CHIKKANNA GOVERNMENT ARTS COLLEGE

DEPARTMENT OF BACHELOR OF COMPUTER APPLICATION

TIRUPUR-641602

(AFFILIATED TO BHARATHIAR UNIVERSITY)

TEAM MEMBERS NAME :

KARTHIK RAJA R (2022J0036)

GOWTHAM V (2022J0032)

PUGAZHENTHIE C (2022J0052)

SASIVIKRAM S (2022J0055)

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

1.INTRODUCTION

1.1 OVERVIEW :

Chronic Kidney Disease (CKD) is a major medical problem and can be cured if treated in the early stages. Usually, people are not aware that medical tests we take for different purposes could contain valuable information concerning kidney diseases. Consequently, attributes of various medical tests are investigated to distinguish which attributes may contain helpful information about the disease. The information says that it helps us to measure the severity of the problem, the predicted survival of the patient after the illness, the pattern of the disease and work for curing the disease. In todays world as we know most of the people are facing so many disease and as this can be cured if we treat people in early stages this project can use a pretrained model to predict the Chronic Kidney Disease which can help in treatments of peoples who are suffer from this disease.

In todays world as we know most of the people are facing so many disease and as this can be cured if we treat people in early stages this project can use a pretrained model to predict the Chronic Kidney Disease which can help in treatments of peoples who are suffer from this disease

1.2. PURPOSE :

Chronic kidney disease includes conditions that damage your kidneys and decrease their ability to keep you healthy by filtering wastes from your blood. If kidney disease worsens, wastes can build to high levels in your blood and make you feel sick. You may develop complications like:

- high blood pressure
- anemia (low blood count)
- weak bones
- poor nutritional health
- nerve damage

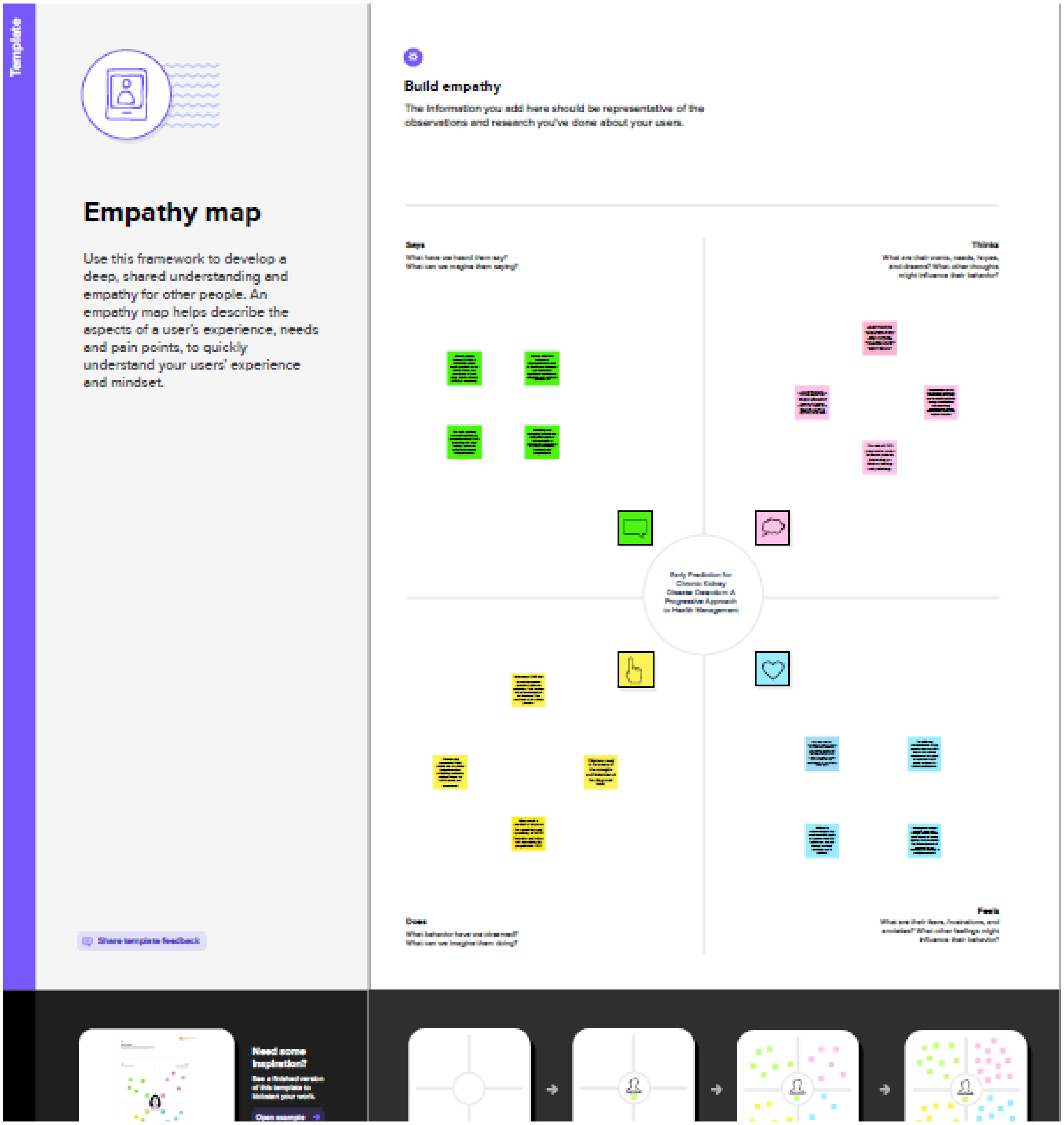
Kidney disease also increases your risk of having heart and blood vessel disease. These problems may happen slowly over a long time. Early detection and treatment can often keep chronic kidney disease from getting worse. When kidney disease progresses, it may eventually lead to kidney failure, which requires dialysis or a kidney transplant to maintain life.

The most common causes of CKD are diabetes and high blood pressure. In the early stages of CKD, there are no symptoms. The disease can progress to complete kidney failure, also called end-stage kidney disease. This occurs when kidney function has worsened to the point that dialysis or kidney transplantation is required to maintain good health and even life, which is typically when kidney function is approximately 10 percent or less of the normal kidney function.

The main goal of treatment is to **prevent** progression of CKD to complete kidney failure. The best way to do this is to diagnose CKD early and control the underlying cause.

2. PROBLEM DEFINITION & DESIGN THINKING

2.1. EMPATHY MAP



Template

Empathy map

Use this framework to develop a deep, shared understanding and empathy for other people. An empathy map helps describe the aspects of a user's experience, needs and pain points, to quickly understand your users' experience and mindset.

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 imagine them saying?

Thinks

What are their wants, needs, hopes, and dreams? What other thoughts might influence their behavior?

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

Does

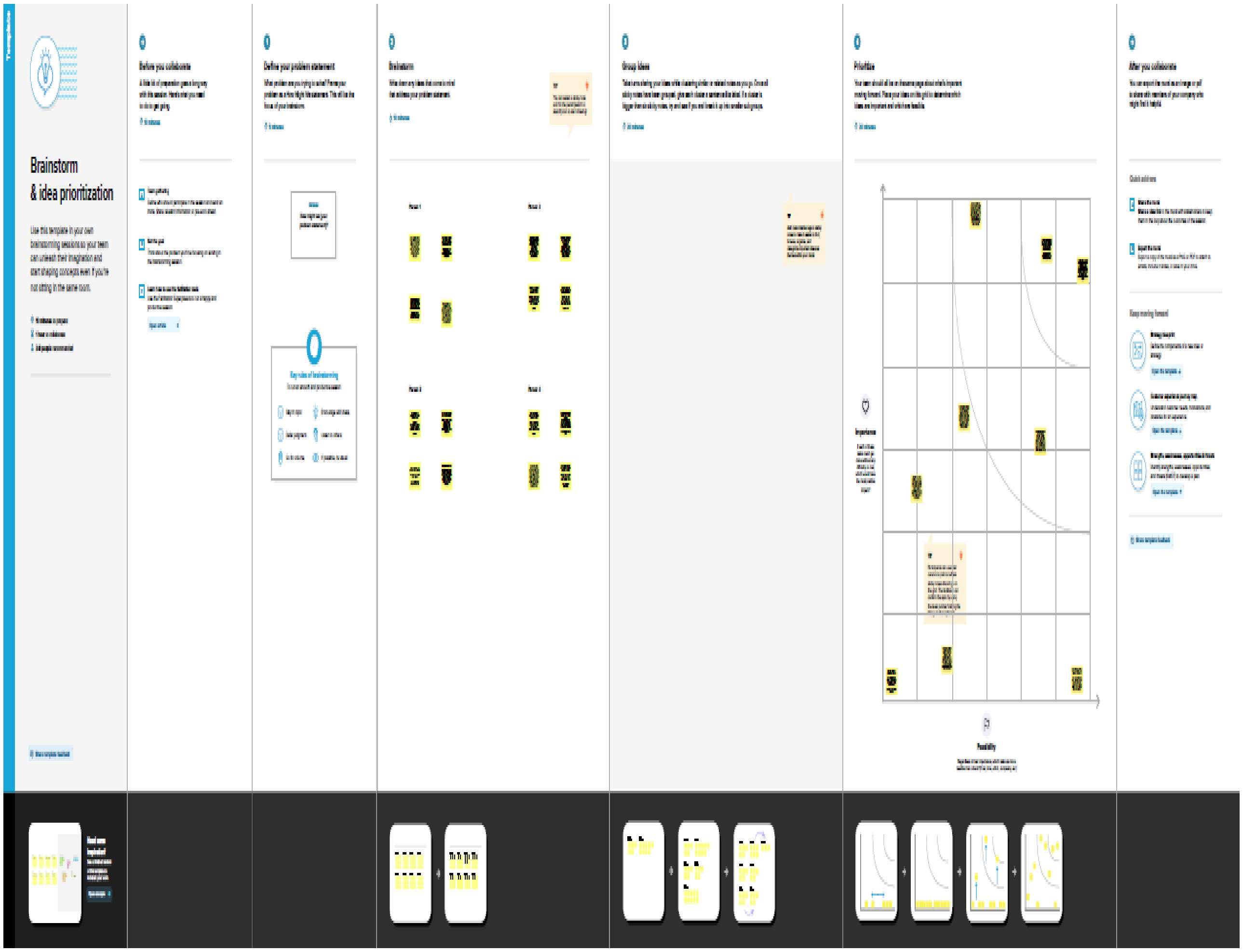
What behavior have we observed?
What can we imagine them doing?

Feels

What are their fears, frustrations, and anxieties? What other feelings might influence their behavior?

Share template feedback
Need some inspiration?
See a finished version of this template to kickstart your work.
Open example

2.2 IDEATION & BRAINSTORMINGS MAP



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.

Before you collaborate

Define your problem statement

Key rules of brainstorming

Brainstorm

Group ideas

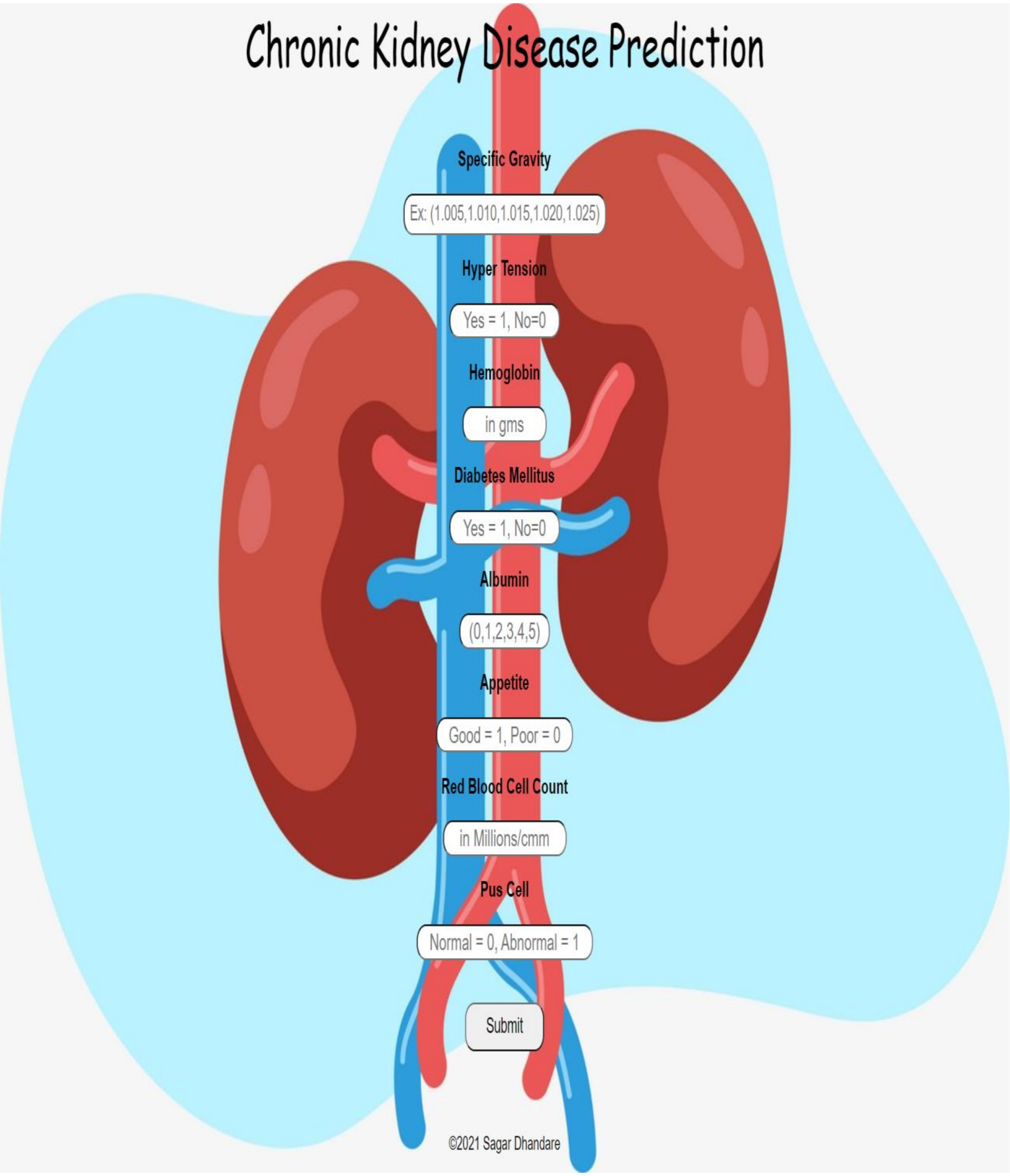
Prioritize

Importance

Feasibility

After you collaborate

3. RESULT



Chronic Kidney Disease Prediction

Specific Gravity

Ex: (1.005,1.010,1.015,1.020,1.025)

Hyper Tension

Yes = 1, No=0

Hemoglobin

in gms

Diabetes Mellitus

Yes = 1, No=0

Albumin

(0,1,2,3,4,5)

Appetite

Good = 1, Poor = 0

Red Blood Cell Count

in Millions/cmm

Pus Cell

Normal = 0, Abnormal = 1

©2021 Sagar Dhandare

4.ADVANTAGES & DISADVANTAGES

ADVANTAGES :

- It is readily available, so the patient doesn't need to wait.
- No chance of rejection.
- No need for major surgery.
- No need to take drugs, such as immuno-suppressants.

DISADVANTAGES :

- Kidney dialysis requires expensive machinery.
- The patient has to do dialysis at least 2-3 times a week for 4-6 hours at a time, so it is very time consuming.
- The patient has to monitor their diet carefully.
- It is painful for the patient
- Can cause infections
- It is more expensive yearly compared to kidney transplants for the NHS.
- It can cause blood clots.

5. APPLICATION

The internet has become one of the most important sources of health information for patients and their families. Recent studies suggest that most adults seek health information online. Many digital educational materials have been available on-line for patients with CKD by professional societies and patient advocacy groups, satisfying the knowledge component of Kolb's learning cycle. Systematic reviews of these educational materials suggest that most are adequate for use as determined by validated instruments, though relatively few are outstanding and many are written at a literacy level too high to be appreciated by most patients with CKD. A well-established repository of educational materials for patients with kidney disease is the National Kidney Disease Education Program (NKDEP), which sponsors an initiative to promote kidney disease education via digital media. The NKDEP website contains several links to kidney disease educational to, including pamphlets available for download. Importantly, the website content is directed at an elementary school level reading capability, and has been modified based on an iterative process of review. This same iterative process was used in the development of the Safe Kidney Care Cohort study webs which provides information to patients, family members and providers, on topics relevant to patient safety in CKD. Health education videos may also be found on these websites as adjuncts to the written educational materials, or they may stand on their own on websites such as YouTube.

