Machine Learning Project

Chronic Kidney Disease

Done by

Sirine HAMDI

Yasmine SLITI

Rima REGGUI

Dhouha HMEM

Asma MARROUKI

Firas CHKOUNDALI

2022-2023

Table of Contents

Introduction

Chronic kidney disease (CKD) has been recognized as a leading public health problem worldwide. The global burden of CKD is rapidly increasing and is projected to become the 5th most common cause of years of life lost by 2040 .
Using machine learning in healthcare like machine learning algorithms can help us find patterns and insights that would be impossible to find manually.
As machine learning in healthcare gains widespread adoption, healthcare providers have an opportunity to take a more predictive approach that creates a more unified system with improved care delivery and patient based processes.

This research work aims to design and implement a machine learning model that, based on data from clinical laboratories, allows predicting the possible diagnosis of CKD in its initial stages, helping reduce the mortality rate and costs for the health system, thus, the time required for diagnosis is reduced, allowing the patient to receive treatment for the disease before it progresses to a stage of no return.

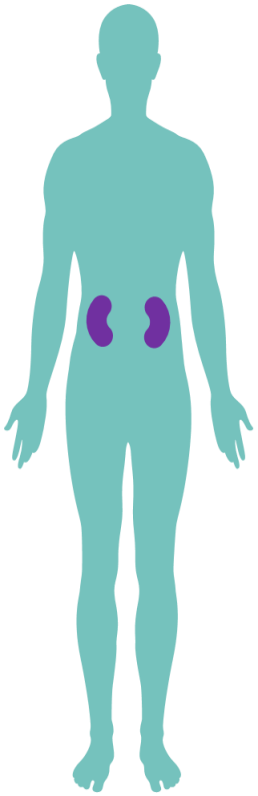
I. Business Understanding

Introduction :

The Business Understanding phase focuses on understanding the objectives and requirements of the project . Any good project starts with a deep understanding of the customer's needs.
In this section, we will start by defining the kidney organ and by explaining how it performs its important functions.
We will then explain chronic kidney diseases , its early signs and risk factors .
And we will conclude by illustrating how we can prevent CKD from happening.

1. Medical Definition of Kidney and their functionality

1.1 What are kidneys :



The kidneys are 2 bean shaped organs, about the size of a fist, located near the lower middle part of your back on either side of your spine in the retroperitoneal space . Your kidneys are vital organs. This means you need at least one healthy kidney to live. They are as critical to life as your heart, lungs, and other organs.
Most people have 2 kidneys, but some are born with just one. It is possible to live with just one kidney.

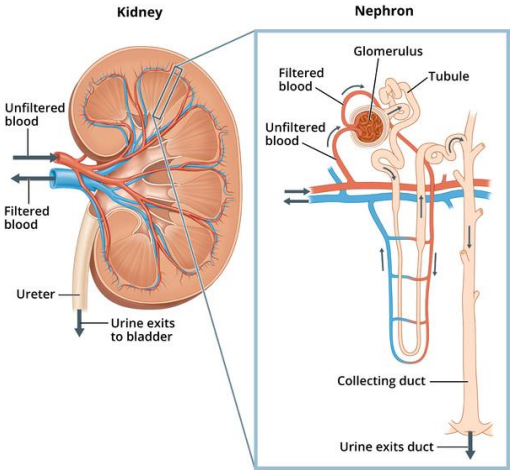
1.2 What do your kidneys do ?

The main job of your kidneys is to filter waste and extra fluid out of your blood. You may not realize it, but your body makes waste all the time. The waste comes from food you eat and from using your muscles.
The kidneys have other important jobs that keep your body working the way it should. Your kidneys also help:

- filter the blood and produce urine to eliminate waste from the body
- produce hormones to control blood pressure,the production of red blood cells etc ...
- control the hydroelectrolytic balance of the body.
- keep the right amount of minerals in your body such as salt, potassium, and calcium.

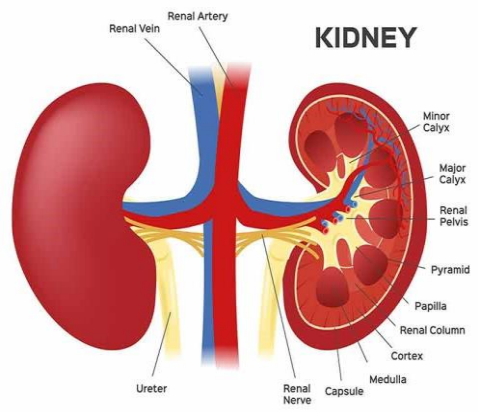
1.3. How do Kidneys work?

Here's how your kidneys work:



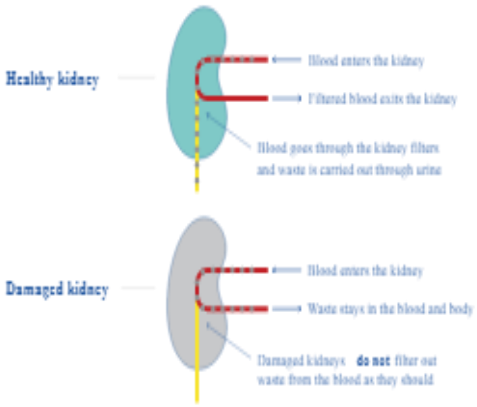
- Blood enters your kidneys that contains millions of filtering units called nephrons each nephron includes a filter called the glomerulus, and a tubule..
- The glomerulus filters blood from waste and extra fluid to make urine .
- The tubule returns needed substances to the blood .
- The clean, filtered blood leaves your kidneys and flows back into the rest of your body
- The urine flows from your kidneys to your bladder through 2 thin tubes called ureters
- When you urinate, the waste leaves your body

1.4 What happens when your kidneys are damaged ?



Different health problems or injury can damage your kidneys.
If your kidneys are damaged, they will not work as well as they should to filter waste and water from your blood.
This causes waste and fluid to build up in the body, which leads to heart and lung problems. It could also lead to bone disease and anemia.
Anemia is a low number of red blood cells, which can make you feel tired and weak.
If the damage (even little damage) to your kidneys gets worse and your kidneys are less able to do their job, you have chronic kidney disease.

2. The definition of chronic kidney disease



2.1 What is chronic kidney disease ?

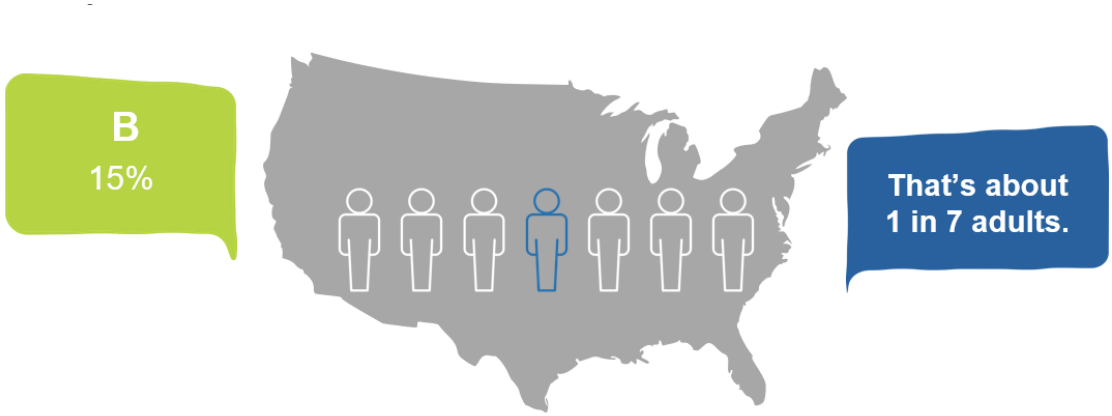
chronic kidney disease is due to a reduction in the number of functional nephrons. the mechanism of this reduction is twofold:

-initial destruction related to the causative disease.

-the hyperfunction of the remaining nephrons will lead to glomerulosclerosis. diuresis and sodium excretion are maintained until the final stage renal failure .

However, if you find out you have CKD early in time and take the suitable steps to keep your kidneys healthy, you can prevent or slow the damage from getting worse**.** Also, there has been an increase in available treatments to prevent CKD from getting worse.

2.2 How many adults in the U.S. have CKD?

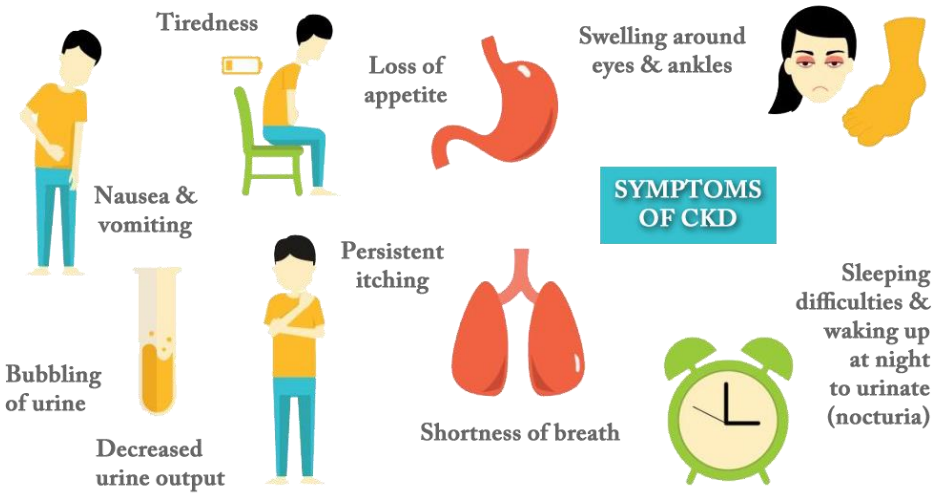


2.3 Stages of CKD

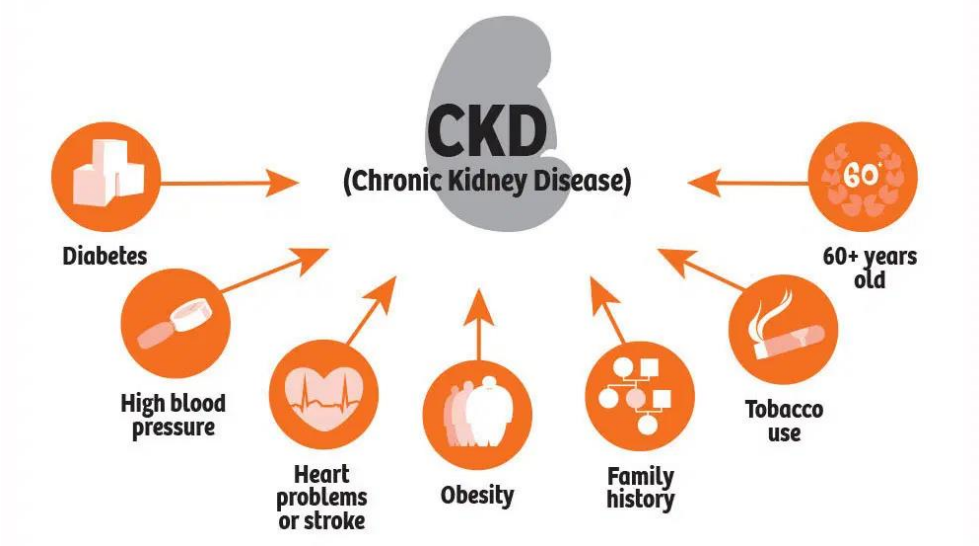
Stage of CKD	How much damage	How well the kidneys function (work)
Stage 1	Mild	Normal to slight loss of function
Stage 2	Mild	Mild loss of function
Stage 3a	Mild to moderate	Mild to medium loss of function
Stage 3b	Moderate to severe	Medium to severe loss of function
Stage 4	Severe	Severe loss of function
Stage 5 (kidney failure, ESRD/ESKD)	Most severe	Near total or total loss of function

2.4 Symptoms of CKD

- Dry itchy skin
- yellowish skin
- Muscle cramps
- Decreased urine
- Anemia
- dyspepsia which is a functional digestive disorder represented by loss of appetite, nausea, burning, hiccups, bloating
- digestive hemorrhage with signs of bloody vomit and black stool which is called melena
- Gout is a type of arthritis that causes pain and swelling in your joints.
- High potassium or hyperkalemia .
- high blood pressure
- dyspnea is the sensation of difficult and uncomfortable breathing also known as shortness of breath



2.5 Risk factor of CKD :



2.6 Etiology of CKD :

the causes of chronic kidney disease are :

- Type 1 or type 2 diabetes
- High blood pressure
- Polycystic kidney disease or other inherited kidney diseases
- Prolonged obstruction of the urinary tract, from conditions such as enlarged prostate, kidney stones and some cancers.
- Vesicoureteral reflux, a condition that causes urine to back up into your kidneys.
- Recurrent kidney infection, also called pyelonephritis .
- Nephrotoxic diseases : intoxication of the kidney caused by medication
- Autoimmune diseases : the result of a dysfunction of the immune system leading it to attack the normal components of the body.

2.7 How to Check for CKD:

The only way to know if you have CKD is to get tested.

- **Urine test: urine albumin-to-creatinine ratio (UACR)**

Uses a microscope to look at urine in a lab

Shows how much of these are in your urine:
•Albumin: a type of protein
•Creatinine: a waste product in the blood

UACR result (mg/g)	What it means
Less than 30	Normal
30 – 300	High
300 and over	Very high

❖ Blood test: eGFR – estimated Glomerular Filtration Rate

•Based on your blood test and other factors
•It tells how well your kidneys are working
•A doctor will look at your eGFR over 3 months to decide if you have kidney disease
•The stages of CKD are based on eGFR

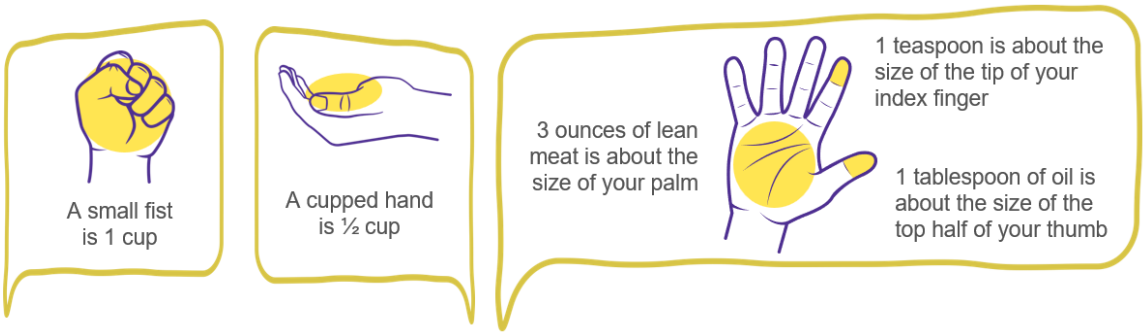
Stage of CKD	Ranges of eGFR	How much damage
Stage 1	90 or more	Kidney function is almost normal
Stage 2	60–89	Kidney function is slightly less than normal
Stage 3a	45–59	Mild to moderate damage to the kidneys
Stage 3b	30–44	Moderate to severe damage to the kidneys
Stage 4	15–29	Serious damage to the kidneys
Stage 5 (kidney failure, ESRD/ESKD)	Less than 15	Kidney failure

3. Prevention of CKD

- Control your blood sugar if you have diabetes
- Quit smoking .
- Control your blood pressure if you have high blood pressure
- Eat healthy :
  - •Limit your intake of salt and sugar
  - •Choose healthy fats
  - •Eat more fruits, vegetables, and whole grains

- Control portion sizes :
  - •Eat slowly (about 20 minutes) and stop when full
  - •Check the nutrition label to find the serving size

  Measure or estimate food for the correct serving size



Conclusion :

In this chapter, we did a deep dive into the world of medicine in order to have a better understanding of the business context of our project .This will help us better understand our data which is our next section.

II. Data Understanding

Introduction :

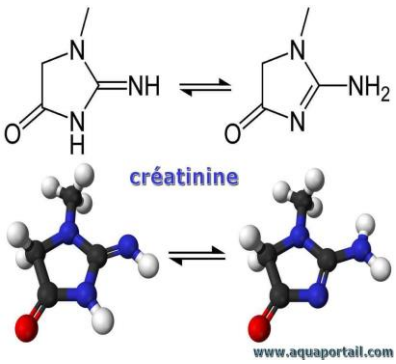
The main goal of data understanding is to gain general insights about the data that will potentially be helpful for the further steps in the data analysis process.

In this section , we dissect each variable in our dataset which belongs to the UCI repository.

The data was collected from the Apollo Hospital, India by b. Jerlin Rubini.

SC (Serum Creatinine)

Creatinine is a waste product in your blood that comes from your muscles. Healthy kidneys filter creatinine out of your blood through your urine.Your serum creatinine level is based on a blood test that measures the amount of creatinine in your blood. It tells how well your kidneys are working. When your kidneys are not working well, your serum creatinine level goes up.



A normal result is :

men - 0.7 to 1.3 mg/dL (61.9 to 114.9 µmol/L)

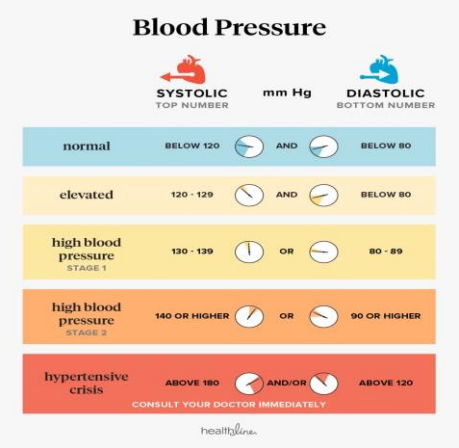
women - 0.6 to 1.1 mg/dL (53 to 97.2 µmol/L)

AL (Albumin)

Albumin is a protein in your blood plasma.
Normal albumin levels in an adult's blood range from 3.5 to 5.5 grams per deciliter (g/dL).

- Low albumin levels might be the result of kidney disease, liver disease, inflammation or infections.

- High albumin levels are usually the result of dehydration or severe diarrhea.



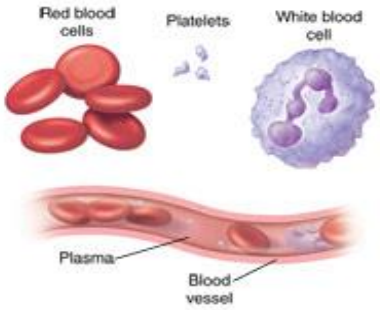
BD (Blood Pressure)

Blood pressure is a measure of the force that your heart uses to pump blood around your body.
Blood pressure is measured in millimetres of mercury (mmHg):

- Systolic pressure – the pressure when your heart pushes blood out
- Diastolic pressure – the pressure when your heart rests between beats

- Ideal blood pressure is considered to be between 90/60mmHg and 120/80mmHg
- High blood pressure is considered to be 140/90mmHg or higher
- Low blood pressure is considered to be 90/60mmHg or lower

RBC (Red Blood Cells)



Red blood cells carry fresh oxygen all over the body. This is important to your health.
Red blood cells are round with a flattish, indented center, like doughnuts without a hole. Your healthcare provider can check on the size, shape, and health of your red blood cells using a blood test.
A normal RBC count be around :

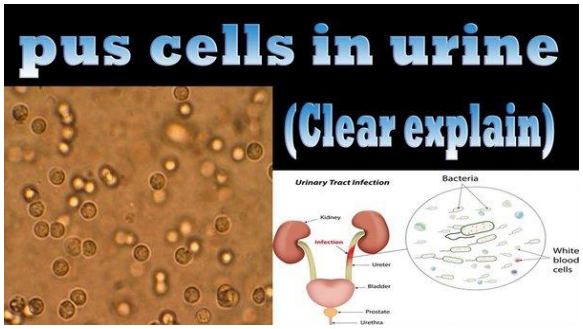
men– 4.0 to 5.9 x 10*12/L
women – 3.8 to 5.2 x 10*12/L

PC (Pus cells)

PC refers to the term given to the accumulation of dead white blood cells (WBCs) at the site of infection. When these pus cells are present in the human urine, the condition is known as Pyuria**.**

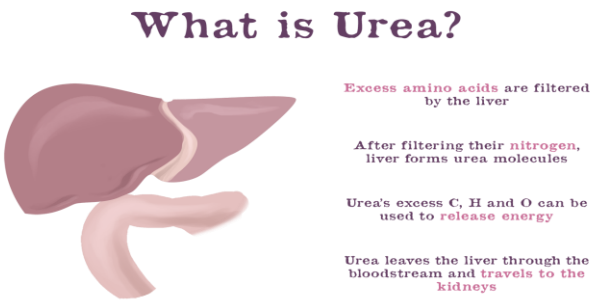
The normal range for pus cells from the urine is 0-5.

The presence of 8-10 pus cells suggests bacterial infection, which is mostly diagnosed as urinary tract infection(UTI).



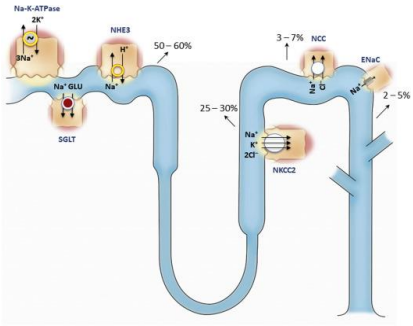
BU (Blood Urea)

The liver produces urea in the urea cycle as a waste product of the digestion of protein. Normal human adult blood should contain 6 to 20 mg/dL (2.1 to 7.1 mmol/L) of urea nitroge



SOD (Sodium)

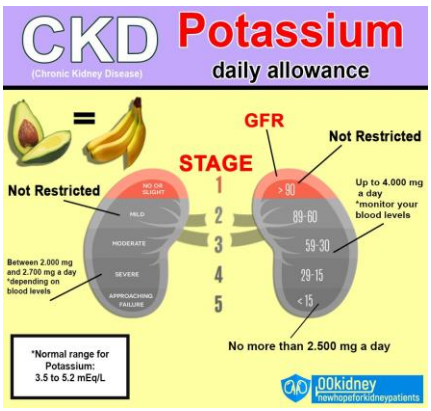
Too much sodium can be harmful for people with kidney disease because your kidneys cannot eliminate excess sodium and fluid from your body. As sodium and fluid build up in your tissues and bloodstream, your blood pressure increases and you feel uncomfortable.



POT (Potassium)

When kidneys fail they can no longer remove excess potassium, so the level builds up in the body. High potassium in the blood is called hyperkalemia, which may occur in people with advanced stages of chronic kidney disease (CKD). Some of the effects of high potassium are nausea, weakness, numbness and slow pulse.

Your blood potassium level is normally 3.6 to 5.2 millimoles per liter (mmol/L).

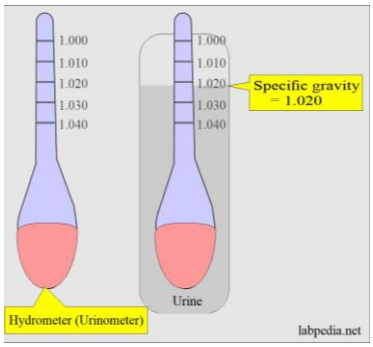


HTN (Hypertension)

Blood pressure is the force exerted by circulating blood against the walls of the body’s arteries, the major blood vessels in the body. Hypertension is when blood pressure is too high.
Since blood pressure is written as in two numbers .Hypertension is diagnosed if, when it is measured on two different days,the systolic blood pressure(the first number) readings on both days is ≥140 mmHg and/or the diastolic(the second number) blood pressure readings on both days is ≥90 mmHg

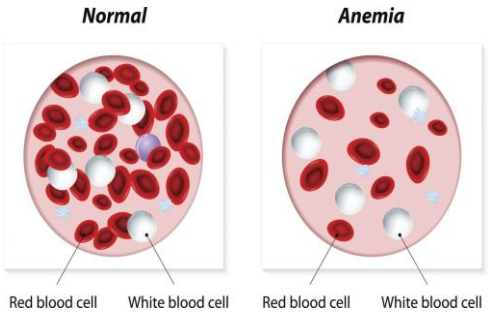
SG (Specific Gravity)

It is a urinalysis parameter commonly used in the evaluation of kidney function and can aid in the diagnosis of various renal diseases.It shows the concentration of all chemical particles in the urine.Normal urine specific gravity is 1.00



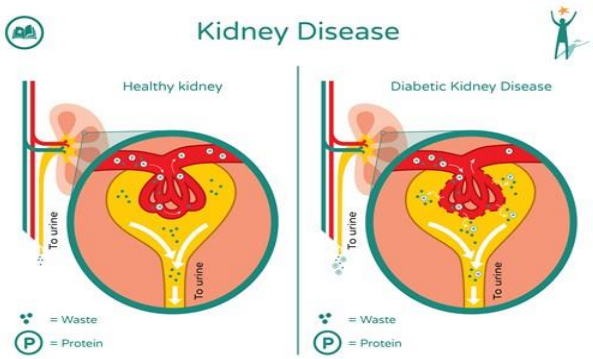
ANE (Anemia)

Anemia is a condition in which you lack enough healthy red blood cells to carry adequate oxygen to your body's tissues. Having anemia, also referred to as low hemoglobin, can make you feel tired and weak.
There are many forms of anemia, each with its own cause such as Iron deficiency,Vitamin deficiency,bone marrow disease.
Anemia can be temporary or long term and can range from mild to severe.



DM (Diabetes Mellitus)

Diabetes is the leading cause of chronic kidney disease.
Over time, having high blood sugar from diabetes can cause damage inside your kidneys. As a result, they filter out some good things along with waste.
Controlling blood sugar helps lessen your risk for getting kidney disease. It can also help slow or even stop kidney disease from getting worse.
In general, the recommended targets for most people are:



- Before meals: 90-130 mg/dL
- Two hours after the start of a meal: Below 180 mg/dL
- A1C Test: Around 7%

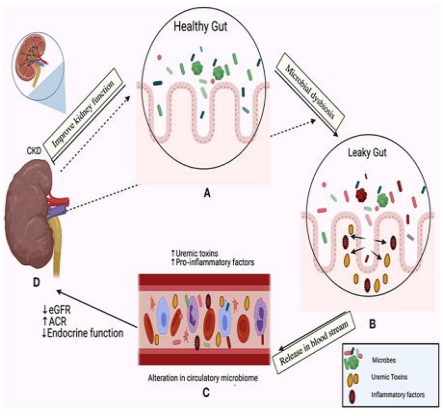
BGR (Blood glucose random):

It is a measure of blood sugar at the time you're tested. It can be measured at any time and it doesn't need fasting or eating first.

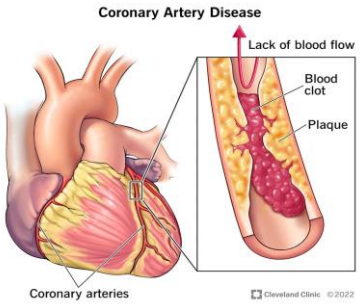
A blood sugar level of 200 mg/dL or higher indicates you have diabetes.

