Chronic Kidney Disease Prediction

A Project Report submitted

in partial fulfillment of the requirements

for the award of the degree of

BACHELOR OF TECHNOLOGY

In

COMPUTER SCIENCE & ENGINEERING

By

1. K.L.N Pravallika(17B01A0567)
2. M Dedeepya (17B01A0585)
3. D. Srilakshmi (17B01A0591)
4. N.S.L Meghana (17B01A05A4)

Under the esteemed guidance of

Mr. K. Bhadrachalam$_{\text{M.Tech. (Ph.D.)}}$

Assistant Professor

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

SHRI VISHNU ENGINEERING COLLEGE FOR WOMEN(A)

(Approved by AICTE, Accredited by NBA & NAAC, Affiliated to JNTU Kakinada)

BHIMAVARAM – 534 202

2020 – 2021

SHRI VISHNU ENGINEERING COLLEGE FOR WOMEN(A)

(Approved by AICTE, Accredited by NBA & NAAC, Affiliated to JNTU Kakinada)

BHIMAVARAM – 534 202

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

CERTIFICATE

*This is to certify that the project entitled **"Chronic Kidney Disease Prediction"**, is being submitted by **K.L.N Pravallika** bearing the Regd. No. **17B01A0567**, **M.Dedeepya** bearing the Regd.No.**17B01A0585**, **D.Srilakshmi** bearing the Regd.No.**17B01A0591**, **N.S.L Meghana** bearing the Regd.No.**17B01A05A4** in partial fulfillment of the requirements for the award of the degree of "**Bachelor of Technology** in **Computer Science & Engineering**" is a record of bonafide work carried out by her under my guidance and supervision during the academic year 2020–2021 and it has been found worthy of acceptance according to the requirements of the university.*

Internal Guide **Head of the Department**

External Examiner

ACKNOWLEDGEMENTS

Behind every Achievement there lies an unfathomable sea of gratitude to those who activated it, without whom it would ever come into existence. To them, we lay a word of gratitude imprinted within us.

I wish to place my deep sense of gratitude on **Sri. K. V. Vishnu Raju**, Chairman of SVECW, for his constant support of each progressive work.

I wish to express my sincere thanks to **Dr. G. Srinivasa Rao**, Principal of svecw, for being a source of inspiration and constant encouragement.

I wish to express my sincere thanks to **Dr. P. Srinivasa Raju**, Vice - Principal of SVECW, for being a source of inspiration and constant encouragement.

I am indebted to **Dr.P.Kiran Sree**, Head of the Department, for his Indispensable suggestions.

I wish to express my thanks to our PRC Members **Mrs.P.R.Sudha Rani**, **Dr.D.Dharmaiah, Dr.VVR.Maheswara Rao** for their valuable suggestions to complete our Main Project Successfully in time.

I wish to express my thanks to our project coordinator **Mrs.P.R.Sudha Rani**, for helping in completion of the project in a smooth way.

I wish to express my thanks to our guide **Mr. K. Bhadrachalam**, for his valuable suggestions to complete our Main Project Successfully in time.

Project Associates

K.L.N.Pravallika(17B01A0567)

M.Dedeepya (17B01A0585)

D.Srilakshmi (17B01A0591)

N.S.L.Meghana (17B01A05A4)

ABSTRACT

The kidneys filter waste and excess fluid from the blood. As kidneys fail, waste builds up. Symptoms develop slowly and aren't specific to the disease. Medication helps manage symptoms. In later stages, filtering the blood with a machine (dialysis) or a transplant may be required. Chronic kidney disease, also called chronic kidney failure, describes the gradual loss of kidney function. When chronic kidney disease reaches an advanced stage, dangerous levels of fluid, electrolytes, and wastes can build up in your body.

The project aims to detect CKD using machine learning algorithms by considering the least number of tests or features. We approach this by applying machine learning classifiers on a small dataset of 400 records. To remove the null values, we perform Data Cleaning. To increase the size of the dataset, we perform data augmentation. To reduce the number of features, the association between variables has been studied. Feature selection methods have been applied to the attributes and found that there are some of the attributes which impact mostly to predict the CKD.

Contents

INTRODUCTION

1. Introduction

Chronic Kidney Disease

The kidneys filter waste and excess fluid from the blood. As kidneys fail, waste builds up. Symptoms develop slowly and aren't specific to the disease. Medication helps manage symptoms. In later stages, filtering the blood with a machine (dialysis) or a transplant may be required.

Chronic kidney disease, also called chronic kidney failure, describes the gradual loss of kidney function. When chronic kidney disease reaches an advanced stage, dangerous levels of fluid, electrolytes, and wastes can build up in your body.

In the early stages of chronic kidney disease, you may have few signs or symptoms. Chronic kidney disease may not become apparent until your kidney function is significantly impaired.

Treatment for chronic kidney disease focuses on slowing the functioning of the kidney. Chronic kidney disease can progress to end-stage kidney failure, which is fatal without artificial filtering (dialysis) or a kidney transplant.

Stages of Chronic Kidney Disease based on estimated Glomerular Filtration Rate (eGFR)

Stage 1 CKD: eGFR 90 or Greater

Stage 2 CKD: eGFR Between 60 and 89

Stage 3 CKD: eGFR Between 30 and 59

Stage 4 CKD: eGFR Between 15 and 29

Stage 5 CKD: eGFR Less than 15

Symptoms:

- Vomiting
- Loss of appetite
- Fatigue and weakness
- Sleep problems
- Urinary issues
- Muscle cramps
- Swelling of feet and ankles
- Itching
- Chest pain, if fluid builds up around the lining of the heart
- Breathing Issues
- High blood pressure

Causes:

- Type 1 or type 2 diabetes
- High blood pressure
- Inflammation of the kidney's filtering units (eGFR)
- Inflammation of the kidney's tubules and surrounding structures
- Prolonged obstruction of the urinary tract, from conditions such as enlarged prostate, kidney stones, and some cancers
- A condition that causes urine to back up into your kidneys
- Recurrent kidney infection

Complications:

- Fluid retention (pulmonary edema)
- A sudden rise in potassium levels in your blood
- Heart and blood vessel (cardiovascular) disease
- Weak bones and an increased risk of bone fractures
- Anemia
- Damage to your central nervous system
- Irreversible damage to your kidneys (end-stage kidney disease)

SYSTEM ANALYSIS

2. System Analysis

2.1. Existing System:

Urine and blood tests are used to detect and monitor kidney disease. Currently, the key markers used include abnormal urine albumin levels and a persistent reduction in the estimated glomerular filtration rate (eGFR). Diabetes and hypertension are the leading causes of CKD in adults. Many diseases that cause kidney failure may have their origins in childhood. Early detection and appropriate treatment may improve prediction in all age groups.

How do I know if I have CKD?

1. eGFR (estimated glomerular filtration rate)
The eGFR is a sign of how well your kidneys are cleaning your blood. Your body makes waste all the time. This waste goes into your blood. Healthy kidneys take the waste out of your blood. One type of waste is called creatinine. If you have too much creatinine in your blood, it might be a sign that your kidneys are having trouble filtering your blood. You will have a blood test to find out how much creatinine is in your blood. Your doctor will use this information to figure out your eGFR. If your eGFR is less than 60 for three months or more, you might have kidney disease.

2. Urine test
This test is done to see if there is blood or protein in your urine (pee). Your kidneys make your urine. If you have blood or protein in your urine, it may be a sign that your kidneys are not working well.

3. Blood pressure
This test is done to see how hard your heart is working to pump your blood. High blood pressure can cause kidney disease, but kidney disease can also cause high blood pressure. Sometimes high blood pressure is a sign that your kidneys are not working well. For most people, normal blood pressure is less than 120/80 (120 over 80).

Limitations of Existing System:

- ➔ Scope for redundancy
- ➔ Time Delay
- ➔ Less accuracy
- ➔ Needs more human effort
- ➔ Requires more laboratory

2.2. Proposed System:

The proposed system focuses on predicting this life-threatening disease Chronic Kidney Disease(CKD) using Classification algorithms.

The proposed system is automation for Chronic Kidney Disease prediction using classification techniques and supervised learning algorithms.

Supervised Learning: Supervised learning is when the model is getting trained on a labeled dataset. The labeled dataset has both input and output parameters.

Algorithms:

Naive Bayes algorithm: It is a classification technique based on Bayes' Theorem with an assumption of independence among predictors. In simple terms, a Naive Bayes classifier assumes that the presence of a particular feature in a class is unrelated to the presence of any other feature.

Support Vector Machine: The goal of the SVM algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in the future. This best decision boundary is called a hyperplane. SVM chooses the extreme points/vectors that help in creating the hyperplane. These extreme cases are called Support Vectors.

Random Forest: It is a Supervised Learning algorithm. It is based on the concept of ensemble learning, which combines multiple classifiers to solve a complex problem and improve the performance of the model. As the name suggests, "Random Forest is a classifier that contains several decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset."

2.3. Feasibility Study:

A feasibility study is an assessment of the practicality of a proposed project or system. The different feasibilities that have to be analyzed are

- Operational Feasibility
- Economic Feasibility
- Technical Feasibility
- Social Feasibility

• Operational Feasibility:

Operational feasibility deals with the study of aspects of the system to be developed. This system operationally predicts whether the person is suffering from CKD or NON-CKD. This kind of prediction involves less intake from the user, which is highly appropriate. Based on the study, the system is proved to be operationally feasible.

• Economic Feasibility:

Economic feasibility is an assessment of the economic justification for the computer-based project. As hardware was installed from the beginning and for lots of purposes, the hardware cost is low. So, the project is economically feasible.

• Technical Feasibility:

Technical Feasibility is the process of validating the technology assumptions, architecture, and design of a product or project. As we need a Machine Learning environment for building the model, it was created in the Google Colaboratory. We can also use a jupyter notebook for running the machine learning model. We used Flask for developing User Interfaces. Thus, the project is technically feasible.

• Social Feasibility:

Social feasibility is one of the feasibility studies where the acceptance of the people considered regarding the product to be launched. It describes the effect on users from the introduction of the new system considering whether there will be a need for retraining the workforce. It also ensures the process of training the user to use the system efficiently.

SYSTEM REQUIREMENTS SPECIFICATION

3. System Requirements Specification

3.1. Software Requirements:

- Operating system: Windows 8/ Windows 10
- Language: Python
- Dataset: MS Excel (CSV-” Comma-Separated Value”)
- Software IDE/Online working tool: Anaconda Navigator/Google Colaboratory
- Documentation: Microsoft Office

3.2. Hardware Requirements:

- Processor: i5
- RAM: 8 GB
- Hard Disk: 1 TB

3.3. Functional Requirements:

- Collection of data set.
- Data Pre-processing.
- Data Visualization
- Data Augmentation
- Feature Selection
- Working on various Machine Learning algorithms.
- The system can predict whether the person is suffering from disease or not.

SYSTEM DESIGN

4. System Design:

4.1. Introduction

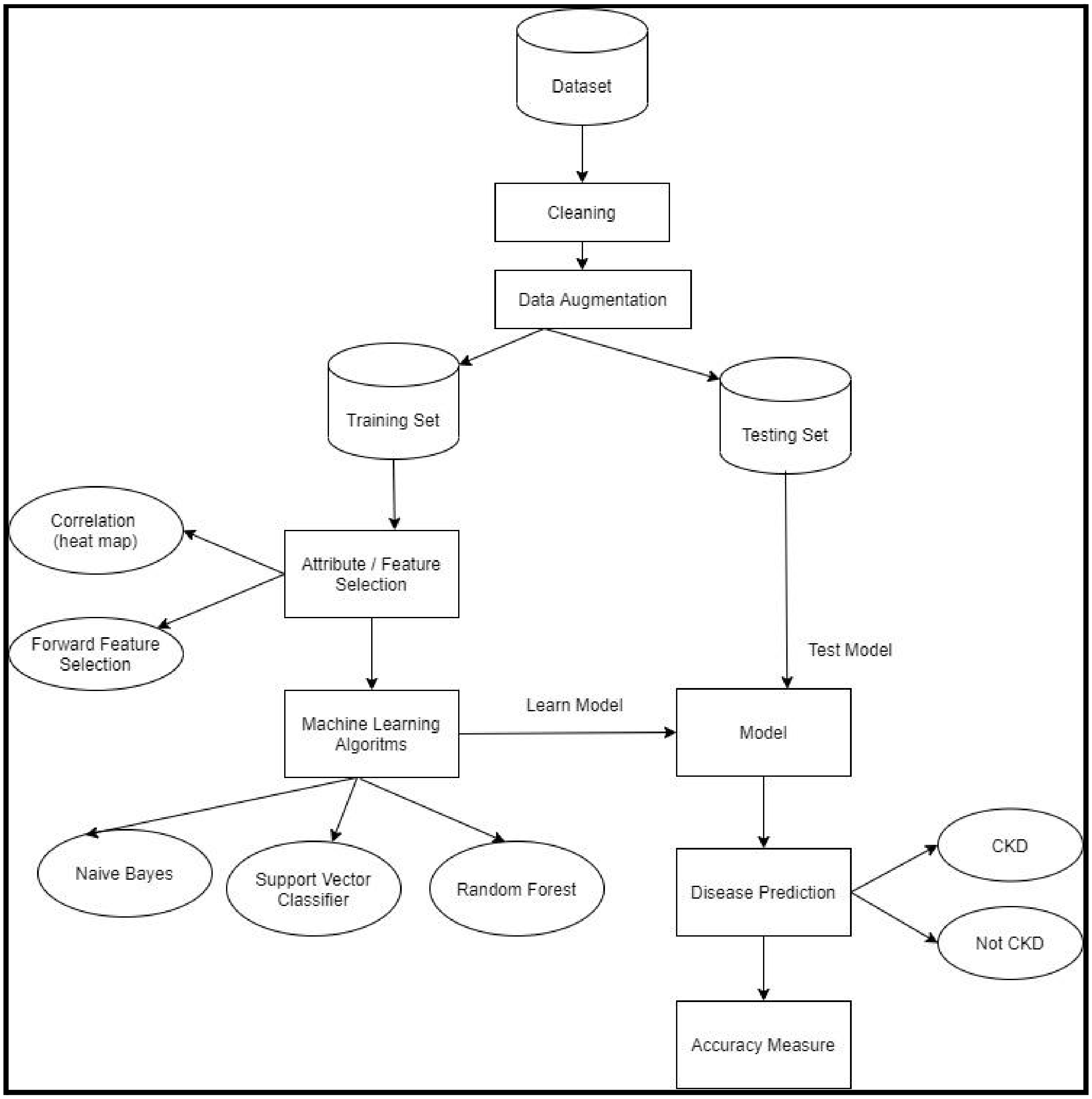
Design is the first step in the development phase of an engineering product or system. Design is the place where quality is considered in software development. Design is the only way that we can accurately translate user requirements into finished software products or systems. Software design serves as the foundation for all software engineers and software maintenance that steps follow. Without design, we risk building an unstable design that will fail when small changes are made, one that may be difficult to test, and one whose quantity cannot be assessed until late in the software engineering process.

System design is the process of designing the elements of a system such as the architecture, modules, and components, the different interfaces of those components, and the data that goes through that system.

System Analysis is the process that decomposes a system into its component pieces to define how well those components interact to accomplish the set requirements.

The System Design process aims to provide sufficient detailed data and information about the system and its system elements to enable the implementation consistent with architectural entities as defined in models and views of the system architecture.

Firstly, we will be considering a dataset that contains 25 attributes and performing data cleaning methods. After Cleaning we perform the Data Augmentation technique to increase the size of the dataset. Later, we divide the dataset into a training set and testing set(80-20; 70-30; 60-40). Further, we carry out feature selection on the training data by implementing filter methods and wrapper methods.
After this, we will apply machine learning algorithms such as Random forests, Support Vector Machine, Naive Bayes, and then will achieve a training model, using this we will be predicting the disease either it is CKD or non-CKD by considering the accuracy measure of the algorithms.



System Architecture

4.2. Data Flow (UML Diagrams)

A data flow diagram (DFD) maps out the flow of information for any process or system. It uses defined symbols like rectangles, circles, and arrows, plus short text labels, to show data inputs, outputs, storage points, and the routes between each destination. Data flowcharts can range from simple, even hand-drawn process overviews, to in-depth, multi-level DFDs that dig progressively deeper into how the data is handled. They can be used to analyze an existing system or model a new one. Like all the best diagrams and charts, a DFD can often visually "say" things that would be hard to explain in words, and they work for both technical and non-technical audiences. That's why DFDs remain so popular after all these years. While they work well for data flow software and systems, they are less applicable nowadays to visualizing interactive, real-time, or database-oriented software or systems.
UML (Unified Modeling Language) is a standard language for specifying, visualizing, constructing, and documenting the artifacts of software systems. It is a method for describing the system architecture in detail using the blueprint. We prepare UML diagrams to understand the system in a better and simple way. A single diagram is not enough to cover all the aspects of the system. UML defines various kinds of diagrams to cover most of the aspects of a system.

4.2.1. Use Case Diagram

Use Case Diagrams are used to depict the functionality of a system or a part of a system. They are widely used to illustrate the functional requirements of the system and its interaction with external agents (actors). A use case is basically a diagram representing different scenarios where the system can be used. A use case diagram gives us a high-level view of what the system or a part of the system does without going into implementation details. When the initial task is complete, use case diagrams are modeled to present the outside view.
In brief, the purposes of use case diagrams can be said to be as follows

Used to gather the requirements of a system.
Used to get an outside view of a system.
Identify the external and internal factors influencing the system.

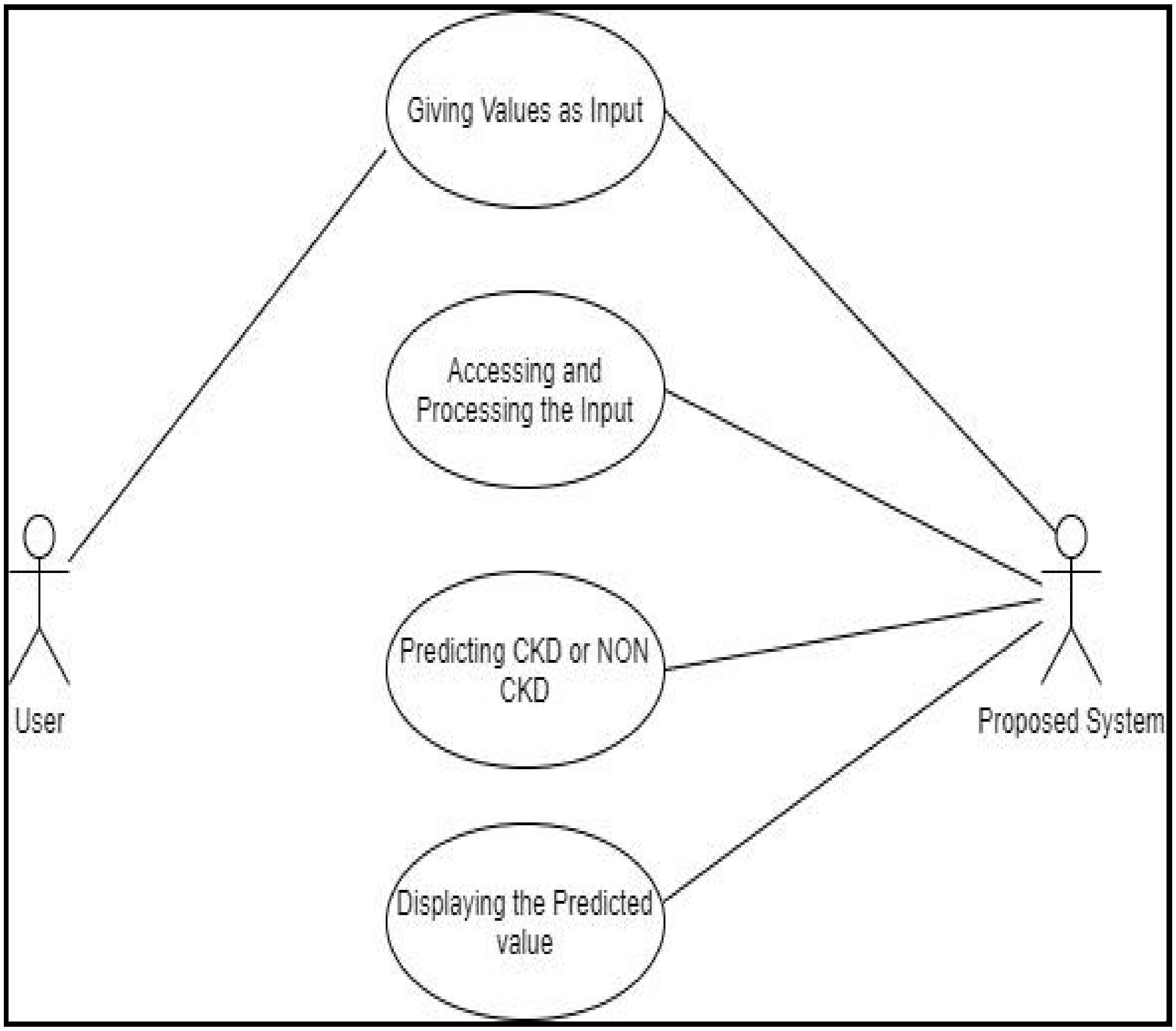
Use case diagrams commonly contain

Use cases
Actors

Dependency, generalization, and association relationships.

Use cases A use case is a software and system engineering term that describes how a user uses a system to accomplish a particular goal.

Actors An actor is a person, organization, or external system that plays a role in one or more interactions with the system.


Giving Values as Input
Accessing and Processing the Input
Predicting CKD or NON CKD
Displaying the Predicted value
User
Proposed System

Use Case Diagram

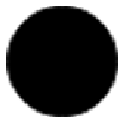
4.2.2. Activity Diagram

An activity diagram is basically a flowchart to represent the flow from one activity to another activity. The activity can be described as an operation of the system. The control flow is drawn from one operation to another. This flow can be sequential, branched, or concurrent. Activity diagrams deal with all types of flow control by using different elements such as fork, join, etc.
The basic purpose of the activity diagram is similar to the other diagrams. It captures the dynamic behavior of the system. Other diagrams are used to show the message flow from one object to another but the activity diagram is used to show message flow from one activity to another. Activity is a particular operation of the system. Activity diagrams are not only used for visualizing the dynamic nature of a system but are also used to construct the executable system by using forward and reverse engineering techniques. The only missing thing in the activity diagram is the message part. It does not show any message flow from one activity to another.
The activity diagram is sometimes considered as the flowchart. Although the diagrams look like a flowchart, they are not. It shows different flows such as parallel, branched, concurrent, and single. The purpose of an activity diagram can be described as −

- ➔ Draw the activity flow of a system.
- ➔ Describe the sequence from one activity to another.
- ➔ Describe the parallel, branched, and concurrent flow of the system.

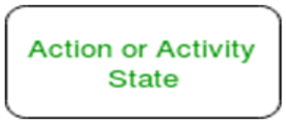
Notations:

Initial State – The starting state before an activity takes place is depicted using the initial state. A process can have only one initial state unless we are depicting nested activities. We use a black-filled circle to depict the initial state of a system.



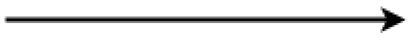
Initial State

Action or Activity State – An activity represents the execution of action on objects or by objects. We represent an activity using a rectangle with rounded corners. Basically, any action or event that takes place is represented using an activity.



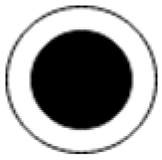
Activity State

Action Flow or Control flows – Action flows or Control flows are also referred to as paths and edges. They are used to show the transition from one activity state to another.

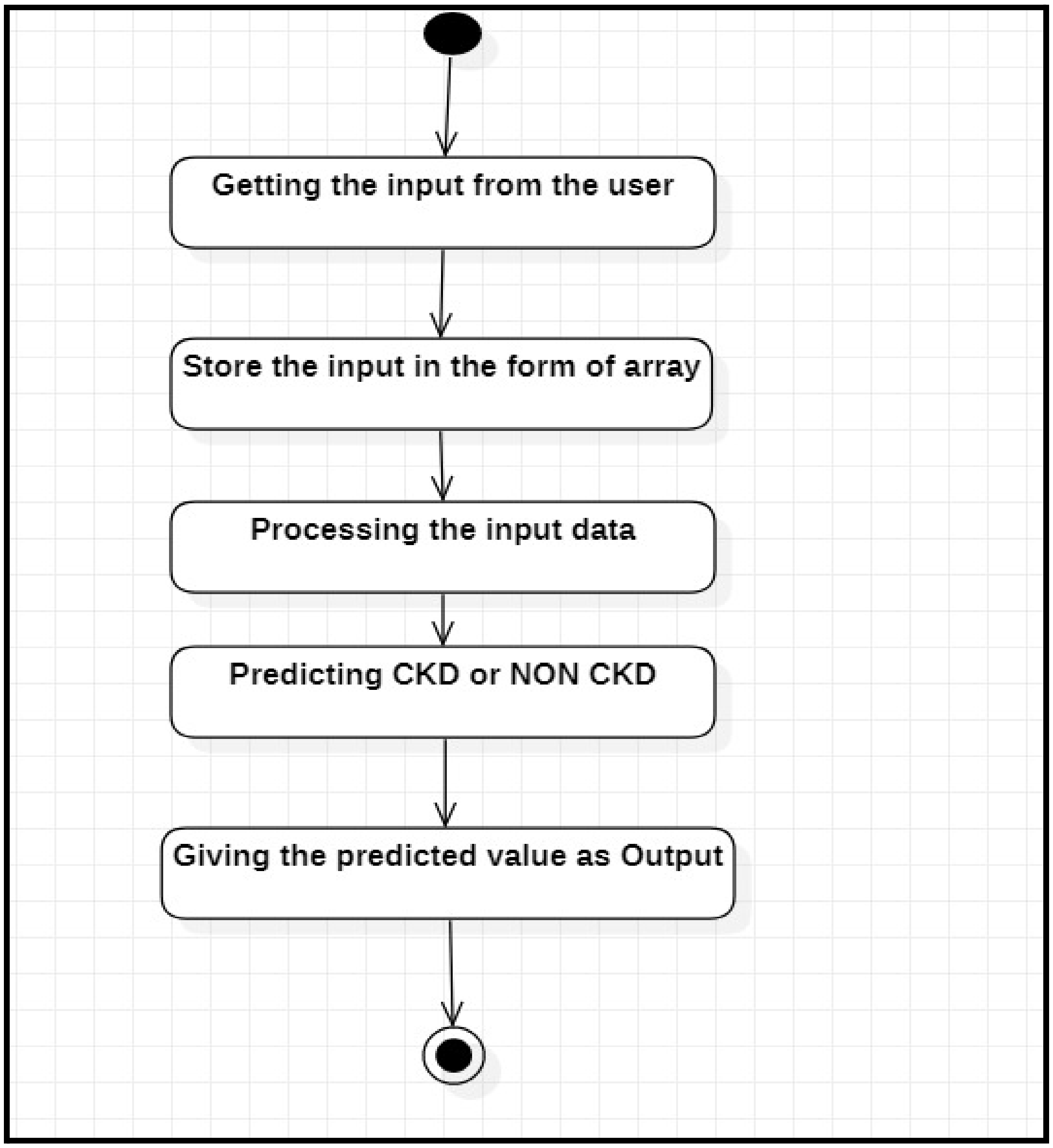


Action Flow

Final State or End State – The state which the system reaches when a particular process or activity ends is known as a Final State or End State. We use a filled circle within a circle notation to represent the final state in a state machine diagram. A system or a process can have multiple final states.



