138.0: 2, 490.0: 2, 20
8.0: 2, 98.0: 2, 204.0: 2, 207.0: 2, 144.0: 2, 253.0: 2, 141.0: 2, 86.0: 2, 360.0: 2, 163.0: 2, 158.0: 2, 165.0: 2, 169.0: 2, 210.0: 2, 1
01.0: 2, 153.0: 2, 213.0: 2, 424.0: 2, 303.0: 2, 192.0: 2, 80.0: 2, 110.0: 2, 96.0: 2, 85.0: 2, 83.0: 2, 75.0: 2, nan: 1, 423.0: 1, 410.
0: 1, 380.0: 1, 157.0: 1, 263.0: 1, 173.0: 1, nan: 1, nan: 1, nan: 1, 156.0: 1, 264.0: 1, nan: 1, 159.0: 1, 270.0: 1, nan: 1, nan: 1, na
n: 1, 162.0: 1, nan: 1, 246.0: 1, nan: 1, nan: 1, nan: 1, nan: 1, 182.0: 1, 146.0: 1, nan: 1, 425.0: 1, 250.0: 1, nan: 1, nan: 1, nan: 1,
415.0: 1, 251.0: 1, 280.0: 1, 295.0: 1, 298.0: 1, 226.0: 1, 143.0: 1, 115.0: 1, 297.0: 1, 233.0: 1, 294.0: 1, nan: 1, nan: 1, nan: 1, na
n: 1, 323.0: 1, nan: 1, nan: 1, 90.0: 1, 308.0: 1, 224.0: 1, nan: 1, 268.0: 1, nan: 1, 256.0: 1, nan: 1, 84.0: 1, nan: 1, 288.0: 1, 273.
0: 1, 242.0: 1, 148.0: 1, nan: 1, 160.0: 1, nan: 1, 307.0: 1, 220.0: 1, 447.0: 1, 309.0: 1, 22.0: 1, 261.0: 1, 215.0: 1, 234.0: 1, 352.0:
1, nan: 1, nan: 1, 239.0: 1, nan: 1, nan: 1, 184.0: 1, nan: 1, 252.0: 1, 230.0: 1, 341.0: 1, nan: 1, 255.0: 1, nan: 1, 238.0: 1, 248.0:
1, 241.0: 1, 269.0: 1, nan: 1, nan: 1, 201.0: 1, 203.0: 1, 463.0: 1, 176.0: 1, nan: 1, nan: 1, 116.0: 1, nan: 1, nan: 1, nan: 1, 134.0:
1, 87.0: 1, nan: 1})
************************************************************************************************************************

Columns: potassium
Counter({5.0: 30, 3.5: 30, 4.9: 27, 4.7: 17, 4.8: 16, 4.0: 14, 4.2: 14, 4.1: 14, 3.8: 14, 3.9: 14, 4.4: 14, 4.5: 13, 3.7: 12, 4.3: 12, 3.
6: 8, 4.6: 7, 3.4: 5, 5.2: 5, 5.7: 4, 5.3: 4, 3.2: 3, 5.5: 3, 2.9: 3, 5.4: 3, 6.3: 3, 3.3: 3, 2.5: 2, 5.8: 2, 5.9: 2, 5.6: 2, 3.0: 2, 6.
5: 2, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, 6.4: 1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, na
n: 1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, 6.6: 1, 39.0: 1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, 7.
6: 1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, na
n: 1, 47.0: 1, nan: 1, 5.1: 1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, 2.8: 1, nan: 1, nan: 1, na
n: 1, nan: 1, 2.7: 1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, na
n: 1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1})
************************************************************************************************************************

Columns: sodium
Counter({135.0: 40, 140.0: 25, 141.0: 22, 139.0: 21, 142.0: 20, 138.0: 20, 137.0: 19, 136.0: 17, 150.0: 17, 147.0: 13, 145.0: 11, 132.0:
10, 146.0: 10, 131.0: 9, 144.0: 9, 133.0: 8, 130.0: 7, 134.0: 6, 143.0: 4, 127.0: 3, 124.0: 3, 114.0: 2, 125.0: 2, 128.0: 2, 122.0: 2, 11
3.0: 2, 120.0: 2, nan: 1, nan: 1, nan: 1, 111.0: 1, nan: 1, 104.0: 1, nan: 1, nan: 1, nan: 1, nan: 1, 4.5: 1, nan: 1, 129.0: 1, nan: 1, n
an: 1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, 163.0: 1, nan: 1, nan: 1, nan: 1, na
n: 1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, n
an: 1, nan: 1, nan: 1, nan: 1, nan: 1, 126.0: 1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, 115.0: 1, nan:
1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, nan:
1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1})
************************************************************************************************************************

Columns: blood_pressure
Counter({80.0: 116, 70.0: 112, 60.0: 71, 90.0: 53, 100.0: 25, 50.0: 5, 110.0: 3, nan: 1, nan: 1, 140.0: 1, 180.0: 1, nan: 1, nan: 1, nan:
1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, 120.0: 1, nan: 1, nan: 1, nan: 1})
************************************************************************************************************************