BA (Bacteria)

The gut microbiome is composed of a diverse population of bacteria that have beneficial and adverse effects on human health. Increasing urea concentration during chronic kidney disease (CKD) leads to alterations in the intestinal flora that can increase production of gut-derived toxins and alter the intestinal epithelial barrier. These changes can lead to an acceleration of the process of kidney injury. A number of strategies have been proposed to interrupt this pathway of injury in CKD.



CAD (Coronary Artery Disease)



Coronary artery disease is a common heart condition.
The major blood vessels that supply the heart (coronary arteries) struggle to send enough blood, oxygen and nutrients to the heart muscle.
Cholesterol deposits (plaques) in the heart arteries and inflammation are usually the cause of coronary artery disease.

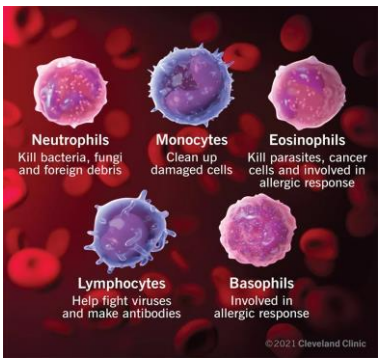
APPET (Appetite)

Appetite is a person's desire to eat food. It is distinct from hunger, which is the body's biological response to a lack of food. A person can have an appetite even if their body is not showing signs of hunger, and vice versa.

PE (Pedal Edema)

Pedal edema causes an abnormal accumulation of fluid in the ankles, feet, and lower legs causing swelling of the feet and ankles.Edema may be caused by drugs, pregnancy, infections, and many other medical issues. Edema occurs when small blood vessels leak fluid into nearby tissues.

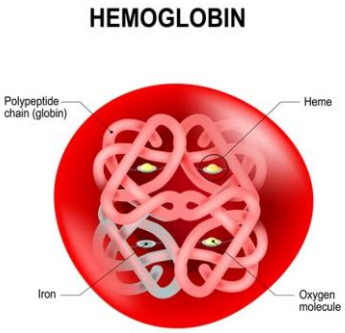
WC (white blood cells count)



White blood cells, also known as leukocytes, are responsible for protecting your body from infection. As part of your immune system, white blood cells circulate in your blood and respond to injury or illness.
The normal number of WBCs in the blood is 4,500 to 11,000 WBCs per microliter (4.5 to 11.0 × 109/L).

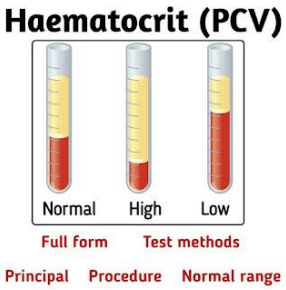
HEMO (Hemoglobin)

Hemoglobin is the iron *NIH external link*-rich protein that allows red blood cells to carry oxygen from your lungs to the rest of your body. With fewer red blood cells or less hemoglobin, your tissues and organs—such as your heart and brain—may not get enough oxygen to work properly.



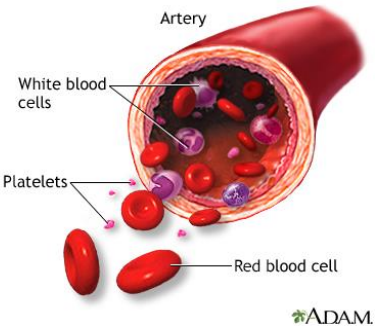
PCV (Packed Cell Volume)

The PCV, reticulocyte count and reticulocyte index seems to worsen with increasing severity of kidney disease. There is need for the provision of erythropoietin for the evidenced-based management of anemia in renal patients.



RC (Red Blood Cell Count):

Red blood cells, also called erythrocytes, cellular components of blood, millions of which in the circulation of vertebrates give the blood its characteristic color and carry oxygen from the lungs to the tissues.
A normal range in adults is generally considered to be 4.35 to 5.65 million red blood cells per microliter (mcL) for men and 3.92 to 5.13 million red blood cells per mcL of blood for women. In children, the threshold for high red blood cell count varies with age and sex.



Conclusion :

In this section , we got a better understanding of the different variables in our dataset , this ensures that the next section data preprocessing is more accurate and meaningful.

III. Data Preprocessing

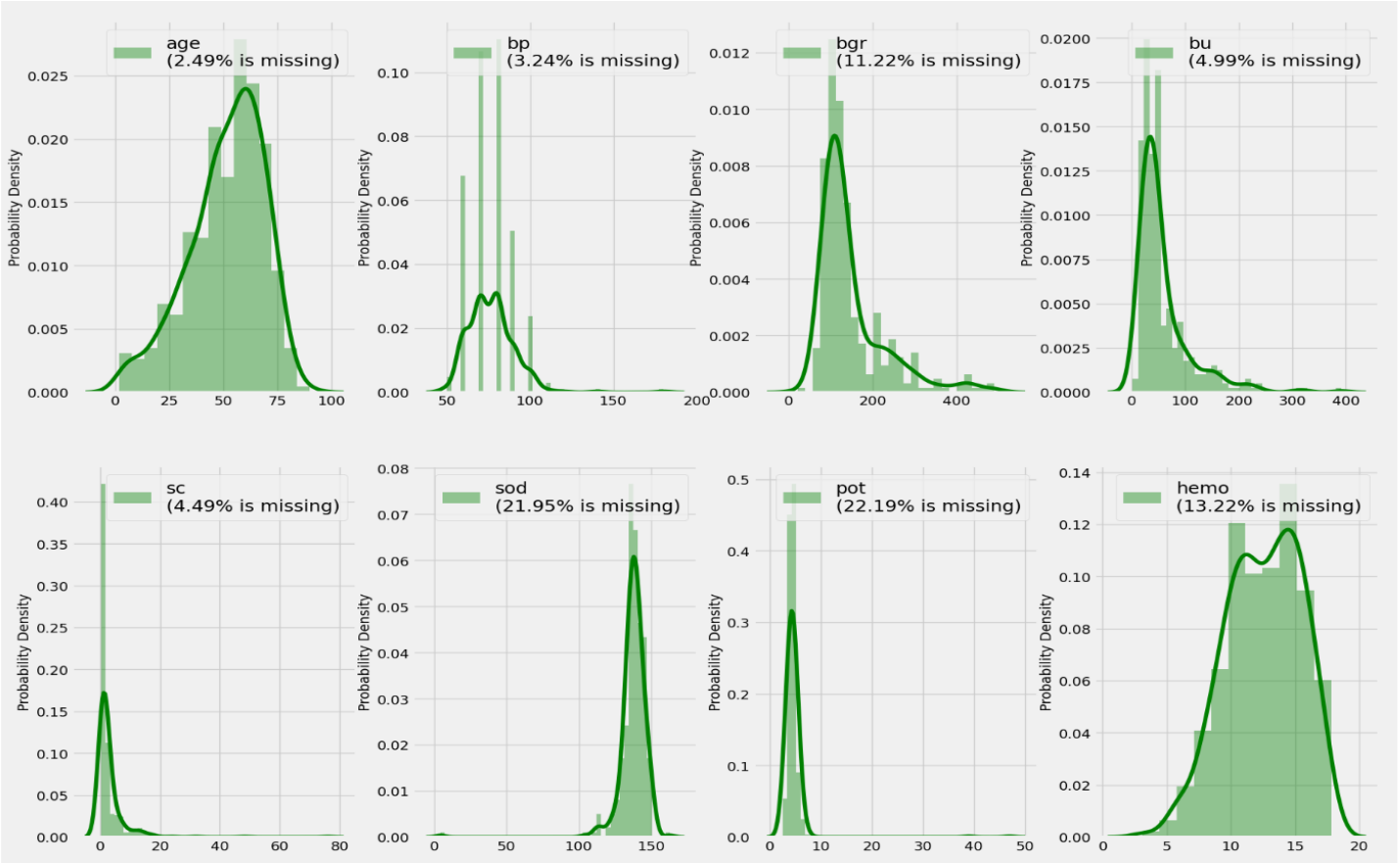
Introduction:

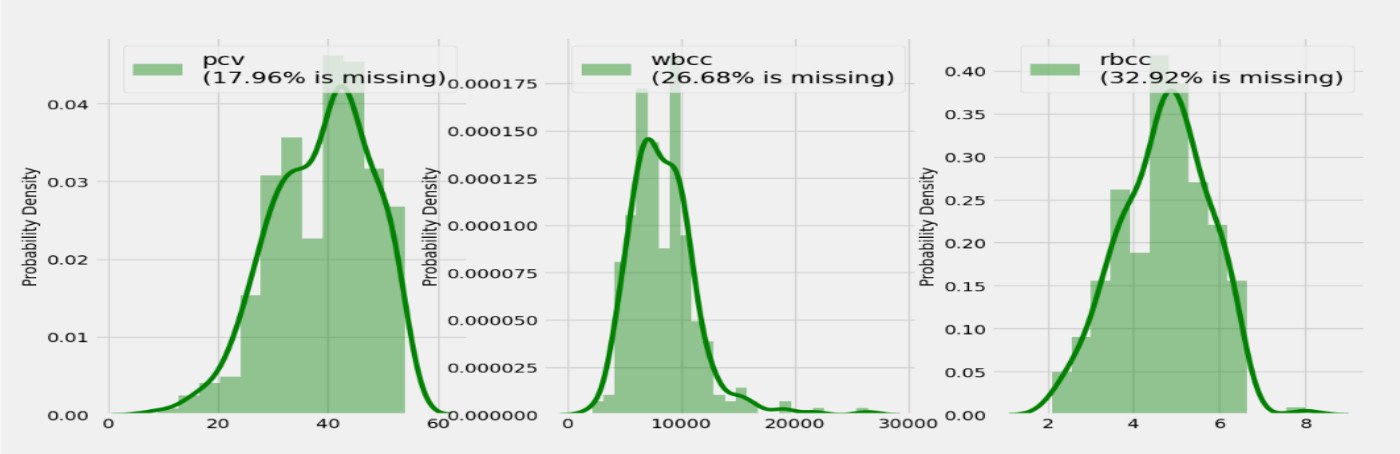
Data preparation (also referred to as “data preprocessing”) is the process of transforming raw data so that data scientists and analysts can run it through machine learning algorithms to uncover insights or make predictions. In this phase, we will be preparing the final data set for modeling.

1. Data Visualization

a. Numeric Features

❖ **The distribution of numerical features**





- Some features are very skewed while others are almost normal
- Some features show some very distant outliers.

❖ Skewness: asymmetry indicator

```
#Skewness of each numerical feature
for col in numerical_features:
    print(f"{col} has {df[col].skew()} values\n")
```

```
age has -0.6682594691593559 values

bp has 1.6054289569770592 values

bgr has 2.010773172514955 values

bu has 2.6343744585903863 values

sc has 7.509538252140634 values

sod has -6.996568560937047 values

pot has 11.582955561754394 values

hemo has -0.3350946791593011 values

pcv has -0.4336785974434392 values

wbcc has 1.621589371911243 values

rbcc has -0.1833293207517324 values
```

⇒High skewness which is an indicator of the asymmetry in the distribution of the data.

- sc and pot present a positive skewness . They are skewed to the right.
- sod presents a negative skewness . They are skewed to the left.
- age shows a moderate negative skewness.
- hemo ,pcv,rbcc : Their skewness is near zero so they show symmetry of the data.

❖ Kurtosis : outlier indicator

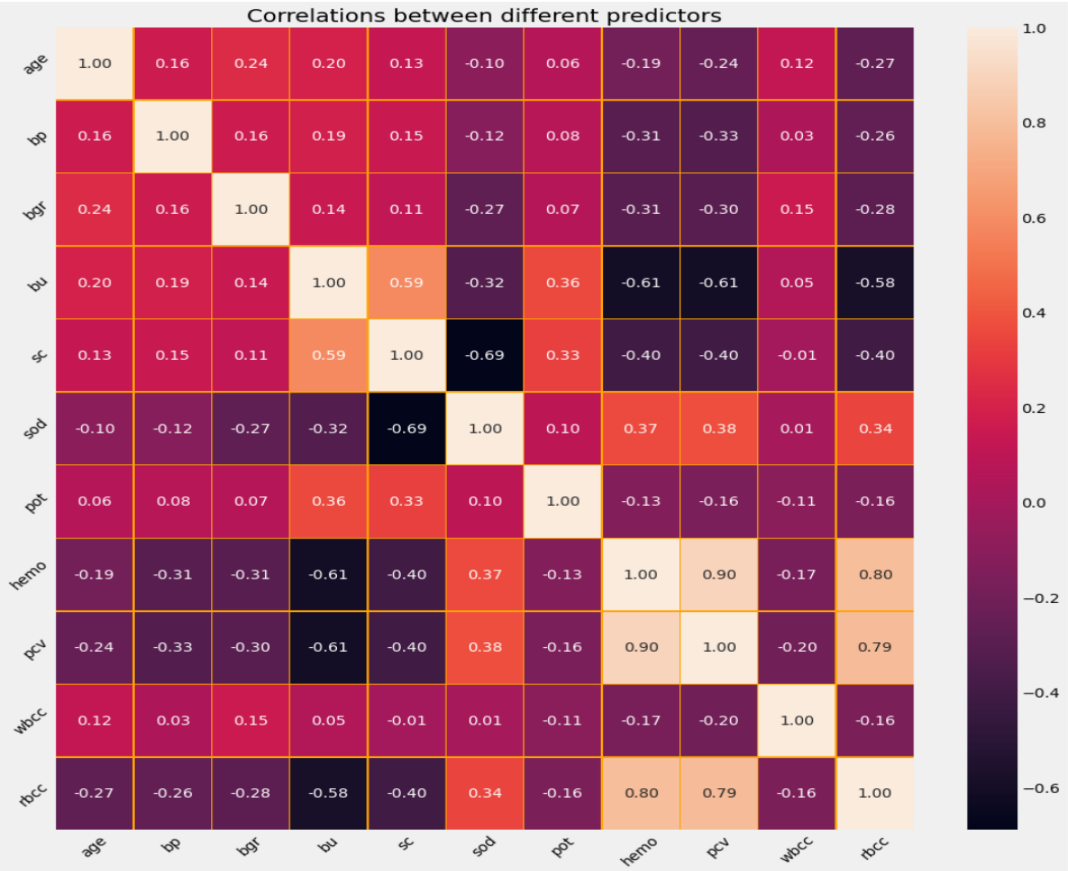
```
#kurtosis of each numerical values
for col in numerical_features:
    print(f"{col} has {df[col].kurt()} values\n")
```

```
age has 0.0578404946021962 values

bp has 8.646095188994593 values

bgr has 4.225593588175688 values

bu has 9.34528857643826 values

sc has 79.30434545204058 values

sod has 85.53436961995402 values

pot has 142.5059115472415 values

hemo has -0.47139804366030624 values

pcv has -0.3205616837789842 values

wbcc has 6.150639815202153 values

rbcc has -0.27004248129474906 values
```

⇒kurtosises : is an indicator of the presence of outliers in these columns.

sod,pot and sc have a significant presence of outliers.

❖ The Correlation :



➔ **Positive Correlation**

- hemoglobin -> red_blood_cell_count,packed_cell_volume, specific_gravity
- red_blood_cell_count -> packed_cell_volume,specific_gravity
- specific_gravity -> packed_cell_volume
- blood_glucose_random -> sugar
- serum_creatinine -> blood_urea

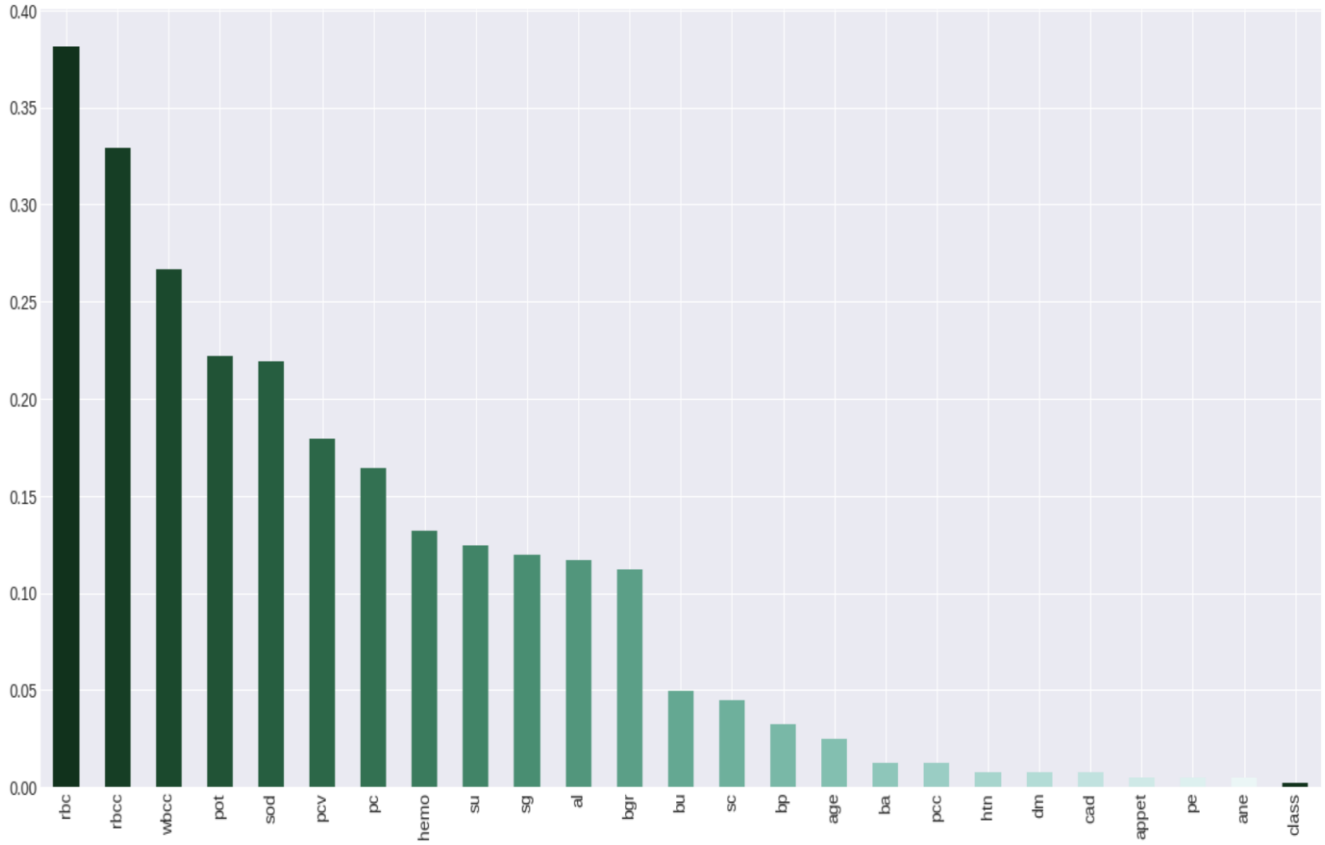
➔ **Negative correlation**

- Albumin -> hemoglobin, packed_cell_volume,specific_gravity,red_blood_cell_count
- serum_creatinine -> sodium
- blood_urea -> hemoglobin, packed_cell_volume,red_blood_cell_count

b. Categorical Features

- Some features have very high percentages of missing values while some have almost none.
- Imbalanced data : the target class has an uneven distribution of observations, CKDhas a very high number of observations (250 patients) and NotCKD has a very low number of observations (150 patients).

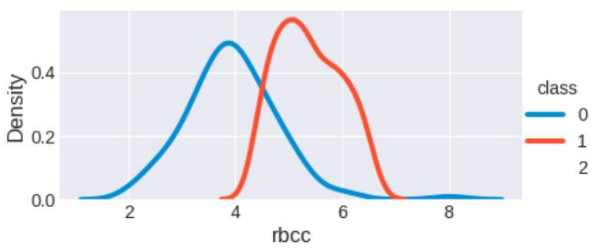
c. Proportions of missing values



- rbc , rbcc, wbcc Have the highest percentage of missing values.
- The target variable class has some missing values.
- We worked with multiple imputation methods : mean,mode,median,knn imputer,iterative imputer ,imputation with linear and logistic regression , with decision trees in order to preserve as much information as possible.

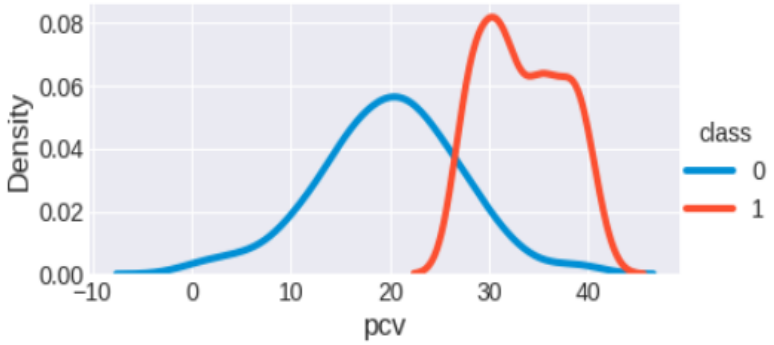
d. Variation of each feature VS target

RBCC, PCV, Hemoglobin



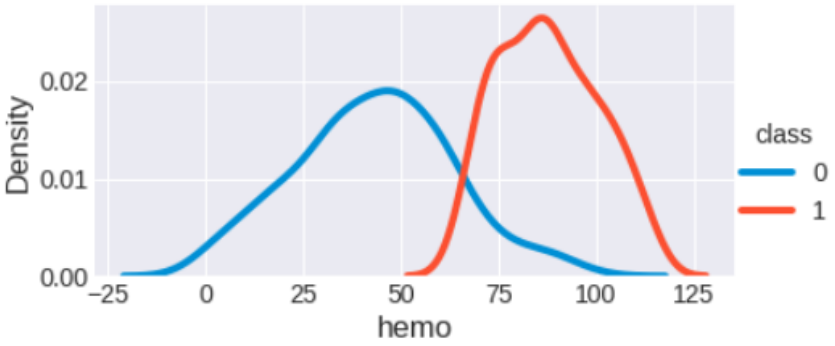
class	mean	median
0	17.357143	17.0
1	31.790210	31.0

- Both distributions are very different
- CKD and not-CKD follow a normal distribution as the mean and median are almost equal.
- A person having RBCC count range <15 is mostly classified as positive for chronic kidney disease.
- A person having RBCC count range >15 is classified as negative for chronic kidney disease.



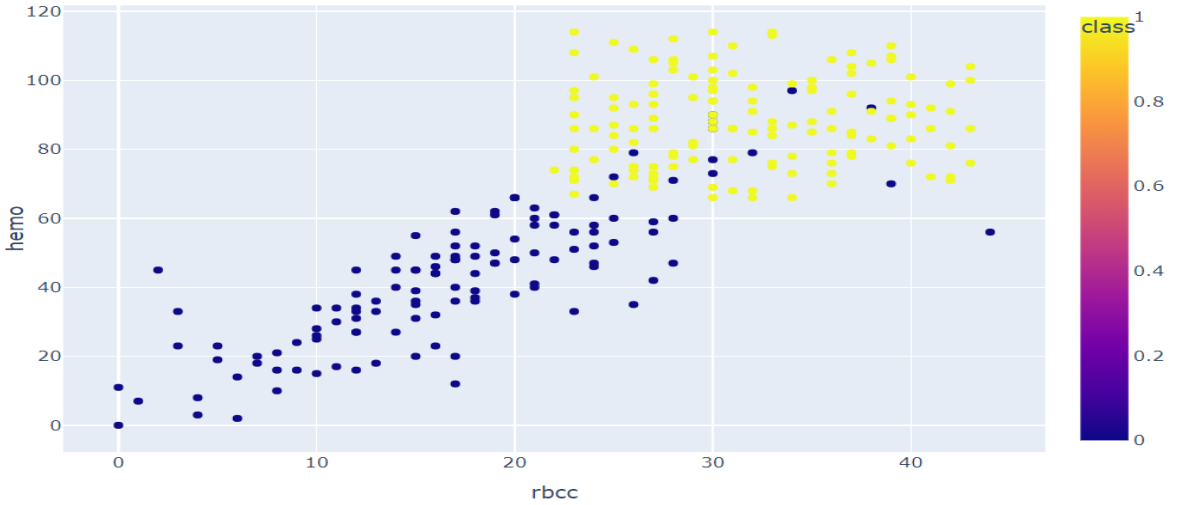
class	mean	median
0	19.961749	20.0
1	33.335616	33.0

- Both distributions are quite different
- For both distribution the mean is equal the the median : we have a normal distribution
- Person having packed_cell_volume <20 are mostly classified as positive for chronic kidney disease.
- Person having packed_cell_volume >20 are classified as negative for chronic kidney disease.