Final State


Getting the input from the user
Store the input in the form of array
Processing the input data
Predicting CKD or NON CKD
Giving the predicted value as Output

Activity Diagram

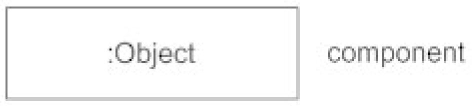
4.2.3. Sequence Diagram

A sequence diagram shows object interactions arranged in a time sequence. It depicts the objects involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario. Sequence diagrams are typically associated with use case realizations in the logical view of the system under development. Sequence diagrams are sometimes called event diagrams or event scenarios.

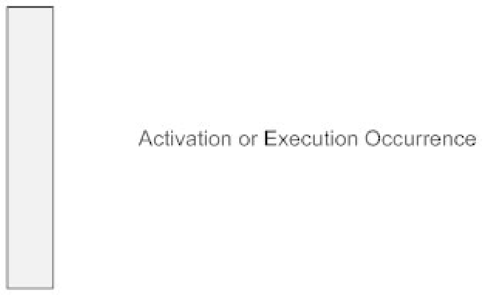
A sequence diagram shows, as parallel vertical lines (lifelines), different processes or objects that live simultaneously, and, as horizontal arrows, the messages exchanged between them, in the order in which they occur. This allows the specification of simple runtime scenarios in a graphical manner.

Sequence Diagram Notations

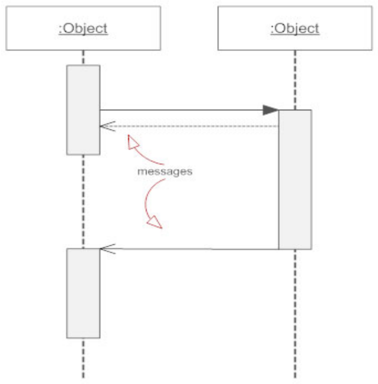
Class Roles or Participants - Class roles describe the way an object will behave in context. Use the UML object symbol to illustrate class roles, but don't list object attributes.



Activation - Activation boxes represent the time an object needs to complete a task. When an object is busy executing a process or waiting for a reply message, use a thin gray rectangle placed vertically on its lifeline.



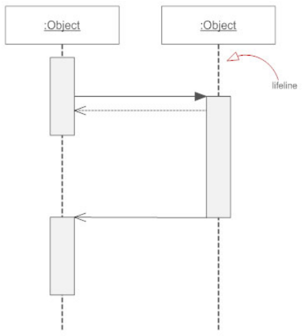
Messages - Messages are arrows that represent communication between objects. Use half-arrowed lines to represent asynchronous messages. Asynchronous messages are sent from an object that will not wait for a response from the receiver before continuing its tasks.



Messages

Lifelines

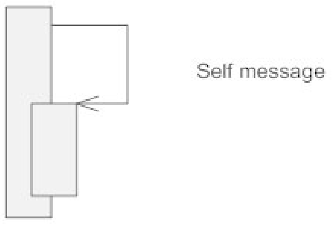
Lifelines are vertical dashed lines that indicate the object's presence over time.

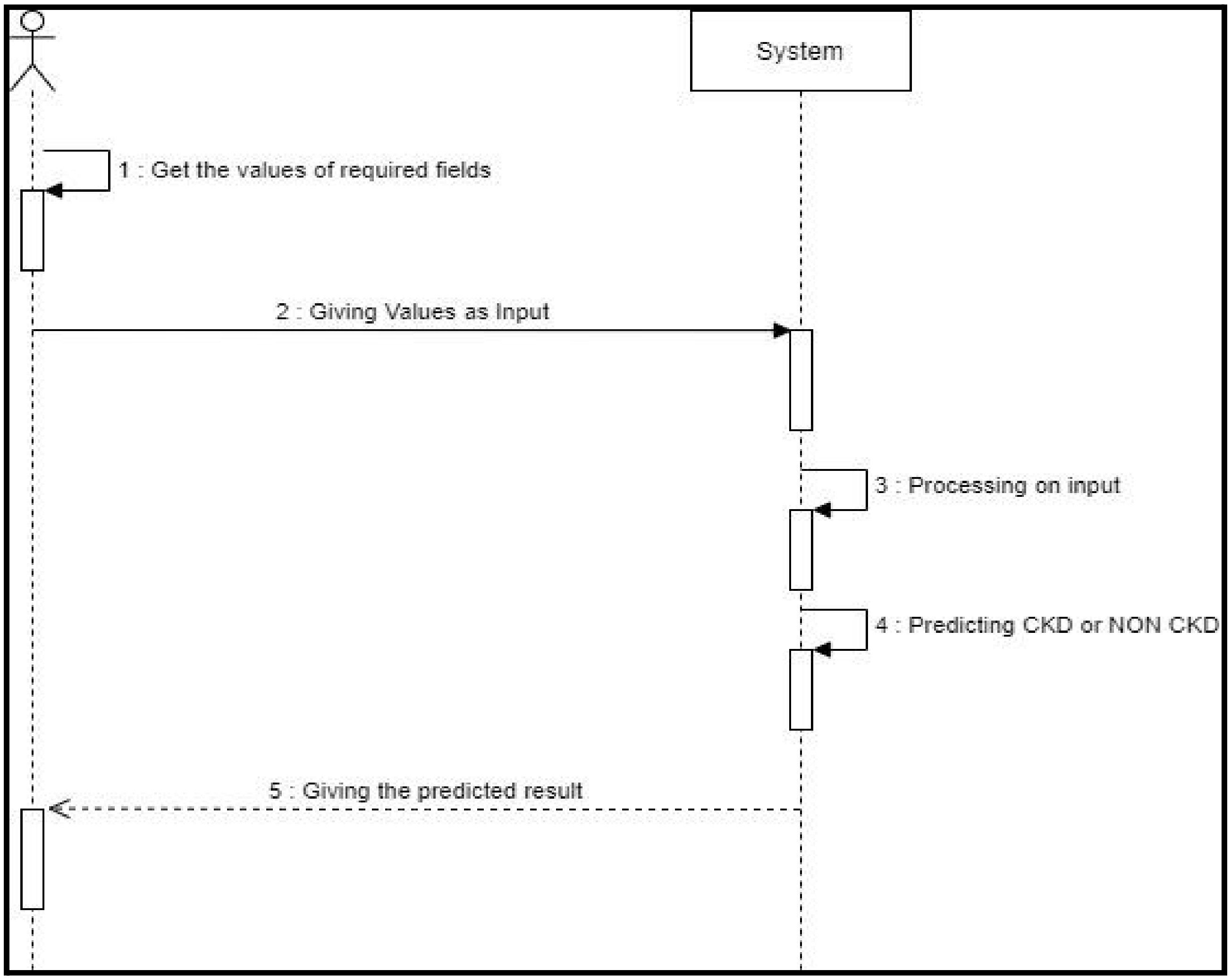


Lifelines

Self Message

A message an object sends to itself, usually shown as a U-shaped arrow pointing back to itself.




System
1 : Get the values of required fields
2 : Giving Values as Input
3 : Processing on input
4 : Predicting CKD or NON CKD
5 : Giving the predicted result

Sequence Diagram

SYSTEM IMPLEMENTATION

5. System Implementation:

5.1. Introduction:

Systems implementation is the process of defining how the information system should be built (i.e., physical system design), ensuring that the information system is operational and used, ensuring that the information system meets quality standards (i.e., quality assurance).

The project aims to detect CKD using Machine Learning algorithms, by considering the least number of tests or features. We approach this by applying Machine Learning Classifiers on a small dataset of 400 records and 26 features. To increase the size of the dataset, Data Augmentation is performed. To reduce the number of features, the association between variables has been studied. Feature Selection Methods are applied to the attributes to find the attributes which impact the most. The resultant features are used for training and testing the model.

- ➔ Our System Implements different modules for the construction of our project. They are Data Collection, Data Visualization, Data Preprocessing, and the algorithms like Support Vector Classifier, Random Forest Classifier, Naive Bayes Classifier.
- ➔ We used libraries like Pandas, Numpy, Matplotlib, Scikit-learn for the implementation.
- ➔ Preprocessing involves the steps like importing libraries, importing datasets, detecting Outliers, finding missing values, filling values, Encoding the Categorical data.
- ➔ We have used three Visualization techniques which include distplot, count plot, and heatmap But for the implementation of our project, we used only two types of data visualization.
- ➔ We perform Data Augmentation to increase the size of the dataset.
- ➔ We have used Flask, HTML, CSS for the front-end implementation.

5.2. Project Modules:

- Data Collection.
- Data Visualization
- Data Preprocessing.
- Data Augmentation
- Dividing the dataset into training data and testing data.
- Feature Selection
- Algorithms with Implementation
- Calculating Accuracies
- User-Interface

5.2.1. Data Collection and Visualization

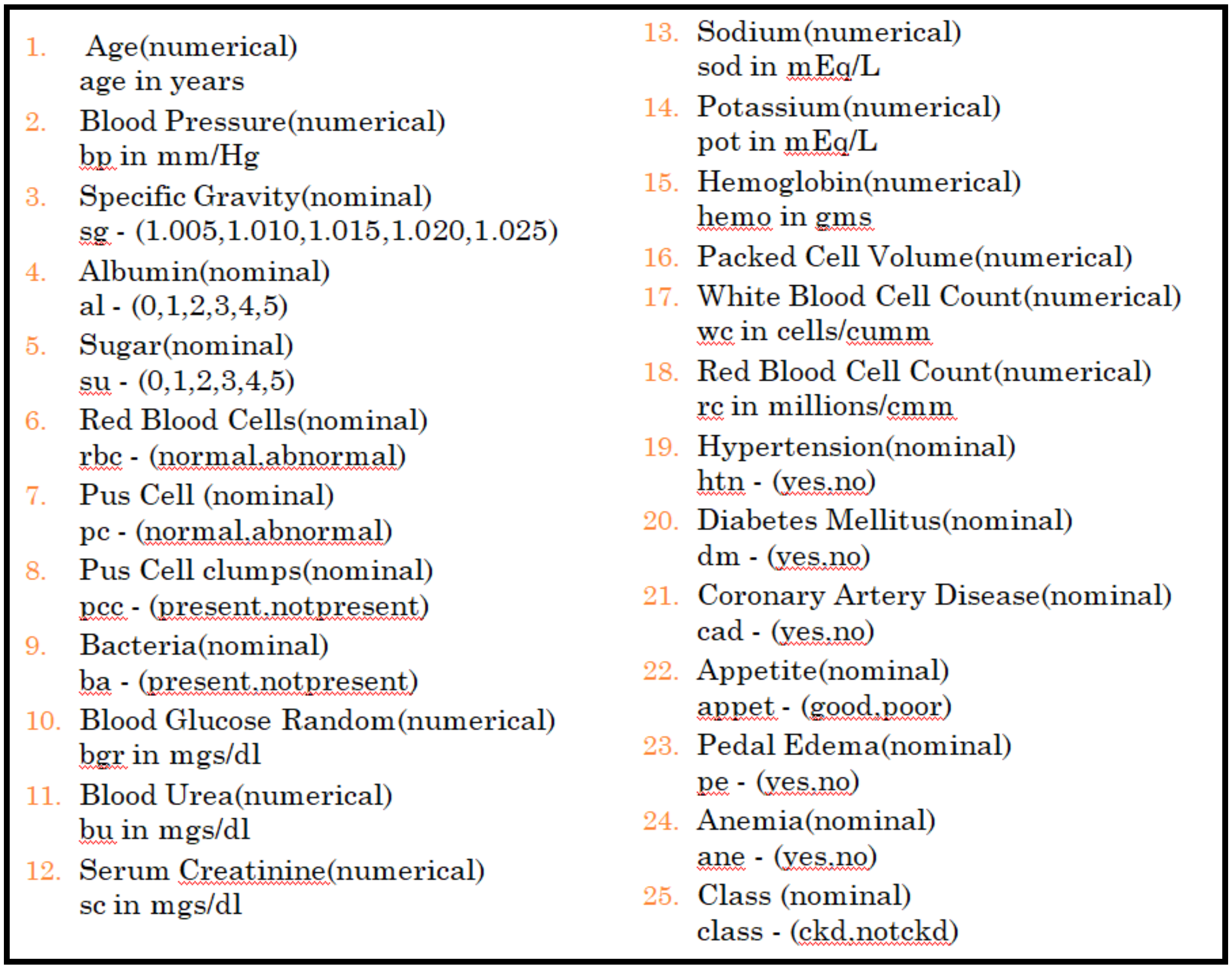
❖ Data Collection:

Data collection is the process of gathering and measuring information from different sources. To use the data we collect to develop machine learning solutions, it must be collected and stored in a way that makes sense. Data collection is the process of gathering quantitative and qualitative information on specific variables to evaluate outcomes or glean actionable insights.

Collecting data allows you to capture a record of past events so that we can use data analysis to find recurring patterns. From those patterns, you build predictive models using machine learning algorithms.

In this project, we are gathering a dataset that consists of data about Chronic Kidney Disease. This dataset includes various features like age, albumin, creatinine, sodium, and many more.

Description of Dataset:



1. Age(numerical)
   age in years
2. Blood Pressure(numerical)
   bp in mm/Hg
3. Specific Gravity(nominal)
   sg - (1.005,1.010,1.015,1.020,1.025)
4. Albumin(nominal)
   al - (0,1,2,3,4,5)
5. Sugar(nominal)
   su - (0,1,2,3,4,5)
6. Red Blood Cells(nominal)
   rbc - (normal,abnormal)
7. Pus Cell (nominal)
   pc - (normal,abnormal)
8. Pus Cell clumps(nominal)
   pcc - (present,notpresent)
9. Bacteria(nominal)
   ba - (present,notpresent)
10. Blood Glucose Random(numerical)
    bgr in mgs/dl
11. Blood Urea(numerical)
    bu in mgs/dl
12. Serum Creatinine(numerical)
    sc in mgs/dl
13. Sodium(numerical)
    sod in mEq/L
14. Potassium(numerical)
    pot in mEq/L
15. Hemoglobin(numerical)
    hemo in gms
16. Packed Cell Volume(numerical)
17. White Blood Cell Count(numerical)
    wc in cells/cumm
18. Red Blood Cell Count(numerical)
    rc in millions/cmm
19. Hypertension(nominal)
    htn - (yes,no)
20. Diabetes Mellitus(nominal)
    dm - (yes,no)
21. Coronary Artery Disease(nominal)
    cad - (yes,no)
22. Appetite(nominal)
    appet - (good,poor)
23. Pedal Edema(nominal)
    pe - (yes,no)
24. Anemia(nominal)
    ane - (yes,no)
25. Class (nominal)
    class - (ckd,notckd)

Specific gravity (sg) - Specific gravity is a urinalysis parameter commonly used in the evaluation of kidney function. The concentration of the secreted molecules determines the urine's specific gravity. 1.010 to 1.030. Increases in specific gravity i.e. increased concentration of solutes in the urine. Decreased specific gravity, i.e. decreased concentration of solutes in urine.

Albumin (al) – It is a protein which is part of blood. It should be present in our body and not in urine. If kidneys are well functioning you should have less amount of protein in urine.

Sugar (su) - (glucose) is usually present in the urine at very low levels or not at all. Abnormally high amounts of sugar in the urine, usually the result of high blood sugar levels. High blood sugar usually occurs in diabetes, especially when untreated.

Normally, when blood is filtered in the kidneys, some sugar remains in the fluid that will later become urine. If the level of blood sugar is low, as is normally the case, the body can reabsorb the sugar from this fluid before it leaves the kidney to be excreted as urine. When the blood sugar is high, there is too much sugar in the fluid leaving the kidney to be reabsorbed, so some sugar passes into the urine.

Red Blood Cells (rbc) – Normal means kidneys are functioning well. If abnormal, there are further tests which have to be taken place like rbc count, wbc count, etc.,

Pus cells (pc) - The presence of pus cells in urine is called pyuria. Urine to look cloudy or as if it contains pus. The urinary tract infection (UTI). In rare cases, it can be a sign of a complicated UTI or sepsis.

Pus cell Clumps (pcc) - clumps in urine indicate the presence of an infection or inflammation, especially in the kidney

Bacteria (ba) - Bacteria that enter your urinary tract through the tube that carries urine from your body (urethra) can multiply and travel to your kidneys. This is the most common cause of kidney infections. Bacteria from an infection elsewhere in your body also can spread through your bloodstream to your kidneys. Normal urine has no bacteria. But if bacteria travel into the bladder, a UTI can occur. The infection most often starts in the bladder but can spread to the kidneys.

Random glucose (bgr) - Blood glucose random testing measures the levels of glucose in the blood at any given point in the day. For a random glucose test, a result of 200 mg/dL or above indicates that a person may have diabetes. However, for a more reliable diagnosis, the doctor will usually repeat the test on another day. The normal range is between 80 to 140.

Blood Urea (bu) - Urea is a waste product of metabolism that is excreted by the kidneys in urine. Kidney disease is associated with reduced urea excretion and a consequent rise in blood concentration.

Serum Creatinine (sc) - Creatinine is a chemical compound leftover from energy-producing processes in your muscles. Healthy kidneys filter creatinine out of the blood. Creatinine exits your body as a waste product in urine. The kidneys are responsible for keeping the level of creatinine in the blood within a normal range. The typical reference range for serum creatinine is 0.7 to 1.2 milligrams per deciliter (mg/dL) for men and 0.5 to 1.0 mg/dL for women.

Sodium (sod) - Sodium balance in patients with renal failure varies with the severity and clinical manifestations of renal disease. Progressive chronic renal insufficiency is typified by an adaptive increase in the sodium excretion rate per nephron as the total glomerular filtration rate declines. The normal blood sodium level is 135 to 145 milliequivalents/liter (mEq/L). Hyponatremia occurs when your blood sodium level goes below 135 mEq/L.

Potassium (pot) - Blood potassium >5.0 indicates potassium imbalance. Arbitrary thresholds are used to indicate the degree of severity, such as mild (>5.0), moderate (>5.5), and severe (>6.0)

Hemoglobin (hemo) - The normal range for hemoglobin is: For men, 13.5 to 17.5 grams per deciliter. For women, 12.0 to 15.5 grams per deciliter.

Packed Cell Volume (pcv) - The packed cell volume (PCV) is a measurement of the proportion of blood that is made up of cells.

White Blood Cell Count (wbc) - The normal range is usually between 4,000 and 11,000 per microliter of blood. A blood test that shows a WBC count of less than 4,000 per microliter could mean your body may not be able to fight infection the way it should.

Red Blood Cell Count (rbc) - The normal RBC range for men is 4.7 to 6.1 million cells per microliter (mcL). The normal RBC range for women who aren't pregnant is 4.2 to 5.4 million mcL. The normal RBC range for children is 4.0 to 5.5 million mcL.

Hypertension (htn) - Hypertension also known as high blood pressure (HBP), is a long-term medical condition in which the blood pressure in the arteries is persistently elevated. High blood pressure typically does not cause symptoms.

Diabetes Mellitus (dm) - Diabetes mellitus is a disorder in which the amount of sugar in the blood is elevated.

Coronary Artery Disease (cad) - Coronary artery disease is the buildup of plaque in the arteries that supply oxygen-rich blood to your heart. Plaque causes a narrowing or blockage that could result in a heart attack. The outcomes of CAD are poorer in patients with CKD.

Appetite (appet) - Appetite is the desire to eat food, sometimes due to hunger. The progressive decline of the glomerular filtration rate in chronic kidney disease patients is associated with a significant reduction in food intake.

Pedal edema (pe) - is the accumulation of fluid in the feet and lower legs. When you have kidney disease, extra fluid and sodium in your circulation may cause edema. The edema associated with kidney disease usually occurs in your legs and around your eyes.

Anemia (ane) - Anemia of chronic disease, also called the anemia of inflammation, is a condition that can be associated with many different underlying disorders including chronic illnesses such as cancer, certain infections, and autoimmune and inflammatory diseases such as rheumatoid arthritis or lupus.

❖ Data Visualization:

Data Visualization is the presentation of data in graphical format. It helps people understand the significance of data by summarizing and presenting a huge amount of data in a simple and easy-to-understand format and helps communicate information clearly and effectively.

Need of Visualization:

We need data visualization because a visual summary of information makes it easier to identify patterns and trends than looking through thousands of rows on a spreadsheet. It’s the way the human brain works. Since the purpose of data analysis is to gain insights, data is much more valuable when it is visualized. Even if a data analyst can pull insights from data without visualization, it will be more difficult to communicate the meaning without visualization. Charts and graphs make communicating data findings easier even if you can identify the patterns without them.

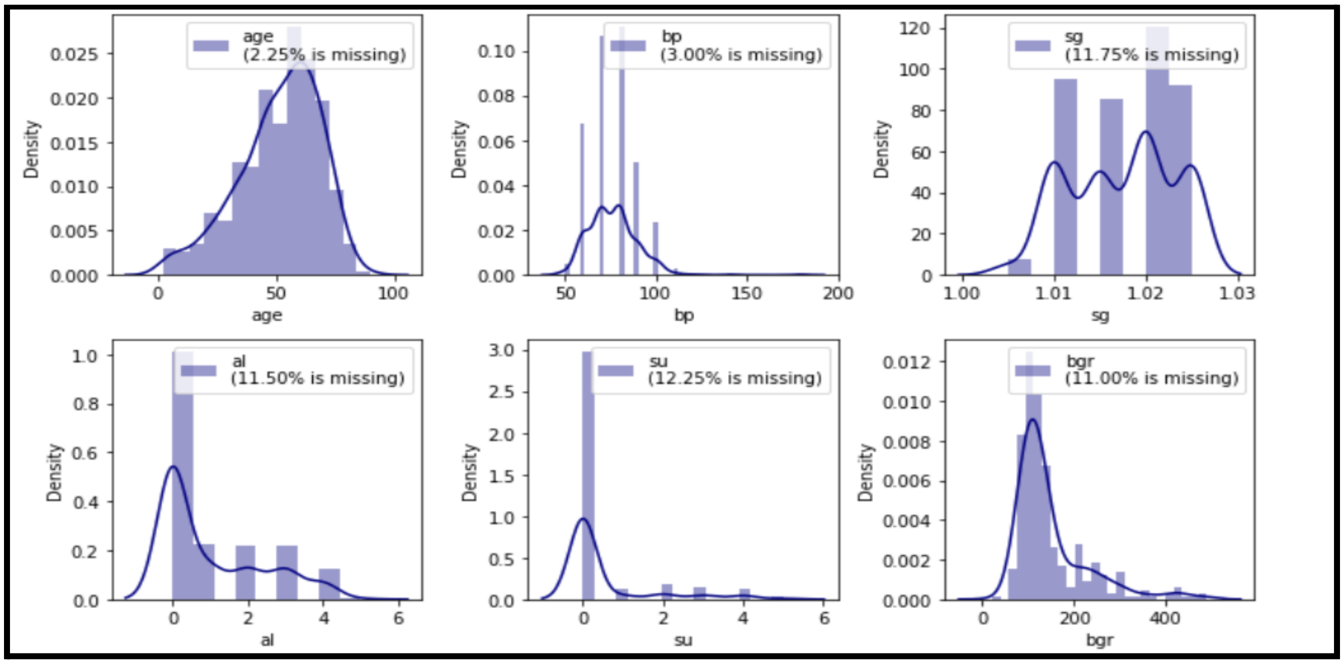
Visualization Techniques

There are many types of data visualization. But for the implementation of our project, we used only some of them.

- **Seaborn Distplot**: A **Distplot** or distribution plot, depicts the variation in the data distribution. Seaborn Distplot represents the overall distribution of continuous data variables. The Seaborn module along with the Matplotlib module is used to depict the distplot with different variations in it. The Distplot depicts the data by a histogram and a line in combination with it.

```
plt.subplot(236)
miss="%.2f"%(100*(1-(data['bgr'].dropna().shape[0])/data.shape[0]))
col='bgr'+"\n({}% is missing)".format(miss)
fig=sns.distplot(data['bgr'], color="navy", label=col, norm_hist=True)
fig=fig.legend(loc='best', fontsize=10)
plt.tight_layout()
plt.show()
```

The below image represents the distplot for age, blood pressure, specific gravity, albumin columns in which, The length of the bar represents the count,
The height of the line represents the density for that numerical value,
The legend represents the percentage of missing values of that particular column.



Distplot

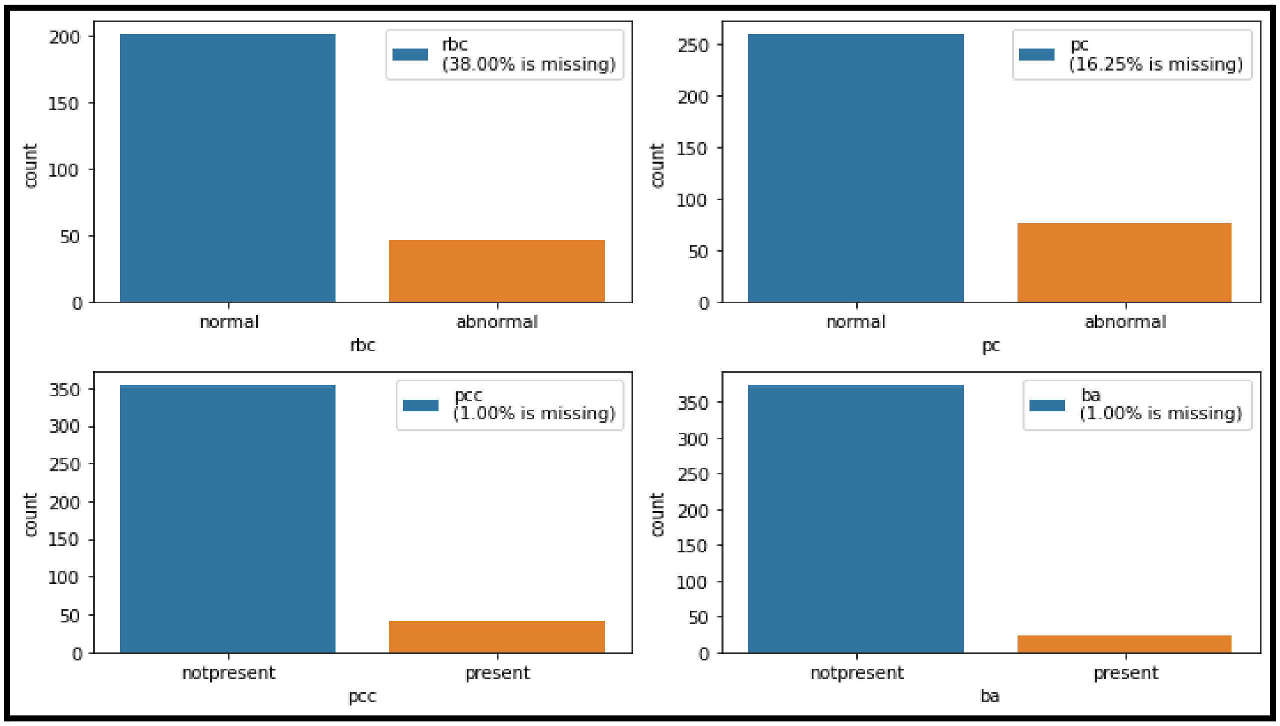
- **Countplot:** seaborn.countplot() method is used to show the counts of observations in each categorical bin using bars. A count plot can be thought of as a histogram across a categorical, instead of a quantitative, variable. The basic API and options are identical to those for **barplot()**, so you can compare counts across nested variables.

```
plt.subplot(224)
miss="%.2f"%(100*(1-(data['ba'].dropna().shape[0])/data.shape[0]))
col='ba'+"\n({}% is missing)".format(miss)
fig=sns.countplot(x='ba',data=data,label=col)
fig=fig.legend(loc='best', fontsize=10)
plt.tight_layout()
plt.show()
```

The below image represent the countplot for red blood cells, pus cell, pus cell clumps, bacteria columns in which, The length of the bar represent the count,

The X-axis represents the variables in that column,

The legend represents the percentage of missing values of that particular column.



