IBM-NALAIYA THIRAN

Early Detection of Chronic Kidney Disease using Machine Learning

SUBMITTED BY:

1)NISHALINI C

2)KAVIYA P

3)RAMYA G

4)MUTHAMIZH SELVAN M

TEAM ID-PNT2022TMID07996

Department of Computer Science and Engineering,
Adhiyamaan College of Engineering-635109.

INSTITUTION MENTOR: N.MORATANCH

INDUSTRIAL MENTOR: LALITHA GAYATHRI

TABLE OF CONTENTS

1.INTRODUCTION

Chronic kidney disease is a slow and progressive loss of kidney function in several years. CKD affects about 37 million people which are approximately 43%of the population. It often goes undetected and undiagnosed until the condition is advanced. As CKD advances, more dangerous levels of waste can rapidly build up inside the body. Treatment aims to slow the progression of kidney dysfunction by controlling the underlying cause. CKD is a condition in which the kidneys are damaged and cannot filter blood and their functionalities. Due to this, excess fluid and waste from blood remain in the body. It may cause other health problems such as heart disease and stroke, etc... CKD can progress to kidney failure and early cardiovascular disease, if left untreated. When the kidneys stop working its process, dialysis or kidney transplant is needed for survival. Not all patients with kidney disease progress to CKD. So, to help prevent CKD and lower the risk for kidney failure, control risk factors for CKD, get tested early, and make lifestyle changes.

To identify and predict chronic kidney disease using a machine learning techniques using data set contains comprises of both the structured data and unstructured data. In this, data preprocessing can done for finding the missing values and recreate to increase the quality of the model. Hence, the prediction accuracy increased. For prediction of CKD using Machine learning algorithms such as Random Forest (RF), K-nearest neighbor (KNN), Decision Tree Algorithm, Logistic Regression. It is believed that the proposed system can reduce the risk of chronic diseases by recognize them prior and reduces the cost for diagnosis, treatment, and doctor consultation.

The structure of the paper is as sections: 1) Introduction 2) Related Works 3) Proposed methodology 4) System Architecture 5) Data Preprocessing 6) Build the Model 7) Performance Evaluation 8) Comparison 9)Conclusion 10) Reference.

1.1.Project Overview

Chronic kidney disease is one of the most widespread diseases among humans. It is

identified and predicted at the earliest stages, so as to prevent extremity of Chronic Kidney Disease. The early CKD prediction can be effective solution to reduce the harm using machine learning. It achieved by using a cutting-edge machine learning technique to ensure that reliably identifies persons with chronic diseases. A collection of disease symptoms along with the person's living habits and doctor consultations are taken for preparing the dataset.The proposed system offers the prediction of CKD using Machine learning algorithms such as Random Forest (RF), K-nearest neighbor (KNN), Decision Tree Algorithm, Logistic Regression. As a result, these techniques aid experts and doctors in making early diagnoses in order to avoid kidney failure.

Keywords: Chronic kidney disease, Random Forest (RF), K-nearest neighbor (KNN), Logistic Regression, Decision Tree.

1.2. PURPOSE

Early detection of CKD should be beneficial because **it enables clinicians to initiate effective treatment of mild disease, preventing loss of kidney function and delaying or avoiding progression to kidney failure**.Early identification of CKD by screening for kidney disease, followed by risk stratification and treatment, **offers the potential to substantially reduce the morbidity and mortality from CKD and its related complications, such as cardiovascular disease**.The rationale for testing asymptomatic people for CKD is that **earlier detection might allow for the implementation of therapeutic interventions and avoidance of inappropriate exposure to nephritic agents**, both of which may slow the progression of CKD to end-stage kidney disease

2. LITERATURE SURVEY

Prediction of Chronic Kidney Disease is one of the most important problem that

has to be detected in the early phases of the commencement of the disease so as to reduce the disease progression rate among the individuals.Various researches have been made to find the basic cause and some have reached to the heights by proposing a system which differentiates the healthy people from those with any disorder's patients using various machine learning techniques. Lots of preprocessing, feature selection and classification techniques have been implemented and developed in the past decades. Following is the given work done in the prediction of Chronic Kidney Disease. We have categories the review based upon Pre-processing techniques and classification methods.

2.1 EXISTING PROBLEM

Chronic kidney disease mainly occurs due to high blood pressure and diabetes. CKD can be easily identified by the researchers based upon two parameters –Glomerular Filtration Rate (GFR) and [1] kidney damage markers. In a Deep Neural Network Techniques, the averages of the associated features need to be replacing all missing values in the database. Recursive Feature Elimination (RFE) is used to identify the important features for prediction. This technique mainly uses four classifiers and used to estimate the comparative analysis.

Creatinine, which is a type of metabolite of blood that is strongly correlated to Glomerular Filtration Rate (GFR). Calculating GFR value is difficult, so creatinine value is directly taken instead of calculating GFR.[2] The medical community accepts a GFR of 60 ml/min is used as the threshold, below which is considered to have CKD. The first step is a regression model which predicts the value of creatinine from 23 attributes, and then combine the predicted value of creatinine with the original 23 attributes to detect the risk of CKD. To improve the results of creatinine predictor, the average results from overall predictors and ensemble the results.

Chronic Kidney Disease is a disorder that affects normal kidney function and exploring preventive measures for CKD through early diagnosis using machine learning techniques. [3] The mean and mode statistical analysis methods were used to replace the missing numerical values and the nominal values, this involves in the process of data pre-processing techniques. The most important attributes can be found by applying Recursive Feature Elimination (RFE) technique.CKD is a

serious life-threatening disease, with high rates of morbidity and mortality and this can be very helpful in the early detection of CKD.

Chronic kidney disease is one of most widespread diseases among humans. It is identified and predicted at the earliest stages, so as to prevent extremity of it. Identification of chronic kidney disease with human is achieved by cutting-edge machine learning techniques. [4] A collection of disease symptoms along with the person's living habits, and doctor consultations are taken for preparing the dataset. It is highly believed proposed system can reduce the risk of chronic diseases by diagnosing them earlier and also reduces the cost for diagnosis, treatment, and doctor consultation.

Chronic kidney disease is a kind of nephritic syndrome in which the kidney capacity to cope steadily deteriorates and remains asymptomatic for a long period as the disease progresses. Machine Learning is a technology helps physicians in the accurate diagnosis of kidney disease[5] and helps in effective treatment prediction by recommending nutrition. A correlation analysis is used to observe the effective finding for renal malfunctioning and identifying the best food products. The result says that the ML algorithm detects the risk of CKD occurrence accurately.

2.2 REFERENCE

1.Singh. Asari ,V. K. Rajasekaran, R. "A Deep Neural Network for Early Detection and Prediction of Chronic Kidney Disease." Diagnostics 2022, 12, 116.

2.Wang, W.; Chakraborty, G.; Chakraborty, B." Predicting the Risk of Chronic Kidney Disease (CKD) using Machine Learning Algorithm." Appl. Sci. **2021**, 11, 202,

3.Hindawi Journal of Healthcare Engineering Volume 2021, Article ID 1004767, 10 pages, https://doi.org/10.1155/2021/1004767

4.Hindawi Journal of Healthcare Engineering Volume 2022, Article ID 2826127, 9

pages https://doi.org/10.1155/2022/2826127

5. Silveira, A.C.M.d.; Sobrinho, Á.; Silva, L.D.d.; Costa, E.d.B.; Pinheiro, M.E.; Perkusich, A. "Exploring Early Prediction of Chronic Kidney Disease Using Machine Learning Algorithms for Small and Imbalanced Datasets." Appl. Sci. **2022**, 12, 3673. https://doi.org/10.3390/app12073673

2.3 PROBLEM STATEMENT DEFINITION

Chronic Kidney Disease (CKD) is a major medical problem and can be cured if treated in the early stages. Usually, people are not aware that medical tests we take for different purposes could contain valuable information concerning kidney diseases. Consequently, attributes of various medical tests are investigated to distinguish which attributes may contain helpful information about the disease. The information says that it helps us to measure the severity of the problem and we make use of such information to build a machine learning model that predicts Chronic Kidney Disease .

In the following Customer Problem Statement Template we created a problem statement to understand our customer's point of view. The Customer Problem Statement template helps us to focus on what matters to create experiences people will love.A well-articulated customer problem statement allows our team to find the ideal solution for the challenges our customers face. Throughout the process, we also empathize with our customers, which helps us to better understand how users perceive our product or service.

