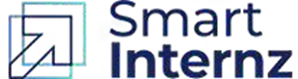


|  |  |
| --- | --- |
|  |  |



|  |  |
| --- | --- |
|  |  |
|  |  |

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

### Project Hand-out, Faculty Development Program

SmartInternz

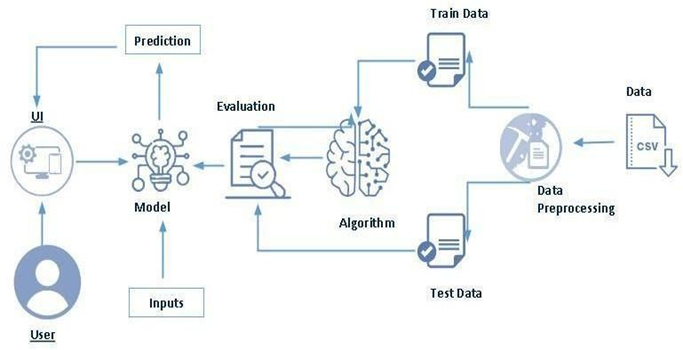
[www.smartinternz.com](http://www.smartinternz.com/)

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

Detecting Chronic Kidney Disease (CKD) through machine learning can really help quickstart the process of identifying the risk for serious knottiness. By utilizing clinical bio markers like blood pressure and serum levels, models can achieve more accurate diagnoses.

Techniques like CatBoost tackle class unbalance and enable quick predictions, which finally boosts model accuracy. Plus, embracing explainable AI (XAI) ensures transparency, paving the way for personalized healthcare and timely detection.

# Technical Architecture:



|  |  |
| --- | --- |
|  |  |
|  |  |

**Project Flow:**

1. User interacts with the UI to enter the input.
2. Entered input is analyzed by the model, which is merged.
3. Once model analyze the input the prediction is showcased on the UI.

To accomplish this, we have to complete all the action listed below,

1. Define Problem / Problem realize
   1. Specify the business problem
   2. Business demand
   3. Literature Survey
   4. Social or Business Impact.
2. Data Collection &Preparation
   1. Collect the dataset
   2. Data Preparation
3. Exploratory Data Analysis
   1. Descriptive statistical
   2. Visual Analysis
4. Model Building
   1. Training the model in multiple algorithms
   2. Testing the model
5. Performance Testing & Hyper parameter Tuning
   1. Testing model with multiple rating metrics
   2. Comparing model accuracy before & after applying hyper parameter tuning
6. Model Deployment
   1. Save the best model
   2. Integrate with Web Framework
7. Project Demonstration &Documentation
   1. Record explanation Video for project end to end solution
   2. Project Documentation- Step by step project growth procedure

# Prior Knowledge:

You must have prior knowledge of the following topics to complete this project.

1. **ML Concepts**
   1. Supervised learning: <https://www.javatpoint.com/supervised-machine-learning>
   2. Unsupervised learning: <https://www.javatpoint.com/unsupervised-machine-learning>
2. **Decision tree:** <https://www.javatpoint.com/machine-learning-decision-tree-classification-algorithm>
3. **Random forest:** <https://www.javatpoint.com/machine-learning-random-forest-algorithm>
4. **KNN**: <https://www.javatpoint.com/k-nearest-neighbor-algorithm-for-machine-learning>
5. **Xgboost:** [https://www.analyticsvidhya.com/blog/2018/09/an-end-to-end-guide-to-](https://www.analyticsvidhya.com/blog/2018/09/an-end-to-end-guide-to-understand-the-math-behind-xgboost/) [understand-the-math-behind-xgboost/](https://www.analyticsvidhya.com/blog/2018/09/an-end-to-end-guide-to-understand-the-math-behind-xgboost/)
6. **Rating metrics**: [https://www.analyticsvidhya.com/blog/2019/08/11-important-model-](https://www.analyticsvidhya.com/blog/2019/08/11-important-model-evaluation-error-metrics/) [rating-error-metrics/](https://www.analyticsvidhya.com/blog/2019/08/11-important-model-evaluation-error-metrics/)
7. **Flask Basics** : <https://www.youtube.com/watch?v=lj4I_CvBnt0>

# Project Structure:

# 

Create the Project folder, which contains files as shown below

1. Were building a flask coating which needs HTML pages stored in the templates folder and a python script app.py for scripting.
2. CLC.pkl is our saved model. Further, we will use this model for flask integration.
3. Data Folder contains the Dataset used
4. The Notebook file contains the procedure for building the model.