6. CONCLUSION

In this paper, we employed a web application which can predict CKD, a comparative analysis was performed between KNN and Naive Bayes algorithm, in which KNN yield a better result. From the result obtained it is observed that overall precision, F-measure, recall is 0.971, 0.985, 1 and accuracy is 97.18 percentage respectively. this shows that KNN yields better result when compared with Naive Bayes. so medical practitioner can use KNN for prediction of CKD stage prediction is done by using GFR, depending on stage diet is recommended by the doctor and they can also upload patient's treatment details. patients can access uploaded treatment and recommended diet by the doctor. The proposed model uses a single algorithm to predict CKD, accuracy of the present work can be enhanced by using hybrid machine learning algorithms, which gives better accuracy compared with KNN algorithm. Along with this additional feature could be added to enhance our web application in future.

Telehealth is largely nascent in the field of nephrology but early examples illustrate great promise to increase general awareness and understanding of kidney disease among patients and to enhance renal knowledge and optimal CKD management among primary care providers. With suboptimal levels of CKD awareness among both of these important stakeholder groups, the use of telehealth applications and other health information technology tools for education rightfully engender great excitement. Care will be needed to ensure that these tools are widely accessible, designed for individuals with all levels of e-literacy. Further, rigorous evaluation will be critical to determine benefits relative to traditional educational modalities, and to identify and mitigate unanticipated consequences. Nephrology education is gearing up for the future—fasten your seatbelts!

7. FUTURE SCOPE

The increasing prevalence of chronic kidney disease is well known, as it is a fact that recorded data in all countries show continuing growth in the number of patients that need substitutive treatment for their renal function. The consequences from the social and economic viewpoint are very significant and we cannot be happy with morbidity and mortality rates in terminal stage renal patients that continue to be unacceptably high.[1,2] There are different reasons for such high mortality rates, amongst them significant increase in the age of patients undergoing treatment, restoration with haemodialysis and peritoneal dialysis of only 15 to 20ml/min of kidney function, and a significant associated co-morbidity. Despite the progress made in haemodialysis (membrane biocompatibility, high-flow membranes, increase frequency in sessions, water quality control, among others) and in peritoneal dialysis (infection risk reduction, introduction of a dialysis machine, etc.) no clear improvement has been shown in the evolution of patients.

Therefore, if so little improvement has been made after so many years, what is in store for the future for renal function replacement? This article aims at highlighting which are the future possibilities to face renal insufficiency, by substitutive techniques such as haemodialysis, peritoneal dialysis or kidney transplant (or creation of new organs), as well as the possibility of regression of chronic kidney disease before total loss of renal function.

As it has been expressed, the situation of patients undergoing haemodialysis means a great sacrifice, overall, both for the patients themselves and their families, especially because of their bad quality of life and the need to move over to the dialysis centre three or more times per week. Furthermore, a high mortality rate (similar to metastatic breast cancer, colon or prostate cancer) forces to move on toward applying different techniques.

The fact that there is evidence of improvement with frequent and prolonged dialysis in quality of life and anaemia control, hypertension control, hospitalisations, medication reduction (i.e. anti-hypertensive or phosphate binders), appetite improvement, volume control improvement, morbidity and mortality reduction, etc., it all leads research toward types of techniques with continous treatment.

It is true that continuous ambulatory peritoneal technique could somehow come closer, as in fact it does, to continous treatment. It has been used for many years in many centres. However, the percentage of patients does not extend beyond 10-15% of those undergoing dialysis and, besides, there is a significant decline as time passes due to loss of ultrafiltration capacity or peritoneum diffusion, which is insufficient in many cases when the residual renal function disappears.

The requirements for new technologies in dialysis are, therefore, based on the following objectives:

1.Continuous function

2. Elimination of molecular weight solutes similar to kidney function.

3. Elimination of water and solutes according to patient’s needs.

4. Biocompatibility.

5. Portable, or even better, implantable.

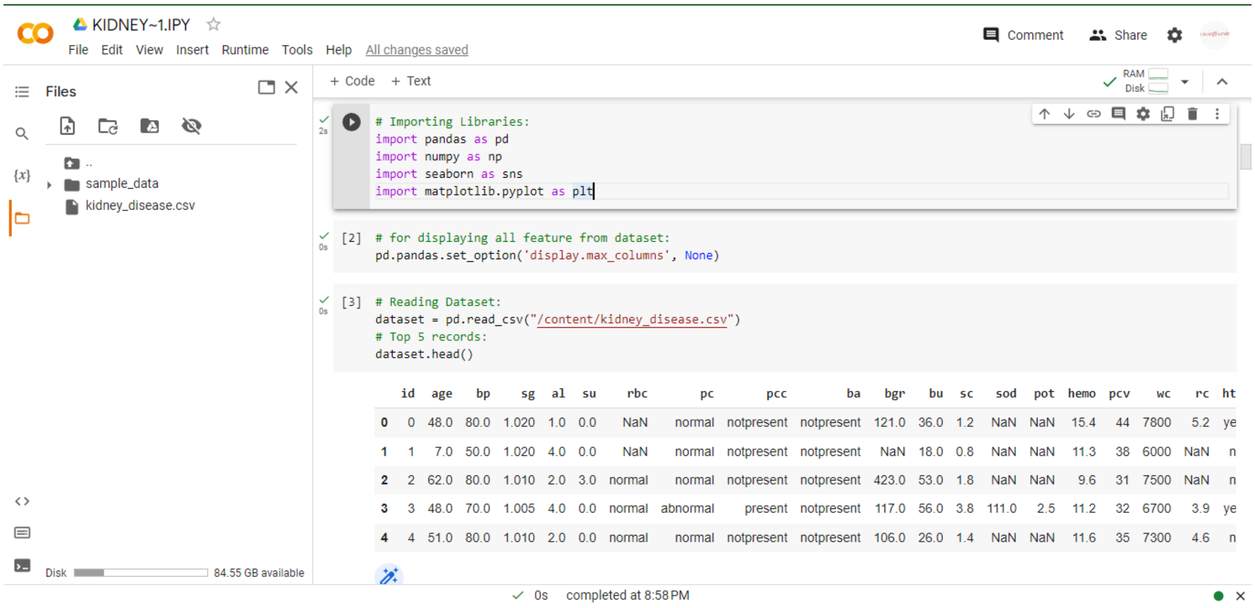
6. Low cost.

7. Safety

There are currently four possible models that could reach these objectives in the future: HNF (Human Nephron Filter), micro-fluid techniques, WAK (Wearable Artificial Kidney) and RAD (Bioartificial Renal Assist Device).

8. APPENDIX

A.SOURCE CODE :



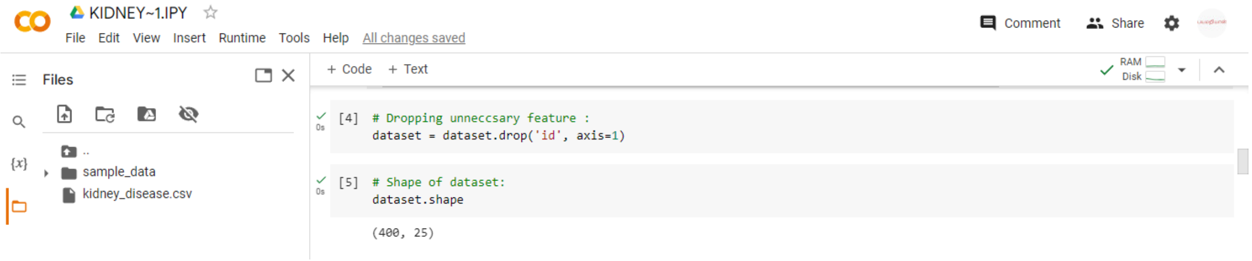
```
# Importing Libraries:
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

```
[2] # for displaying all feature from dataset:
    pd.pandas.set_option('display.max_columns', None)
```

```
[3] # Reading Dataset:
    dataset = pd.read_csv("/content/kidney_disease.csv")
    # Top 5 records:
    dataset.head()
```

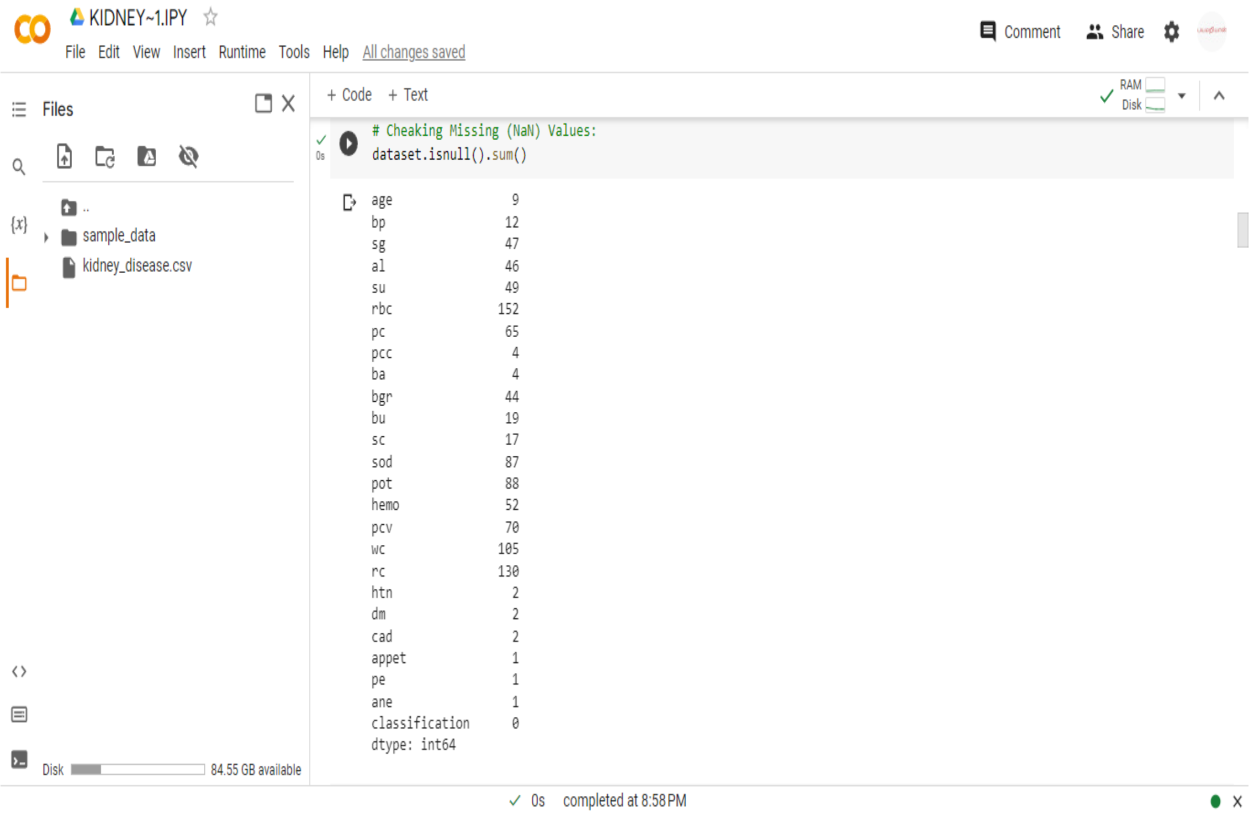
```
   id  age    bp     sg   al   su     rbc        pc         pcc          ba    bgr    bu   sc    sod  pot  hemo  pcv    wc   rc  ht
0   0  48.0  80.0  1.020  1.0  0.0     NaN    normal  notpresent  notpresent  121.0  36.0  1.2    NaN  NaN  15.4   44  7800  5.2  ye
1   1   7.0  50.0  1.020  4.0  0.0     NaN    normal  notpresent  notpresent    NaN  18.0  0.8    NaN  NaN  11.3   38  6000  NaN   n
2   2  62.0  80.0  1.010  2.0  3.0  normal    normal  notpresent  notpresent  423.0  53.0  1.8    NaN  NaN   9.6   31  7500  NaN   n
3   3  48.0  70.0  1.005  4.0  0.0  normal  abnormal     present  notpresent  117.0  56.0  3.8  111.0  2.5  11.2   32  6700  3.9  ye
4   4  51.0  80.0  1.010  2.0  0.0  normal    normal  notpresent  notpresent  106.0  26.0  1.4    NaN  NaN  11.6   35  7300  4.6   n
```

✓ 0s completed at 8:58 PM



```
[4] # Dropping unneccsary feature :
    dataset = dataset.drop('id', axis=1)
```

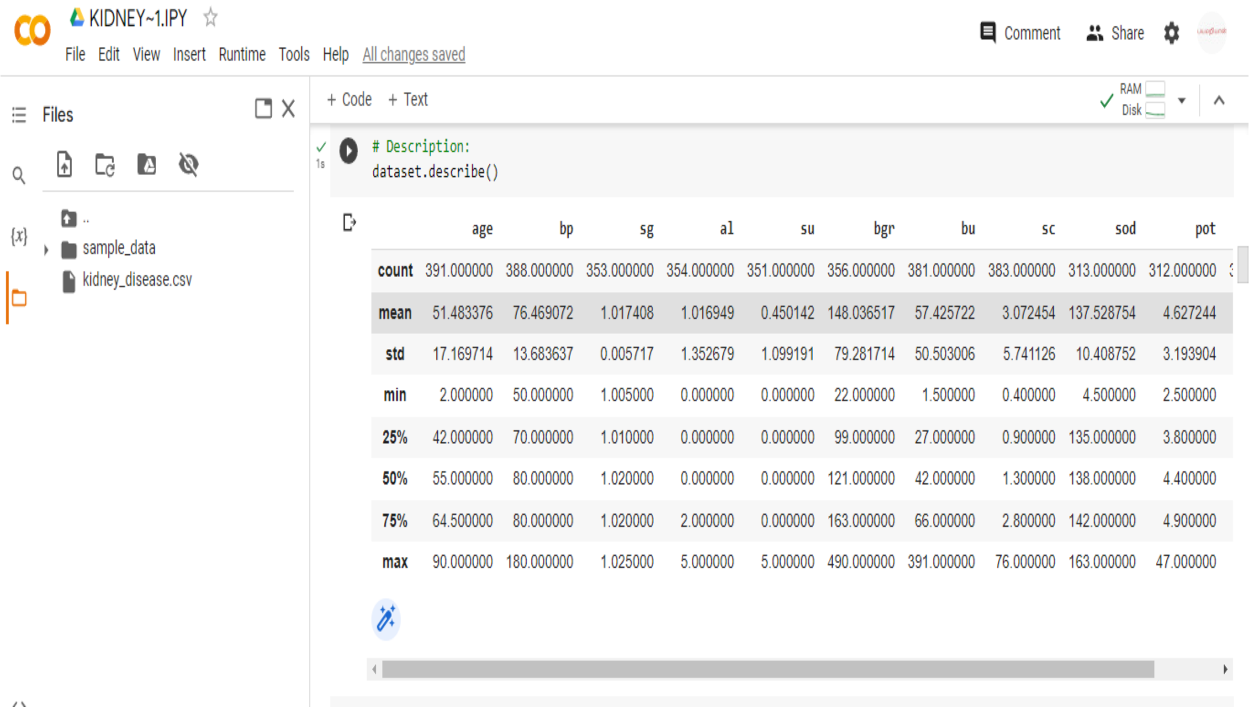
```
[5] # Shape of dataset:
    dataset.shape