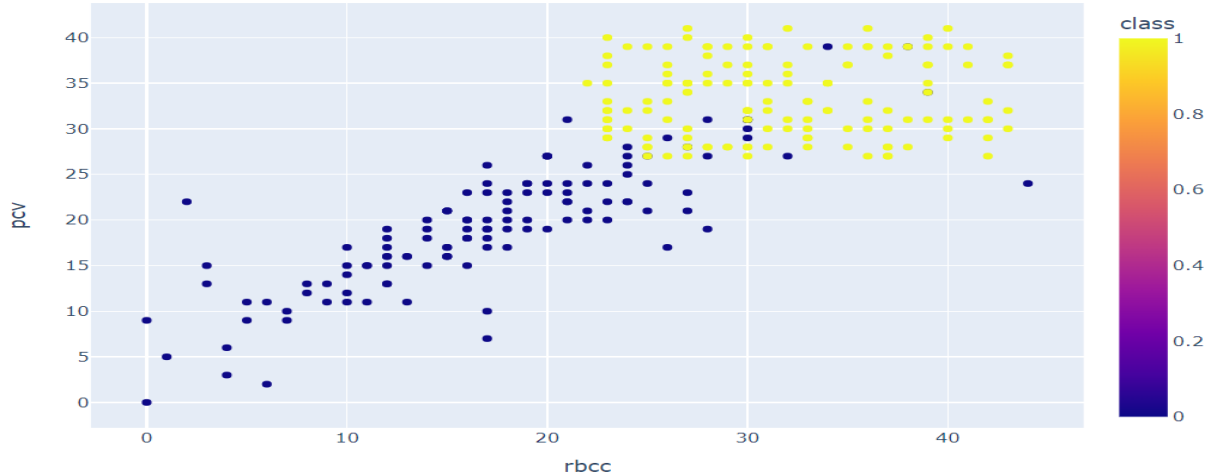


class	mean	median
0	19.961749	20.0
1	33.335616	33.0

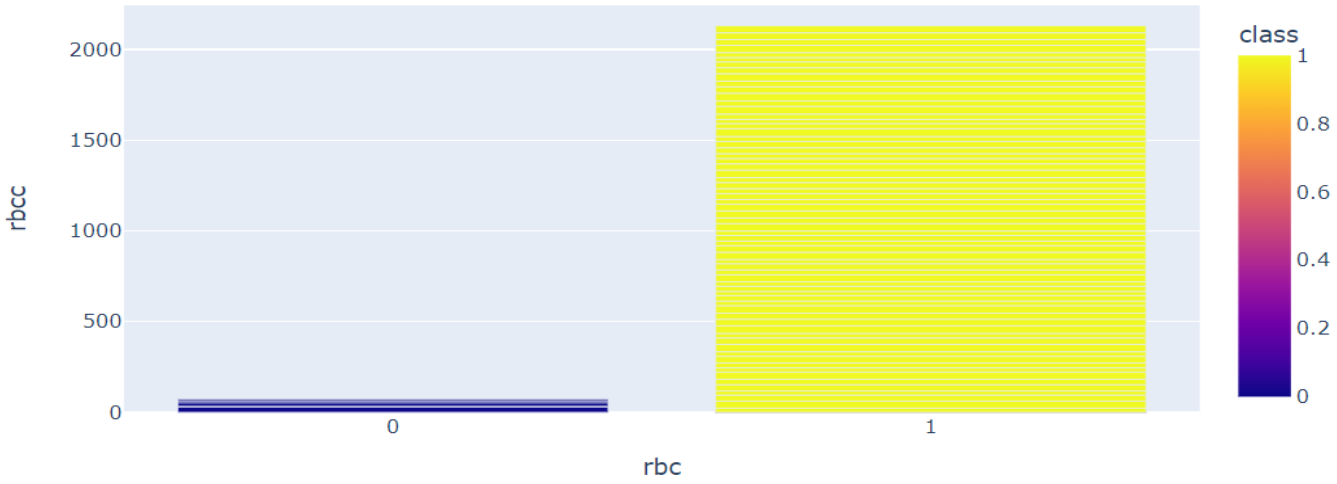
- Both distributions cvd and notckd are quite different
- For both distribution the mean is equal the the median : we have a normal distribution
- People having hemoglobin level <50 are mostly classified as positive for chronic kidney disease.
- People having hemoglobin level >50 are classified as negative for chronic kidney disease.



- There is a linear correlation between the red blood cell count and the hemoglobin level.
- As red_blood_cell_count increases hemoglobin(high positive correlation(0.8))
- People having RBCC count range ~0 to <23 and Hemoglobin between to <66 are mostly classified as positive for chronic kidney disease.
- People having RBCC count range >23 and Hemoglobin between >66 are classified as negative for chronic kidney disease.
- There are few cases where even a person having a normal range of RBC count and hemoglobin still have chronic kidney disease because of other factors will check this later.



- There is a linear correlation between the red blood cell count and the packed cell volume.
- As red_blood_cell_count increases packed_cell_volume increases (high positive correlation(0.79))
- People having RBCC count range ~0 to <23 and packed_cell_volume between to <27 are mostly classified as positive for chronic kidney disease.
- People having RBCC count range >23 and packed_cell_volume between >27 are classified as negative for chronic kidney disease.
- There are few cases where even a person having a normal range of RBCC count and packed_cell_volume still has chronic kidney disease because of other factors will check this later.

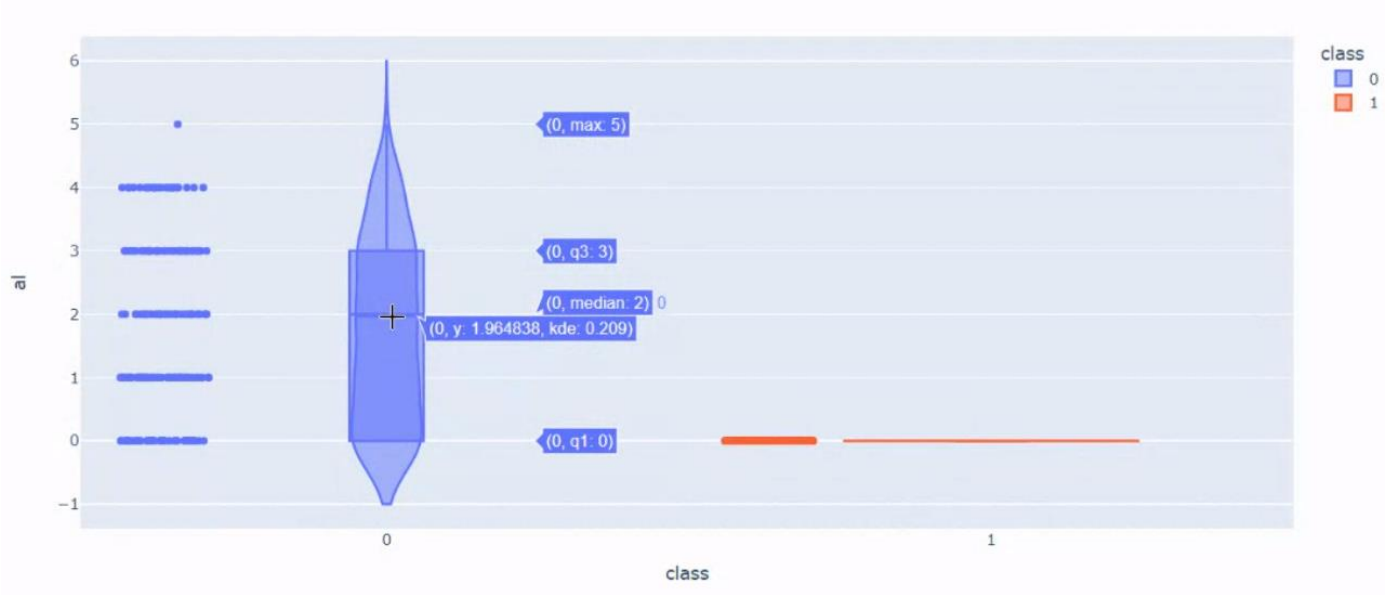


Conclusion

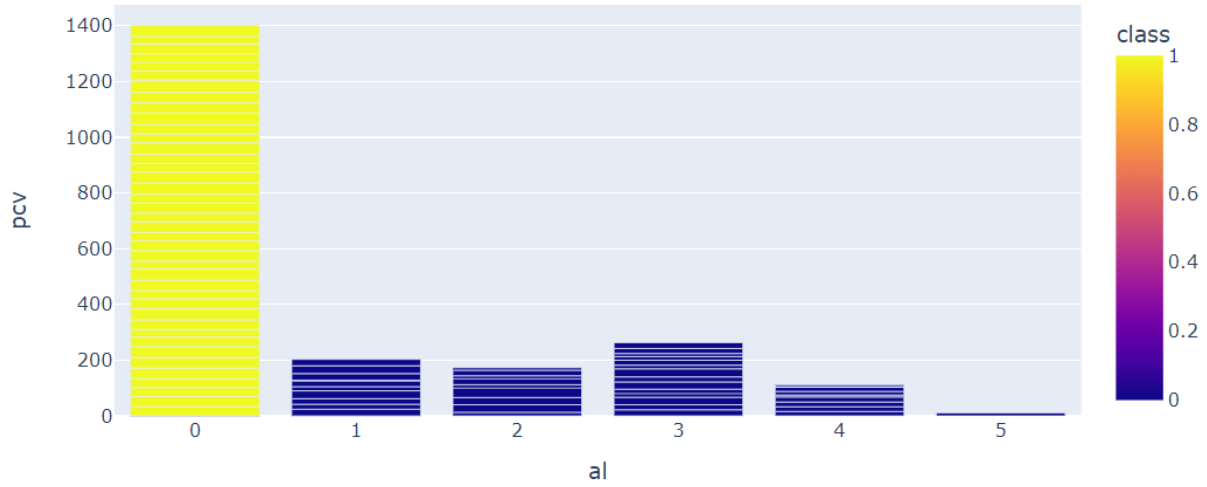
Also we have seen through plots that those having

- A normal range of hemoglobin is mostly non-ckd.
- All who fall in abnormal red_blood_cells level are suffering from chronic kidney disease.
- for those whose red_blood_cells are normal but having high and low red_blood_cell_counts are prone to have CKD

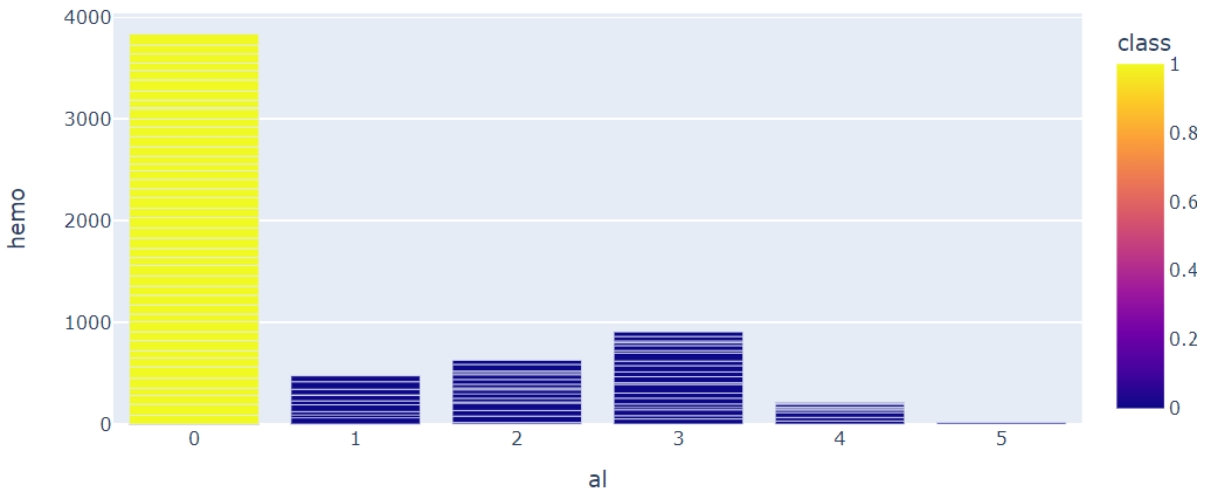
Albumin :



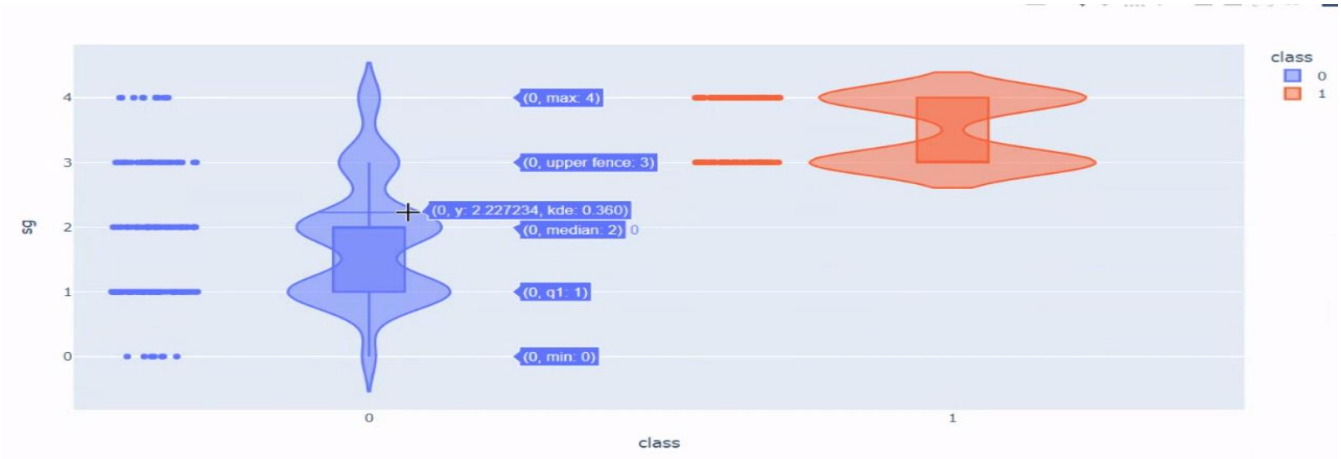
- Above Level 0 for albumin is symptoms of CKD.
- Level 0 of albumin do not have CKD.



- People who have (<23) packed cell volume and (>0) levels of albumin suffer from chronic kidney disease.

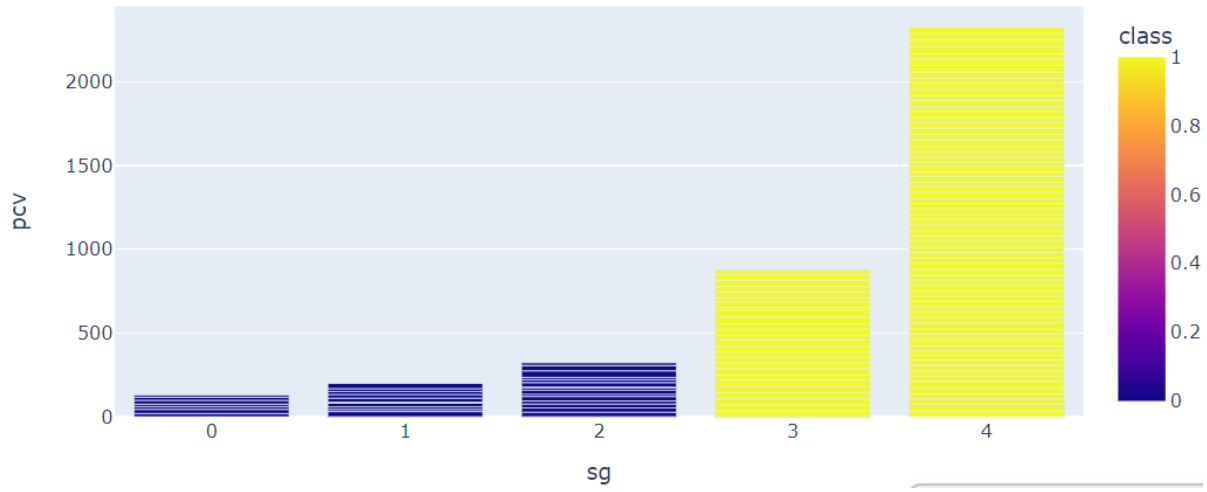


- People who have (<50) hemoglobin and (>0) levels of albumin suffer from chronic kidney disease.

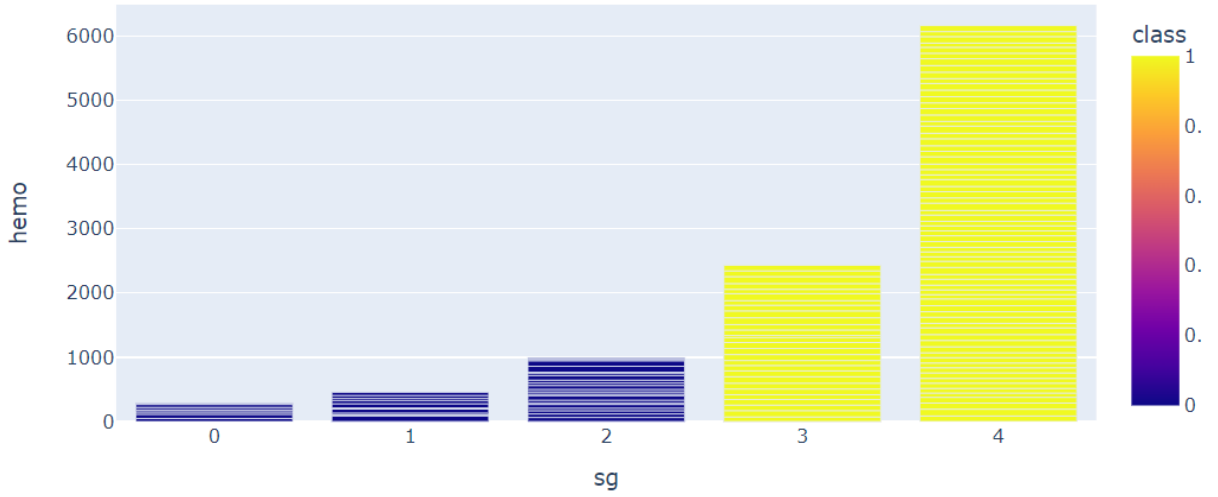


- Patients whose specific gravity values are 1.020 or 1.025 do not have chronic kidney disease.

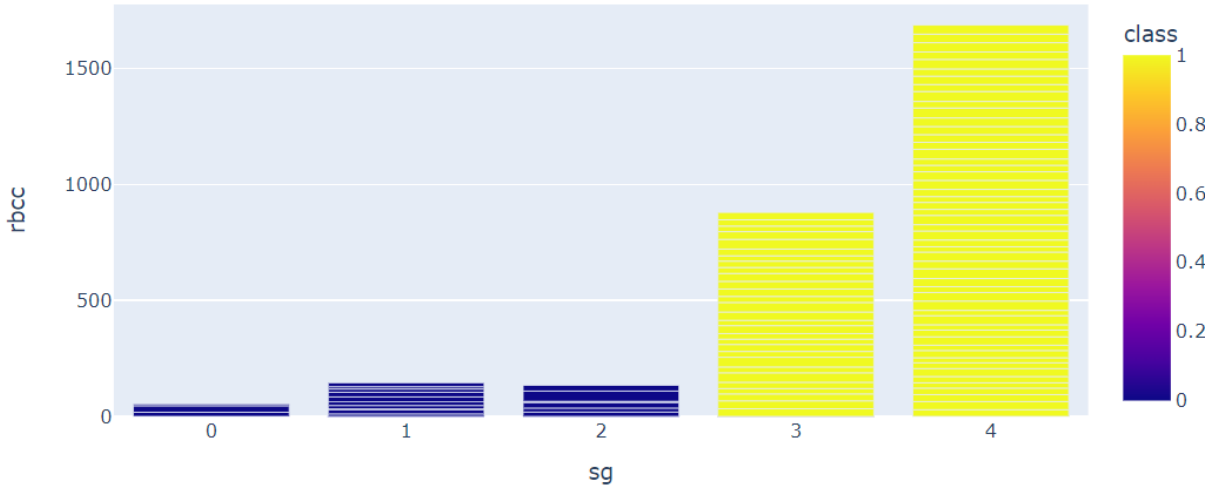
- Patients whose specific gravity values are 1.005,1.010 or 1.015 do have chronic kidney disease.



- Patients who have packed cell volume<40 and specific gravity <1.02 have chronic kidney disease.
- Patients who have packed cell volume>40 and specific gravity >1.02 do not have chronic kidney disease.

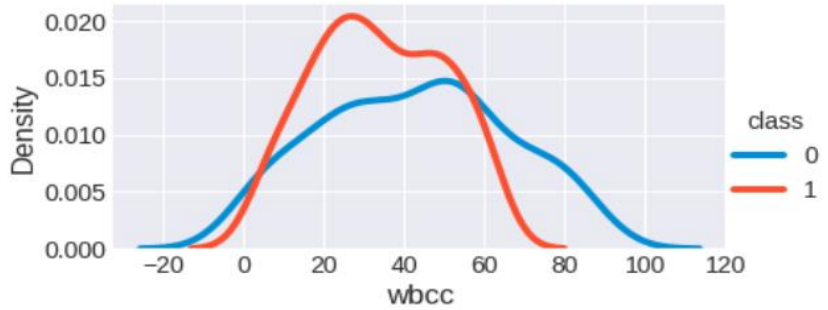


- Chances of having CKD is high if a person has a specific gravity level 1.005,1.01,1.015 and hemoglobin below normal range i.e <70.
- higher the specific_gravity and hemoglobin lesser the chances of having CKD



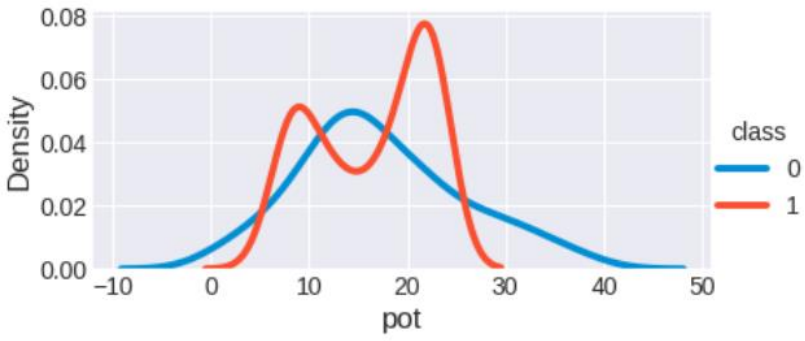
- Chances of having CKD is high if a person has a specific gravity level 1.005,1.01,1.015 and red blood cell count below normal range i.e <30.
- higher the specific_gravity and normal count of red blood cell count mean reduced chances of having CKD

White blood cell count :

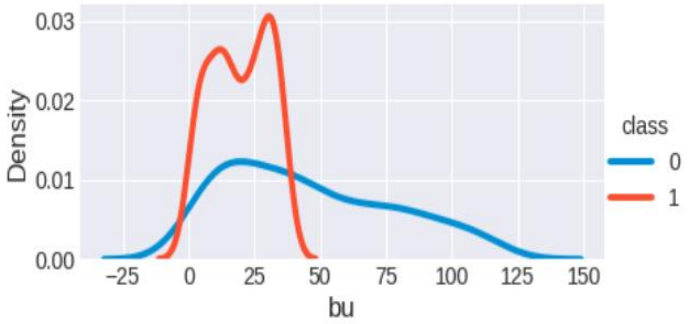


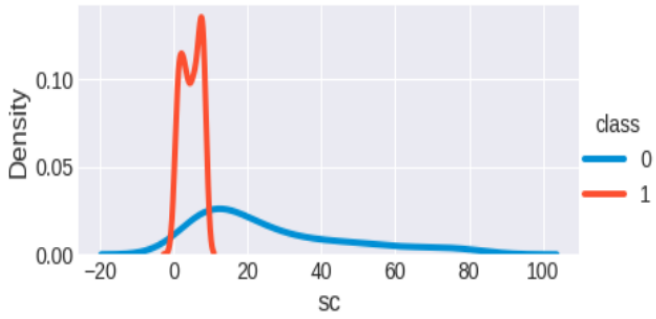
- Those having a normal range of white_blood_cell_count they are also having chronic diseases

Potassium, blood urea and serum_creatinine

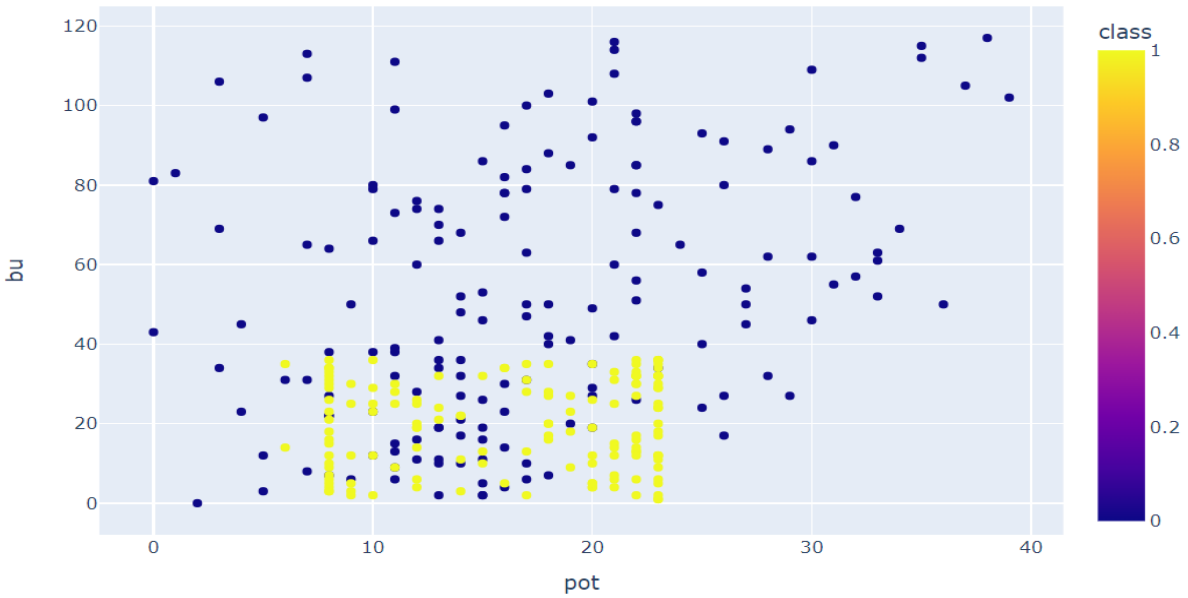


- Your body uses the potassium it needs. The extra potassium that your body does not need is removed from your blood by your kidneys. When you have kidney disease, your kidneys cannot remove extra potassium in the right way, and too much potassium can stay in your blood.
- From the above two scatter plots we can see that those people who are suffering from chronic kidney disease contain higher potassium since their kidney is unable to remove extra potassium.
- We have also observed that people are still suffering through chronic disease even if they have a normal range of potassium.



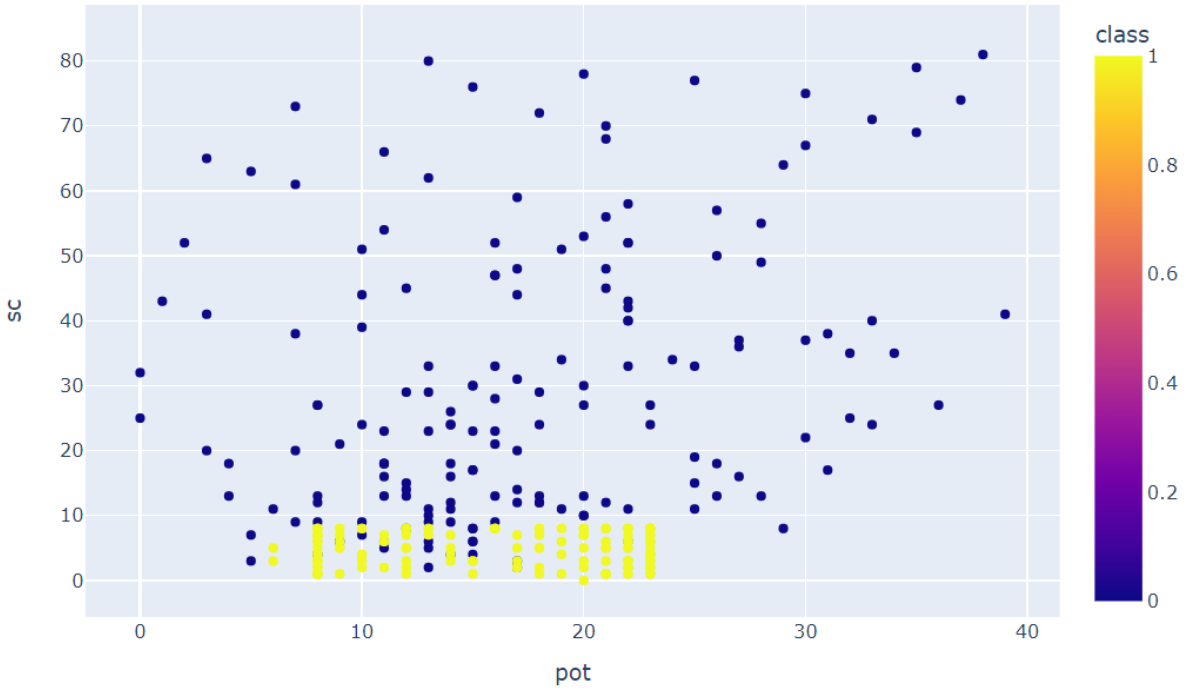


- From the correlation plot, we can see that potassium has a positive relation with blood_urea (0.36) and serum_creatinine(0.33).
- Let's check if it helps to find any pattern or certain behavior.