Columns: blood_urea
Counter({46.0: 15, 25.0: 13, 19.0: 11, 40.0: 10, 18.0: 9, 50.0: 9, 15.0: 9, 48.0: 9, 26.0: 8, 27.0: 8, 32.0: 8, 49.0: 8, 36.0: 7, 28.0:
7, 20.0: 7, 17.0: 7, 38.0: 7, 16.0: 7, 30.0: 7, 44.0: 7, 31.0: 6, 45.0: 6, 39.0: 6, 29.0: 6, 24.0: 6, 37.0: 6, 22.0: 6, 23.0: 6, 53.0: 5,
55.0: 5, 33.0: 5, 66.0: 5, 35.0: 5, 42.0: 5, 47.0: 4, 51.0: 4, 34.0: 4, 68.0: 4, 41.0: 4, 60.0: 3, 107.0: 3, 80.0: 3, 96.0: 3, 52.0: 3, 1
06.0: 3, 125.0: 3, 56.0: 2, 54.0: 2, 72.0: 2, 86.0: 2, 90.0: 2, 87.0: 2, 155.0: 2, 153.0: 2, 77.0: 2, 89.0: 2, 111.0: 2, 73.0: 2, 98.0:
2, 82.0: 2, 132.0: 2, 58.0: 2, 10.0: 2, 162.0: 1, 148.0: 1, 180.0: 1, 163.0: 1, nan: 1, 75.0: 1, 65.0: 1, 103.0: 1, 70.0: 1, 202.0: 1, 11
4.0: 1, nan: 1, nan: 1, 164.0: 1, 142.0: 1, 391.0: 1, nan: 1, nan: 1, 92.0: 1, 139.0: 1, 85.0: 1, 186.0: 1, 217.0: 1, 88.0: 1, 118.0: 1,
50.1: 1, 71.0: 1, nan: 1, 21.0: 1, 219.0: 1, 166.0: 1, 208.0: 1, 176.0: 1, nan: 1, 145.0: 1, 165.0: 1, 322.0: 1, 235.0: 1, 76.0: 1, nan:
1, nan: 1, 113.0: 1, 1.5: 1, 146.0: 1, 133.0: 1, 137.0: 1, 67.0: 1, 115.0: 1, 223.0: 1, 98.6: 1, 158.0: 1, 94.0: 1, 74.0: 1, nan: 1, 150.
0: 1, nan: 1, 61.0: 1, 57.0: 1, nan: 1, 95.0: 1, 191.0: 1, nan: 1, 93.0: 1, 241.0: 1, 64.0: 1, 79.0: 1, 215.0: 1, 309.0: 1, nan: 1, nan:
1, nan: 1, nan: 1, nan: 1, nan: 1})
************************************************************************************************************************

Columns: hemoglobin
Counter({15.0: 16, 10.9: 8, 9.8: 7, 11.1: 7, 13.0: 7, 13.6: 7, 11.3: 6, 10.3: 6, 12.0: 6, 13.9: 6, 15.4: 5, 11.2: 5, 10.8: 5, 9.7: 5, 12.
6: 5, 7.9: 5, 10.0: 5, 14.0: 5, 14.3: 5, 14.8: 5, 12.2: 4, 12.4: 4, 12.5: 4, 15.2: 4, 9.1: 4, 11.9: 4, 13.5: 4, 16.1: 4, 14.1: 4, 13.2:
4, 13.8: 4, 13.7: 4, 13.4: 4, 17.0: 4, 15.5: 4, 15.8: 4, 9.6: 3, 11.6: 3, 9.5: 3, 9.4: 3, 12.7: 3, 9.9: 3, 10.1: 3, 8.6: 3, 11.0: 3, 15.
6: 3, 8.1: 3, 8.3: 3, 10.4: 3, 11.8: 3, 11.4: 3, 11.5: 3, 15.9: 3, 14.5: 3, 16.2: 3, 14.4: 3, 14.2: 3, 16.3: 3, 16.5: 3, 15.7: 3, 16.4:
3, 14.9: 3, 15.3: 3, 17.8: 3, 12.1: 2, 9.3: 2, 10.2: 2, 10.5: 2, 6.0: 2, 11.7: 2, 8.0: 2, 12.3: 2, 8.7: 2, 13.1: 2, 8.8: 2, 13.3: 2, 14.
6: 2, 16.9: 2, 16.0: 2, 14.7: 2, 16.6: 2, 16.7: 2, 16.8: 2, 15.1: 2, 17.1: 2, 17.2: 2, 17.4: 2, 5.6: 1, 7.6: 1, 7.7: 1, nan: 1, nan: 1, 1
2.9: 1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, 6.6: 1, nan: 1, nan: 1, 7.5: 1, nan: 1, nan: 1, 4.8: 1, nan: 1, nan: 1, 7.1: 1, nan: 1, nan: 1, n
an: 1, 9.2: 1, nan: 1, 6.2: 1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, 8.2: 1, nan: 1, nan: 1, 6.1: 1, nan: 1, nan: 1, nan: 1, na
n: 1, 8.4: 1, nan: 1, 9.0: 1, nan: 1, nan: 1, 10.6: 1, nan: 1, nan: 1, nan: 1, nan: 1, 10.7: 1, nan: 1, 5.5: 1, nan: 1, 5.8: 1, 6.8: 1, 8.5: 1,
7.3: 1, nan: 1, nan: 1, 12.8: 1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, 6.3: 1, nan: 1, 3.1: 1, nan: 1, 17.3: 1, nan: 1,
nan: 1, nan: 1, nan: 1, 17.7: 1, 17.5: 1, nan: 1, 17.6: 1})
************************************************************************************************************************

Columns: specific_gravity
Counter({1.02: 106, 1.01: 84, 1.025: 81, 1.015: 75, 1.005: 7, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, na
n: 1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, na
n: 1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, nan: 1, na
n: 1, nan: 1, nan: 1, nan: 1})
************************************************************************************************************************

