Intelligent Admission :

PROJECT RECORD TEMPLATE

CHAPTER

1.INTRODUCTION

1.1 OVER VIEW

"Early prediction for chronic kidney disease detection: A progressive approach to health management" suggests a focus on utilizing technology and data analysis to detect chronic kidney disease at an early stage. Chronic kidney

disease is a serious medical condition that can lead to kidney failure if left untreated. Early detection and management of the disease can significantly improve patient outcomes and quality of life.

The approach to health management outlined in the title likely involves using machine learning algorithms and predictive modeling to analyze patient data and identify those at high risk for developing chronic kidney disease. By identifying patients early, healthcare providers can intervene with preventative measures and early treatment options, such as lifestyle changes and medication, to slow the progression of the disease.

The use of technology in healthcare is becoming increasingly prevalent, and the application of data analysis and machine learning in chronic kidney disease detection is a promising area of research. A progressive

approach to health management, as suggested in the title, involves continuously improving and refining these technologies and approaches to provide better patient care and outcomes.

1.1 PURPOSE

"Early prediction for chronic kidney disease detection: A progressive approach to health management" is to highlight the importance of early detection and management of chronic kidney disease (CKD) in order to prevent its progression and associated complications.

Chronic kidney disease is a serious and potentially life-threatening condition that affects millions of people worldwide. Unfortunately, many individuals may not even be aware they have CKD until it has reached an advanced stage, which can result in irreversible damage to the kidneys and other organs.

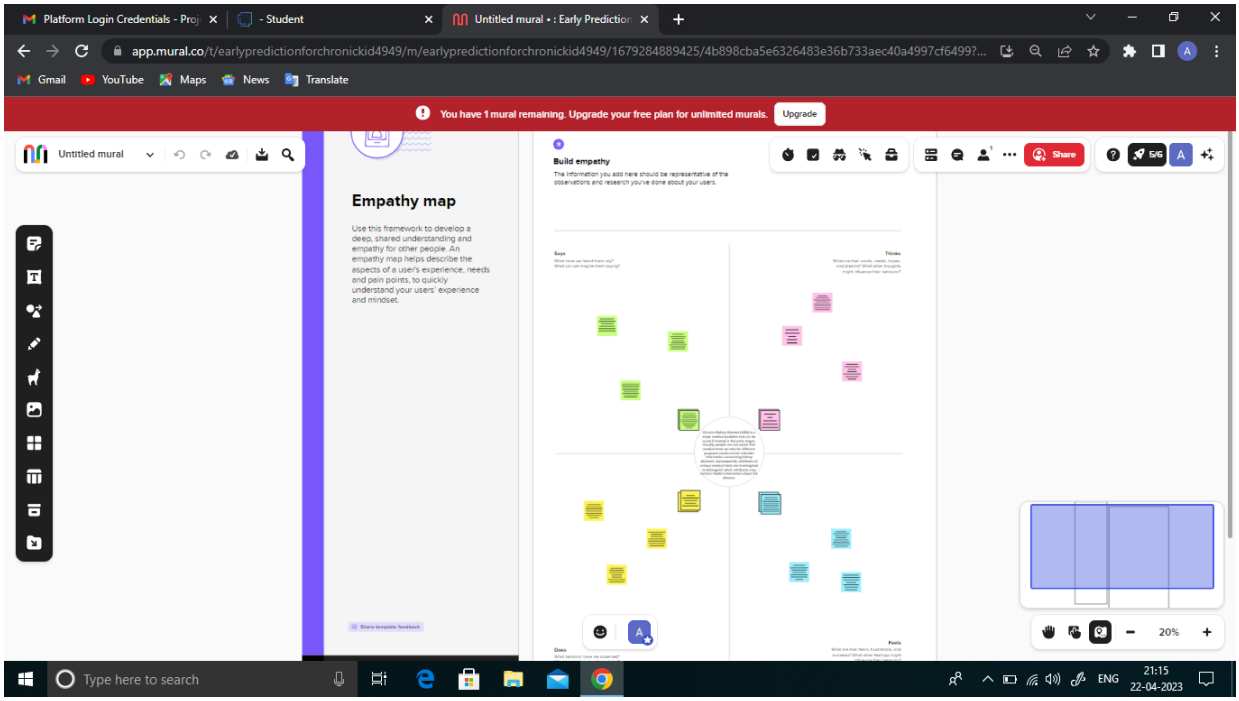
The use of a progressive approach to health management, including early detection and prediction of CKD, can help identify individuals who are at risk of developing CKD, and allow for interventions that can slow or halt its progression. This can improve outcomes for patients and reduce the burden on the healthcare system by preventing the need for costly and invasive treatments like dialysis or kidney transplants.

Overall, the purpose of the title is to emphasize the importance of early detection and management of CKD in improving patient outcomes and reducing the burden on the healthcare system

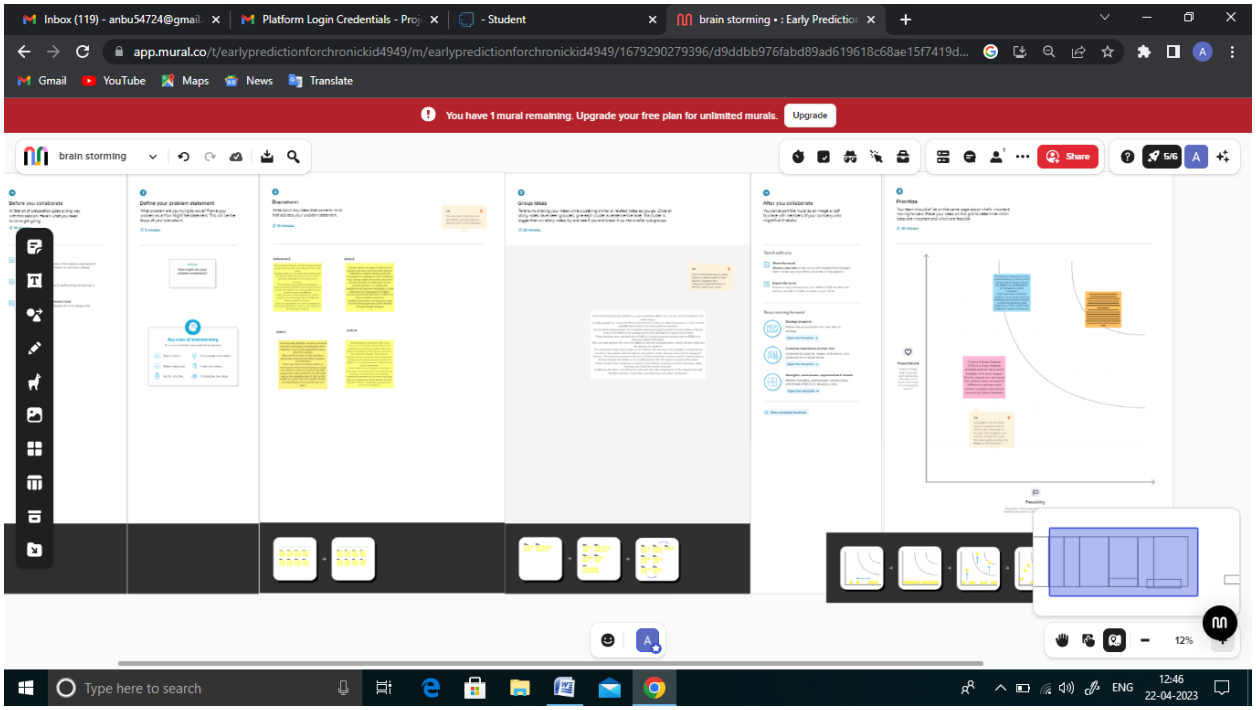
CHAPTER 2

2. PROBLEM DEFINITION & DESIGN THINKING

2.1 PROBLEM DEFINITION:



2.2 IDEATION & BRAINSTROMING MAP:



CHAPTER 3

3.Result

Result 1:

	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

Result 2:

```
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')
```

Result 3:

```
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')
```

Result 4:

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

Result 5:

```
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
```

Result 6:

