A PROJECT REPORT

Early Prediction for Chronic Kidney Disease Detection: A Progressive Approach to Health Management

Team ID: NM2023TMID17387

Team Leader : VINISHA I

Team member: THANUSHA S

Team member: UTHAYAN S

Team member : VENKATESH T

INDEX

Early Prediction for Chronic Kidney Disease Detection: A Progressive Approach to Health Management

1. Introduction

1.1OVERVIEW

Early prediction for chronic kidney disease (CKD) detection is an important aspect of health management, as CKD is a prevalent and progressive condition that can lead to serious complications if not identified and managed in its early stages. A progressive approach to health management involves utilizing various techniques and strategies to detect CKD early, monitor its progression, and implement appropriate interventions to slow down its advancement.

The early prediction for CKD detection typically involves a combination of clinical data, laboratory tests, and machine learning algorithms to identify individuals at risk of developing CKD. These algorithms analyze a range of factors, such as patient demographics, medical history, lifestyle factors, and biomarkers, to generate predictive models that can estimate the probability of an individual developing CKD in the future. These predictive models can help healthcare providers identify high-risk individuals and initiate targeted interventions to prevent or delay the onset of CKD.

A progressive approach to health management for CKD also involves monitoring the progression of the disease over time. This can be done through regular follow-up visits, laboratory tests, and imaging studies to assess kidney function, identify any changes in disease severity, and adjust treatment plans accordingly.

1.2 Purpose

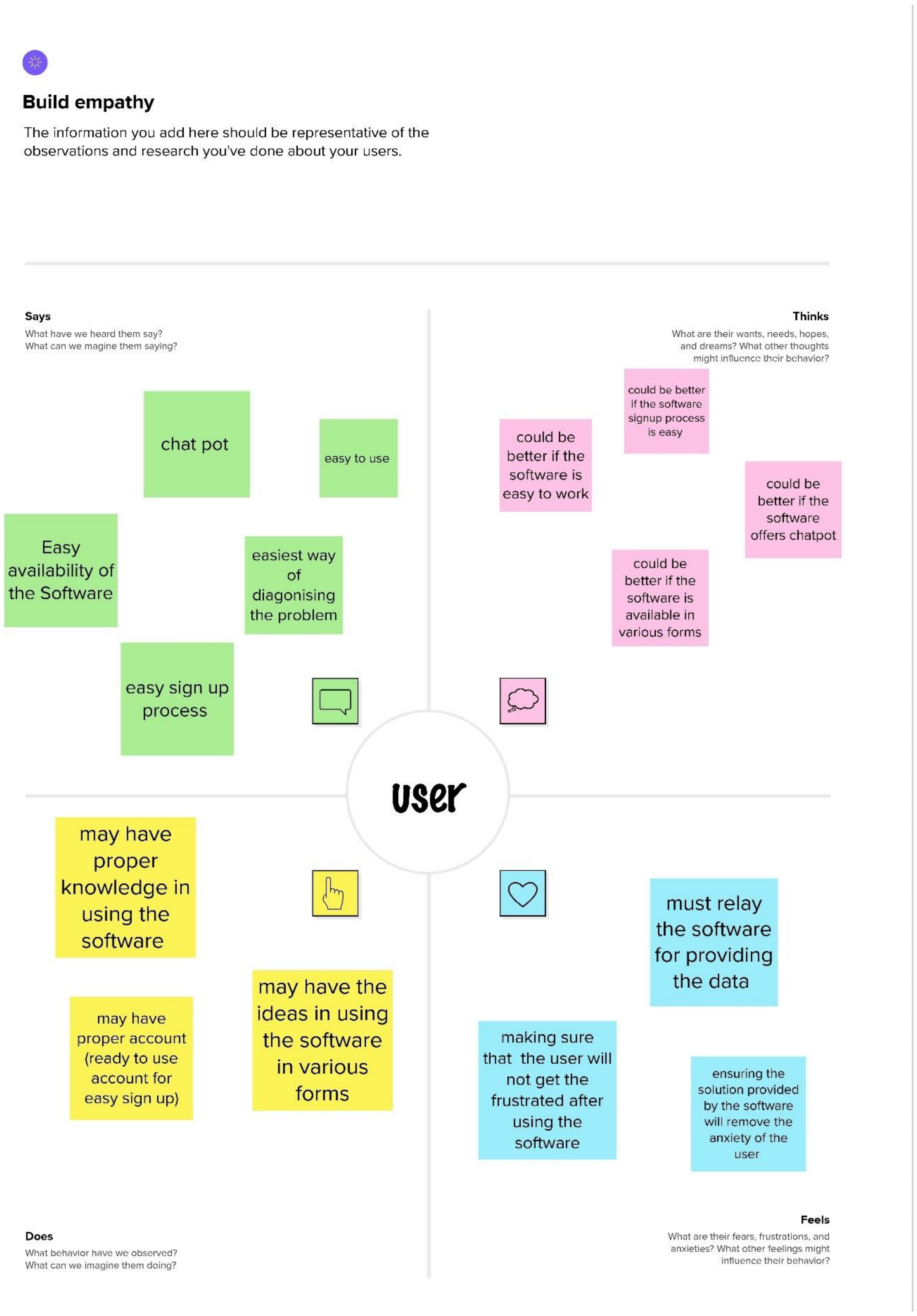
The purpose of early prediction for chronic kidney disease (CKD) detection using a progressive approach to health management is to identify individuals at risk of developing CKD in its early stages, monitor disease progression, and implement appropriate interventions to slow down the advancement of CKD. This approach aims to improve patient outcomes, prevent complications, and optimize healthcare management of CKD.

Early prediction of CKD is crucial because the condition often progresses silently, with few noticeable symptoms until it reaches advanced stages. By identifying individuals at risk of CKD early, healthcare providers can initiate timely interventions to prevent or delay the onset of CKD. This can include lifestyle modifications, medication management, and other targeted interventions to reduce risk factors and promote kidney health.