Count Plot

- **Heatmap:** Heatmap is defined as a graphical representation of data using colors to visualize the value of the matrix. In this, to represent more common values or higher activities brighter colors basically, reddish colors are used, and to represent less common or activity values, darker colors are preferred. Heatmap is also defined by the name of the shading matrix. Heatmaps in Seaborn can be plotted by using the seaborn.heatmap() function.

5.2.2. Data Preprocessing

❖ Data Preprocessing:

Data preprocessing is a process of preparing the raw data and making it suitable for a machine learning model. It is the first and crucial step while creating a machine learning model.
When creating a machine learning project, it is not always the case that we come across clean and formatted data. And while doing any operation with data, it is mandatory to clean it and put it in a formatted way. So for this, we use data preprocessing tasks. It includes the below steps:

- ➢ Getting the dataset
- ➢ Importing libraries
- ➢ Importing dataset
- ➢ Detecting Outliers
- ➢ Identifying and Handling Missing Values
- ➢ Encoding Categorical Data

➢ **Getting the dataset:**

To build and develop Machine Learning models, you must first acquire the relevant dataset. This dataset will be composed of data gathered from multiple and disparate sources which are then combined in a proper format to form a dataset. Dataset formats differ according to use cases. For instance, a business dataset will be entirely different from a medical dataset. While a business dataset will contain relevant industry and business data, a medical dataset will include healthcare-related data.

➢ **Importing Libraries:**

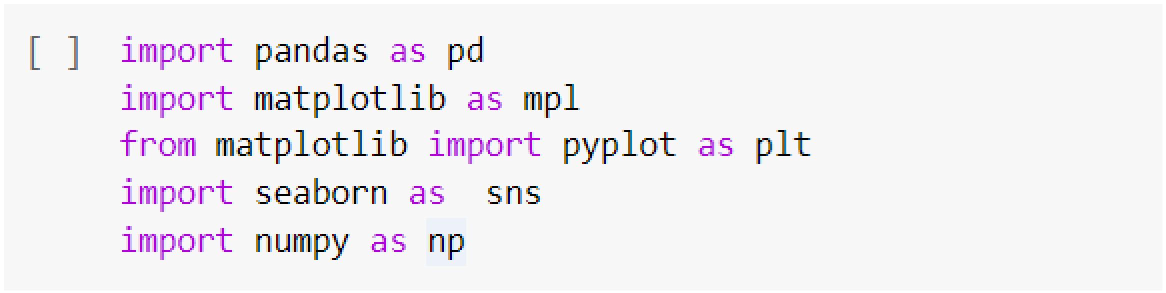
To perform data preprocessing using Python, we need to import some predefined Python libraries. These libraries are used to perform some specific jobs. There are three specific libraries that we will use for data preprocessing, which are:

Numpy: Numpy Python library is used for including any type of mathematical operation in the code. It is the fundamental package for scientific calculation in Python. It also supports adding large, multidimensional arrays and matrices.

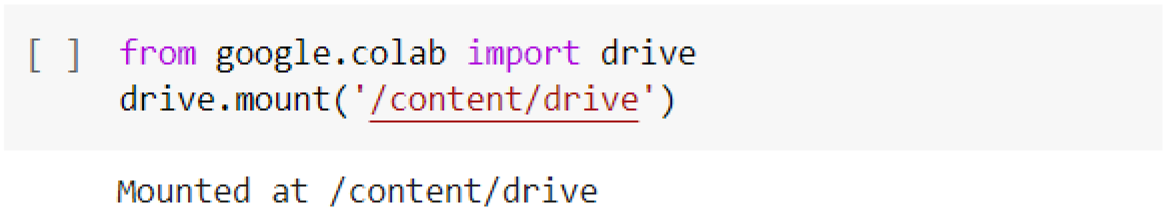
Matplotlib: The second library is **matplotlib**, which is a Python 2D plotting library, and with this library, we need to import a sub-library **pyplot**. This library is used to plot any type of chart in Python for the code.

Pandas: The last library is the Pandas library, which is one of the most famous Python libraries and is used for importing and managing datasets. It is an open-source data manipulation and analysis library.

Seaborn: Seaborn is a Python data visualization library based on matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics.



```
[ ]  import pandas as pd
     import matplotlib as mpl
     from matplotlib import pyplot as plt
     import seaborn as  sns
     import numpy as np
```



```
[ ]  from google.colab import drive
     drive.mount('/content/drive')

     Mounted at /content/drive
```

➢ **Importing Dataset:**

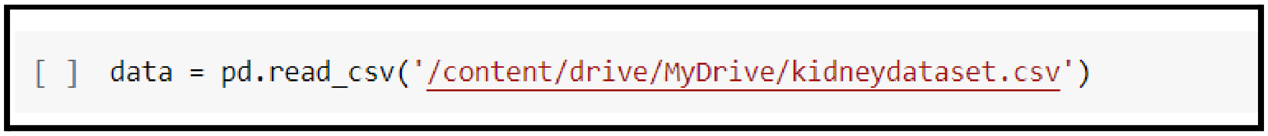
In this step, you need to import the dataset/s that you have gathered for the ML project at hand. We can import the dataset using the below steps:

➔ Gather the dataset.

➔ Upload it in your drive.

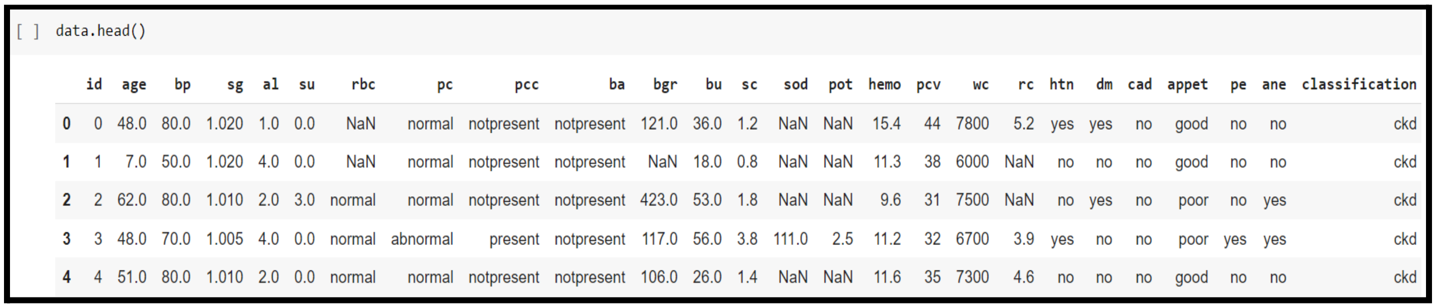
➔ Mount the drive to google colaboratory.

➔ Provide the path of the dataset inside the read_csv() method.



```
[ ]  data = pd.read_csv('/content/drive/MyDrive/kidneydataset.csv')
```

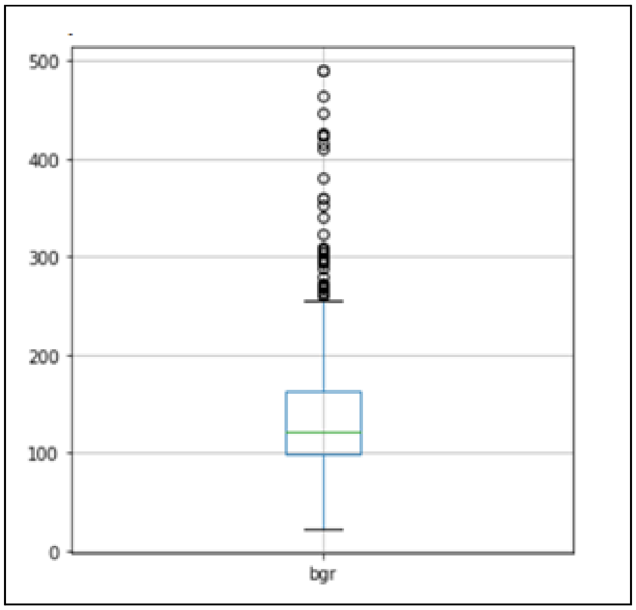
We use **data.head()** for viewing the first 5 rows.



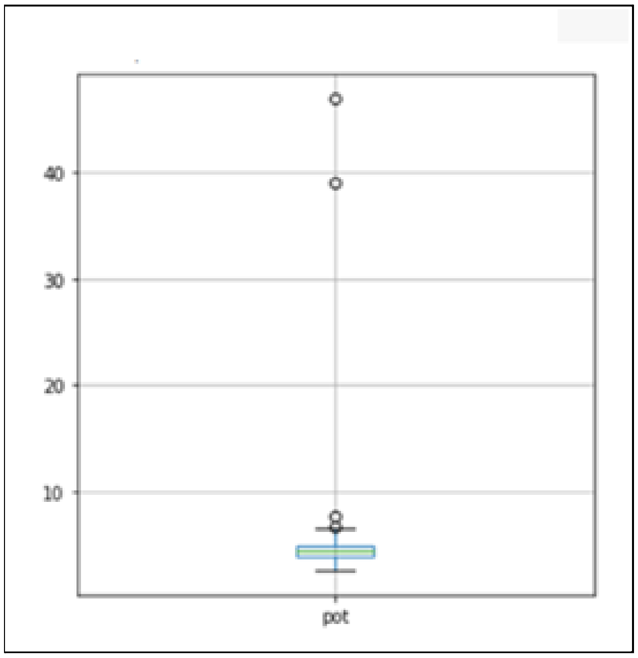
```
[ ]  data.head()

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

➢ **Detecting Outliers:**

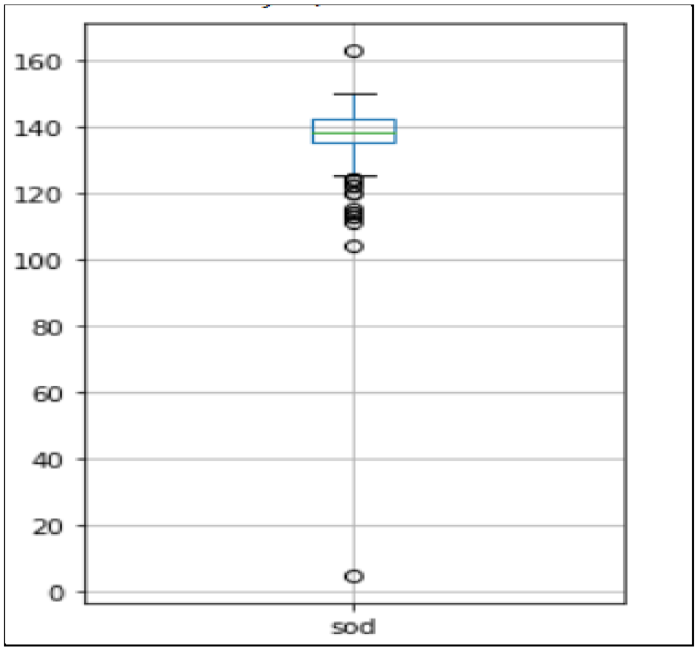
Outliers are extreme values that deviate from other observations on data, they may indicate variability in measurement, experimental errors, or a novelty. In other words, an outlier is an observation that diverges from an overall pattern on a sample. These can be detected by using Boxplot.



The **diagram shows** that there are some outliers detected for blood glucose randomly that reached 500 mg/dl. However, the highest blood glucose level recorded in 2008 for a surviving patient reached 2,656 mg/dl. So, these outliers are legitimate and we should not change them.

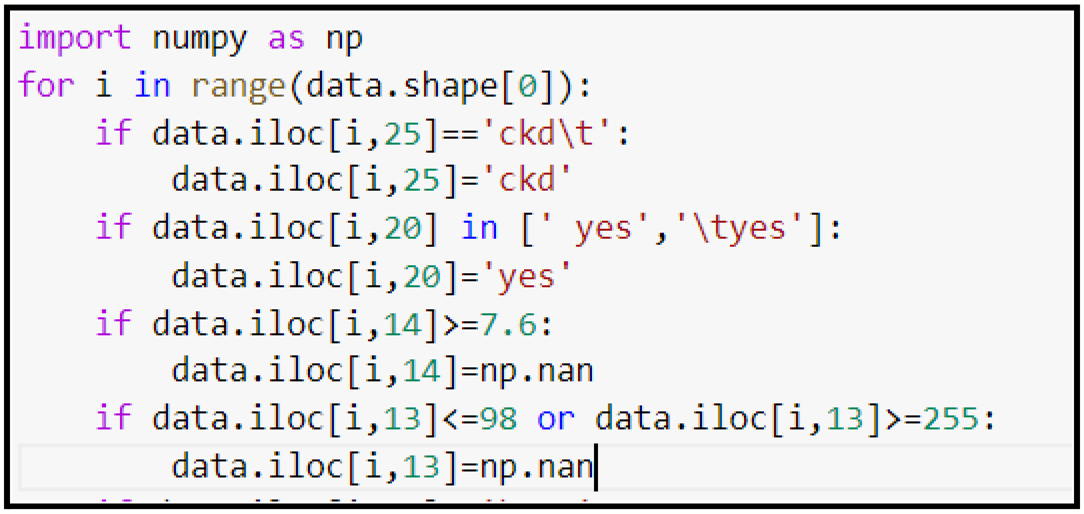


The **above diagram** represents the outliers for potassium. The highest potassium level observed was 7.6 mEq/L. This means that a potassium level with 39 and 47 as shown above is impossible and usually due to a mistake.



The **above diagram** represents the outliers for sodium.Similarly, with sodium, there is one extreme data point detected, which is 4.5. Normally, sodium levels should be between 135 and 145 mEq/L. For this reason, a value of 4.5 is unacceptable or impossible.

Handling Mistyped and Outlier Values:

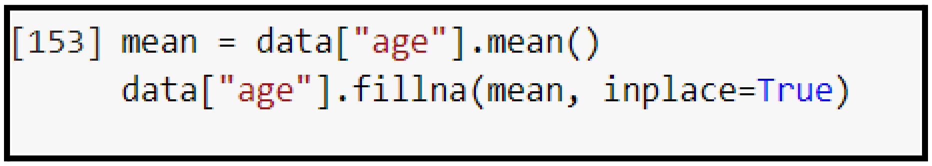


```
import numpy as np
for i in range(data.shape[0]):
    if data.iloc[i,25]=='ckd\t':
        data.iloc[i,25]='ckd'
    if data.iloc[i,20] in [' yes','\tyes']:
        data.iloc[i,20]='yes'
    if data.iloc[i,14]>=7.6:
        data.iloc[i,14]=np.nan
    if data.iloc[i,13]<=98 or data.iloc[i,13]>=255:
        data.iloc[i,13]=np.nan
```

➢ **Identifying and Handling Missing Values:**

In data preprocessing, it is important to identify and correctly handle the missing values, failing to do this, you might draw inaccurate and faulty conclusions and inferences from the data.

➔ **Calculating the mean:** This method is useful for features having numeric data like age, blood pressure, specific gravity, etc. Here, you can calculate the mean of a particular feature or column or row that contains a missing value and replace the result for the missing value.

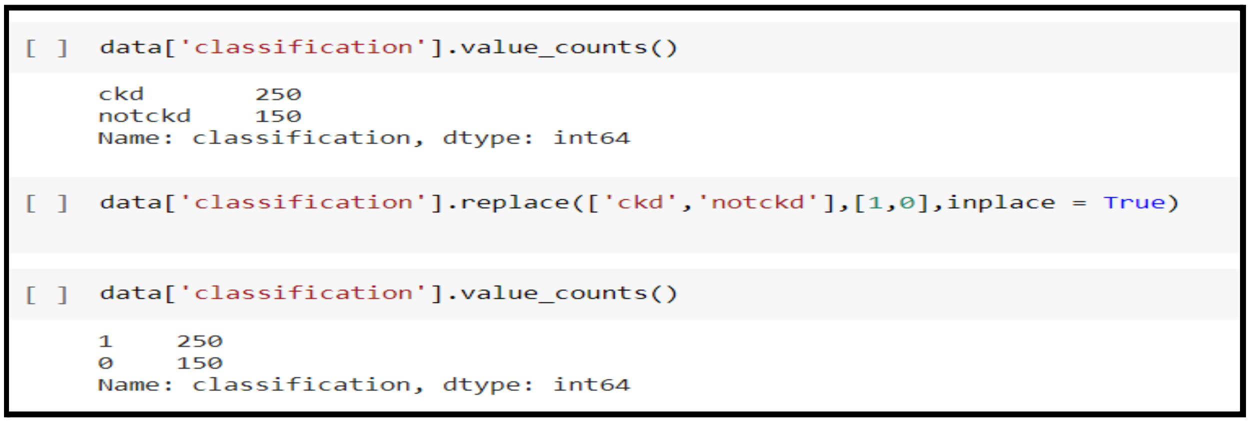


```
[153] mean = data["age"].mean()
      data["age"].fillna(mean, inplace=True)
```

➔ **Calculating the mode**: This method is useful for features having categorical data like hypertension, pus cell, red blood cells, etc. Here, you can calculate the mode of a particular feature or column or row that contains a missing value and replace the result with the missing value.

➢ **Encoding Categorical Data**

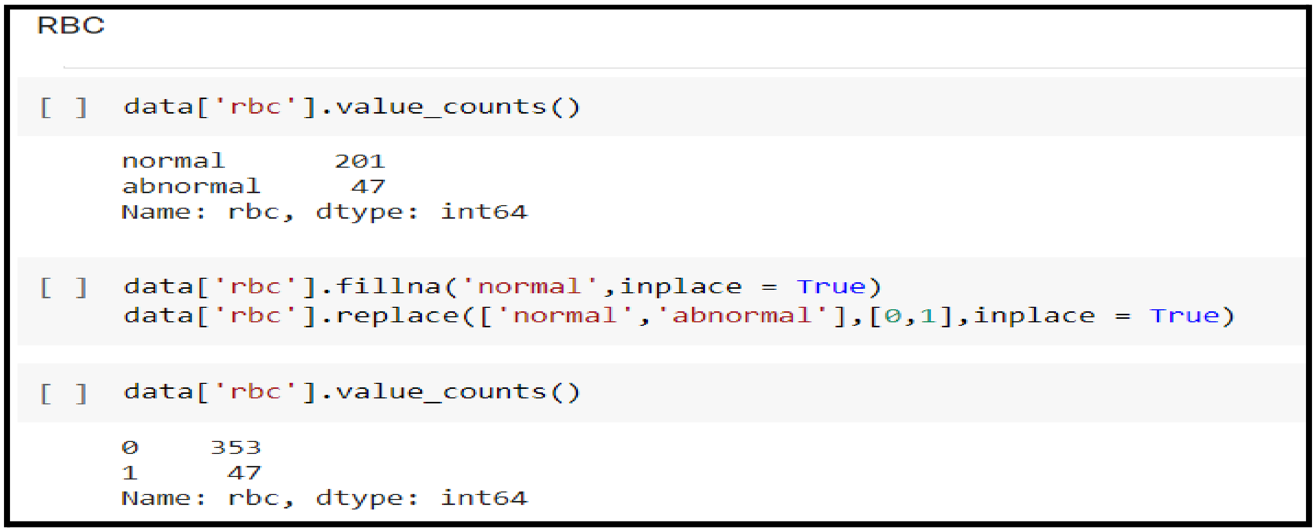
The performance of a machine learning model not only depends on the model and the hyperparameters but also on how we process and feed different types of variables to the model. Since most machine learning models only accept numeric variables, preprocessing the categorical variables becomes a necessary step. We need to convert these categorical variables to numbers such that the model can understand and extract valuable information.



```
[ ]  data['classification'].value_counts()

     ckd        250
     notckd     150
     Name: classification, dtype: int64

[ ]  data['classification'].replace(['ckd','notckd'],[1,0],inplace = True)

[ ]  data['classification'].value_counts()

     1    250
     0    150
     Name: classification, dtype: int64
```

In the above image replace([‘ckd’, ‘notckd’],[1,0]) replaces the values in the column as 1 for CKD and 0 for notckd.

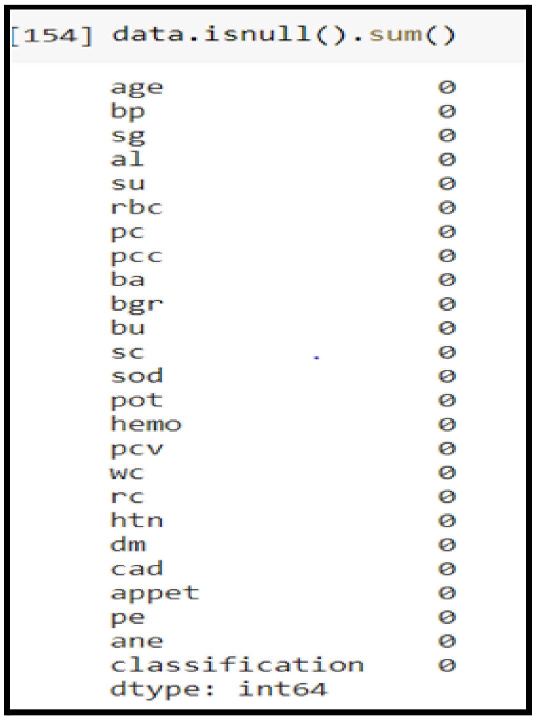
In the below image, replace([‘normal’ ‘abnormal’],[0,1]) replaces the values in the column as 1 for abnormal and 0 for normal.



RBC

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

     normal       201
     abnormal      47
     Name: rbc, dtype: int64

[ ]  data['rbc'].fillna('normal',inplace = True)
     data['rbc'].replace(['normal','abnormal'],[0,1],inplace = True)

[ ]  data['rbc'].value_counts()

     0     353
     1      47
     Name: rbc, dtype: int64
```

After performing the above steps, the dataset does not contain any missing values.



```
[154] data.isnull().sum()

      age               0
      bp                0
      sg                0
      al                0
      su                0
      rbc               0
      pc                0
      pcc               0
      ba                0
      bgr               0
      bu                0
      sc                0
      sod               0
      pot               0
      hemo              0
      pcv               0
      wc                0
      rc                0
      htn               0
      dm                0
      cad               0
      appet             0
      pe                0
      ane               0
      classification    0
      dtype: int64
```

5.2.3. Data Augmentation

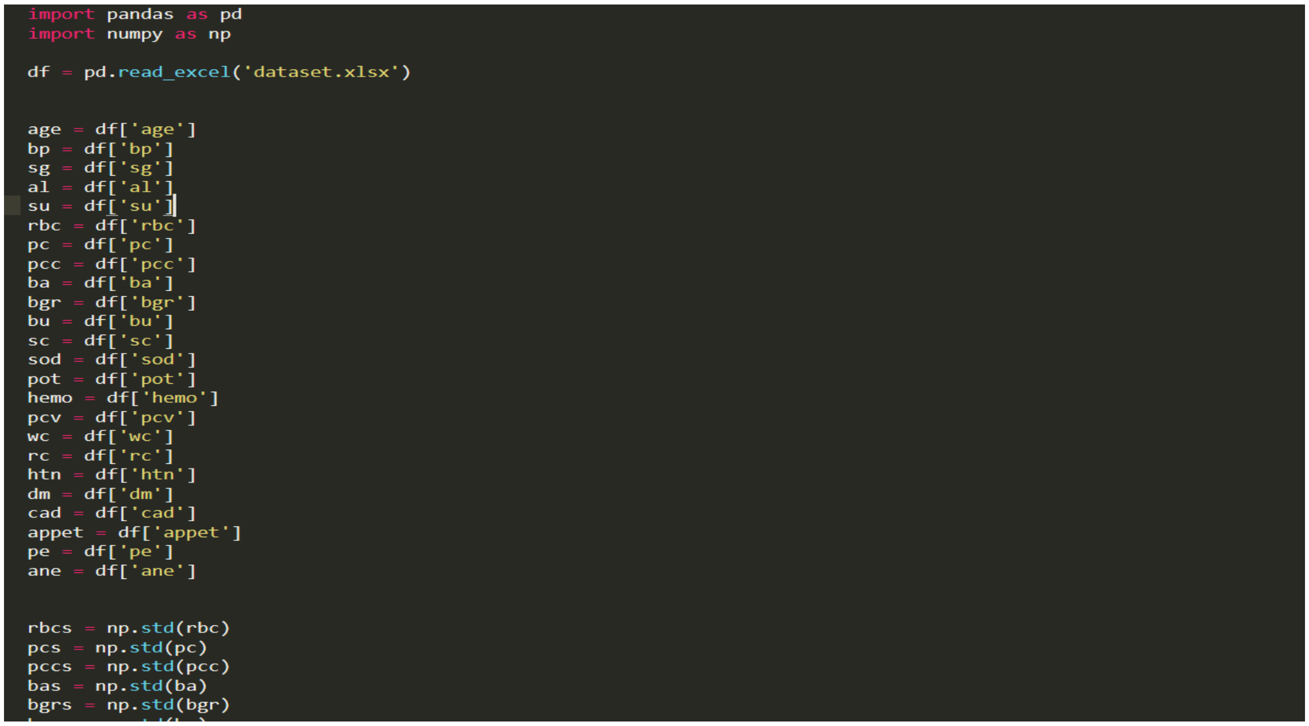
❖ Data Augmentation

Data augmentation in data analysis is a technique used to increase the amount of data by adding slightly modified copies of already existing data or newly created synthetic data from existing data. It acts as a regularizer and helps reduce overfitting when training a machine learning model. It is closely related to oversampling in data analysis.