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

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

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

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

Columns : red_blood_cell_count
Counter({nan: 130, '5.2': 18, '4.5': 16, '4.9': 14, '4.7': 11, '3.9': 10, '4.8': 10, '4.6': 9, '3.4': 9, '3.7': 8, '5.0': 8, '6.1': 8, '5.
5': 8, '5.9': 8, '3.8': 7, '5.4': 7, '5.8': 7, '5.3': 7, '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, '6.0': 4, '6.3': 4, '4.0': 3, '4': 3, '3.5': 3, '3.3': 3, '5': 2, '2.6': 2, '2.8': 2, '2.5': 2,
'3.1': 2, '2.1': 2, '2.9': 2, '2.7': 2, '3.0': 2, '2.3': 1, '8.0': 1, '3': 1, '2.4': 1, '\t?': 1})
**********************************************************************************************************************
```

Result 7:

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

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

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

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

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

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

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': 7, '940
0': 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, '710
0': 2, '13200': 2, '9000': 2, '8200': 2, '15200': 2, '12400': 2, '12800': 2, '8800': 2, '5700': 2, '9300': 2, '6600': 2, '12100': 1, '1220
0': 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, '1200
0': 1, '15700': 1, '4100': 1, '11500': 1, '10800': 1, '9900': 1, '5200': 1, '5900': 1, '9700': 1, '5100': 1})
**********************************************************************************************************************
```

Result 8:

```
LABEL ENCODING OF: anemia
Counter({'no': 340, 'yes': 60})
Counter({0: 340, 1: 60})
**********************************************************************************************************
LABEL ENCODING OF: pedal_edema
Counter({'no': 324, 'yes': 76})
Counter({0: 324, 1: 76})
**********************************************************************************************************
LABEL ENCODING OF: appetite
Counter({'good': 318, 'poor': 82})
Counter({0: 318, 1: 82})
**********************************************************************************************************
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: coronary_artery_disease
Counter({'no': 366, 'yes': 34})
Counter({0: 366, 1: 34})
**********************************************************************************************************
LABEL ENCODING OF: diabetesmellitus
Counter({'no': 263, 'yes': 137})
Counter({0: 263, 1: 137})
**********************************************************************************************************
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: pus_cell_clumps
Counter({'notpresent': 358, 'present': 42})
Counter({0: 358, 1: 42})
**********************************************************************************************************
LABEL ENCODING OF: red_blood_cells
Counter({'normal': 353, 'abnormal': 47})
```

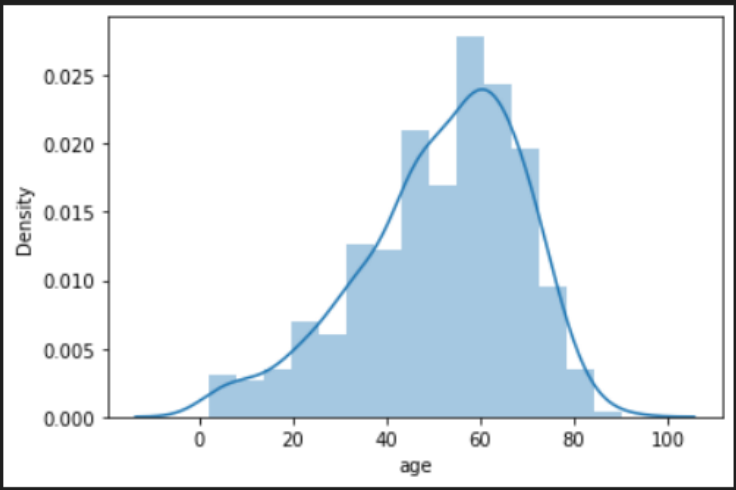
Result 9:

	age	blood_pressure	specific_gravity	albumin	sugar	blood glucose random	blood_urea	serum_creatinine	sodium
count	391.000000	388.000000	353.000000	354.000000	351.000000	356.000000	381.000000	383.000000	313.000000
mean	51.483376	76.469072	1.017408	1.016949	0.450142	148.036517	57.425722	3.072454	137.528754
std	17.169714	13.683637	0.005717	1.352679	1.099191	79.281714	50.503006	5.741126	10.408752
min	2.000000	50.000000	1.005000	0.000000	0.000000	22.000000	1.500000	0.400000	4.500000
25%	42.000000	70.000000	1.010000	0.000000	0.000000	99.000000	27.000000	0.900000	135.000000
50%	55.000000	80.000000	1.020000	0.000000	0.000000	121.000000	42.000000	1.300000	138.000000
75%	64.500000	80.000000	1.020000	2.000000	0.000000	163.000000	66.000000	2.800000	142.000000
max	90.000000	180.000000	1.025000	5.000000	5.000000	490.000000	391.000000	76.000000	163.000000

Result 10:

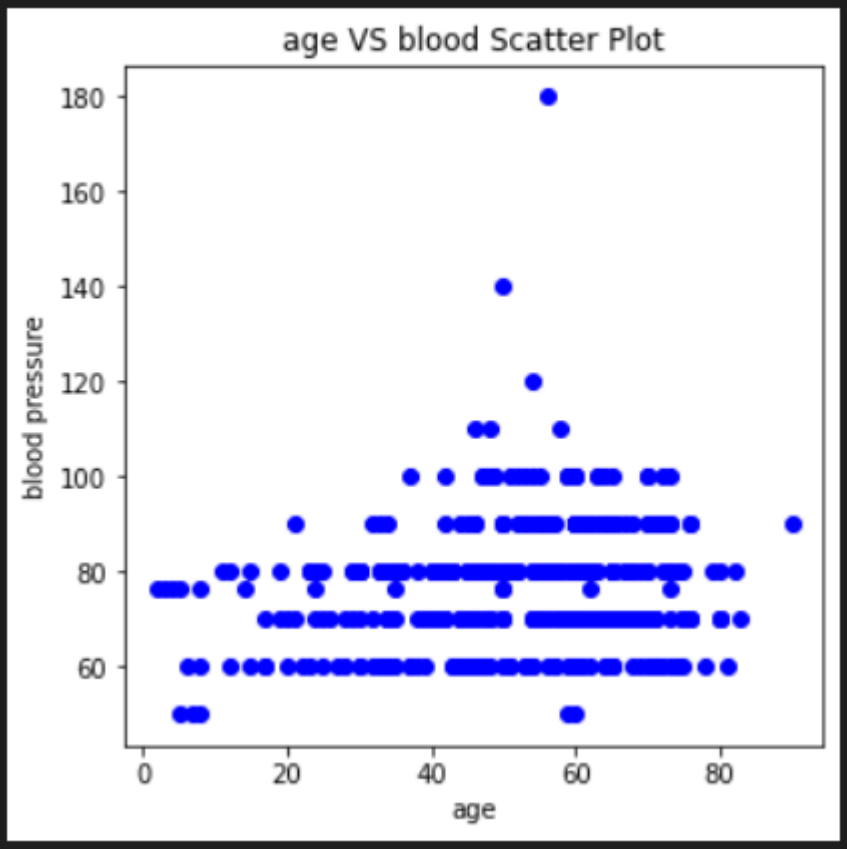
```
...  C:\Users\Saumya\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWa
     your code to use either `displot` (a figure-level function with similar flexibility
       warnings.warn(msg, FutureWarning)

     <AxesSubplot:xlabel='age', ylabel='Density'>
```