Columns: age
Counter({60.0: 19, 65.0: 17, 48.0: 12, 50.0: 12, 55.0: 12, 47.0: 11, 62.0: 10, 45.0: 10, 54.0: 10, 59.0: 10, 56.0: 10, 61.0: 9, 70.0: 9,
46.0: 9, 34.0: 9, 68.0: 8, 73.0: 8, 64.0: 8, 71.0: 8, 57.0: 8, 63.0: 7, 72.0: 7, 67.0: 7, 30.0: 7, 42.0: 6, 69.0: 6, 35.0: 6, 44.0: 6, 4
3.0: 6, 33.0: 6, 51.0: 5, 52.0: 5, 53.0: 5, 75.0: 5, 76.0: 5, 58.0: 5, 41.0: 5, 66.0: 5, 24.0: 4, 40.0: 4, 39.0: 4, 80.0: 4, 23.0: 4, 74.
0: 3, 38.0: 3, 17.0: 3, 8.0: 3, 32.0: 3, 37.0: 3, 25.0: 3, 29.0: 3, 21.0: 2, 15.0: 2, 5.0: 2, 12.0: 2, 49.0: 2, 19.0: 2, 36.0: 2, 20.0:
2, 28.0: 2, 7.0: 1, nan: 1, 82.0: 1, 11.0: 1, 26.0: 1, nan: 1, nan: 1, nan: 1, nan: 1, 81.0: 1, 14.0: 1, 27.0: 1, nan: 1, 83.0: 1, 4.0:
1, 3.0: 1, 6.0: 1, nan: 1, 90.0: 1, 78.0: 1, nan: 1, 2.0: 1, nan: 1, 22.0: 1, 79.0: 1})
************************************************************************************************************************
```

In [25]:

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

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

In [26]:

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

```
Columns: anemia
Counter({'no': 339, 'yes': 60, nan: 1})
************************************************************************************************************************

Columns: appetite
Counter({'good': 317, 'poor': 82, nan: 1})
************************************************************************************************************************

Columns: pus_cell_clumps
Counter({'notpresent': 354, 'present': 42, nan: 4})
************************************************************************************************************************

Columns: packed_cell_volume
Counter({nan: 70, '52': 21, '41': 21, '44': 19, '48': 19, '40': 16, '43': 14, '45': 13, '42': 13, '32': 12, '36': 12, '33': 12, '28': 12,
'50': 12, '37': 11, '34': 11, '35': 9, '29': 9, '30': 9, '46': 9, '31': 8, '39': 7, '24': 7, '26': 6, '38': 5, '47': 4, '49': 4, '53': 4,
'51': 4, '54': 4, '27': 3, '22': 3, '25': 3, '23': 2, '19': 2, '16': 1, '\t?': 1, '14': 1, '18': 1, '17': 1, '15': 1, '21': 1, '20': 1,
'\t43': 1, '9': 1})
************************************************************************************************************************

Columns: red_blood_cell_count
Counter({nan: 130, '5.2': 18, '4.5': 16, '4.9': 14, '4.7': 11, '3.9': 10, '5': 10, '4.8': 10, '4.6': 9, '3.4': 9, '3.7': 8, '6.1': 8, '5.
5': 8, '5.9': 8, '3.8': 7, '5.4': 7, '5.8': 7, '5.3': 7, '4': 6, '4.3': 6, '4.2': 6, '5.6': 6, '4.4': 5, '3.2': 5, '4.1': 5, '6.2': 5, '
5.1': 5, '6.4': 5, '5.7': 5, '6.5': 5, '3.6': 4, '6': 4, '6.3': 4, '3.5': 3, '3.3': 3, '3': 3, '2.6': 2, '2.8': 2, '2.5': 2, '3.1': 2, '
2.1': 2, '2.9': 2, '2.7': 2, '2.3': 1, '8': 1, '2.4': 1, '\t?': 1})
************************************************************************************************************************

Columns: class
Counter({'ckd': 250, 'notckd': 150})
************************************************************************************************************************

Columns: bacteria
Counter({'notpresent': 374, 'present': 22, nan: 4})
************************************************************************************************************************

Columns: hypertension
Counter({'no': 251, 'yes': 147, nan: 2})
************************************************************************************************************************

Columns: pus_cell
Counter({'normal': 259, 'abnormal': 76, nan: 65})
************************************************************************************************************************

Columns: pedal_edema
Counter({'no': 323, 'yes': 76, nan: 1})
************************************************************************************************************************

Columns: red_blood_cells
Counter({'normal': 201, nan: 152, 'abnormal': 47})
************************************************************************************************************************

Columns: coronary_artery_disease
Counter({'no': 362, 'yes': 34, '\tno': 2, nan: 2})
************************************************************************************************************************

Columns: white_blood_cell_count
Counter({nan: 105, '9800': 11, '6700': 10, '9600': 9, '9200': 9, '7200': 9, '6900': 8, '11000': 8, '5800': 8, '7800': 7, '9100': 7, '9400
': 7, '7000': 7, '4300': 6, '6300': 6, '10700': 6, '10500': 6, '7500': 5, '8300': 5, '7900': 5, '8600': 5, '5600': 5, '10200': 5, '5000':
5, '8100': 5, '9500': 5, '6000': 4, '6200': 4, '10300': 4, '7700': 4, '5500': 4, '10400': 4, '6800': 4, '6500': 4, '4700': 4, '7300': 3,
'4500': 3, '8400': 3, '6400': 3, '4200': 3, '7400': 3, '8000': 3, '5400': 3, '3800': 2, '11400': 2, '5300': 2, '8500': 2, '14600': 2, '71
00': 2, '13200': 2, '9000': 2, '8200': 2, '15200': 2, '12400': 2, '12800': 2, '8800': 2, '5700': 2, '9300': 2, '6600': 2, '12100': 1, '12
200': 1, '18900': 1, '21600': 1, '11300': 1, '\t6200': 1, '11800': 1, '12500': 1, '11900': 1, '12700': 1, '13600': 1, '14900': 1, '16300
': 1, '\t8400': 1, '10900': 1, '2200': 1, '11200': 1, '19100': 1, '\t?': 1, '12300': 1, '16700': 1, '2600': 1, '26400': 1, '4900': 1, '12
000': 1, '15700': 1, '4100': 1, '11500': 1, '10800': 1, '9900': 1, '5200': 1, '5900': 1, '9700': 1, '5100': 1})
************************************************************************************************************************