Problem Statement (PS)	Iam (Customer)	I'm trying to	But	Because	Which makes me feel
PS-1	Patient	to know whether I am having disease or not.	can't able to identify the symptoms	Symptoms will occur at last stage	Unaware
PS-2	Patient	find the chances of getting the disease	Not able to spent lot of time and amount to predict the chances of getting disease everytime.	I can't able to go often to hospitals for diagnosing the disease	fear about the life
PS-3	Patient	Find the preventive measures and treatment provided to the affected patients	Not able to reach out the doctors every time for consulting	Less awareness about the disease	Stressed

3. IDEATION&PROPOSED SOLUTION

3.1 EMPATHY MAP CANVAS

Empathy Map Canvas:

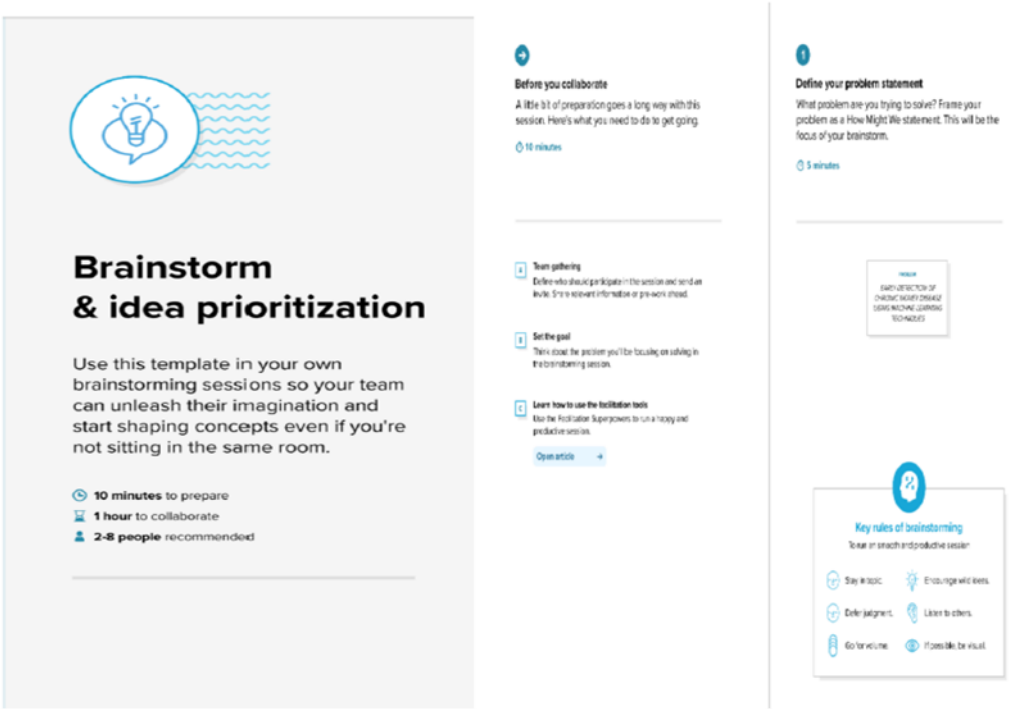
An empathy map is a simple, easy-to-digest visual that captures knowledge about a user's behaviours and attitudes.

It is a useful tool to helps teams better understand their users.
Creating an effective solution requires understanding the true problem and the person who is experiencing it. The exercise of creating the map helps participants consider things from the user's perspective along with his or her goals and challenges.

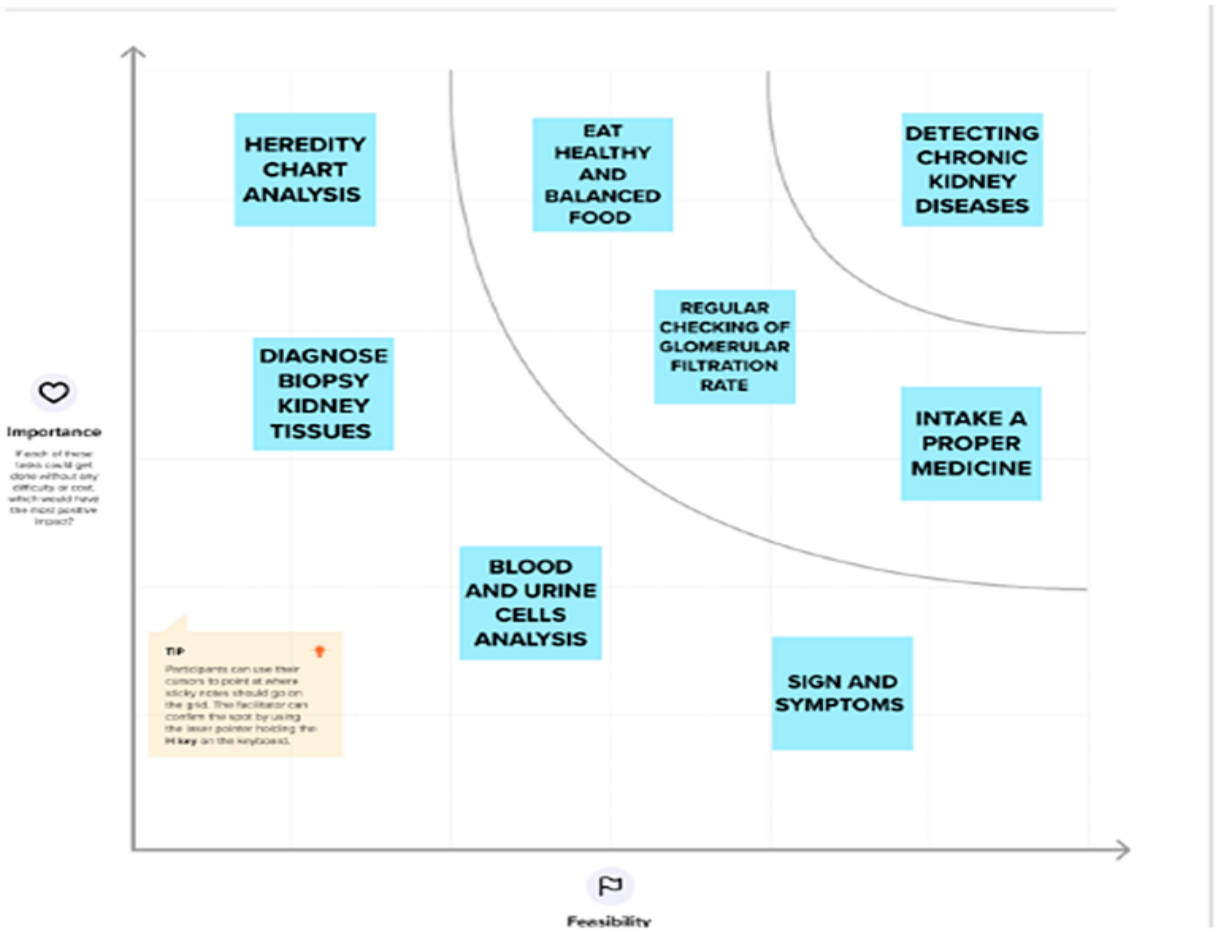
3.2 IDEATION &BRAINSTORMING

Brainstorming combines a relaxed, informal approach to problem solving with lateral thinking. It encourages people to come up with thoughts and ideas that can, at first, seem a bit crazy. Some of these ideas can be crafted into original, creative solutions to a problem, while others can spark even more ideas. This helps to get people unstuck by "jolting" them out of their normal ways of thinking.

Step-1: Team Gathering, Collaboration and Select the Problem Statement



Step-2: Brainstorm, Idea Listing and Grouping



3.3 PROPOSED SOLUTION

- A Web Application is developed to predict the chronic kidney disease at the earlier stage based on the occurrence of symptoms that the particular patients gets infected by chronic kidney disease.
- Suggestion of Prevention measures for the users who has a less chances of getting disease.
- Suggestion of Treatment for the users who has a high chances of getting disease.
- Suggestion of regular healthy diet routine information for the users who are healthy and not having chances of getting disease.