</>

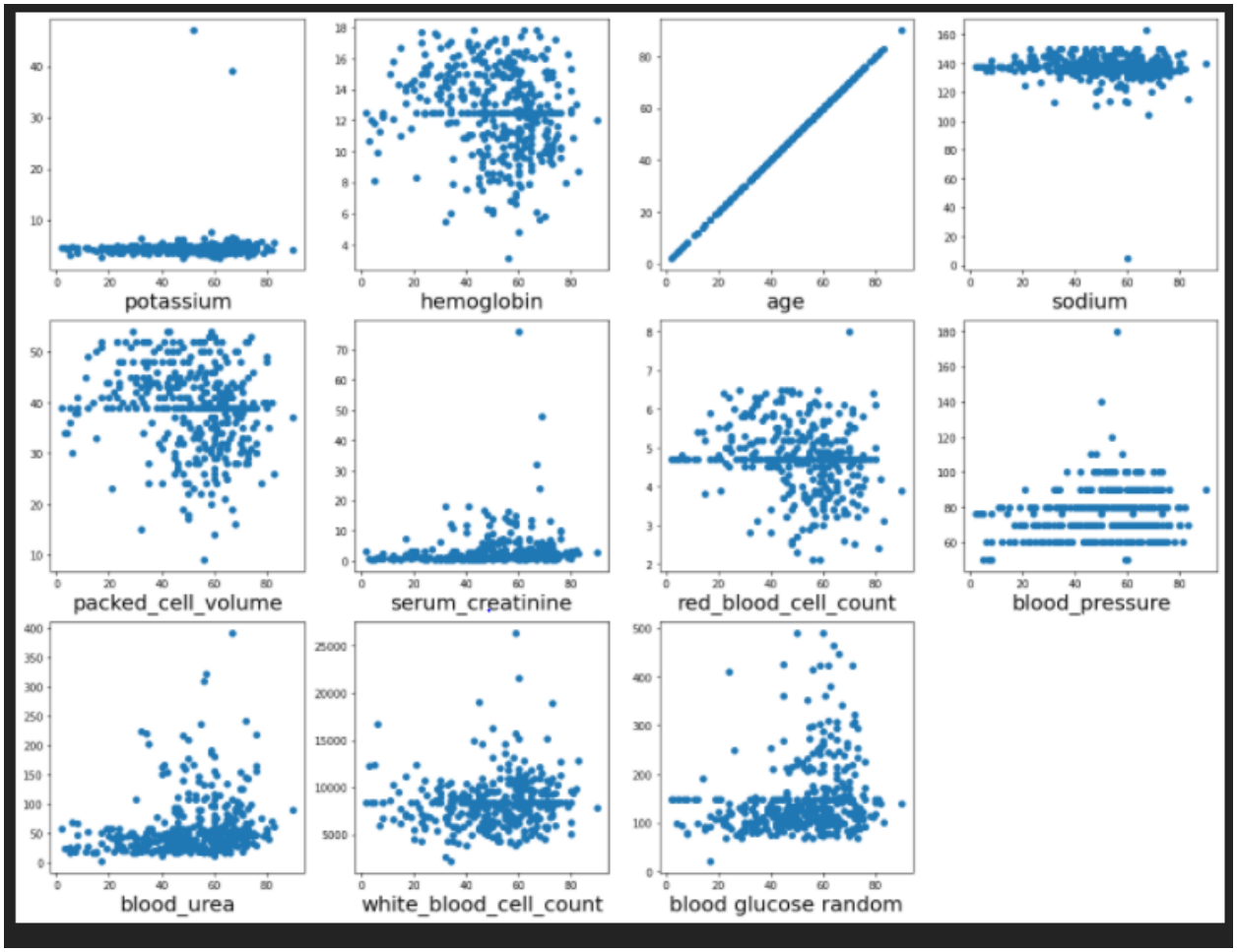


Result 11:

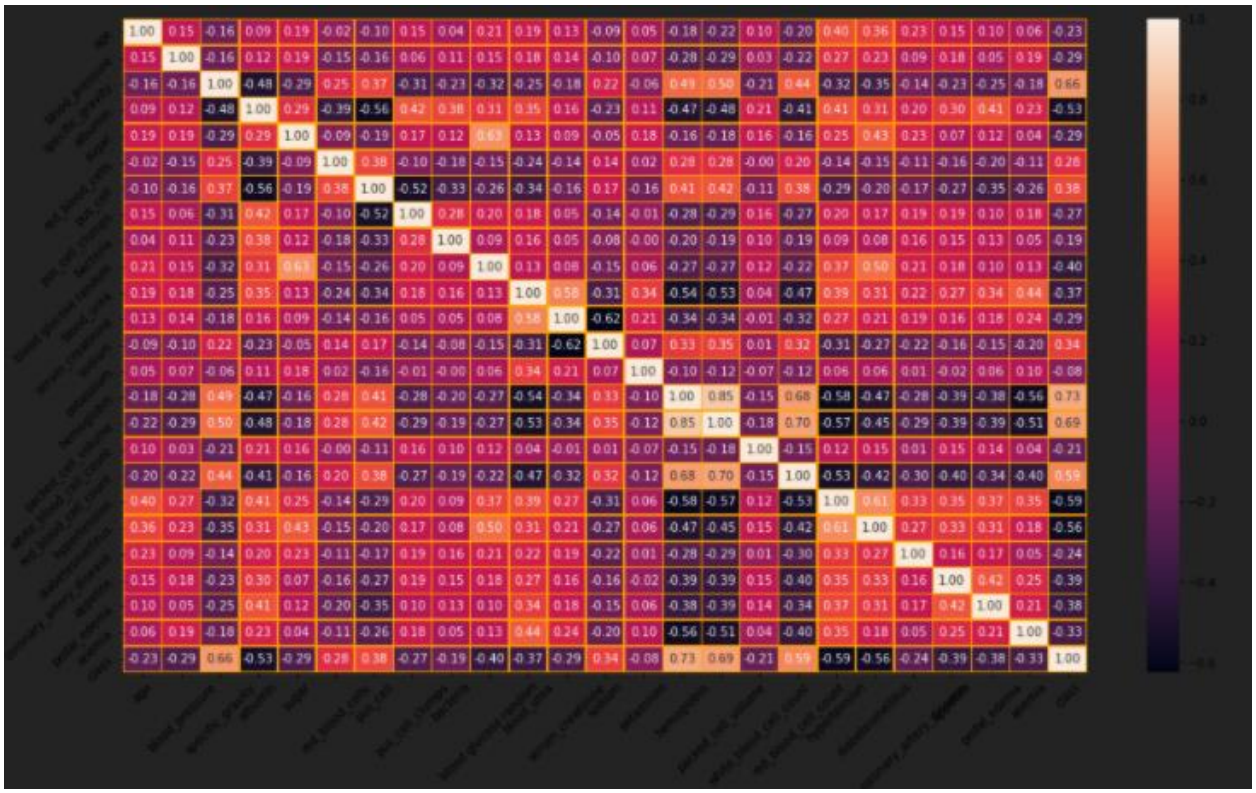
```
Text(0.5, 1.0, 'age VS blood Scatter Plot')
```



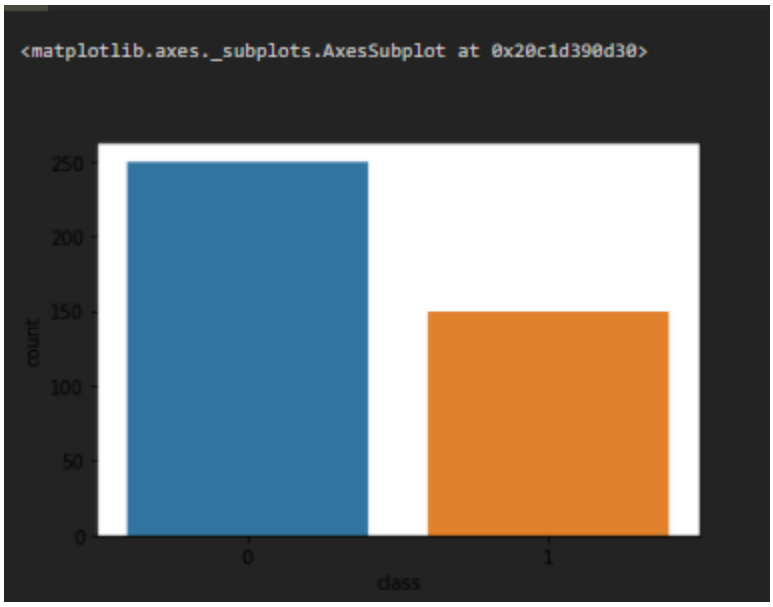
Result 12:


potassium
hemoglobin
age
sodium
packed_cell_volume
serum_creatinine
red_blood_cell_count
blood_pressure
blood_urea
white_blood_cell_count
blood glucose random

Result 13:



Result 14:



Result 15:

```
Output exceeds the size limit. Open the full output data in a text editor
Epoch 1/100
26/26 [==============================] - 0s 6ms/step - loss: 0.1151 - accuracy: 0.9531 - val_loss: 0.2476 - val_accuracy: 0.9062
Epoch 2/100
26/26 [==============================] - 0s 4ms/step - loss: 0.1171 - accuracy: 0.9570 - val_loss: 0.2498 - val_accuracy: 0.9062
Epoch 3/100
26/26 [==============================] - 0s 4ms/step - loss: 0.1146 - accuracy: 0.9531 - val_loss: 0.2317 - val_accuracy: 0.9219
Epoch 4/100
26/26 [==============================] - 0s 4ms/step - loss: 0.1305 - accuracy: 0.9531 - val_loss: 0.2855 - val_accuracy: 0.8906
Epoch 5/100
26/26 [==============================] - 0s 4ms/step - loss: 0.1387 - accuracy: 0.9492 - val_loss: 0.2068 - val_accuracy: 0.9219
Epoch 6/100
26/26 [==============================] - 0s 4ms/step - loss: 0.1230 - accuracy: 0.9492 - val_loss: 0.2576 - val_accuracy: 0.9062
Epoch 7/100
26/26 [==============================] - 0s 4ms/step - loss: 0.1241 - accuracy: 0.9531 - val_loss: 0.2688 - val_accuracy: 0.8906
Epoch 8/100
26/26 [==============================] - 0s 4ms/step - loss: 0.1128 - accuracy: 0.9570 - val_loss: 0.2334 - val_accuracy: 0.9219
Epoch 9/100
26/26 [==============================] - 0s 4ms/step - loss: 0.1180 - accuracy: 0.9531 - val_loss: 0.2435 - val_accuracy: 0.9062
Epoch 10/100
```

Result 16:

```
[==============================] - 0s 4ms/step - loss: 0.1139 - accuracy: 0.9531 - val_loss: 0.2799 - val_accuracy: 0.8906 Ep
...
Epoch 99/100 26/26 [==============================] - 0s 3ms/step - loss: 0.1074 - accuracy: 0.9570 - val_loss: 0.2439 - va
[==============================] - 0s 4ms/step - loss: 0.1062 - accuracy: 0.9570 - val_loss: 0.2572 - val_accuracy: 0.9062

<tensorflow.python.keras.callbacks.History at 0x1fdf3ca7b20>
```

Result 17:

```
array([0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 1, 1,
       0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0,
       0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0,
       0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0])
```

Result 18:

```
Output exceeds the size limit. Open the full output data in a text editor
array([[2.07892948e-12],
       [7.16007332e-13],
       [0.00000000e+00],
       [6.47086192e-23],
       [9.99349952e-01],
       [1.47531908e-22],
```

Result 19:

```
Output exceeds the size limit. Open the full output data in a te
array([[False],
       [False],
       [False],
       [False],
       [ True],
       [False],
       [False],
```

Result 20:

LogReg

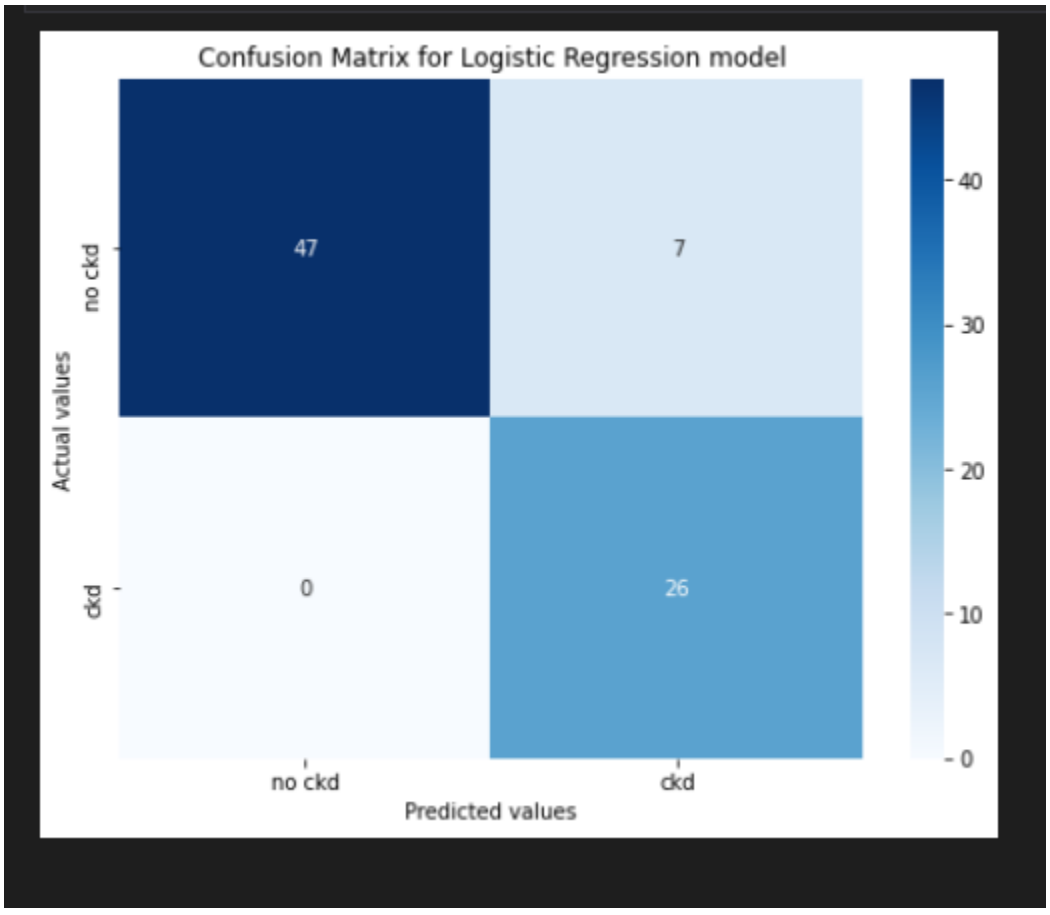
	precision	recall	f1-score	support
NO CKD	1.00	0.87	0.93	54
CKD	0.79	1.00	0.88	26
accuracy			0.91	80
macro avg	0.89	0.94	0.91	80
weighted avg	0.93	0.91	0.91	80

Result 21:

LogReg

	precision	recall	f1-score	support
NO CKD	1.00	0.87	0.93	54
CKD	0.79	1.00	0.88	26
accuracy			0.91	80
macro avg	0.89	0.94	0.91	80
weighted avg	0.93	0.91	0.91	80

Result 22:

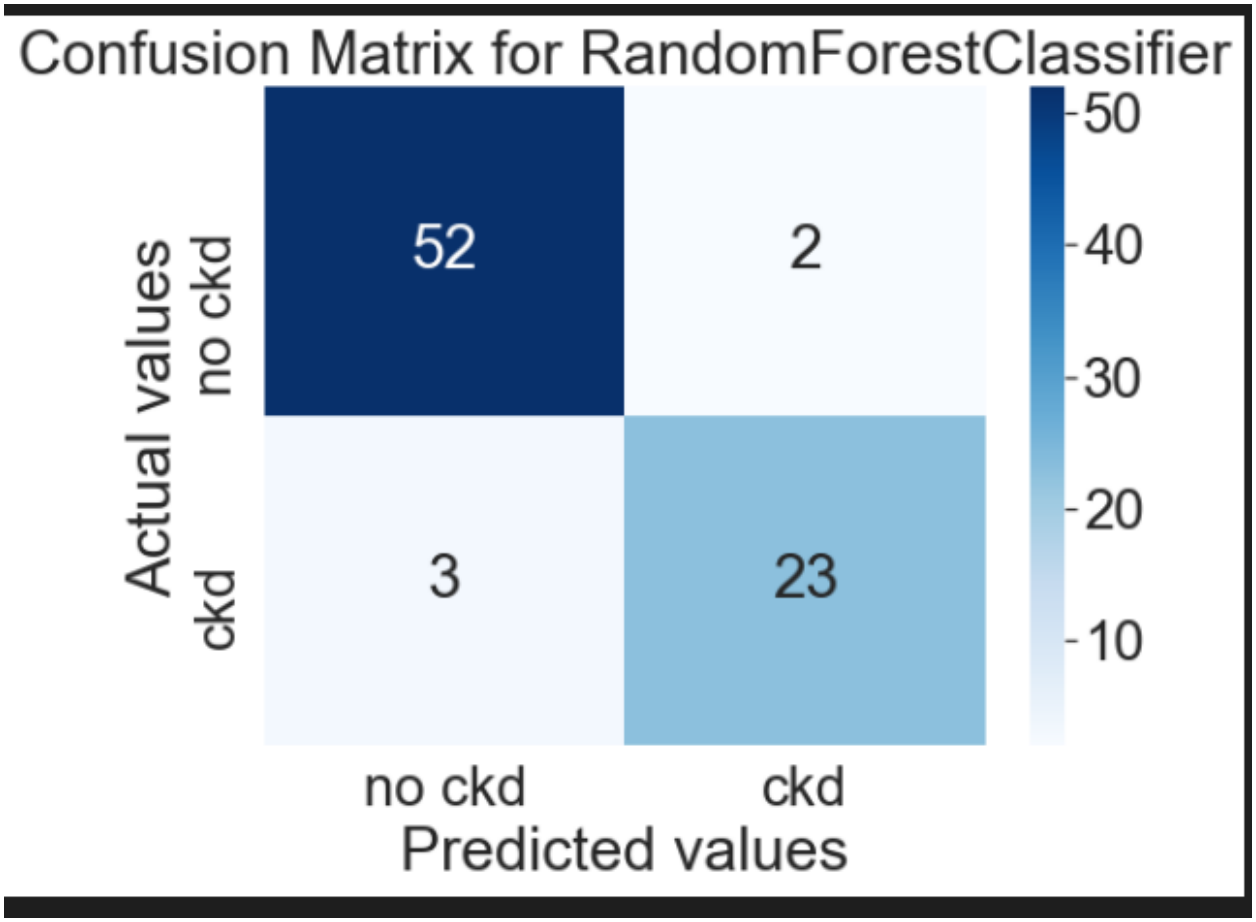


Result 23:

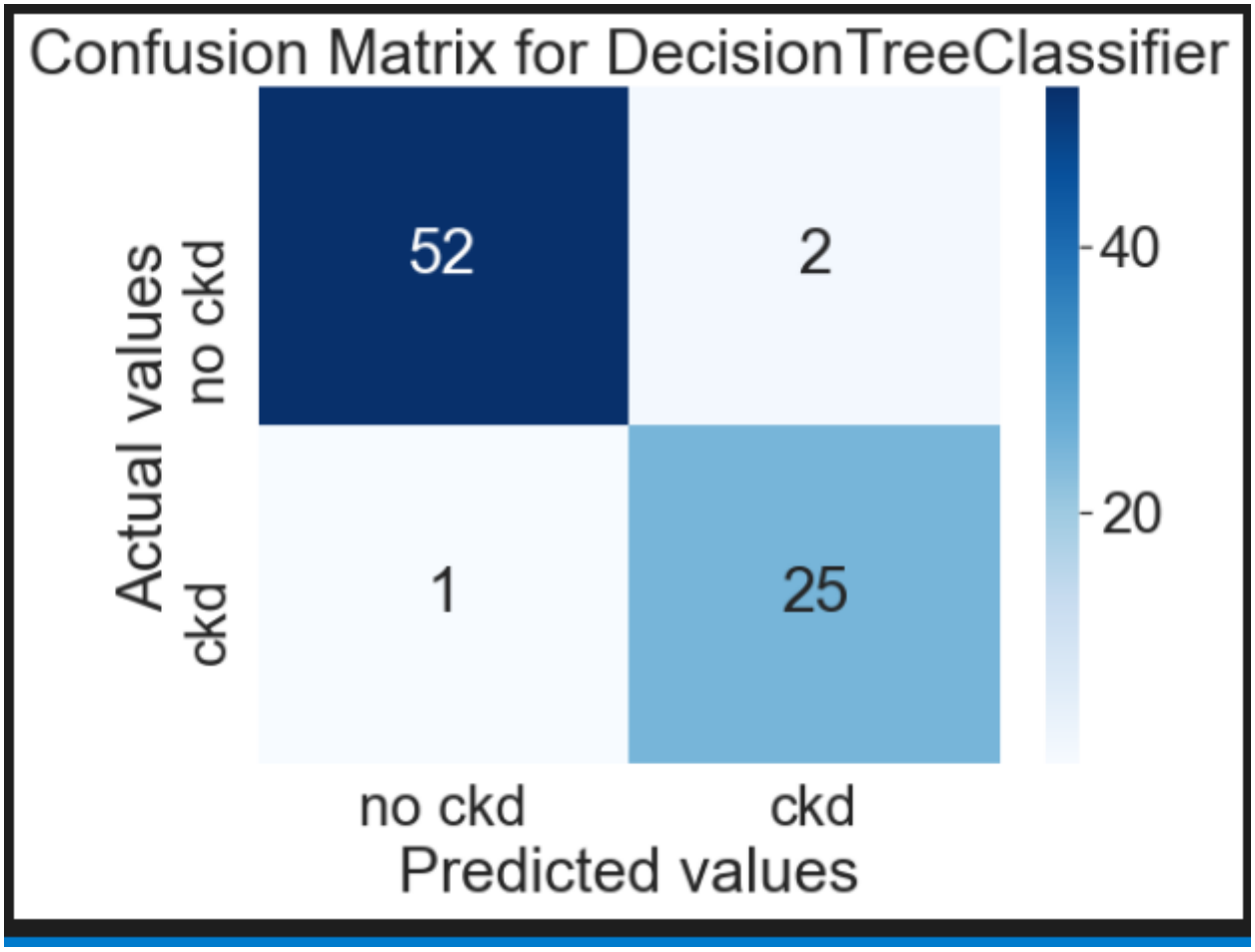
RF

	precision	recall	f1-score	support
NO CKD	0.96	0.96	0.96	54
CKD	0.92	0.92	0.92	26
accuracy			0.95	80
macro avg	0.94	0.94	0.94	80
weighted avg	0.95	0.95	0.95	80

Result 24:



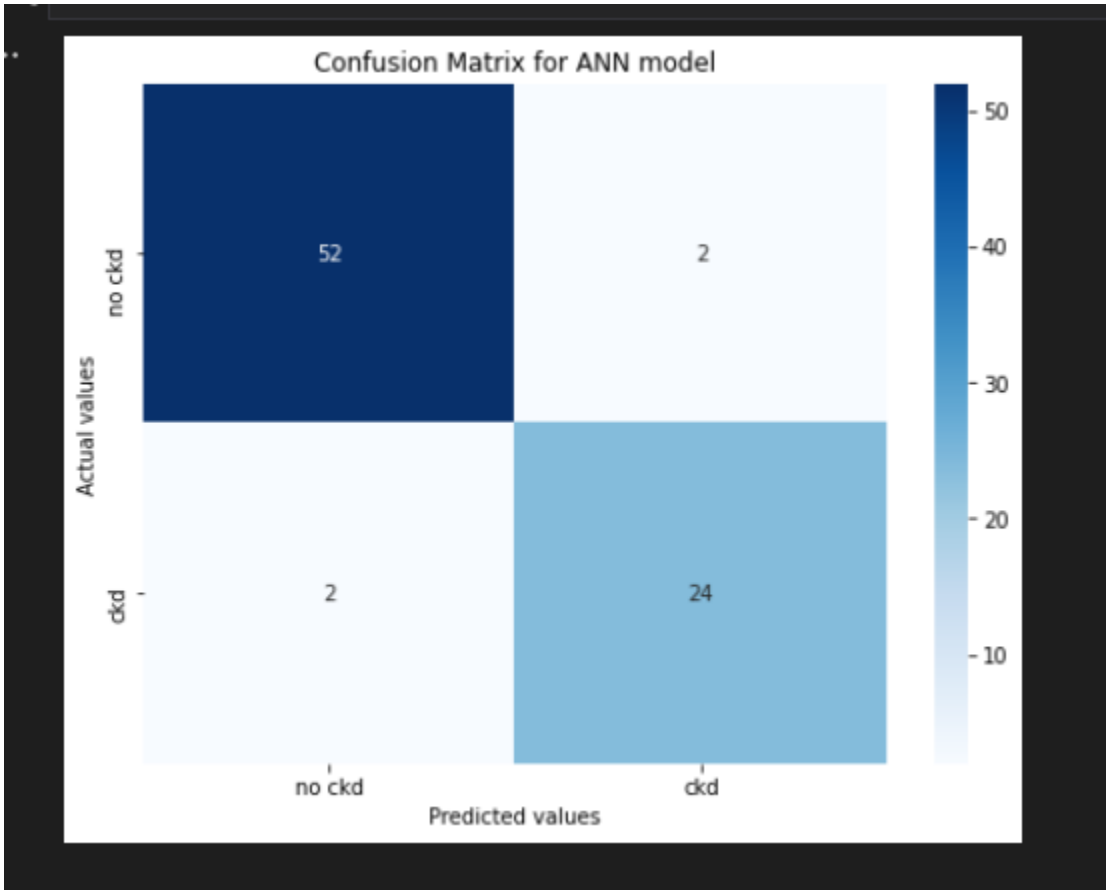
Result 25:



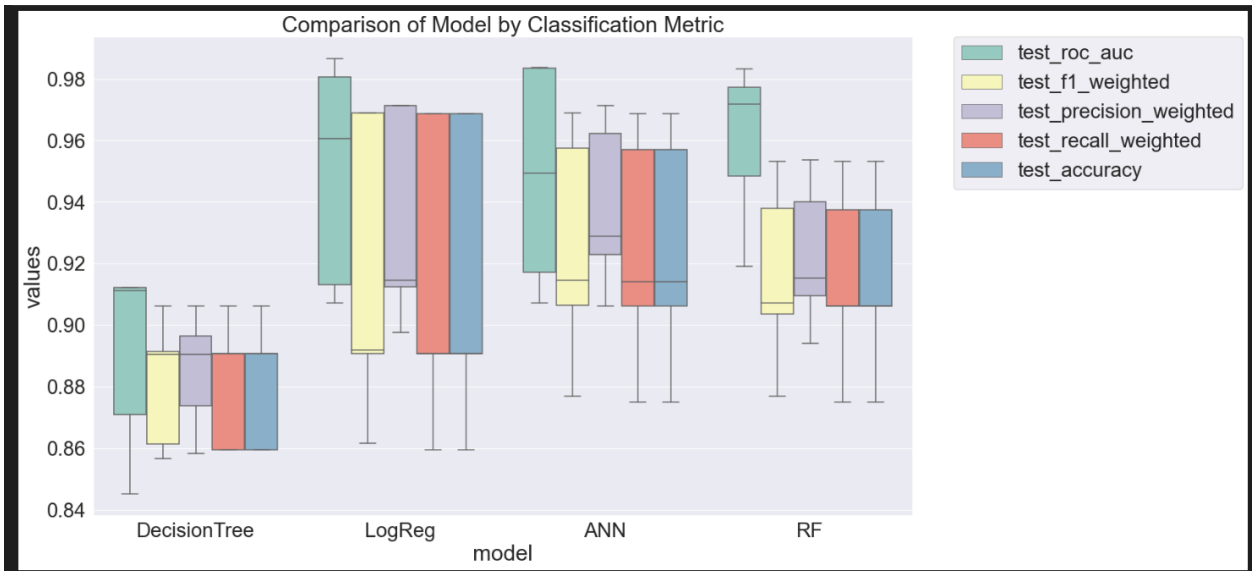
Result 26:

	precision	recall	f1-score	support
0	0.96	0.96	0.96	54
1	0.92	0.92	0.92	26
accuracy			0.95	80
macro avg	0.94	0.94	0.94	80
weighted avg	0.95	0.95	0.95	80

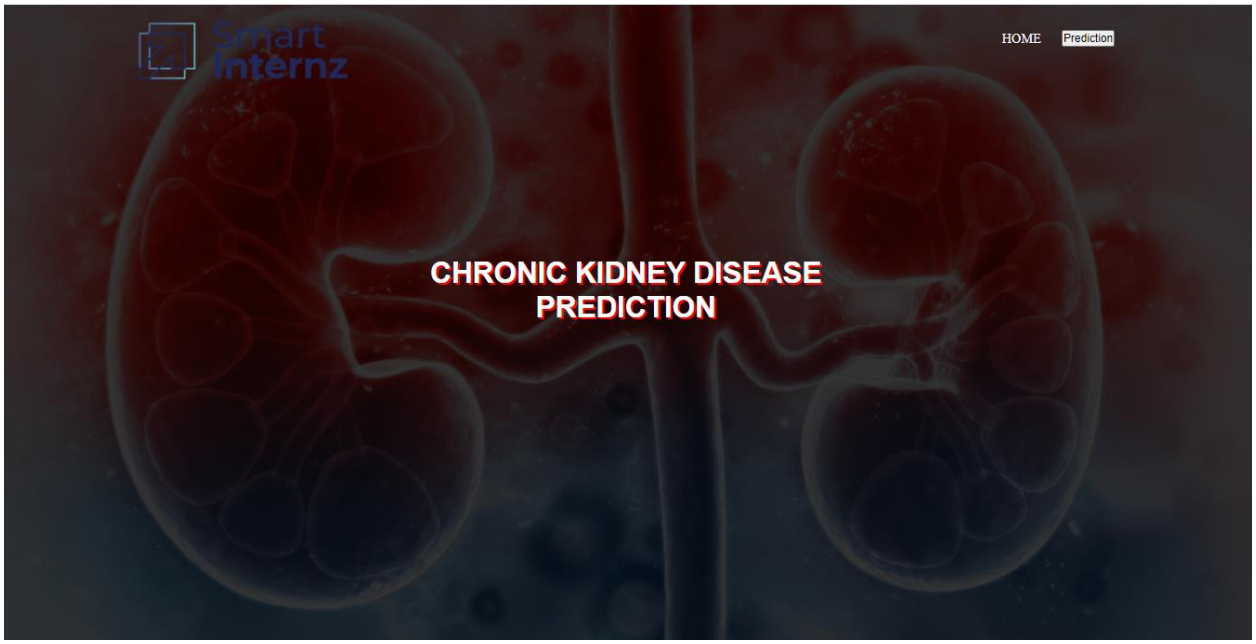
Result 27:



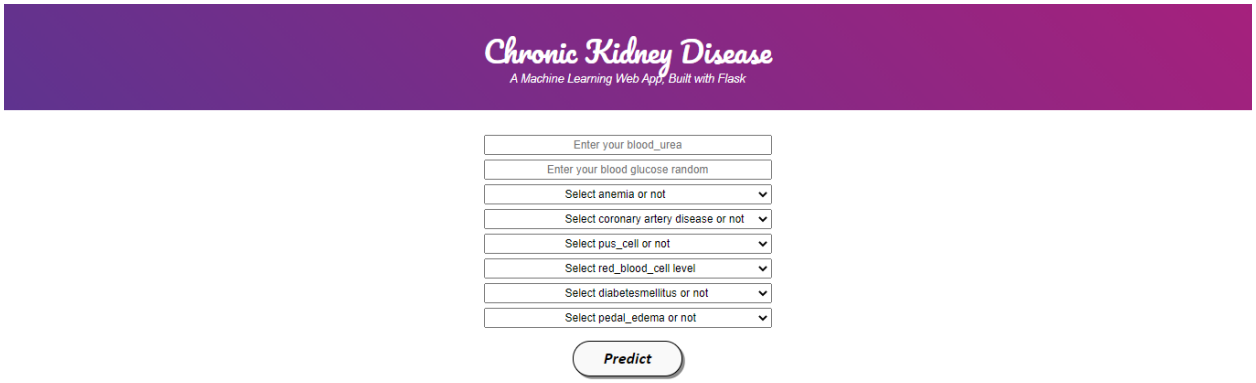
Result 28:



Result 29:



Result 30:



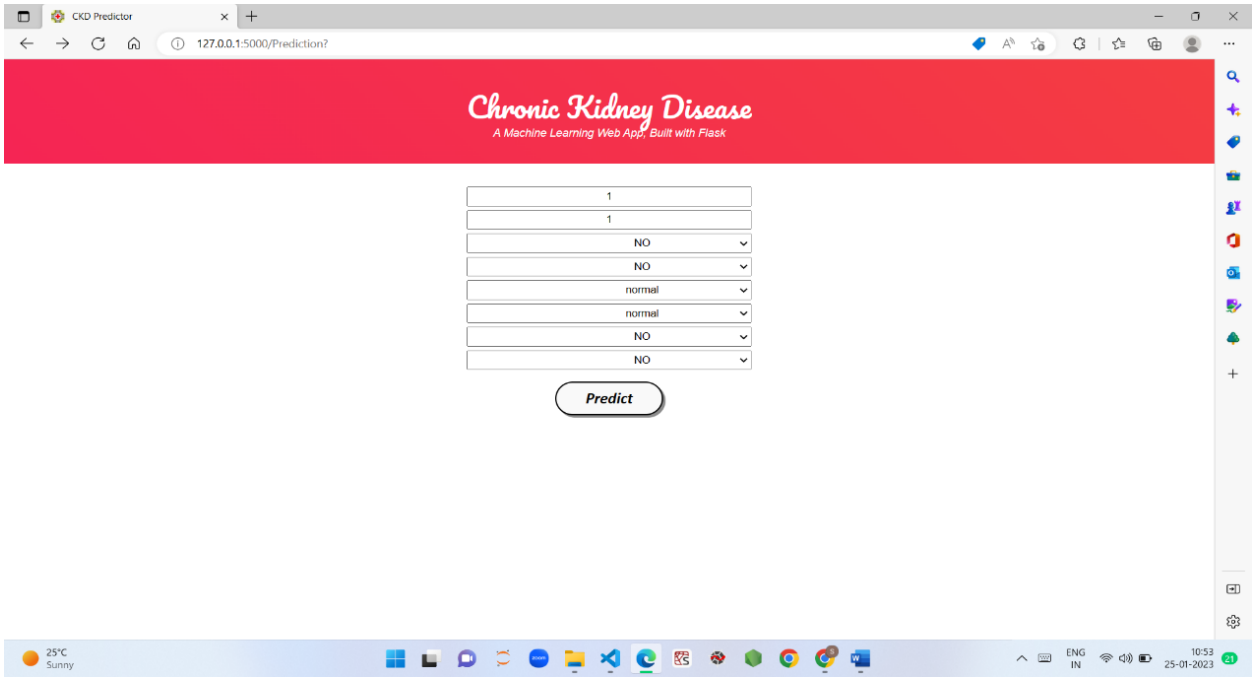
Result 31:

Chronic Kidney Disease

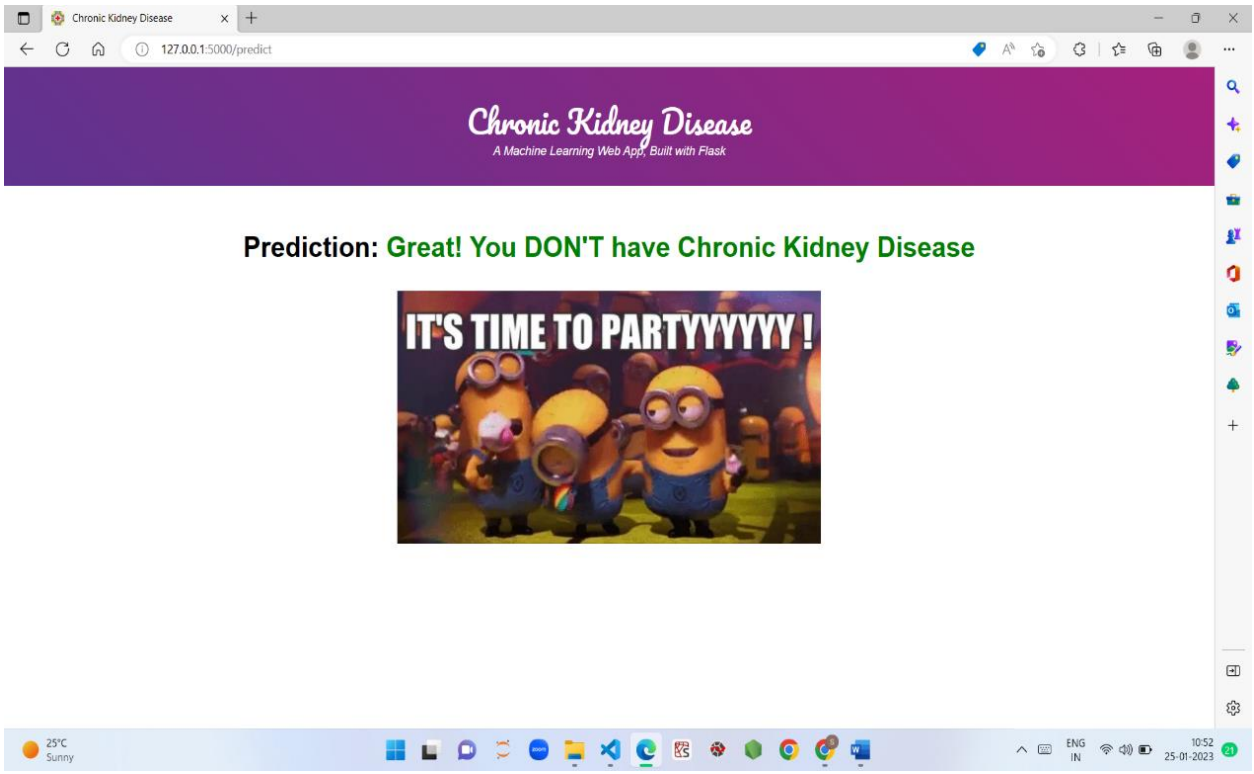
A Machine Learning Web App, Built with Flask

Prediction: Oops! You have Chronic Kidney Disease.

Result 32:



Result 33:


Chronic Kidney Disease
127.0.0.1:5000/predict
Chronic Kidney Disease
A Machine Learning Web App, Built with Flask
Prediction: Great! You DON'T have Chronic Kidney Disease
IT'S TIME TO PARTYYYYYY !
25°C
Sunny
ENG
IN
10:52
25-01-2023

CHAPTER 4
4. ADVANTAGES & DISADVANTAGES

ADVANTAGES:

The advantages of early prediction for chronic kidney disease detection and a progressive approach to health management are numerous.

Firstly, early detection of CKD can allow for interventions to slow or halt its progression, which can prevent the development of serious complications such as kidney failure, cardiovascular disease, and anemia. This can improve patient outcomes and quality of life, and also reduce the burden on the healthcare system by avoiding the need for costly and invasive treatments like dialysis or kidney transplants.

Secondly, a progressive approach to health management can help identify individuals who are at risk of developing CKD and implement strategies to prevent its onset. This can include lifestyle modifications such as exercise, healthy diet, and smoking cessation, as well as regular monitoring and screening for kidney function.

Finally, a progressive approach to health management can also involve the use of technology and data analytics to predict CKD onset and progression in high-risk individuals. This can allow for

targeted interventions and personalized treatment plans, which can improve patient outcomes and reduce healthcare costs.

Overall, the advantages of early prediction for chronic kidney disease detection and a progressive approach to health management are improved patient outcomes, reduced healthcare costs, and a better understanding of CKD as a chronic disease that can be effectively managed through early detection and targeted interventions.

DISADVANTAGES:

There are also some potential disadvantages or challenges associated with early prediction for chronic kidney disease detection and a progressive approach to health management.
These include

1. False positives: Early prediction models may identify individuals as being at risk for CKD who do not actually develop the disease, leading to unnecessary testing and anxiety.
2. Limited access: Access to advanced screening and testing technologies, as well as specialized healthcare professionals who can provide personalized care, may be limited in certain geographic regions or for certain patient populations.
3. Cost: Early prediction and management of CKD may require additional healthcare resources, such as increased testing and monitoring, which can result in higher costs for patients and healthcare systems.

4. Privacy concerns: The use of technology and data analytics to predict and manage CKD may raise concerns around patient privacy and data security.
5. Overdiagnosis: Early prediction and management may lead to overdiagnosis and overtreatment of CKD, potentially exposing patients to unnecessary interventions and risks.
6. Patient adherence: Patients may struggle to adhere to lifestyle modifications or medication regimens necessary for effective CKD management.

Overall, while early prediction for chronic kidney disease detection and a progressive approach to health management can offer many advantages, it is important to be aware of the potential disadvantages and challenges associated with these approaches in order to address them and ensure effective, patient-centered care.

CHAPTER 5

5. APPLICATIONS

There are several applications for early prediction for chronic kidney disease detection and a progressive approach to health management. These include:

Screening and monitoring programs: Healthcare organizations can develop screening and monitoring programs to identify individuals at risk for CKD and provide early interventions to prevent or slow its progression.

Telemedicine: Telemedicine and remote monitoring technologies can be used to improve access to specialized healthcare professionals and allow for real-time monitoring of kidney function.

Personalized treatment plans: Data analytics and predictive models can be used to develop personalized treatment plans for high-risk individuals, improving outcomes and reducing healthcare costs.

Public health campaigns: Public health campaigns can be used to raise awareness of CKD risk factors and encourage healthy lifestyle behaviors that can reduce the risk of developing the disease.

Research: Early prediction and management approaches can be the subject of research studies, leading to new insights into the

pathogenesis of CKD and the development of new interventions.

Overall, the applications for early prediction for chronic kidney disease detection and a progressive approach to health management are numerous, and hold great promise for improving patient outcomes and reducing the burden on the healthcare system. By implementing these approaches, healthcare organizations can identify and manage CKD earlier, leading to better outcomes and improved quality of life for patients.