Columns: diabetesmellitus
Counter({'no': 258, 'yes': 134, '\tno': 3, '\tyes': 2, nan: 2, ' yes': 1})
************************************************************************************************************************
```

```
In [30]: contcols.remove('specific_gravity')
         contcols.remove('albumin')
         contcols.remove('sugar')
         print(contcols)

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

In [31]: contcols.add('red_blood_cell_count')
         contcols.add('packed_cell_volume')
         contcols.add('white_blood_cell_count')
         print(contcols)

{'serum_creatinine', 'blood glucose random', 'red_blood_cell_count', 'potassium', 'packed_cell_volume', 'white_blood_cell_count', 'sodium
', 'blood_pressure', 'blood_urea', 'hemoglobin', 'age'}

In [32]: catcols.add('specific_gravity')
         catcols.add('albumin')
         catcols.add('sugar')
         print(catcols)

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

In [33]: data['coronary_artery_disease']=data.coronary_artery_disease.replace('\tno','no')
         c(data['coronary_artery_disease'])

Out[33]: Counter({'no': 364, 'yes': 34, nan: 2})

In [34]: data['diabetesmellitus']=data.diabetesmellitus.replace(to_replace={'\tno':'no','\tyes':'yes',' yes':'yes',' no':'no'})
         c(data['diabetesmellitus'])

Out[34]: Counter({'yes': 137, 'no': 261, nan: 2})
```

```
In [35]: data.isnull().any()

Out[35]: 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

In [36]: data.isnull().sum()

Out[36]: age                        9
         blood_pressure            12
         specific_gravity          47
         albumin                   46
         sugar                     49
         red_blood_cells          152
         pus_cell                  65
         pus_cell_clumps            4
         bacteria                   4
         blood glucose random      44
         blood_urea                19
         serum_creatinine          17
         sodium                    87
         potassium                 88
         hemoglobin                52
         packed_cell_volume        70
         white_blood_cell_count   105
         red_blood_cell_count     130
         hypertension               2
         diabetesmellitus           2
         coronary_artery_disease    2
         appetite                   1
         pedal_edema                1
         anemia                     1
         class                      0
         dtype: int64
```

In [37]:
```
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')
```

In [38]:
```
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)
```

In [39]:
```
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)
```

In [40]:
```
from sklearn.preprocessing import LabelEncoder
```

In [41]:
```
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)
```

```
LABEL ENCODING OF: albumin
Counter({0.0: 245, 1.0: 44, 2.0: 43, 3.0: 43, 4.0: 24, 5.0: 1})
Counter({0: 245, 1: 44, 2: 43, 3: 43, 4: 24, 5: 1})
****************************************************************************************************
LABEL ENCODING OF: sugar
Counter({0.0: 339, 2.0: 18, 3.0: 14, 4.0: 13, 1.0: 13, 5.0: 3})
Counter({0: 339, 2: 18, 3: 14, 4: 13, 1: 13, 5: 3})
****************************************************************************************************
LABEL ENCODING OF: anemia
Counter({'no': 340, 'yes': 60})
Counter({0: 340, 1: 60})
****************************************************************************************************
LABEL ENCODING OF: appetite
Counter({'good': 318, 'poor': 82})
Counter({0: 318, 1: 82})
****************************************************************************************************
LABEL ENCODING OF: pus_cell_clumps
Counter({'notpresent': 358, 'present': 42})
Counter({0: 358, 1: 42})
****************************************************************************************************
LABEL ENCODING OF: class
Counter({'ckd': 250, 'notckd': 150})
Counter({0: 250, 1: 150})
****************************************************************************************************
LABEL ENCODING OF: bacteria
Counter({'notpresent': 378, 'present': 22})
Counter({0: 378, 1: 22})
****************************************************************************************************
LABEL ENCODING OF: hypertension
Counter({'no': 253, 'yes': 147})
Counter({0: 253, 1: 147})
****************************************************************************************************
LABEL ENCODING OF: pus_cell
Counter({'normal': 324, 'abnormal': 76})
Counter({1: 324, 0: 76})
****************************************************************************************************
LABEL ENCODING OF: pedal_edema
Counter({'no': 324, 'yes': 76})
Counter({0: 324, 1: 76})
****************************************************************************************************
LABEL ENCODING OF: red_blood_cells
Counter({'normal': 353, 'abnormal': 47})
Counter({1: 353, 0: 47})
****************************************************************************************************
LABEL ENCODING OF: coronary_artery_disease
Counter({'no': 366, 'yes': 34})
Counter({0: 366, 1: 34})
****************************************************************************************************
LABEL ENCODING OF: specific_gravity
Counter({1.02: 153, 1.01: 84, 1.025: 81, 1.015: 75, 1.005: 7})
Counter({3: 153, 1: 84, 4: 81, 2: 75, 0: 7})
****************************************************************************************************
LABEL ENCODING OF: diabetesmellitus
Counter({'no': 263, 'yes': 137})
Counter({0: 263, 1: 137})
****************************************************************************************************
```

In [42]:
```
selcols=['red_blood_cells','pus_cell','blood glucose random','blood_urea','pedal_edema','anemia','diabetesmellitus','coronary_artery_dis
```

In [43]:
```
x=pd.DataFrame(data,columns=selcols)
y=pd.DataFrame(data,columns=['class'])
print(x.shape)
print(y.shape)
```

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

In [44]:
```
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)
```

In [45]:
```
from sklearn.ensemble import RandomForestClassifier
lgr = RandomForestClassifier()
```

In [46]:
```
#from sklearn.linear_model import LogisticRegression
```

In [47]:
```
#lgr=LogisticRegression(solver='lbfgs', max_iter=1000)
lgr.fit(x_train.values, y_train.values.ravel())
```

Out[47]:
```
RandomForestClassifier()
```

In [48]:
```
y_pred=lgr.predict(x_test.values)
```

In [49]:
```
y_pred1=lgr.predict([[90,157,0,1,1,0,1,1]])
print(y_pred1)
c(y_pred1)
```