    (400, 25)
```



```
# Cheaking Missing (NaN) Values:
dataset.isnull().sum()
```

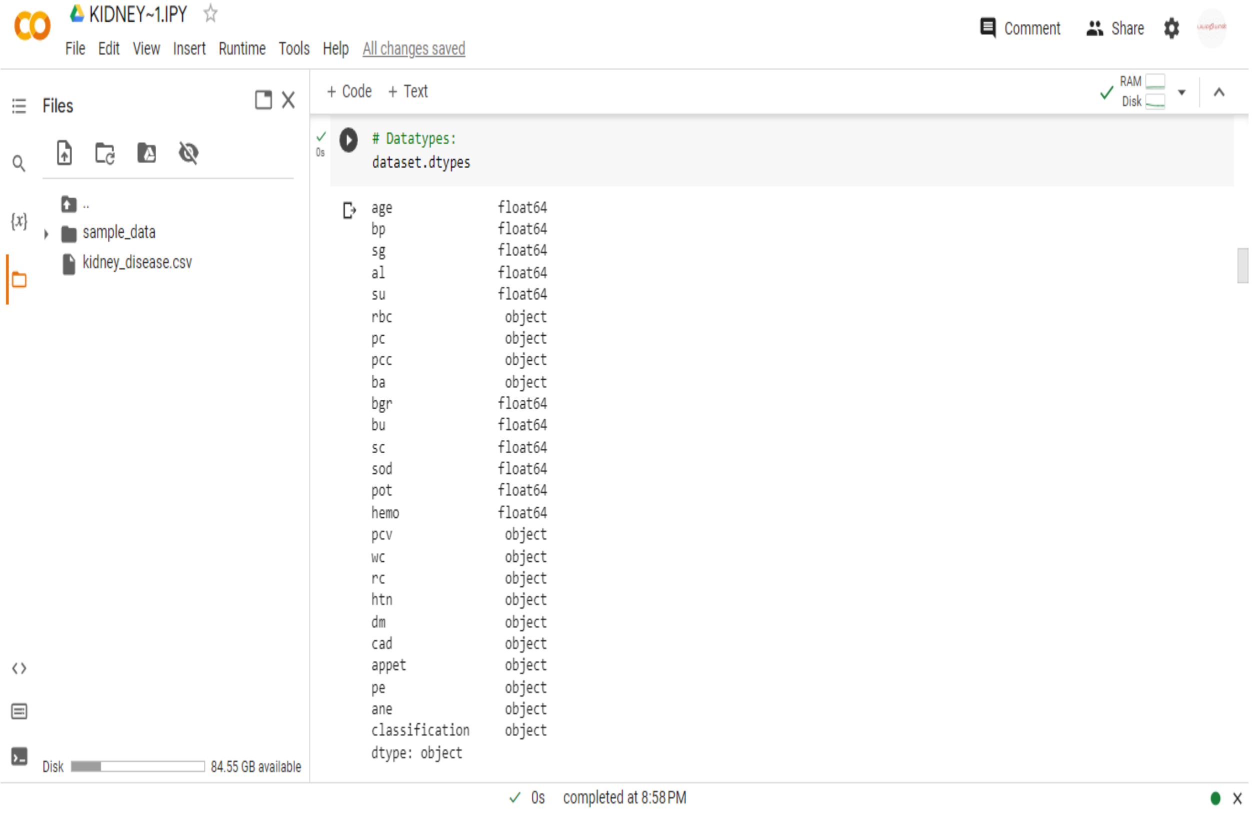
```
age                 9
bp                 12
sg                 47
al                 46
su                 49
rbc               152
pc                 65
pcc                 4
ba                  4
bgr                44
bu                 19
sc                 17
sod                87
pot                88
hemo               52
pcv                70
wc                105
rc                130
htn                 2
dm                  2
cad                 2
appet               1
pe                  1
ane                 1
classification      0
dtype: int64
```

✓ 0s completed at 8:58 PM



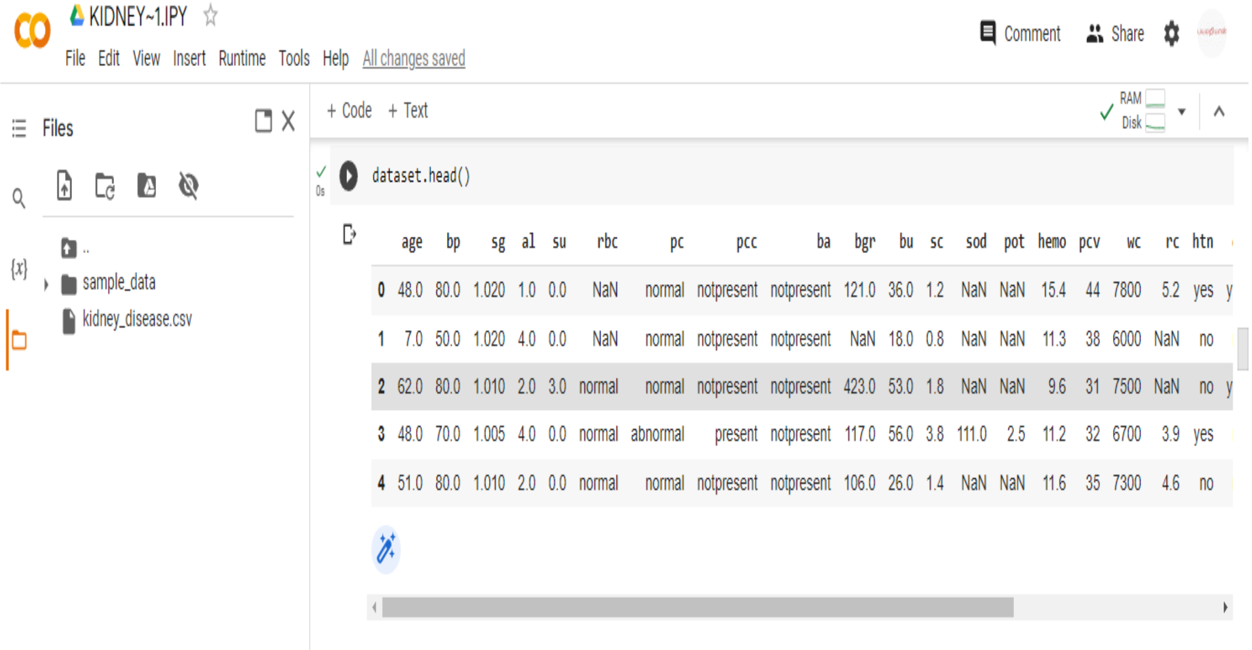
```
# Description:
dataset.describe()
```

	age	bp	sg	al	su	bgr	bu	sc	sod	pot
count	391.000000	388.000000	353.000000	354.000000	351.000000	356.000000	381.000000	383.000000	313.000000	312.000000
mean	51.483376	76.469072	1.017408	1.016949	0.450142	148.036517	57.425722	3.072454	137.528754	4.627244
std	17.169714	13.683637	0.005717	1.352679	1.099191	79.281714	50.503006	5.741126	10.408752	3.193904
min	2.000000	50.000000	1.005000	0.000000	0.000000	22.000000	1.500000	0.400000	4.500000	2.500000
25%	42.000000	70.000000	1.010000	0.000000	0.000000	99.000000	27.000000	0.900000	135.000000	3.800000
50%	55.000000	80.000000	1.020000	0.000000	0.000000	121.000000	42.000000	1.300000	138.000000	4.400000
75%	64.500000	80.000000	1.020000	2.000000	0.000000	163.000000	66.000000	2.800000	142.000000	4.900000
max	90.000000	180.000000	1.025000	5.000000	5.000000	490.000000	391.000000	76.000000	163.000000	47.000000



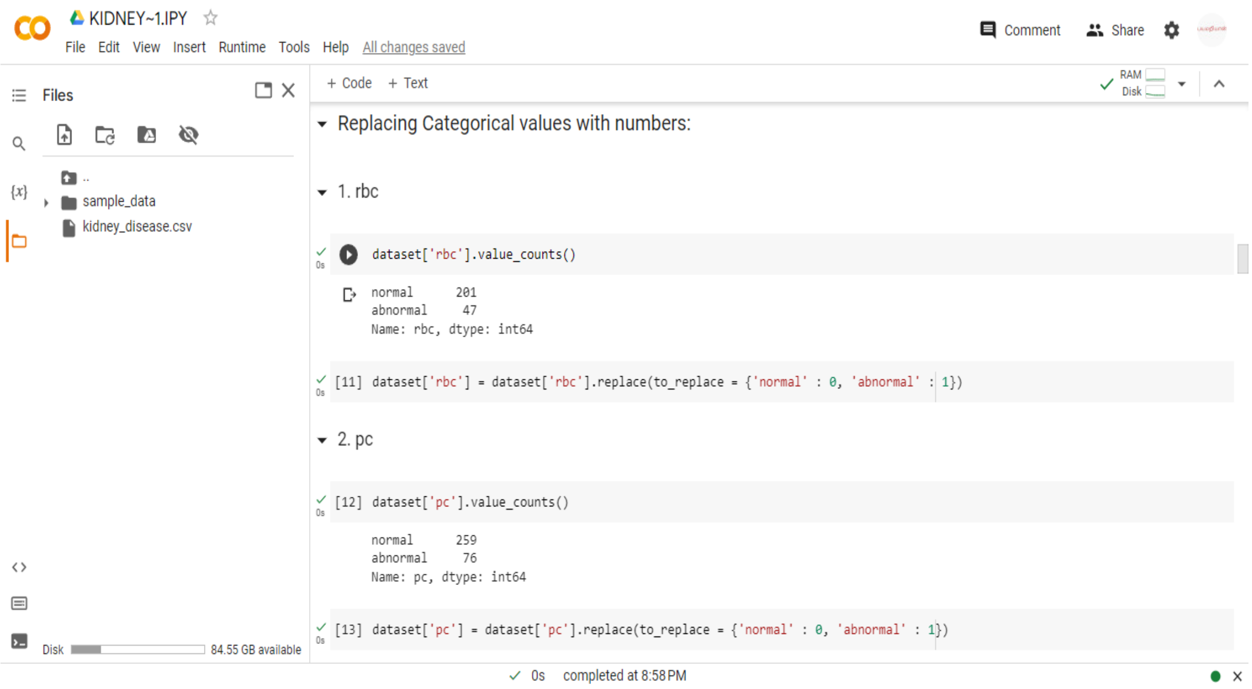
```
# Datatypes:
dataset.dtypes
```

```
age               float64
bp                float64
sg                float64
al                float64
su                float64
rbc                object
pc                 object
pcc                object
ba                 object
bgr               float64
bu                float64
sc                float64
sod               float64
pot               float64
hemo              float64
pcv                object
wc                 object
rc                 object
htn                object
dm                 object
cad                object
appet              object
pe                 object
ane                object
classification     object
dtype: object
```



```
dataset.head()
```

	age	bp	sg	al	su	rbc	pc	pcc	ba	bgr	bu	sc	sod	pot	hemo	pcv	wc	rc	htn	
0	48.0	80.0	1.020	1.0	0.0	NaN	normal	notpresent	notpresent	121.0	36.0	1.2	NaN	NaN	15.4	44	7800	5.2	yes	y
1	7.0	50.0	1.020	4.0	0.0	NaN	normal	notpresent	notpresent	NaN	18.0	0.8	NaN	NaN	11.3	38	6000	NaN	no	
2	62.0	80.0	1.010	2.0	3.0	normal	normal	notpresent	notpresent	423.0	53.0	1.8	NaN	NaN	9.6	31	7500	NaN	no	y
3	48.0	70.0	1.005	4.0	0.0	normal	abnormal	present	notpresent	117.0	56.0	3.8	111.0	2.5	11.2	32	6700	3.9	yes	
4	51.0	80.0	1.010	2.0	0.0	normal	normal	notpresent	notpresent	106.0	26.0	1.4	NaN	NaN	11.6	35	7300	4.6	no	



Replacing Categorical values with numbers:

1. rbc

```
dataset['rbc'].value_counts()
```

```
normal      201
abnormal     47
Name: rbc, dtype: int64
```

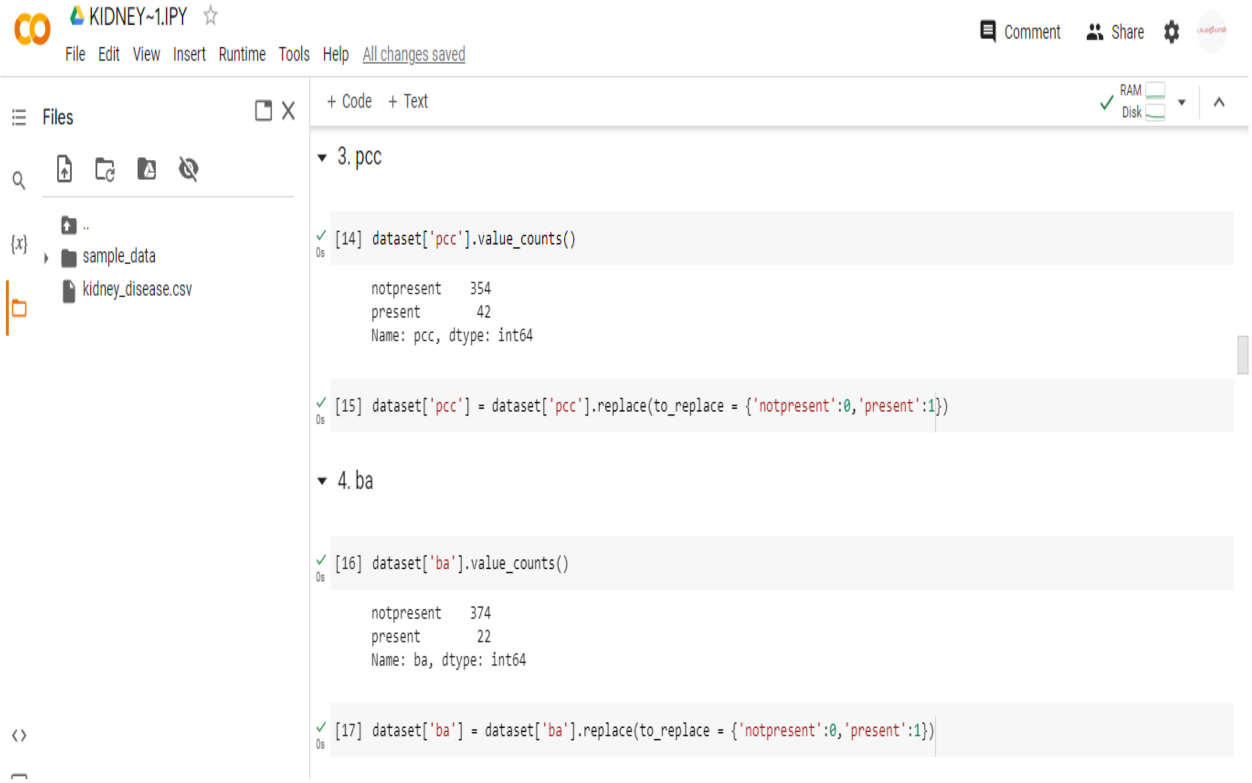
```
[11] dataset['rbc'] = dataset['rbc'].replace(to_replace = {'normal' : 0, 'abnormal' : 1})
```

2. pc

```
[12] dataset['pc'].value_counts()
```

```
normal      259
abnormal     76
Name: pc, dtype: int64
```

```
[13] dataset['pc'] = dataset['pc'].replace(to_replace = {'normal' : 0, 'abnormal' : 1})
```



3. pcc

```
[14] dataset['pcc'].value_counts()
```

```
notpresent    354
present        42
Name: pcc, dtype: int64
```

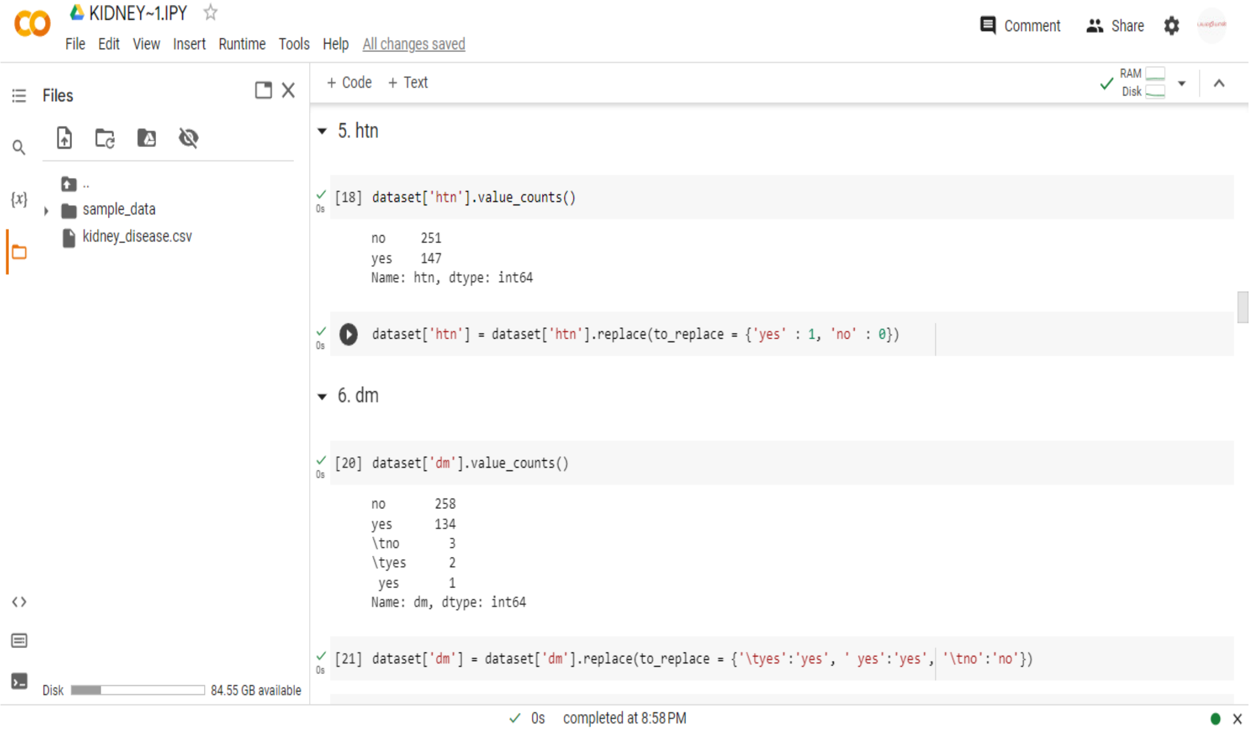
```
[15] dataset['pcc'] = dataset['pcc'].replace(to_replace = {'notpresent':0,'present':1})
```

4. ba