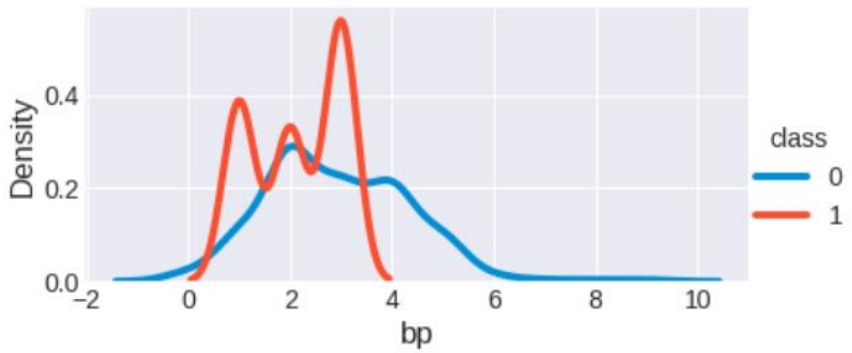
- People have blood urea in the range 9 to 23 and potassium between 8 and 23 are majorly classified as not-CKD.

- Blood urea more than 50 and potassium higher than 25 are classified as CKD patient
- There are few cases where a person having potassium and blood urea within range still suffers from CKD.



- People have Serum creatinine in the range 1 to 8 and potassium between 6 and 23 are majorly classified as a not-CKD.
- Serum creatinine more than 9 and potassium higher than 25 are classified as CKD patient
- There are few cases where a person having potassium and Serum creatinine within range still suffers from CKD.

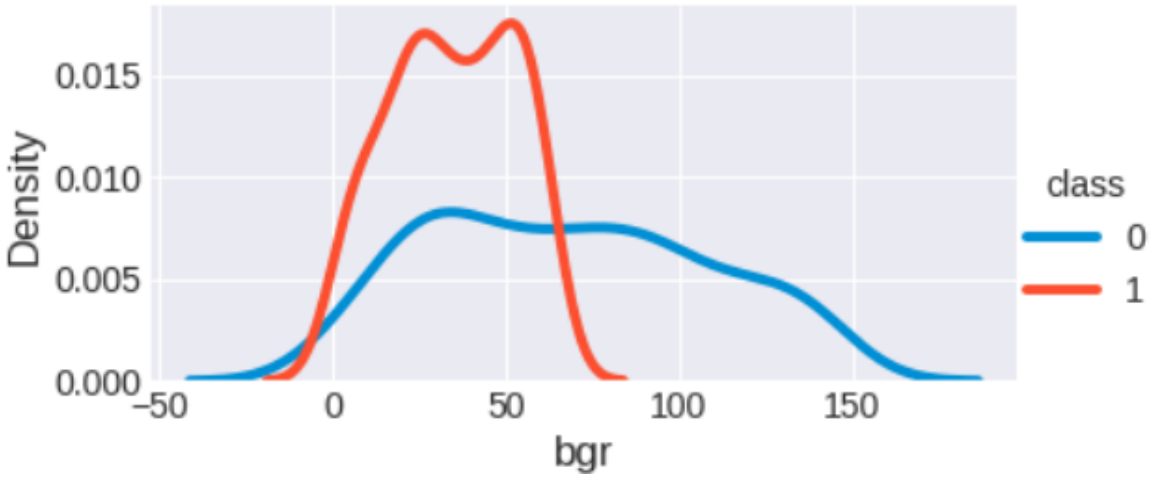
Blood pressure



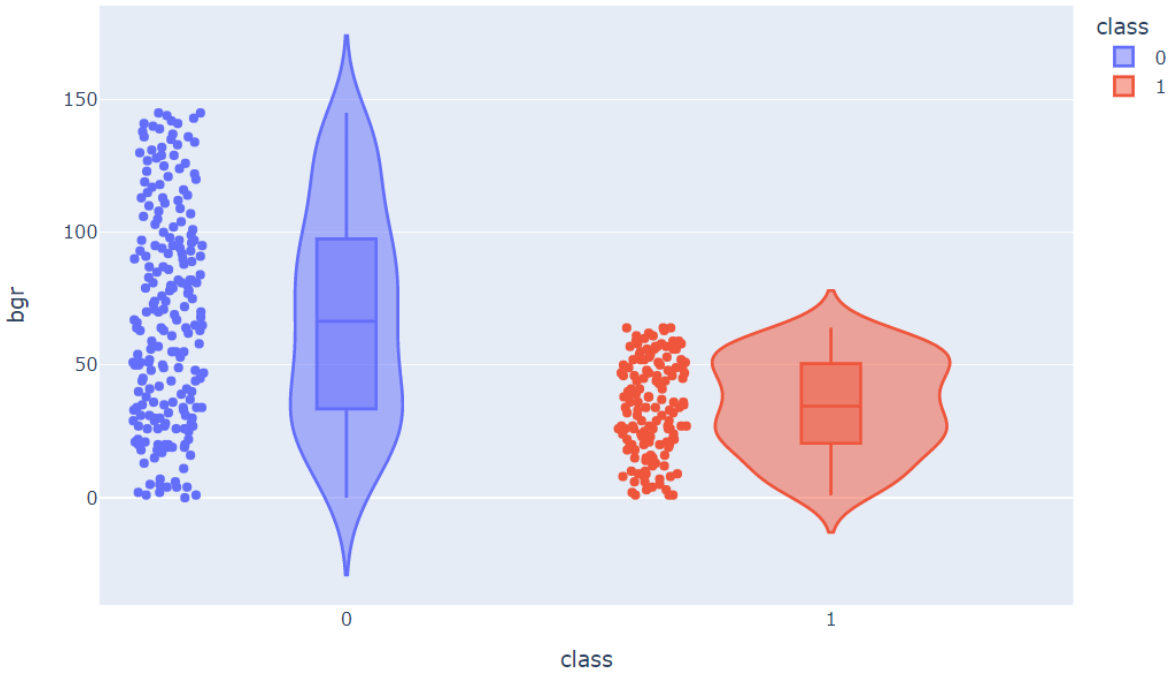
- People who suffer from chronic kidney disease tend to have high blood pressure.

- There are few cases where a person has a high blood pressure and still they are suffering from CKD.

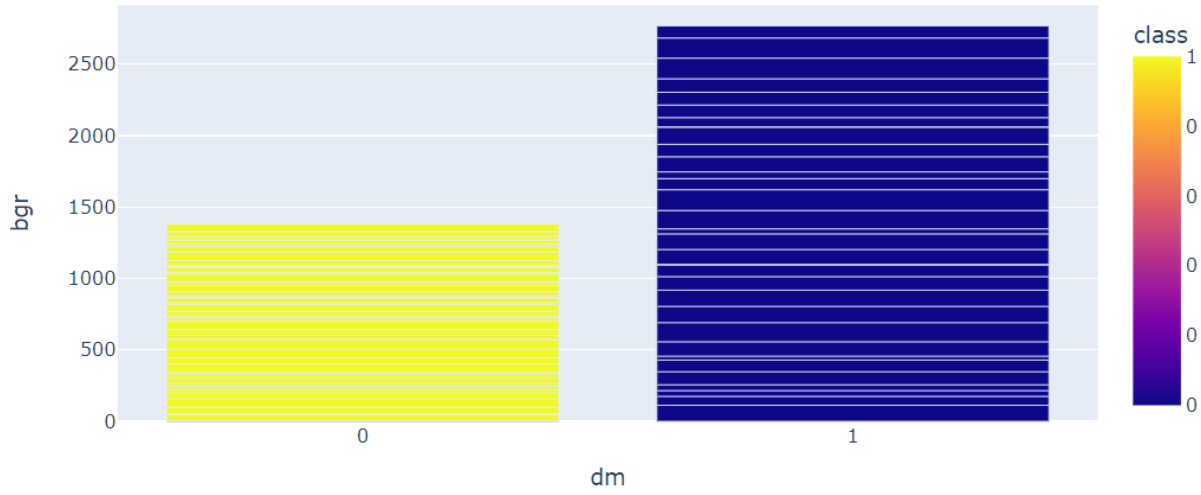
blood glucose random, sugar, diabetes_mellitus



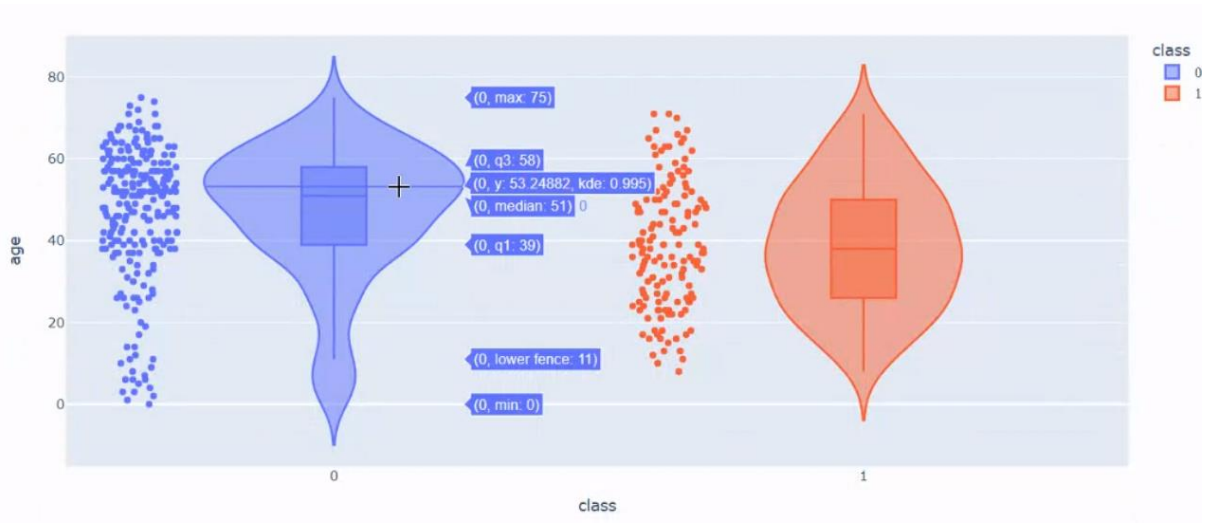
- People who have high glucose levels suffer from Chronic kidney disease.



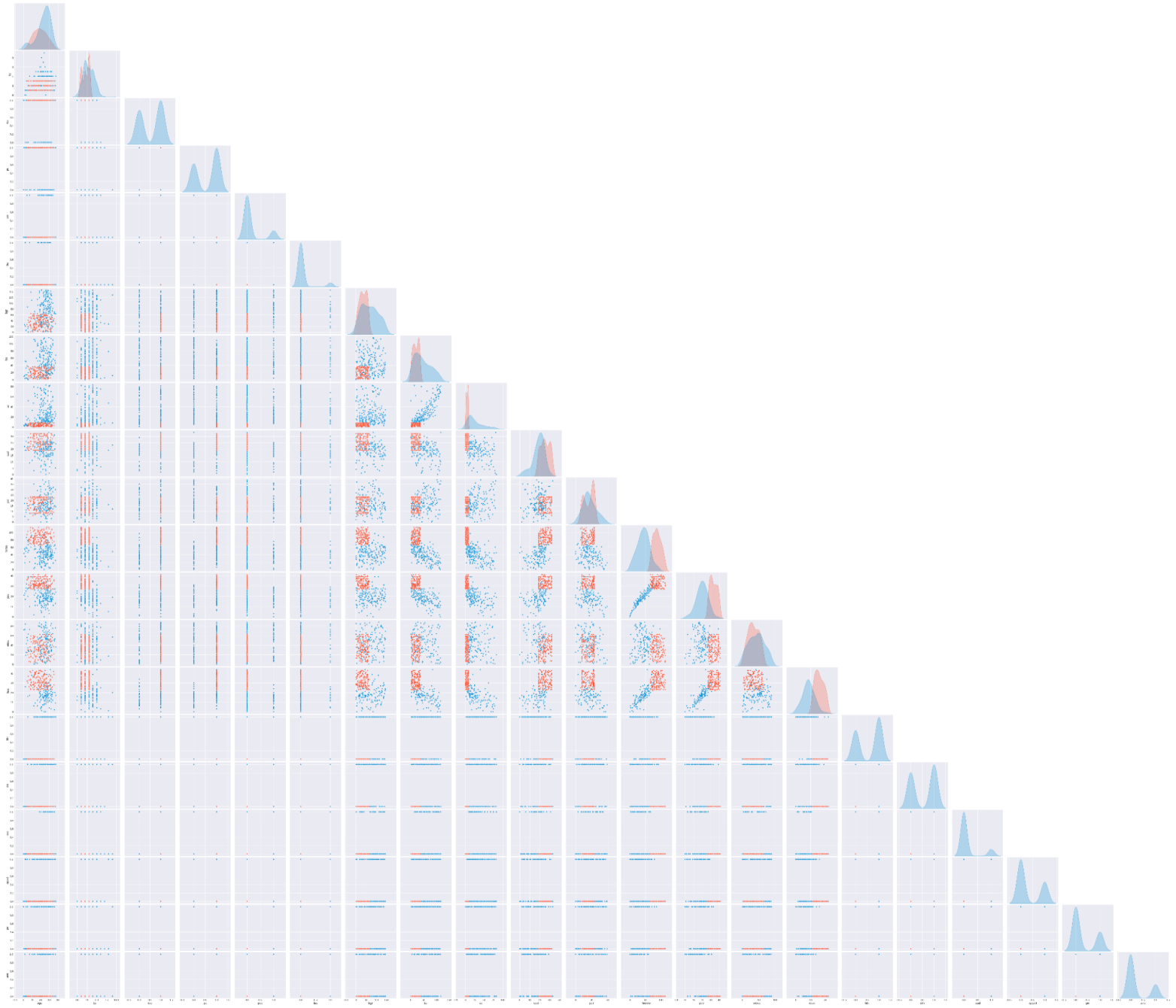
- Higher the glucose have more chances of having chronic kidney disease



- High blood sugar level causes diabetes mellitus
- Mostly the people who suffer from diabetes mellitus also suffer from chronic kidney disease.



- Age is between 2 to 90, that is even 2 years old and 90 year olds suffering from chronic kidney diseases, so through visualization we can say that there is no such specific age group that is suffering from CKd.



Conclusion

- RBC count is highly correlated with hemoglobin and packed_volume_area
- When red blood cell level is abnormal then the person is most likely to have CKD.
- When RBC count high or low than a normal range than person likely to have CKD
- white_blood_cell, sodium, and potassium, blood_press no such relation with other variables
- if a person is having less specific gravity(levels 1.005,1.01,1.015) with lesser hemoglobin(<13) and less packed cell volume(<40) , they have higher chances of having CKD

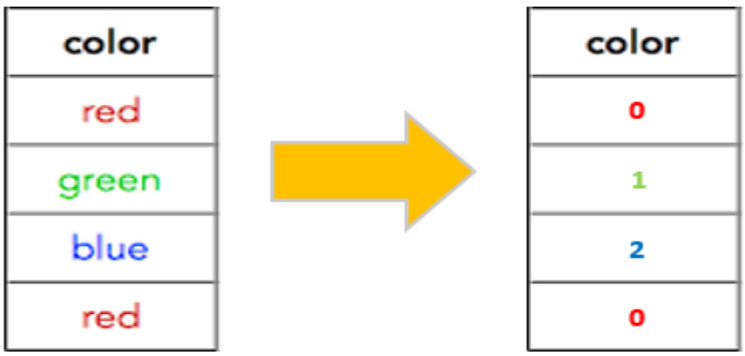
- Albumin and Hemoglobin have a negative correlation(correlation matrix) * * Mostly albumin level above 0 and hemoglobin higher than normal range is an indication CKD
- If blood urea level is higher than 150 than there are higher chances of having a chronic kidney disease
- Higher the serum creatinine level i.e >1.2, people likely to have chronic kidney diseases.
- People who have blood pressure <60 to >80 are prone to have a chronic disease.
- People with a high range of blood glucose random, high levels of sugar also suffer from diabetes_mellitus are majorly classified as chronic kidney disease.
- Age has no such correlation with other variables red_blood_cell - red_blood_cell_count
- From the above analysis, we can see that presence of even one - abnormal red cell count, bacteria, hypertension, pus cells, diabetes, coronary disease, lack of appetite, anemia , increase the chances of occurrence of CKD to substantial levels.

2. Categorical feature encoding

We start the data preprocessing with feature encoding which is one of the most crucial steps in any machine learning project.

Feature encoding is the process of turning categorical data in a dataset into numerical data. It is essential that we perform feature encoding because most machine learning models can only interpret numerical data and not data in text form.

Label Encoding would be perfect here as most categorical features have only 2 values, and they do not have an order or rank which means we won't be increasing dimensions.
It is a popular encoding technique for handling categorical variables.
In this technique, each label is assigned a unique integer based on alphabetical ordering.



```
[ ]  from sklearn.preprocessing import LabelEncoder
     df= df.apply(lambda series: pd.Series(
         LabelEncoder().fit_transform(series[series.notnull()]),
         index=series[series.notnull()].index
     ))
```

Before :

```
df.head()
```

	age	bp	sg	al	su	rbc	pc	pcc	ba	bgr	...	pcv	wbcc	rbcc	htn	dm	cad	appet	pe	ane	class
0	48.0	80.0	1.020	1	0	NaN	normal	notpresent	notpresent	121.0	...	44.0	7800.0	5.2	yes	yes	no	good	no	no	ckd
1	7.0	50.0	1.020	4	0	NaN	normal	notpresent	notpresent	NaN	...	38.0	6000.0	NaN	no	no	no	good	no	no	ckd
2	62.0	80.0	1.010	2	3	normal	normal	notpresent	notpresent	423.0	...	31.0	7500.0	NaN	no	yes	no	poor	no	yes	ckd
3	48.0	70.0	1.005	4	0	normal	abnormal	present	notpresent	117.0	...	32.0	6700.0	3.9	yes	no	no	poor	yes	yes	ckd
4	51.0	80.0	1.010	2	0	normal	normal	notpresent	notpresent	106.0	...	35.0	7300.0	4.6	no	no	no	good	no	no	ckd

5 rows × 25 columns

After :

```
df.head()
```

	age	bp	sg	al	su	rbc	pc	pcc	ba	bgr	...	pcv	wbcc	rbcc	htn	dm	cad	appet	pe	ane	class
0	40.0	3.0	3.0	1.0	0.0	NaN	1.0	0.0	0.0	48.0	...	31.0	35.0	30.0	1.0	1.0	0.0	0.0	0.0	0.0	0
1	5.0	0.0	3.0	4.0	0.0	NaN	1.0	0.0	0.0	NaN	...	25.0	19.0	NaN	0.0	0.0	0.0	0.0	0.0	0.0	0
2	54.0	3.0	1.0	2.0	3.0	1.0	1.0	0.0	0.0	140.0	...	18.0	33.0	NaN	0.0	1.0	0.0	1.0	0.0	1.0	0
3	40.0	2.0	0.0	4.0	0.0	1.0	0.0	1.0	0.0	44.0	...	19.0	25.0	17.0	1.0	0.0	0.0	1.0	1.0	1.0	0
4	43.0	3.0	1.0	2.0	0.0	1.0	1.0	0.0	0.0	33.0	...	22.0	31.0	24.0	0.0	0.0	0.0	0.0	0.0	0.0	0

5 rows × 25 columns

3. Feature Scaling

Definition of feature scaling

- Feature scaling in machine learning is one of the most critical steps during the pre-processing of data before creating a machine learning model.
- Scaling can make a difference between a weak machine learning model and a better one.

Why do we need scaling?

- Machine learning algorithms just see numbers . if there is a vast difference in the range say few ranging in thousands and few ranging in the tens, and it makes the underlying assumption that higher ranging numbers have superiority of some sort. So these more significant numbers start playing a more decisive role while training the model.

- Another reason why feature scaling is applied is that few algorithms like Neural network gradient descent converge much faster with feature scaling than without it.

When do we need scaling?

- K-nearest neighbors (KNN) with a Euclidean distance measure.
- K-Means uses the Euclidean distance measure.
- Scaling is critical while performing Principal Component Analysis(PCA). PCA tries to get the features with maximum variance, and the variance is high for high magnitude features and skews the PCA towards high magnitude features.
- We can speed up gradient descent by scaling because it oscillates inefficiently down to the optimum when the variables are very uneven.

The most suitable scaling method is **Min-Max scaler :**

- This Scaler shrinks the data within the range of -1 to 1 if there are negative values.
- It responds well if the distribution is not gaussian but the scaler is sensitive to outliers..

Before :

```
[62] df.head()
```

	age	bp	sg	al	su	rbc	pc	pcc	ba	bgr	...	pcv	wbcc	rbcc	htn	dm	cad	appet	pe	ane	class
0	40.0	3.0	3.0	1.0	0.0	NaN	1.0	0.0	0.0	48.0	...	31.0	35.0	30.0	1.0	1.0	0.0	0.0	0.0	0.0	0
1	5.0	0.0	3.0	4.0	0.0	NaN	1.0	0.0	0.0	NaN	...	25.0	19.0	NaN	0.0	0.0	0.0	0.0	0.0	0.0	0
2	54.0	3.0	1.0	2.0	3.0	1.0	1.0	0.0	0.0	140.0	...	18.0	33.0	NaN	0.0	1.0	0.0	1.0	0.0	1.0	0
3	40.0	2.0	0.0	4.0	0.0	1.0	0.0	1.0	0.0	44.0	...	19.0	25.0	17.0	1.0	0.0	0.0	1.0	1.0	1.0	0
4	43.0	3.0	1.0	2.0	0.0	1.0	1.0	0.0	0.0	33.0	...	22.0	31.0	24.0	0.0	0.0	0.0	0.0	0.0	0.0	0

5 rows × 25 columns

After :

```
df.head()
```

	age	bp	sg	al	su	rbc	pc	pcc	ba	bgr	...	pcv	wbcc	rbcc
0	-0.228411	0.283626	0.454071	-0.012548	-0.410106	NaN	0.541698	-0.344447	-0.242536	-0.182777	...	0.570935	-0.171861	0.492958
1	-2.362065	-2.041307	0.454071	2.208413	-0.410106	NaN	0.541698	-0.344447	-0.242536	-1.068431	...	-0.100313	-0.949948	-0.502825
2	0.625051	0.283626	-1.297699	0.727772	2.323069	0.483561	0.541698	-0.344447	-0.242536	2.311516	...	-0.883436	-0.269121	-0.854920
3	-0.228411	-0.491352	-2.173584	2.208413	-0.410106	0.483561	-1.846048	2.903200	-0.242536	-0.291224	...	-0.771561	-0.658165	-0.796402
4	-0.045526	0.283626	-1.297699	0.727772	-0.410106	0.483561	0.541698	-0.344447	-0.242536	-0.589455	...	-0.435937	-0.366382	-0.102131

5 rows × 25 columns

4. Handling the missing values

Causes of missing values :

- ❖ Data is not being intentionally filled especially if it is an optional field.
- ❖ Data being corrupted.
- ❖ Human error.
- ❖ Fraudulent behavior of intentionally deleting data.

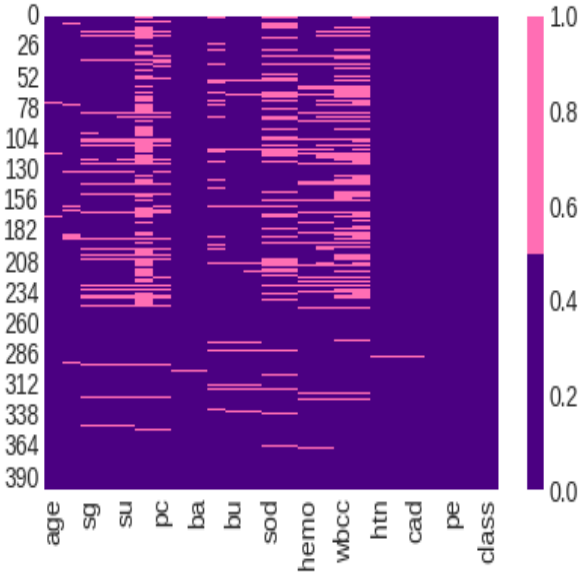
The problem of missing values :

Most machine learning algorithms are unable to handle incomplete data.

Solutions :

- ❖ Removing the columns having missing values : If you have a column with more than 80% missing values, then it is better to drop the column.
- ❖ Removing the rows having missing values : If you have a small percentage of rows with missing values, then you can just drop those rows.
- ❖ Imputing the missing values.

Percentage of missing values per column :




```
print(df.isnull().sum()*100/len(df))

age       0.00
bp        0.00
sg       11.75
al       11.50
su       12.25
rbc      38.00
pc       16.25
pcc       1.00
ba        1.00
bgr       0.00
bu        0.00
sc        0.00
sod       0.00
pot       0.00
hemo      0.00
pcv       0.00
wbcc      0.00
rbcc      0.00
htn       0.50
dm        0.50
cad       0.50
appet     0.25
pe        0.25
ane       0.25
class     0.00
dtype: float64
```

To impute the missing values in the numeric features we worked with :

- **mean** : the mean of the observed values for each variable is computed and the missing values for that variable are imputed by this mean.

- **median** : the missing values are replaced with the median value of the entire feature column.

- **KNNImputer** : identifies the neighboring points through a measure of distance and the missing values can be estimated using completed values of neighboring observations.

- **IterativeImputer with ExtraTreesRegressor as an estimator**:
  Iterative imputation refers to a process where each feature is modeled as a function of the other features, a regression problem where missing values are predicted. Each feature is imputed sequentially, one after the other, allowing prior imputed values to be used as part of a model in predicting subsequent features.

  It is iterative because this process is repeated multiple times, allowing ever improved estimates of missing values to be calculated as missing values across all features are estimated.

  Different regression algorithms can be used to estimate the missing values for each feature, so we chose Extra trees (short for extremely randomized trees) is an ensemble supervised machine learning method that uses decision trees.