```
[0]
```

Out[49]:
```
Counter({0: 1})
```

In [50]:
```
accuracy_score(y_test,y_pred)
```

Out[50]:
```
0.95
```

In [51]:
```
conf_mat=confusion_matrix(y_test,y_pred)
conf_mat
```

Out[51]:
```
array([[52,  2],
       [ 2, 24]], dtype=int64)
```

In [52]:
```
pickle.dump(lgr,open('CKD.pkl','wb'))
```

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

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


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


@app.route('/Prediction', methods=['POST', 'GET'])
def prediction():
    return render_template('indexnew.html')


@app.route('/Home', methods=['POST', 'GET'])
def my_home():
    return render_template('home.html')


@app.route('/predict', methods=['POST'])
def predict():
    #input_features = ([int(x) for x in request.form.values()])
    blood_urea = request.form["blood_urea"]
    blood_glucose_random = request.form["blood_glucose_random"]
    anemia = request.form["Anemia"]
    if (anemia == "no"):
        anemia = 0
    if (anemia == "yes"):
```

Building a flask file :

```python
        red_blood_cell = 1

    diabetics_mellitus = request.form["diabetics_mellitus"]
    if (diabetics_mellitus == "no"):
        diabetics_mellitus = 0
    if (diabetics_mellitus == "yes"):
        diabetics_mellitus = 1

    pedal_edema = request.form["pedal_edema"]
    if (pedal_edema == "no"):
        pedal_edema = 0
    if (pedal_edema == "yes"):
        pedal_edema = 1

    input_features = [int(blood_urea),int(blood_glucose_random),int(anemia),int(coronary_artery_disease),int(pus_cell),int(red_blood_cell
    #input_features = [int(red_blood_cell),int(pus_cell),int(blood_glucose_random),int(blood_urea),int(pedal_edema),int(anemia),int(diabe
    print(input_features)
    features_value = [np.array(input_features)]


    #features_name = ['red_blood_cells','pus_cell','blood glucose random','blood_urea','pedal_edema','anemia','diabetesmellitus','coronar
    features_name = ['blood_urea','blood glucose random','anemia','coronary_artery_disease','pus_cell','red_blood_cells','diabetesmellitu
    df = pd.DataFrame(features_value, columns=features_name)
    output = model.predict(df)
    return render_template('result.html', prediction_text=output)


# Press the green button in the gutter to run the script.
if __name__ == '__main__':
    app.run(host='localhost', debug=True)

# See PyCharm help at https://www.jetbrains.com/help/pycharm/
```

Home.html screen shots:

```html
<html lang="en">
<head>

    <title>Chronic Kidney Disease prediction</title>

        <!-- <link rel="stylesheet" href="css/tooplate-style.css">-->

</head>
<body>
    <form action="{{url_for('prediction')}}" method="POST">
    <!--<img src='C:\Users\kgaru\Desktop\ibm_chronic_kidney_disease\app_ibm\templates\images\slider5.jpeg' width="400" height="200" alt=

   <section id="home">

        <div class="container">
            <div class="row">
                    <div class="owl-carousel owl-theme">
                        <div class="item item-first">
                            <div class="caption">
                                <div style="background-color: pink;text-align: center;">
                            <h1>Chronic Kidney Disease</h1><h3>Machine Learning web app built with Flask</h3>
                        </div>
                        <div class="col-md-12 col-sm-12">
                                <button type="submit" class="form-control" id="cf-submit" name="submit">prediction</button>
                                    <!--<img id="test" src="images\slider5.jpeg" width="1400" height="650" alt="No image" />-->

                                </div>
                        </div>

                            </div>
                        </div>
                    </div>
            </div>
```

```html
                                <option value = "no">no </option>
                            </select>
                        </div>

                <div class="col-md-6 col-sm-6">
                            <label for="select">Select Diabetics Mellitus</label>
                            <select name = "diabetics_mellitus">
                                <option value = "yes"> yes</option>
                                <option value = "no">no </option>
                            </select>
                        </div>

                        <div class="col-md-6 col-sm-6">
                            <label for="select">Select Pedal Edema</label>
                            <select name = "pedal_edema">
                                <option value = "yes"> yes</option>
                                <option value = "no">no </option>
                            </select>
                        </div>
                        <div class="col-md-12 col-sm-12">
                            <button type="submit" class="form-control" id="cf-submit" name="submit">predict</button>

                        </div>
                    </div>
                </form>

            </div>
        </section>

    </body>
  </html>
```