# toneless 1 : Define Problem/ Problem discernment

## Action 1: Specify the business problem

Refer to Project Description

## Action 2: Business demand

## In terms of business demand, Early Prediction of Chronic Kidney Disease (CKD) aims to develop a machine learning-based diagnostic tool that will aid clinicians with early detection and risk prediction for CKD in patients. The model needs to take into account clinical bio markers (CBs), balancing the data grams (DGs), and performing the prediction in real-time to help the clinician's gain insights. Also, it needs to be scalable, inexpensive, and work within the existing medical applied science substructure utilized in hospitals and clinics to be usable, and still enable the clinician to make informed decisions.

## 

## Action 3: Literature Survey(Student Will Write)

## Many research papers are investigating machine learning-based early prediction of Chronic Kidney Disease (CKD) as a means of addressing the misdiagnosis and ultimately improving the patient journey. This research has shown to use both CNN model and CatBoost model with large data and sequence prediction. It has shown that techniques like SMOTE, deal with class imbalance as well as models that have a chance of not throwing away potential good model performance. This research is also discussing the feature selection algorithms (i.e., RFE and Tabor) to make it more abstract and interoperable. This will also enhance the transparency of the medical decision-making process with some adaptation of the explainable AI (XAI) philosophy. This level of incorporation or adoption will lead to advances in personalized health systems, clinical care, and early appointment programs for CKD patients..

## 

## Action 4: Social or Business Impact.

**Social impact:** When CKD is predicted , the public health will improve, decrease hospital care, and increase quality of life. in addition, it is working towards preventive health and equity in health equity by making the diagnostic more accessible.

**Business impact**: CKD diagnosing will save in treatment costs, resource allotment, and medical efficiency with ML-aided diagnostics. in addition, for medicine research, studying drug based on the patient with personalized medicines.

.

# Milestone 2: Data Collection & Preparation

ML depends heavily-on data. It is the most crucial-aspect that makes-algorithm training possible. So, this section allows you to download the required dataset.

## Action 1: Collect the dataset

There are many well known open sources available to collect the data. Eg: kaggle.com, UCI deposit, etc.

In this project-we worked with .csv data. This dataset was downloaded from kaggle.com. Please-refer to the link below to download the dataset.

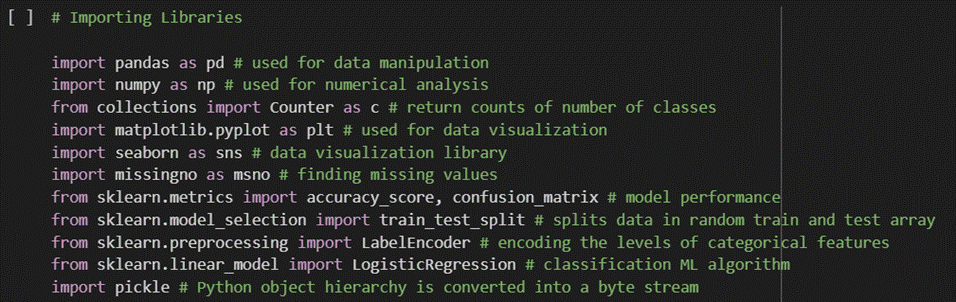
Link:

Now that the datasets are downloaded. Lets look at and comprehend the data as thoroughly as possible, utilizing some visual image techniques and some study analyzing.

**Note:** There are many ways to be able to analyze the data. However, here we do have used some of it, and additionally, you can use multiple ways.

## Action 1.1: Importing the libraries

Import the essential libraries as shown in the image. (optional) Here, we have used the visual image style as featherweight.

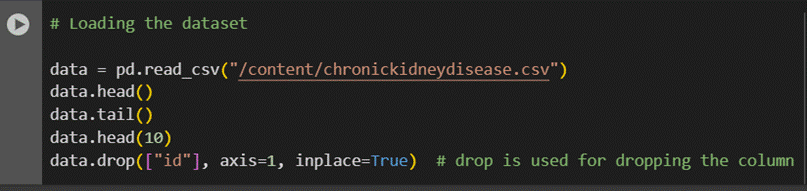


## Action 1.2: Read the Dataset

Our dataset format might be in .csv, excel files, .txt, .json, etc. We can read the dataset with the help of pandas.

In pandas we have a function calledread\_csv() to read the dataset.As a parameter we have to give the directory of the csv file.

* For checking the null values, df.isna().any( ) function is used. To sum those null values we use .sum() function. From the below-image, we found that there are no null values present in our dataset. So we can skip handling the missing values step.