  The extra trees algorithm, like the random forests algorithm, creates many decision trees, but the sampling for each tree is random, without replacement. This creates a dataset for each tree with unique samples. A specific number of features, from the total set of features, are also selected randomly for each tree. The most important and unique characteristic of extra trees is the random selection of a splitting value for a feature. Instead of calculating a locally optimal value using Gini or entropy to split the data, the algorithm randomly selects a split value. This makes the trees diversified and uncorrelated.

- **Linear regression preceded by Simple Imputer :** When we have multiple variables with missing values, we can't just directly use Regression Imputation to impute one of them as the predictors contain missing data themselves. But then, how can we impute one variable without imputing another?

  We can initially impute all the variables with missing values using some trivial methods like Simple Random Imputation (we impute the missing data with

random observed values of the variable) which is later followed by Regression Imputation of each of the variables iteratively.

To impute the missing values in the categorical features we worked with:

- **mode** : replaces missing values of a categorical variable by the most frequent value of non-missing cases of that variable.

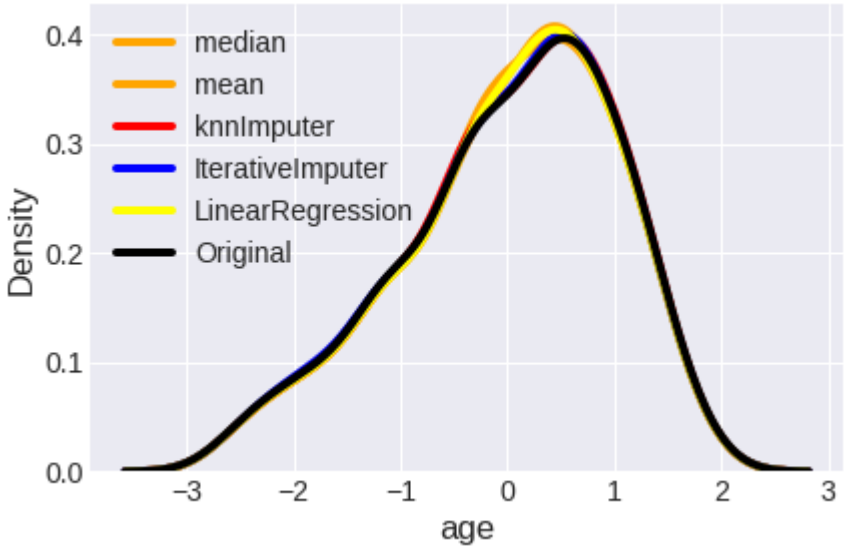
- **KNN Imputer**

- **Logistic regression** preceded by Simple Imputer : We can initially impute all the categorical variables with missing values using some trivial methods like Simple Random Imputation which is later followed by Logistic Regression Imputation of each of the variables iteratively.
  Logistic Regression is a type of statistical model (also known as *logit model*) that is often used for classification and predictive analytics. Logistic regression estimates the probability of an event occurring
  We define a training set which contains all the rows without missing values.
  and a testing set which contains all the rows with missing values.

- **Decision trees** preceded by Simple Imputer : We initially impute all the categorical variables with missing values mode imputation which is later followed by Logistic Decision trees Imputation of each of the variables.
  Decision Tree is the most powerful and popular tool for classification and prediction. A Decision tree is a flowchart-like tree structure, where each internal node denotes a test on an attribute, each branch represents an outcome of the test, and each leaf node (terminal node) holds a class label.
  We define a training set which contains all the rows without missing values.
  and a testing set which contains all the rows with missing values.

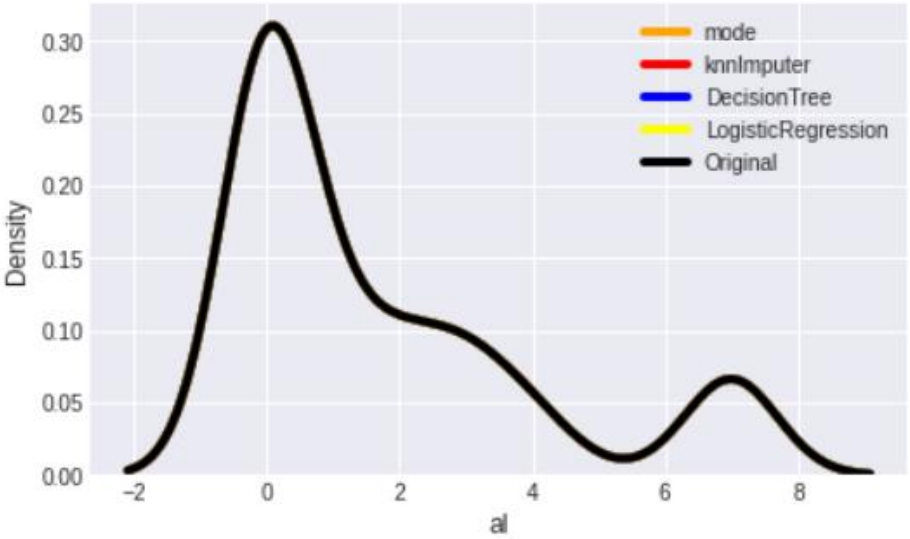
To compare between the different imputation methods we compared each distribution curve of the imputed data with the original distribution curve.
The best suited technique is the most compatible one to the original data.

Example of imputation of missing values in a numerical feature :



We have used 6 techniques to amputate the missing values and we noticed that the IterativeImputer has the best compatible distribution curve on the original distribution curve.
Therefore, it is the best suited for imputation technique for the column age.

Example of imputation of missing values in a categorical feature :



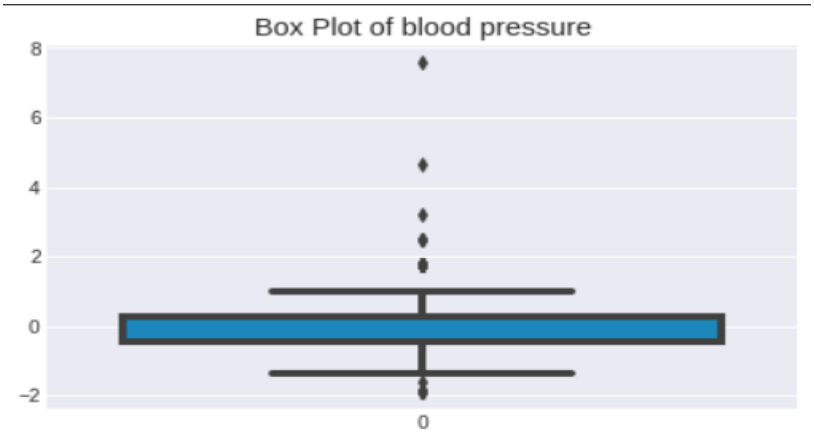
We have used 5 techniques to amputate the missing values and we noticed that All the techniques give compatible distribution curves on the original distribution curve. Therefore, we choose the technique that requires less computation (mode)

5. Outlier Handling

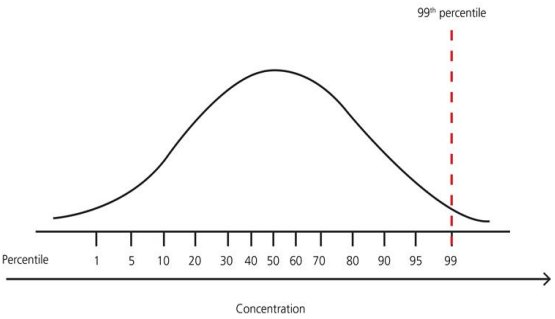
An outlier is an observation that lies an abnormal distance from other values in a random sample from a population.

To observe the outliers we used boxplots :

Box plot is a data visualization plotting function. It shows the min, max, median, first quartile, and third quartile
The outliers will be the values that are out of the (1.5*interquartile range) from the 25 or 75 percentile.



To impute the outliers we used:



- **Percentile based method**
  You can simply fix a percentile for the upper limit and lower limit.
  For Example, Data points that are far from 99% percentile and less than 1 percentile are considered an outlier.
  We can then identify outliers as those examples that fall outside of the defined lower and upper limits:

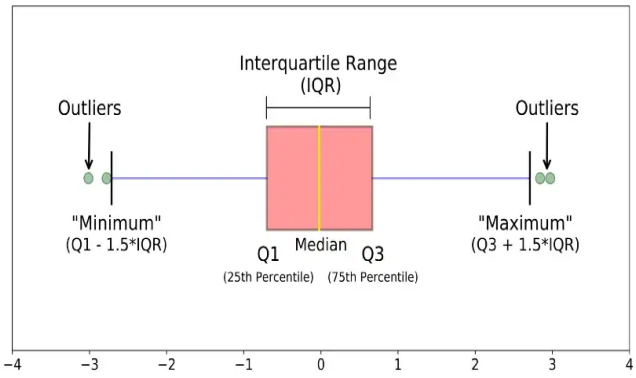
    - perct_lower = np.percentile(data[feature], level)
    - perct_upper = np.percentile(data[feature], 100 - level)

❖ **Interquartile Range Method**

Not all data is normal or normal enough to treat it as being drawn from a Gaussian distribution.

A good statistic for summarizing a non-Gaussian distribution sample of data is the Interquartile Range, or IQR for short.

The IQR is calculated as the difference between the 75th and the 25th percentiles of the data and defines the box in a box and whisker plot.



We can then identify outliers as those examples that fall outside of the defined lower and upper limits:

- iqr_lower = 25th percentile - 1.5 * iqr_range
- iqr_upper = 75th percentile + 1.5 * iqr_range

❖ **Standard Deviation Method (Empirical Rule) :**

If we know that the distribution of values in the sample is Gaussian or Gaussian-like, we can use the standard deviation of the sample as a cut-off for identifying outliers.
We can then identify outliers as those examples that fall outside of the defined lower and upper limits:

- lower_level = np.mean(data[feature]) + 3 * std
- upper_level = np.mean(data[feature]) + 3 * std

❖ **DBSCAN followed by randomForestRegression :**

Step 1 : Identifying the outliers using the clustering algorithm DBSCAN:

The DBSCAN algorithm is a density based algorithm. It looks at the density of data points in a neighborhood to decide whether they belong to the same cluster or not. If a point is too far from all other points then it is considered an outlier and is assigned a label of -1.

Step 2 : Imputing the outliers Using a supervised machine learning algorithm randomForestRegression :

We split our data into a train and test set .
Our train set contains data that was assigned by the Dbscan a label different from -1(Non Outliers)
Our test set contains data that was assigned by the Dbscan a label equal to -1(Outliers)

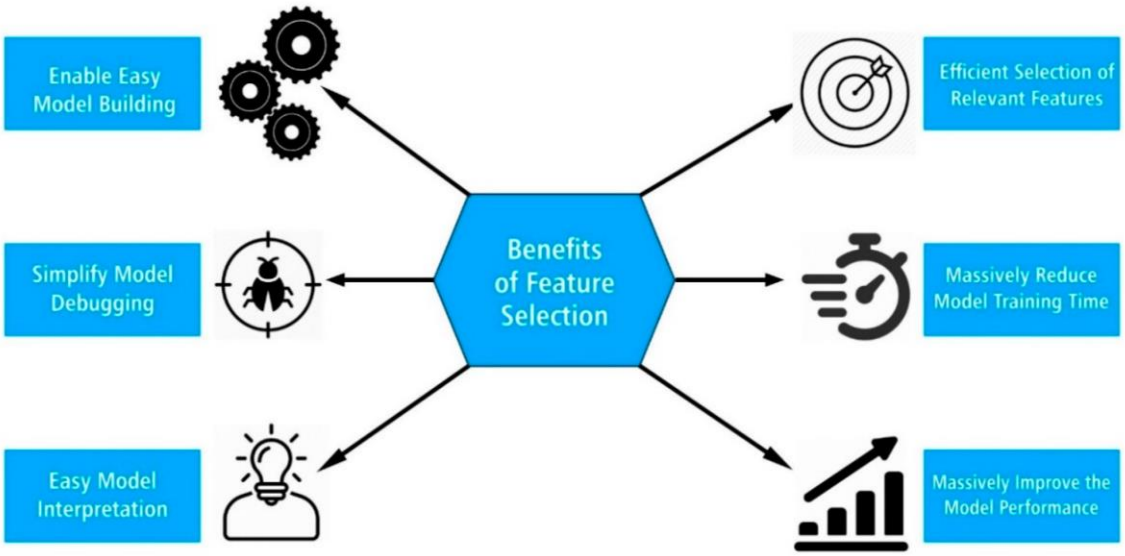
To compare between the different methods we chose the one who reduces the kurtosis the most (the kurtosis reveals the degree of presence of outliers in the distribution.

6. Feature Selection

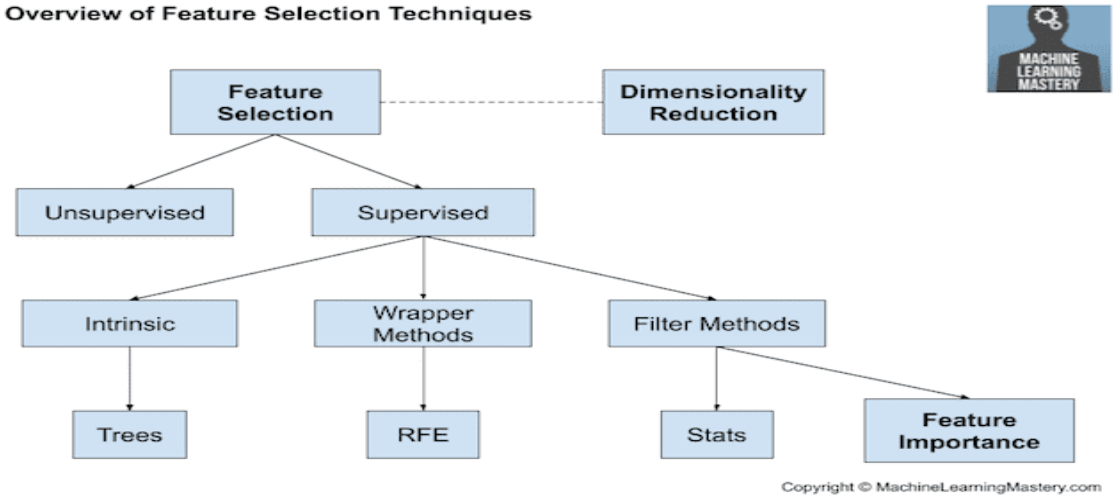
Feature selection is a process where you automatically select those features in your data that contribute most to the prediction variable or output in which you are interested.

Three benefits of performing feature selection before modeling your data are:

- **Reduces Overfitting**: Less redundant data means less opportunity to make decisions based on noise.
- **Improves Accuracy**: Less misleading data means modeling accuracy improves
- **Reduces Training Time**: Less data means that algorithms train faster.



Feature Selection methods :



- **Unsupervised**: Do not use the target variable . It removes the redundant variables.

- **Supervised**: Use the target variable. It removes the irrelevant variables.
  - **Wrapper**: Search for well-performing subsets of features.
    - RFE
  - **Filter**: Select subsets of features based on their relationship with the target.
    - Statistical Methods such a chi2
    - Correlation
  - **Intrinsic**: Algorithms that perform automatic feature selection during training.
    - Decision Trees

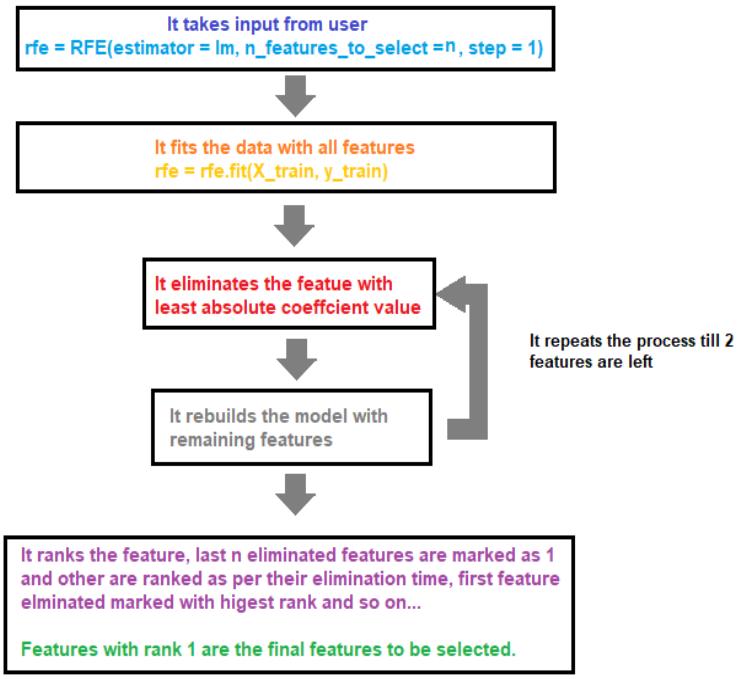
- **Supervised Methods :**
  - **Wrapper Methods :**
    - **<u>Recursive feature elimination (article 2):</u>**

      RFE is a wrapper-type feature selection algorithm. This means that a different machine learning algorithm is given and used in the core of the method, is wrapped by RFE, and used to help select features. This is in contrast to filter-based feature selections that score each feature and select those features with the largest (or smallest) score. Technically,

      RFE is a wrapper-style feature selection algorithm that also uses filter-based feature selection internally.

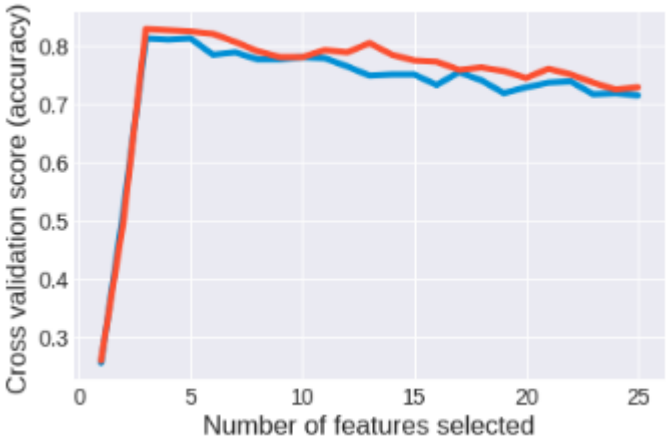
How RFE works :

RFE works by searching for a subset of features by starting with all features in the training dataset and successfully removing features until the desired number remains. This is achieved by fitting the given machine learning algorithm used in the core of the model, ranking features by importance, discarding the least important features, and re-fitting the model. This process is repeated until a specified number of features remains.



Features selected with RFECV :

RFECV (Recursive Feature Elimination with Cross-Validation) performs recursive feature elimination with a cross-validation loop to extract the optimal features.



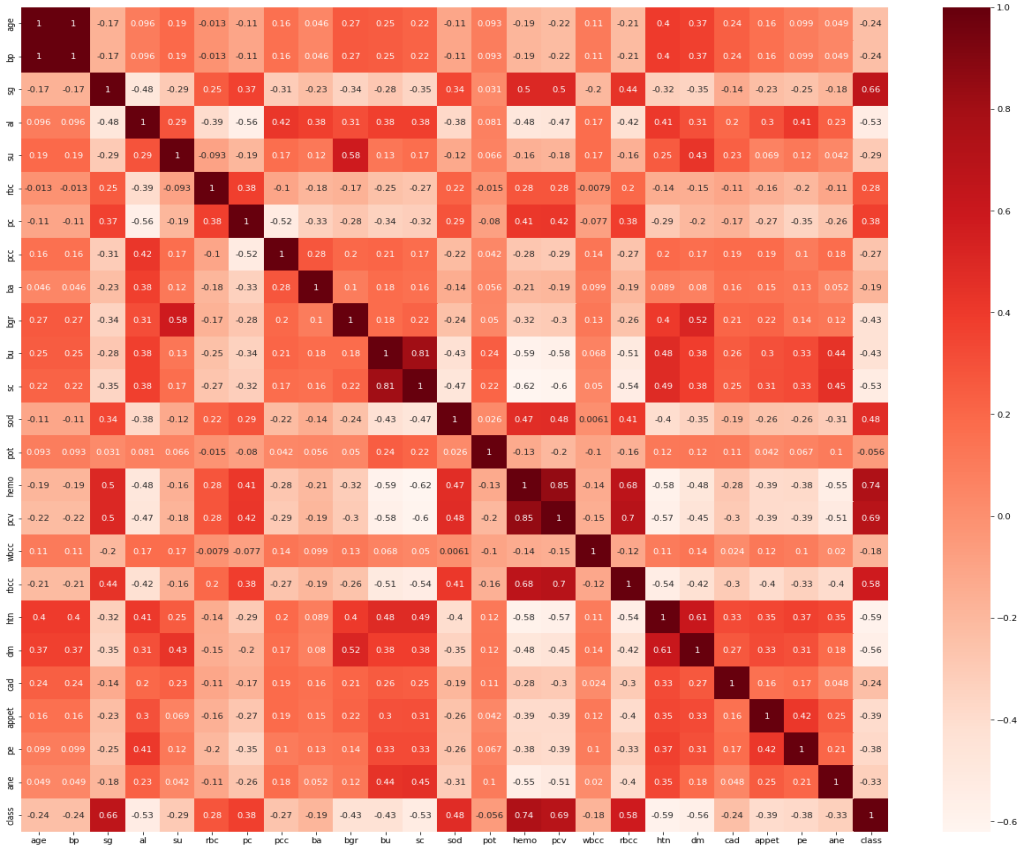
=>here the optimal number of features selected by cross validation is 3.

➢ **Filter**

Filter methods are generally used as a preprocessing step. The selection of features is independent of any machine learning algorithms. Instead, features are selected on the basis of their scores in various statistical tests for their correlation with the outcome variable.

➢ **Correlation (article 1)**:

Correlation is a statistical term which in common usage refers to how close two variables are to having a linear relationship with each other.



➢ **Correlation method by choosing the correlation threshold :**
We evaluate the relationship between each feature and target using a correlation and select those features that have the strongest relationship with the target variable.
We manually set a threshold , we select only the features that have a correlation to our target variable , higher than the threshold.

```
print(str((cor_selector1(X, y,0.3))), 'selected features')
```

```
Index(['sg', 'al', 'su', 'rbc', 'bgr', 'bu', 'sod', 'hemo', 'pcv', 'rbcc',
       'htn', 'dm', 'appet'],
      dtype='object') selected features
```

by choosing manually the threshold as 0.3 ,the number of features selected by the algorithm is 13.

➢ **Correlation method by choosing the number of features to keep :**

Features with high correlation are more linearly dependent and hence have almost the same effect on the dependent variable. So, when two features have high correlation, we can drop one of the two features.

```
cor_feature = cor_selector2(X,y,10)
print(str((cor_feature)), 'selected features')
```

```
['rbc', 'bu', 'bgr', 'sod', 'dm', 'al', 'htn', 'rbcc', 'pcv', 'hemo'] selected features
```

By manually choosing the number of features to keep as 10 ,here as the selected features.

- **Chi-square Methods :**

Chi-square test is used for categorical features in a dataset. We calculate Chi-square between each feature and the target and select the desired number of features with best Chi-square scores. It determines if the association between two categorical variables of the sample would reflect their real association in the population.
Chi- square score is given by :

$$X^2 = \frac{(Observed\ frequency - Expected\ frequency)^2}{Expected\ frequency}$$

where –

- Observed frequency = No. of observations of class
- Expected frequency = No. of expected observations of class if there was no relationship between the feature and the target.

```
print(str(len(chi2_selector(X, y,num_feats))), 'selected features')
```

```
10 selected features
```

➢ **Intrinsic :**

This method combines the qualities of both the Filter and Wrapper method to create the best subset.

- **Feature importance with decision trees :**

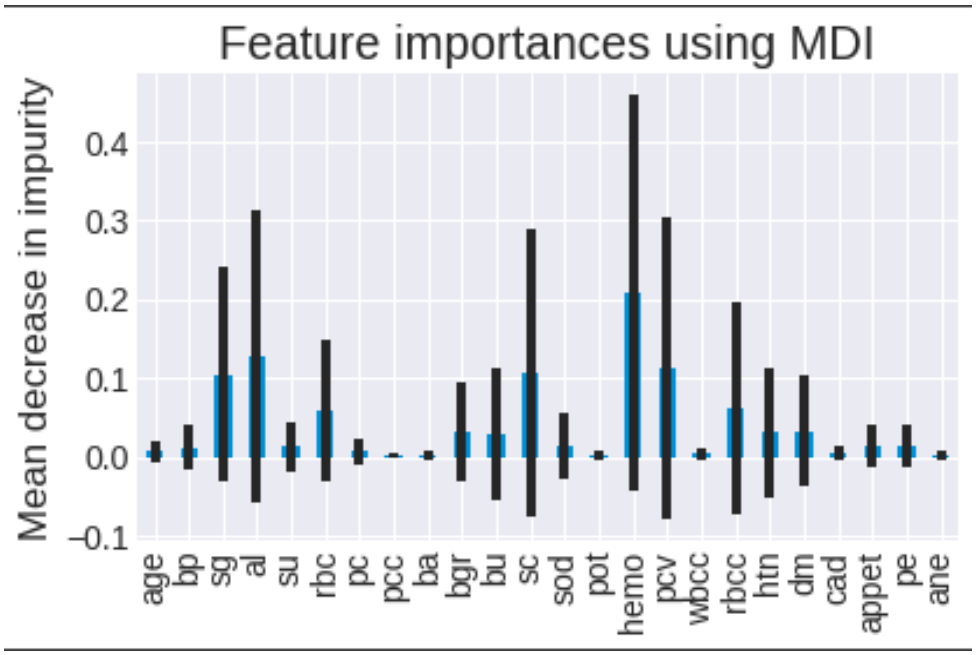
The decision trees algorithm creates a binary tree — each node has exactly two outgoing edges — finding the best numerical or categorical feature to split using an appropriate impurity criterion. For classification, *Gini impurity* or *twoing criterion* can be used.

Feature importance is calculated as the decrease in node impurity weighted by the probability of reaching that node. The node probability can be calculated by the number of samples that reach the node, divided by the total number of samples. *The higher the value the more important the feature.*

```
impurity_based_selected_features=forest_importances[forest_importances>0.028].sort_values().index
print(impurity_based_selected_features)
```

```
Index(['bu', 'htn', 'bgr', 'dm', 'rbc', 'rbcc', 'sg', 'sc', 'pcv', 'al',
       'hemo'],
      dtype='object')
```

Plot of impurity-based importance



=> We select the features that have an importance larger than 0.028.

Comparison between the feature selection features :

We perform the KNN classification with the selected features of each method.
The one that allows us to obtain the best accuracy is the feature selection technique that we will adapt.