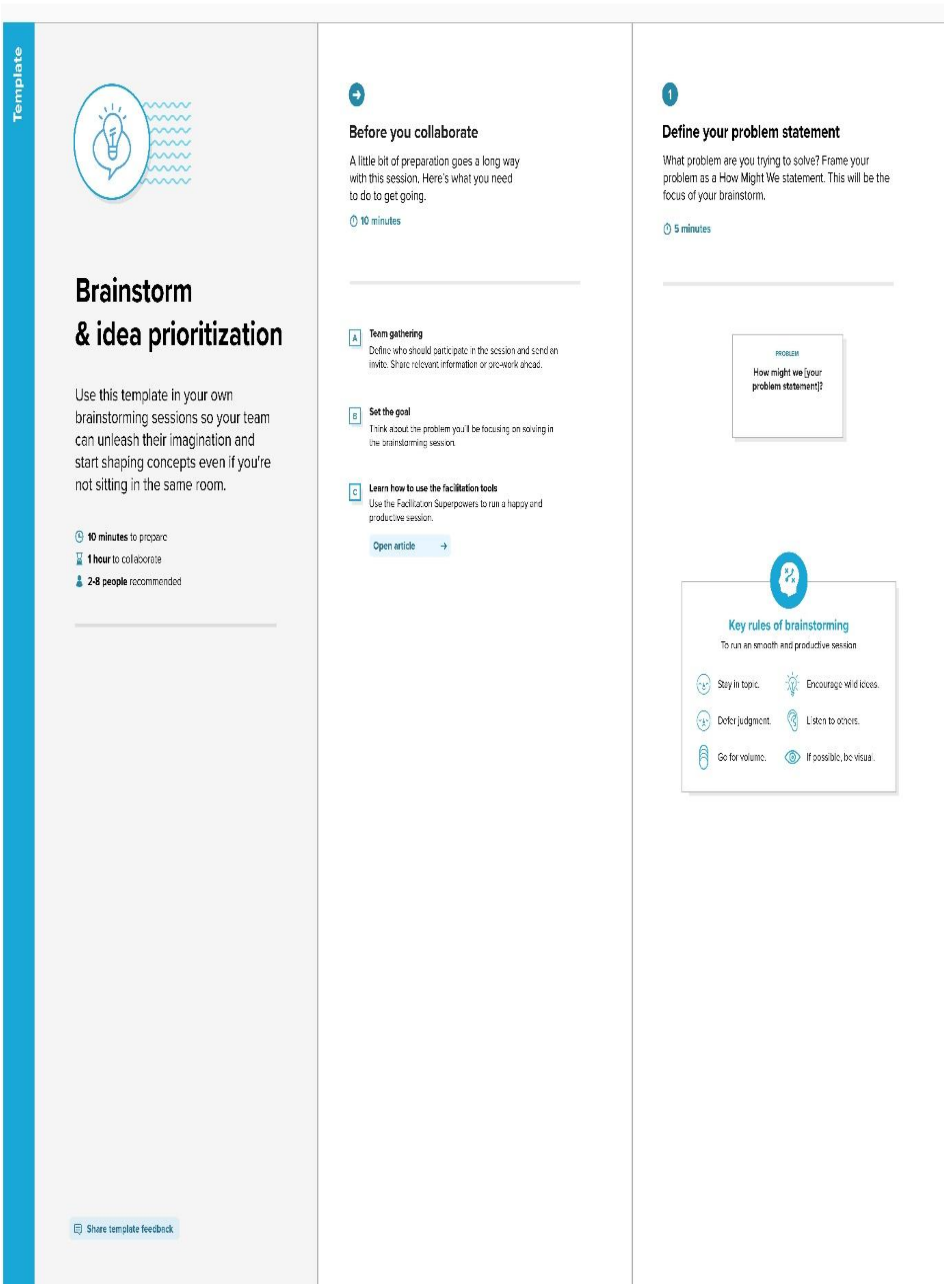
Monitoring CKD progression is essential to assess the severity of the disease, identify any changes in kidney function, and adjust treatment plans accordingly. Regular monitoring allows healthcare providers to intervene early if disease progression is detected, enabling timely adjustments to treatment strategies and preventing complications associated with advanced CKD.

Problem Definition & Design Thinking

2.1 Empathy Map


Build empathy
The information you add here should be representative of the observations and research you've done about your users.
Says
What have we heard them say?
What can we magine them saying?
chat pot
easy to use
Easy availability of the Software
easiest way of diagonising the problem
easy sign up process
Thinks
What are their wants, needs, hopes, and dreams? What other thoughts might influence their behavior?
could be better if the software signup process is easy
could be better if the software is easy to work
could be better if the software offers chatpot
could be better if the software is available in various forms
user
may have proper knowledge in using the software
may have proper account (ready to use account for easy sign up)
may have the ideas in using the software in various forms
must relay the software for providing the data
making sure that the user will not get the frustrated after using the software
ensuring the solution provided by the software will remove the anxiety of the user
Does
What behavior have we observed?
What can we imagine them doing?
Feels
What are their fears, frustrations, and anxieties? What other feelings might influence their behavior?

2.2 Ideation & Brainstorming Map



Template

Brainstorm & idea prioritization

Use this template in your own brainstorming sessions so your team can unleash their imagination and start shaping concepts even if you're not sitting in the same room.

10 minutes to prepare

1 hour to collaborate

2-8 people recommended

Share template feedback

Before you collaborate

A little bit of preparation goes a long way with this session. Here's what you need to do to get going.

10 minutes

A **Team gathering**
Define who should participate in the session and send an invite. Share relevant information or pre-work ahead.

B **Set the goal**
Think about the problem you'll be focusing on solving in the brainstorming session.

C **Learn how to use the facilitation tools**
Use the Facilitation Superpowers to run a happy and productive session.