Result to display the screen shot

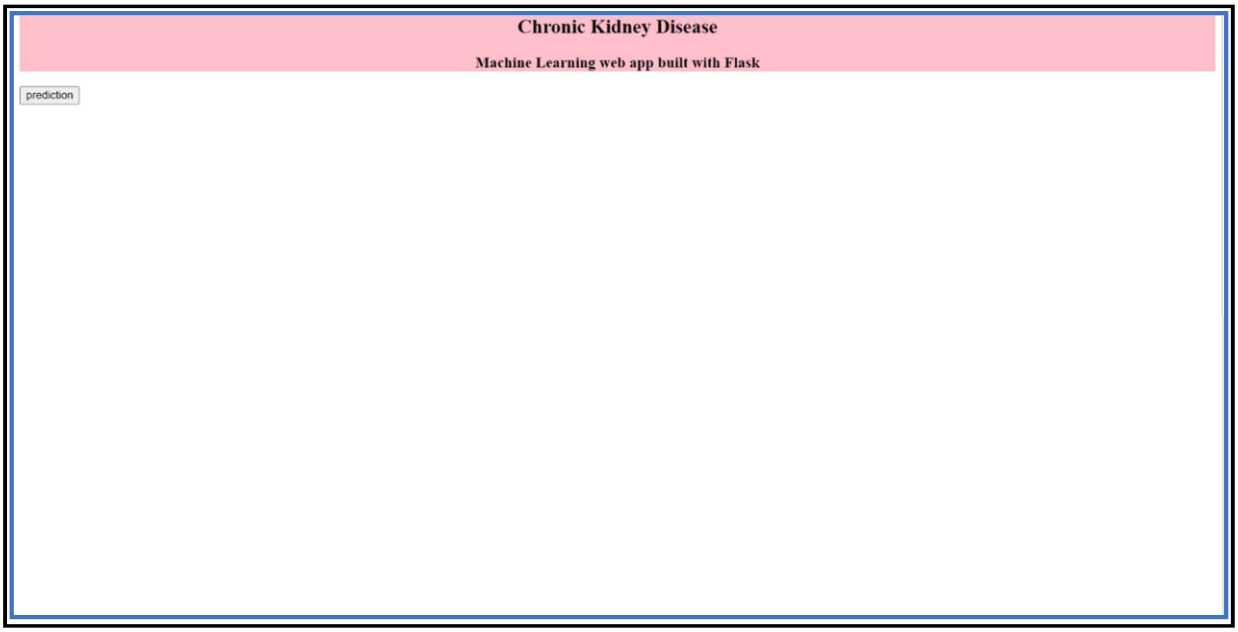
```
CKD FINAL
> .idea
> .ipynb_checkpoints
v templates
  > images
    home.html
    indexnew.html
    result.html
> venv
  app.py
  CKD Prediction Notebook.ipynb
  CKD.pkl
  kidney_disease.csv
```

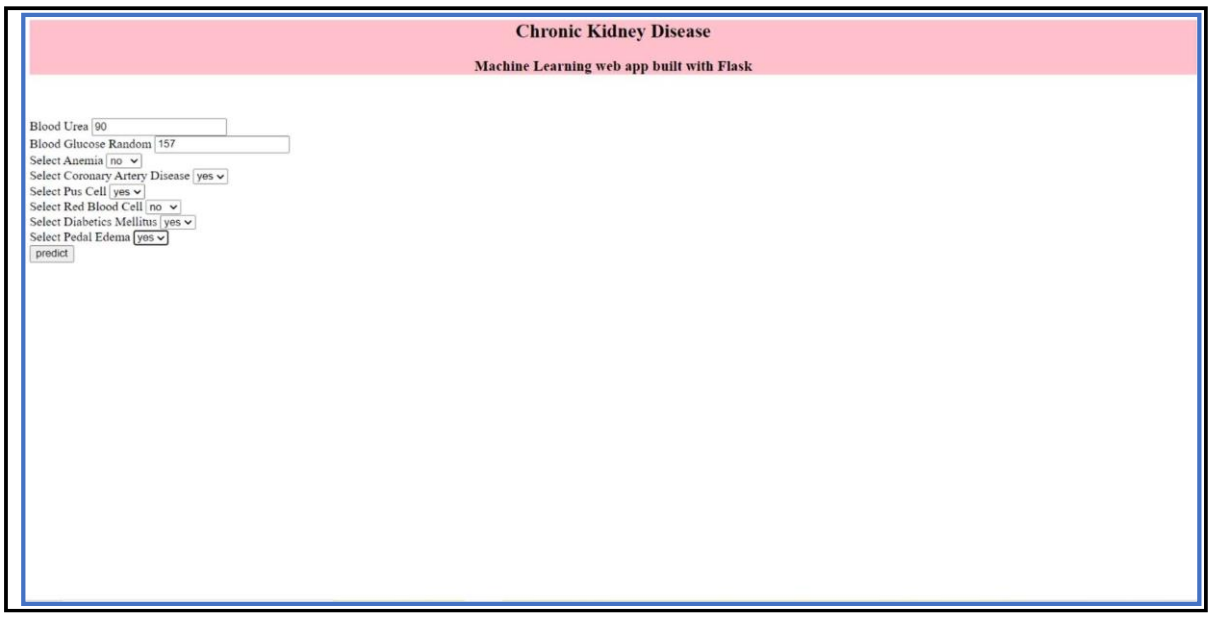
```
templates > result.html > ...
    </head>
<body id="top">
            <section>
        <div class="container">
            <div class="row">

                <div style="background-color: pink;text-align: center;">
                <h1>Chronic Kidney Disease</h1><h3>Machine Learning web app built with Flask</h3>
            </div>
                <br/>
                <br/>

            <div class="col-md-8 col-sm-7">
                {{prediction_text}}
                {% if prediction_text==1%}

                <div >
                    <h2>Medical Result</h2>
                    <p>Your Condition is normal.</p>
                    <blockquote>You are not having chronic kidney disease.</blockquote>
                      <!--<img src="images\result1.jpg" width="300" height="300"/>-->
                    <p></p>

                </div>
                {%else%}
                <div >
                    <h2>Medical Result</h2>
                    <p>Your Condition is abnormal.</p>
                    <blockquote>You are having chronic kidney disease.</blockquote>
                    <!--<img src="images\result2.jpeg" width="300" height="300"/>-->
                    <p></p>
```