```
[16] dataset['ba'].value_counts()
```

```
notpresent    374
present        22
Name: ba, dtype: int64
```

```
[17] dataset['ba'] = dataset['ba'].replace(to_replace = {'notpresent':0,'present':1})
```



5. htn

```
[18] dataset['htn'].value_counts()
```

```
no     251
yes    147
Name: htn, dtype: int64
```

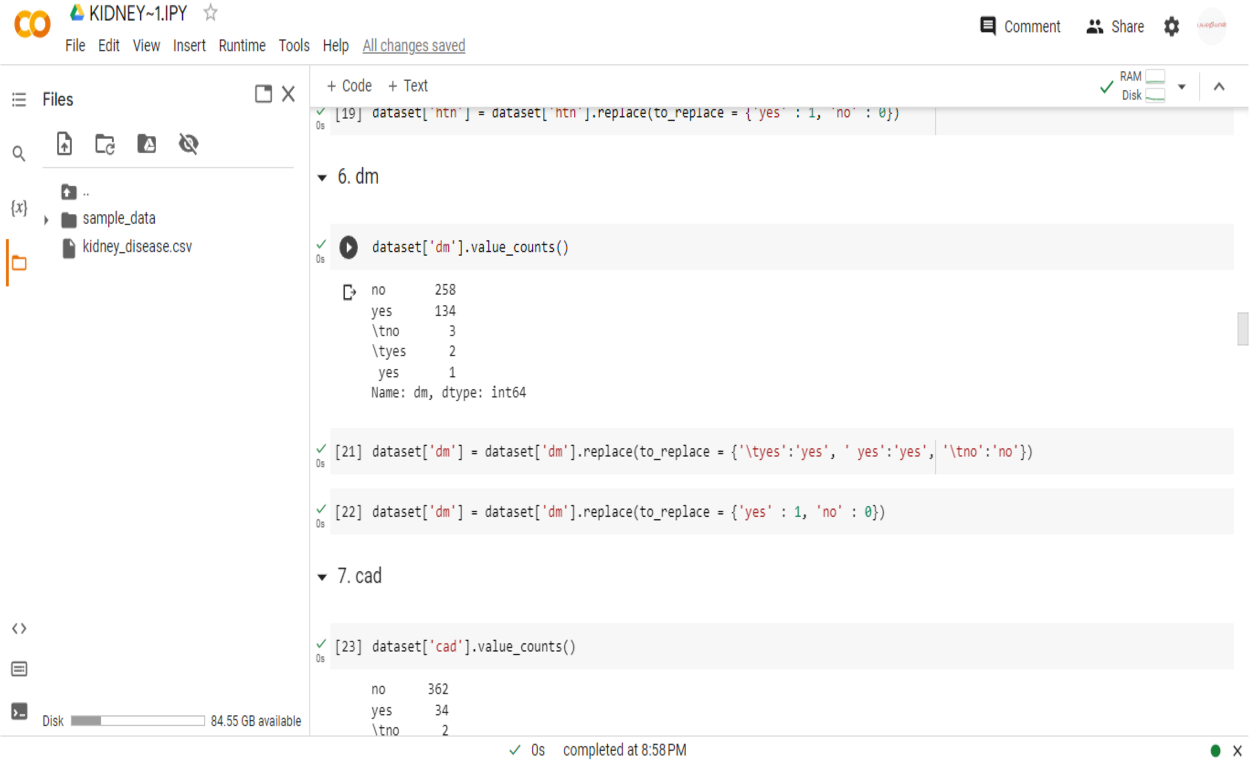
```
dataset['htn'] = dataset['htn'].replace(to_replace = {'yes' : 1, 'no' : 0})
```

6. dm

```
[20] dataset['dm'].value_counts()
```

```
no       258
yes      134
\tno       3
\tyes      2
 yes       1
Name: dm, dtype: int64
```

```
[21] dataset['dm'] = dataset['dm'].replace(to_replace = {'\tyes':'yes', ' yes':'yes', '\tno':'no'})
```



```
[19] dataset['htn'] = dataset['htn'].replace(to_replace = {'yes' : 1, 'no' : 0})
```

6. dm

```
dataset['dm'].value_counts()
```

```
no       258
yes      134
\tno       3
\tyes      2
 yes       1
Name: dm, dtype: int64
```

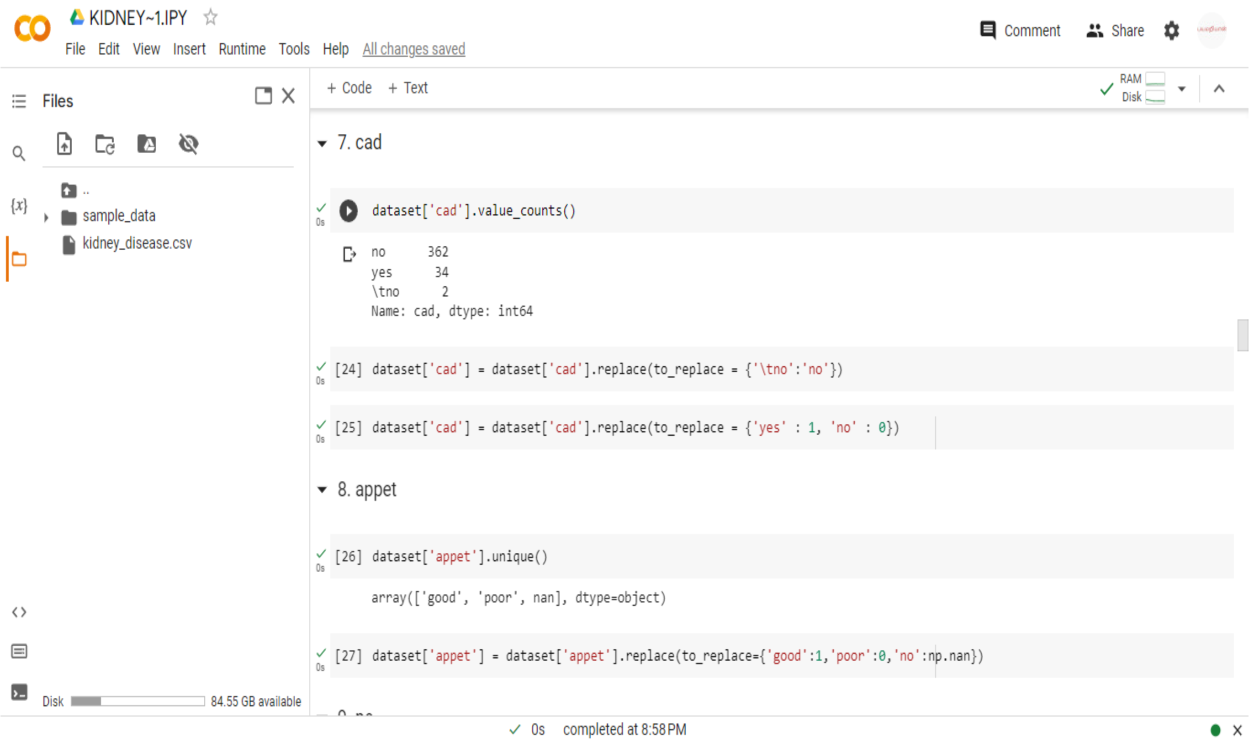
```
[21] dataset['dm'] = dataset['dm'].replace(to_replace = {'\tyes':'yes', ' yes':'yes', '\tno':'no'})
```

```
[22] dataset['dm'] = dataset['dm'].replace(to_replace = {'yes' : 1, 'no' : 0})
```

7. cad

```
[23] dataset['cad'].value_counts()
```

```
no      362
yes      34
\tno      2
```



7. cad

```
dataset['cad'].value_counts()
```

```
no      362
yes      34
\tno      2
Name: cad, dtype: int64
```

```
[24] dataset['cad'] = dataset['cad'].replace(to_replace = {'\tno':'no'})
```

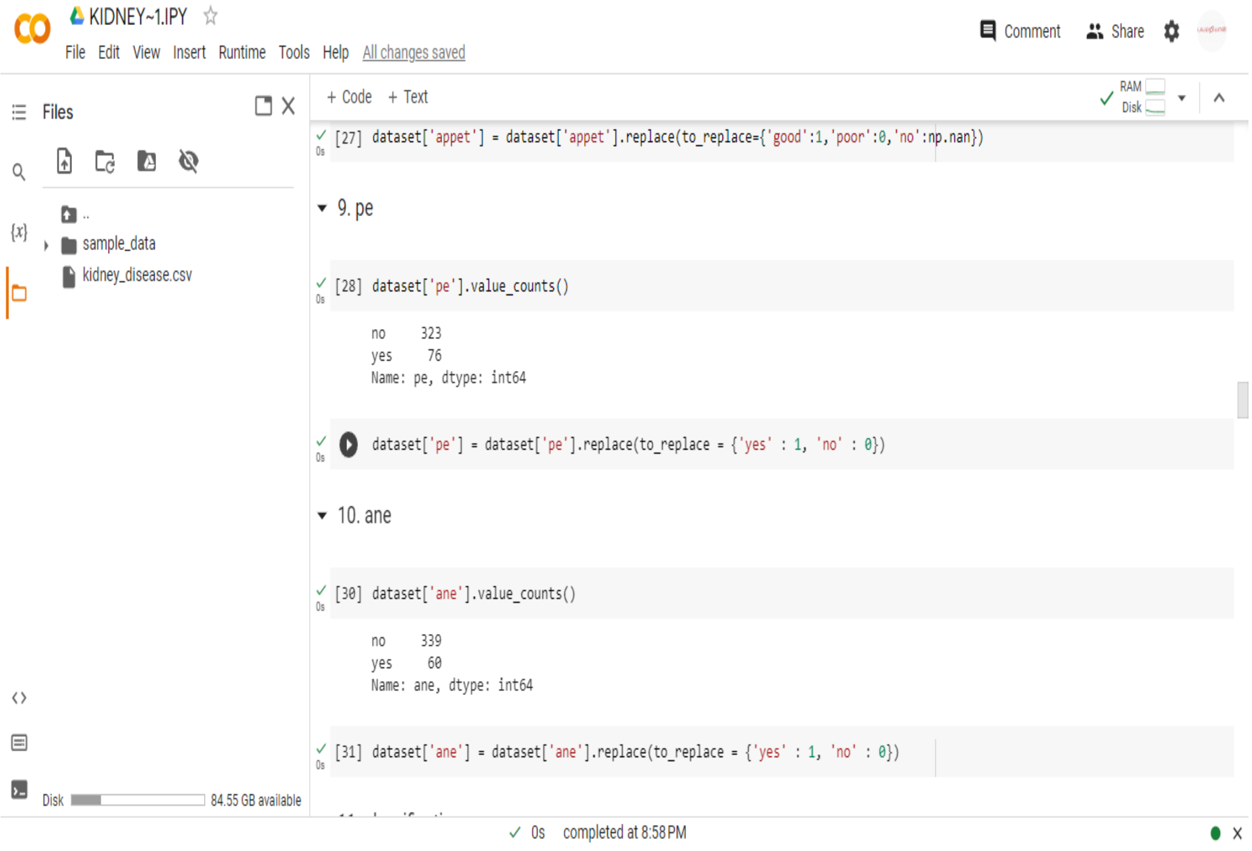
```
[25] dataset['cad'] = dataset['cad'].replace(to_replace = {'yes' : 1, 'no' : 0})
```

8. appet

```
[26] dataset['appet'].unique()
```

```
array(['good', 'poor', nan], dtype=object)
```

```
[27] dataset['appet'] = dataset['appet'].replace(to_replace={'good':1,'poor':0,'no':np.nan})
```



```
[27] dataset['appet'] = dataset['appet'].replace(to_replace={'good':1,'poor':0,'no':np.nan})
```

9. pe

```
[28] dataset['pe'].value_counts()
```

```
no     323
yes     76
Name: pe, dtype: int64
```

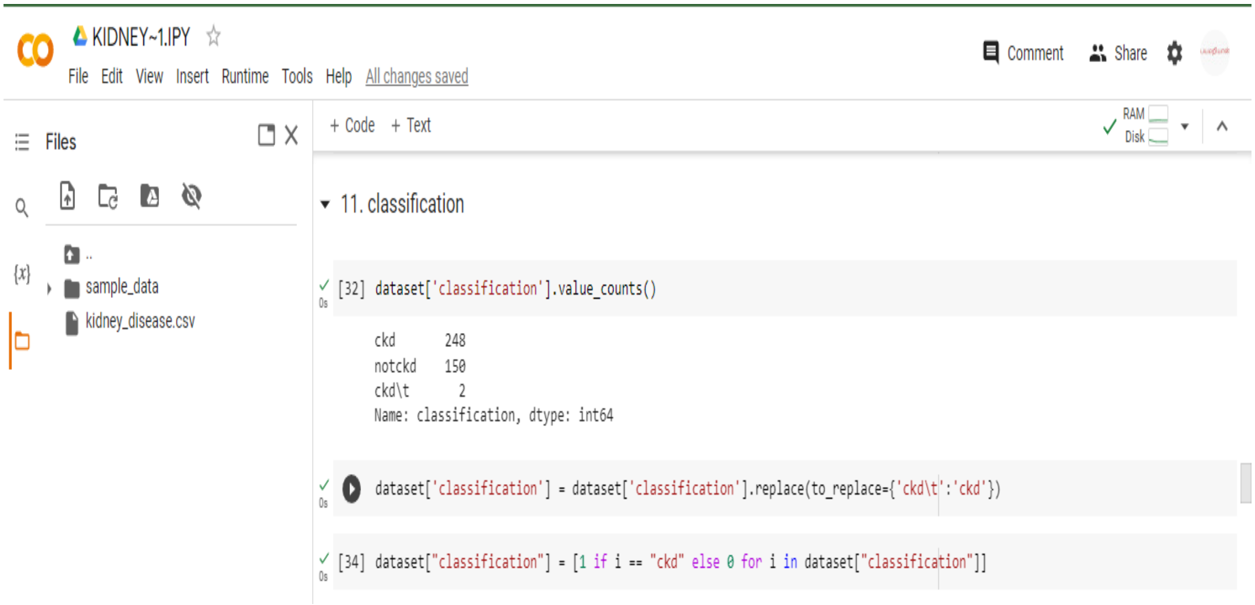
```
dataset['pe'] = dataset['pe'].replace(to_replace = {'yes' : 1, 'no' : 0})
```

10. ane