## 

## Action 2: Data Preparation

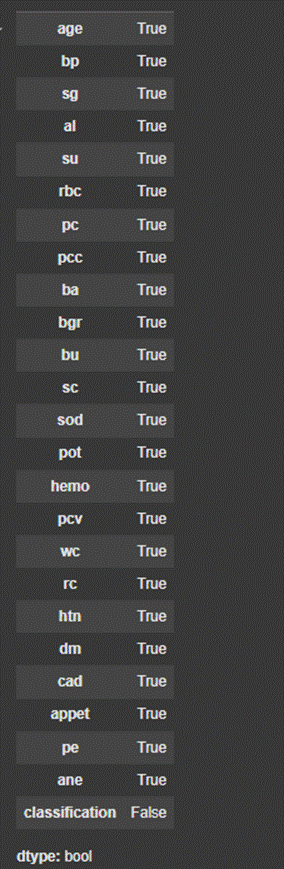
As we have understood how the data is, let's pre-process the collected data.

The download data set is not suitable for training the machine learning model as it might have so much randomness, so we need to clean the dataset properly in order to fetch good results. This action includes the following steps.

* 1. Handling missing values
  2. Handling Outliers

Note: These are the general-steps of pre-processing the data before-using it for machine learning. Depending on the condition of your dataset, you may or may not have to go through all these steps.

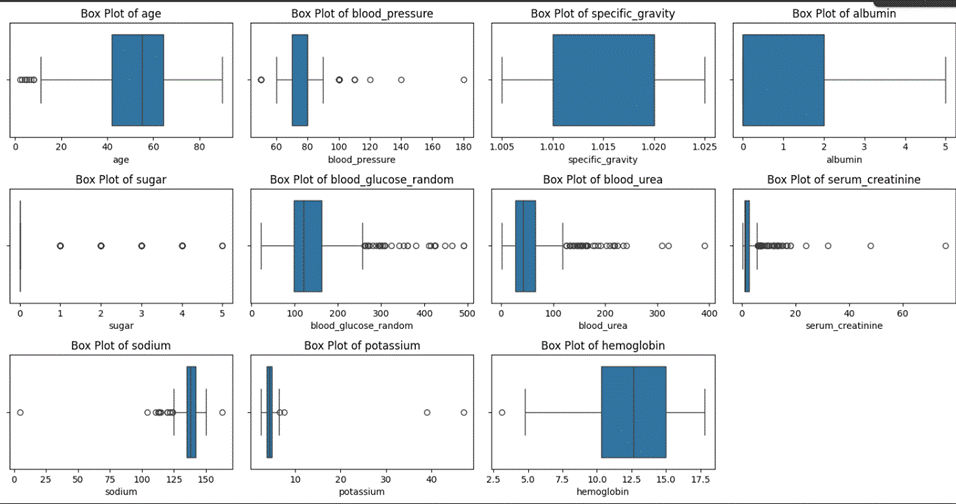
## Action 2.1: Handling missing values

* For checking the null values, df.isna().any( ) function is used. To sum those-null values we use .sum() function. From the below image, we found that there are no null values present in our dataset. So we can skip handling the missing values step.
  + 1. 

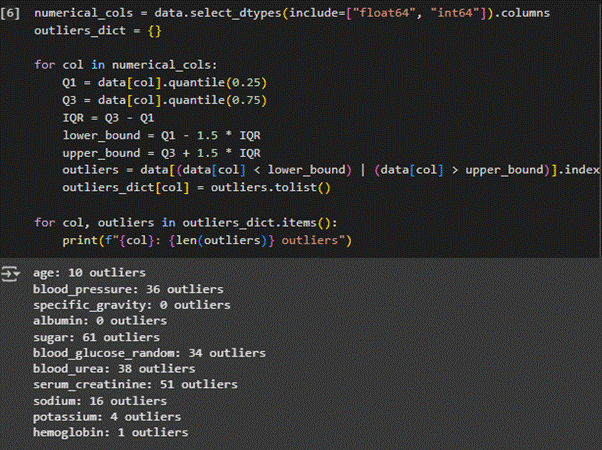
## Action 2.2: Handling Outliers

With the help of a box-plot,outliers are envisioned. And here we are going-to find upper bound and lower bound of policy\_annual\_premium feature with some science formula.

From the below diagram,we could envision that policy\_annual\_premium feature has outliers. Box plot from the sea born library is used here.