Steps Involved:

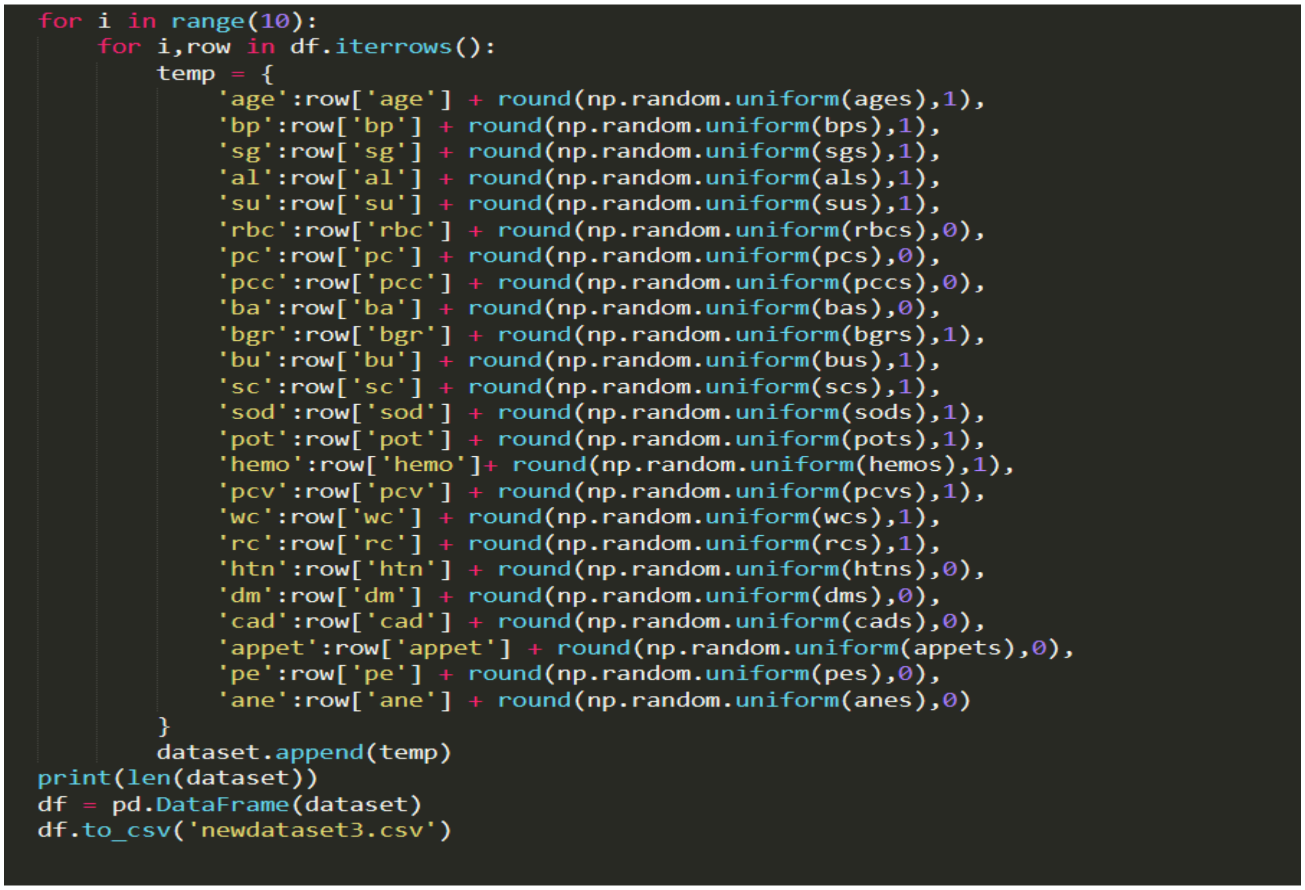
- ➔ First, we read the dataset in the form of exce.
- ➔ We will separate the columns, in their respective variables.
- ➔ Later we will find the Standard Deviation for every column and store it into another variable.
- ➔ We consider an empty list, and add the existing rows into the list, by considering a dictionary in which column name is key and data is considered as value.
- ➔ Later, we will add the new rows with random values in the range of standard deviation of the columns using the random() function in NumPy.
- ➔ Finally, we will convert the list into a data frame and it is converted into a csv file.

Code:



```
import pandas as pd
import numpy as np

df = pd.read_excel('dataset.xlsx')


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


rbcs = np.std(rbc)
pcs = np.std(pc)
pccs = np.std(pcc)
bas = np.std(ba)
bgrs = np.std(bgr)
```

Code 1 (data augmentation)



```
for i in range(10):
    for i,row in df.iterrows():
        temp = {
            'age':row['age'] + round(np.random.uniform(ages),1),
            'bp':row['bp'] + round(np.random.uniform(bps),1),
            'sg':row['sg'] + round(np.random.uniform(sgs),1),
            'al':row['al'] + round(np.random.uniform(als),1),
            'su':row['su'] + round(np.random.uniform(sus),1),
            'rbc':row['rbc'] + round(np.random.uniform(rbcs),0),
            'pc':row['pc'] + round(np.random.uniform(pcs),0),
            'pcc':row['pcc'] + round(np.random.uniform(pccs),0),
            'ba':row['ba'] + round(np.random.uniform(bas),0),
            'bgr':row['bgr'] + round(np.random.uniform(bgrs),1),
            'bu':row['bu'] + round(np.random.uniform(bus),1),
            'sc':row['sc'] + round(np.random.uniform(scs),1),
            'sod':row['sod'] + round(np.random.uniform(sods),1),
            'pot':row['pot'] + round(np.random.uniform(pots),1),
            'hemo':row['hemo']+ round(np.random.uniform(hemos),1),
            'pcv':row['pcv'] + round(np.random.uniform(pcvs),1),
            'wc':row['wc'] + round(np.random.uniform(wcs),1),
            'rc':row['rc'] + round(np.random.uniform(rcs),1),
            'htn':row['htn'] + round(np.random.uniform(htns),0),
            'dm':row['dm'] + round(np.random.uniform(dms),0),
            'cad':row['cad'] + round(np.random.uniform(cads),0),
            'appet':row['appet'] + round(np.random.uniform(appets),0),
            'pe':row['pe'] + round(np.random.uniform(pes),0),
            'ane':row['ane'] + round(np.random.uniform(anes),0)
        }
        dataset.append(temp)
print(len(dataset))
df = pd.DataFrame(dataset)
df.to_csv('newdataset3.csv')
```

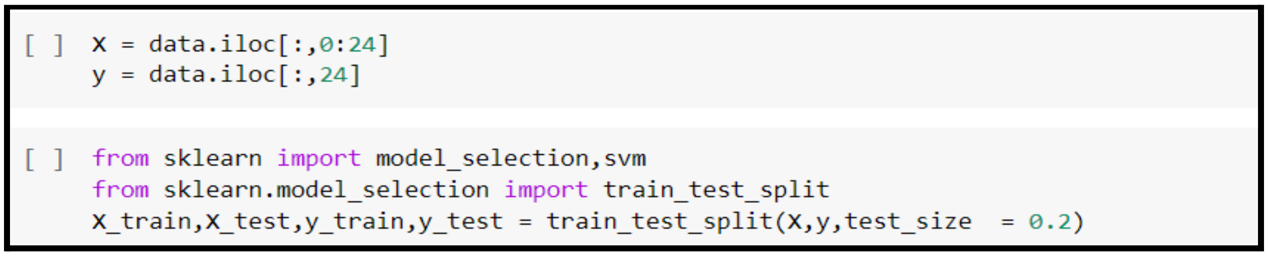
Code -2 (data augmentation)

❖ Dividing the dataset into training data and testing data:

Splitting the Dataset into the Training set and Test set. We will train our machine learning models on our training set. Our machine learning models will try to understand any correlations in our training set and then we will test the models on our test set to examine how accurately they will predict. A general rule of the thumb is to assign 70% of the dataset to the training set and therefore the remaining 30% to the test set.

Training set — a subset to train a model.

Test set — a subset to test the trained model.


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

[ ]  from sklearn import model_selection,svm
     from sklearn.model_selection import train_test_split
     X_train,X_test,y_train,y_test = train_test_split(X,y,test_size  = 0.2)
```

➔ We use sklearn.model_selection to import train_test_split. X_train, y_train contains fields other than classification, whereas x_test, y_test contains the classification field.

5.2.4. Feature Selection

❖ Feature Selection:

To perform any machine learning task from high dimensional data, feature selection becomes very important. Since some features may be irrelevant or less significant to the dependent variable so their unnecessary inclusion in the model leads to

➔ An increase in the complexity of a model makes it harder to interpret.
➔ Increase in time complexity for a model to get trained.
➔ This results in a model with inaccurate or less reliable predictions.

Hence, it gives an indispensable need to perform feature selection. Feature selection is very crucial and a must component in machine learning workflows especially while dealing with high-dimensional datasets.

Feature Selection Techniques: Our project includes

- Correlation Coefficient
- Wrapper Method - Forward Feature Selection

- **Correlation Coefficient:**

Correlation is a measure of the linear relationship of 2 or more variables. Through correlation, we can predict one variable from the other. The logic behind using correlation for feature selection is that the good variables are highly correlated with the target. Furthermore, variables should be correlated with the target but should be uncorrelated among themselves.

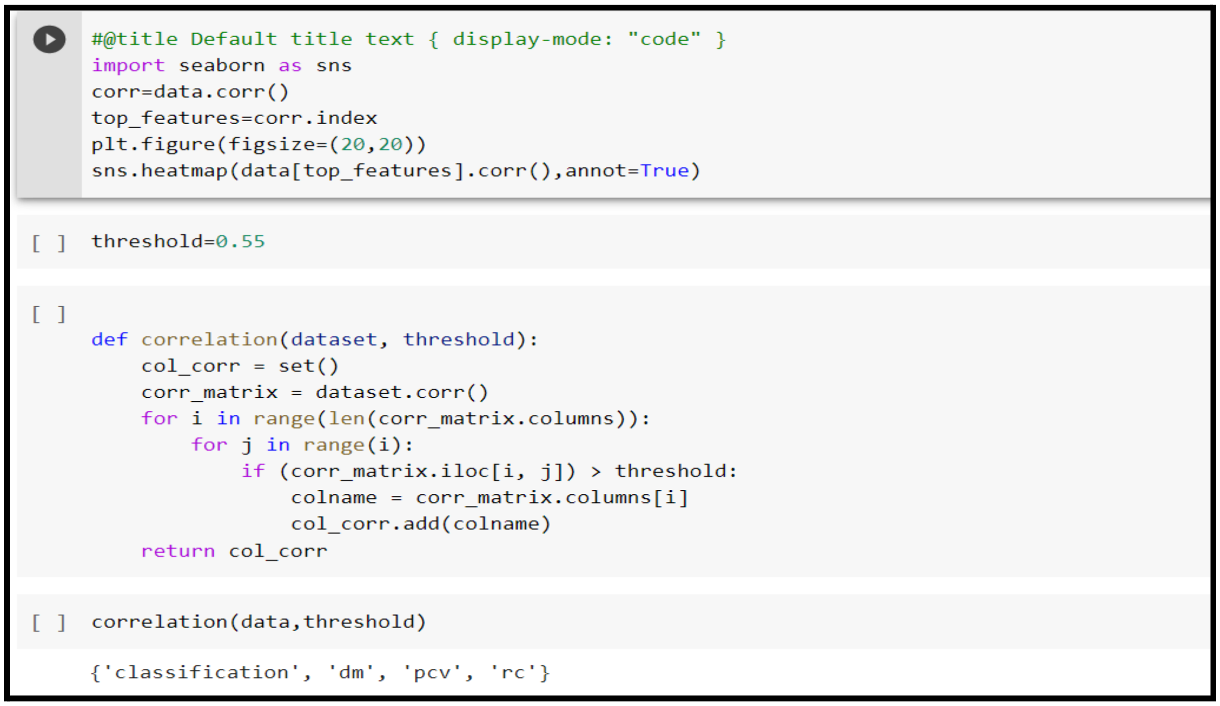
If two variables are correlated, we can predict one from the other. Therefore, if two features are correlated, the model only really needs one of them, as the second one does not add additional information.

Heatmap is defined as a graphical representation of data using colors to visualize the value of the matrix. In this, to represent more common values or higher activities brighter colors basically, reddish colors are used, and to represent less common or activity values, darker colors are preferred. Heatmap is also defined by the name of the shading matrix. Heatmaps in Seaborn can be plotted by using the seaborn.heatmap() function. We can use the below

$$r = \frac{\sum(X-\overline{X})(Y-\overline{Y})}{\sqrt{\sum(X-\overline{X})^2}\sqrt{(Y-\overline{Y})^2}}$$

Where, $\overline{X}$ = mean of X variable
$\overline{Y}$ = mean of Y variable

Formula for calculating Correlation Coefficient



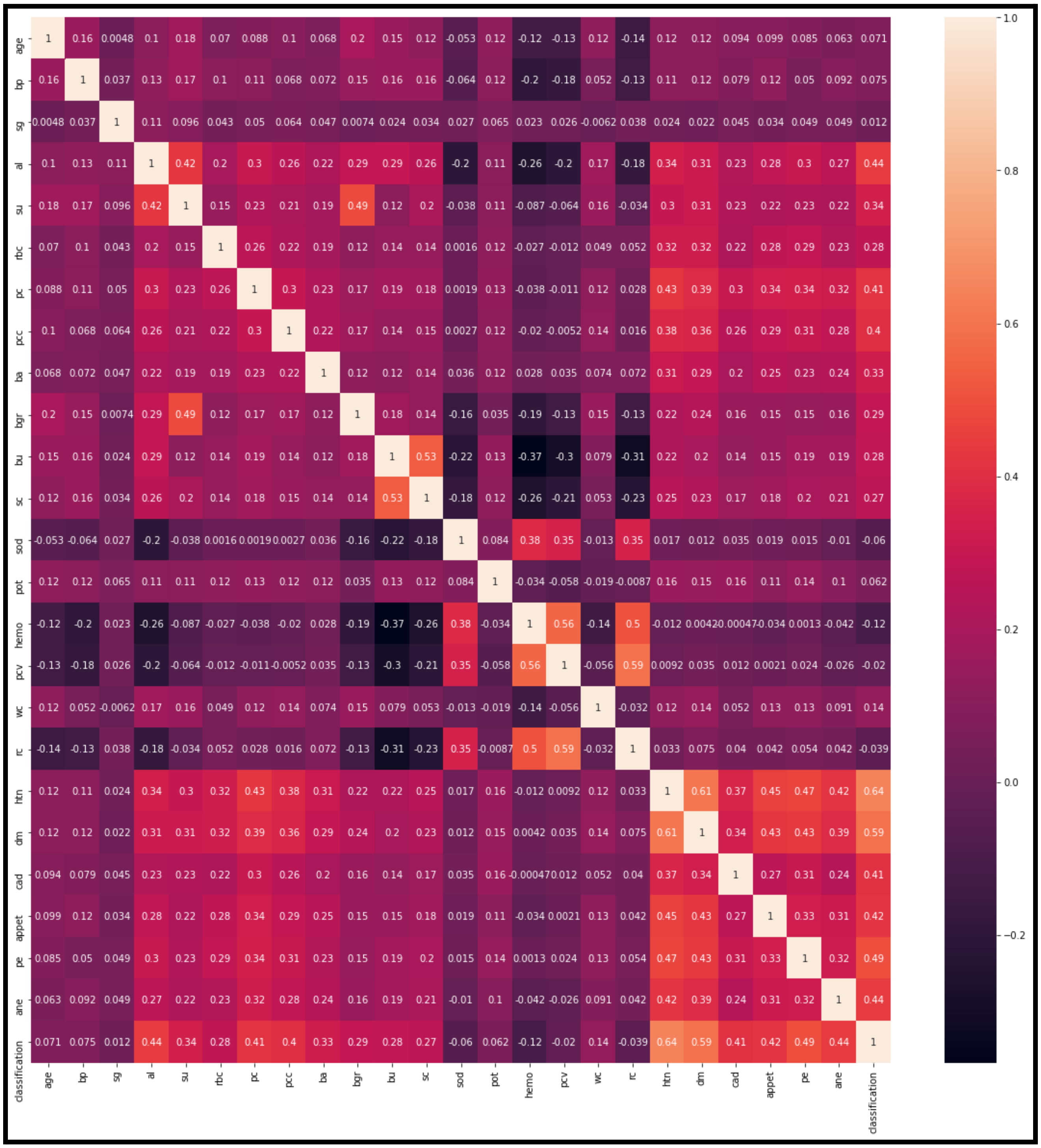
```
#@title Default title text { display-mode: "code" }
import seaborn as sns
corr=data.corr()
top_features=corr.index
plt.figure(figsize=(20,20))
sns.heatmap(data[top_features].corr(),annot=True)
```

```
[ ] threshold=0.55
```

```
[ ]
    def correlation(dataset, threshold):
        col_corr = set()
        corr_matrix = dataset.corr()
        for i in range(len(corr_matrix.columns)):
            for j in range(i):
                if (corr_matrix.iloc[i, j]) > threshold:
                    colname = corr_matrix.columns[i]
                    col_corr.add(colname)
        return col_corr
```

```
[ ] correlation(data,threshold)

    {'classification', 'dm', 'pcv', 'rc'}
```

Code of Correlation Heat Map

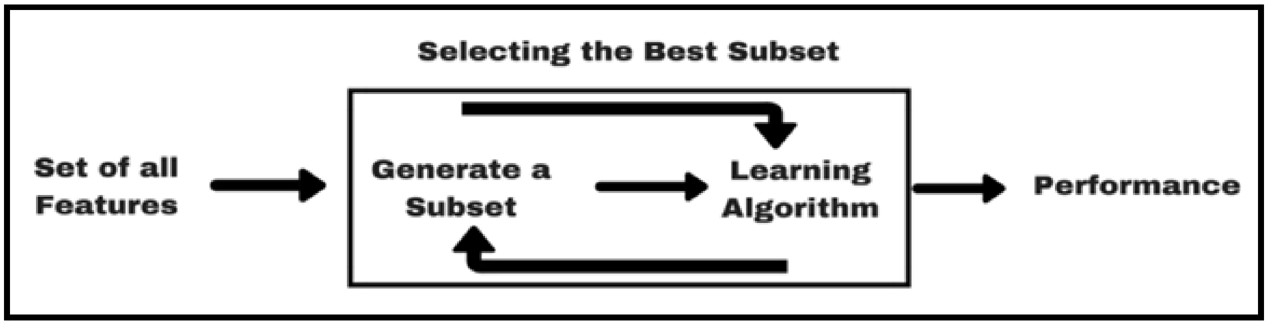


	age	bp	sg	al	su	rbc	pc	pcc	ba	bgr	bu	sc	sod	pot	hemo	pcv	wc	rc	htn	dm	cad	appet	pe	ane	classification
age	1	0.16	0.0048	0.1	0.18	0.07	0.088	0.1	0.068	0.2	0.15	0.12	-0.053	0.12	-0.12	-0.13	0.12	-0.14	0.12	0.12	0.094	0.099	0.085	0.063	0.071
bp	0.16	1	0.037	0.13	0.17	0.1	0.11	0.068	0.072	0.15	0.16	0.16	-0.064	0.12	-0.2	-0.18	0.052	-0.13	0.11	0.12	0.079	0.12	0.05	0.092	0.075
sg	0.0048	0.037	1	0.11	0.096	0.043	0.05	0.064	0.047	0.0074	0.024	0.034	0.027	0.065	0.023	0.026	-0.0062	0.038	0.024	0.022	0.045	0.034	0.049	0.049	0.012
al	0.1	0.13	0.11	1	0.42	0.2	0.3	0.26	0.22	0.29	0.29	0.26	-0.2	0.11	-0.26	-0.2	0.17	-0.18	0.34	0.31	0.23	0.28	0.3	0.27	0.44
su	0.18	0.17	0.096	0.42	1	0.15	0.23	0.21	0.19	0.49	0.12	0.2	-0.038	0.11	-0.087	-0.064	0.16	-0.034	0.3	0.31	0.23	0.22	0.23	0.22	0.34
rbc	0.07	0.1	0.043	0.2	0.15	1	0.26	0.22	0.19	0.12	0.14	0.14	0.0016	0.12	-0.027	-0.012	0.049	0.052	0.32	0.32	0.22	0.28	0.29	0.23	0.28
pc	0.088	0.11	0.05	0.3	0.23	0.26	1	0.3	0.23	0.17	0.19	0.18	0.0019	0.13	-0.038	-0.011	0.12	0.028	0.43	0.39	0.3	0.34	0.34	0.32	0.41
pcc	0.1	0.068	0.064	0.26	0.21	0.22	0.3	1	0.22	0.17	0.14	0.15	0.0027	0.12	-0.02	-0.0052	0.14	0.016	0.38	0.36	0.26	0.29	0.31	0.28	0.4
ba	0.068	0.072	0.047	0.22	0.19	0.19	0.23	0.22	1	0.12	0.12	0.14	0.036	0.12	0.028	0.035	0.074	0.072	0.31	0.29	0.2	0.25	0.23	0.24	0.33
bgr	0.2	0.15	0.0074	0.29	0.49	0.12	0.17	0.17	0.12	1	0.18	0.14	-0.16	0.035	-0.19	-0.13	0.15	-0.13	0.22	0.24	0.16	0.15	0.15	0.16	0.29
bu	0.15	0.16	0.024	0.29	0.12	0.14	0.19	0.14	0.12	0.18	1	0.53	-0.22	0.13	-0.37	-0.3	0.079	-0.31	0.22	0.2	0.14	0.15	0.19	0.19	0.28
sc	0.12	0.16	0.034	0.26	0.2	0.14	0.18	0.15	0.14	0.14	0.53	1	-0.18	0.12	-0.26	-0.21	0.053	-0.23	0.25	0.23	0.17	0.18	0.2	0.21	0.27
sod	-0.053	-0.064	0.027	-0.2	-0.038	0.0016	0.0019	0.0027	0.036	-0.16	-0.22	-0.18	1	0.084	0.38	0.35	-0.013	0.35	0.017	0.012	0.035	0.019	0.015	-0.01	-0.06
pot	0.12	0.12	0.065	0.11	0.11	0.12	0.13	0.12	0.12	0.035	0.13	0.12	0.084	1	-0.034	-0.058	-0.019	-0.0087	0.16	0.15	0.16	0.11	0.14	0.1	0.062
hemo	-0.12	-0.2	0.023	-0.26	-0.087	-0.027	-0.038	-0.02	0.028	-0.19	-0.37	-0.26	0.38	-0.034	1	0.56	-0.14	0.5	-0.012	0.0042	-0.00047	-0.034	0.0013	-0.042	-0.12
pcv	-0.13	-0.18	0.026	-0.2	-0.064	-0.012	-0.011	-0.0052	0.035	-0.13	-0.3	-0.21	0.35	-0.058	0.56	1	-0.056	0.59	0.0092	0.035	0.012	0.0021	0.024	-0.026	-0.02
wc	0.12	0.052	-0.0062	0.17	0.16	0.049	0.12	0.14	0.074	0.15	0.079	0.053	-0.013	-0.019	-0.14	-0.056	1	-0.032	0.12	0.14	0.052	0.13	0.13	0.091	0.14
rc	-0.14	-0.13	0.038	-0.18	-0.034	0.052	0.028	0.016	0.072	-0.13	-0.31	-0.23	0.35	-0.0087	0.5	0.59	-0.032	1	0.033	0.075	0.04	0.042	0.054	0.042	-0.039
htn	0.12	0.11	0.024	0.34	0.3	0.32	0.43	0.38	0.31	0.22	0.22	0.25	0.017	0.16	-0.012	0.0092	0.12	0.033	1	0.61	0.37	0.45	0.47	0.42	0.64
dm	0.12	0.12	0.022	0.31	0.31	0.32	0.39	0.36	0.29	0.24	0.2	0.23	0.012	0.15	0.0042	0.035	0.14	0.075	0.61	1	0.34	0.43	0.43	0.39	0.59
cad	0.094	0.079	0.045	0.23	0.23	0.22	0.3	0.26	0.2	0.16	0.14	0.17	0.035	0.16	-0.00047	0.012	0.052	0.04	0.37	0.34	1	0.27	0.31	0.24	0.41
appet	0.099	0.12	0.034	0.28	0.22	0.28	0.34	0.29	0.25	0.15	0.15	0.18	0.019	0.11	-0.034	0.0021	0.13	0.042	0.45	0.43	0.27	1	0.33	0.31	0.42
pe	0.085	0.05	0.049	0.3	0.23	0.29	0.34	0.31	0.23	0.15	0.19	0.2	0.015	0.14	0.0013	0.024	0.13	0.054	0.47	0.43	0.31	0.33	1	0.32	0.49
ane	0.063	0.092	0.049	0.27	0.22	0.23	0.32	0.28	0.24	0.16	0.19	0.21	-0.01	0.1	-0.042	-0.026	0.091	0.042	0.42	0.39	0.24	0.31	0.32	1	0.44
classification	0.071	0.075	0.012	0.44	0.34	0.28	0.41	0.4	0.33	0.29	0.28	0.27	-0.06	0.062	-0.12	-0.02	0.14	-0.039	0.64	0.59	0.41	0.42	0.49	0.44	1

Correlation Heat Map

- **Wrapper Methods:**

Wrappers require some method to search the space of all possible subsets of features, assessing their quality by learning and evaluating a classifier with that feature subset. The feature selection process is based on a specific machine learning algorithm that we are trying to fit on a given dataset. It follows a greedy search approach by evaluating all the possible combinations of features against the evaluation criterion. The wrapper methods usually result in better predictive accuracy than filter methods.



- **Forward Feature Selection:**

This is an iterative method wherein we start with the best performing variable against the target. Next, we select another variable that gives the best performance in combination with the first selected variable. This process continues until the preset criterion is achieved.

Parameters Involved:

➔ **K_features , default=None**

The number of features to select. If None, half of the features are selected. If an integer, the parameter is the absolute number of features to select.

➔ **direction{‘forward’, ‘backward’}, default=’forward’**

Whether to perform forward selection or backward selection.

- ➔ **scoring str,neg_mean_squared_error,accuracy, list/tuple, default=None**

Single str to evaluate the predictions on the test set.

- ➔ **Cv int, cross-validation generator or an iterable, default=None**

Determines the cross-validation splitting strategy. Possible inputs for cv are:

None, to use the default 5-fold cross-validation,

integer, to specify the number of folds in a KFold

CV splitter,

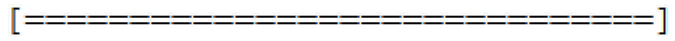
An iterable yielding (train, test) splits as arrays of indices.

- ➔ **n_jobs*int, default*=*None***

The number of jobs to run in parallel. When evaluating a new feature to add or remove, the cross-validation procedure is parallel over the folds.-1 means using all processors.

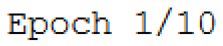
- ➔ **Verbose:**

By setting verbose 0, 1, or 2 you just say how you want to 'see' the training progress for each epoch.
verbose=0 will show you nothing (silent)
verbose=1 will show you an animated progress bar like this:



```
[==============================]
```

verbose=2 will just mention the number of epoch like this:

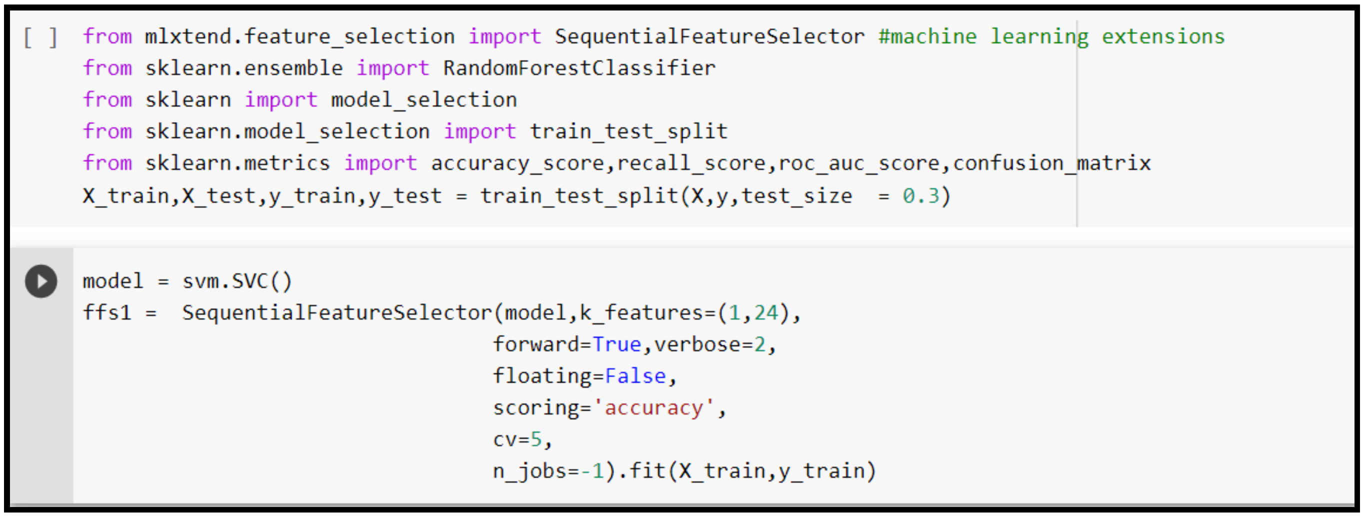


```
Epoch 1/10
```

Implementation of Forward Feature Selection:

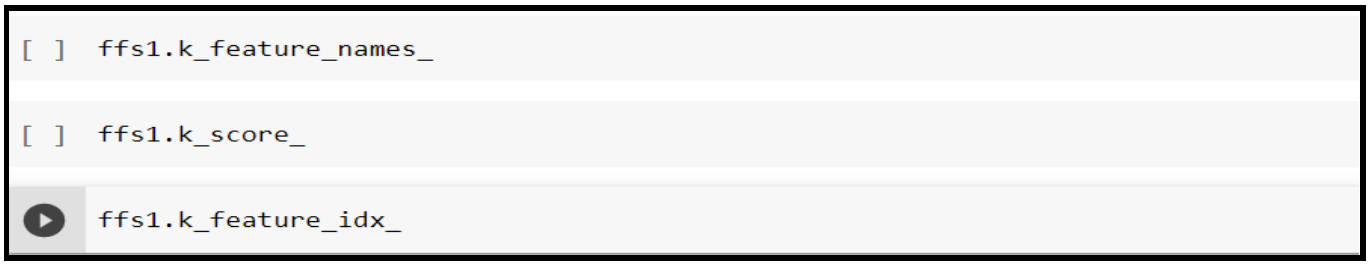
Steps Involved:

- ➔ We will import the Sequential Feature Selector from mlextend.feature_selection.
- ➔ Because our SVM model accuracy is 84, we perform feature selection based on SVM.
- ➔ We first generate the SVM model using SVM().
- ➔ Pass this model as a parameter to the SquentialFeatureSelector, along with the other parameters like k_features, forward, scoring, verbose, cv, n_jobs.
- ➔ Use fit() method to fit X_train, y_train.
- ➔ In return, we will get a SquentialFeatureSelector object ffs1.
- ➔ **ffs1.k_feature_names_** gives the feature names as output.
- ➔ **Ffs1.k_score** gives the accuracy by considering the only required fields.
- ➔ **Ffs1.k_feature_idx_** gives the indices of the selected features.