```
[30] dataset['ane'].value_counts()
```

```
no     339
yes     60
Name: ane, dtype: int64
```

```
[31] dataset['ane'] = dataset['ane'].replace(to_replace = {'yes' : 1, 'no' : 0})
```



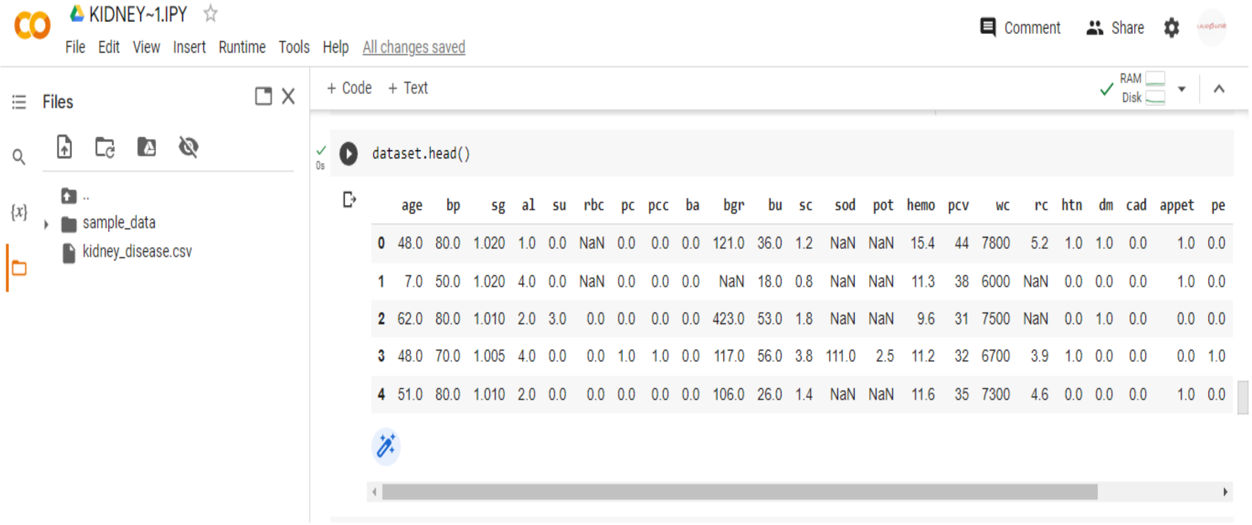
11. classification

```
[32] dataset['classification'].value_counts()
```

```
ckd       248
notckd    150
ckd\t       2
Name: classification, dtype: int64
```

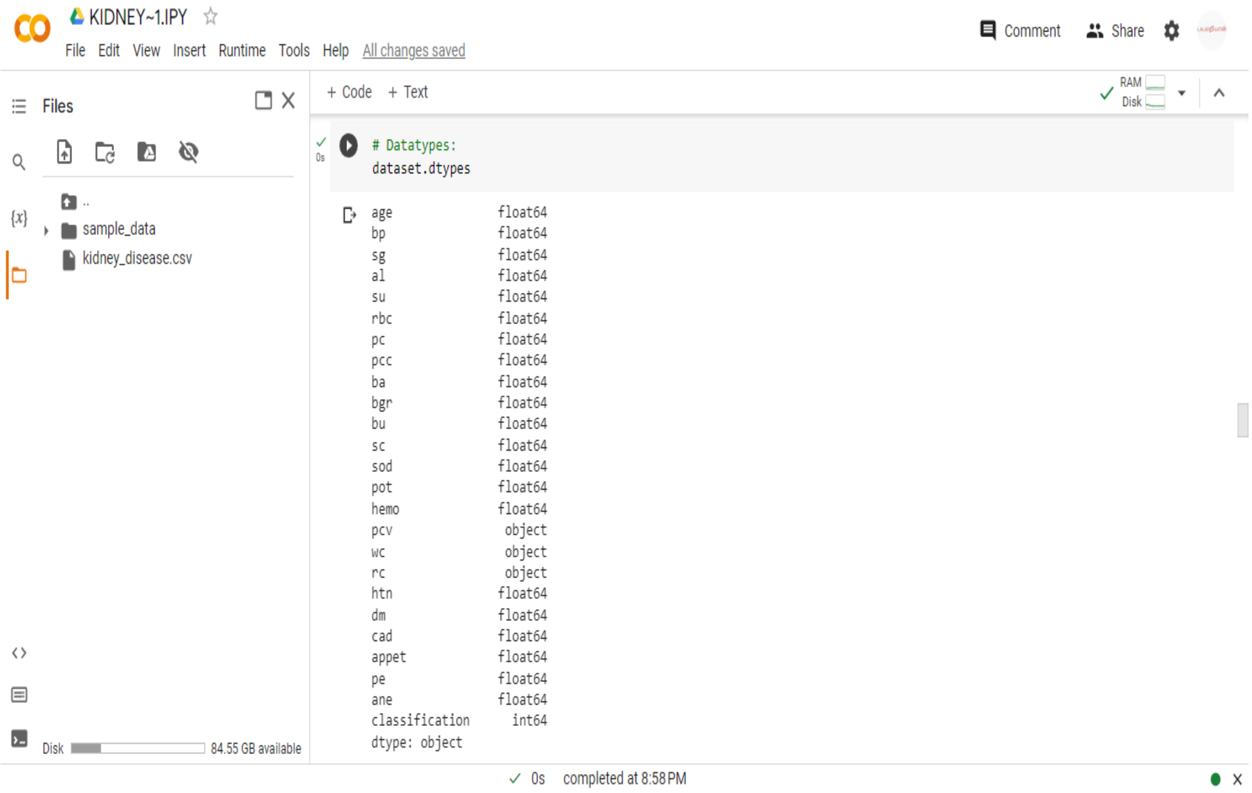
```
dataset['classification'] = dataset['classification'].replace(to_replace={'ckd\t':'ckd'})
```

```
[34] dataset["classification"] = [1 if i == "ckd" else 0 for i in dataset["classification"]]
```



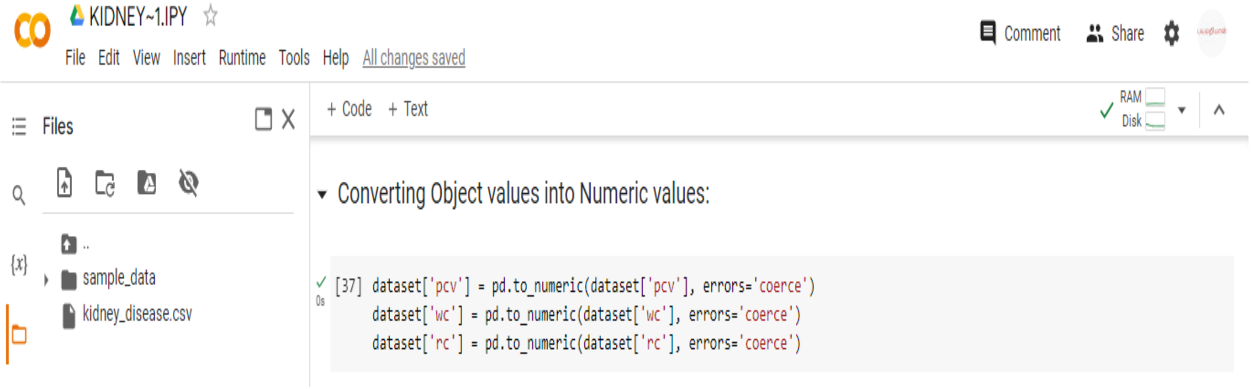
```
dataset.head()
```

	age	bp	sg	al	su	rbc	pc	pcc	ba	bgr	bu	sc	sod	pot	hemo	pcv	wc	rc	htn	dm	cad	appet	pe
0	48.0	80.0	1.020	1.0	0.0	NaN	0.0	0.0	0.0	121.0	36.0	1.2	NaN	NaN	15.4	44	7800	5.2	1.0	1.0	0.0	1.0	0.0
1	7.0	50.0	1.020	4.0	0.0	NaN	0.0	0.0	0.0	NaN	18.0	0.8	NaN	NaN	11.3	38	6000	NaN	0.0	0.0	0.0	1.0	0.0
2	62.0	80.0	1.010	2.0	3.0	0.0	0.0	0.0	0.0	423.0	53.0	1.8	NaN	NaN	9.6	31	7500	NaN	0.0	1.0	0.0	0.0	0.0
3	48.0	70.0	1.005	4.0	0.0	0.0	1.0	1.0	0.0	117.0	56.0	3.8	111.0	2.5	11.2	32	6700	3.9	1.0	0.0	0.0	0.0	1.0
4	51.0	80.0	1.010	2.0	0.0	0.0	0.0	0.0	0.0	106.0	26.0	1.4	NaN	NaN	11.6	35	7300	4.6	0.0	0.0	0.0	1.0	0.0



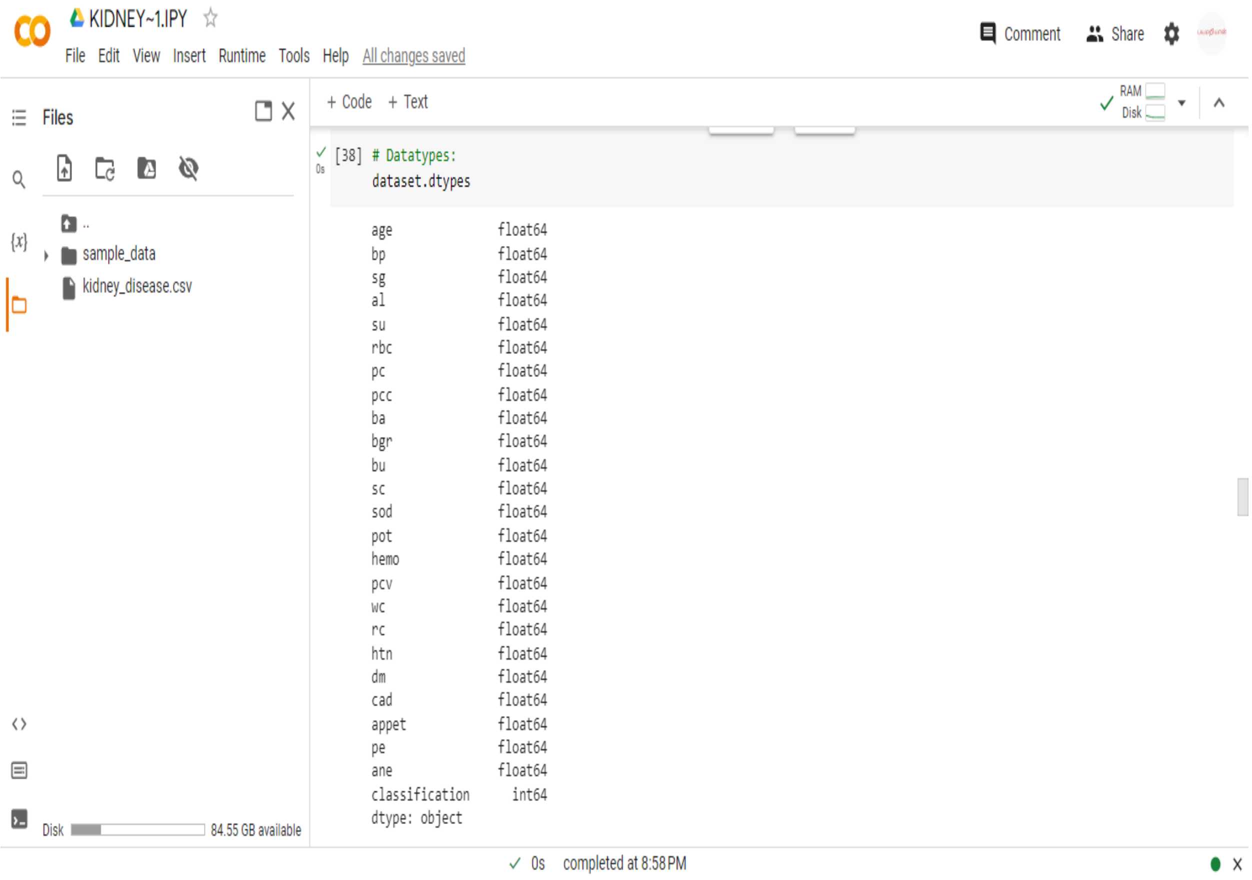
```
# Datatypes:
dataset.dtypes
```

```
age               float64
bp                float64
sg                float64
al                float64
su                float64
rbc               float64
pc                float64
pcc               float64
ba                float64
bgr               float64
bu                float64
sc                float64
sod               float64
pot               float64
hemo              float64
pcv                object
wc                 object
rc                 object
htn               float64
dm                float64
cad               float64
appet             float64
pe                float64
ane               float64
classification      int64
dtype: object
```

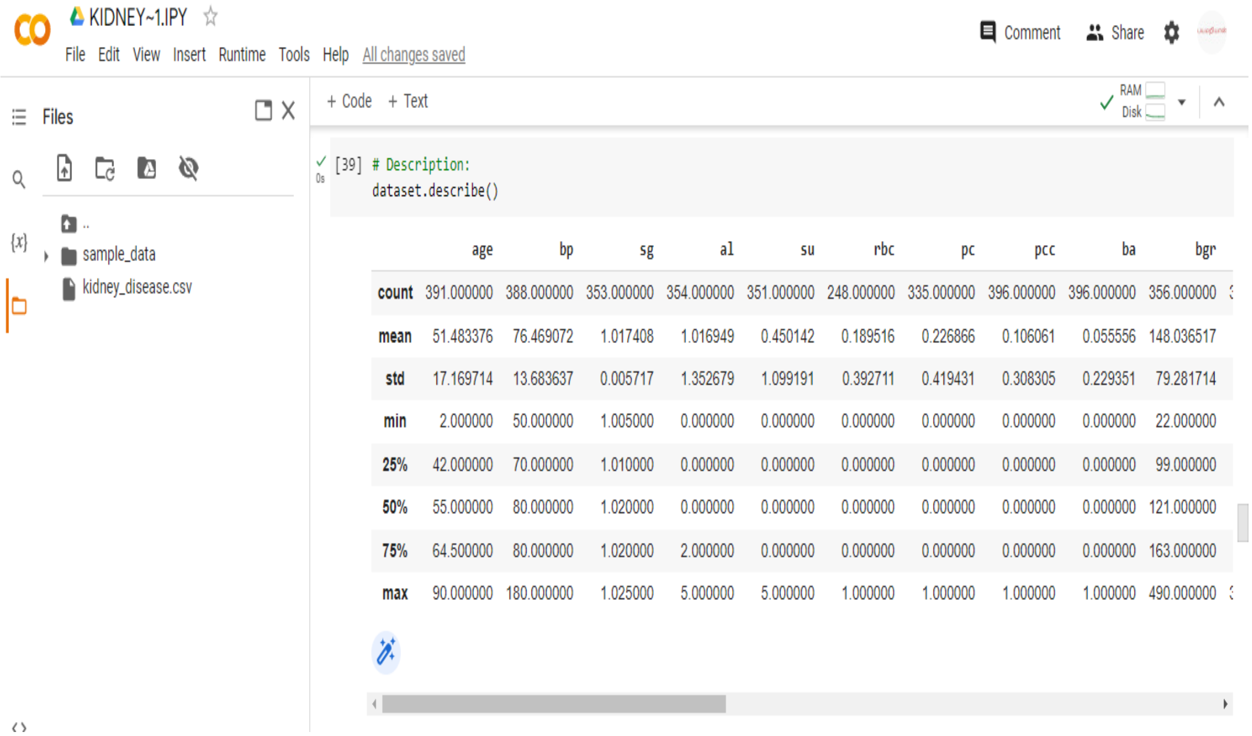


Converting Object values into Numeric values:

```
[37] dataset['pcv'] = pd.to_numeric(dataset['pcv'], errors='coerce')
     dataset['wc'] = pd.to_numeric(dataset['wc'], errors='coerce')
     dataset['rc'] = pd.to_numeric(dataset['rc'], errors='coerce')
```

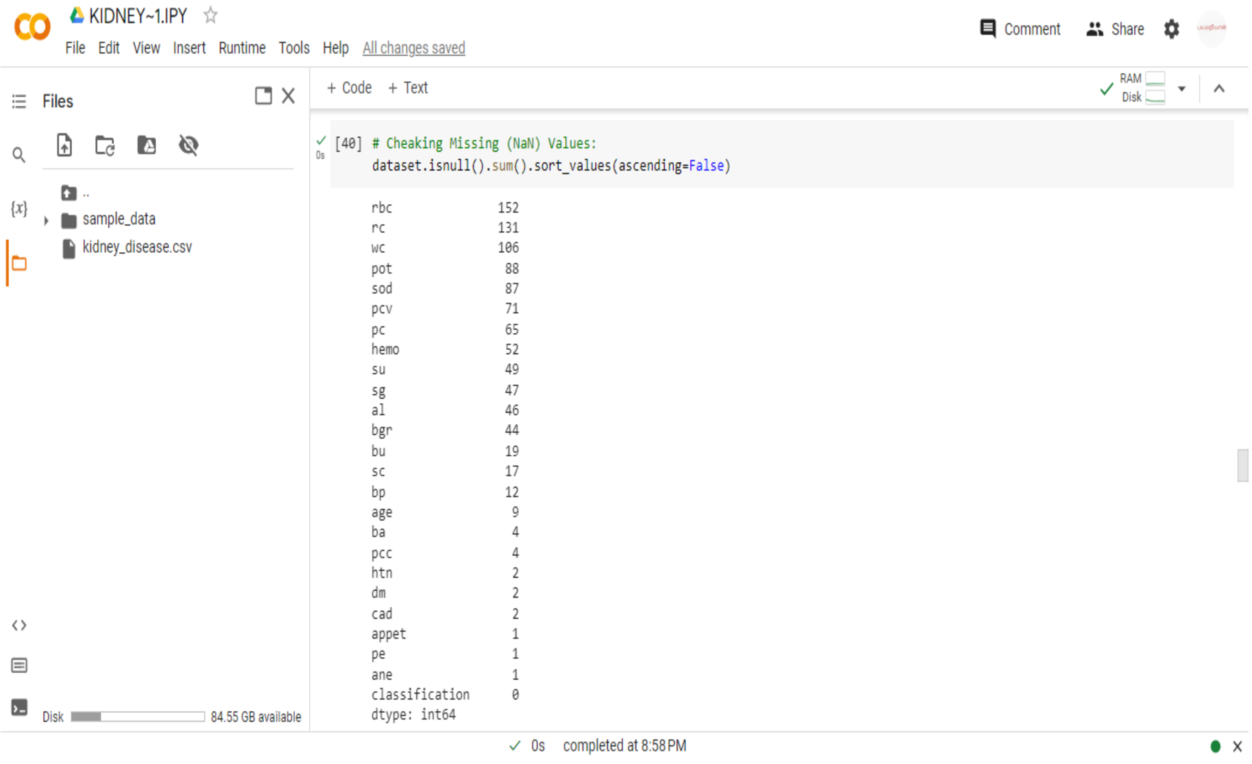