ADVANTAGES:

• Best Suitable treatment Recommendation

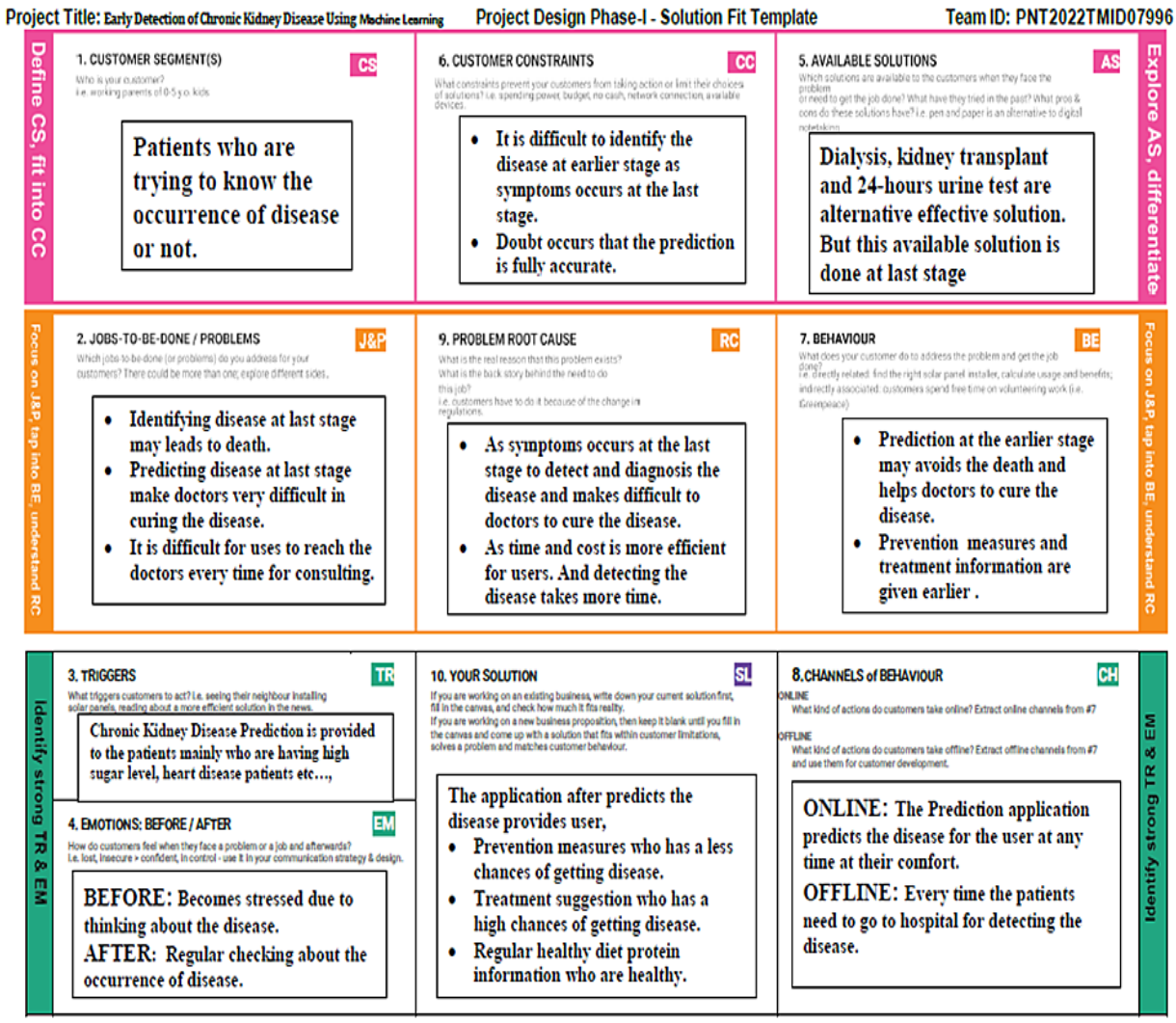
• Preventing Measures recommendation

• Accurate disease prediction

S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	Chronic Kidney Disease is a disorder that affects normal kidney function and exploring preventive measures for CKD is to detect the disease at early stages.
2.	Idea / Solution description	Information concerning with kidney diseases may helps us to measure the severity of the problem using machine learning model that detects the chronic kidney disease at early stages.
3.	Novelty / Uniqueness	➢ Suggestion of Prevention measures for the users, who has a less chances of getting disease. ➢ Suggestion of Treatment for the users who has a high chances of getting disease. ➢ Suggestion of regular healthy diet routine information for the users who are healthy and not having chances of getting disease.
4.	Social Impact / Customer Satisfaction	Detecting and curing of the disease at early stages lead to reduce the death rate.
5.	Business Model (Revenue Model)	Now a days, a disease is growing faster over a years, this project will help us to detecting the chronic kidney diseases at early stages which will have a great impact on medical science.
6.	Scalability of the Solution	More people can able to get their normal life back.

3.4 PROPOSED SOLUTION FIT

The Problem-Solution Fit simply means that we have found a problem with your customer and that the solution you have realized for it actually solves the customer's problem.It is important to have a good picture of your customer, not only the demographics but preferably also socio graphic data.The more informed the customer is about their problem, the better. It may even be the case that your customer has realized an inefficient homegrown solution to try to solve the problem.We tried as hard as you can to identify how well the customer is aware of the problem and when the customer will run into the problem.



4.REQUIREMENT ANALYSIS

Requirements analysis, also called requirements engineering, is the process of determining user expectations for a new or modified product. These features, called requirements, must be quantifiable, relevant and detailed. In software engineering, such requirements are often called functional specifications.

4.1 FUNCTIONAL REQUIREMENT

Functional Requirements:

Following are the functional requirements of the proposed solution.

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Registration	Registration through Gmail
FR-2	User Confirmation	Confirmation via Email
FR-3	User Login	User enters into platform using Email and Password that has been created in the registration
FR-4	Patient's Profile	User's details and already predicted results and the date of prediction will be displayed for the existing users.
FR-5	Input Kidney Disease Parameters manually	User enters the parameters of disease for the occurrences of diseases or not.
FR-6	View Results of Chronic Kidney disease Risk	Then the application shows the users about the chances of getting the disease. • Less chance predicted user's gets more information about the prevention measures. • High chance predicted user's gets more information about the best treatment to cure the disease. • No chance predicted user's gets more information about the regular diet routine methods to keep them healthy.
FR-7	Application gets confirmation about the regular updates	Regularly email will be sent to the user for prediction of disease.