```
[ ] from mlxtend.feature_selection import SequentialFeatureSelector #machine learning extensions
    from sklearn.ensemble import RandomForestClassifier
    from sklearn import model_selection
    from sklearn.model_selection import train_test_split
    from sklearn.metrics import accuracy_score,recall_score,roc_auc_score,confusion_matrix
    X_train,X_test,y_train,y_test = train_test_split(X,y,test_size  = 0.3)
```

```
model = svm.SVC()
ffs1 =  SequentialFeatureSelector(model,k_features=(1,24),
                                  forward=True,verbose=2,
                                  floating=False,
                                  scoring='accuracy',
                                  cv=5,
                                  n_jobs=-1).fit(X_train,y_train)
```



```
[ ]  ffs1.k_feature_names_

[ ]  ffs1.k_score_

     ffs1.k_feature_idx_
```

5.2.5. Algorithms and Calculating Accuracies

❖ **Algorithms with Implementation:**

◆ **Naive Bayes:**

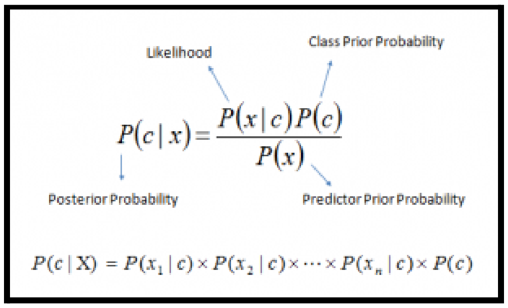
It is a classification technique based on the Bayes Theorem with an assumption of independence among predictors. In simple terms, a Naive Bayes classifier assumes that the presence of a particular feature in a class is unrelated to the presence of any other feature.

The Naive Bayes algorithm is comprised of two words Naive and Bayes, Which can be described as:

➔ **Naive**: It is called Naive because it assumes that the occurrence of a certain feature is independent of the occurrence of other features. Such as if the fruit is identified based on color, shape, and taste, then red, spherical, and sweet fruit is recognized as an apple. Hence each feature individually contributes to identifying that it is an apple without depending on each other.

➔ **Bayes**: It is called Bayes because it depends on the principle of the Bayes Theorem.

Bayes theorem provides a way of calculating posterior probability P(c|x) from P(c), P(x), and P(x|c). Look at the equation below:


Likelihood
Class Prior Probability
$P(c|x)=\frac{P(x|c)P(c)}{P(x)}$
Posterior Probability
Predictor Prior Probability
$P(c|X) = P(x_1|c)\times P(x_2|c)\times\cdots\times P(x_n|c)\times P(c)$

calculating posterior probability

Above,

- ➔ P(c|x) is the posterior probability of class (c, target) given predictor (x, attributes).
- ➔ P(c) is the prior probability of class.
- ➔ P(x|c) is the likelihood which is the probability of the predictor given class.
- ➔ P(x) is the prior probability of the predictor.

First, the total number of training records is calculated from the counts stored in the summary statistics. This is used in the calculation of the probability of a given class or P(class) as the ratio of rows with a given class of all rows in the training data.

Next, probabilities are calculated for each input value in the row using the Gaussian probability density function and the statistics for that column and of that class. Probabilities are multiplied together as they accumulate. This process is repeated for each class in the dataset.

For our dataset where we have 24 input variables, the calculation of the probability that a row belongs to the first-class 0 (non-ckd) can be calculated as:

P(class=0|X1,X2,...X24) = P(X1|class=0) * P(X2|class=0) …P(X24|class=0)* P(class=0)

Similarly, for class 1 (ckd):

P(class=1|X1,X2,...X24) = P(X1|class=1) * P(X2|class=1) …P(X24|class=1)* P(class=1)

Probability can be calculated using Gaussian Probability Distribution Function (or Gaussian PDF) as:

f(x) = (1 / sqrt(2 * PI) * sigma) * exp(-((x-mean)^2 / (2 * sigma^2)))

Or

$$\boxed{p(x = v \mid C_k) = \frac{1}{\sqrt{2\pi\sigma_k^2}}\, e^{-\frac{(v-\mu_k)^2}{2\sigma_k^2}}}$$

Gaussian Probability Distribution Function

Where sigma is the standard deviation for x, mean is the mean for x and PI is the value of pi.

◆ **Support Vector Machine:**

Support Vector Machine (SVM) is a supervised machine learning algorithm that can be used for both classification or regression challenges. It is mostly used for Classification. The goal of the SVM algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in the future. This best decision boundary is called Hyperplane.

SVM chooses the extreme points/vectors that help in creating the hyperplane. The extreme cases are called support vectors, and hence the algorithm is termed a Support Vector Machine. Consider the below diagram in which two different categories are classified using a decision boundary.

Hyperplane:

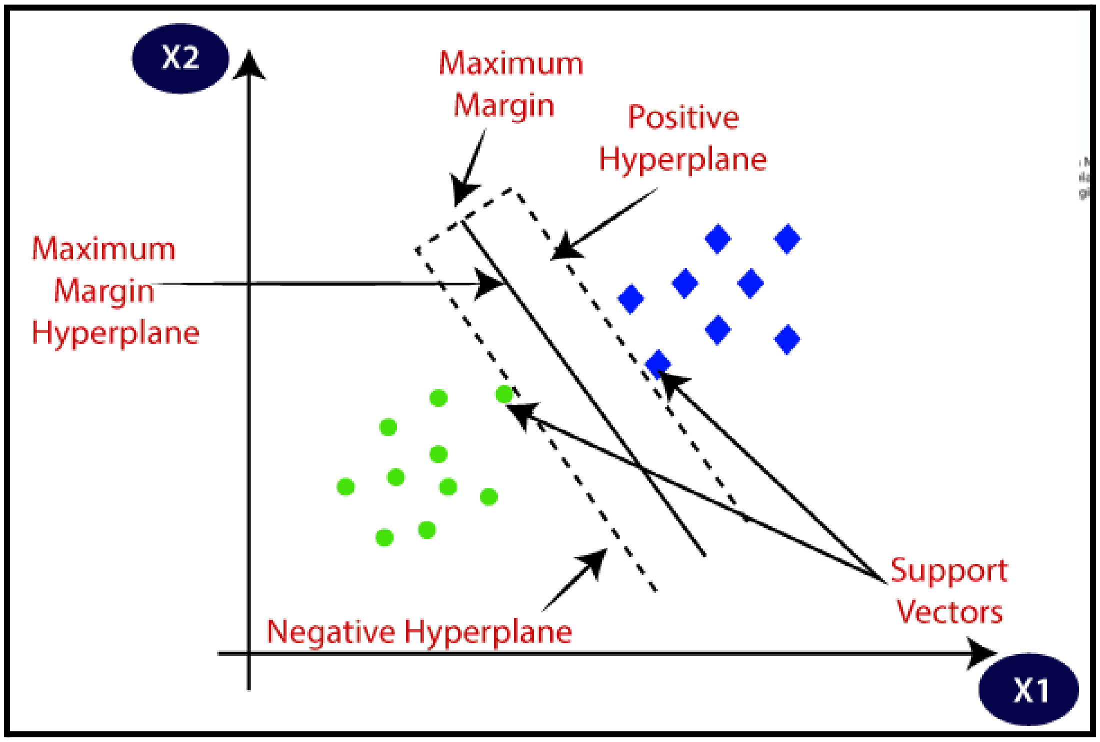
There can be multiple lines/decision boundaries to segregate the classes in n-dimensional space, but we need to find out the best decision boundary that helps to classify the data points. This best boundary is known as the hyperplane of SVM.

The dimensions of the hyperplane depend on the features present in the dataset, which means if there are 2 features (as shown in the image), then the hyperplane will be a straight line. And if there are 3 features, then the hyperplane will be a 2-dimension plane.

We always create a hyperplane that has a maximum margin, which means the maximum distance between the data points.

Support Vectors:

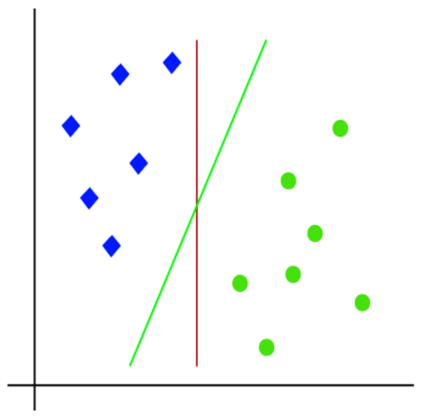
The data points or vectors that are the closest to the hyperplane and which affect the position of the hyperplane are termed Support Vectors. These vectors support the hyperplane, hence called a Support vector.



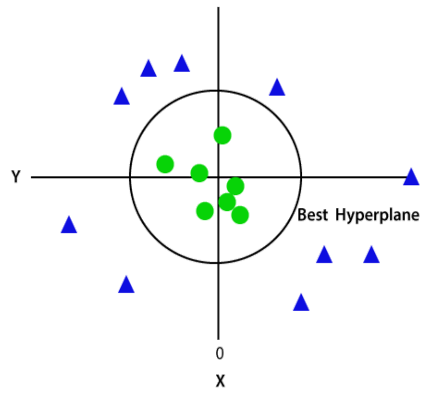
Types of SVM:

Linear SVM: Linear SVM is used for linearly separable data, which means if a dataset can be classified into two classes by using a single straight line, then such data is termed as linearly separable data, and classifier is used called as Linear SVM classifier.

Non-Linear SVM: Non-Linear SVM is used for non-linearly separated data, which means if a dataset cannot be classified by using a straight line, then such data is termed as non-linear data, and classifier used is called as Non-linear SVM classifier.



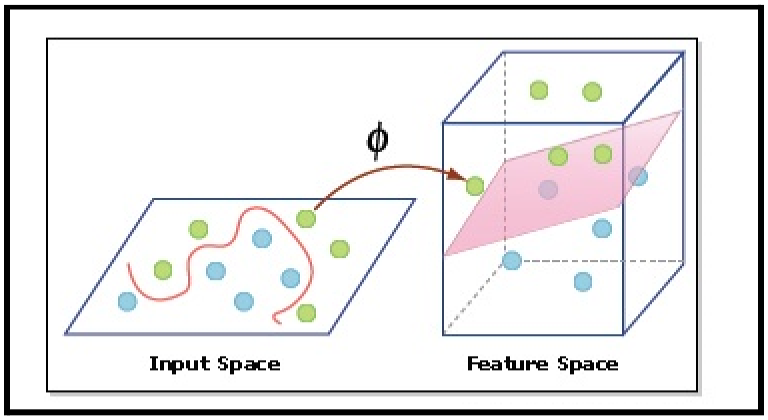
Linear SVM



Non-Linear SVM

Kernels in SVM:

A kernel helps us find a hyperplane in the higher dimensional space without increasing the computational cost. Usually, the computational cost will increase if the dimension of the data increases. This increase in dimension is required when we are unable to find a separating hyperplane in a given dimension and are required to move in a higher dimension:



Kernels in SVM

Types of Kernels used in the project:

★ Gaussian Radial Basis Function (RBF) Kernel
★ Polynomial kernel
★ Sigmoid Kernel

★ **Gaussian Radial Basis Function:**

RBF kernel is a function whose value depends on the distance from the origin or from some point. Using the distance in the original space we calculate the dot product (similarity) of X1 & X2.

Parameters:

C: Inverse to the strength of regularization. Behavior: As the value of 'c' increases the model gets overfits. As the value of 'c' decreases the model underfits.

γ: Gamma (used only for RBF kernel) Behavior: As the value of ' γ' increases the model gets overfit. As the value of ' γ' decreases the model underfits.

★ **Polynomial Kernel:**

In machine learning, the polynomial kernel is a kernel function commonly used with support vector machines (SVMs) and other kernelized models, that represent the similarity of vectors (training samples) in a feature space over polynomials of the original variables, allowing learning of non-linear models.

Parameters:

- ➔ C: Inverse to the strength of regularization. Behavior: As the value of ‘c’ increases the model gets overfits. As the value of ‘c’ decreases the model underfits.
- ➔ γ: Gamma (used only for RBF kernel) Behavior: As the value of ‘ γ’ increases the model gets overfit. As the value of ‘ γ’ decreases the model underfits.
- ➔ Degree: Degree of the polynomial kernel function (‘poly’). Ignored by all other kernels.

★ **Sigmoid Kernel:**

The Sigmoid Kernel comes from the Neural Networks field, where the bipolar sigmoid function is often used as an activation function for artificial neurons. It is interesting to note that an SVM model using a sigmoid kernel function is equivalent to a two-layer, perceptron neural network.

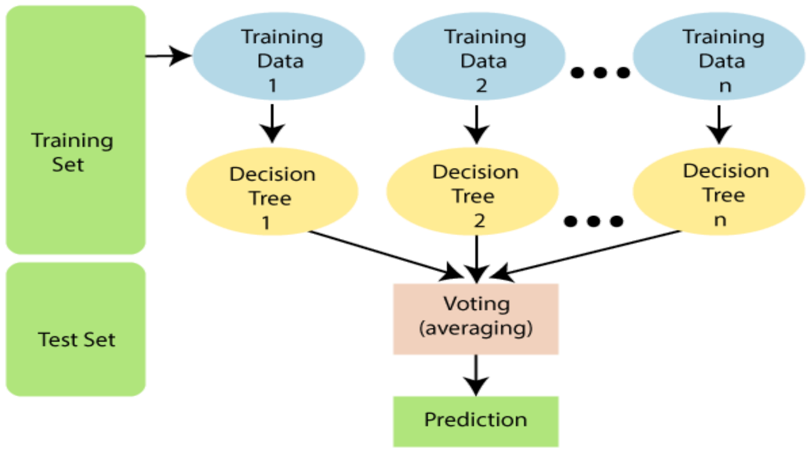
Parameters:

- ➔ C: Inverse to the strength of regularization. Behavior: As the value of ‘c’ increases the model gets overfits. As the value of ‘c’ decreases the model underfits.
- ➔ γ: Gamma (used only for RBF kernel) Behavior: As the value of ‘ γ’ increases the model gets overfit. As the value of ‘ γ’ decreases the model underfits.

◆ Random Forest:

It is a Supervised Learning algorithm. It is based on the concept of ensemble learning, which combines multiple classifiers to solve a complex problem and improve the performance of the model.

As the name suggests, "Random Forest is a classifier that contains several decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset." Instead of relying on one decision tree, the random forest takes the prediction from each tree and is based on the majority votes of predictions, and it predicts the final output. The greater number of trees in the forest leads to higher accuracy and prevents the problem of overfitting.



Random Forest

Parameters:

➔ N_estimators - The required number of trees in the Random Forest. The default value is 10. We can choose any number but need to take care of the overfitting issue.

➔ N_jobs - The number of jobs that run in parallel. -1 means use all the processors that are present.

➔ Max_depth - represents the depth of each tree in the forest. The deeper the tree, the more splits it has and it captures more information about the data.

Implementation:

Naive Bayes:

➔ We will use scikit-learn, and we'll import our Gaussian Naive Bayes from sklearn.naive_bayes.

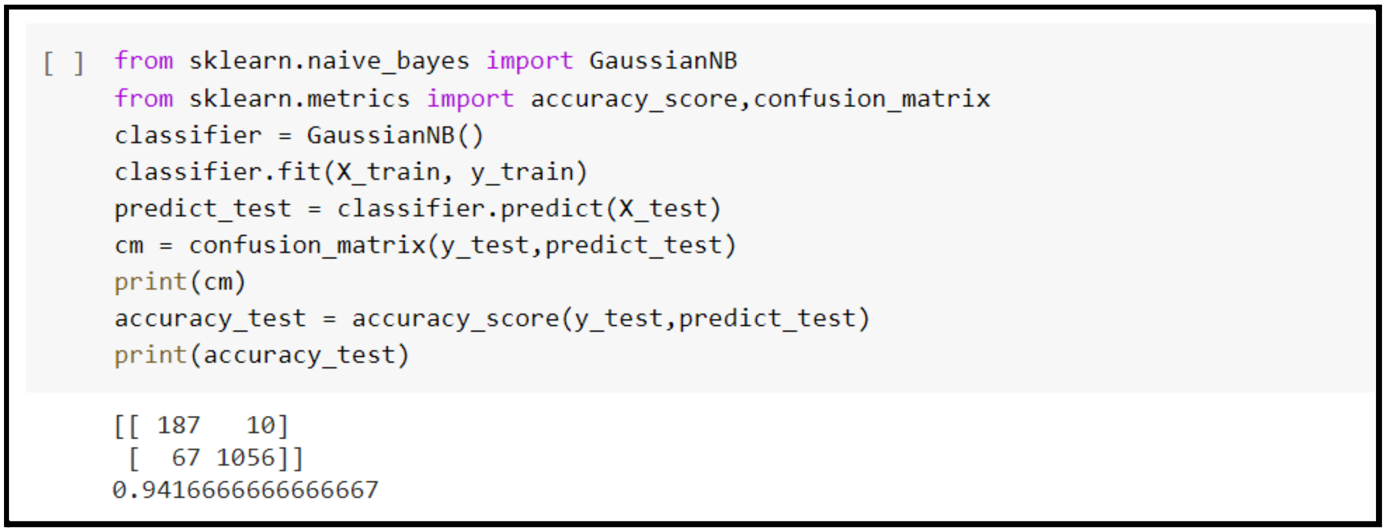
➔ Firstly, we call GuassianNB() function which is a Naive Bayes Classifier function used to build a Naive Bayes model in Machine Learning.

➔ We use X_train and y_train obtained above to train our model and We use the model.fit() method to fit the model.

➔ We test the Model by calling the function classifier.predict() by passing X_test (i.e., The test values of independent variables). We get a predict_test as a result.

➔ After we call the confusion_matrix() method by passing y_test and predict_test.

- ➔ We get the accuracy of the model by calling accuracy_score() by passing y_test and predict_test as parameters.
- ➔ Finally, we get a confusion matrix and accuracy as output.



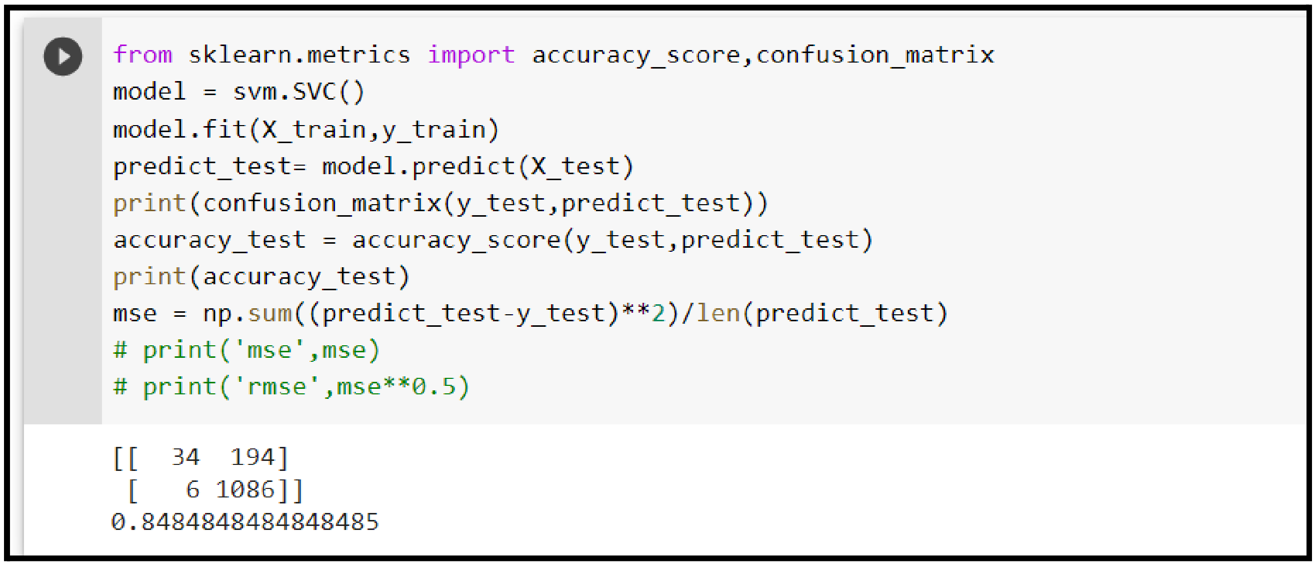
```
[ ]  from sklearn.naive_bayes import GaussianNB
     from sklearn.metrics import accuracy_score,confusion_matrix
     classifier = GaussianNB()
     classifier.fit(X_train, y_train)
     predict_test = classifier.predict(X_test)
     cm = confusion_matrix(y_test,predict_test)
     print(cm)
     accuracy_test = accuracy_score(y_test,predict_test)
     print(accuracy_test)

     [[ 187   10]
      [  67 1056]]
     0.9416666666666667
```

Support Vector Machine:

- ➔ We will use scikit-learn, and we'll import SVM.
- ➔ Firstly, we call the SVM() function which is a Support Vector Machine Classifier function used to build a Support Vector Machine model in Machine Learning.
- ➔ For the usage of the non-linear model, we pass kernel as a parameter with values **sigmoid, RBF.**
- ➔ We use X_train and y_train obtained above to train our model and We use the model.fit() method to fit the model.
- ➔ We test the Model by calling the function classifier.predict() by passing X_test (i.e., The test values of independent variables). We get a predict_test as a result.
- ➔ After we call the confusion_matrix() method by passing y_test and predict_test.

➔ We get the accuracy of the model by calling accuracy_score() by passing y_test and predict_test as parameters.

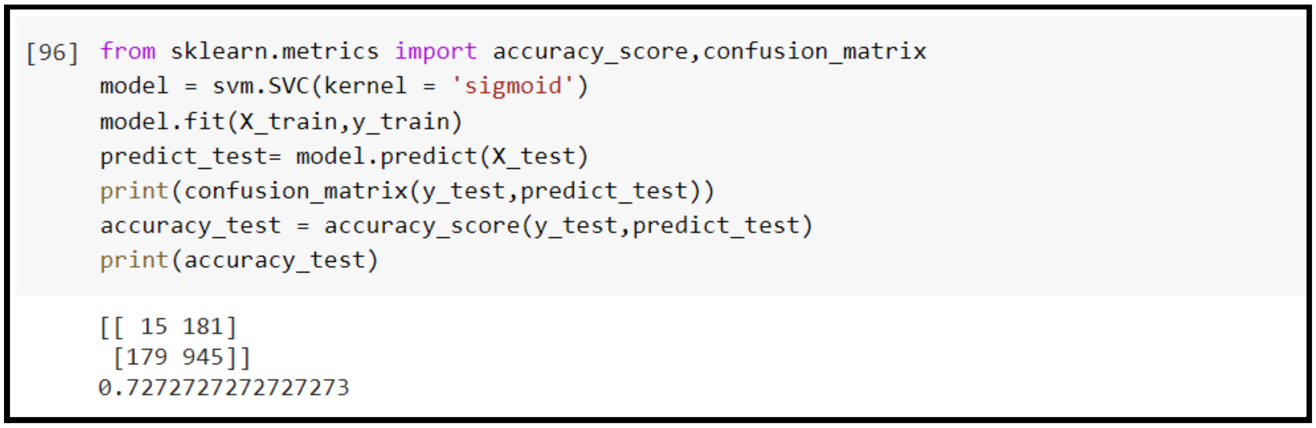
➔ Finally, we get a confusion matrix and accuracy as output.