```
[38] # Datatypes:
     dataset.dtypes

age               float64
bp                float64
sg                float64
al                float64
su                float64
rbc               float64
pc                float64
pcc               float64
ba                float64
bgr               float64
bu                float64
sc                float64
sod               float64
pot               float64
hemo              float64
pcv               float64
wc                float64
rc                float64
htn               float64
dm                float64
cad               float64
appet             float64
pe                float64
ane               float64
classification      int64
dtype: object
```



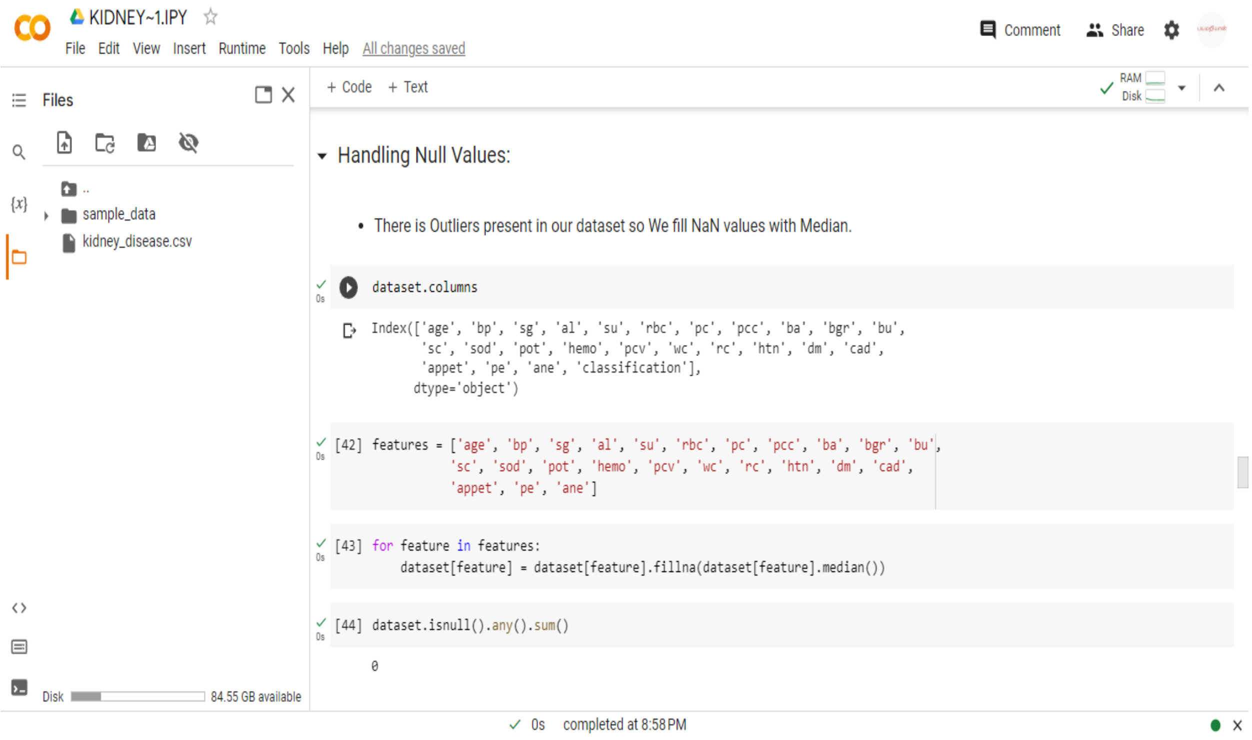
```
[39] # Description:
     dataset.describe()
```

	age	bp	sg	al	su	rbc	pc	pcc	ba	bgr
count	391.000000	388.000000	353.000000	354.000000	351.000000	248.000000	335.000000	396.000000	396.000000	356.000000
mean	51.483376	76.469072	1.017408	1.016949	0.450142	0.189516	0.226866	0.106061	0.055556	148.036517
std	17.169714	13.683637	0.005717	1.352679	1.099191	0.392711	0.419431	0.308305	0.229351	79.281714
min	2.000000	50.000000	1.005000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	22.000000
25%	42.000000	70.000000	1.010000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	99.000000
50%	55.000000	80.000000	1.020000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	121.000000
75%	64.500000	80.000000	1.020000	2.000000	0.000000	0.000000	0.000000	0.000000	0.000000	163.000000
max	90.000000	180.000000	1.025000	5.000000	5.000000	1.000000	1.000000	1.000000	1.000000	490.000000



```
[40] # Cheaking Missing (NaN) Values:
     dataset.isnull().sum().sort_values(ascending=False)
```

```
rbc               152
rc                131
wc                106
pot                88
sod                87
pcv                71
pc                 65
hemo               52
su                 49
sg                 47
al                 46
bgr                44
bu                 19
sc                 17
bp                 12
age                 9
ba                  4
pcc                 4
htn                 2
dm                  2
cad                 2
appet               1
pe                  1
ane                 1
classification      0
dtype: int64
```



Handling Null Values:

- There is Outliers present in our dataset so We fill NaN values with Median.

```
dataset.columns
```

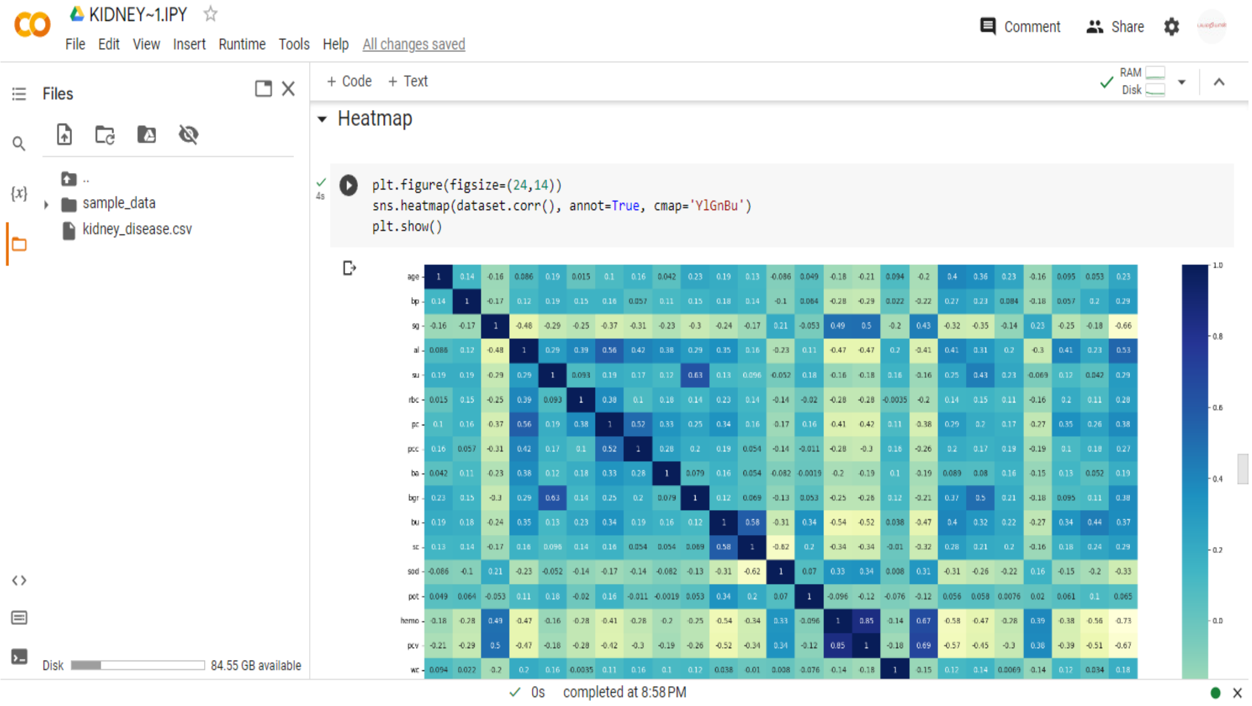
```
Index(['age', 'bp', 'sg', 'al', 'su', 'rbc', 'pc', 'pcc', 'ba', 'bgr', 'bu',
       'sc', 'sod', 'pot', 'hemo', 'pcv', 'wc', 'rc', 'htn', 'dm', 'cad',
       'appet', 'pe', 'ane', 'classification'],
      dtype='object')
```

```
[42] features = ['age', 'bp', 'sg', 'al', 'su', 'rbc', 'pc', 'pcc', 'ba', 'bgr', 'bu',
                'sc', 'sod', 'pot', 'hemo', 'pcv', 'wc', 'rc', 'htn', 'dm', 'cad',
                'appet', 'pe', 'ane']
```

```
[43] for feature in features:
         dataset[feature] = dataset[feature].fillna(dataset[feature].median())
```

```
[44] dataset.isnull().any().sum()
```

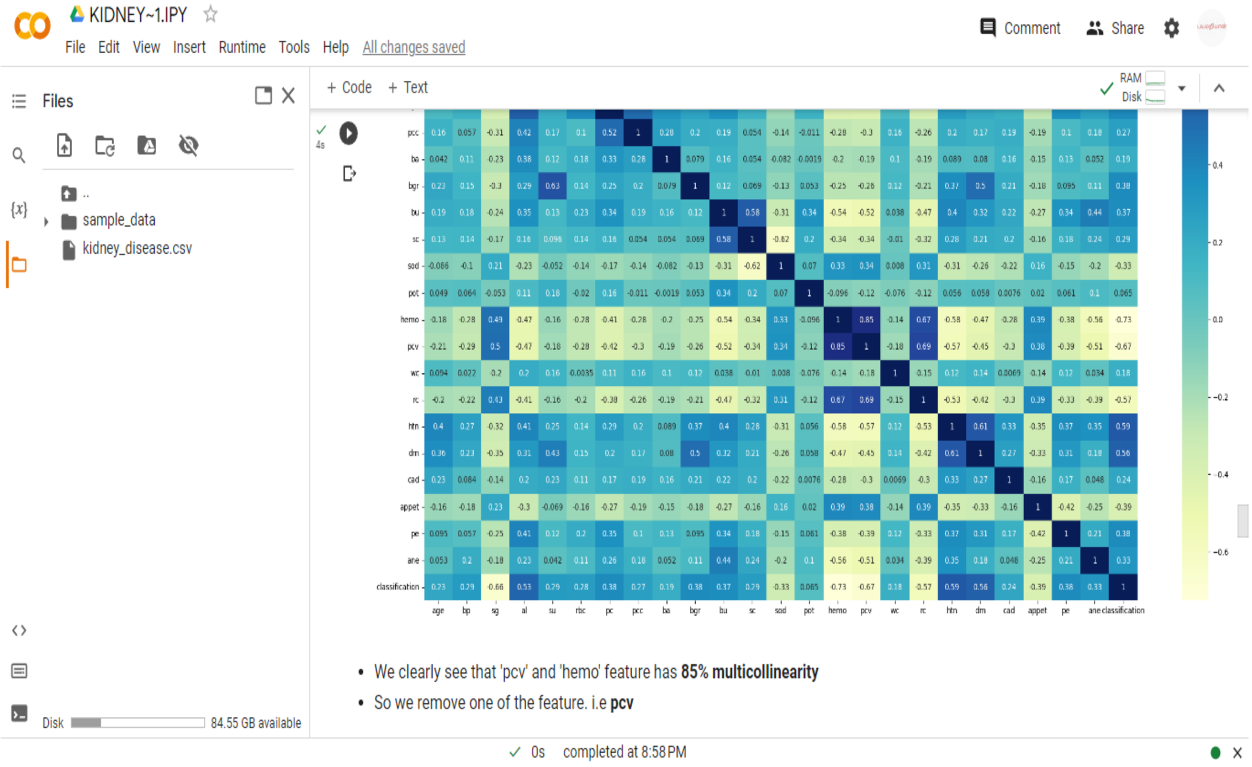
```
0
```



Heatmap

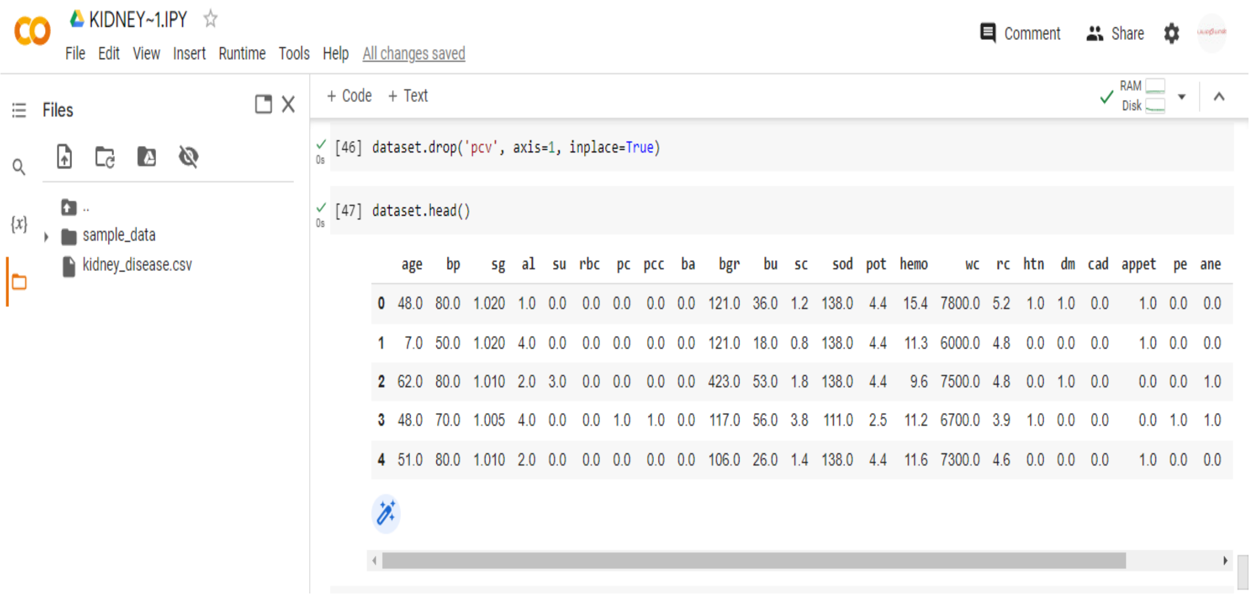
```
plt.figure(figsize=(24,14))
sns.heatmap(dataset.corr(), annot=True, cmap='YlGnBu')
plt.show()
```

✓ 0s completed at 8:58 PM


KIDNEY~1.IPY
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
Files
+ Code + Text
RAM Disk
sample_data
kidney_disease.csv
Disk 84.55 GB available
pcc ba bgr bu sc sod pot hemo pcv wc rc htn dm cad appet pe ane classification
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
0.4 0.2 0.0 −0.2 −0.4 −0.6

- We clearly see that 'pcv' and 'hemo' feature has **85% multicollinearity**
- So we remove one of the feature. i.e **pcv**

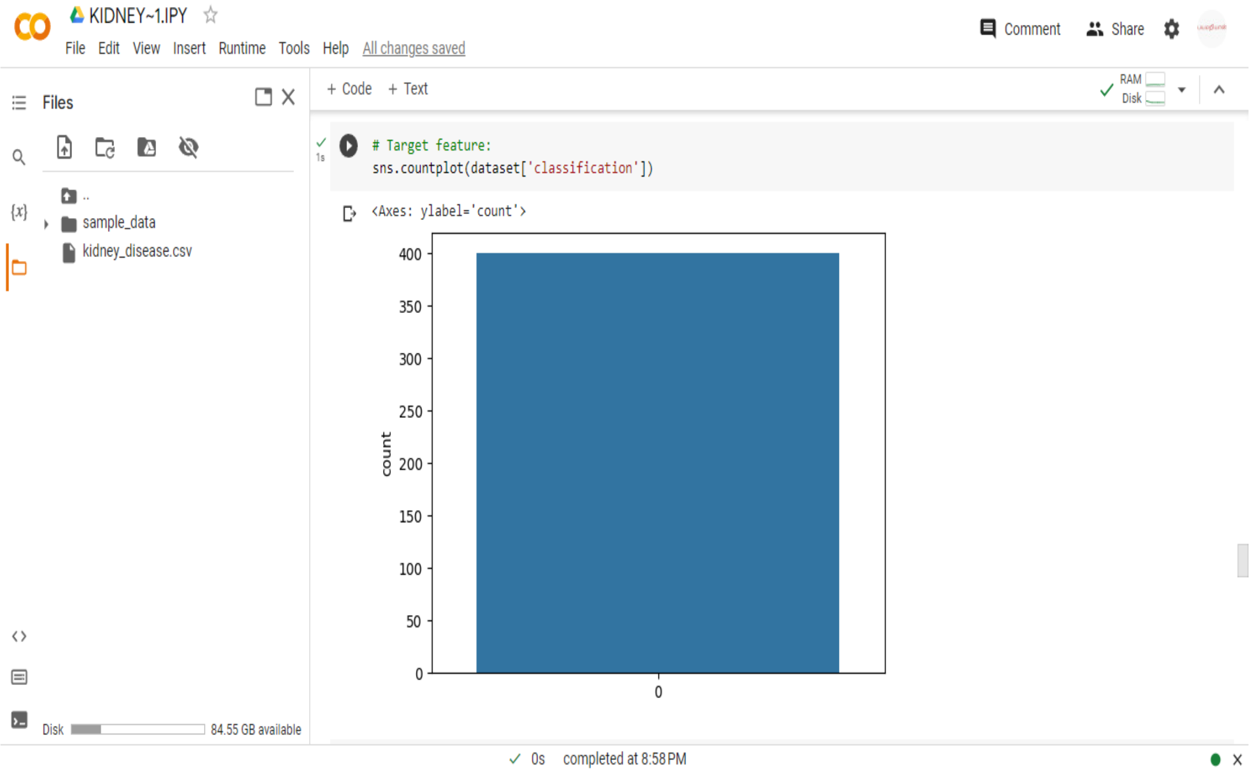
✓ 0s completed at 8:58 PM