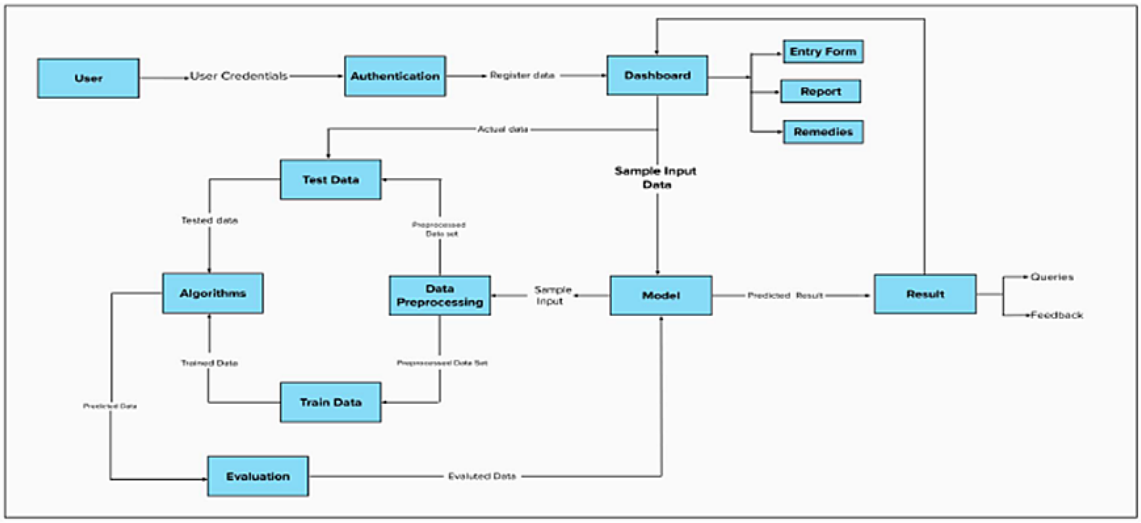
4.2 NON-FUNCTIONAL REQUIREMENT

5.PROJECT DESIGN

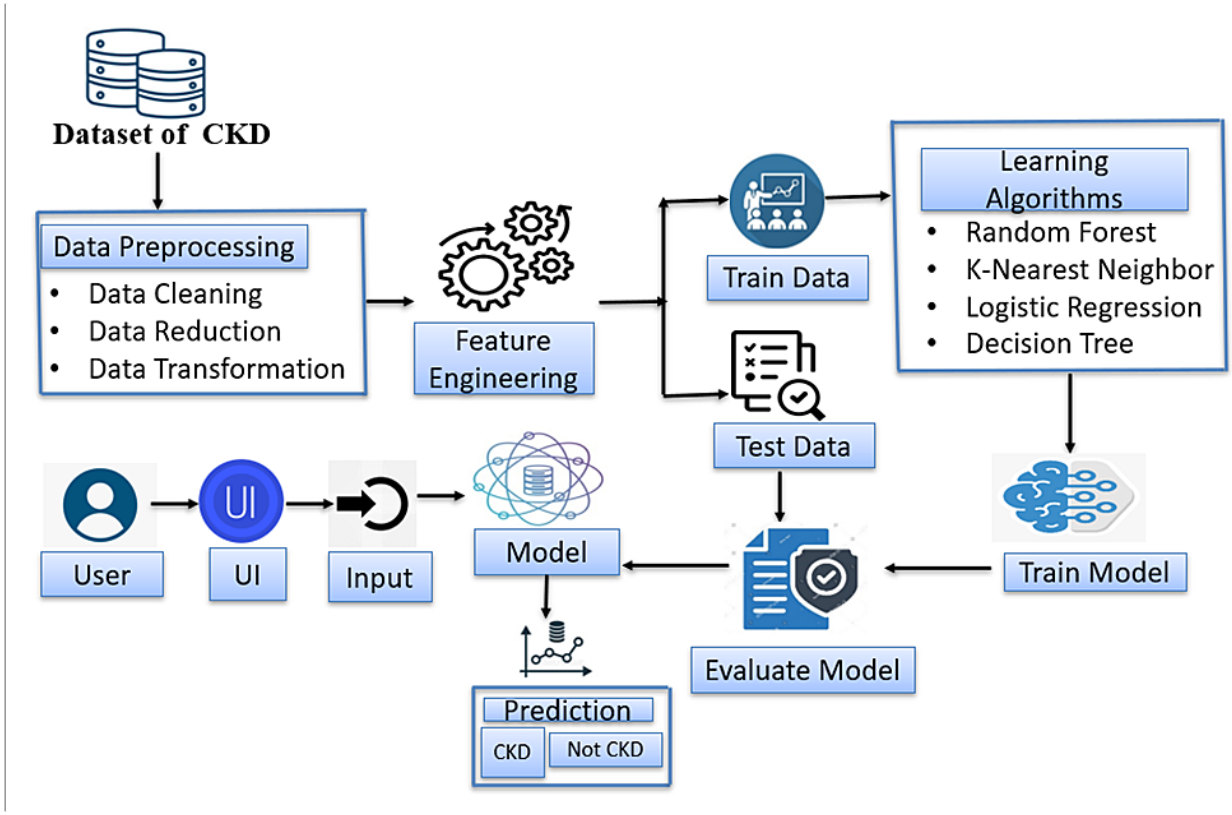
When it comes to managing projects, it can be hard to get everyone on the same page. With multiple moving parts, different deliverables, and cross-departmental collaboration, sometimes an initial project meeting just isn’t enough. Project design is an opportunity to align on ideas, processes, and deliverables. It’s an early phase in the project lifecycle and often comes before a project plan or charter. This is because it focuses on the project overview rather than the specific details.Project design is an early phase of the project lifecycle where ideas, processes, resources, and deliverables are planned out. A project design comes before a project plan as it’s a broad overview whereas a project plan includes more detailed information.

5.1 DATA FLOW DIAGRAM

DATA FLOW DIAGRAM:



5.2 SOLUTION&TECHNICAL ARCHITECTURE



5.3 USER STORIES

A user story is an informal, general explanation of a software feature written from the perspective of the end user or customer. The purpose of a user story is to articulate how a piece of work will deliver a particular value back to the customer.

User Stories

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer (Web user)	Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password.	I can access my account / dashboard.	High	Sprint-1
		USN-2	As a user, I will receive confirmation email once I have registered for the application.	I can receive confirmation email & click confirm/verify through the OTP.	High	Sprint-1
	Login	USN-3	As a user, I can log into the application by entering email & password.	See in a Dashboard.	High	Sprint-1
	Dashboard	USN-4	As a user, I can see my past records and activities.	I can access my past records and can solve my queries through Q/A.	High	Sprint-2
	Entry Form	USN-5	As a user, I must enter my pre-diagnostic test results.	I have to fill the form with my test results.	High	Sprint-2
	Report	USN-6	As a user, I can view the report generated by the tool.	I will be able to view my test results after diagnosis.	High	Sprint-3
Customer Care Executive	Remedies	USN-7	Will be able to give some suggestions to improve my health.	The suggestions are helpful to recover from CKD.	Medium	Sprint-3
	Feedback	USN-8	As a user, I will be filling the feedback form.	I will be reading those feedbacks to improve User Experience.	Low	Sprint-4
	Queries	USN-9	As a customer care executive, I must assist users that face problems through Q&A.	The queries of the customer have to be sorted within a period.	Low	Sprint-4
Administrator	Feature importance	USN-10	As an administrator, I should identify the most significant factors that lead to a CKD based on the present trend.	I must identify important features.	High	Sprint-2
	Train Model	USN-11	As an administrator, I must use the most suitable ML model for detection of CKD.	I should efficiently train the ML model.	High	Sprint-2

6. PROJECT PLANNING & SCHEDULING

Project planning is at the heart of the project life cycle, and tells everyone involved where you're going and how you're going to get there. The planning phase is when the project plans are documented, the project deliverables and requirements are defined, and the project schedule is created. It involves creating a set of plans to help guide your team through the implementation and closure phases of the project. The plans created during this phase will help you manage time, cost, quality, changes, risk, and related issues. They will also help you control staff and external suppliers to ensure that you deliver the project on time, within budget, and within schedule.

6.1 Sprint Planning & Estimation

Product Backlog, Sprint Schedule, and Estimation

The purpose of the project planning phase is to:

- Establish business requirements
- Establish cost, schedule, list of deliverables, and delivery dates
- Establish resources plans

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Collecting the Dataset	USN-1	I need to collect dataset provided by IBM resource	10	High	Team Lead Team Member 1 Team Member 2 Team Member 3
Sprint-1	Data Pre-processing	USN-2	I need to take raw data from dataset and transform it into a format which is more understandable and analysed by machine learning algorithms.	10	High	Team Lead Team Member 1 Team Member 2 Team Member 3

Sprint-2	Building the Model	USN-3	I need to explore the various types of algorithms which can be used to train and test the model for predicting the best accurate output.	12	Medium	Team Lead Team Member 1 Team Member 2 Team Member 3
Sprint-2	Model Evaluation	USN-4	After building the model I need to select the best ML model based on high accuracy for accurate prediction.	18	High	Team Lead Team Member 1 Team Member 2 Team Member 3
Sprint-3	UI Designing	USN-5	As a developer, I need to design a simple and understandable user interactive UI	8	Medium	Team Lead Team Member 1 Team Member 2 Team Member 3
Sprint-3	UI Integration	USN-6	As a developer, I need to integrate the user interface and the trained model using Flask.	14	High	Team Lead Team Member 1 Team Member 2 Team Member 3
Sprint-4	Integration with IBM Cloud	USN-7	Integrating trained IBM Cloud Model with scoring endpoints using flask.	12	Medium	Team Lead Team Member 1 Team Member 2 Team Member 3
Sprint-4	Deployment	USN-8	Deploying the IBM model, backend, flask and frontend application in Cloud	11	High	Team Lead Team Member 1 Team Member 2 Team Member 3
Sprint-4	Further Classification	USN-9	Providing user about the prevention measures ,treatment suggestion and healthy diet routine suggestion.	7	Medium	Team Lead Team Member 1 Team Member 2 Team Member 3

Project Tracker, Velocity & Burndown Chart:

The planning phase refines the project's objectives, which were gathered during the initiation phase. It includes planning the steps necessary to meet those objectives by further identifying the specific activities and resources required to complete the project.

Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint-1	20	6 Days	24 Oct 2022	29 Oct 2022	20	30 Oct 2022
Sprint-2	30	6 Days	31 Oct 2022	05 Nov 2022	30	06 Nov 2022
Sprint-3	22	6 Days	07 Nov 2022	12 Nov 2022	22	13 Nov 2022
Sprint-4	30	6 Days	14 Nov 2022	19 Nov 2022	30	20 Nov 2022

Velocity:

We have a 6-day sprint duration, and the velocity of the team is 20 to 30 (points per sprint). Let's calculate the team's average velocity (AV) per iteration unit (story points per day)

Sprint 1 AV = Sprint duration/velocity = 20/6 = 3.33

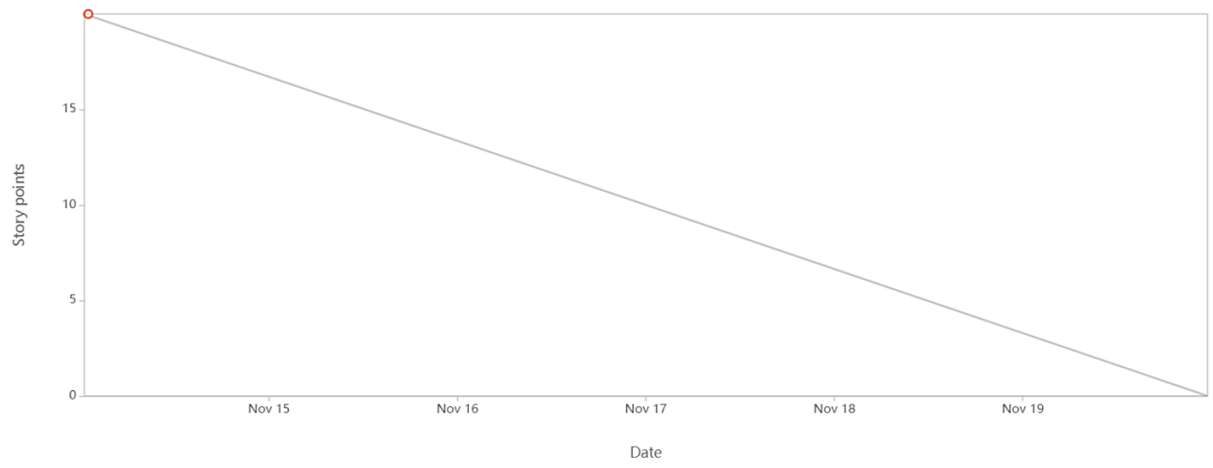
Sprint 2 AV = Sprint duration/velocity = 30/6 = 5