```
from sklearn.metrics import accuracy_score,confusion_matrix
model = svm.SVC()
model.fit(X_train,y_train)
predict_test= model.predict(X_test)
print(confusion_matrix(y_test,predict_test))
accuracy_test = accuracy_score(y_test,predict_test)
print(accuracy_test)
mse = np.sum((predict_test-y_test)**2)/len(predict_test)
# print('mse',mse)
# print('rmse',mse**0.5)
```

```
[[   34  194]
 [    6 1086]]
0.8484848484848485
```

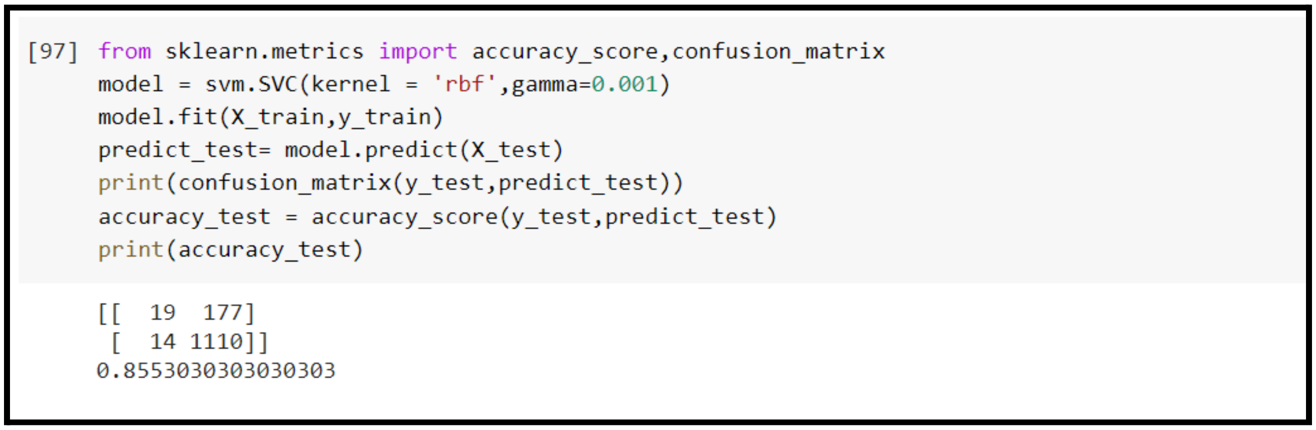
- Sigmoid Kernel



```
[96] from sklearn.metrics import accuracy_score,confusion_matrix
     model = svm.SVC(kernel = 'sigmoid')
     model.fit(X_train,y_train)
     predict_test= model.predict(X_test)
     print(confusion_matrix(y_test,predict_test))
     accuracy_test = accuracy_score(y_test,predict_test)
     print(accuracy_test)
```

```
[[ 15 181]
 [179 945]]
0.7272727272727273
```

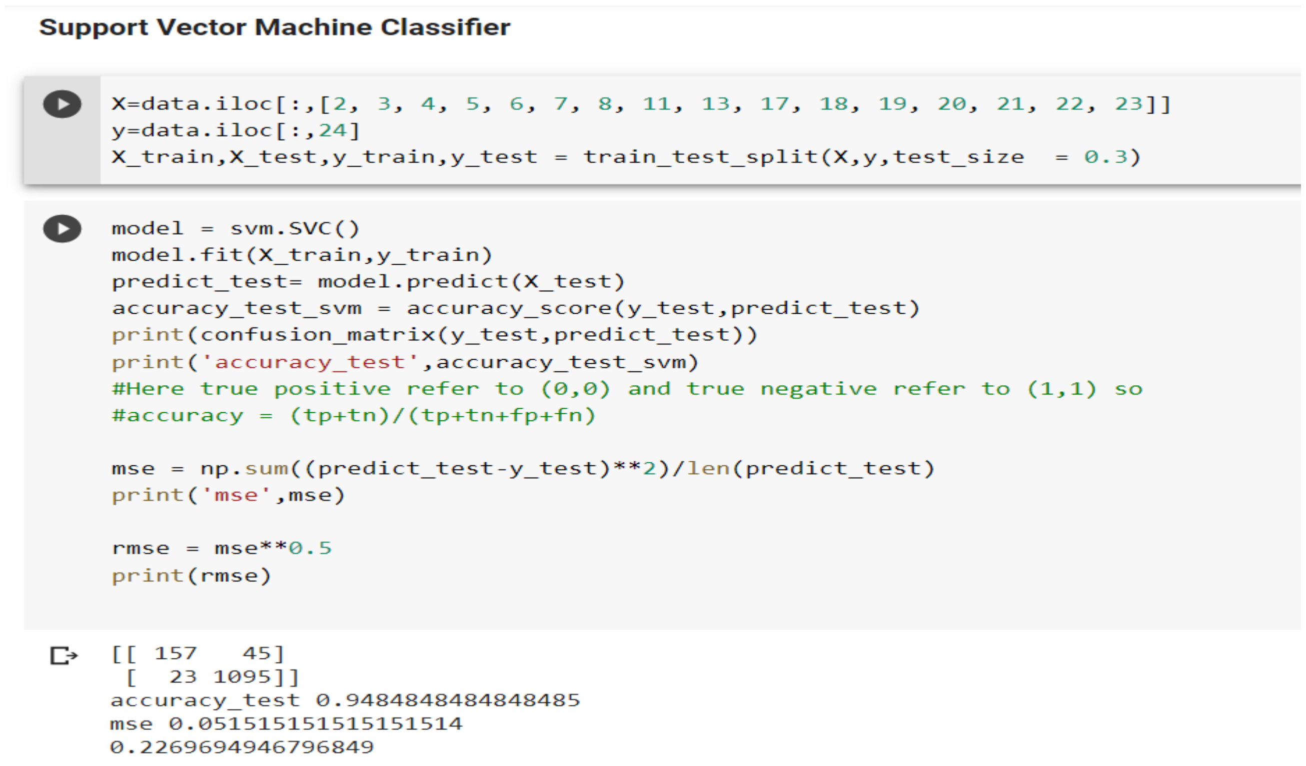
- RBF Kernel



```
[97] from sklearn.metrics import accuracy_score,confusion_matrix
     model = svm.SVC(kernel = 'rbf',gamma=0.001)
     model.fit(X_train,y_train)
     predict_test= model.predict(X_test)
     print(confusion_matrix(y_test,predict_test))
     accuracy_test = accuracy_score(y_test,predict_test)
     print(accuracy_test)

     [[  19  177]
      [  14 1110]]
     0.8553030303030303
```

- Support Vector Machine After Feature Selection

  - ➔ After Feature selection, we will get a definite number of attributes or fields which are highly suitable for prediction.
  - ➔ We will choose only those fields using the data.iloc function.
  - ➔ The remaining process remains the same which includes generating model, fitting the data, predicting the results.


Support Vector Machine Classifier

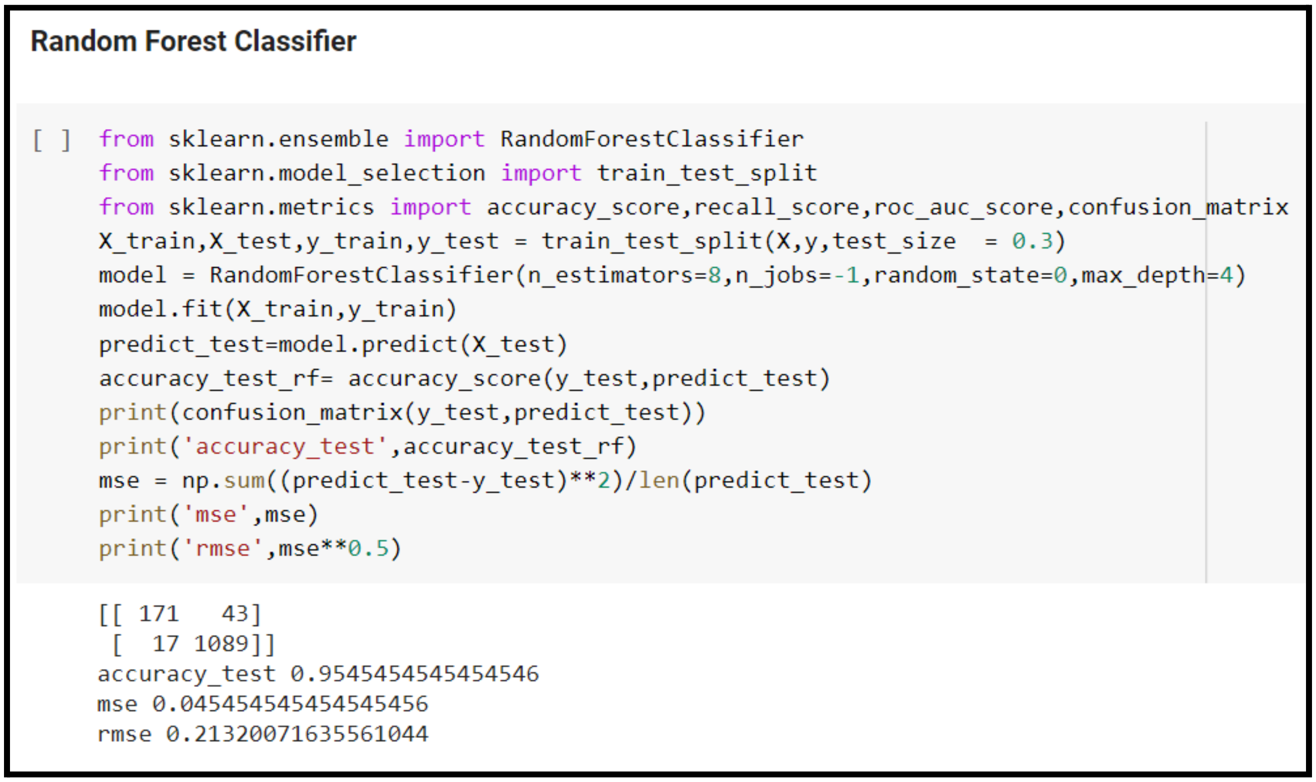
```
X=data.iloc[:,[2, 3, 4, 5, 6, 7, 8, 11, 13, 17, 18, 19, 20, 21, 22, 23]]
y=data.iloc[:,24]
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size  = 0.3)
```

```
model = svm.SVC()
model.fit(X_train,y_train)
predict_test= model.predict(X_test)
accuracy_test_svm = accuracy_score(y_test,predict_test)
print(confusion_matrix(y_test,predict_test))
print('accuracy_test',accuracy_test_svm)
#Here true positive refer to (0,0) and true negative refer to (1,1) so
#accuracy = (tp+tn)/(tp+tn+fp+fn)

mse = np.sum((predict_test-y_test)**2)/len(predict_test)
print('mse',mse)

rmse = mse**0.5
print(rmse)
```

```
[[ 157   45]
 [   23 1095]]
accuracy_test 0.9484848484848485
mse 0.051515151515151514
0.22696949467968949
```

Random Forest:

- ➔ We will use scikit-learn.ensemble, and we'll import Random Forest Classifier.
- ➔ Firstly, we call the RandomForestClassifier() function along with **n_estimators, max_depth, and n_jobs** as parameters used to build a Random Forest Classifier model in Machine Learning.
- ➔ We use X_train and y_train obtained above to train our model and We use the model.fit() method to fit the model.
- ➔ We test the Model by calling the function classifier.predict() by passing X_test (i.e., The test values of independent variables). We get a predict_test as a result.
- ➔ After we call the confusion_matrix() method by passing y_test and predict_test.
- ➔ We get the accuracy of the model by calling accuracy_score() by passing y_test and predict_test as parameters.
- ➔ Finally, we get a confusion matrix and accuracy as output.



Random Forest Classifier

```
[ ]  from sklearn.ensemble import RandomForestClassifier
     from sklearn.model_selection import train_test_split
     from sklearn.metrics import accuracy_score,recall_score,roc_auc_score,confusion_matrix
     X_train,X_test,y_train,y_test = train_test_split(X,y,test_size  = 0.3)
     model = RandomForestClassifier(n_estimators=8,n_jobs=-1,random_state=0,max_depth=4)
     model.fit(X_train,y_train)
     predict_test=model.predict(X_test)
     accuracy_test_rf= accuracy_score(y_test,predict_test)
     print(confusion_matrix(y_test,predict_test))
     print('accuracy_test',accuracy_test_rf)
     mse = np.sum((predict_test-y_test)**2)/len(predict_test)
     print('mse',mse)
     print('rmse',mse**0.5)
```

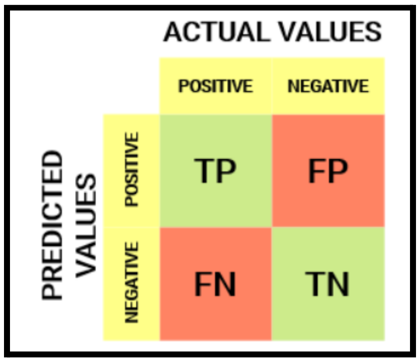
```
[[ 171   43]
 [  17 1089]]
accuracy_test 0.9545454545454546
mse 0.045454545454545456
rmse 0.21320071635561044
```

❖ Calculating Accuracies:

A Confusion matrix is an N x N matrix used for evaluating the performance of a classification model, where N is the number of target classes. The matrix compares the actual target values with those predicted by the machine learning model. This gives us a holistic view of how well our classification model is performing and what kinds of errors it is making.

Accuracy= (TP + TN) / (TP+FP+FN+TN)

We can use the accuracy_score function of sklearn.metrics to compute the accuracy of our classification model.



Confusion Matrix for 2 target classes

➔ **Mean Squared Error(MSE):**

Mean Squared Error is a model evaluation metric often used with classification models. The mean squared error of a model concerning a test set is the mean of the squared prediction errors over all instances in the test set. It is the sum, overall the data points, of the square of the difference between the predicted and actual target variables, divided by the number of data points. The prediction error is the difference between the true value and the predicted value for an instance. The value lies between 0 to ∞.

$$\text{MSE} = \frac{1}{n} \sum_{i=1}^{n} (y_i - \tilde{y}_i)^2$$

```
mse = np.sum((predict_test-y_test)**2)/len(predict_test)
print('mse',mse)
```

➔ **Root Mean Squared Error (RMSE):**

RMSE is calculated as the square root of the mean of the squared differences between actual outcomes and predictions. Squaring each error forces the values to be positive, and the square root of the mean squared error returns the error metric back to the original units for comparison.RMSE is highly affected by outlier values. Hence, make sure you’ve removed outliers from your data set before using this metric. As compared to mean absolute error, RMSE gives higher weightage and punishes large errors. The value lies between 0 to ∞.

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N} (Predicted_i - Actual_i)^2}{N}}$$

```
rmse = mse**0.5
print(rmse)
```

5.2.6. User-Interface

❖ User Interface:

User Interface is created by using HTML, CSS, Flask. Flask is used to connect the server or the system with the web environment.

To use Flask we need to install Flask and then it must be imported. In the program, we have imported the libraries needed and the flask takes the current module __name__as argument.

@app.route('/') -> '/' is the URL which is bound with the home() function. When the homepage of the web server is opened in the browser the output of this particular function will be rendered. So we will get the output of the home() function in the browser.

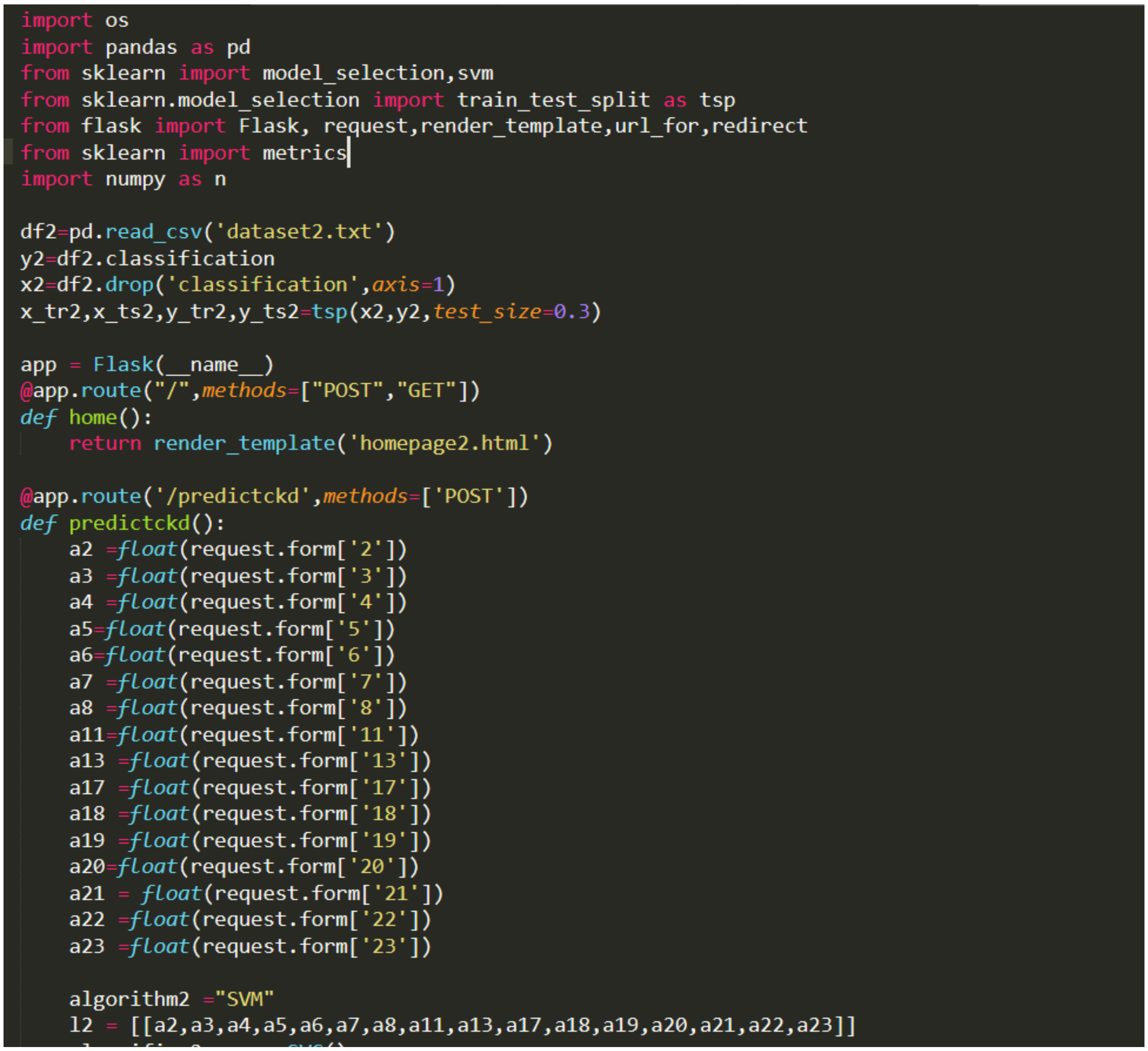
@app.route('/predictckd',methods=['POST']) ->The route decorator in the flask is basically used to bind the URL to a function. Here the URL is '/predictckd', this rule will be bound to the predictckd() function. POST is a request method supported by HTTP used by the World Wide Web.that

The data is split into train and test. In the predictckd() function the input given by the user in the web browser is taken by using request.form(). The input is normalized along with the dataset. The input values are taken as a list and are converted into an array using NumPy. We have used the Naive Bayes and Support Vector Classifier algorithms to train the model. By using the trained model the output is predicted for the given input values.

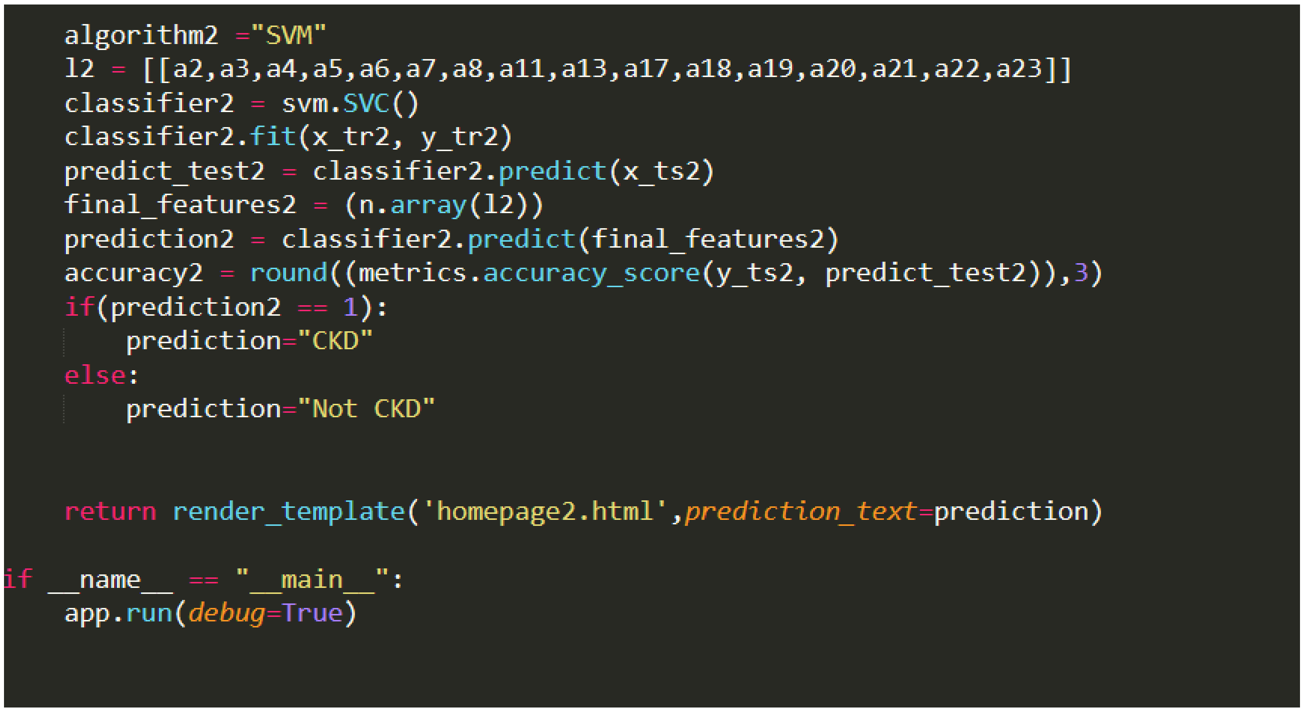
render_template() is a Flask function from the flask.templating package. render_template is used to generate output from a template file based on the Jinja2 engine that is found in the application's templates folder.

In the last we call **app.run(debug=True)** ->Flask application started by calling run() method. The run method of the flask class actually runs the application on the local server. It will generate a URL as we paste it on the browser.

Code for front end implementation using flask:

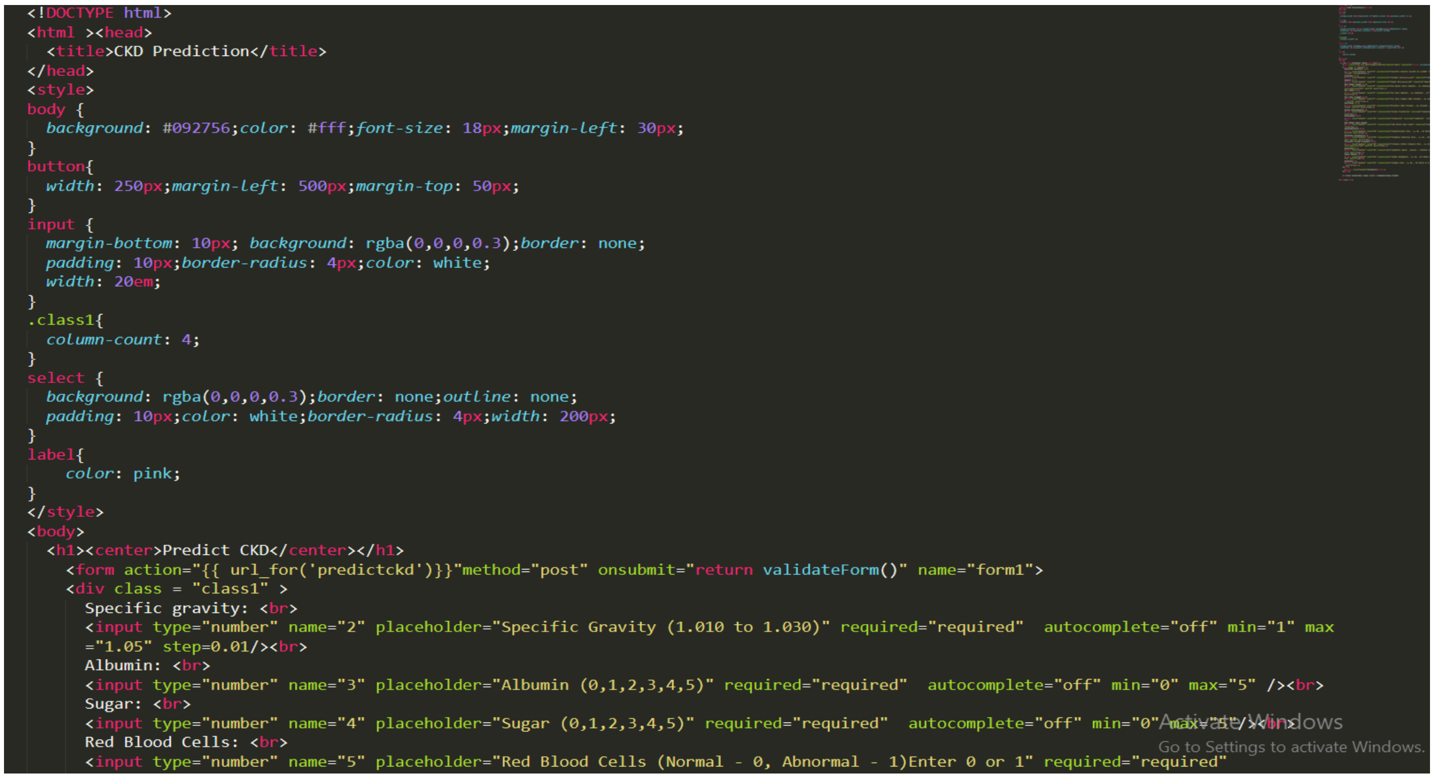


```
import os
import pandas as pd
from sklearn import model_selection,svm
from sklearn.model_selection import train_test_split as tsp
from flask import Flask, request,render_template,url_for,redirect
from sklearn import metrics
import numpy as n

df2=pd.read_csv('dataset2.txt')
y2=df2.classification
x2=df2.drop('classification',axis=1)
x_tr2,x_ts2,y_tr2,y_ts2=tsp(x2,y2,test_size=0.3)

app = Flask(__name__)
@app.route("/",methods=["POST","GET"])
def home():
    return render_template('homepage2.html')

@app.route('/predictckd',methods=['POST'])
def predictckd():
    a2 =float(request.form['2'])
    a3 =float(request.form['3'])
    a4 =float(request.form['4'])
    a5=float(request.form['5'])
    a6=float(request.form['6'])
    a7 =float(request.form['7'])
    a8 =float(request.form['8'])
    a11=float(request.form['11'])
    a13 =float(request.form['13'])
    a17 =float(request.form['17'])
    a18 =float(request.form['18'])
    a19 =float(request.form['19'])
    a20=float(request.form['20'])
    a21 = float(request.form['21'])
    a22 =float(request.form['22'])
    a23 =float(request.form['23'])

    algorithm2 ="SVM"
    l2 = [[a2,a3,a4,a5,a6,a7,a8,a11,a13,a17,a18,a19,a20,a21,a22,a23]]
```

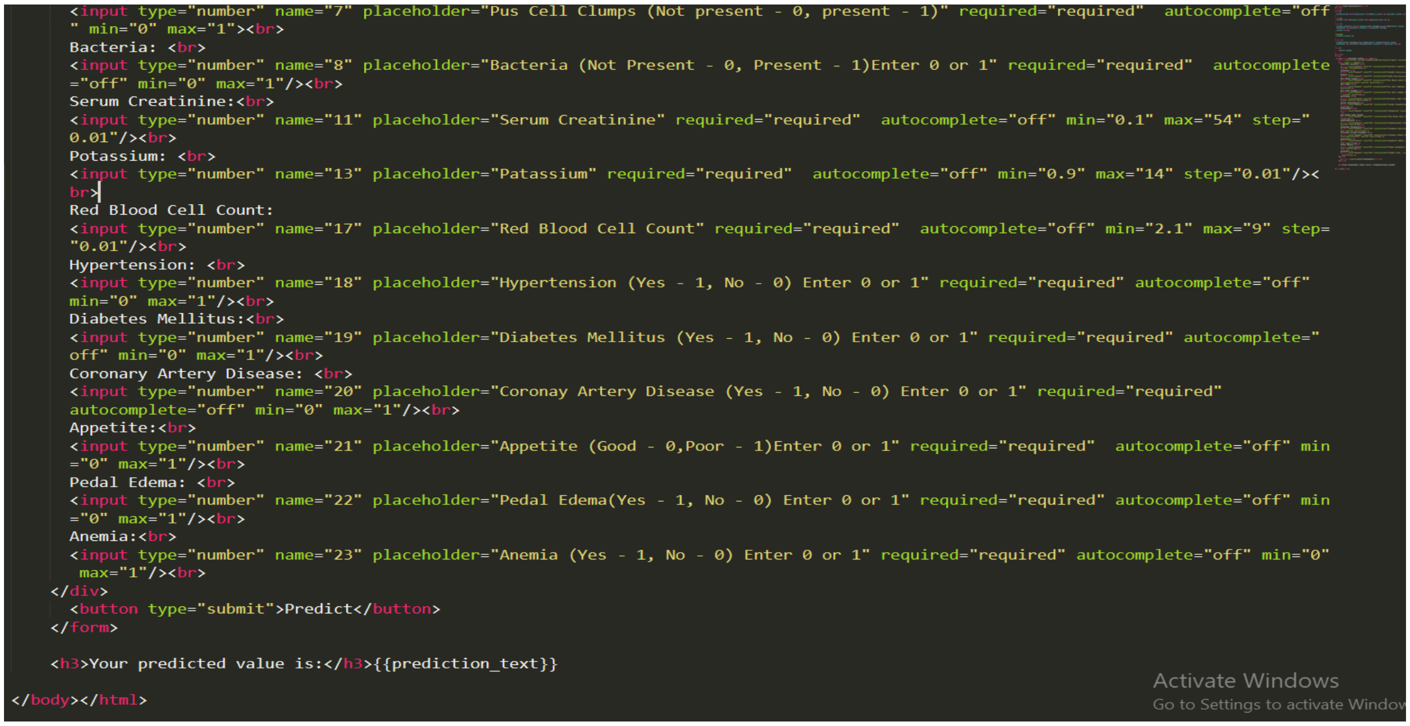