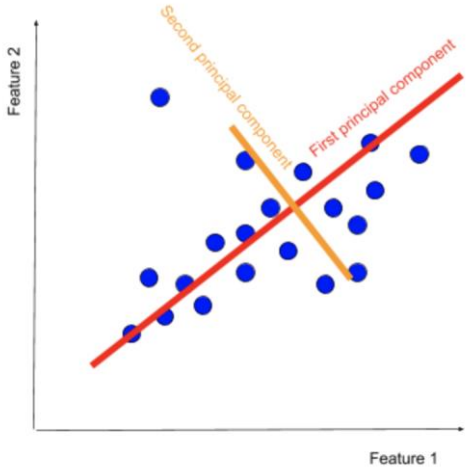
```
cor_selector1,  Test accuracy = 0.86
cor_selector2,  Test accuracy = 0.90
rfe_selector,   Test accuracy = 0.95
chi2_selector,  Test accuracy = 0.81
impurity-based importance,      Test accuracy = 0.90
```

=> The RFE technique has the highest accuracy so it is the appropriate technique.

7. Feature Reduction

Principle component analysis(PCA):

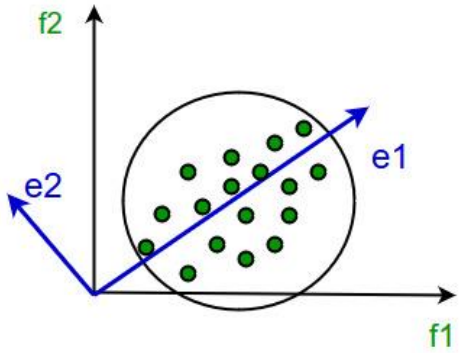
PCA is a statistical method that uses the process of linear, orthogonal transformation to transform a higher-dimensional set of features that could be possibly correlated into a lower-dimensional set of linearly uncorrelated features. These transformed and newly created features are also known as Principal Components or PCs.
In any PCA transformation, the total number of PCs is always less than or equal to the initial number of features.
The first principal component tries to capture the maximum variance of the original set of features.
The second component is orthogonal to the first, and it explains the greatest amount of variance left after the first principal component.
In very large data sets principal components remove noise by reducing a large number of features to just a couple of principal components.



```
[ ]  pca = make_pipeline(StandardScaler(),
                         PCA(n_components=2,
                             random_state=random_state))
```

how does it work:

- Construct the covariance matrix of the data.
- Compute the eigenvectors of this matrix.
- Eigenvectors corresponding to the largest eigenvalues are used to reconstruct a large fraction of variance of the original data.



Hence, we are left with a lesser number of eigenvectors, and there might have been some data loss in the process. But, the most important variances should be retained by the remaining eigenvectors.

Advantages of PCA:

- Easy to calculate:PCA is based on linear algebra, which is easy for computers to solve.
- Speeds up other machine learning algorithms. Machine learning algorithms converge faster when trained on the principal components rather than the original data set.
- Compensates for problems with high-dimensional data. Highly dimensional data makes regression-based algorithms easily overfit. By using PCA beforehand to reduce the dimensions of the training dataset, we prevent predictive algorithms from overfitting.

Disadvantages of PCA:

- Low interpretability of principal components:it is difficult to tell which are the most important features in the dataset after computing principal components.
- The trade-off between information loss and dimensionality reduction:Although dimensionality reduction is useful, it comes at a cost. Information loss is a necessary part of PCA. Balancing the trade-off between dimensionality reduction and information loss is unfortunately a necessary compromise that we have to make when using PCA

Incremental Principle component analysis(IPCA):

Incremental principal component analysis (IPCA) is typically used as a replacement for principal component analysis (PCA) when the dataset to be decomposed is too large to fit in memory. It only keeps the most significant singular vectors to project the data into a space to reduce size.

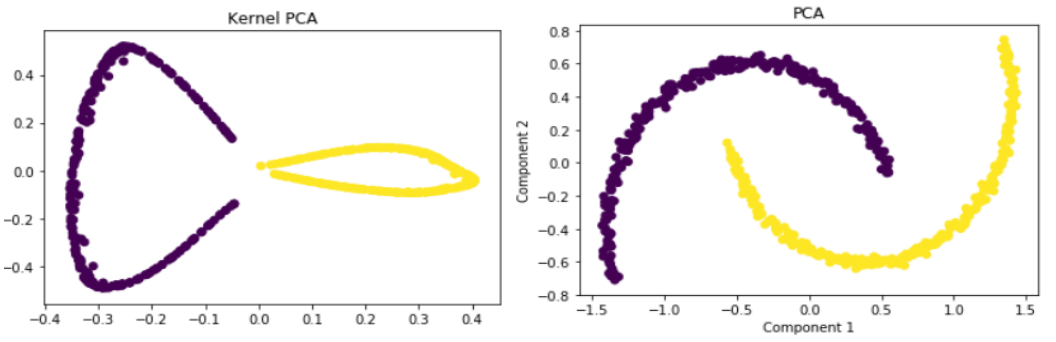
```
[ ]  inc_pca = make_pipeline(StandardScaler(),
                             IncrementalPCA(n_components=2))
```

Kernel PCA (KPCA):

Kernel PCA uses a kernel function to project a dataset into a higher dimensional feature space, where it is linearly separable. It is similar to the idea of Support Vector Machines.

There are various kernel methods like linear, polynomial, and gaussian.

In the kernel space the two classes are linearly separable. Kernel PCA uses a kernel function to project the dataset into a higher-dimensional space, where it is linearly separable.

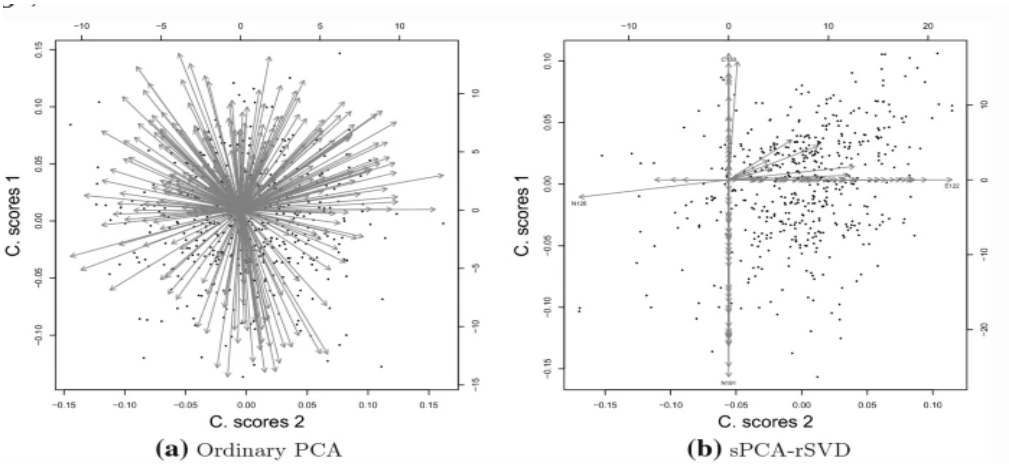



```
# kernel : “linear” | “poly” | “rbf” | “sigmoid” | “cosine” | “precomputed”
kpca = make_pipeline(StandardScaler(),
                     KernelPCA(kernel="cosine",
                               n_components=2,
                               gamma=None,
                               fit_inverse_transform=True,
                               random_state=random_state,
                               n_jobs=1))
```

Sparse PCA (KPCA):

Sparse PCA is a specialized variant of Principal Component Analysis (PCA) in machine learning that is used in statistical analysis, especially when analyzing multivariate data it can exploit the natural sparsity of data while extracting the principal components so It is used to reduce the dimensionality of a dataset by introducing sparsity structures in the input features.

So, by using the power of the sparse method, we can solve many more dimensionality reduction problems more efficiently than a standard Principal Component Analysis method.



(a) Ordinary PCA **(b)** sPCA-rSVD

```
[ ]  sparsepca = make_pipeline(StandardScaler(),
                               SparsePCA(n_components=2,
                                         alpha=0.0001,
                                         random_state=random_state,
                                         n_jobs=-1))
```

Singular Value Decomposition (SVD):

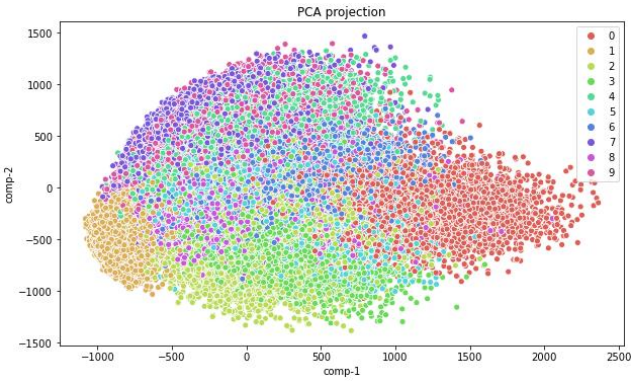
Singular Value Decomposition(SVD) is one of the most widely used Unsupervised learning algorithms, that is at the center of many recommendation and Dimensionality reduction systems that are the core of global companies.

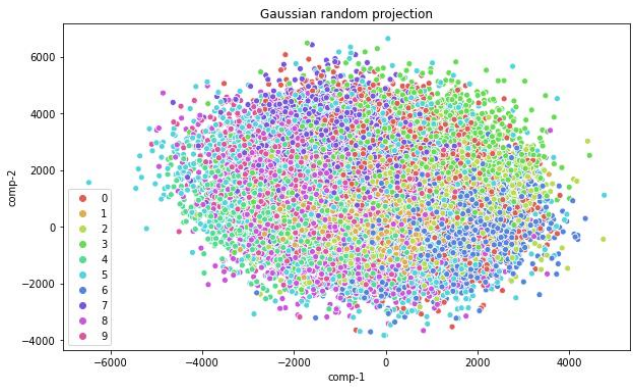
SVD can be applied even on rectangular matrices; whereas, eigenvalues are defined only for square matrices. The equivalent of eigenvalues obtained through the SVD method are called singular values, and vectors obtained equivalent to eigenvectors are known as singular vectors. However, as they are rectangular in nature, we need to have left singular vectors and right singular vectors respectively for their dimensions.

```
[ ]  SVD = make_pipeline(StandardScaler(),
                         TruncatedSVD(n_components=2,
                                      algorithm='randomized',
                                      random_state=random_state,
                                      n_iter=5))
```

Gaussian random projection (GRP):

Random projection states that the data in a high-dimensional space can be projected to a much lower dimensional space with little distortions of distances. Gaussian random distribution reduces the dimensionality by projecting the original input space on a randomly generated matrix where components are drawn from the following a normal (gaussian) distribution.



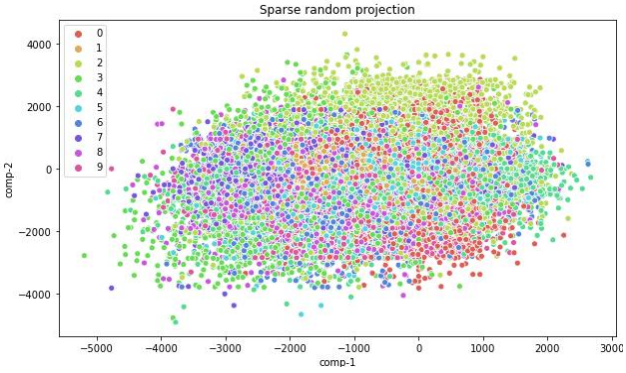


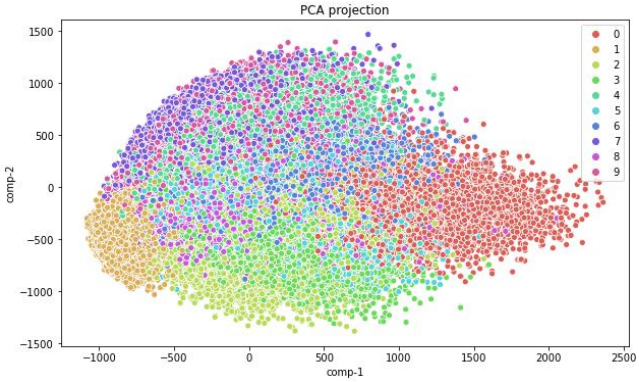

```
[ ]  GRP = make_pipeline(StandardScaler(),
                         GaussianRandomProjection(n_components=2,
                                                  eps = 0.5,
                                                  random_state=random_state))
```

Sparse random projection (SRP):

Reduces the dimensionality by projecting the original input space using a sparse random matrix. Sparse random method projects the original input space using a sparse random matrix to reduce dimension; it guarantees similar embedding quality while being much more memory efficient and allowing faster computation of the projected data.

This can be used in place of the Gaussian method in order to save some memory and improve performance. This is because the values in sparse matrices are often much more spread out, and randomly projecting this data onto a matrix with reasonable euclidean distance is always going to give us low-dimensional data



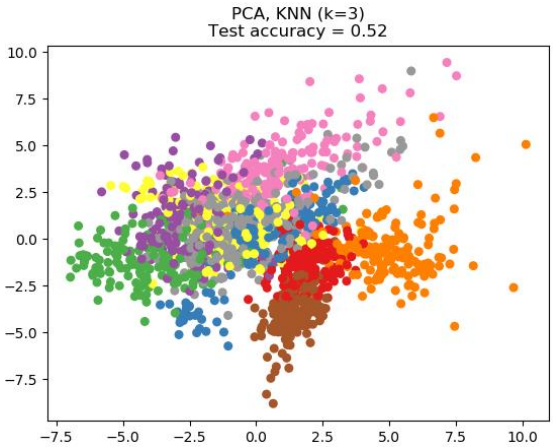


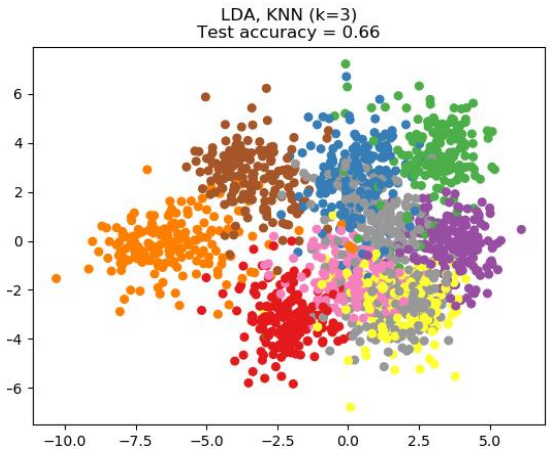

```
[ ]  SRP = make_pipeline(StandardScaler(),
                         SparseRandomProjection(n_components=2,
                                                density = 'auto',
                                                eps = 0.5,
                                                random_state=random_state,
                                                dense_output = False))
```

Linear Discriminant Analysis(LDA):

Linear Discriminant Analysis seeks to best separate (or discriminate) the samples in the training dataset by their class value. Specifically, the model seeks to find a linear combination of input variables that achieves the maximum separation for samples between classes (class centroids or means) and the minimum separation of samples within each class

Linear Discriminant Analysis (LDA) tries to identify attributes that account for the most variance between classes. In particular, LDA, in contrast to PCA, is a supervised method, using known class labels.





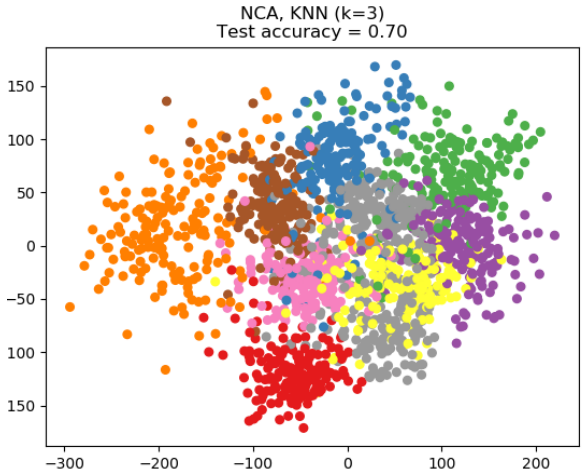
```
[ ]  lda = make_pipeline(StandardScaler(),
                         LinearDiscriminantAnalysis(n_components=2))
```

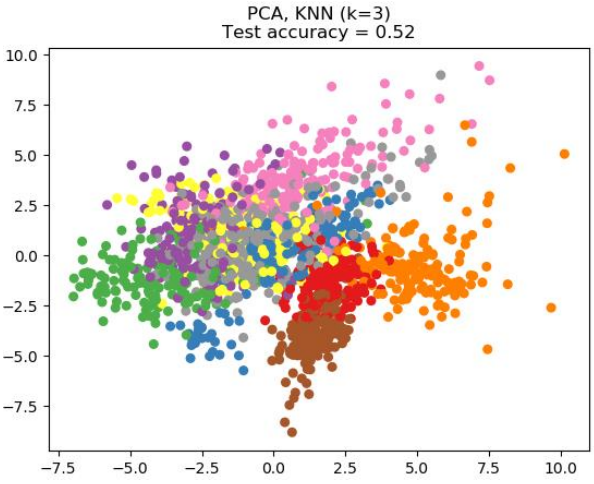
Neighborhood component analysis (NCA):

Neighborhood Component Analysis (NCA) is a machine learning algorithm for metric learning. It learns a linear transformation in a supervised fashion to improve the classification accuracy of a stochastic nearest neighbors rule in the transformed space.

NCA tries to find a feature space such that a stochastic nearest neighbor algorithm will give the best accuracy

We can see below that NCA enforces a clustering of the data that is visually meaningful despite the large reduction in dimension.

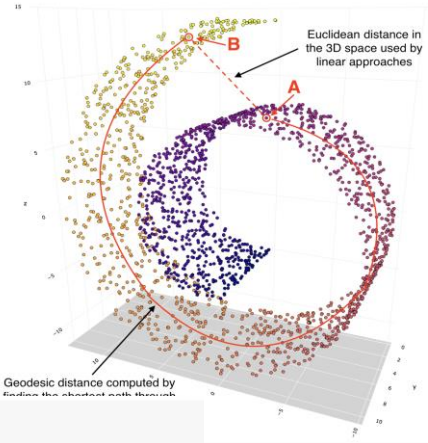





```
[ ]  nca = make_pipeline(StandardScaler(),
                         NeighborhoodComponentsAnalysis(n_components=2,
                                                        random_state=random_state))
```

Isometric mapping (Isomap):

Isomap stands for isometric mapping. Isomap is a non-linear dimensionality reduction method based on the spectral theory which tries to preserve the geodesic distances in the lower dimension.




```
[ ]  isomap = make_pipeline(StandardScaler(),
                            Isomap(n_components=2,
                                   n_jobs = 4,
                                   n_neighbors = 5))
```

MiniBatch Dictionary Learning:

Dictionary-based learning solves a problem of matrix factorization which amounts to finding a dictionary that can give good results under the condition of parsimony of the code.

```
[ ]  miniBatchDictLearning = make_pipeline(StandardScaler(),
                                           MiniBatchDictionaryLearning(n_components=2,
                                                                       batch_size = 200,
                                                                       alpha = 1,
                                                                       n_iter = 25,
                                                                       random_state=random_state))
```

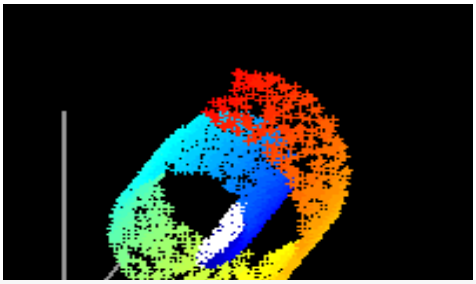
Independent Component analysis(ICA):

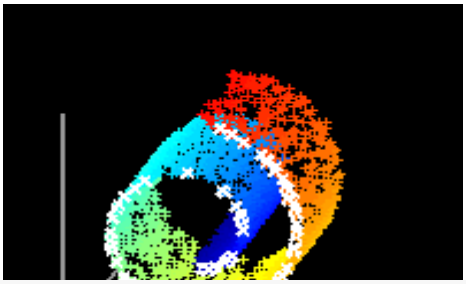
ICA stands for Independent Components Analysis. ICA is a linear dimension reduction method, which transforms the dataset into columns of independent components, It assumes that each sample of data is a mixture of independent components and it aims to find these independent components.

```
[ ]  FastICA = make_pipeline(StandardScaler(),
                             FastICA(n_components=2,
                                     algorithm = 'parallel',
                                     whiten = True,
                                     max_iter = 100,
                                     random_state=random_state))
```

Locally Linear Embedding(LLE):

LLE works by first measuring how each training instance linearly relates to its closest neighbors (c.n.), and then looking for a low-dimensional representation of the training set where these local relationships are best preserved (more details shortly). This makes it particularly good at unrolling twisted manifolds, especially when there is not too much noise.
LLE first finds the k-nearest neighbors of the points. Then, it approximates each data vector as a weighted linear combination of its k-nearest neighbors. Finally, it computes the weights that best reconstruct the vectors from its neighbors, then produce the low-dimensional vectors best reconstructed by these weights

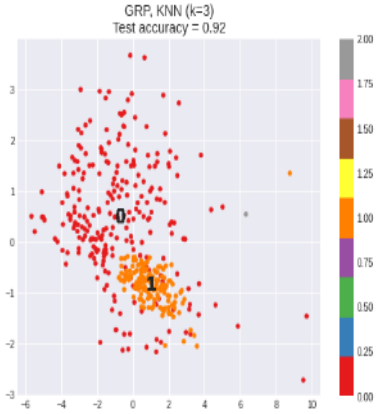


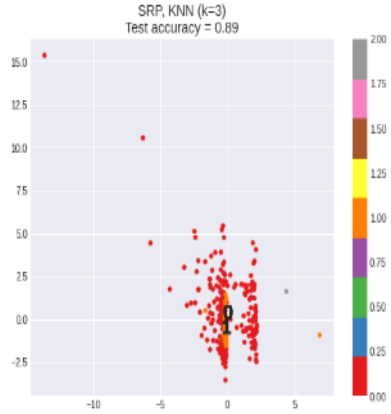


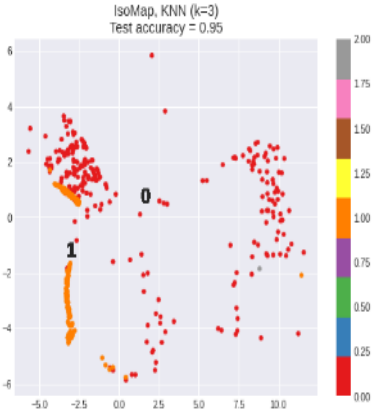
```
lle = make_pipeline(StandardScaler(),
                    LocallyLinearEmbedding(n_components=2,
                                           n_neighbors = 10,
                                           method = 'modified',
                                           n_jobs = 4,
                                           random_state=random_state))
```

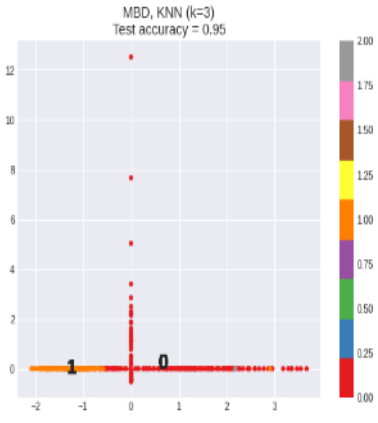
Comparison between the methods :

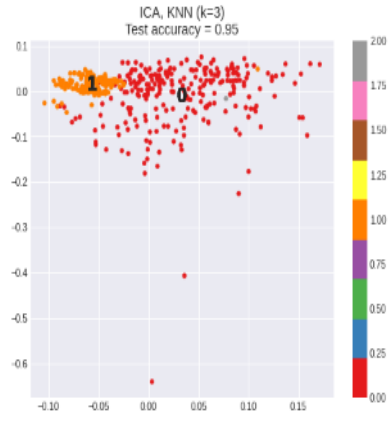
Here we are going to use K nearest neighbor to evaluate the methods . Each method will have its own chart with the feature distribution in 2D and in 3D and the test accuracy of the KNN.