Sprint 3 AV = Sprint duration/velocity = 22/6 = 3.66

Sprint 4 AV = Sprint duration/velocity = 30/6 = 5

Total Average Velocity=4.24

Burndown Chart:



6.2 Sprint Delivery Schedule

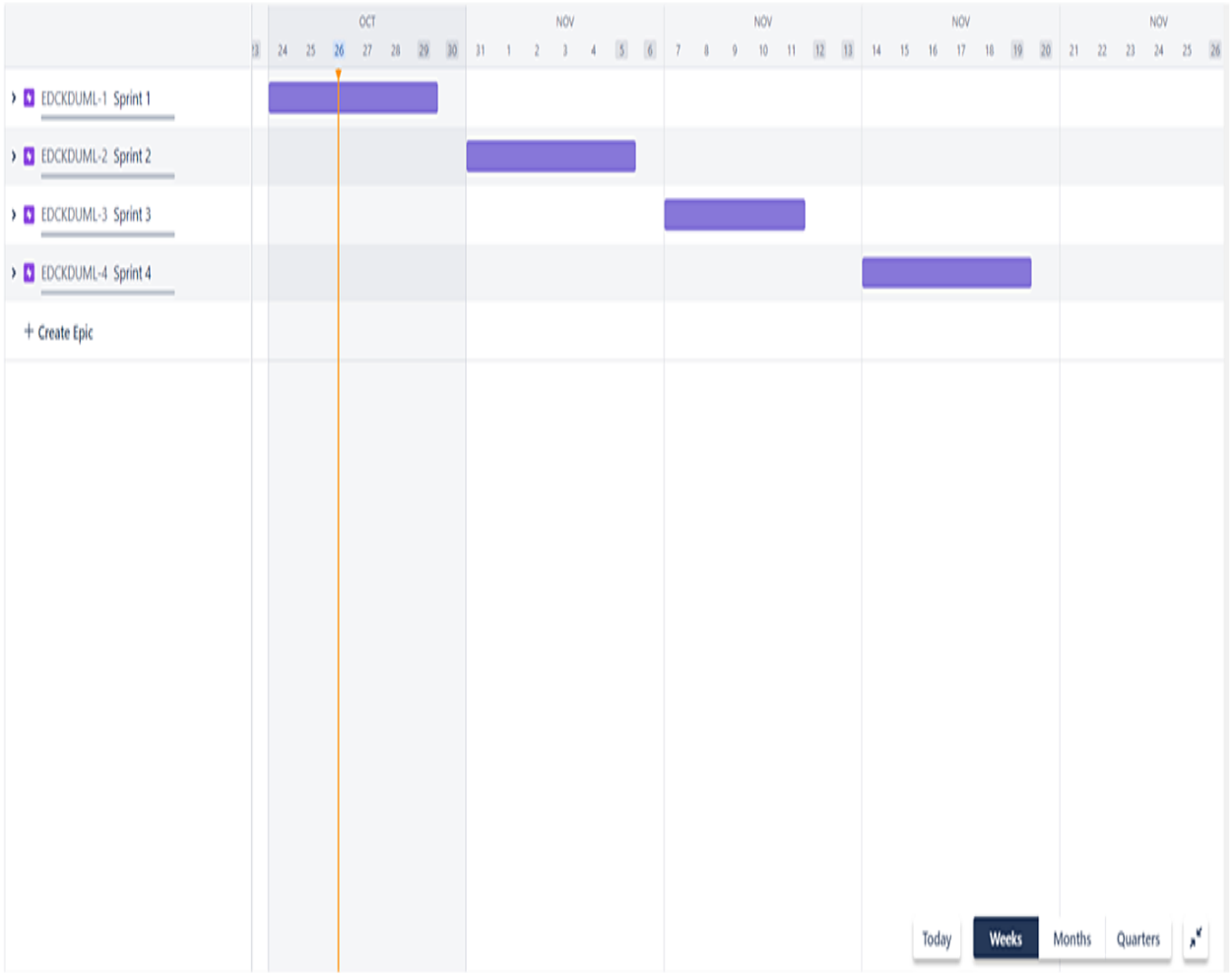
TITLE	DESCRIPTION	DATE
Literature survey & information gathering	Collect the relevant use cases and refer to existing solutions.	19 SEPTEMBER 2022
Prepare empathy map	Prepare Empathy Map canvas and list of problem statements	19 SEPTEMBER 2022
Ideation	List the ideas by organizing the brainstorming sessionand prioritize the top 3 ideas based on the feasibility & importance	19 SEPTEMBER 2022
Problem solution fit	Prepare problem - solution fit document & solution architecture	07 OCTOBER 2022

Proposed Solution	Preparing the new idea for our problem statement	07 OCTOBER 2022
Solution Architecture	Prepare Solution Architecture document	14 OCTOBER 2022
Customer journey	Prepare the customer journey maps to understand the user interactions & experiences with the application	17 OCTOBER 2022
Solution requirement	Prepare the Functional Requirement Document	17 OCTOBER 2022
Data flow diagrams	Prepare the Data Flow Diagrams	17 OCTOBER 2022
Technology architecture	Prepare Technology Architecture of the solution	17 OCTOBER 2022
Prepare Milestone & activity list	Prepare the Milestone& activity list of the project	26 OCTOBER 2022
Sprint Delivery Plan	Prepare the plan for all the sprints in the project	26 OCTOBER 2022
Project development – delivery of sprint –1,2,3 & 4	Develop & submit the developed code by testing it	19 NOVEMBER 2022

6.3 Reports from JIRA

The reports from JIRA includes as

- Original Estimate - The original estimate of the total amount of time it would take to complete this issue.
- Estimated Time Remaining - The current estimate of the remaining amount of time it would take to complete this issue.
- Time Spent - The amount of time spent on the issue. This is the aggregate amount of time that has been logged against this issue.
- Accuracy- The accuracy of the original estimate compared to the current estimate for the issue. It is the difference between the sum of the Time Spent and Estimated Time Remaining fields, and the Original Estimate field.



7. CODING & SOLUTIONING

7.1 Feature 1

The classification algorithms Random Forest, KNN, decision tree, and logistic regression were feed with specific features and all classifier parameters gives the best classification performance, and the results from all methods were positive. Thus the results show the Random Forest algorithm provides an accuracy of 98% that is higher than that of the other three algorithms. It is highly believed that the proposed system can reduce the risk of chronic diseases by predicting them earlier and also reduces the cost for diagnosis, treatment, and doctor consultation.

Thus we increased the four standard metrics - accuracy,precision,f1-score and recall using the Random Forest Algorithm and our proposed model as been achieved and this is the addition features included in code.

Feature 1 code:

Model Training

```
# Importing Performance Metrics:
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report, precision_score, recall_score, f1_score
```

[31] Python

```
# RandomForestClassifier:
from sklearn.ensemble import RandomForestClassifier
from sklearn import metrics
RandomForest = RandomForestClassifier()
RandomForest = RandomForest.fit(x_train,y_train)

# Predictions:
y_pred = RandomForest.predict(x_test)

# Performance:
q=round(metrics.precision_score(y_test,y_pred)*100,2)
w=round(metrics.accuracy_score(y_test,y_pred)*100,2)
e=round(metrics.recall_score(y_test,y_pred)*100,2)
r=round(metrics.f1_score(y_test,y_pred)*100,2)

print('\nPrecision: ' ,str(q))
print('Accuracy: ' ,str(w))
print('Recall: ' ,str(e))
print('F1-score: ' ,str(r))
print(classification_report(y_test,y_pred))

#display confusion matrix
print('\nConfusion Matrix: \n' ,metrics.confusion_matrix(y_test,y_pred))
#plot confusion matrix
sns.heatmap(metrics.confusion_matrix(y_test,y_pred), annot = True)
plt.show()
```

[32] Python

Output:

```
Precision:  96.88
Accuracy:  98.33
Recall:  100.0
F1-score:  98.41
              precision    recall  f1-score   support

           0       1.00      0.97      0.98        58
           1       0.97      1.00      0.98        62

    accuracy                           0.98       120
   macro avg       0.98      0.98      0.98       120
weighted avg       0.98      0.98      0.98       120


Confusion Matrix:
 [[56  2]
 [ 0 62]]
```

While comparing the existing solution, we get more accuracy of 98.33% using the random forest algorithms. Space Complexity and Time complexity would be reduced. So, the speed of getting the final outcome would be increased.