Open article →

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

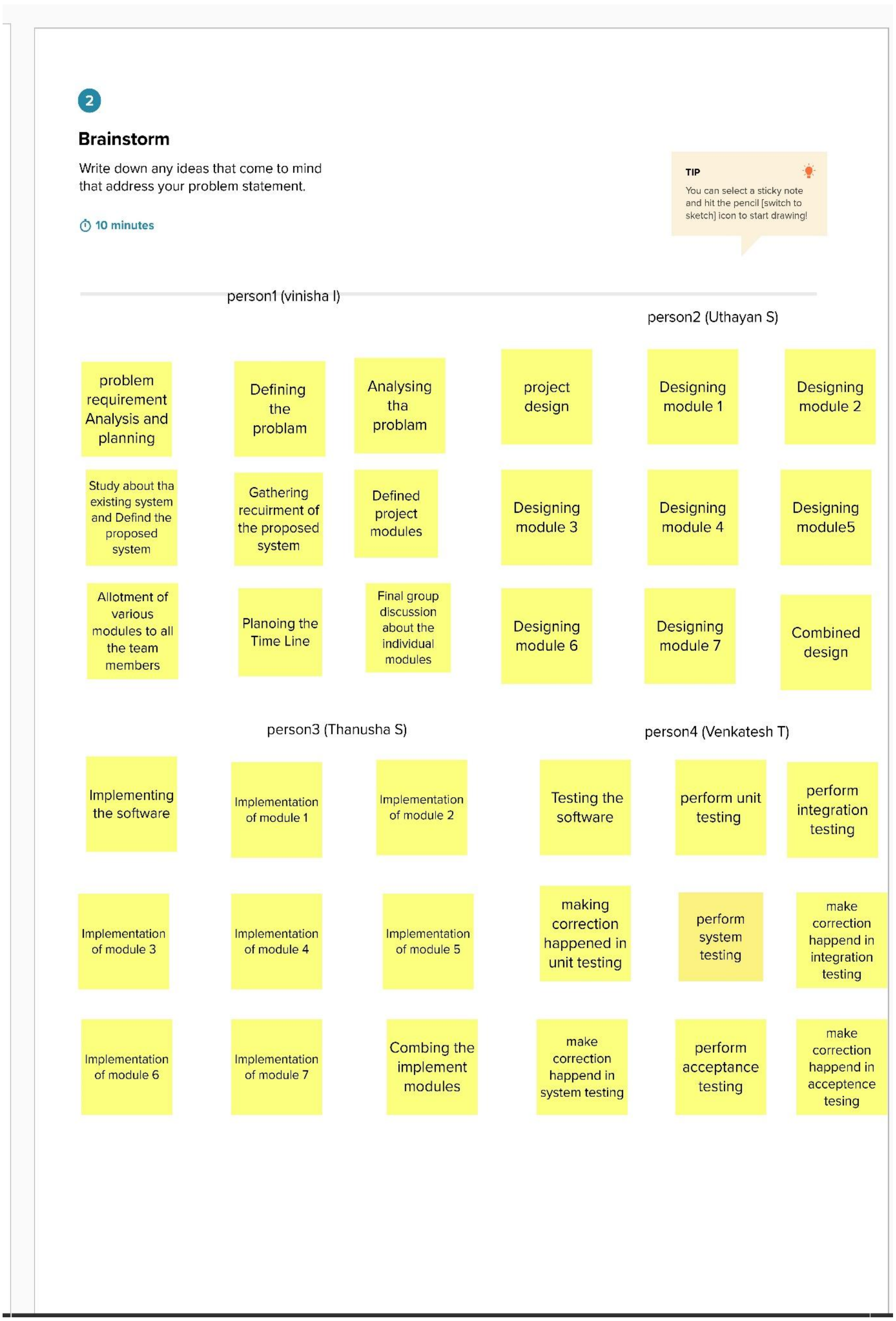
PROBLEM

How might we [your problem statement]?

Key rules of brainstorming

To run an smooth and productive session

- Stay in topic.
- Encourage wild ideas.
- Defer judgment.
- Listen to others.
- Go for volume.
- If possible, be visual.



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!

person1 (vinisha I)

- problem requirement Analysis and planning
- Defining the problam
- Analysing tha problam
- Study about tha existing system and Defind the proposed system
- Gathering recuirment of the proposed system
- Defined project modules
- Allotment of various modules to all the team members
- Planoing the Time Line
- Final group discussion about the individual modules

person2 (Uthayan S)

- project design
- Designing module 1
- Designing module 2
- Designing module 3
- Designing module 4
- Designing module5
- Designing module 6
- Designing module 7
- Combined design

person3 (Thanusha S)

- Implementing the software
- Implementation of module 1
- Implementation of module 2
- Implementation of module 3
- Implementation of module 4
- Implementation of module 5
- Implementation of module 6
- Implementation of module 7
- Combing the implement modules

person4 (Venkatesh T)

- Testing the software
- perform unit testing
- perform integration testing
- making correction happened in unit testing
- perform system testing
- make correction happend in integration testing
- make correction happend in system testing
- perform acceptance testing
- make correction happend in acceptence tesing

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

idea

one potencial application of a progressive approch to health management for chronic kidney dieases(CKD) is to devolop a predictive model that can identifuy patients at high risk of devoloping the disease.

by analyzing patient data such as age, gender, blood presure, blood sugar level a machine learning model could predict the likelihood of a patient devoloping CKD in the future

Another application is early detection and diagnosis of CKD by analyzingpatient data such as blood test results, urine test result and medical history.

TIP

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

A progrssive approach to health management CKD could also involve devoloping personlized treatment plants for patients based onn their individual health data.

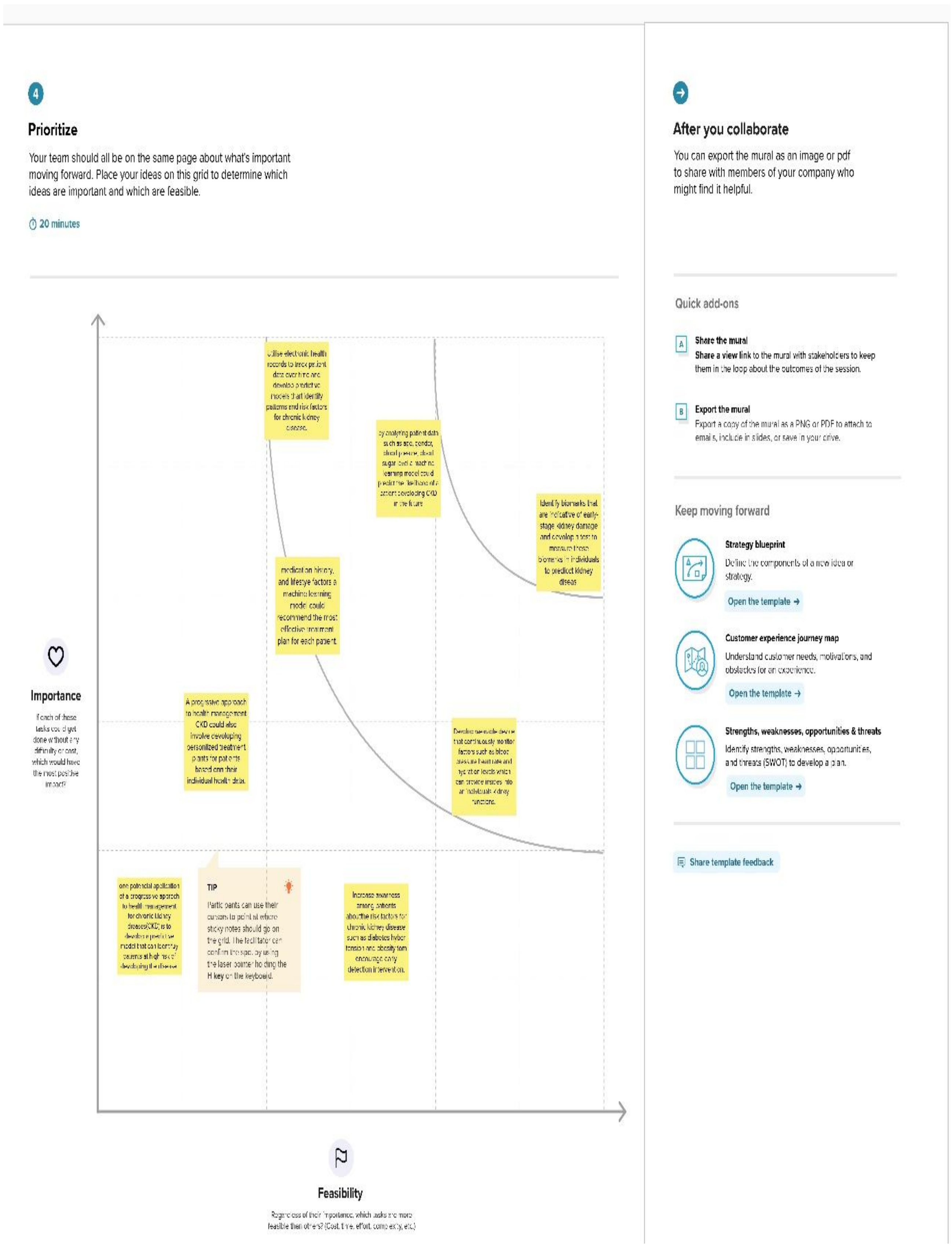
By analyzing patient on their individual health data. by analyzing patient dfata such as kidney function.