We can see the existence of outliers in each feature by analyzing the box plots of the dataset's different features in the above figure. Box plots show the data's median, quartile, and possible outliers. Outliers, or values that deviate from the expected range, are delineated by the tiny circles outside the whiskers.  
It is evident that there are notable outliers in a number of features, including blood\_glucose\_random, blood\_urea, serum\_creatinine, and K. There are fewer or no outliers and a more uniform dispersion of features like specific\_gravity and albumin.  
Outliers can impact model performance, so it is critical to place them. One popular technique for dealing with outliers is to compute the IQR (Interquartile Range) as follows:  
Q3 + 1.5 × IQR is the upper bound; Q1 - 1.5 × IQR is the lower bound.  
This aids in locating and controlling extreme values in attributes such as policy\_annual\_premium.



This code uses the IQR (Interquartile Range) method to determine the number of outliers in each numeric column.

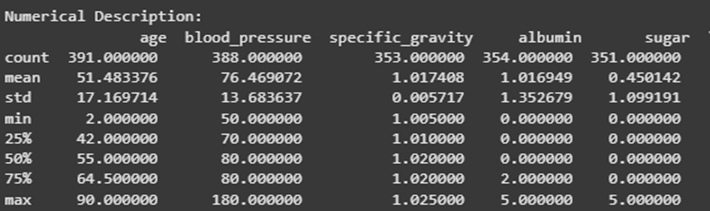
**Actions taken:**  
All of the dataset's numeric columns are first chosen.  
For every column, we compute:  
Q1 (the 25th percentile)  
Q3 (the 75th percentile)  
IQR is equal to Q3 minus Q1.  
Next, we use the following to calculate the lower and upper bounds:  
Q1 - 1.5 × IQR is the lower bound.  
Q3 + 1.5 × IQR is the upper bound.  
Outliers are data points that fall outside of these ranges.  
The number of outliers for each column is printed, and the results are saved in a dictionary.  
The output shows that while specific\_gravity and albumin have no outliers, features like sugar and serum\_creatinine do.

# Milestone 3: Exploratory Data Analysis

# 

## Action 1: Descriptive statistical

Descriptive analysis is to study the basic features of data with the statistical process. Here, pandas have a worthy function called describe. With this describe function, we can understand the unique, top, and frequent values of collection features. And we can find indicate, standard, minimum, maximum, and percentile values of continuous features.



## Action 2: Visual analysis

Visual analysis is the process of using visual legacy, such as charts, plots, and graphs, to explore and understand data. It is a way to place patterns, trends, and outliers in the data, which can help to gain insights and make informed decisions.

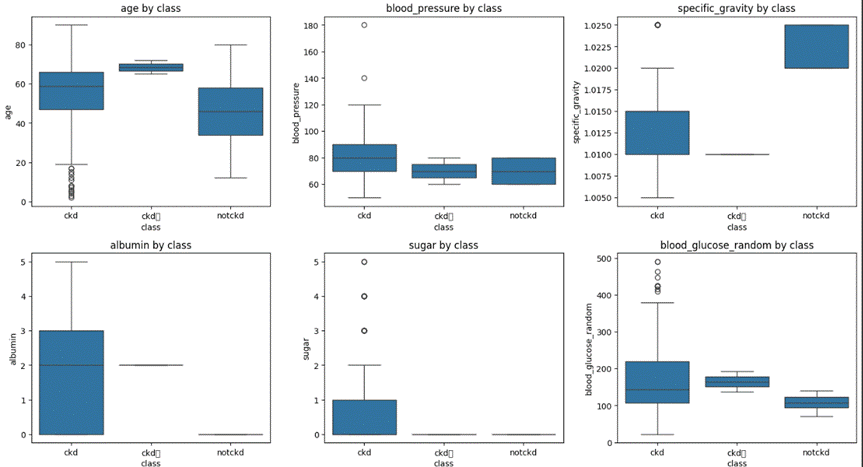
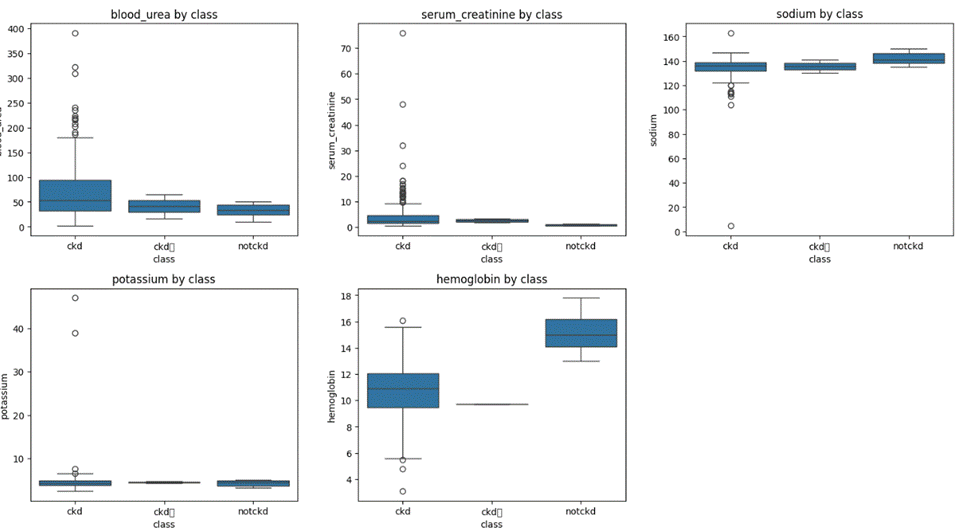
## Action 2.1: Uni variate analysis