8. TESTING

Software Testing is Important because if there are any bugs or errors in the software, it can be identified early and can be solved before delivery of the software product. Properly tested software product ensures reliability, security and high performance which further results in time saving, cost effectiveness and customer satisfaction.

Here are the benefits of using software testing:

- **Cost-Effective:** It is one of the important advantages of software testing. Testing any IT project on time helps you to save your money for the long term. In case if the bugs caught in the earlier stage of software testing, it costs less to fix.
- **Security:** It is the most vulnerable and sensitive benefit of software testing. People are looking for trusted products. It helps in removing risks and problems earlier.

- **Product quality:** It is an essential requirement of any software product. Testing ensures a quality product is delivered to customers.
- **Customer Satisfaction:** The main aim of any product is to give satisfaction to their customers. UI/UX Testing ensures the best user experience.

8.1 Test Cases

A test case template is a document containing an organized list of test cases for different test scenarios that check whether or not the software has the intended functionality. A test case is a set of steps carried out to test a specific feature of an application.This report shows the number of test cases that have passed, failed, and untested.

A set of test inputs, execution conditions, and expected results developed for a particular objective, such as to exercise a particular program path or to verify compliance with a specific requirement.To write the test case, we must have the requirements to derive the inputs, and the test scenarios must be written so that we do not miss out on any features for testing.

Section	TotalCases	Not Tested	Fail	Pass
Home Page	7	0	0	7
Client user Application	51	0	0	51
Prediction	2	0	0	2
Pops up	43	0	0	43
URL port	9	0	0	9
FinalPredictionOutput	14	0	0	14
Redirecting	2	0	0	2

8.2 User Acceptance Testing

1.PurposeofDocument

The purpose of this document is to briefly explain thetestcoverageandopenissuesofthe [ProductName] project at the time of the release to User Acceptance Testing (UAT).

2.DefectAnalysis

This report shows the number of resolved or closed bugs at each severity level, and how they were resolved.

Resolution	Severity1	Severity2	Severity3	Severity4	Subtotal
By Design	8	4	2	3	18
Duplicate	1	0	4	0	5
External	2	3	0	2	7
Fixed	11	2	4	10	27
Not Reproduced	0	0	1	0	1
Skipped	0	1	1	1	3
Won'tFix	0	5	2	11	18
Totals	22	15	14	27	82

9. RESULTS

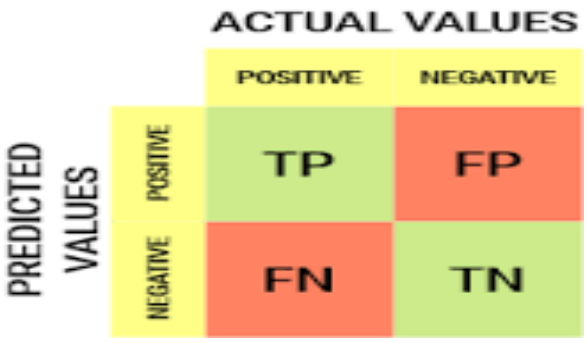
Machine learning methods are effective in CKD prediction. This work proposes a workflow to predict CKD status based on clinical data, incorporating data prepossessing, a missing value handling method with collaborative filtering and attributes selection.In this we used chronic kidney disease dataset completely and used a four types of algorithm - Decision Tree , Random Forest , Logestic Regression and K-Nearest Neibhour .The study concluded that it may be used to build an automated system for the detection of severity of CKD.

9.1 Performance Metrics

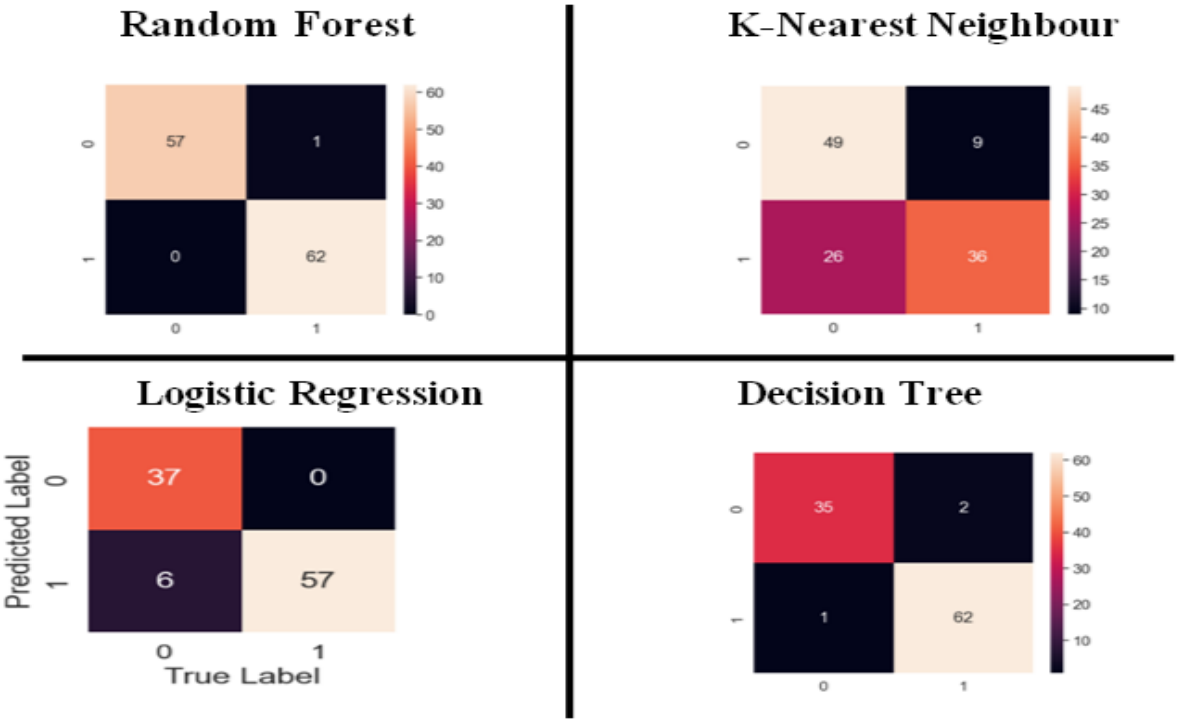
Models are evaluated using four standard metrics: Confusion matrix, Accuracy, Precision, Recall, and F1-score.

A. **Confusion matrix**

The confusion matrix is a method of evaluating the performance of a classification model. The idea behind this is to count the number of times instances of class 1 are classified as class 2. The confusion matrix is a matrix used to determine the performance of the classification models for a given set of test data.



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.



B.Accuracy

Accuracy refers to the proportion of correct prediction to total predictions. Accuracy can be explored as the ability to accurately predict the outcome of a situation.Accuracy is a metric that generally describes how the model performs across all classes. It is useful when all classes are of equal importance. It is calculated as the ratio between the number of correct predictions to the total number of predictions.The result is 0.9833, which means the model is 98.33% accurate in making a correct prediction.

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}}$$

C.Precision

Precision helps us to visualize the reliability of the machine learning model in classifying the model as positive. Precision is the ratio between the True Positives and all the Positives. In this project we measure of patients that we correctly identify having a kidney disease out of all the patients actually having it. Mathematically:

$$Precision = \frac{TruePositive}{TruePositive + FalsePositive}$$

D.Recall

The recall is calculated as the ratio between the number of *Positive* samples correctly classified as *Positive* to the total number of *Positive* samples. The recall measures the model's ability to detect *Positive* samples. The higher the recall, the more positive samples detected.

$$Recall = \frac{TruePositive}{TruePositive + FalseNegative}$$

E.F1 score

F1-score is ameasure of model accuracy on a dataset. It is used to evaluated binary classification systems which classify class label into 'positive' or 'negative’. It is a

harmonic mean alternative metric for the more common arithmetic mean. It is used when computing an average rate.

$$F1\ Score = 2 \times \frac{recall \times precision}{recall + precision}$$

Comparative analysis of the Performance Metrics of all four Algorithm used in this project

Algorithms	Precision (%)	Accuracy (%)	Recall (%)	F1-score (%)
Random Forest	96.8	98.33	100.0	98.41
K-Nearest Neighbor	80.0	70.83	58.06	67.29
Logistic Regression	88.71	86.0	88.71	88.71
Decision Tress	96.61	93.0	91.94	94.21

10. ADVANTAGES & DISADVANTAGES

Advantages of predicting Chronic Kidney Disease at earlier stage:

- The goals of early CKD detection are to prevent CKD progression and associated complications, thus improving patient outcomes and reducing the impact of CKD on health-care resources.
- With the many functions of our kidneys and the toxins they filter, these vital organs are prone to diseases that can greatly affect one's health. This is also the reason behind the importance of research in getting an in-depth view of

kidney diseases, such as chronic kidney disease prediction and end-stage renal disease.