medication history, and lifestye factors a machine learning model could recommend the most effective treatment plan for each patient.

Identify biomarks that are indicative of early-stage kidney damage and devolop a test to measure these biomarks in individuals to predicct kidney diseas

Implement a telemedicine program that allows patients to receive regular check-ups and assessments of their kidney function which can help detect early sings of kidney damage.

Devolop a mobile application that allows individuals to track their daily fluid intake exercise, and diet to provide early warning signs of kidney damage.



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

Utilise electronic health records to track patient data over time and develop predictive models thart identify patterns and risk factors for chronic kidney disease.

By analyzing patient data such as age, gender, blood pressure, blood sugar level a machine learning model could predict the likelihood of a patient developing CKD in the future

Identify biomarks that are indicative of early-stage kidney damage and develop a test to measure these biomarks in individuals to predict kidney diseas

medication history, and lifestye factors a machine learning model could recommend the most effective treatment plan for each patient.

A progressive approach to health management CKD could also involve developing personlized treatment plants for patients based onn their individual health data.

Develop wearable device that continuously monitor factors such as blood pressure heart rate and hydration levels which can provide insights into an individuals kidney functions.

one potencial apllication of a progressive aproach to health management for chronic kidney disease(CKD) is to develop a predictive model that can identify patients at high risk of developing the disease

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.

Increase awarness among patients aboutthe risk factors for chronic kidney disease such as diabetes hyber tension and obesity tom encourage early detection intervention.

Importance

If each of these tasks could get done without any difficulty or cost, which would have the most positive impact?

Feasibility

Regardless of their importance, which tasks are more feasible then others? (Cost, time, effort, complexity, etc.)

After you collaborate

You can export the mural as an image or pdf to share with members of your company who might find it helpful.

Quick add-ons

A **Share the mural**

Share a view link to the mural with stakeholders to keep them in the loop about the outcomes of the session.

B **Export the mural**

Export a copy of the mural as a PNG or PDF to attach to emails, include in slides, or save in your drive.

Keep moving forward

Strategy blueprint

Define the components of a new idea or strategy.

Open the template →

Customer experience journey map

Understand customer needs, motivations, and obstacles for an experience.