## 

Each numeric feature's dispersion is displayed in the above histograms (Uni-variate Analysis). This aids in crucial whether the data is skewed, normally distributed, or contains outliers.  
For instance:  
The dispersion of hemoglobin and age is fairly normal.  
Potential outliers are bespeak by the right-skewed, long-tail blood urea, serum production, and blood glucose random data.  
Specific gravity and albumin, for example, have discrete values (low fluctuation).  
Choosing data shift and preprocessing procedures for additional modeling is made easier by this analysis.

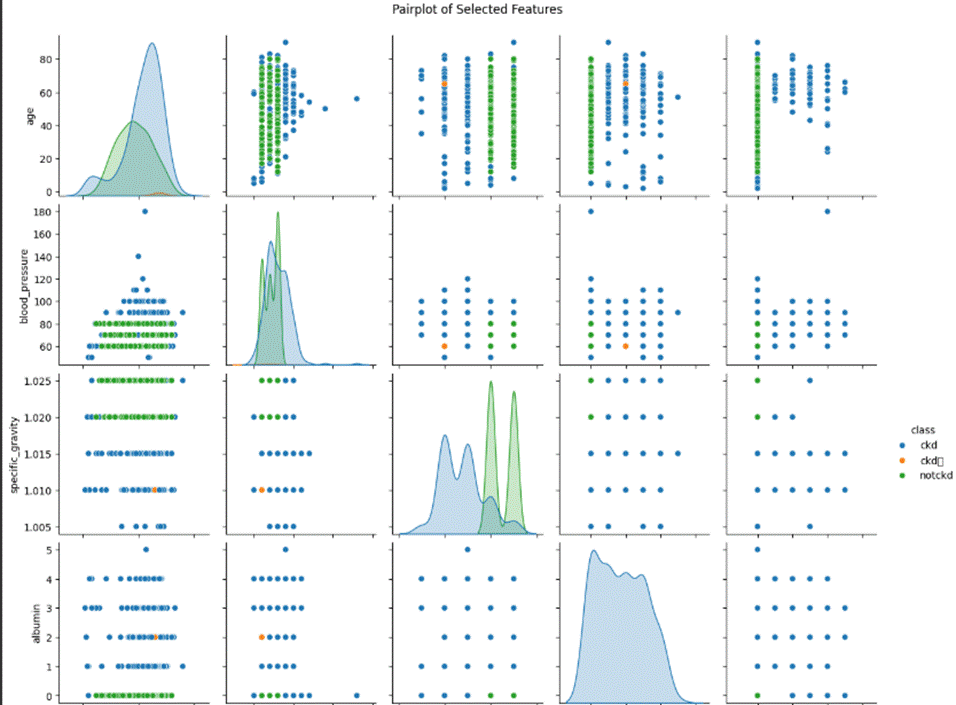
The dispersion of collection features is displayed in the bar plots above. This aids in our comprehension of the frequency with which each class occurs within the dataset.  
For instance:  
The Yes/No dispersion for diabetes mellitus and hypertension is fairly balanced.  
In the dataset, structure artery disease is uncommon.  
The dispersion of appetite and pedal edema are unbalanced, with more "yes" answers.  
The red blood cell count exhibits a broad range of values with no evident trend.  
Finding class imbalance and directing preprocessing procedures like encoding or oversampling are two benefits of this analysis.