```
    algorithm2 ="SVM"
    l2 = [[a2,a3,a4,a5,a6,a7,a8,a11,a13,a17,a18,a19,a20,a21,a22,a23]]
    classifier2 = svm.SVC()
    classifier2.fit(x_tr2, y_tr2)
    predict_test2 = classifier2.predict(x_ts2)
    final_features2 = (n.array(l2))
    prediction2 = classifier2.predict(final_features2)
    accuracy2 = round((metrics.accuracy_score(y_ts2, predict_test2)),3)
    if(prediction2 == 1):
        prediction="CKD"
    else:
        prediction="Not CKD"


    return render_template('homepage2.html',prediction_text=prediction)

if __name__ == "__main__":
    app.run(debug=True)
```

Webpage implementation using HTML(HyperText Markup Language) and CSS (Cascading Style Sheets):

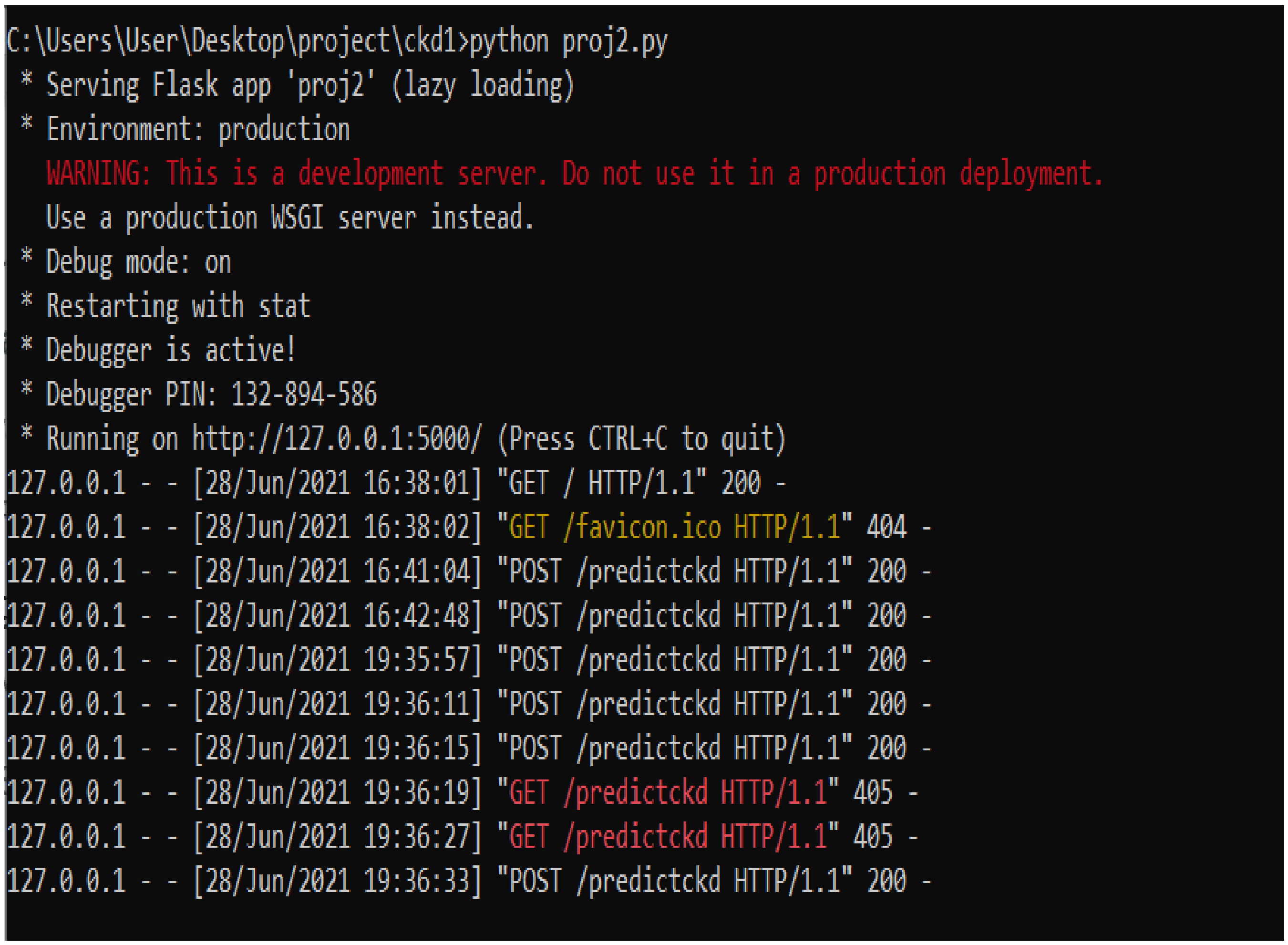


```html
<!DOCTYPE html>
<html ><head>
  <title>CKD Prediction</title>
</head>
<style>
body {
  background: #092756;color: #fff;font-size: 18px;margin-left: 30px;
}
button{
  width: 250px;margin-left: 500px;margin-top: 50px;
}
input {
  margin-bottom: 10px; background: rgba(0,0,0,0.3);border: none;
  padding: 10px;border-radius: 4px;color: white;
  width: 20em;
}
.class1{
  column-count: 4;
}
select {
  background: rgba(0,0,0,0.3);border: none;outline: none;
  padding: 10px;color: white;border-radius: 4px;width: 200px;
}
label{
    color: pink;
}
</style>
<body>
  <h1><center>Predict CKD</center></h1>
    <form action="{{ url_for('predictckd')}}"method="post" onsubmit="return validateForm()" name="form1">
    <div class = "class1" >
      Specific gravity: <br>
      <input type="number" name="2" placeholder="Specific Gravity (1.010 to 1.030)" required="required"  autocomplete="off" min="1" max
      ="1.05" step=0.01/><br>
      Albumin: <br>
      <input type="number" name="3" placeholder="Albumin (0,1,2,3,4,5)" required="required"  autocomplete="off" min="0" max="5" /><br>
      Sugar: <br>
      <input type="number" name="4" placeholder="Sugar (0,1,2,3,4,5)" required="required"  autocomplete="off" min="0" max="5"/><br>
      Red Blood Cells: <br>
      <input type="number" name="5" placeholder="Red Blood Cells (Normal - 0, Abnormal - 1)Enter 0 or 1" required="required"
```



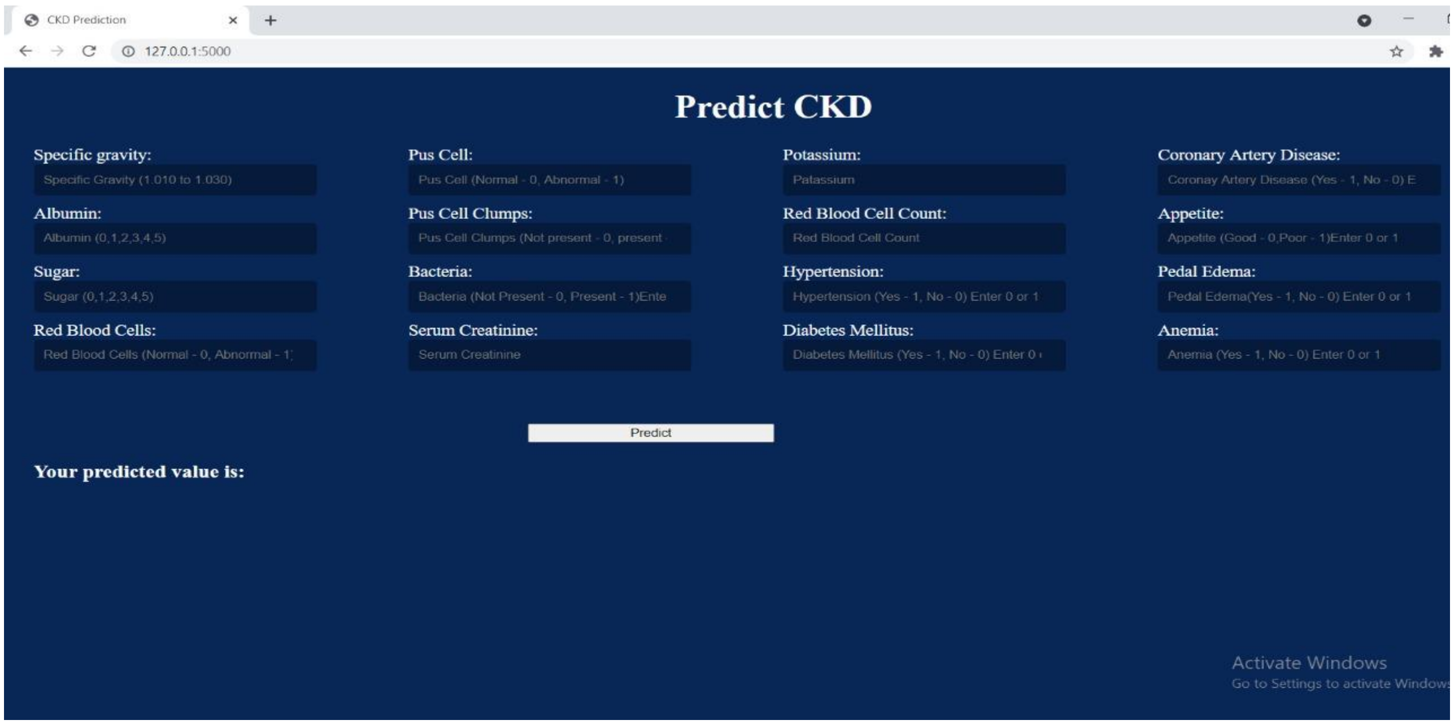
```html
      <input type="number" name="7" placeholder="Pus Cell Clumps (Not present - 0, present - 1)" required="required"  autocomplete="off
      " min="0" max="1"><br>
      Bacteria: <br>
      <input type="number" name="8" placeholder="Bacteria (Not Present - 0, Present - 1)Enter 0 or 1" required="required"  autocomplete
      ="off" min="0" max="1"/><br>
      Serum Creatinine:<br>
      <input type="number" name="11" placeholder="Serum Creatinine" required="required"  autocomplete="off" min="0.1" max="54" step="
      0.01"/><br>
      Potassium: <br>
      <input type="number" name="13" placeholder="Patassium" required="required"  autocomplete="off" min="0.9" max="14" step="0.01"/><
      br>
      Red Blood Cell Count:
      <input type="number" name="17" placeholder="Red Blood Cell Count" required="required"  autocomplete="off" min="2.1" max="9" step=
      "0.01"/><br>
      Hypertension: <br>
      <input type="number" name="18" placeholder="Hypertension (Yes - 1, No - 0) Enter 0 or 1" required="required" autocomplete="off"
      min="0" max="1"/><br>
      Diabetes Mellitus:<br>
      <input type="number" name="19" placeholder="Diabetes Mellitus (Yes - 1, No - 0) Enter 0 or 1" required="required" autocomplete="
      off" min="0" max="1"/><br>
      Coronary Artery Disease: <br>
      <input type="number" name="20" placeholder="Coronay Artery Disease (Yes - 1, No - 0) Enter 0 or 1" required="required"
      autocomplete="off" min="0" max="1"/><br>
      Appetite:<br>
      <input type="number" name="21" placeholder="Appetite (Good - 0,Poor - 1)Enter 0 or 1" required="required"  autocomplete="off" min
      ="0" max="1"/><br>
      Pedal Edema: <br>
      <input type="number" name="22" placeholder="Pedal Edema(Yes - 1, No - 0) Enter 0 or 1" required="required" autocomplete="off" min
      ="0" max="1"/><br>
      Anemia:<br>
      <input type="number" name="23" placeholder="Anemia (Yes - 1, No - 0) Enter 0 or 1" required="required" autocomplete="off" min="0"
       max="1"/><br>
    </div>
      <button type="submit">Predict</button>
    </form>

    <h3>Your predicted value is:</h3>{{prediction_text}}

</body></html>
```

5.3. Screens



```
C:\Users\User\Desktop\project\ckd1>python proj2.py
 * Serving Flask app 'proj2' (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: on
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 132-894-586
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
127.0.0.1 - - [28/Jun/2021 16:38:01] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [28/Jun/2021 16:38:02] "GET /favicon.ico HTTP/1.1" 404 -
127.0.0.1 - - [28/Jun/2021 16:41:04] "POST /predictckd HTTP/1.1" 200 -
127.0.0.1 - - [28/Jun/2021 16:42:48] "POST /predictckd HTTP/1.1" 200 -
127.0.0.1 - - [28/Jun/2021 19:35:57] "POST /predictckd HTTP/1.1" 200 -
127.0.0.1 - - [28/Jun/2021 19:36:11] "POST /predictckd HTTP/1.1" 200 -
127.0.0.1 - - [28/Jun/2021 19:36:15] "POST /predictckd HTTP/1.1" 200 -
127.0.0.1 - - [28/Jun/2021 19:36:19] "GET /predictckd HTTP/1.1" 405 -
127.0.0.1 - - [28/Jun/2021 19:36:27] "GET /predictckd HTTP/1.1" 405 -
127.0.0.1 - - [28/Jun/2021 19:36:33] "POST /predictckd HTTP/1.1" 200 -
```

Screen1: Execution of the project. http://127.0.0.1:5000/ needs to be pasted in the browser.



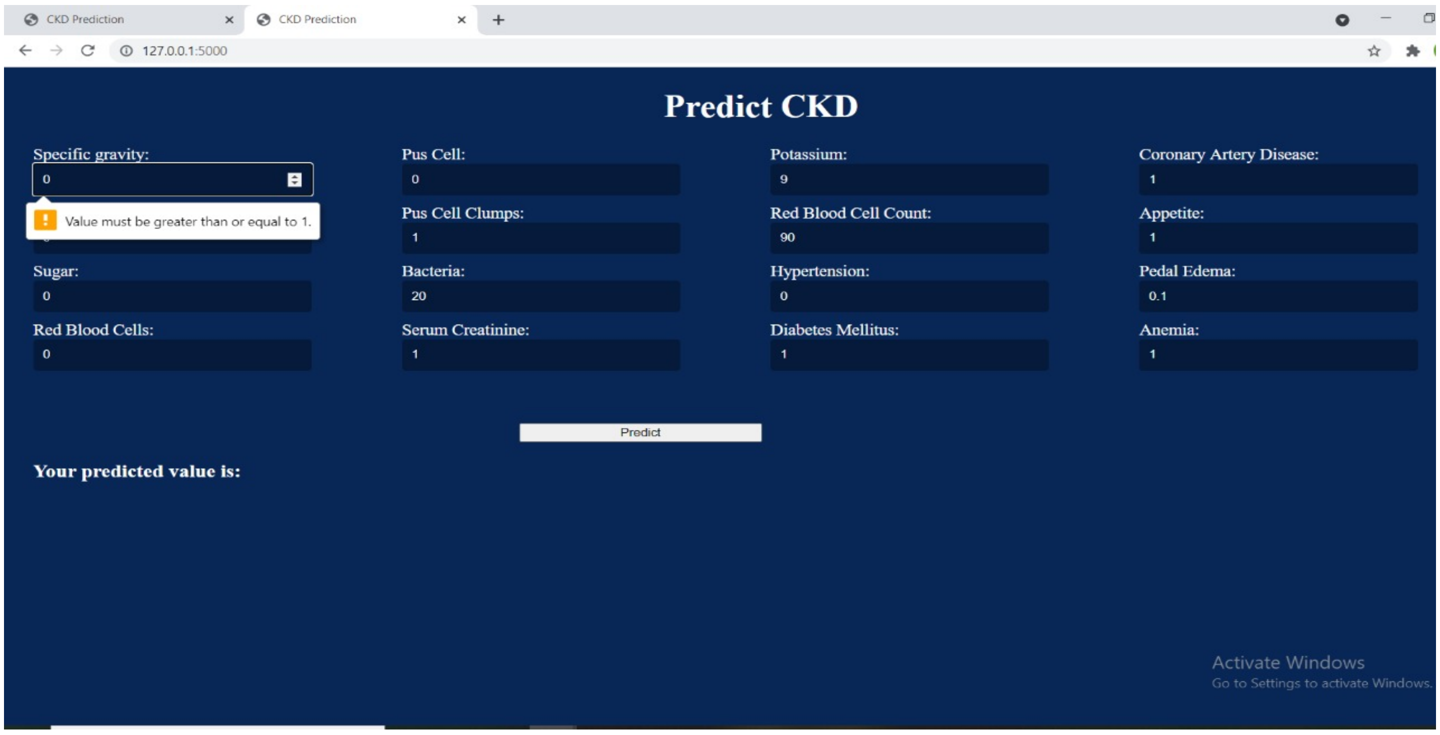
Predict CKD

Specific gravity: Specific Gravity (1.010 to 1.030)
Albumin: Albumin (0,1,2,3,4,5)
Sugar: Sugar (0,1,2,3,4,5)
Red Blood Cells: Red Blood Cells (Normal - 0, Abnormal - 1)
Pus Cell: Pus Cell (Normal - 0, Abnormal - 1)
Pus Cell Clumps: Pus Cell Clumps (Not present - 0, present
Bacteria: Bacteria (Not Present - 0, Present - 1)Ente
Serum Creatinine: Serum Creatinine
Potassium: Patassium
Red Blood Cell Count: Red Blood Cell Count
Hypertension: Hypertension (Yes - 1, No - 0) Enter 0 or 1
Diabetes Mellitus: Diabetes Mellitus (Yes - 1, No - 0) Enter 0
Coronary Artery Disease: Coronay Artery Disease (Yes - 1, No - 0) E
Appetite: Appetite (Good - 0,Poor - 1)Enter 0 or 1
Pedal Edema: Pedal Edema(Yes - 1, No - 0) Enter 0 or 1
Anemia: Anemia (Yes - 1, No - 0) Enter 0 or 1

Predict

Your predicted value is:

Screen 2: User Interface



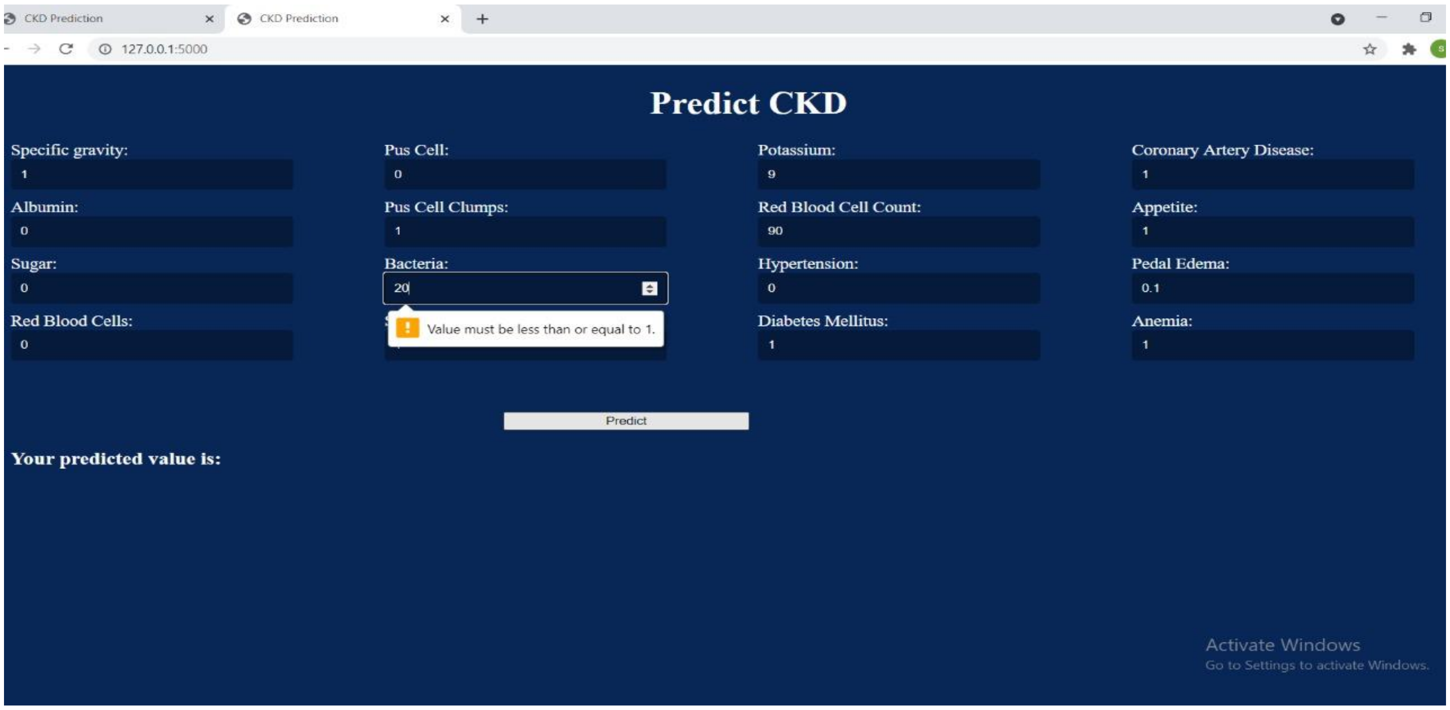
Predict CKD

Specific gravity: 0
Value must be greater than or equal to 1.
Sugar: 0
Red Blood Cells: 0
Pus Cell: 0
Pus Cell Clumps: 1
Bacteria: 20
Serum Creatinine: 1
Potassium: 9
Red Blood Cell Count: 90
Hypertension: 0
Diabetes Mellitus: 1
Coronary Artery Disease: 1
Appetite: 1
Pedal Edema: 0.1
Anemia: 1

Predict

Your predicted value is:

Screen 3: Validating entered Values


CKD Prediction

127.0.0.1:5000

Predict CKD

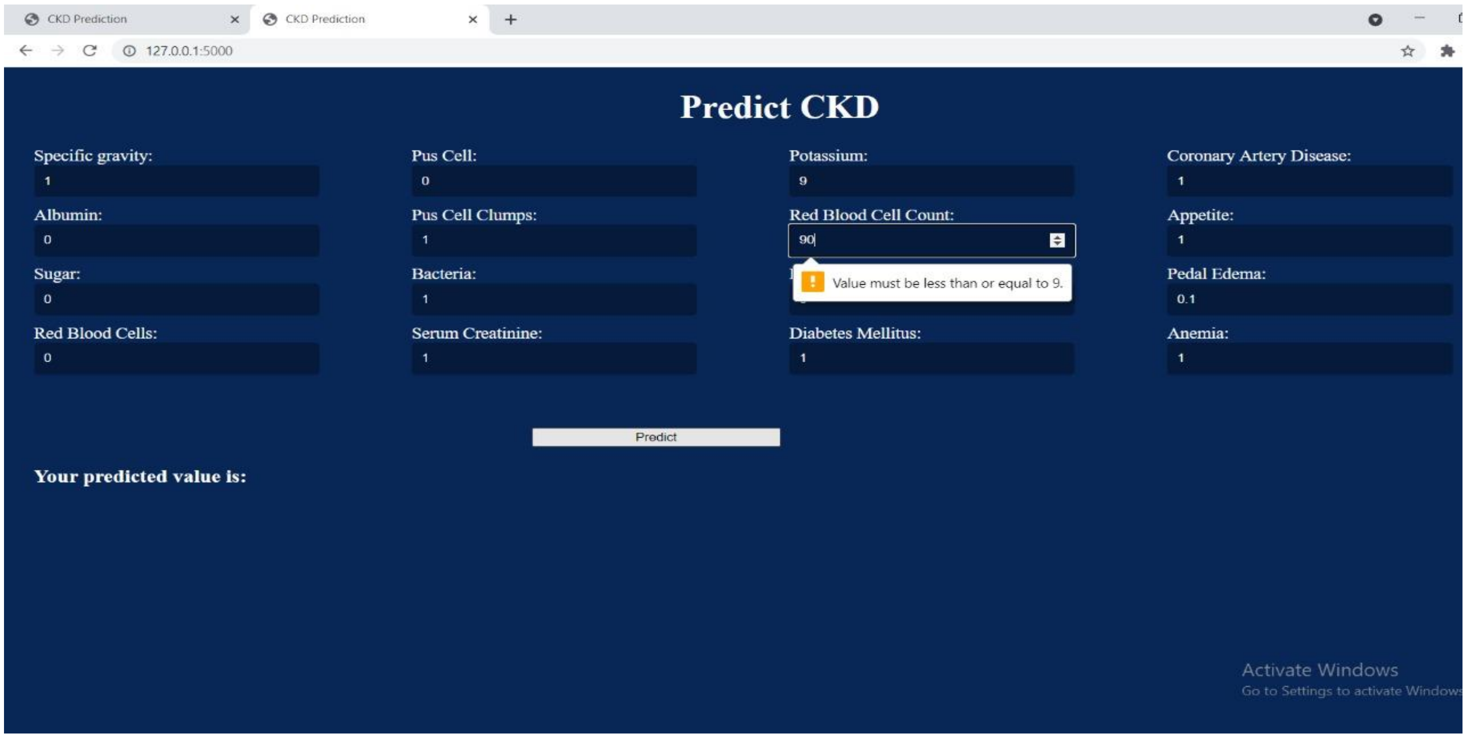
Specific gravity: 1	Pus Cell: 0	Potassium: 9	Coronary Artery Disease: 1
Albumin: 0	Pus Cell Clumps: 1	Red Blood Cell Count: 90	Appetite: 1
Sugar: 0	Bacteria: 20	Hypertension: 0	Pedal Edema: 0.1
Red Blood Cells: 0		Diabetes Mellitus: 1	Anemia: 1

Value must be less than or equal to 1.

Predict

Your predicted value is:

Screen 4: Validating entered values


CKD Prediction

127.0.0.1:5000

Predict CKD

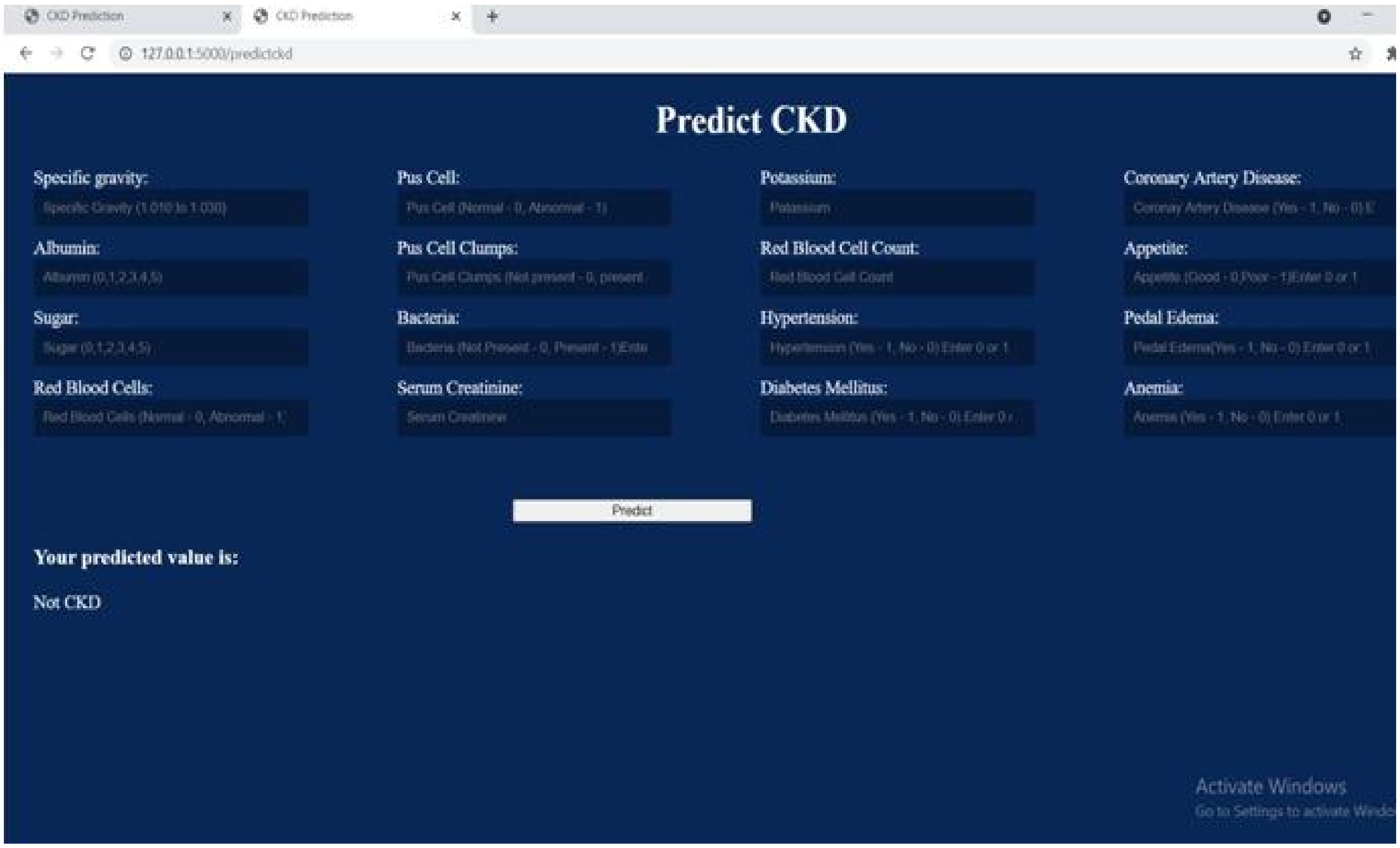
Specific gravity: 1	Pus Cell: 0	Potassium: 9	Coronary Artery Disease: 1
Albumin: 0	Pus Cell Clumps: 1	Red Blood Cell Count: 90	Appetite: 1
Sugar: 0	Bacteria: 1		Pedal Edema: 0.1
Red Blood Cells: 0	Serum Creatinine: 1	Diabetes Mellitus: 1	Anemia: 1

Value must be less than or equal to 9.

Predict

Your predicted value is:

Screen 5: Validating entered values



Predict CKD

Specific gravity: Specific Gravity (1.010 to 1.030)

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

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

Red Blood Cells: Red Blood Cells (Normal - 0, Abnormal - 1,

Pus Cell: Pus Cell (Normal - 0, Abnormal - 1)

Pus Cell Clumps: Pus Cell Clumps (Not present - 0, present

Bacteria: Bacteria (Not Present - 0, Present - 1)Ente

Serum Creatinine: Serum Creatinine

Potassium: Potassium

Red Blood Cell Count: Red Blood Cell Count

Hypertension: Hypertension (Yes - 1, No - 0) Enter 0 or 1

Diabetes Mellitus: Diabetes Mellitus (Yes - 1, No - 0) Enter 0 ‹

Coronary Artery Disease: Coronary Artery Disease (Yes - 1, No - 0) E

Appetite: Appetite (Good - 0,Poor - 1)Enter 0 or 1

Pedal Edema: Pedal Edema(Yes - 1, No - 0) Enter 0 or 1

Anemia: Anemia (Yes - 1, No - 0) Enter 0 or 1

Your predicted value is:

Not CKD

Screen 6: Result

SYSTEM TESTING

6. System Testing

6.1. Introduction

System Testing is a type of software testing that is performed on a complete integrated system to evaluate the compliance of the system with the corresponding requirements.

System testing detects defects within both the integrated units and the whole system. The result of system testing is the observed behavior of a component or a system when it is tested.

System Testing is carried out on the whole system in the context of either system requirement specifications or functional requirement specifications or in the context of both. System testing tests the design and behavior of the system and also the expectations of the customer. It is performed to test the system beyond the bounds mentioned in the software requirements specification (SRS).

System Testing is basically performed by a testing team that is independent of the development team that helps to test the quality of the system impartially. It has both functional and non-functional testing.

6.2. Testing Methods

Every software development life cycle includes testing as one of the essential steps to creating a great product. Various aspects of the software system are checked using different tests that can be divided into functional and non-functional.

Functional Testing:

Functional testing verifies that the operational execution of a program or mobile app happens according to the technical and business requirements. Only if every feature of a software system works correctly, can it pass a functional test.

- Unit testing - Unit tests are done to check individual units or components of the system.
- Integration testing - To test whether multiple software components function well together as a group, you will need integration testing.

- Interface testing - To check whether the users will interact with the interface as expected, you need interface testing. To conduct it, you gather an end-user group that will perform specific tasks. It helps identify the elements of the software that the user often uses and whether these elements behave according to the requirements. It is essential not to confuse interface testing with usability testing. User interface testing checks that the interface works fine and passes the data to the system. Usability testing is done to see whether end-users will find the application convenient and useful.

Non Functional Testing:

Non-functional testing checks all the aspects not covered in functional tests. It includes the performance, usability, scalability, and reliability of the software.

- Availability testing - You need to test how often the product is going to be used and whether it is accessible when the users need it. For example, you want to minimize failure events or predict how much time the repair can take before the system goes back to normal.
- Compatibility testing - This type of testing validates how seamlessly the product operates with other components: OS, browsers, hardware, and so on.
- Scalability testing - The software testing procedure that ensures that the product can grow in proportion to the increasing demands of the end-users is called scalability testing.
- Portability testing - It determines how easy it is to transport a software component or application from one hardware or OS to another.

6.3. Test Cases

Test Case1:

row =[1.01, 3.0, 2.0, 0.0, 1.0, 1.0, 1.0, 4.1, 6.4, 2.6, 1.0, 1.0, 1.0, 1.0, 1.0, 0.0]

Sg - 1.01	Pot - 6.4
Al - 3	Rc - 2.6
Su - 2	Htn - 1
Rbc - 0	Dm - 1
Pc - 1	Cad - 1
Pcc - 1	Appetite - 1
Ba - 1	Pe - 1
Sc - 4.1	Ane - 0

Output:

```
1
```

Test Case2:

row = [1.01, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,1.0, 6.5, 8.0, 0.0, 0.0, 0.0, 1.0, 1.0, 0.0]

Sg - 1.01	Pot - 6.5
Al - 0	Rc - 8
Su - 0	Htn - 0
Rbc - 0	Dm - 0
Pc - 0	Cad - 0
Pcc - 0	Appetite - 1
Ba - 0	Pe - 1
Sc - 1	Ane - 0

Output:

```
0
```

CONCLUSION

7. Conclusion:

Our system will finally predict whether a person is suffering from CKD or not. Using modern-day technologies like machine learning to solve basic tasks which are done with a blink of an eye. In the attempt to develop a model for predicting the result based on the various parameters of our body condition, Many conclusions can be drawn from the outcomes of the different models built. The parameters which are most useful for predicting the result are only some of them in the dataset. With some of the parameters, all of them are also considered while providing the result. One conclusion drawn while visualizing the data is, we found that dm is highly correlated with htn. The system uses three Supervised Machine learning algorithms and gives the best result based on accuracy. The algorithms used are Support Vector Classifier, Random Forest, and Naïve Bayes. The accuracies of the three algorithms will be compared and the one giving the best and accurate output will be selected. The entered values are tested against the selected algorithm and the appropriate result is displayed. The proposed model is justified by a properly made dataset and machine learning algorithms.

Future Enhancements

The dataset used in this project is small even when we perform data augmentation, So in the future, we aim to validate our results by using a big dataset or compare the results using another dataset that contains the same features. We would like to predict CKD using lesser features than those which are considered at present. We also like to work on other fields which define whether any of the family members of the user is affected by kidney failure or any other kidney related issues along with the present fields.

BIBLIOGRAPHY

8. Bibliography:

About Kidney Disease

https://www.kidneyfund.org/kidney-disease/chronic-kidney-disease-ckd/

Stages of CKD

https://www.kidneyfund.org/kidney-disease/chronic-kidney-disease-ckd/stages-of-chronic-kidney-disease/

Dataset

https://www.kaggle.com/akshayksingh/kidney-disease-dataset

Description of Dataset

https://archive.ics.uci.edu/ml/datasets/chronic_kidney_disease

Download Dataset from Google Colaboratory

https://www.youtube.com/watch?v=e8Syou92H8U

About Classification

https://machinelearningmastery.com/types-of-classification-in-machine-learning/

https://www.analytixlabs.co.in/blog/classification-in-machine-learning/

Feature Selection

https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.SequentialFeatureSelector.html

https://www.datacamp.com/community/tutorials/feature-selection-python

System Design

https://www.tutorialspoint.com/uml/index.htm

https://ijarcce.com/wp-content/uploads/2018/11/IJARCCE.2018.71021.pdf

System Testing

https://u-tor.com/topic/functional-vs-non-functional

SVM Kernels

https://dataaspirant.com/svm-kernels/

Random Forest Classifier

https://scikit-learn.org/stable/modules/generated/

Heatmap

https://towardsdatascience.com/tagged/heatmap

Installation of Python in Windows

https://phoenixnap.com/kb/how-to-install-python-3-windows

Usage of Flask

https://pythonbasics.org/flask-template-data/

Performance Metrics

https://machinelearningmastery.com/implement-machine-learning-algorithm-performance-metrics-scratch-python/

APPENDIX

9. Appendix

9.1. Introduction to Python

Python is a general-purpose interpreted, interactive, object-oriented, and high-level programming language. It was created by Guido van Rossum during 1985- 1990. Like Perl, Python source code is also available under the GNU General Public License (GPL). This tutorial gives enough understanding of the Python programming language.

Python is a high-level, interpreted, interactive, and object-oriented scripting language. Python is designed to be highly readable. It uses English words frequently whereas other languages use punctuation, and it has fewer syntactic constructions than other languages.

Python is a very popular programming language. Python programming language is being used in web development, Machine Learning applications, along all cutting edge technology in the Software Industry. Python Programming Language is very well suited for beginners and experienced programmers with other programming languages like C++ and Java.

Python is one of the most popular and fastest-growing programming languages. Inherently, it is interpreted, high-level, general-purpose, and object-oriented scripting language, which means the following:

➢ **Interpreted**

An interpreter processes the source file at runtime, it reads the lines of code one by one and performs what is said. Similar to Perl and PHP, Python does not require that you compile your program before executing it. So, you do not have to invoke a compiler. Instead of running the compiler that helps turn source files into compiled class files, you simply run a .py file. Python byte code compilation is automatic and entirely implicit.

➢ **High-level**

Python relies on easy-to-read structures that are later translated into a low-level language, the original code that is run on a computer's central processing unit (CPU). A high-level language is intended to be used by a programmer and the written code is further interpreted into a low-level language. Like C++ or Java, before running, Python has to be processed. This enables Python's portability — it can run on different kinds of computers with nearly no modifications.

- **General-purpose**

Python can be used for nearly everything. It is applicable to almost every field for a variety of tasks. Be it the execution of such short-term tasks as software testing or long-term product development that involves roadmap planning, Python works well for them all, it is applicable all over the map. Its roles are unlimited. It is popular not only among software engineers, but also among specialists in other fields: mathematics, data analysis, science, accounting, and network engineering. Likewise, Python cliques with young people because it's a very beginner-friendly scripting language.

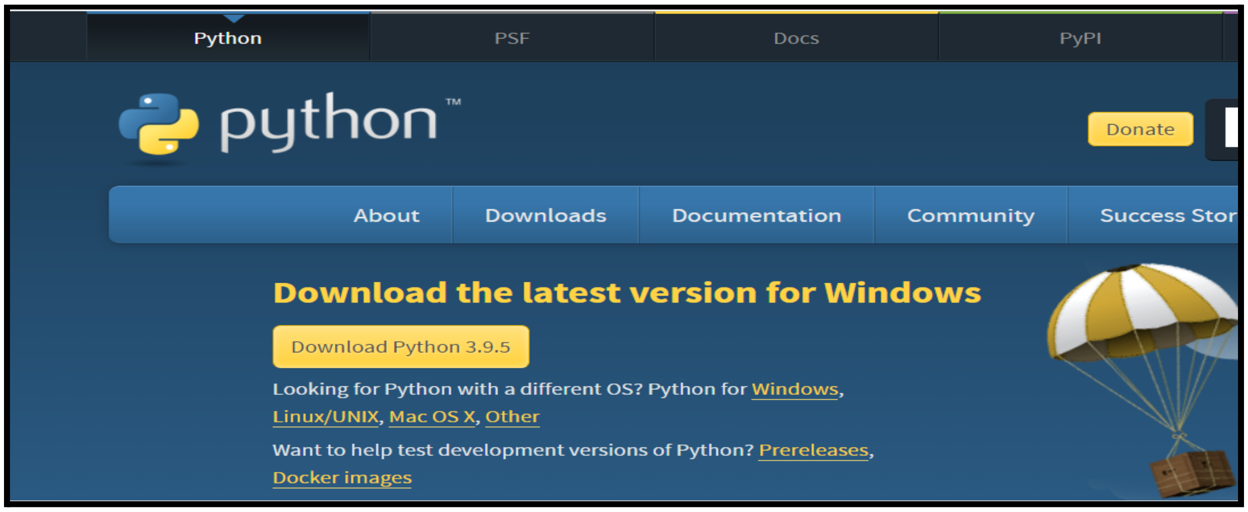
- **Object-oriented**

This programming paradigm gives an overall orientation towards scripting and powerful code structuring. This object-oriented approach allows thinking of problems in terms of classes and objects. Then, objects are composed in such a way to make up complex computer programs. Besides object-oriented programming, Python also supports a procedural paradigm. With OOP being only one of the options, you can make Python programming more advanced by going for an object-oriented programming approach. Developers can create reusable patterns of code thus curtailing redundancy in development projects.

Installation of Python in Windows

Step1: Downloading

Click on https://www.python.org/downloads/

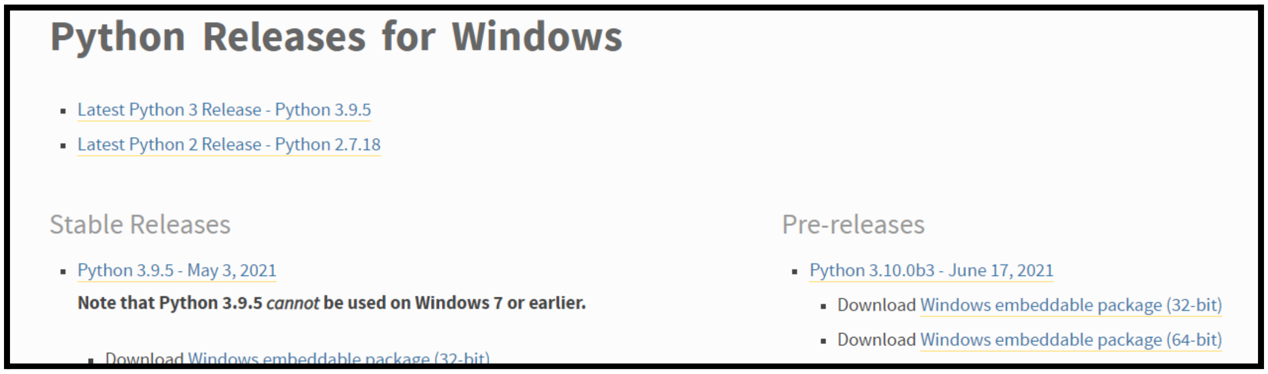


Download the latest version for Windows

Looking for Python with a different OS? Python for Windows, Linux/UNIX, Mac OS X, Other

Want to help test development versions of Python? Prereleases, Docker images

Click the **Windows** link



Python Releases for Windows

- Latest Python 3 Release - Python 3.9.5
- Latest Python 2 Release - Python 2.7.18

Stable Releases

- Python 3.9.5 - May 3, 2021

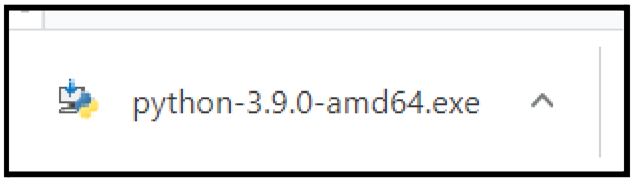
  Note that Python 3.9.5 *cannot* be used on Windows 7 or earlier.

  - Download Windows embeddable package (32-bit)

Pre-releases

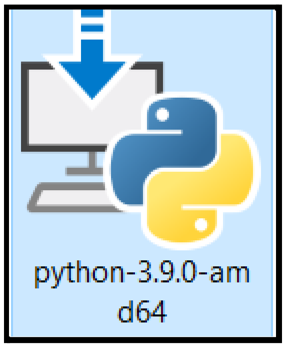
- Python 3.10.0b3 - June 17, 2021
  - Download Windows embeddable package (32-bit)
  - Download Windows embeddable package (64-bit)

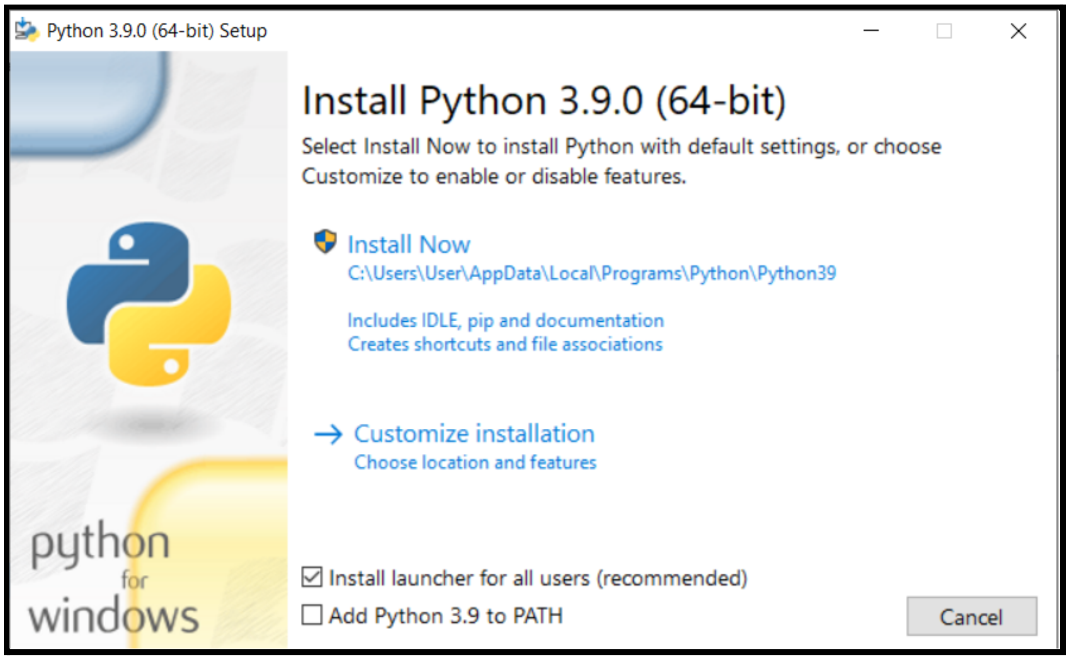
Click on the **Download Windows x86-64 executable installer** link under the **Stable Releases**.



The file named **python-3.9.0-amd64.exe** should start downloading into your download folder.

Step 2: Double click on the downloaded file.




Python 3.9.0 (64-bit) Setup

Install Python 3.9.0 (64-bit)

Select Install Now to install Python with default settings, or choose Customize to enable or disable features.

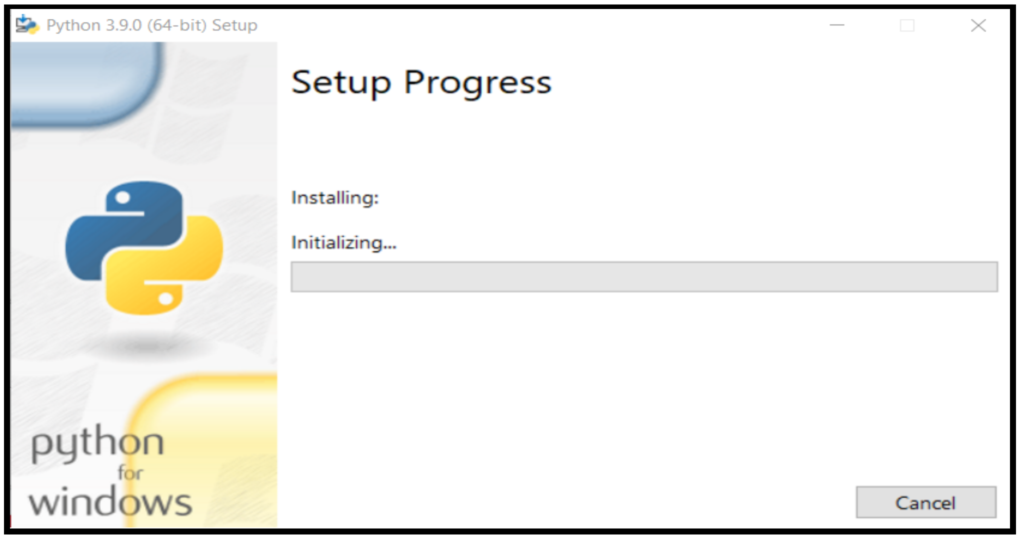
Install Now
C:\Users\User\AppData\Local\Programs\Python\Python39

Includes IDLE, pip and documentation
Creates shortcuts and file associations

Customize installation
Choose location and features

[✓] Install launcher for all users (recommended)
[ ] Add Python 3.9 to PATH

Step 3: Click on Install Now

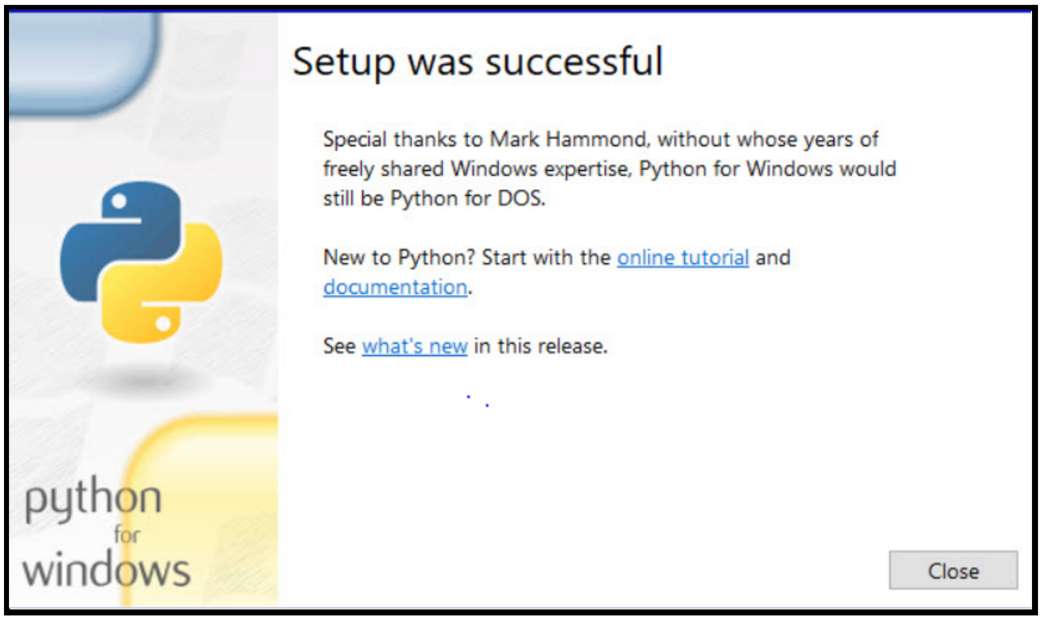

Python 3.9.0 (64-bit) Setup

Setup Progress

Installing:

Initializing...

Setup pop-up window will appear with a Setup successfully message.


Setup was successful

Special thanks to Mark Hammond, without whose years of freely shared Windows expertise, Python for Windows would still be Python for DOS.

New to Python? Start with the online tutorial and documentation.

See what's new in this release.

Step 4: Click on the **Close** Button

9.2. Introduction and Installation of Flask

Flask is a micro web framework written in Python. It is classified as a microframework because it does not require particular tools or libraries. It has no database abstraction layer, form validation, or any other components where pre-existing third-party libraries provide common functions. Flask is considered more Pythonic than the Django web framework because in common situations the equivalent Flask web application is more explicit. Flask is also easy to get started with as a beginner because there is little boilerplate code for getting a simple app up and running.

Use the following command to install Flask:

> **pip install flask**

9.3. HTML and CSS

HTML

The HyperText Markup Language or HTML is the standard markup language for documents designed to be displayed in a web browser.HTML elements are the building blocks of HTML pages. With HTML constructs, images and other objects such as interactive forms may be embedded into the rendered page. HTML provides a means to create structured documents by denoting structural semantics for text such as headings, paragraphs, lists, links, quotes, and other items. HTML elements are delineated by *tags*, written using angle brackets. Tags such as `<img />` and `<input />` directly introduce content into the page. Other tags such as `<p>` surround and provide information about document text and include other sub-elements. Browsers do not display the HTML tags but use them to interpret the content of the page.

CSS

Cascading Style Sheets (CSS) is a style sheet language used for describing the presentation of a document written in a markup language such as HTML. CSS is a cornerstone technology of the World Wide Web, alongside HTML and JavaScript.

CSS is designed to enable the separation of presentation and content, including layout, colors, and fonts. This separation can improve content accessibility, provide more flexibility and control in the specification of presentation characteristics, enable multiple web pages to share formatting by specifying the relevant CSS in a separate .css file which reduces complexity and repetition in the structural content as well as enabling the .css file to be cached to improve the page load speed between the pages that share the file and its formatting.

The name cascading comes from the specified priority scheme to determine which style rule applies if more than one rule matches a particular element. This cascading priority scheme is predictable.