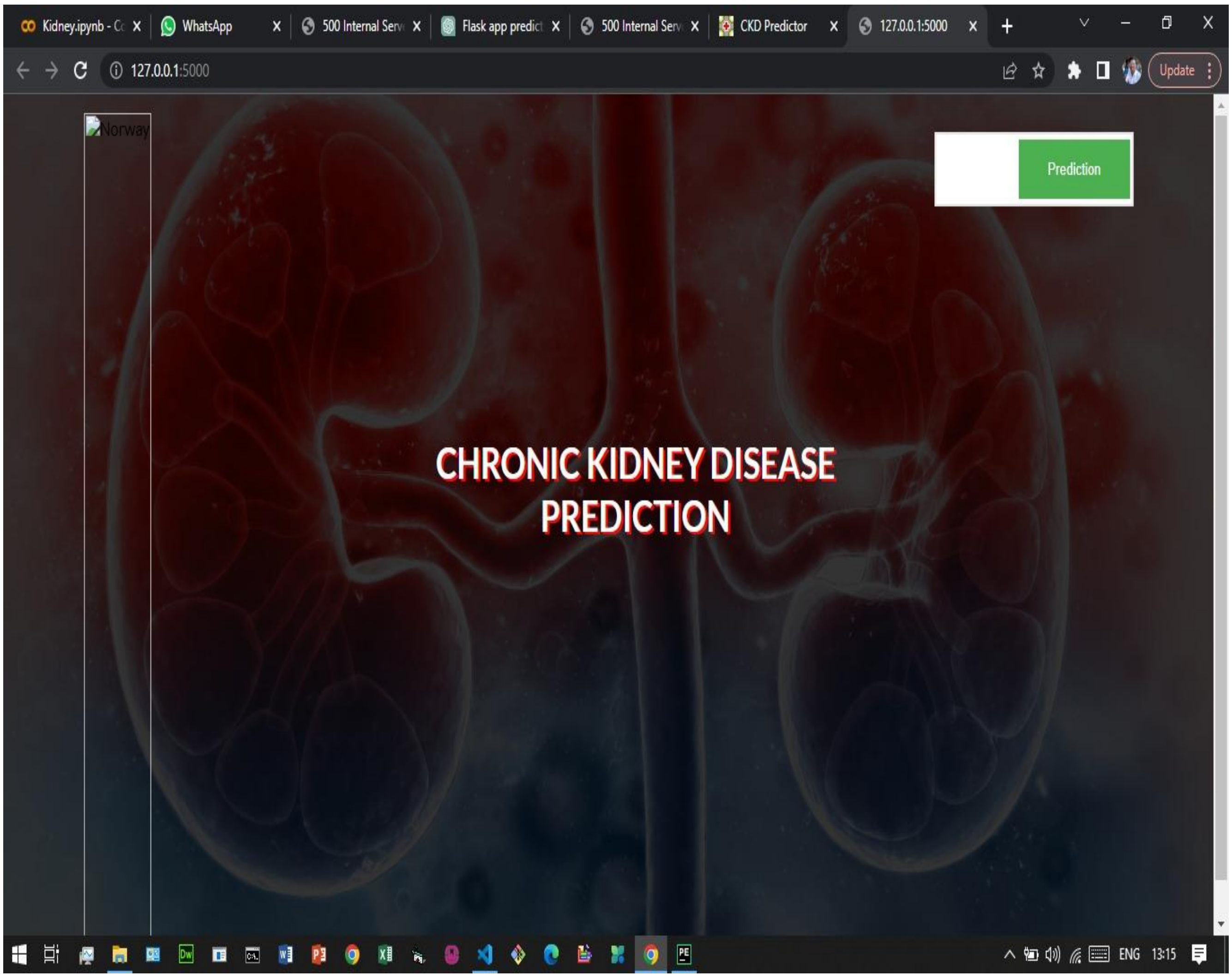
Open the template →

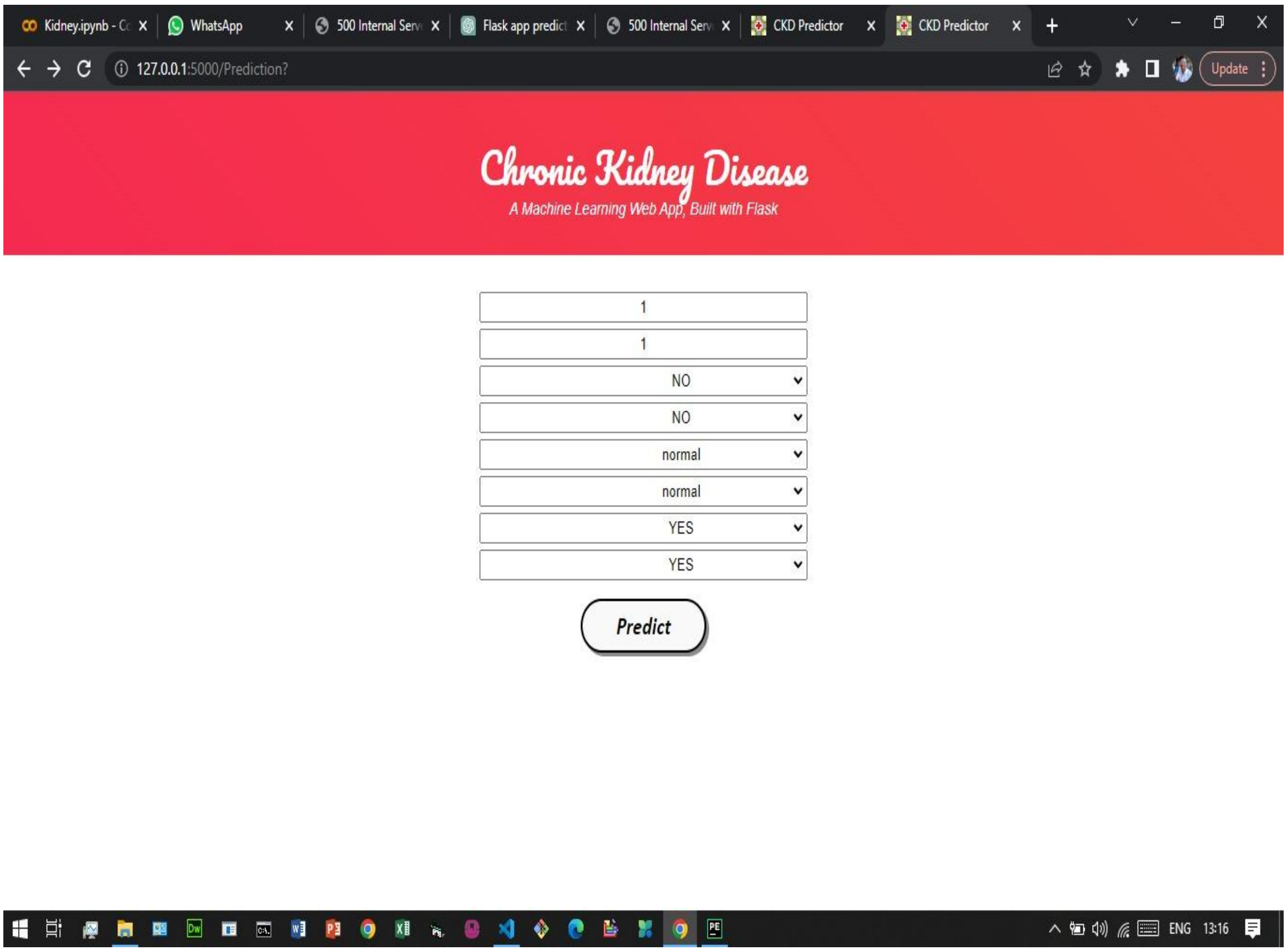
Strengths, weaknesses, opportunities & threats

Identify strengths, weaknesses, opportunities, and threats (SWOT) to develop a plan.

Open the template →

Share template feedback


Kidney.ipynb - Co
WhatsApp
500 Internal Serv
Flask app predict
500 Internal Serv
CKD Predictor
127.0.0.1:5000
127.0.0.1:5000
Update
Norway
Prediction
CHRONIC KIDNEY DISEASE PREDICTION
ENG
13:15