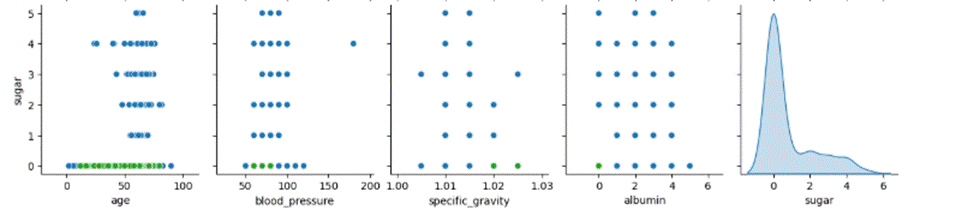
## Action2.2: Variate analysis



The divergence in numeric features between CKD and non-CKD classes are displayed in box plots. It is evident that features such as age, serum-producing Heb, and blood urea vary greatly amongst the classes. This aids in locating crucial feature for CKD prediction.

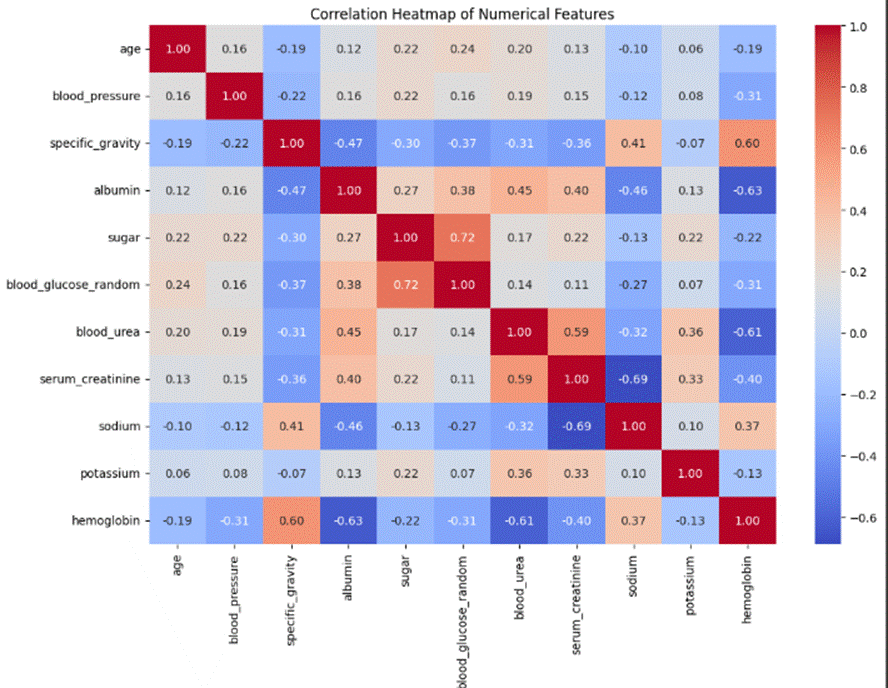
**Action 2.3: Analysis of multiple varying**





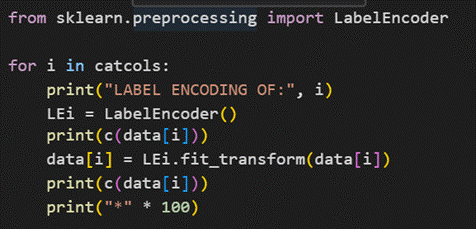
Through the display of scatter plots and dispersion of specific features across CKD and non-CKD classes, the pair plot above carries out multivariate analysis. This enables us to observe how various feature pairs relate to one another and how, in effect, they can distinguish between classes.

combining such as age versus blood pressure or specific gravity versus albumin, for instance, clearly distinguish patients with CKD from those without it. Selecting crucial feature interactions that can enhance model performance is made easier with the aid of such insights.



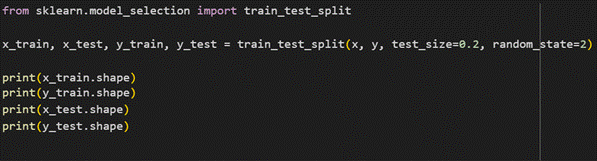
This correlation heat map illustrates the kinship between numerical features. The color scale goes from -1 (strong negative correlation, blue) to +1 (strong positive correlation, red).  
We can observe:  
Blood\_glucose\_random and sugar have a strong positive correlation (0.72).  
Hemoglobin has a negative kinship with serum creation, albumin, and blood urea.  
Many other features have weak correlations with one another.  
We can choose the most pertinent features for modeling, avoid multidisciplinary approaches, and comprehend the kinship between features with the aid of this analysis.