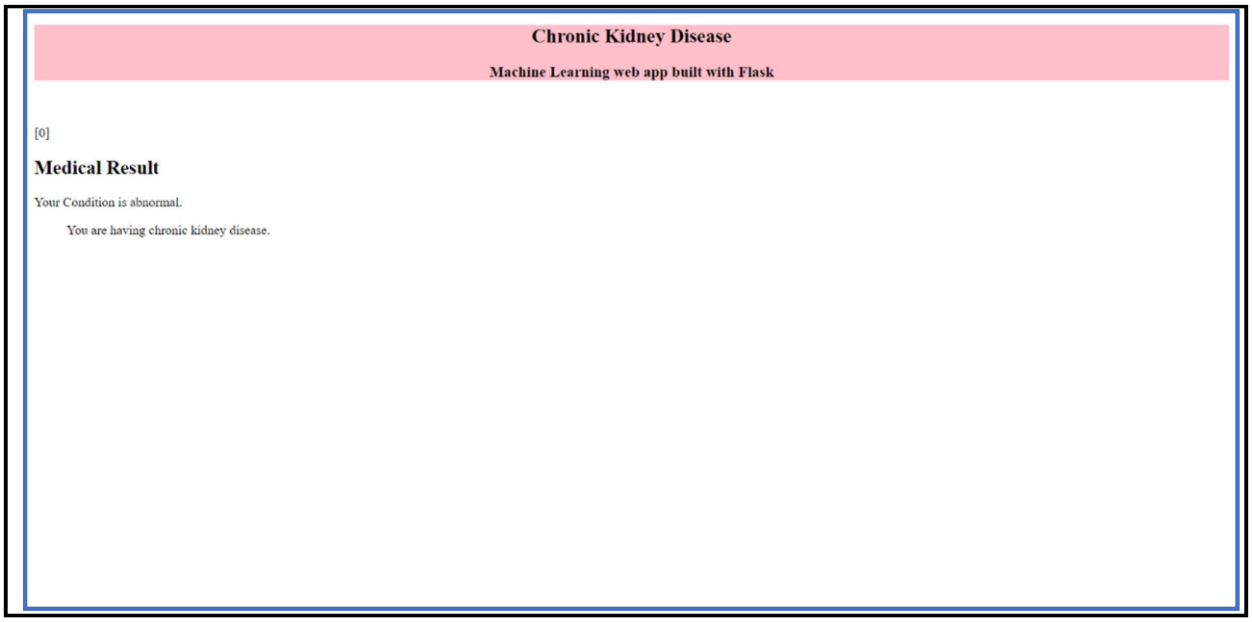
Running the app

```
(c) Microsoft Corporation. All rights reserved.

C:\Users\kgaru\Desktop\ckd final>c:\users\kgaru\anaconda3\scripts\activate

(base) C:\Users\kgaru\Desktop\ckd final>conda activate deployment

(deployment) C:\Users\kgaru\Desktop\ckd final>python app.py
c:\Users\kgaru\anaconda3\envs\deployment\lib\site-packages\sklearn\base.py:329: UserWarning: Trying to unpickle estimator DecisionTreeClassifier fro
m version 1.0.2 when using version 1.1.2. This might lead to breaking code or invalid results. Use at your own risk. For more info please refer to:
https://scikit-learn.org/stable/model_persistence.html#security-maintainability-limitations
  warnings.warn(
c:\Users\kgaru\anaconda3\envs\deployment\lib\site-packages\sklearn\base.py:329: UserWarning: Trying to unpickle estimator RandomForestClassifier fro
m version 1.0.2 when using version 1.1.2. This might lead to breaking code or invalid results. Use at your own risk. For more info please refer to:
https://scikit-learn.org/stable/model_persistence.html#security-maintainability-limitations
  warnings.warn(
```