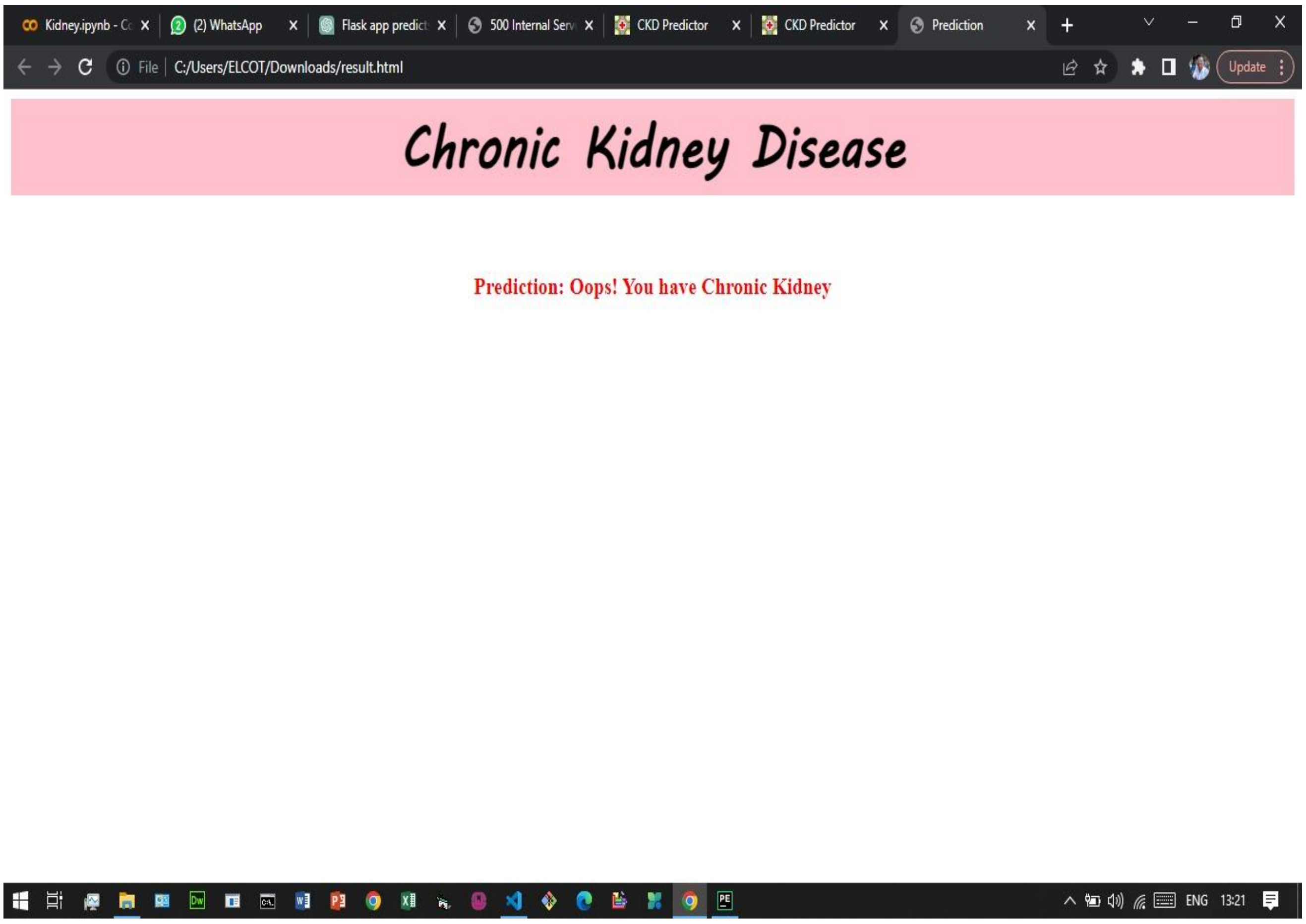

Kidney.ipynb - Co | WhatsApp | 500 Internal Serv | Flask app predict | 500 Internal Serv | CKD Predictor | CKD Predictor

127.0.0.1:5000/Prediction?

Chronic Kidney Disease

A Machine Learning Web App, Built with Flask

1
1
NO
NO
normal
normal
YES
YES



Chronic Kidney Disease

Prediction: Oops! You have Chronic Kidney

4ADVANTAGE AND DISADVANTAGE

Advantages :

- Early intervention: Early prediction of CKD allows healthcare providers to initiate timely interventions to prevent or delay the onset of CKD. This can include lifestyle modifications, medication management, and other targeted interventions to reduce risk factors and promote kidney health, potentially slowing down the progression of the disease and preventing complications.
- Improved patient outcomes: Detecting CKD early and managing it proactively can lead to improved patient outcomes. Patients who receive early interventions and appropriate management strategies are more likely to have better kidney function, experience fewer complications, and have a higher quality of life compared to those with late-stage CKD.
- Cost-effective: Early prediction and management of CKD can potentially reduce healthcare costs associated with treating advanced stages of CKD, including costly interventions such as dialysis or kidney transplantation. Early interventions are generally less expensive than managing complications associated with advanced CKD, making it a cost-effective approachtohealthcare management**.**

Disadvantages:

- False positives/negatives: Predictive models used for early CKD detection may have limitations, including false positives or false negatives. False positives may lead to unnecessary interventions or anxiety for patients, while false negatives may result in missed opportunities for early intervention.
- Resource-intensive: Implementing a progressive approach to health management for CKD, including early prediction, may require significant resources in terms of time, personnel, and technology. This could be a

challenge, especially in resource-limited settings, where access to advanced healthcare facilities and technologies may be limited.

- Ethical considerations: Early prediction of CKD may raise ethical concerns related to issues such as privacy, informed consent, and potential discrimination. The use of patient data for predictive modeling and early detection needs to be handled carefully to ensure patient privacy and autonomy are respected.

5 APPLICATION

Application of Job Prediction

I apologize, but I'm not sure I understand your question. It seems like you are asking about job prediction in the context of early prediction for chronic kidney disease (CKD) detection and a progressive approach to health management. However, CKD detection and health management do not typically involve job prediction as they are related to medical and healthcare practices.

If you are looking for information about job prediction or career development in a different context, please provide more details or clarify your question, and I would be happy to assist you to the best of my abilities.

The concept of "job prediction" seems to be unrelated to the topic of "Early Prediction for Chronic Kidney Disease Detection: A Progressive Approach to Health Management." It appears to be a misunderstanding or an error in the query. The topic of early prediction for chronic kidney disease detection involves utilizing various techniques and strategies to identify individuals at risk of developing CKD in its early stages and managing the condition proactively. It does not have direct relevance to job prediction or employment-related applications. If you have any specific questions related to early prediction for CKD detection or health management, I would be happy to assist**.**

6 CONCLUSION

Conclusion:

In conclusion, early prediction for chronic kidney disease (CKD) detection using a progressive approach to health management is a valuable strategy to identify individuals at risk of CKD in its early stages and manage the condition proactively. By implementing interventions at an early stage, such as lifestyle modifications, medication management, and other targeted interventions, healthcare providers can potentially slow down the progression of CKD, prevent complications, and improve patient outcomes.

7 Future Scope

Future Scope

1.Enhanced predictive models: The development of more advanced and accurate predictive models using machine learning, artificial intelligence (AI), and big data analytics could improve the accuracy and reliability of early prediction for CKD. This could involve incorporating additional risk factors, genetic data, and other relevant health information to further refine the prediction models.

2.Personalized medicine: The use of personalized medicine approaches, such as genomic profiling and precision medicine, could enable more tailored and individualized interventions for CKD patients. This could involve identifying specific genetic markers or biomarkers that are associated with CKD risk and using them to personalize treatment plans and interventions.

3.Remote monitoring and telehealth: The advancement of telehealth and remote monitoring technologies could enable more efficient and convenient monitoring of CKD patients, especially in remote or underserved areas. This could involve the use of wearable devices, remote sensors, and telecommunication technologies to monitor CKD patients' health status, provide timely interventions, and optimize health management remotely.

Digital health interventions: The use of digital health interventions, such as mobile apps, virtual health programs, and online education, could facilitate early prediction and health management for CKD. These interventions could provide patients with information, tools, and resources to self-monitor, manage risk factors, and adhere to treatment plans, improving patient engagement and outcomes.

8.APPENDIX

8.1Source Code

1.app.py

```python
# -*- coding: utf-8 -*-
"""app.ipynb

Automatically generated by Colaboratory.

Original file is located at
    https://colab.research.google.com/drive/1QmIjKVSnvOAHq
HZk70-qAej7wH0FErli
"""

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

app=Flask(__name__)
model=pickle.load(open('lgr.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=[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','diabetesmellitus','pedal_edema']
    df=pd.DataFrame(features_value, columns=features_name)
    output=model.predict(df)
    render_template('result.html',prediction_text=output)

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

Kidney.ipynb

```
# -*- coding: utf-8 -*-
"""Kidney.ipynb

Automatically generated by Colaboratory.