**Encoding the Categorical Features:**

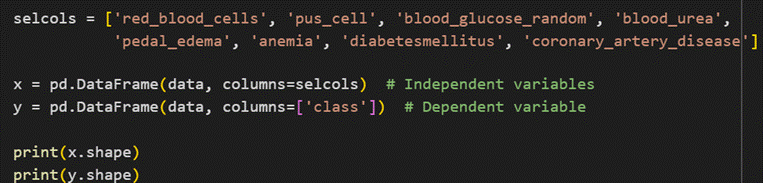


**Splitting data into train and test**

.

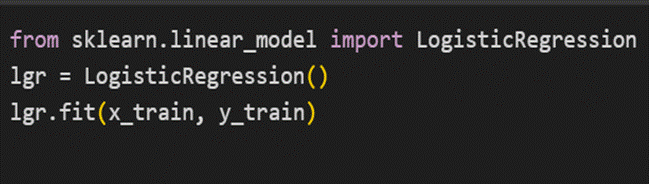


#### Handling Imbalanced dataset

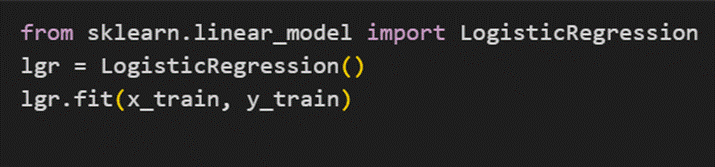
.

#### Scaling

#### 



# Milestone 4: Model Building



The code demonstrates how to initialize and train a Logistic Regression model using the learn library, which is a well-known library for machine learning in Python. It starts with importing LogisticRegression from sklearn.linear\_model. Then, it creates an instance of the LogisticRegression model as mgr. Finally, it fits the model to the training data (x\_train and y\_train). This is the step where the model learns the relationships between the feature and the target so that it can make predictions on new, unseen data.

# Milestone 5: Performance Testing & Hyper parameter Tuning

## Action 1: Testing model with multiple rating metrics

## Action 2: Comparing model accuracy before & after applying hyper-parameter tuning (Hyperparameter tuning is optional.

## 

## 

## 

## 

# Milestone 6: Model Deployment

## action 1: Save the best model

## 

## 

## Action 2: Integrate with Web Framework

In this section, we will be building a web coating that is merged to the model we built. A UI is provided for the uses where he has to enter the values for predictions. The enter values are given to the saved model and prediction is showcased on the UI.

This section has the following tasks

1. Building HTML Pages
2. Building server-side script
3. Run the web coating

## Action 2.1: Building Html Page:

For this project, create an HTML file

### index.html

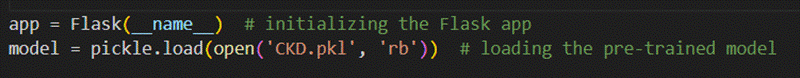
and save them in the templates folder. Refer to this link for templates.

## Action 2.2: Build Pythoncode:

Import the libraries

|  |  |
| --- | --- |
|  |  |
|  |  |

Load the saved model. Importing the flask module in the project is required. An object of Flask class is our WSGI coating. Flask constructor takes the name of the current module (name) as argument.



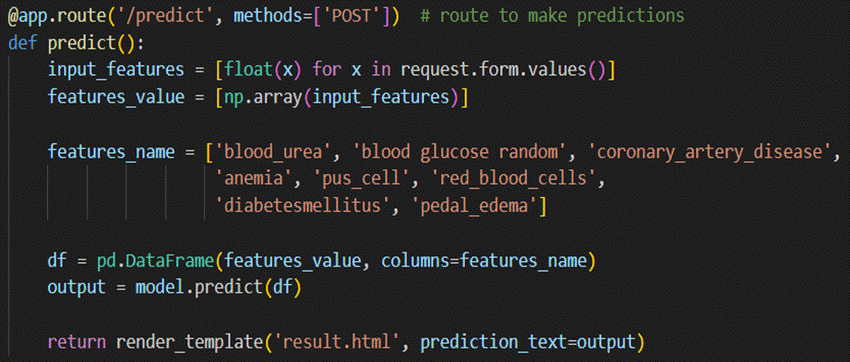
Render HTML page:

|  |  |
| --- | --- |
|  |  |
|  |  |

Here, we will be using a declared constructor to route to the HTML page which we have created earlier.

In the above example, ‘/’ URL is bound with the index.html function. Hence, when the home page of the web server is opened in the browser,the html page will be rendered. Whenever you enter the values from the html page, the values can be retrieved using the POST Method.