- Machine Learning can review large volumes of data and discover specific trends and patterns that would not be apparent to humans.

Disadvantages of predicting Chronic Kidney Disease at earlier stage:

- Machine Learning requires massive data sets to train on, and these should be inclusive/unbiased, and of good quality. There can also be times where they must wait for new data to be generated.
- CKD is associated with increased risks of cardiovascular morbidity, premature mortality, and has severe impact on quality of life.hronic kidney disease can cause other problems throughout your body including: Heart and blood vessel problems. Anemia (low red blood cell count) Bone problems.So that the patients always get tension about the occurence of kidney disease or not.

11. CONCLUSION

Chronic kidney disease (CKD) is a major global public health problem, but early-stage diagnosis is problematic due to asymptomatic presentation. Currently, there are no widely accepted predictive instruments for early CKD; therefore, physicians must make clinical decisions about which patients to treat. In this study, our aim was to explore important risk factors of early CKD and discuss the associations between them. Importantly, early CKD awareness is essential for potential patients to participate in, and comply with, health examination programs, and is of great clinical and economic significance.

Thus the study on the predicted of CKD patients to address their condition in the early stages of the disease. The data was gathered from 24 + class label = 25(11 numeric, 14 nominal) were found in 400 patients. 30% of the dataset was used for testing and validation, and the remaining 70% was used for training. Replace

missing numerical and nominal values and remove outliers from the dataset for mean and mode statistical measures. The classification algorithms Random Forest, KNN, decision tree, and logistic regression were feed with specific features and all classifier parameters gives the best classification performance, and the results from all methods were positive. Thus the results show the Random Forest algorithm provides an accuracy of 98% that is higher than that of the other three algorithms. It is highly believed that the proposed system can reduce the risk of chronic diseases by predicting them earlier and also reduces the cost for diagnosis, treatment, and doctor consultation.

12. FUTURE SCOPE

The rapid growth of digital data and global accessibility through the modern internet has seen a massive rise in machine learning research. Predictive analytics supports risk identification and management by applying machine learning algorithms to aggregated data sets in order to uncover patterns, correlations, and vulnerabilities, as well as map changes within any given industry.

Since machine learning needs you to know computer programming, statistics and data evaluation, the future scope of your machine learning career can also be in leadership roles in automation or analytics environments that use data science, big data analysis, AI integration etc.

13. APPENDIX

Source Code

Google Colaboratory

Data Preprocessing,Building the model and evaluating model .ipynb

Importing necessary libraries

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as mlt
import seaborn as sns
```

%matplotlib inline

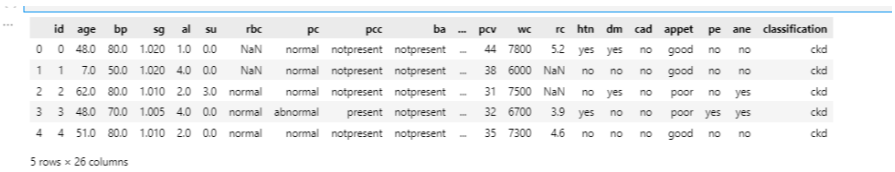
Loading Dataset

```
import os, types
import pandas as pd
from botocore.client import Config
import ibm_boto3

def __iter__(self): return 0

# @hidden_cell
# The following code accesses a file in your IBM Cloud Object Storage. It includes your credentials.
# You might want to remove those credentials before you share the notebook.
cos_client = ibm_boto3.client(service_name='s3',

ibm_api_key_id='WiDi833wSEHQZbeoeJfZgz2thD4bULAm72z9f_zt3AfY',
    ibm_auth_endpoint="https://iam.cloud.ibm.com/oidc/token",
    config=Config(signature_version='oauth'),
    endpoint_url='https://s3.private.us.cloud-object-storage.appdomain.cloud')

bucket = 'earlydetectionofchronickidneydise-donotdelete-pr-w6e8tegrdivdvc'
object_key = 'chronickidneydisease.csv'
body = cos_client.get_object(Bucket=bucket,Key=object_key)['Body']
# add missing __iter__ method, so pandas accepts body as file-like object
if not hasattr(body, "__iter__"): body.__iter__ = types.MethodType( __iter__,
```

body)

df_data_1 = pd.read_csv(body)

df_data_1.head()

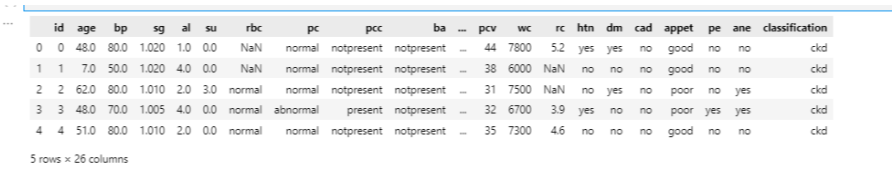


	id	age	bp	sg	al	su	rbc	pc	pcc	ba	...	pcv	wc	rc	htn	dm	cad	appet	pe	ane	classification
0	0	48.0	80.0	1.020	1.0	0.0	NaN	normal	notpresent	notpresent	...	44	7800	5.2	yes	yes	no	good	no	no	ckd
1	1	7.0	50.0	1.020	4.0	0.0	NaN	normal	notpresent	notpresent	...	38	6000	NaN	no	no	no	good	no	no	ckd
2	2	62.0	80.0	1.010	2.0	3.0	normal	normal	notpresent	notpresent	...	31	7500	NaN	no	yes	no	poor	no	yes	ckd
3	3	48.0	70.0	1.005	4.0	0.0	normal	abnormal	present	notpresent	...	32	6700	3.9	yes	no	no	poor	yes	yes	ckd
4	4	51.0	80.0	1.010	2.0	0.0	normal	normal	notpresent	notpresent	...	35	7300	4.6	no	no	no	good	no	no	ckd

5 rows × 26 columns

data=df_data_1

data.head()



	id	age	bp	sg	al	su	rbc	pc	pcc	ba	...	pcv	wc	rc	htn	dm	cad	appet	pe	ane	classification
0	0	48.0	80.0	1.020	1.0	0.0	NaN	normal	notpresent	notpresent	...	44	7800	5.2	yes	yes	no	good	no	no	ckd
1	1	7.0	50.0	1.020	4.0	0.0	NaN	normal	notpresent	notpresent	...	38	6000	NaN	no	no	no	good	no	no	ckd
2	2	62.0	80.0	1.010	2.0	3.0	normal	normal	notpresent	notpresent	...	31	7500	NaN	no	yes	no	poor	no	yes	ckd
3	3	48.0	70.0	1.005	4.0	0.0	normal	abnormal	present	notpresent	...	32	6700	3.9	yes	no	no	poor	yes	yes	ckd
4	4	51.0	80.0	1.010	2.0	0.0	normal	normal	notpresent	notpresent	...	35	7300	4.6	no	no	no	good	no	no	ckd

5 rows × 26 columns

Renaming the columns

data.columns

Index(['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'],

dtype='object')

Dropping unneccsary feature :

data = data.drop('id', axis=1)

Shape of dataset:

data.shape

(400, 25)

#manullay renaming the column names

data.columns=['age','bloodpressure','specificgravity','albumin','sugar','redbloo

dcells','puscell','puscellclumps','bacteria','bloodglucoserandom','bloodurea','serumcreatinine','sodium','potassium','haemoglobin','packedcellvolume','whitebloodcellcount','redbloodcellcount','hypertension','diabetesmellitus','coronaryarterydisease','appetite','pedaledema','anemia','class']

Descriptive statistics of the dataset

data.describe()

	age	bloodpressure	specificgravity	albumin	sugar	bloodglucoserandom	bloodurea	serumcreatinine	sodium	potassium	haemoglobin
count	391.000000	388.000000	353.000000	354.000000	351.000000	356.000000	381.000000	383.000000	313.000000	312.000000	348.000000
mean	51.483376	76.469072	1.017408	1.016949	0.450142	148.036517	57.425722	3.072454	137.528754	4.627244	12.526437
std	17.169714	13.683637	0.005717	1.352679	1.099191	79.281714	50.503006	5.741126	10.408752	3.193904	2.912587
min	2.000000	50.000000	1.005000	0.000000	0.000000	22.000000	1.500000	0.400000	4.500000	2.500000	3.100000
25%	42.000000	70.000000	1.010000	0.000000	0.000000	99.000000	27.000000	0.900000	135.000000	3.800000	10.300000
50%	55.000000	80.000000	1.020000	0.000000	0.000000	121.000000	42.000000	1.300000	138.000000	4.400000	12.650000
75%	64.500000	80.000000	1.020000	2.000000	0.000000	163.000000	66.000000	2.800000	142.000000	4.900000	15.000000
max	90.000000	180.000000	1.025000	5.000000	5.000000	490.000000	391.000000	76.000000	163.000000	47.000000	17.800000

Data Preprocessing

data.info()

Output exceeds the size limit. Open the full output data in a text editor

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 400 entries, 0 to 399

Data columns (total 25 columns):

#	Column	Non-Null Count	Dtype
0	age	391 non-null	float64
1	bloodpressure	388 non-null	float64
2	specificgravity	353 non-null	float64
3	albumin	354 non-null	float64
4	sugar	351 non-null	float64
5	redbloodcells	248 non-null	object
6	puscell	335 non-null	object
7	puscellclumps	396 non-null	object
8	bacteria	396 non-null	object
9	bloodglucoserandom	356 non-null	float64