Original file is located at
    https://colab.research.google.com/drive/1oEti91eLvrwd0
v4wc9DJW_xSIrtZj4Pn
"""

import pandas as pd
import numpy as np
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 LinearRegression
import pickle

data=pd.read_csv("kidney_disease.csv")
data

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

data

data.columns

data.columns=['age','blood_pressure','specific_gravity','a
lbumin',
              'sugar','red_blood_cells','pus_cell','pus_ce
ll_clumps','bacteria',
              'blood glucose
random','blood_urea','serum_creatinine','sodium','potassiu
m',
              'hemoglobin','packed_cell_volume','white_blo
od_cell_count','red_blood_cell_count',
              'hypertension','diabetesmellitus','coronary_
artery_disease','appetite',
              'pedal_edema','anemia','class'] # manually
giving the name  of the columns
data.columns

data['class'].unique()

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

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

```
for i in 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)

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

for i in contcols:
  print("Continuos 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_disea
se.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.packed_cell_volume=pd.to_numeric(data.packed_cell_vol
ume, 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_cel
l_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(),inplac
e=True)
data['hemoglobin'].fillna(data['hemoglobin'].mean(),inplac
e=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_c
ount'].mean(),inplace=True)
data['serum_creatinine'].fillna(data['serum_creatinine'].m
ean(),inplace=True)
data['sodium'].fillna(data['sodium'].mean(),inplace=True)
data['white_blood_cell_count'].fillna(data['white_blood_ce
ll_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'].mod
e()[0],inplace=True)
data['appetite'].fillna(data['appetite'].mode()[0],inplace
=True)
data['albumin'].fillna(data['albumin'].mode()[0],inplace=T
rue)
data['pus_cell'].fillna(data['pus_cell'].mode()[0],inplace
=True)
data['red_blood_cells'].fillna(data['red_blood_cells'].mod
e()[0],inplace=True)
data['coronary_artery_disease'].fillna(data['coronary_arte
ry_disease'].mode()[0],inplace=True)
data['bacteria'].fillna(data['bacteria'].mode()[0],inplace
=True)
data['anemia'].fillna(data['anemia'].mode()[0],inplace=Tru
e)
data['sugar'].fillna(data['sugar'].mode()[0],inplace=True)
data['diabetesmellitus'].fillna(data['diabetesmellitus'].m
ode()[0],inplace=True)
data['pedal_edema'].fillna(data['pedal_edema'].mode()[0],i
nplace=True)
data['pedal_edema'].fillna(data['pedal_edema'].mode()[0],i
nplace=True)

for i in catcols:
  print("LABEL ENCODING OF:",i)
  LEi = LabelEncoder() # creating an object of
LabelEncoder
  print(c(data[i])) #getting the classes values before
transformation
  data[i] = LEi.fit_transform(data[i])# trannsforming our
text classes to numerical values
  print(c(data[i])) #getting the classes values after
transformation
  print("*"*100)
```

```
data

x=data.iloc[:,0:25]
y=data.iloc[:,24]



x_train,x_test,y_train,y_test=train_test_split(x,y,test_si
ze=0.2,random_state=2)#train test split
print(x_train.shape)
print(y_train.shape)
print(x_test.shape)
print(y_test.shape)

from sklearn.ensemble import RandomForestRegressor
rf = RandomForestRegressor()

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

y_pred = lgr.predict(x_test)

print(y_pred)

from sklearn.metrics import r2_score
acc= r2_score(y_pred,y_test)
acc

import pickle
pickle.dump(lgr,open('lgr.pkl','wb'))

conf_mat = confusion_matrix(y_test,y_pred)
conf_mat

x_train.shape
```

```
data.columns
```

```
import pandas as pd
import numpy as np
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 LinearRegression
import pickle
```

```
data=pd.read_csv("kidney_disease.csv")
data
```

	id	age	bp	sg	al	su	rbc	pc	pcc	ba	...	pcv	
0	0	48.0	80.0	1.020	1.0	0.0	NaN	normal	notpresent	notpresent	...	44	7
1	1	7.0	50.0	1.020	4.0	0.0	NaN	normal	notpresent	notpresent	...	38	6
2	2	62.0	80.0	1.010	2.0	3.0	normal	normal	notpresent	notpresent	...	31	7
3	3	48.0	70.0	1.005	4.0	0.0	normal	abnormal	present	notpresent	...	32	6
4	4	51.0	80.0	1.010	2.0	0.0	normal	normal	notpresent	notpresent	...	35	7
...	...	...	...	...	...	...	...	...	...	...	...	...	
395	395	55.0	80.0	1.020	0.0	0.0	normal	normal	notpresent	notpresent	...	47	6
396	396	42.0	70.0	1.025	0.0	0.0	normal	normal	notpresent	notpresent	...	54	7
397	397	12.0	80.0	1.020	0.0	0.0	normal	normal	notpresent	notpresent	...	49	6
398	398	17.0	60.0	1.025	0.0	0.0	normal	normal	notpresent	notpresent	...	51	7
399	399	58.0	80.0	1.025	0.0	0.0	normal	normal	notpresent	notpresent	...	53	6

400 rows × 26 columns

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

```
data
```

	age	bp	sg	al	su	rbc	pc	pcc	ba	bgr	...	pcv
0	48.0	80.0	1.020	1.0	0.0	NaN	normal	notpresent	notpresent	121.0	...	44
1	7.0	50.0	1.020	4.0	0.0	NaN	normal	notpresent	notpresent	NaN	...	38
2	62.0	80.0	1.010	2.0	3.0	normal	normal	notpresent	notpresent	423.0	...	31
3	48.0	70.0	1.005	4.0	0.0	normal	abnormal	present	notpresent	117.0	...	32
4	51.0	80.0	1.010	2.0	0.0	normal	normal	notpresent	notpresent	106.0	...	35
...	...	...	...	...	...	...	...	...	...	...	...	...
395	55.0	80.0	1.020	0.0	0.0	normal	normal	notpresent	notpresent	140.0	...	47
396	42.0	70.0	1.025	0.0	0.0	normal	normal	notpresent	notpresent	75.0	...	54
397	12.0	80.0	1.020	0.0	0.0	normal	normal	notpresent	notpresent	100.0	...	49
398	17.0	60.0	1.025	0.0	0.0	normal	normal	notpresent	notpresent	114.0	...	51
399	58.0	80.0	1.025	0.0	0.0	normal	normal	notpresent	notpresent	131.0	...	53