```
[46] dataset.drop('pcv', axis=1, inplace=True)
```

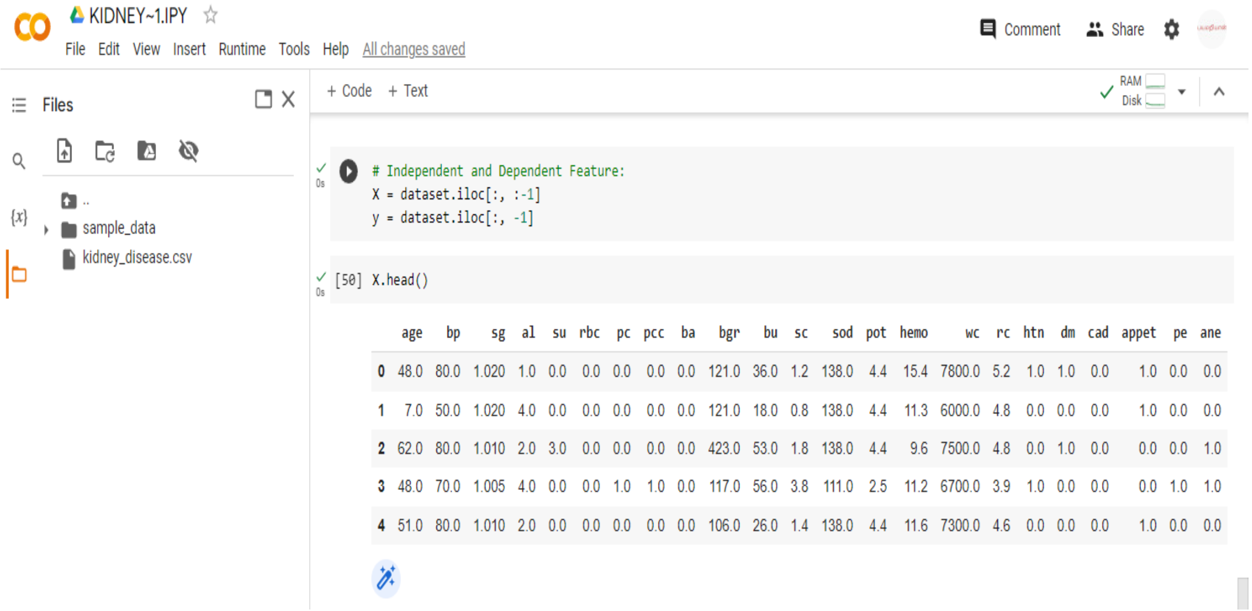
```
[47] dataset.head()
```

	age	bp	sg	al	su	rbc	pc	pcc	ba	bgr	bu	sc	sod	pot	hemo	wc	rc	htn	dm	cad	appet	pe	ane
0	48.0	80.0	1.020	1.0	0.0	0.0	0.0	0.0	0.0	121.0	36.0	1.2	138.0	4.4	15.4	7800.0	5.2	1.0	1.0	0.0	1.0	0.0	0.0
1	7.0	50.0	1.020	4.0	0.0	0.0	0.0	0.0	0.0	121.0	18.0	0.8	138.0	4.4	11.3	6000.0	4.8	0.0	0.0	0.0	1.0	0.0	0.0
2	62.0	80.0	1.010	2.0	3.0	0.0	0.0	0.0	0.0	423.0	53.0	1.8	138.0	4.4	9.6	7500.0	4.8	0.0	1.0	0.0	0.0	0.0	1.0
3	48.0	70.0	1.005	4.0	0.0	0.0	1.0	1.0	0.0	117.0	56.0	3.8	111.0	2.5	11.2	6700.0	3.9	1.0	0.0	0.0	0.0	1.0	1.0
4	51.0	80.0	1.010	2.0	0.0	0.0	0.0	0.0	0.0	106.0	26.0	1.4	138.0	4.4	11.6	7300.0	4.6	0.0	0.0	0.0	1.0	0.0	0.0



```
# Target feature:
sns.countplot(dataset['classification'])
```

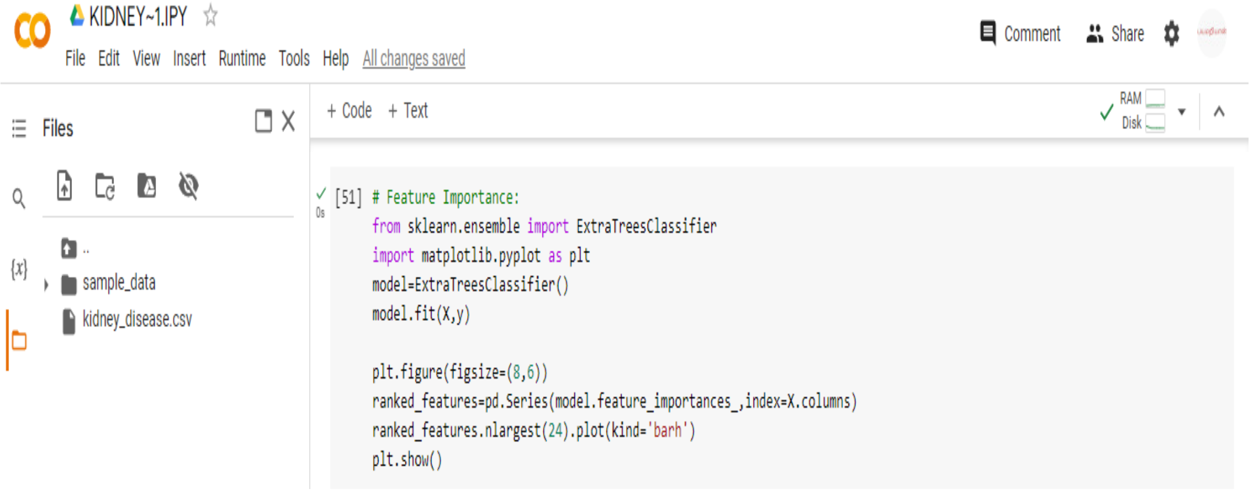
```
<Axes: ylabel='count'>
```



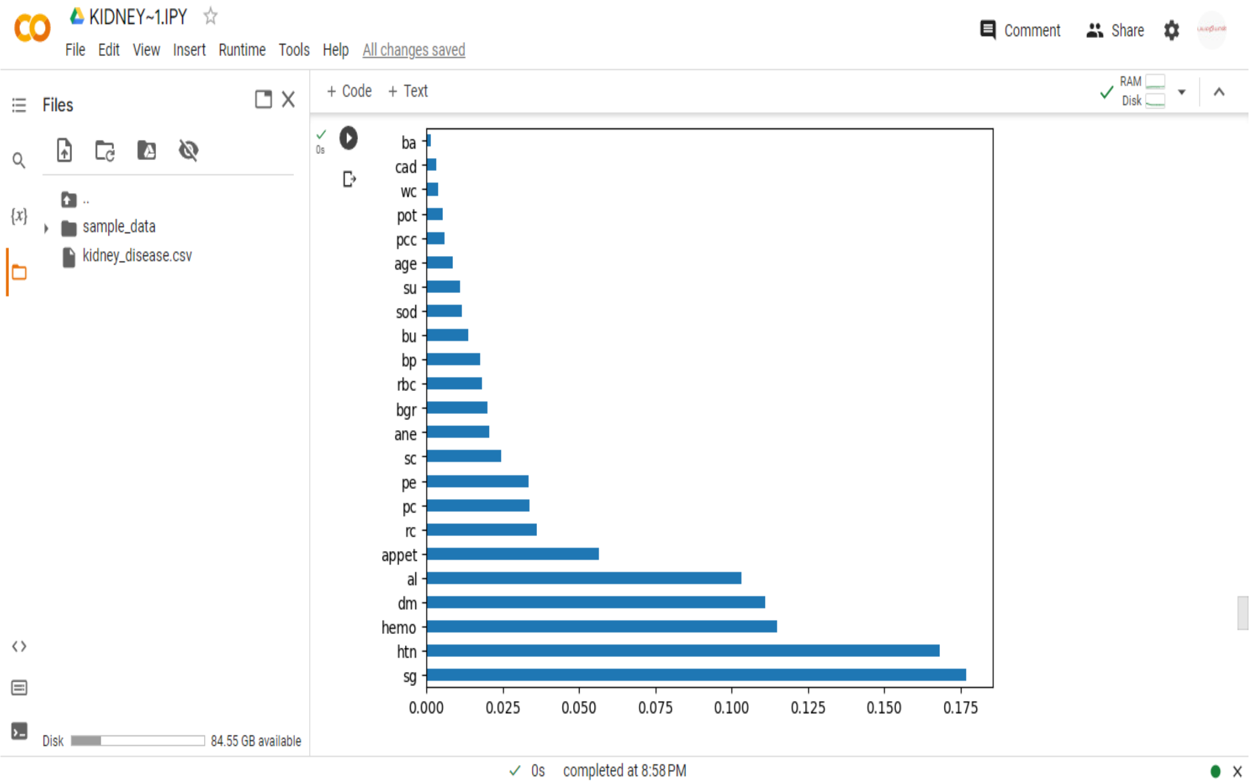
```
# Independent and Dependent Feature:
X = dataset.iloc[:, :-1]
y = dataset.iloc[:, -1]
```

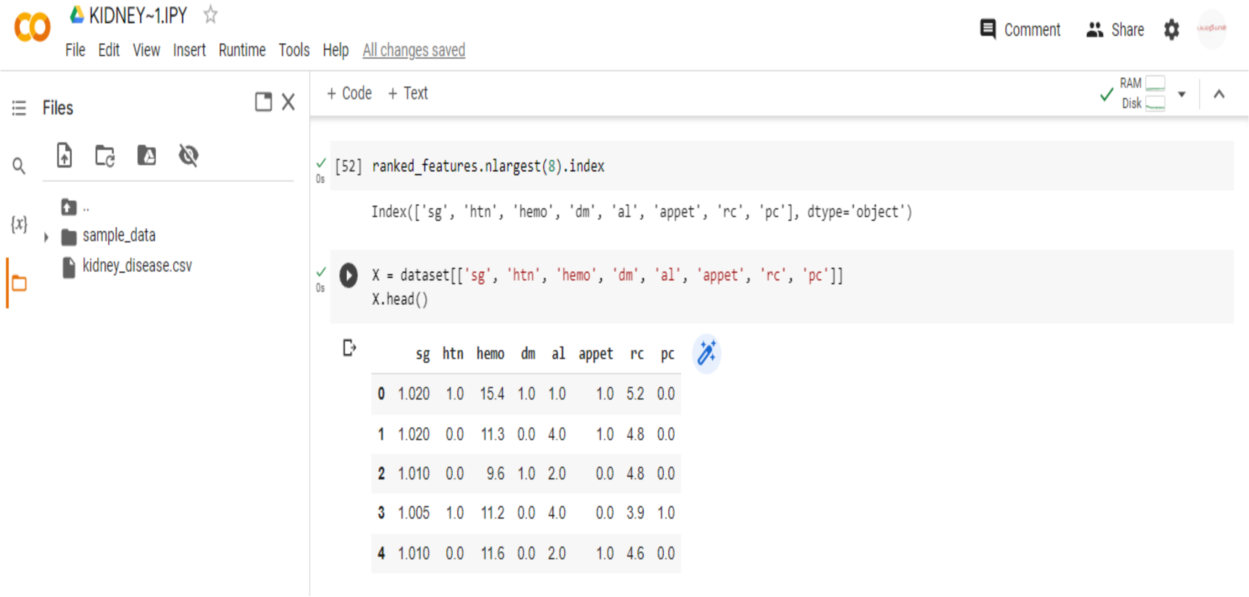
```
[50] X.head()
```

```
    age    bp     sg   al   su  rbc   pc  pcc   ba    bgr    bu   sc    sod  pot  hemo      wc   rc  htn   dm  cad  appet   pe  ane
0  48.0  80.0  1.020  1.0  0.0  0.0  0.0  0.0  0.0  121.0  36.0  1.2  138.0  4.4  15.4  7800.0  5.2  1.0  1.0  0.0    1.0  0.0  0.0
1   7.0  50.0  1.020  4.0  0.0  0.0  0.0  0.0  0.0  121.0  18.0  0.8  138.0  4.4  11.3  6000.0  4.8  0.0  0.0  0.0    1.0  0.0  0.0
2  62.0  80.0  1.010  2.0  3.0  0.0  0.0  0.0  0.0  423.0  53.0  1.8  138.0  4.4   9.6  7500.0  4.8  0.0  1.0  0.0    0.0  0.0  1.0
3  48.0  70.0  1.005  4.0  0.0  0.0  1.0  1.0  0.0  117.0  56.0  3.8  111.0  2.5  11.2  6700.0  3.9  1.0  0.0  0.0    0.0  1.0  1.0
4  51.0  80.0  1.010  2.0  0.0  0.0  0.0  0.0  0.0  106.0  26.0  1.4  138.0  4.4  11.6  7300.0  4.6  0.0  0.0  0.0    1.0  0.0  0.0
```