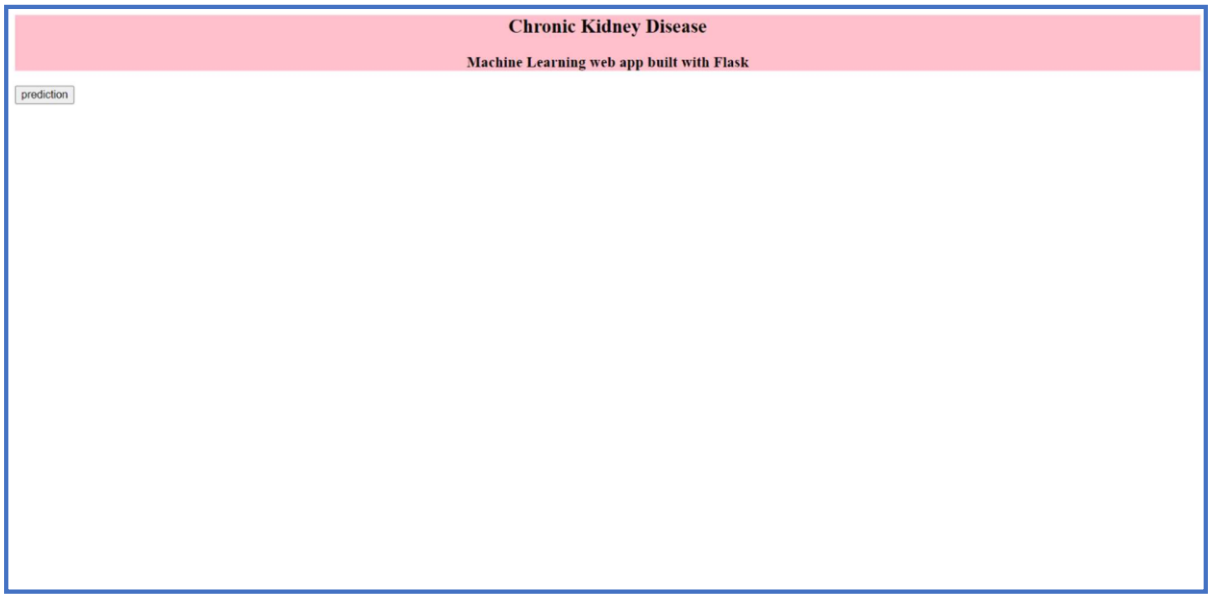
TEST CASE 1: CKD

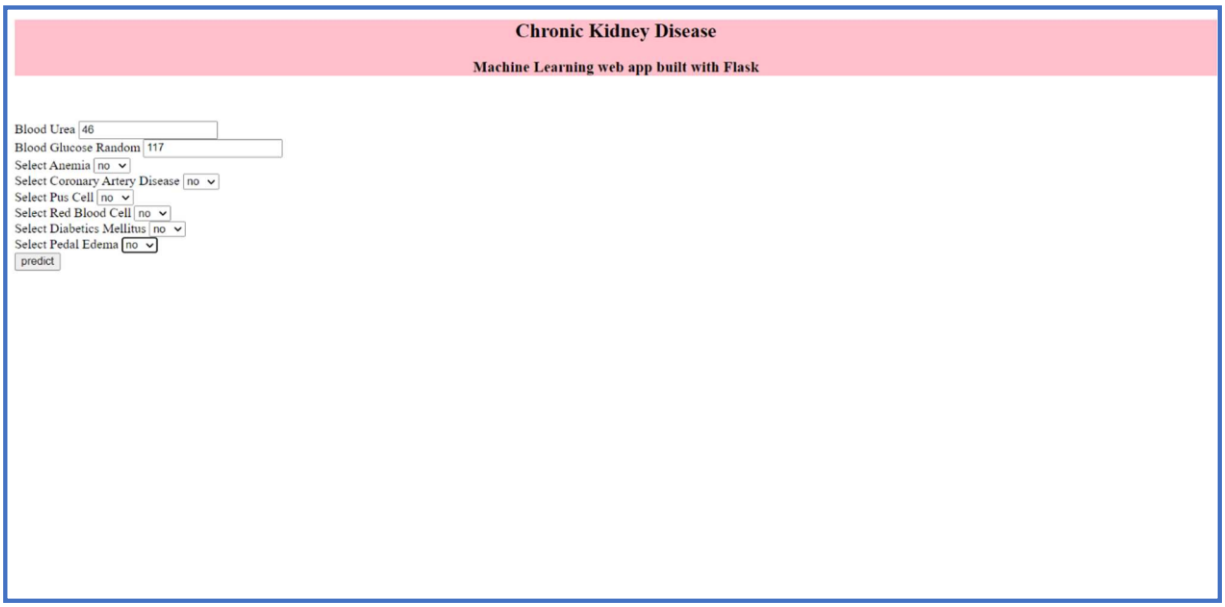


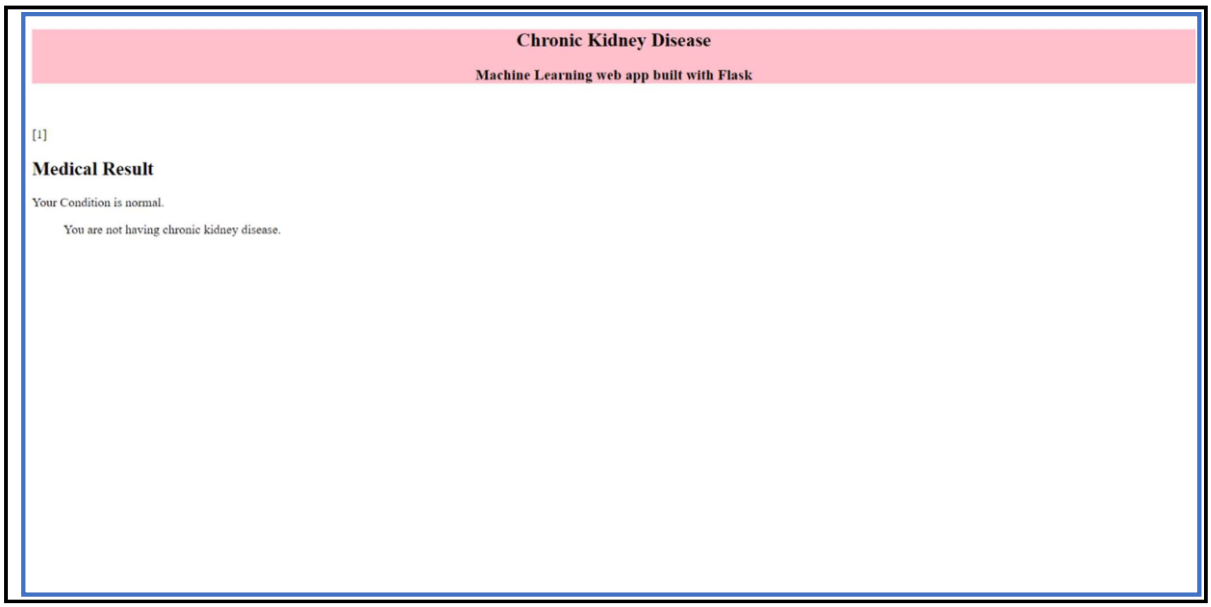




TEST CASE 2: NO CKD




Chronic Kidney Disease
Machine Learning web app built with Flask
Blood Urea 46
Blood Glucose Random 117
Select Anemia no
Select Coronary Artery Disease no
Select Pus Cell no
Select Red Blood Cell no
Select Diabetics Mellitus no
Select Pedal Edema no
predict


Chronic Kidney Disease
Machine Learning web app built with Flask
[1]
Medical Result
Your Condition is normal.
You are not having chronic kidney disease.

7.0 Conclusion:

This Project has helped team members to understand various concepts of Machine learning, Flask file, IBM cloud and Python notebook.

This project can be scaled for usage in prediction of other chronic diseases which will help doctors in diagnosis of disease at an early stage thereby helping in early detection of various disease.

Github link:

https://github.com/IBM-EPBL/IBM-Project-53532-1661414687