400 rows × 25 columns

```
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','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'] # manually giving the name  of the columns
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')
```

```
data['class'].unique()
```

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

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

```
0         ckd
1         ckd
2         ckd
3         ckd
4         ckd
        ...
395    notckd
396    notckd
397    notckd
398    notckd
399    notckd
Name: class, Length: 400, dtype: object
```

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

```
{'bacteria', 'class', 'appetite', 'coronary_artery_disease', 'red_blood_cells', 'pus_cell', 'red_blood_cell_count', 'anemia', 'hype
```

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

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

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

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

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

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

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

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'
************************************************************************************************************************

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

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

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

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':
************************************************************************************************************************

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

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

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, '
************************************************************************************************************************
```

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

```
{'bacteria', 'class', 'appetite', 'coronary_artery_disease', 'red_blood_cells', 'pus_cell', 'anemia', 'hypertension', 'pus_cell_clu
```

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

```
{'age', 'serum_creatinine', 'blood glucose random', 'sodium', 'specific_gravity', 'sugar', 'albumin', 'hemoglobin', 'potassium', 'b
```

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

```
Continuos 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.
************************************************************************************************************************

Continuos 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
************************************************************************************************************************

Continuos 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:
************************************************************************************************************************

Continuos 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, 1
************************************************************************************************************************

Continuos 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:
************************************************************************************************************************

Continuos 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:
************************************************************************************************************************

Continuos 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:
************************************************************************************************************************

Continuos 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:
************************************************************************************************************************

Continuos 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:
************************************************************************************************************************

Continuos 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
************************************************************************************************************************

Continuos 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, 2
************************************************************************************************************************
```

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

```
print(contcols)
```

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

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

```
{'age', 'red_blood_cell_count', 'serum_creatinine', 'blood glucose random', 'sodium', 'hemoglobin', 'white_blood_cell_count', 'pota
```

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

```
{'bacteria', 'class', 'appetite', 'coronary_artery_disease', 'red_blood_cells', 'pus_cell', 'anemia', 'specific_gravity', 'hyperten
```

```
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': 137, 'no': 261, nan: 2})
```

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

```
for i in catcols:
  print("LABEL ENCODING OF:",i)
  LEi = LabelEncoder() # creating an object of LabelEncoder
  print(c(data[i])) #getting the classes values before transformation
  data[i] = LEi.fit_transform(data[i])# trannsforming our text classes to numerical values
  print(c(data[i])) #getting the classes values after transformation
  print("*"*100)
```

```
LABEL ENCODING OF: bacteria
Counter({'notpresent': 378, 'present': 22})
Counter({0: 378, 1: 22})
****************************************************************************************************
LABEL ENCODING OF: class
Counter({'ckd': 250, 'notckd': 150})
Counter({0: 250, 1: 150})
****************************************************************************************************
LABEL ENCODING OF: appetite
```

```
Counter({'good': 318, 'poor': 82})
Counter({0: 318, 1: 82})
****************************************************************************************************
LABEL ENCODING OF: coronary_artery_disease
Counter({'no': 366, 'yes': 34})
Counter({0: 366, 1: 34})
****************************************************************************************************
LABEL ENCODING OF: red_blood_cells
Counter({'normal': 353, 'abnormal': 47})
Counter({1: 353, 0: 47})
****************************************************************************************************
LABEL ENCODING OF: pus_cell
Counter({'normal': 324, 'abnormal': 76})
Counter({1: 324, 0: 76})
****************************************************************************************************
LABEL ENCODING OF: anemia
Counter({'no': 340, 'yes': 60})
Counter({0: 340, 1: 60})
****************************************************************************************************
LABEL ENCODING OF: 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, nar
Counter({3: 106, 1: 84, 4: 81, 2: 75, 5: 47, 0: 7})
****************************************************************************************************
LABEL ENCODING OF: hypertension
Counter({'no': 253, 'yes': 147})
Counter({0: 253, 1: 147})
****************************************************************************************************
LABEL ENCODING OF: pus_cell_clumps
Counter({'notpresent': 358, 'present': 42})
Counter({0: 358, 1: 42})
****************************************************************************************************
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: pedal_edema
Counter({'no': 324, 'yes': 76})
Counter({0: 324, 1: 76})
****************************************************************************************************
LABEL ENCODING OF: diabetesmellitus
Counter({'no': 263, 'yes': 137})
Counter({0: 263, 1: 137})
****************************************************************************************************
```

```
data
```

	age	blood_pressure	specific_gravity	albumin	sugar	red_blood_cells	pus_cell	pus_cell_clumps	bacteria	blood glucose random	...	p
0	48.0	80.0	3	1	0	1	1	0	0	121.000000	...	
1	7.0	50.0	3	4	0	1	1	0	0	148.036517	...	
2	62.0	80.0	1	2	3	1	1	0	0	423.000000	...	
3	48.0	70.0	0	4	0	1	0	1	0	117.000000	...	
4	51.0	80.0	1	2	0	1	1	0	0	106.000000	...	
...	...	...	...	...	...	...	...	...	...	...	...	
395	55.0	80.0	3	0	0	1	1	0	0	140.000000	...	
396	42.0	70.0	4	0	0	1	1	0	0	75.000000	...	
397	12.0	80.0	3	0	0	1	1	0	0	100.000000	...	
398	17.0	60.0	4	0	0	1	1	0	0	114.000000	...	
399	58.0	80.0	4	0	0	1	1	0	0	131.000000	...	

400 rows × 25 columns

```
x=data.iloc[:,0:25]
y=data.iloc[:,24]
```

```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=2)#train test split
print(x_train.shape)
print(y_train.shape)
print(x_test.shape)
print(y_test.shape)
```

```
(320, 25)
(320,)
(80, 25)
(80,)
```

```
from sklearn.ensemble import RandomForestRegressor
rf = RandomForestRegressor()
```

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

```
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:458: ConvergenceWarning: lbfgs failed to converge (status
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
▾ LogisticRegression
LogisticRegression()
```

```
y_pred = lgr.predict(x_test)

print(y_pred)
```

```
[0 0 0 0 1 0 0 0 1 0 0 0 0 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 1 1 0 1 0 0 0 1 0 1 1 1 0 0 0 0 0 0 0 1 0 0 0 1 1 0 0 0 0 1 1 1 1 0 0 0
 0 0 0 1 1 0]
```

```
from sklearn.metrics import r2_score
acc= r2_score(y_pred,y_test)
acc
```

```
0.6703296703296704
```

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

```
conf_mat = confusion_matrix(y_test,y_pred)
conf_mat
```

```
array([[50,  4],
       [ 2, 24]])
```

```
x_train.shape
```

```
(320, 25)
```

```
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')
```