```
[51] # Feature Importance:
     from sklearn.ensemble import ExtraTreesClassifier
     import matplotlib.pyplot as plt
     model=ExtraTreesClassifier()
     model.fit(X,y)

     plt.figure(figsize=(8,6))
     ranked_features=pd.Series(model.feature_importances_,index=X.columns)
     ranked_features.nlargest(24).plot(kind='barh')
     plt.show()
```

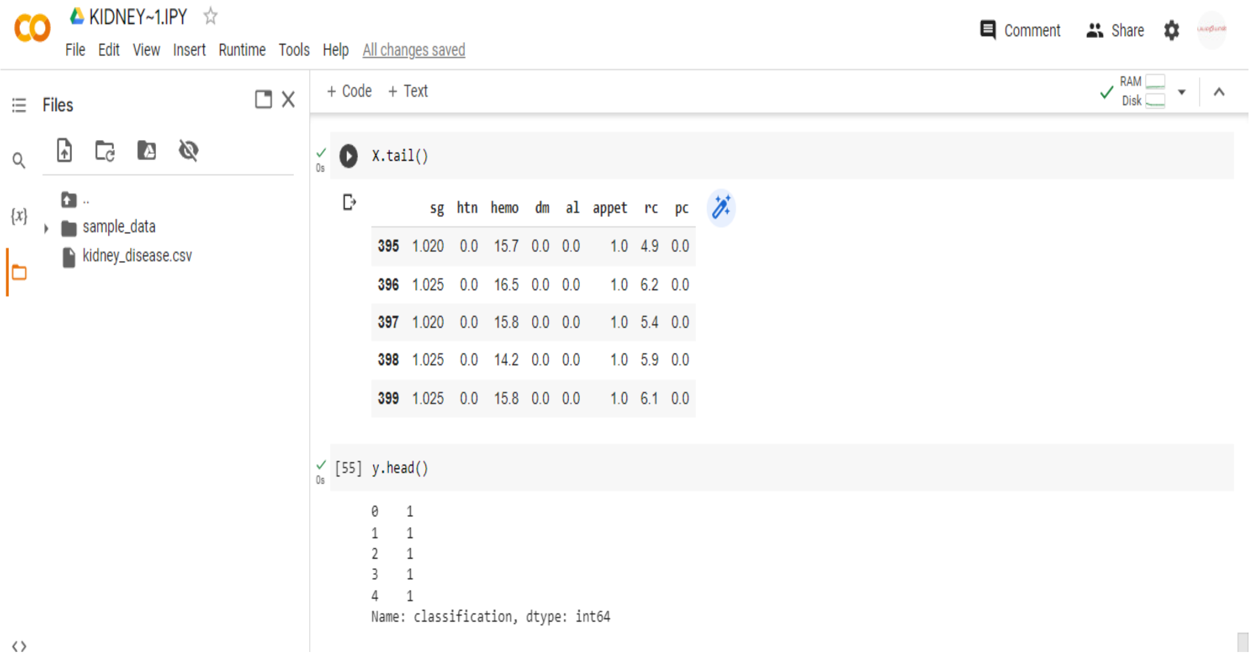




```
[52] ranked_features.nlargest(8).index

     Index(['sg', 'htn', 'hemo', 'dm', 'al', 'appet', 'rc', 'pc'], dtype='object')
```

```
X = dataset[['sg', 'htn', 'hemo', 'dm', 'al', 'appet', 'rc', 'pc']]
X.head()
```

	sg	htn	hemo	dm	al	appet	rc	pc
0	1.020	1.0	15.4	1.0	1.0	1.0	5.2	0.0
1	1.020	0.0	11.3	0.0	4.0	1.0	4.8	0.0
2	1.010	0.0	9.6	1.0	2.0	0.0	4.8	0.0
3	1.005	1.0	11.2	0.0	4.0	0.0	3.9	1.0
4	1.010	0.0	11.6	0.0	2.0	1.0	4.6	0.0

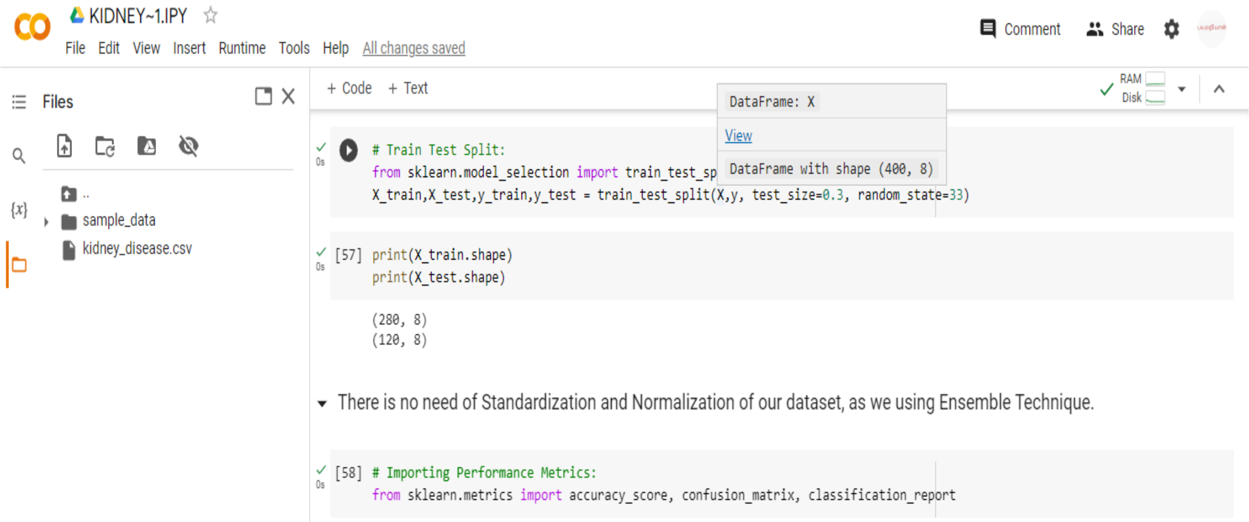


```
X.tail()
```

	sg	htn	hemo	dm	al	appet	rc	pc
395	1.020	0.0	15.7	0.0	0.0	1.0	4.9	0.0
396	1.025	0.0	16.5	0.0	0.0	1.0	6.2	0.0
397	1.020	0.0	15.8	0.0	0.0	1.0	5.4	0.0
398	1.025	0.0	14.2	0.0	0.0	1.0	5.9	0.0
399	1.025	0.0	15.8	0.0	0.0	1.0	6.1	0.0

```
[55] y.head()
```

```
0    1
1    1
2    1
3    1
4    1
Name: classification, dtype: int64
```



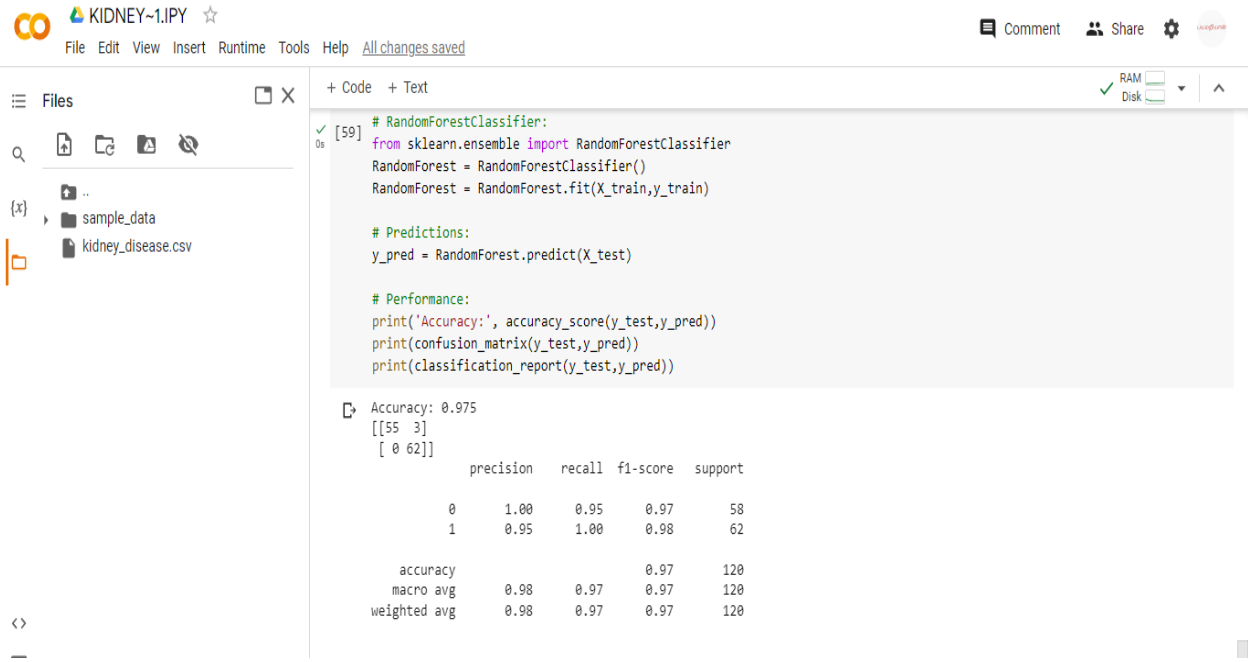
```
# Train Test Split:
from sklearn.model_selection import train_test_sp
X_train,X_test,y_train,y_test = train_test_split(X,y, test_size=0.3, random_state=33)
```

```
[57] print(X_train.shape)
     print(X_test.shape)
```

```
(280, 8)
(120, 8)
```

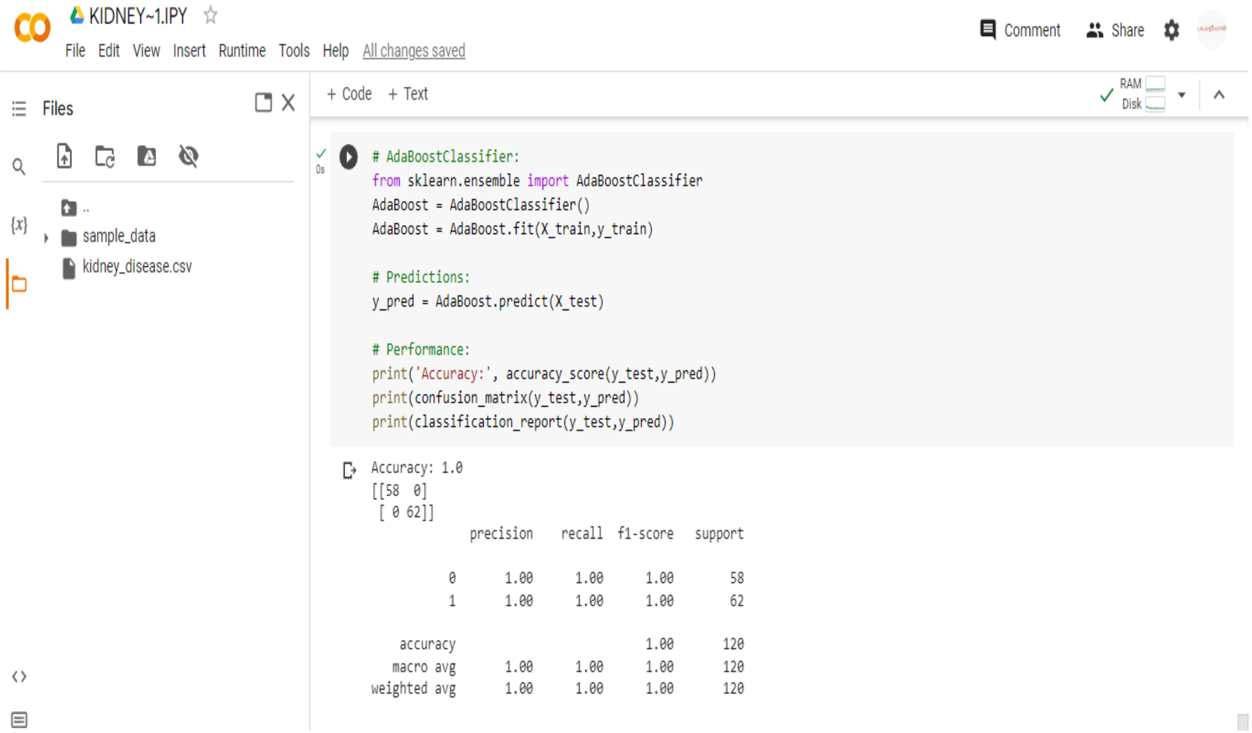
There is no need of Standardization and Normalization of our dataset, as we using Ensemble Technique.

```
[58] # Importing Performance Metrics:
     from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
```



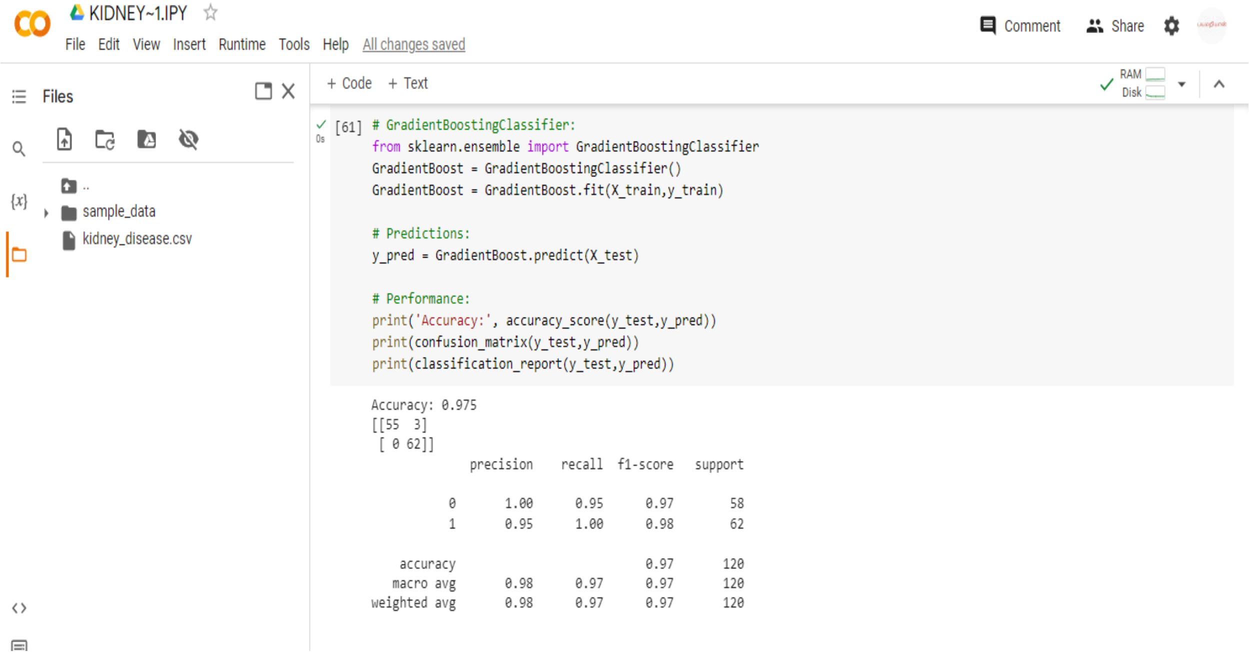
```
[59]
# RandomForestClassifier:
from sklearn.ensemble import RandomForestClassifier
RandomForest = RandomForestClassifier()
RandomForest = RandomForest.fit(X_train,y_train)

# Predictions:
y_pred = RandomForest.predict(X_test)

# Performance:
print('Accuracy:', accuracy_score(y_test,y_pred))
print(confusion_matrix(y_test,y_pred))
print(classification_report(y_test,y_pred))
```

```
Accuracy: 0.975
[[55  3]
 [ 0 62]]
              precision    recall  f1-score   support

           0       1.00      0.95      0.97        58
           1       0.95      1.00      0.98        62

    accuracy                           0.97       120
   macro avg       0.98      0.97      0.97       120
weighted avg       0.98      0.97      0.97       120
```



```
# AdaBoostClassifier:
from sklearn.ensemble import AdaBoostClassifier
AdaBoost = AdaBoostClassifier()
AdaBoost = AdaBoost.fit(X_train,y_train)

# Predictions:
y_pred = AdaBoost.predict(X_test)

# Performance:
print('Accuracy:', accuracy_score(y_test,y_pred))
print(confusion_matrix(y_test,y_pred))
print(classification_report(y_test,y_pred))
```

```
Accuracy: 1.0
[[58  0]
 [ 0 62]]
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        58
           1       1.00      1.00      1.00        62

    accuracy                           1.00       120
   macro avg       1.00      1.00      1.00       120
weighted avg       1.00      1.00      1.00       120
```



```
# GradientBoostingClassifier:
from sklearn.ensemble import GradientBoostingClassifier
GradientBoost = GradientBoostingClassifier()
GradientBoost = GradientBoost.fit(X_train,y_train)

# Predictions:
y_pred = GradientBoost.predict(X_test)

# Performance:
print('Accuracy:', accuracy_score(y_test,y_pred))
print(confusion_matrix(y_test,y_pred))
print(classification_report(y_test,y_pred))
```

```
Accuracy: 0.975
[[55  3]
 [ 0 62]]
              precision    recall  f1-score   support

           0       1.00      0.95      0.97        58
           1       0.95      1.00      0.98        62

    accuracy                           0.97       120
   macro avg       0.98      0.97      0.97       120
weighted avg       0.98      0.97      0.97       120
```