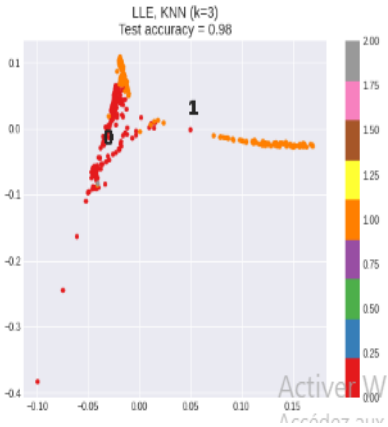

GRP, KNN (k=3)
Test accuracy = 0.92

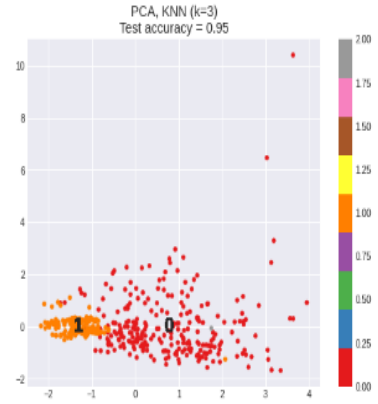

SRP, KNN (k=3)
Test accuracy = 0.89

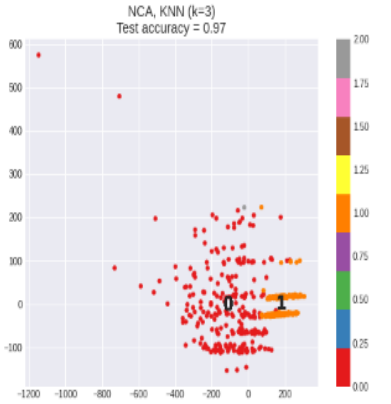

IsoMap, KNN (k=3)
Test accuracy = 0.95

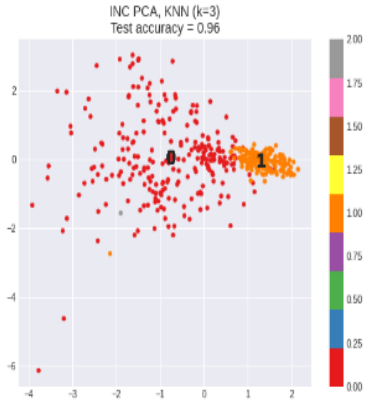

MBD, KNN (k=3)
Test accuracy = 0.95

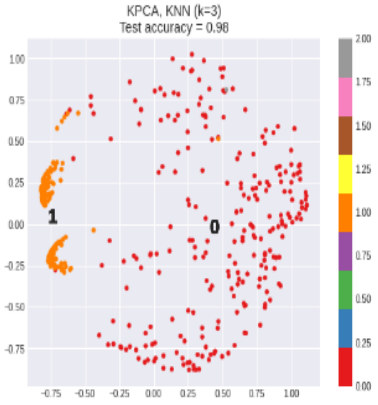

ICA, KNN (k=3)
Test accuracy = 0.95

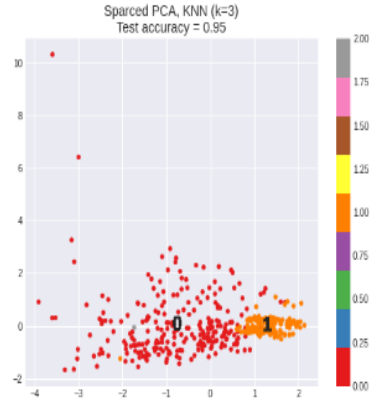

LLE, KNN (k=3)
Test accuracy = 0.98

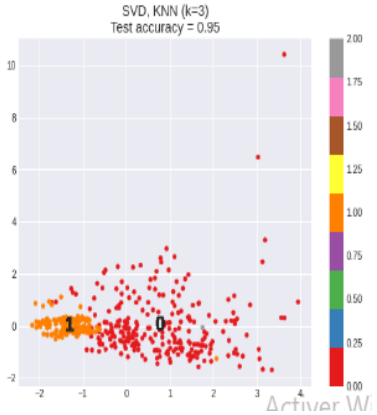

PCA, KNN (k=3)
Test accuracy = 0.95

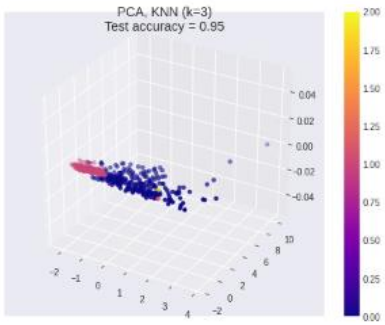

NCA, KNN (k=3)
Test accuracy = 0.97

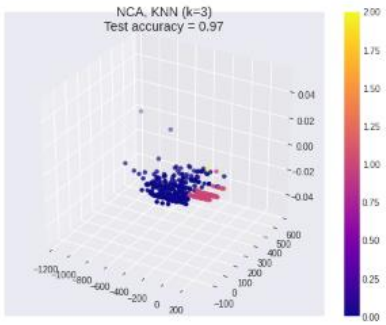

INC PCA, KNN (k=3)
Test accuracy = 0.96

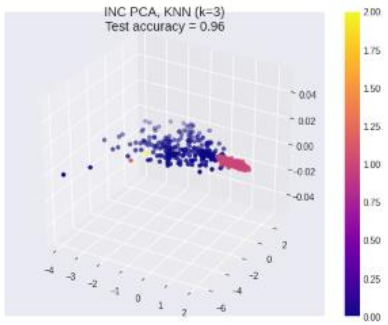

KPCA, KNN (k=3)
Test accuracy = 0.98

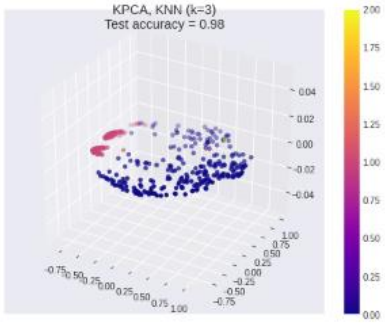

Sparced PCA, KNN (k=3)
Test accuracy = 0.95

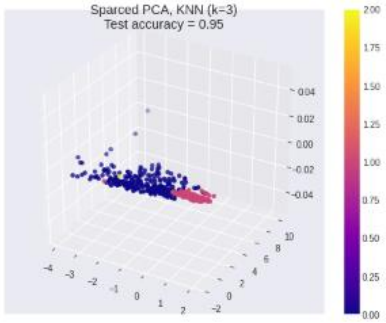

SVD, KNN (k=3)
Test accuracy = 0.95

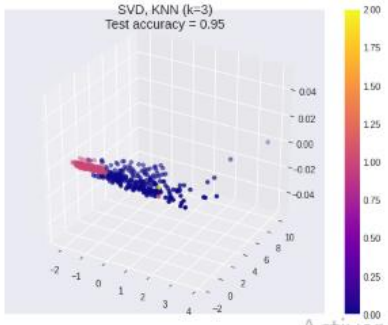

PCA, KNN (k=3)
Test accuracy = 0.95

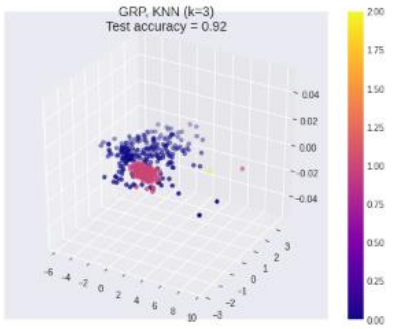

NCA, KNN (k=3)
Test accuracy = 0.97

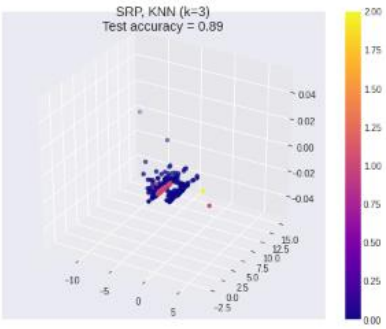

INC PCA, KNN (k=3)
Test accuracy = 0.96

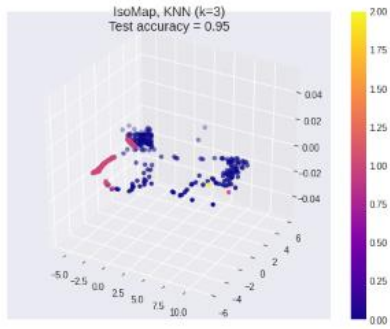

KPCA, KNN (k=3)
Test accuracy = 0.98

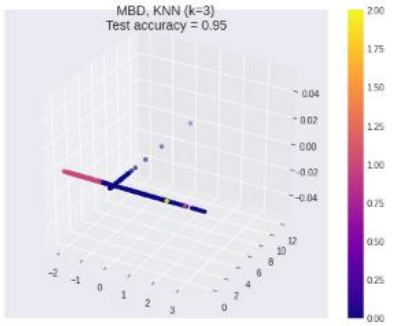

Sparced PCA, KNN (k=3)
Test accuracy = 0.95

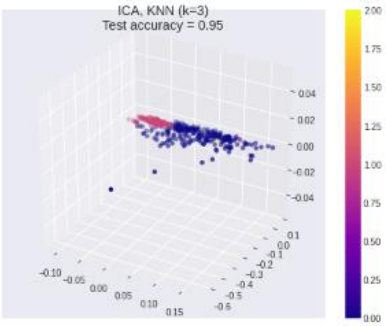

SVD, KNN (k=3)
Test accuracy = 0.95

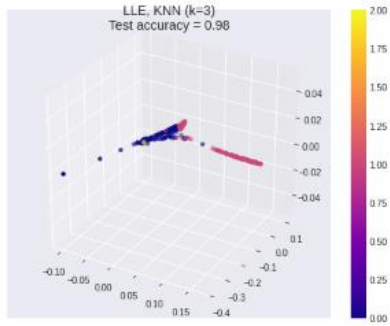

GRP, KNN (k=3)
Test accuracy = 0.92

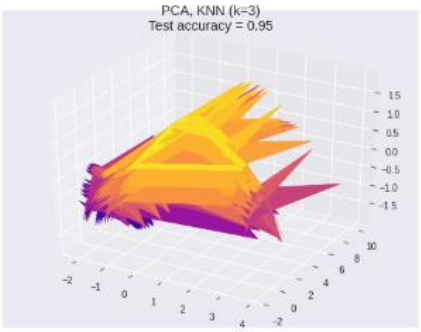

SRP, KNN (k=3)
Test accuracy = 0.89

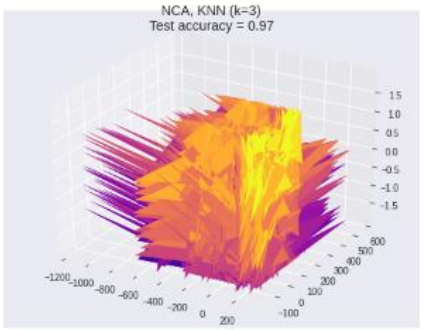

IsoMap, KNN (k=3)
Test accuracy = 0.95

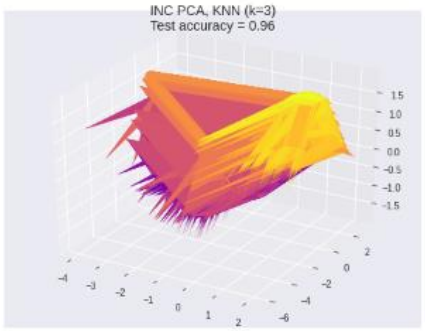

MBD, KNN (k=3)
Test accuracy = 0.95

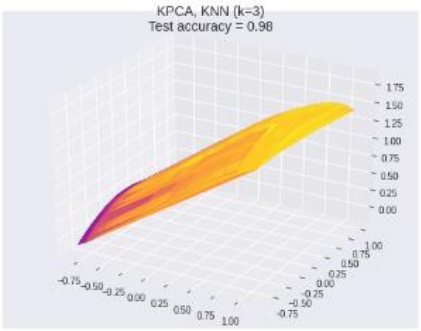

ICA, KNN (k=3)
Test accuracy = 0.95

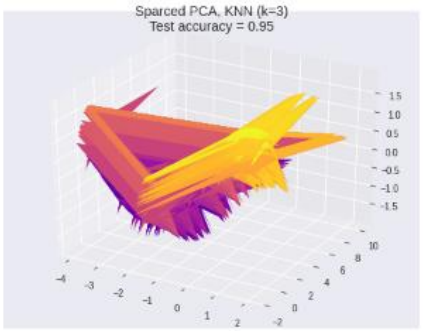

LLE, KNN (k=3)
Test accuracy = 0.98

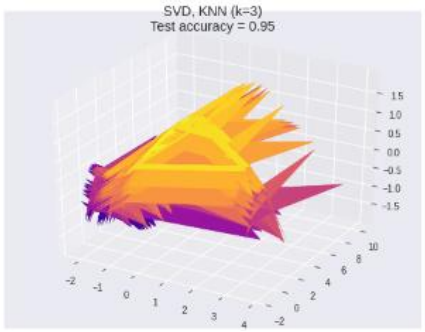


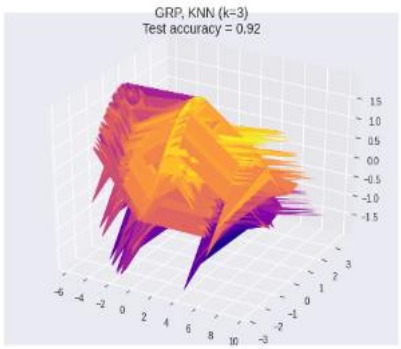


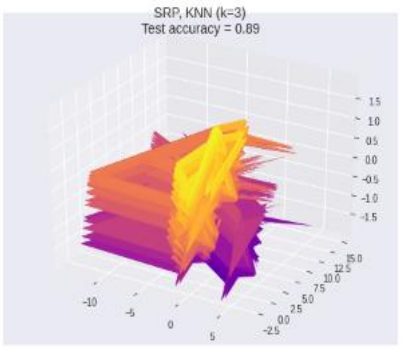


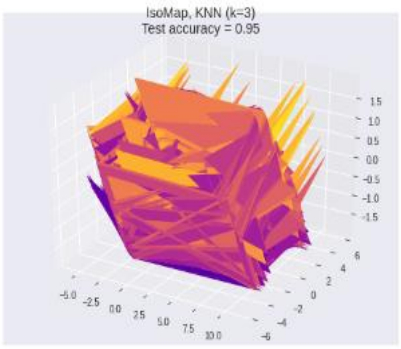


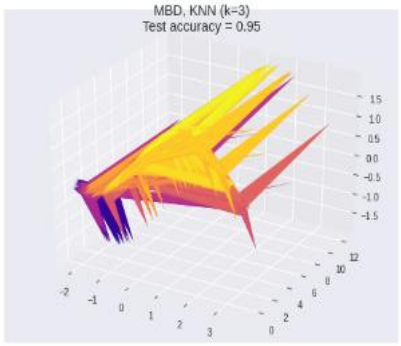


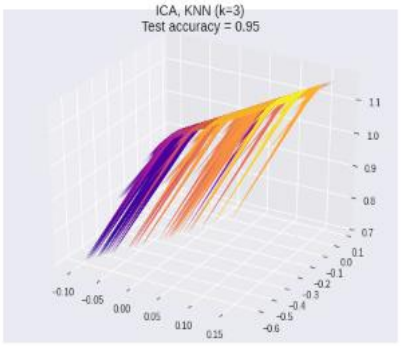


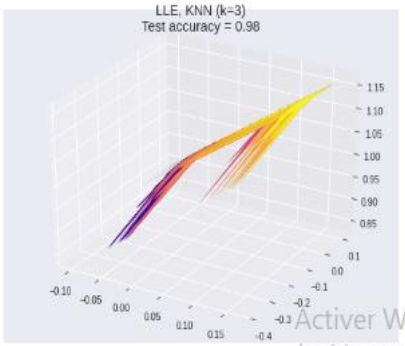












=> NCA, KPCA have the highest accuracy ,We choose to continue with Kernel PCA.

Conclusion : we did different tasks in data preprocessing to prepare our data for the next phase which is the phase of modeling.

8. Modeling :1st Article Boosted Classifier and Features Selection for Enhancing Chronic Kidney Disease Diagnose

Introduction :

In this chapter, we will be treating the first article named “Article Boosted Classifier and Features Selection”. In this article, we will treat our data with 3 different methods , we will compare our results and choose the best method for our modeling process.
In the first method , We will use our models on our data pre-feature selection.
In the second part, we will use a feature selection for our data and treat the new data with our models.
In the third method, we apply AdaBoost using our different models and the data after the feature selection. Then we compare the results to the 3 different methods used.

We will start by giving a brief explanation to the algorithms used :

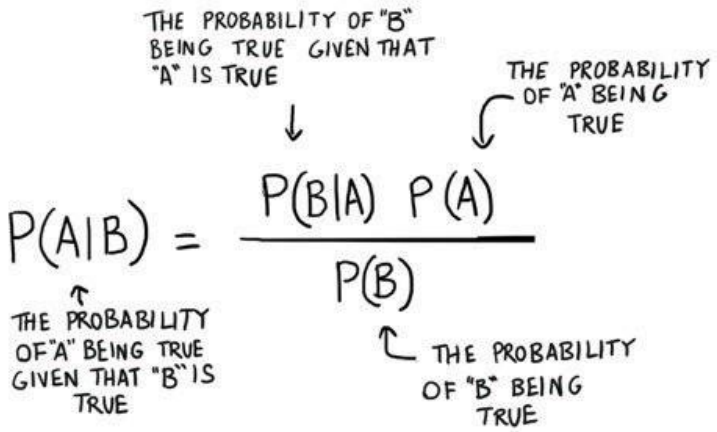
1. Naive Bayes :

Principle of Naive Bayes Classifier:

A Naive Bayes classifier is a probabilistic machine learning model that’s used for classification tasks. The crux of the classifier is based on the Bayes theorem.

Bayes Theorem:

Using Bayes theorem, we can find the probability of A happening, given that B has occurred. Here, B is the evidence and A is the hypothesis. The assumption made here is that the predictors/features are independent. That is, the presence of one particular feature does not affect the other. Hence it is called naive.



Types of Naive Bayes Classifier:

- **Multinomial Naive Bayes:**
  This is mostly used for document classification problems, i.e whether a document belongs to the category of sports, politics, technology etc. The features/predictors used by the classifier are the frequency of the words present in the document.

- **Bernoulli Naive Bayes:**
  This is similar to the multinomial naive bayes but the predictors are boolean variables. The parameters that we use to predict the class variable take up only values yes or no, for example if a word occurs in the text or not.

- **Gaussian Naive Bayes:**
  When the predictors take up a continuous value and are not discrete, we assume that these values are sampled from a gaussian distribution.

Advantages of Using Naive Bayes Classifier

- **Simple to Implement.** The conditional probabilities are easy to evaluate.
- **Very fast** – no iterations since the probabilities can be directly computed. So this technique is useful where speed of training is important.
- If **the conditional Independence** assumption holds, it could give great results.

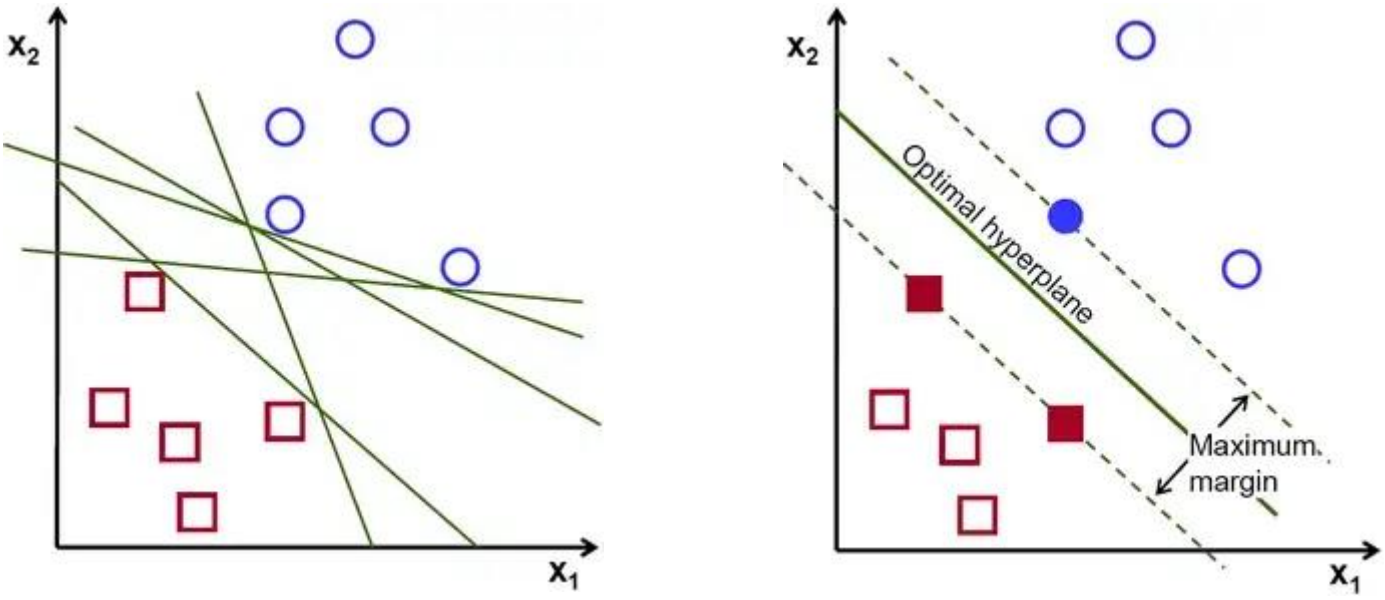
Disadvantages of Using Naive Bayes Classifier

- **Conditional Independence Assumption** does not always hold. In most situations, the feature shows some form of dependency.
- **Zero probability problem :** When we encounter words in the test data for a particular class that are not present in the training data, we might end up with zero class probabilities.

2. Support Vector Machine (SVM) :

What is Support Vector Machine?

A support vector machine (SVM) is a supervised machine learning model that uses classification algorithms for two-group classification problems.
The objective of the support vector machine algorithm is to find a hyperplane in an N-dimensional space(N — the number of features) that distinctly classifies the data points.



Possible hyperplanes

To separate the two classes of data points, there are many possible hyperplanes that could be chosen. Our objective is to find a plane that has the maximum margin, which is the maximum distance between data points of both classes. Maximizing the margin distance provides some reinforcement so that future data points can be classified with more confidence.

Advantages:

- SVM works relatively well when there is a clear margin of separation between classes.
- It is more productive in high dimensional spaces.
- Its memory efficient as it uses a subset of training points in the decision function called support vectors
- SVM is effective in cases where the number of dimensions is greater than the number of samples.
- It is effective in instances where the number of dimensions is larger than the number of specimens.

Disadvantages :

- Support vector machine algorithms are not acceptable for large data sets.
- It does not execute very well when the data set has more sound i.e. target classes are overlapping.
- In cases where the number of properties for each data point outstrips the number of training data specimens, the support vector machine will underperform.
- As the support vector classifier works by placing data points, above and below the classifying hyperplane there is no probabilistic clarification for the classification.

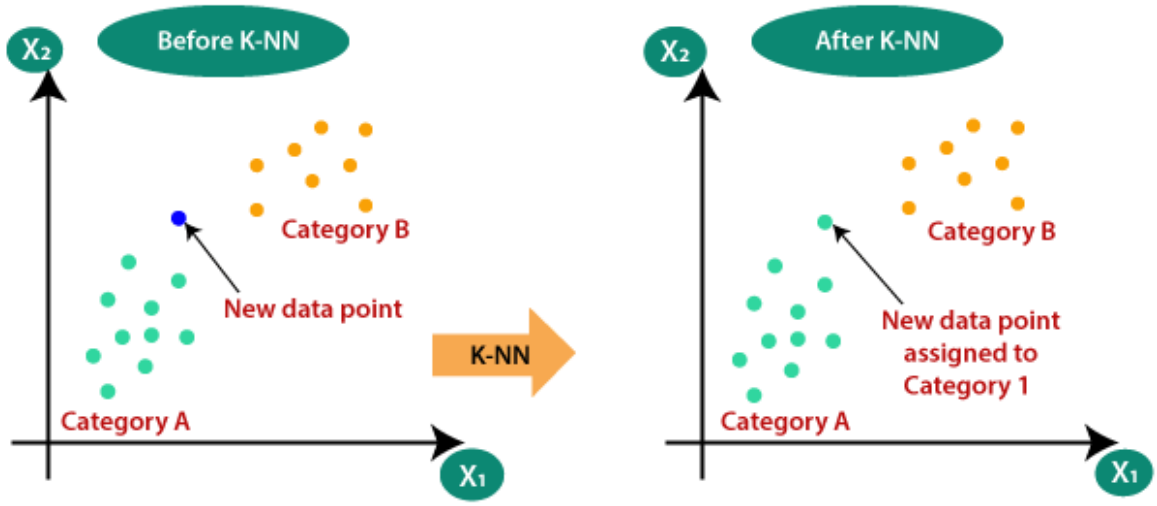
3. K-nearest neighbors (kNN):

The k-nearest neighbors classifier (kNN) is a non-parametric supervised machine learning algorithm. It's distance-based: it classifies objects based on their proximate neighbors' classes. kNN is most often used for classification.

How does the K-NN Classifier work?

The K-NN working can be explained on the basis of the below algorithm:

- **Step-1:** Select the number K of the neighbors
- **Step-2:** Calculate the Euclidean distance of K number of neighbors
- **Step-3:** Take the K nearest neighbors as per the calculated Euclidean distance.
- **Step-4:** Among these k neighbors, count the number of the data points in each category.
- **Step-5:** Assign the new data points to that category for which the number of the neighbor is maximum.
- **Step-6:** Our model is ready.



Weighted KNN

Weighted KNN is a modified version of the KNN algorithm.

In weighted kNN, the nearest k points are assigned a weight.

The intuition behind weighted KNN is to give more weight to the points which are nearby and less weight to the points which are farther away.

The simple function which is used is the inverse distance function which implies that as the distance increases weight decreases and as the distance decreases, weight increases.

```
model = KNeighborsClassifier(n_neighbors=5, weights='distance')
```

Advantages of KNN Algorithm:

- It is simple to implement.
- It is robust to the noisy training data
- It can be more effective if the training data is large.
- There's no need to build a model, tune several parameters, or make additional assumptions.

Disadvantages of KNN Algorithm:

- Always needs to determine the value of K which may be complex some time.
- The computation cost is high because of calculating the distance between the data points for all the training samples.
- The algorithm gets significantly slower as the number of examples and/or predictors/independent variables increase.

4. k-Folds Cross Validation

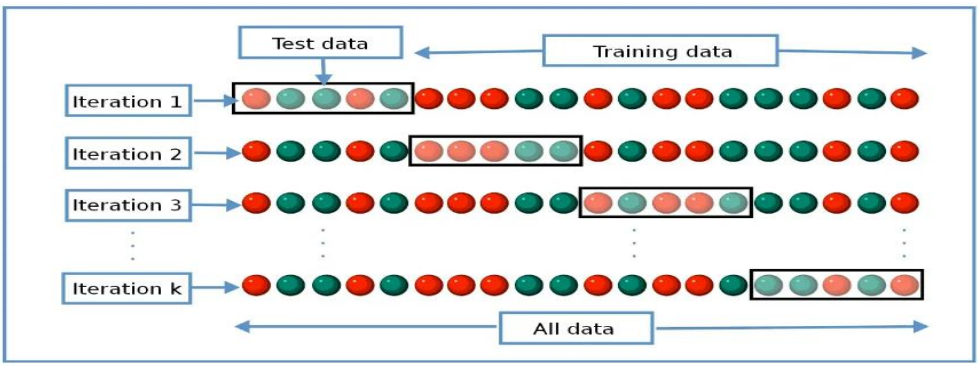
Cross validation is an evaluation method used in machine learning to find out how well your machine learning model can predict the outcome of unseen data.

How is k-fold cross validation performed?

The general strategy is quite straightforward and the following steps can be used:

1. First, shuffle the dataset and split into k number of subsamples. .
2. In the first iteration, the first subset is used as the test data while all the other subsets are considered as the training data.
3. Train the model with the training data and evaluate it using the test subset. Keep the evaluation score or error rate, and get rid of the model.
4. Now, in the next iteration, select a different subset as the test data set, and make everything else.
5. Re-train the model with the training data and test it using the new test data set, keep the evaluation score and discard the model.

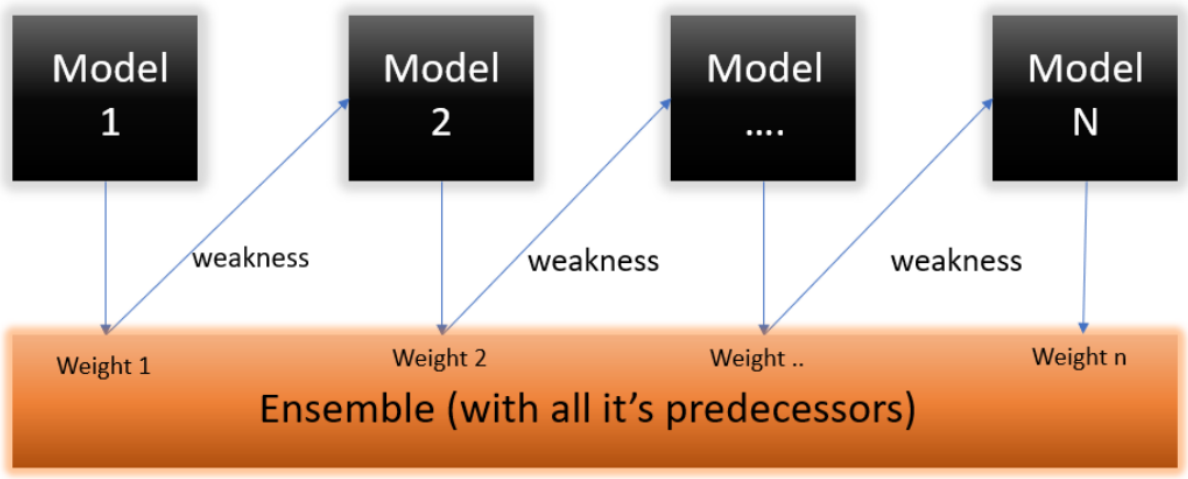
6. Continue iterating the above k times. Each data subsamples will be used in each iteration until all data is considered. You will end up with a k number of evaluation scores.
7. The total error rate is the average of all these individual evaluation scores.