CHAPTER 6

6.CONCLUSION

In conclusion, early prediction for chronic kidney disease detection and a progressive approach to health management are essential for improving patient outcomes and reducing the burden on the healthcare system. Early detection and interventions can prevent or slow the progression of CKD, avoiding serious complications and costly treatments like dialysis or kidney transplantation. A progressive approach to health management, including lifestyle modifications, regular monitoring, and personalized treatment plans, can identify high-risk individuals and provide targeted interventions to improve outcomes and reduce healthcare costs.

While there are potential challenges associated with early prediction and management of CKD, such as false positives, limited access, and cost, these can be addressed through innovative technologies, public health campaigns, and research studies. By implementing these approaches, healthcare organizations can improve the identification and management of CKD, leading to better outcomes and improved

quality of life for patients. Overall, early prediction for chronic kidney disease detection and a progressive approach to health management represent a promising pathway to better kidney health and improved patient care.

CHAPTER 7

7. FUTURE SCOPE

The future scope of early prediction for chronic kidney disease detection and a progressive approach to health management is vast and exciting. Some potential future developments include:

Advances in technology: Rapid advancements in healthcare technology, such as artificial intelligence, machine learning, and wearable sensors, can help improve early prediction and management of CKD.

Personalized medicine: As more is learned about the genetics and underlying causes of CKD, personalized medicine can be used to tailor treatment plans to the individual patient.

Health equity: Addressing disparities in CKD risk and management among different patient populations can help improve health equity and reduce health disparities.

Integrated care: Integration of primary care and specialty care providers can improve coordination and collaboration, leading to better patient outcomes.

Patient engagement: Improved patient engagement through education and health literacy can empower patients to be active participants in their own care and improve health outcomes.

Prevention: By identifying and addressing risk factors for CKD earlier, prevention strategies can be developed to reduce the incidence of CKD.

Overall, the future scope of early prediction for chronic kidney disease detection and a progressive approach to health management is focused on improving patient outcomes, reducing healthcare costs, and advancing our understanding of CKD. As healthcare organizations and researchers continue to explore new technologies, treatment strategies, and prevention methods, the future of CKD management looks promising.

CHAPTER 8

8. APPENDIX

```
             import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import accuracy_score, confusion_matrix
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.linear_model import LogisticRegression
import pickle
data.columns=['id','age','blood_pressure','specific_gravity','albumin',
'sugar','red_blood_cell','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_diseas','appetite',
'pedal_edema','anemia','class']#manually giving the name of the columns
data.columns
data['blood glucose random'].fillna(data['blood glucose random'].mean(),in
place=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(),inplac
e=True)
```

```
data['potassium'].fillna(data['potassium'].mean(),inplace=True)
data['red_blood_cell_count'].fillna(data['red_blood_cell_count'].mean(),in
place=True)
data['serum_creatinine'].fillna(data['serum_creatinine'].mean(),inplace=Tr
ue)
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=T
rue)
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=T
rue)
data['coronary_artery_diseas'].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['coronary_artery_disease'].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)

catcols.removed('red_blood_cell_count')
catcols.remove('packed_cell_volume')
catcols.remove('white_blood_cell_count')
print(catcols)
import matplotlib.pyplot as plt
fig=plt.figure(figsize=(5,5))
plt.scatter(data['age'],data['blood_pressure'],color='blue')
plt.xlabel('age')
plt.ylabel('blood pressure')
plt.title("ageVSblood scatter plot")
plt.figure(figsize=(20,15), facecolor='white')
plotnumber = 1


for column in contcols:
   if plotnumber<=11  :
```

```
      ax = plt.subplot(3,4,plotnumber)
      plt.scatter(data['age'],data[column])
      plt.xlabel(column,fontsize=20)

    plotnumber+=1
 plt.show()
f,ax=plt.subplots(figsize=(18,10))
sns.heatmap(data.corr(),annot=True,fmt="2f",ax=ax,linewidths=0.5,linecolor
="orange")
plt.xticks(roation=45)
plt.yticks(rotation=45)
plt.show()
selcols=['red_blood_cell','pus_cell','blood glucose random','blood_urea',
         'pedal_edema','anemia','diabetesmelliitus','coronary_artery_disea
se']
x=pd.Datframea(data,columns=selcols)
y=pd.dataframe(data,columns=['class'])
print(x.shape)
print(y.shape)
classification=sequential()
classification.add(Dense(30,activation='relu'))
classification.add(Dense(128,activation='relu'))
classification.add(Dene(64,activation='relu'))
classification.add(Dense(32,activation='relu'))
classification.add(Dense(1,activation='sogmoid'))
from os import name
from sklearn.exceptions import FitFailedWarning
KFold,
      dfs = []
models = [
          ('LogReg',LogisticRegression()),
          ('RF',RandomForestclassifier()),
          ('DecisionTree',DecisionTreeClassifier()),
         ]
results = []
     names = []
     scoring = ['accuracy','precision_weighted','recall_weighted','f1_weig
hted','roc_auc']
     target_name = ['NO CKD','CKD']
     for name,model in models:
           Kfold = model_selection.KFold(n_splits=5,shuffle=true,random_st
ate=90210)
           CV_results = model_selection.cross_validate(model,x_train,y_tra
in,cv=kfold,scoring=scoring)
           clf = model.Fit(x_train,y_train)
```

```
            y_pred = clf.predict(x_test)
            print(name)
            print(classification_report(y_test,y_pred,target_names=target_n
ames))
            results.append(cv_results)
            names.append(name)
            this_df = pd.dataframe(cv_results)
            this_df['model'] = name
            dfs.append(this_df)
     final = pd.concat(dfs,ignore_index=true)
     return final

plt.figure(figsize-(8,6))
sns.heatmap(cm,cmap-'blues',annot-
true,xticklabels=['no ckd','ckd'],yticklabels=['no ckd''ckd'])
plt.xlabel('predicted values')
plt.ylabel('Actual values')
plt.title('confusion matrix for randomforestclassifier')
plt.show()

plt.figure(figsize=(8,6))
sns.heatmap(cm,cmap='Blues',annot=true,xtricklabels=['no ckd','ckd'],ytick
labels=['no ckd','ckd'])
plt.xlabel('predicted values')
plt.ylabel('Actual values')
plt.title('confusion matrix for DecisionTreeClassifier')
plt.show()

from sqlalchemy.sql.expression import true
bootstraps = []
for model in list(set(final.model.values)):
    model_df = final.loc[final.model == model]
    bootstrap = model_df.sample(n=30,replace=true)
    bootstraps.append(bootstrap)
bootstrap_df = pd.concat(bootstraps,igore_index=true)
results_long = pd.melt(bootstrap_df,id_vars=['model'],var_name='metrcs',va
lue_name='values')
time_metrics = ['fit_time','score_time']# fit time matrics
## PERFORMANCE METRICS
results_long_nofit = results_long.loc[~results_long['metrics'].isin(time_m
etrics)]# get df without fit data
results_long_nofit = results_long_nofit.sort_values(by='values')
## TIME METRICS
results_long_fit = results_long.loc[results_long['metrics'].isin(time_metr
ics)] # df with fit data
```

```
results_long_fit = results_long_fit.sort_values(by='values')



import matplotlip.pyplot as plt
import seaborn as sns
plt.figure(figsize=(20,12))
sns.set(font_scale=2.5)
g = sns.boxplot(x="model",y="values",hue="metrics",data=result_long_nofit,
palette="set3")
plt.legend(bbox_to_anchor=(1.05,1),loc=2,borderaxespad=0.)
plt.title('comparison of model by classificartion metric')
plt.savefig('./benchmark_models_performance.png',dpi=300)

@app.route('/prediction',methods=['POST','GET'])
def prediction():
    return render_template('indexnew.html')
@app.route('/Home',methods=['POST','GET'])S
def my_home():
    return render_template('home.html')
@app.route('/predict',methods=['POST']) # route to show the predictions in
 a web UI
def predict():
    #reading the inouts given by the user
    input_features = [float(x) for x in request.form.values()]
    features_values = [np.array(input_features)]
    features_name = ['blood_urea','blood glucose random','anemia',
                     'coronary_artery_disease','pus_cell','red_blood_cells
',
                     'diabetesmellitus','pedal_edema']
    df = pd.Dataframe(features_value,columns=features_name)
    # predictions using the loaded model file
    output = model.predict(df)
```