**Retrieves the value from UI:**



Here, we are routing our app to the predict() function. This function retrieves all the values from the HTML page using Post request. That is stored in an array. This array is passed to the model.predict() function. This function returns the prediction. And this prediction value will be rendered to the text that we have mentioned in the submit.html page earlier.

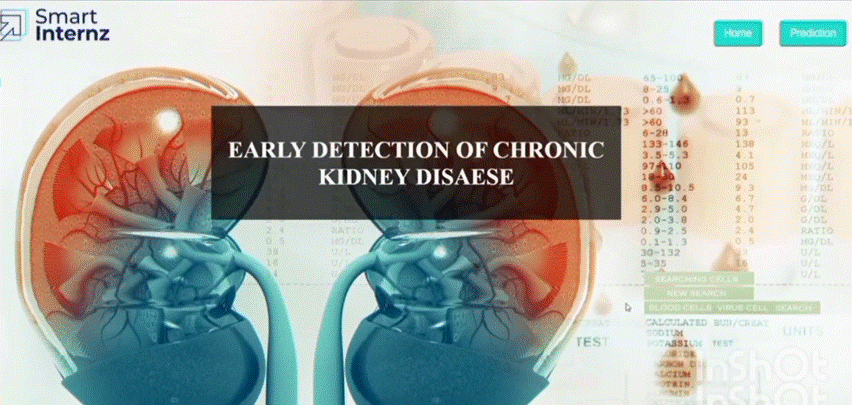
**Main Function:**

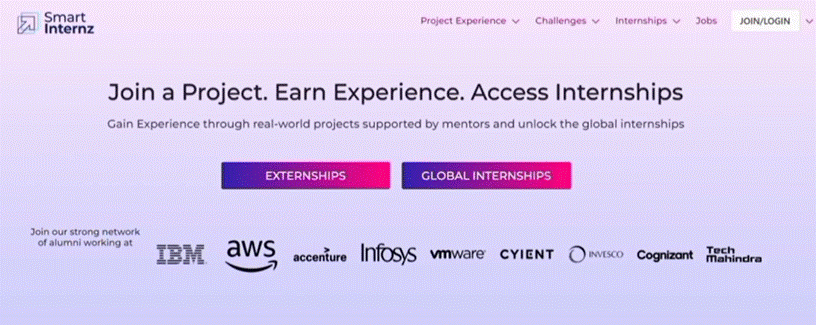
|  |  |
| --- | --- |
|  |  |
|  |  |

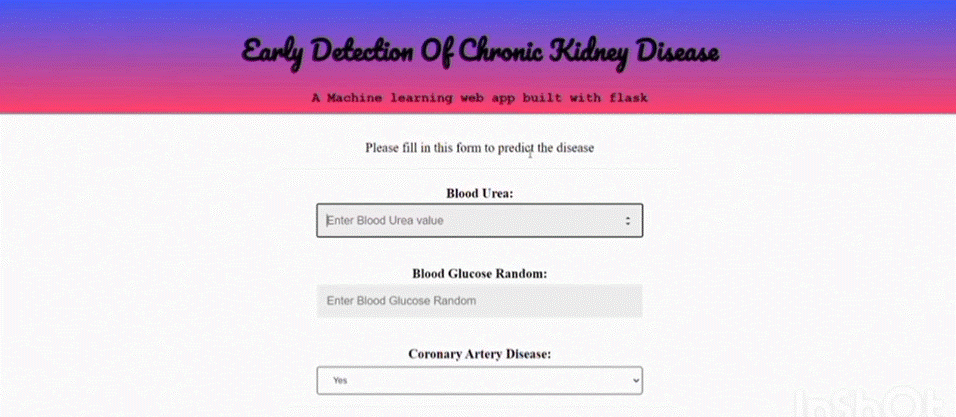
## Action 2.3: Run the web coating

* 1. Open anaconda prompt from the start menu
  2. Navigate to the folder where your python script is.
  3. Now type “pythonapp.py” command
  4. Navigate to the localhost where you can view your web page.
  5. Click on the predict button from the top left corner, enter the inputs,click on the submit button, and see the result/prediction on the web.

**Now,Go the web browser and write the localhost url** [**(http://127.0.0.1:5000)**](http://127.0.0.1:5000/) **to get the below result**







**Milestone 7: Project Demonstration & Documentation**

Below mentioned transportation to be submitted along with other deliverables

## Activity 1:- Record explanation Video for project end to end solution

## Link :https://go.screenpal.com/watch/cT1YirnXCnL

**Activity 2:- Project Documentation-Step by step project development procedure**

Create document as per the template provided