5. AdaBoost Algorithm

AdaBoost also called Adaptive Boosting is a technique in Machine Learning used as an Ensemble Method. The most common algorithm used with AdaBoost is decision trees

What this algorithm does is that it builds a model and gives equal weights to all the data points. It then assigns higher weights to points that are wrongly classified. Now all the points which have higher weights are given more importance in the next model. It will keep training models until and unless a lower error is received.



6. Classification Result :

First Method : base

Here are the metrics we used :

Accuracy

Accuracy is the fraction of predictions our model got right $\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}}$

Recall

Recall gives us the percentage of positives well predicted by our model. $recall = \frac{True\ Positive}{True\ Positive + False\ Negative}$

Precision

It shows the number of positive predictions well made. $precision = \frac{True\ Positive}{True\ Positive + False\ Positive}$

F1 Score

Although useful, neither precision nor recall can fully **evaluate a Machine Learning model.** $F1\ Score = 2 \times \frac{recall \times precision}{recall + precision}$

The F1 Score provides a good evaluation of the performance of our model.

	labels	Accuracy	Precision	Recall	F1
0	KNN	0.910221	0.946667	0.9425	0.926279
1	Naive Bayes	0.796011	1.000000	0.9000	0.884693
2	SVM	0.949661	0.966667	0.9675	0.956863

Second Method :Classification with CFS

	labels	Accuracy	Precision	Recall	F1
0	KNN	0.948057	0.920000	0.9500	0.930511
1	Naive Bayes	0.714952	0.986667	0.8425	0.827472
2	SVM	0.913911	0.946667	0.9450	0.927796

Third Method : Classification with CFS and Adaboost

	labels	Accuracy	Precision	Recall	F1
0	AdaBoost with SVM	0.963141	0.926667	0.9575	0.940748
1	AdaBoost with weighted KNN	NaN	NaN	NaN	NaN
2	AdaBoost with Naive Bayes	0.476907	0.993333	0.5875	0.644184

7. Comparing the classification results :

For this section ,we will be comparing our results.

In this process , we will compare the f1 score of each model through the 3 methods used.

We chose the f1 score as our metric because our dataset is imbalanced .We have an uneven class distribution.

That means for each model we will compare the f1 score in : base, after feature selection,and finally with ada boost and feature selection and then we can choose what went well for our model and the best f1 score we can get.

- **KNN:** we got 0.92 F1 score applying our model on data pre-feature selection. With feature selection, our F1 score became 0.93 .
- **Naive Bayes :** we got 0.884 F1 score applying our model on data pre-feature selection. With feature selection, our F1 score became 0.827 and then with ada boost and feature selection , our F1 score became 0.644. As we can see , the F1 score declined with adaboost and CFS drastically , which is an indicator that we overfitted
- **SVM :** we got 0.95 F1 score applying our model on data pre-feature selection. With feature selection, our F1 score became 0.927 and then with ada boost and feature selection , our F1 score became 0.94.

Conclusion :

In this chapter, we went in detail through the algorithms used for modeling according to the first article and we applied different methods so we can get the best results.

9. Modeling :2nd Article Diagnosis of Chronic Kidney Disease Using Effective Classification Algorithms and Recursive Feature Elimination Techniques

Introduction :

Here we have built and assessed various models based on several different modeling techniques.Whereas the Assess Model task of the Modeling phase focuses on technical model assessment, the Evaluation phase looks more broadly at which model best meets the business and what to do next.

1-Steps of modeling and evaluation:

- **Generate test design**: Pending your modeling approach, you might need to split the data into training, test, and validation sets.
- **Build model:** As glamorous as this might sound, this might just be executing a few lines of code like “reg = LinearRegression().fit(X, y)”.
- **Assess model:** Generally, multiple models are competing against each other, and the data scientist needs to interpret the model results based on domain knowledge, the pre-defined success criteria, and the test design.
- **Evaluate results**: Do the models meet the business success criteria? Which one(s) should we approve for the business?

- **Review process**: Review the work accomplished. Was anything overlooked? Were all steps properly executed? Summarize findings and correct anything if needed.

- **Determine next steps**: Based on the previous three tasks, determine whether to proceed to deployment, iterate further, or initiate new projects.

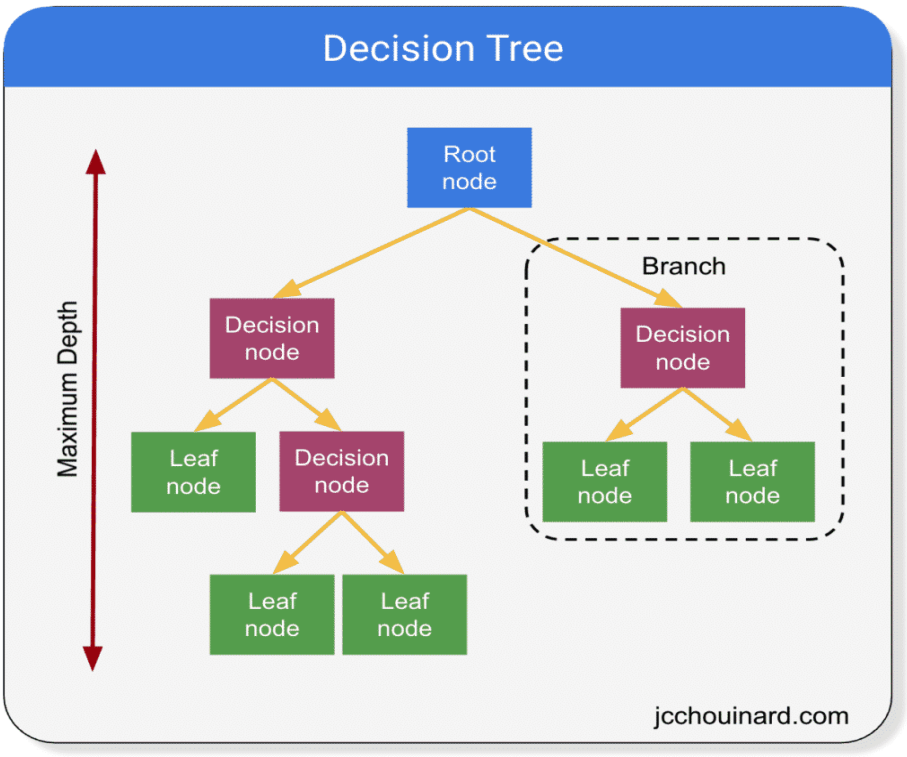
  In this phase we have built 4 models with 4 different techniques:

  - **KNN** : K-nearest neighbors (mentioned in Article 1)
  - **SVM** : Support vector machine with Grid search for hyperparameter tuning (Article1)
  - **RF** : Random Forest with Grid search for hyperparameter tuning
  - **DT**: Decision Tree with Grid search for hyperparameter tuning

2-Definition of the algorithms :

2.1-Decision Tree Classifier :

A Decision tree is a flowchart-like tree structure, where each internal node denotes a test on an attribute, each branch represents an outcome of the test, and each leaf node (terminal node) holds a class label.



The goal is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features. A tree can be seen as a piecewise constant approximation.

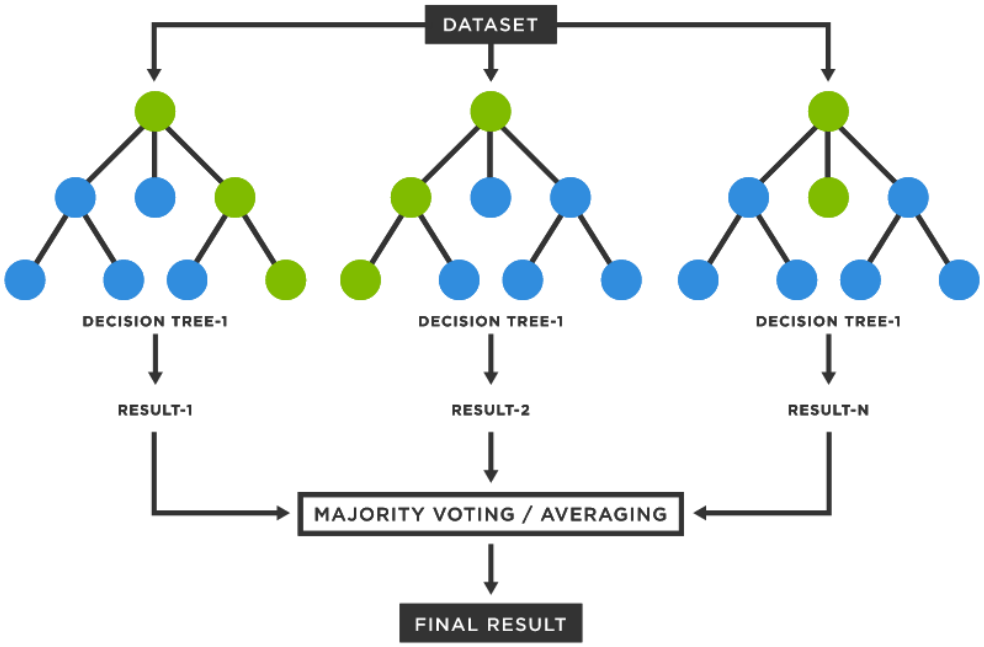
The strengths of decision tree methods are:

- Simple to understand, interpret, visualize.
- Decision trees require relatively little effort from users for data preparation.
- Decision trees perform classification without requiring much computation.
- Decision trees are able to handle both continuous and categorical variables.
- Decision trees provide a clear indication of which fields are most important for prediction or classification.
- Nonlinear relationships between parameters do not affect tree performance.

The weaknesses of decision tree methods :

- Decision trees are prone to errors in classification problems with many classes and a relatively small number of training examples.
- Prone to overfitting: Complex decision trees tend to overfit and do not generalize well to new data.
- Decision trees can be unstable because small variations in the data might result in a completely different tree being generated. This is called variance, which needs to be lowered by methods like bagging and boosting.
- More costly: Given that decision trees take a greedy search approach during construction, they can be more expensive to train compared to other algorithms.
- Decision tree learners create biased trees if some classes dominate. It is therefore recommended to balance the data set prior to fitting with the decision tree.

9.2-Random Forest Classifier :



Random forest is a supervised learning algorithm. The “forest” it builds is an ensemble of decision trees, usually trained with the “bagging” method. The general idea of the bagging method is that a combination of learning models increases the overall result.

The random forest algorithm establishes the outcome based on the predictions of the decision trees. It predicts by taking the average or mean of the output from various trees. Increasing the number of trees increases the precision of the outcome.

Steps involved in random forest algorithm:

Step 1: n number of random records are taken from the data set having k number of records.

Step 2: Individual decision trees are constructed for each sample.

Step 3: Each decision tree will generate an output.

Step 4: Final output is considered based on Majority Voting or Averaging for Classification and regression respectively.

Advantages of random forest

- A random forest produces good predictions that can be understood easily.
- It can handle large datasets efficiently.
- The random forest algorithm provides a higher level of accuracy in predicting outcomes over the decision tree algorithm.
- It is immune to the curse of dimensionality. Since each tree does not consider all the attributes, feature space is reduced.
- It maintains diversity as all the attributes are not considered while making each decision tree though it is not true in all cases.
- It solves the problem of overfitting as output is based on majority voting or averaging.

Disadvantages of random forest

- When using a random forest, more resources are required for computation.
- It consumes more time compared to a decision tree algorithm.
- Random forest is highly complex when compared to decision trees where decisions can be made by following the path of the tree.

Difference Between Decision Tree & Random Forest

Random forest is a collection of decision trees; still, there are a lot of differences in their behavior.

Decision trees	Random Forest
1. Decision trees normally suffer from the problem of overfitting if it's allowed to grow without any control.	1. Random forests are created from subsets of data and the final output is based on average or majority ranking and hence the problem of overfitting is taken care of.
2. A single decision tree is faster in computation.	2. It is comparatively slower.
3. When a data set with features is taken as input by a decision tree it will formulate some set of rules to do prediction.	3. Random forest randomly selects observations, builds a decision tree and the average result is taken. It doesn't use any set of formulas.

Thus random forests are much more successful than decision trees only if the trees are diverse and acceptable.

Splitting :

First we are going to split the data into training data and test data .

```
# split into x and y dataset
X=finalDf.drop('class',axis=1)
y=finalDf['class']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=42)
#Training : 300 patients
#Test : 100 patients
```

Training for random forest regressor:

```
[ ]  #Train the model using the training sets
     rf_model.fit(X_train, y_train)
```

Training for Decision Tree:

```
#Train the model using the training sets
DT_model.fit(X_train, y_train)
```

Training for KNN:

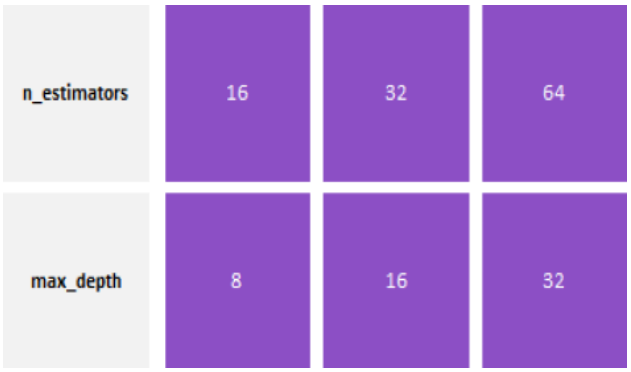
```
#Train the model using the training sets
KNN.fit(X_train, y_train)
```

Training for SVM:

```
# fitting the model for grid search
grid.fit(X_train, y_train)
```

2.3-Hyperparameter tuning using Grid search :

The idea behind the grid search technique is quite simple. We have a model with parameters, and the challenge is to test various configurations until we are satisfied with the result. Grid search is exhaustive in that it tests all permutations of a parameter grid. The number of model variants results from the parameter grid and the specified parameters.



The grid search algorithm requires us to provide the following information:

- The hyperparameters that we want to configure (e.g., tree depth)
- For each hyperparameter a range of values (e.g., [50, 100, 150])

- A performance metric so that the algorithm knows how to measure performance (e.g., accuracy for a classification model)

Strengths and Weaknesses of Grid Search

The advantage of the grid search is that the algorithm automatically identifies the optimal parameter configuration from the parameter grid. However, the number of possible configurations increases exponentially with the number of values in the parameter grid. So, in practice, defining a sparse parameter grid or defining stopping criteria is essential.

Hyperparameters for random forest regressor :

Random decision forests have several hyperparameters that we can use to influence their behavior.They are either used to increase the predictive power of the model or to make the model faster.

1. max_depth: The *max_depth* of a tree in Random Forest is defined as the longest path between the root node and the leaf node.

2. min_sample_split: Parameter that tells the decision tree in a random forest the minimum required number of observations in any given node to split it. Default = 2

3. min_samples_leaf: This Random Forest hyperparameter specifies the minimum number of samples that should be present in the leaf node after splitting a node. Default = 1

4. n_estimators: Number of trees in the forest.

5. max_features: This resembles the number of maximum features provided to each tree in a random forest.

```
[ ]  rf_random.best_params_

     {'n_estimators': 313,
      'min_samples_split': 5,
      'min_samples_leaf': 6,
      'max_features': 'sqrt',
      'max_depth': 80}
```

Hyperparameters for Decision Tree :

1. **min_samples_split** :Minimum number of samples a node must possess before splitting.
2. **min_samples_leaf** : Minimum number of samples a leaf node must possess.
3. **min_weight_fraction_leaf** :Minimum fraction of the sum total of weights required to be at a leaf node.
4. **max_leaf_nodes** : Maximum number of leaf nodes a decision tree can have.
5. **max_features** : Maximum number of features that are taken into the account for splitting each node.

```
grid_search_cv.best_estimator_

DecisionTreeClassifier(max_leaf_nodes=2, random_state=42)
```

Hyperparameters for SVM :

1. **Kernels**: The main function of the kernel is to take a low-dimensional input space and transform it into a higher dimensional space. It is most useful in nonlinear separation problems.
2. **C (Regularization):** C is the penalty parameter, which represents the misclassification or error term. The misclassification or error term tells the SVM optimization the tolerable error level.
3. **gamma**: It defines the influence of the plausible separation line calculation.

```
{'C': 1000, 'gamma': 1, 'kernel': 'rbf'}
SVC(C=1000, gamma=1)
```

Evaluation for Random Forest Regressor :

```
Accuracy: 0.9702970297029703
Precision: 0.9705882352941176
Recall: 0.9428571428571428
F1: 0.9565217391304348
```

Evaluation for Decision Tree :

```
Accuracy: 0.9801980198019802
Precision: 0.9714285714285714
Recall: 0.9714285714285714
F1: 0.9714285714285714
```

Evaluation for KNN :

```
Accuracy: 0.9603960396039604
Precision: 0.9696969696969697
Recall: 0.9142857142857143
F1: 0.9411764705882354
```

Evaluation for SVM :

```
Accuracy: 0.9702970297029703
Precision: 0.9705882352941176
Recall: 0.9428571428571428
F1: 0.9565217391304348
```

3-Comparison between the models :

	labels	Accuracy	Precision	Recall	F1
0	SVM	0.970297	0.970588	0.942857	0.956522
1	KNN	0.960396	0.969697	0.914286	0.941176
2	Decision Trees	0.980198	0.971429	0.971429	0.971429
3	Random Forest	0.970297	0.970588	0.942857	0.956522

Since our dataset is imbalanced , our metric is the F1 score.
=> Decision Trees has the highest F1-score so it is the best model to deploy.

Conclusion :

In this chapter, we went in detail through our process for modeling according to the second article and applying Recursive Feature Elimination on different models so we can get the best results.

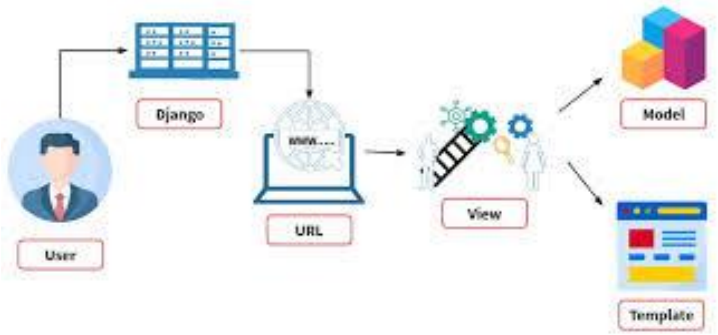
IV. Deployment :

Building your own machine learning model is great, but using it for others is even more appealing. But users may or may not be familiar with the technology, and may or may not have a technical background.
So we used Django to deploy our model.

Django is a high-level Python web framework that encourages rapid development and clean, pragmatic design. Built by experienced developers, it takes care of much of the hassle of web development, so you can focus on writing your app without needing to reinvent the wheel. It’s free and open source.

- Ridiculously fast.
  Django was designed to help developers take applications from concept to completion as quickly as possible.
- Reassuringly secure.
  Django takes security seriously and helps developers avoid many common security mistakes.

Architecture mvt

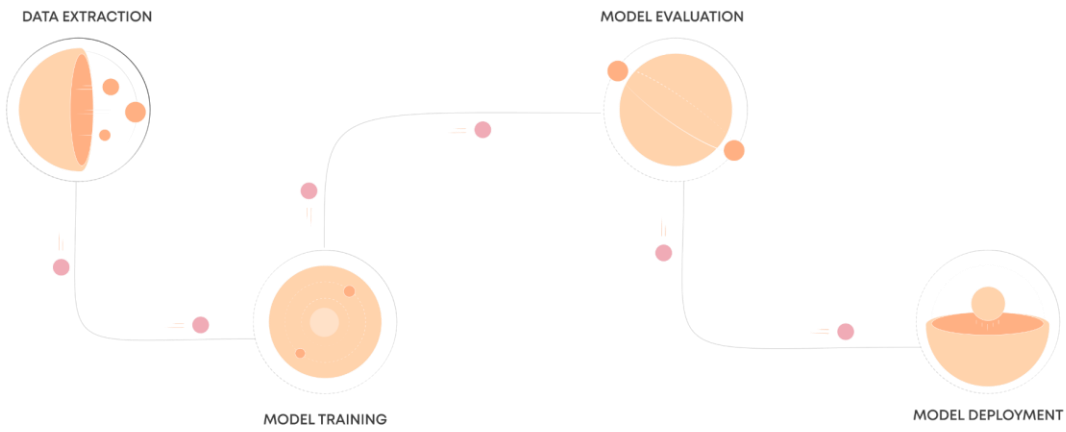


Django is based on MVT (Model-View-Template) architecture. MVT is a software design pattern for developing a web application.

MVT Structure has the following three parts :

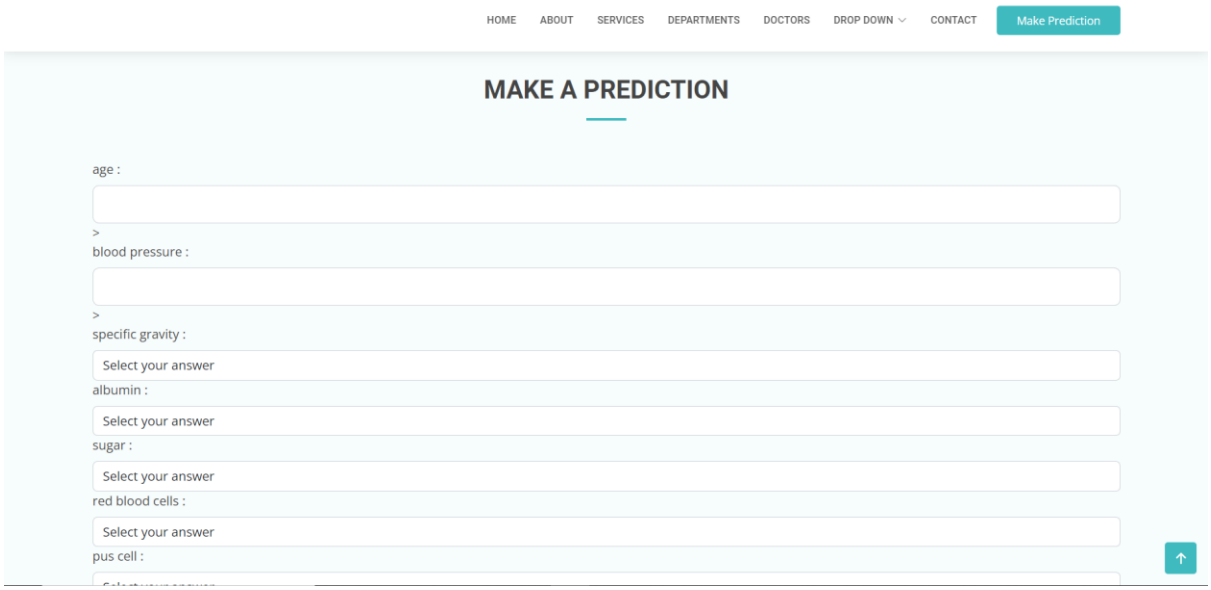
- Model: The model is going to act as the interface of your data.
- View: The View is the user interface — what you see in your browser when you render a website. It is represented by HTML/CSS/Javascript .
- Template: A template consists of static parts of the desired HTML output as well as some special syntax describing how dynamic content will be inserted.

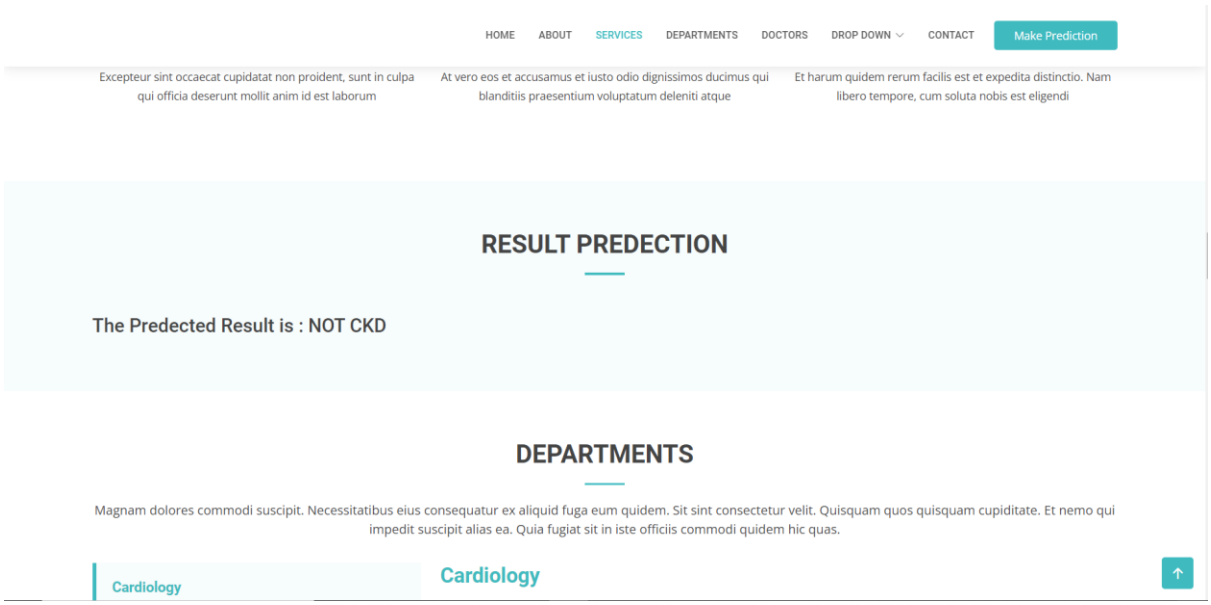
We used also pipeline



Pipelines are important in deployment because they provide a consistent and reproducible way to build, train, and deploy machine learning models. A pipeline typically consists of a series of steps or stages that are designed to prepare and preprocess data, train a model, and make predictions. By using a pipeline, you can ensure that the same steps are followed every time you deploy a model, which can help to reduce the risk of errors and improve the reliability of the results. In addition, pipelines can make it easier to automate the deployment process, so you can deploy new models or update existing ones more quickly and efficiently. Finally, pipelines can help to improve the performance and scalability of a model by allowing different steps

to be run in parallel, which can help to reduce the overall time required to make predictions.





General conclusion :

To conclude ,this project was designed to diagnose chronic kidney disease based on 24 attributes which includes symptoms, signs and risk factors of chronic kidney disease.
We reproduced two scientific articles .
This project was generally conducted into five main stages, namely business understanding, data understanding , data preprocessing , modeling, evaluation and deployment.