```
 10  bloodurea             381 non-null    float64
 11  serumcreatinine       383 non-null    float64
 12  sodium                313 non-null    float64
 13  potassium             312 non-null    float64
 14  haemoglobin           348 non-null    float64
 15  packedcellvolume      330 non-null    object
 16  whitebloodcellcount   295 non-null    object
 17  redbloodcellcount     270 non-null    object
 18  hypertension          398 non-null    object
 19  diabetesmellitus      398 non-null    object
...
 23  anemia                399 non-null    object
 24  class                 400 non-null    object
dtypes: float64(11), object(14)
memory usage: 78.2+ KB
#checking the missing values
data.isnull().sum()
age                   9
bloodpressure         12
specificgravity       47
albumin               46
sugar                 49
redbloodcells         152
puscell               65
puscellclumps         4
bacteria              4
bloodglucoserandom    44
bloodurea             19
serumcreatinine       17
```

```
sodium                  87
potassium               88
haemoglobin             52
packedcellvolume        70
whitebloodcellcount     105
redbloodcellcount       130
hypertension            2
diabetesmellitus        2
coronaryarterydisease   2
appetite                1
pedaledema              1
anemia                  1
class                   0
dtype: int64
#Replacing Categorical values with numbers
#red blood cells
data['redbloodcells'].value_counts()
data['redbloodcells'] = data['redbloodcells'].replace(to_replace = {'normal' : 0, 'abnormal' : 1})
#pus cell
data['puscell'].value_counts()
data['puscell'] = data['puscell'].replace(to_replace = {'normal' : 0, 'abnormal' : 1})
#pus cell clumps
data['puscellclumps'].value_counts()
data['puscellclumps'] = data['puscellclumps'].replace(to_replace = {'notpresent':0,'present':1})
#bacteria
data['bacteria'].value_counts()
```

```
data['bacteria'] = data['bacteria'].replace(to_replace =
{'notpresent':0,'present':1})
#hypertension
data['hypertension'].value_counts()
data['hypertension'] = data['hypertension'].replace(to_replace = {'yes' : 1, 'no'
: 0})
#diabetes mellitus
data['diabetesmellitus'].value_counts()
data['diabetesmellitus'] = data['diabetesmellitus'].replace(to_replace =
{'\tyes':'yes', ' yes':'yes', '\tno':'no'})
data['diabetesmellitus'] = data['diabetesmellitus'].replace(to_replace = {'yes' :
1, 'no' : 0})
#coronary artery disease
data['coronaryarterydisease'].value_counts()
data['coronaryarterydisease'] =
data['coronaryarterydisease'].replace(to_replace = {'\tno':'no'})
data['coronaryarterydisease'] =
data['coronaryarterydisease'].replace(to_replace = {'yes' : 1, 'no' : 0})
#appetite
data['appetite'].unique()
data['appetite'] =
data['appetite'].replace(to_replace={'good':1,'poor':0,'no':np.nan})
#pedal edema
data['pedaledema'].value_counts()
data['pedaledema'] = data['pedaledema'].replace(to_replace = {'yes' : 1, 'no' :
0})
#anemia
data['anemia'].value_counts()
data['anemia'] = data['anemia'].replace(to_replace = {'yes' : 1, 'no' : 0})
```

```
#class
data['class'].value_counts()
data['class'] = data['class'].replace(to_replace={'ckd\t':'ckd'})
data["class"] = [1 if i == "ckd" else 0 for i in data["class"]]

data.dtypes
```

```
age                   float64
bloodpressure         float64
specificgravity       float64
albumin               float64
sugar                 float64
redbloodcells         float64
puscell               float64
puscellclumps         float64
bacteria              float64
bloodglucoserandom    float64
bloodurea             float64
serumcreatinine       float64
sodium                float64
potassium             float64
haemoglobin           float64
packedcellvolume       object
whitebloodcellcount    object
redbloodcellcount      object
hypertension          float64
diabetesmellitus      float64
coronaryarterydisease float64
appetite              float64
pedaledema            float64
```

```
anemia                 float64
class                  int64
dtype: object
```

```
#Converting Object values into Numeric values
data['packedcellvolume'] = pd.to_numeric(data['packedcellvolume'],
errors='coerce')
data['whitebloodcellcount'] = pd.to_numeric(data['whitebloodcellcount'],
errors='coerce')
data['redbloodcellcount'] = pd.to_numeric(data['redbloodcellcount'],
errors='coerce')
```

```
data.dtypes
age                  float64
bloodpressure          float64
specificgravity        float64
albumin                float64
sugar                  float64
redbloodcells          float64
puscell                float64
puscellclumps          float64
bacteria               float64
bloodglucoserandom     float64
bloodurea              float64
serumcreatinine        float64
sodium                 float64
potassium              float64
haemoglobin            float64
```

```
packedcellvolume         float64
whitebloodcellcount      float64
redbloodcellcount        float64
hypertension             float64
diabetesmellitus         float64
coronaryarterydisease    float64
appetite                 float64
pedaledema               float64
anemia                   float64
class                    int64
dtype: object
```

Handling Null Values

```
features=['age','bloodpressure','specificgravity','albumin','sugar','redbloodcells','puscell','puscellclumps','bacteria','bloodglucoserandom','bloodurea','serumcreatinine','sodium','potassium','haemoglobin','packedcellvolume','whitebloodcellcount','redbloodcellcount','hypertension','diabetesmellitus','coronaryarterydisease','appetite','pedaledema','anemia']
for i in features:
    data[i] = data[i].fillna(data[i].median())
data = data.rename(columns={'class':'target'})
print(data)
```

Output exceeds the size limit. Open the full output data in a text editor

```
     age  bloodpressure  specificgravity  albumin  sugar  redbloodcells \
0   48.0           80.0            1.020      1.0    0.0            0.0
1    7.0           50.0            1.020      4.0    0.0            0.0
2   62.0           80.0            1.010      2.0    3.0            0.0
3   48.0           70.0            1.005      4.0    0.0            0.0
```

```
4    51.0         80.0          1.010     2.0    0.0          0.0
..    ...          ...           ...      ...    ...          ...
395  55.0         80.0          1.020     0.0    0.0          0.0
396  42.0         70.0          1.025     0.0    0.0          0.0
397  12.0         80.0          1.020     0.0    0.0          0.0
398  17.0         60.0          1.025     0.0    0.0          0.0
399  58.0         80.0          1.025     0.0    0.0          0.0

     puscell  puscellclumps  bacteria  bloodglucoserandom  ...  \
0        0.0            0.0       0.0               121.0  ...
1        0.0            0.0       0.0               121.0  ...
2        0.0            0.0       0.0               423.0  ...
3        1.0            1.0       0.0               117.0  ...
4        0.0            0.0       0.0               106.0  ...
..       ...            ...       ...                 ...  ...
395      0.0            0.0       0.0               140.0  ...
396      0.0            0.0       0.0                75.0  ...
397      0.0            0.0       0.0               100.0  ...
398      0.0            0.0       0.0               114.0  ...
399      0.0            0.0       0.0               131.0  ...
...
398       0
399       0

[400 rows x 25 columns]

data.isnull().any().sum()
0
data.isnull().sum()
```

age 0
bloodpressure 0
specificgravity 0
albumin 0
sugar 0
redbloodcells 0
puscell 0
puscellclumps 0
bacteria 0
bloodglucoserandom 0
bloodurea 0
serumcreatinine 0
sodium 0
potassium 0
haemoglobin 0
packedcellvolume 0
whitebloodcellcount 0
redbloodcellcount 0
hypertension 0
diabetesmellitus 0
coronaryarterydisease 0
appetite 0
pedaledema 0
anemia 0
target 0

dtype: int64

Exploratory Data Analaysis

#correlation

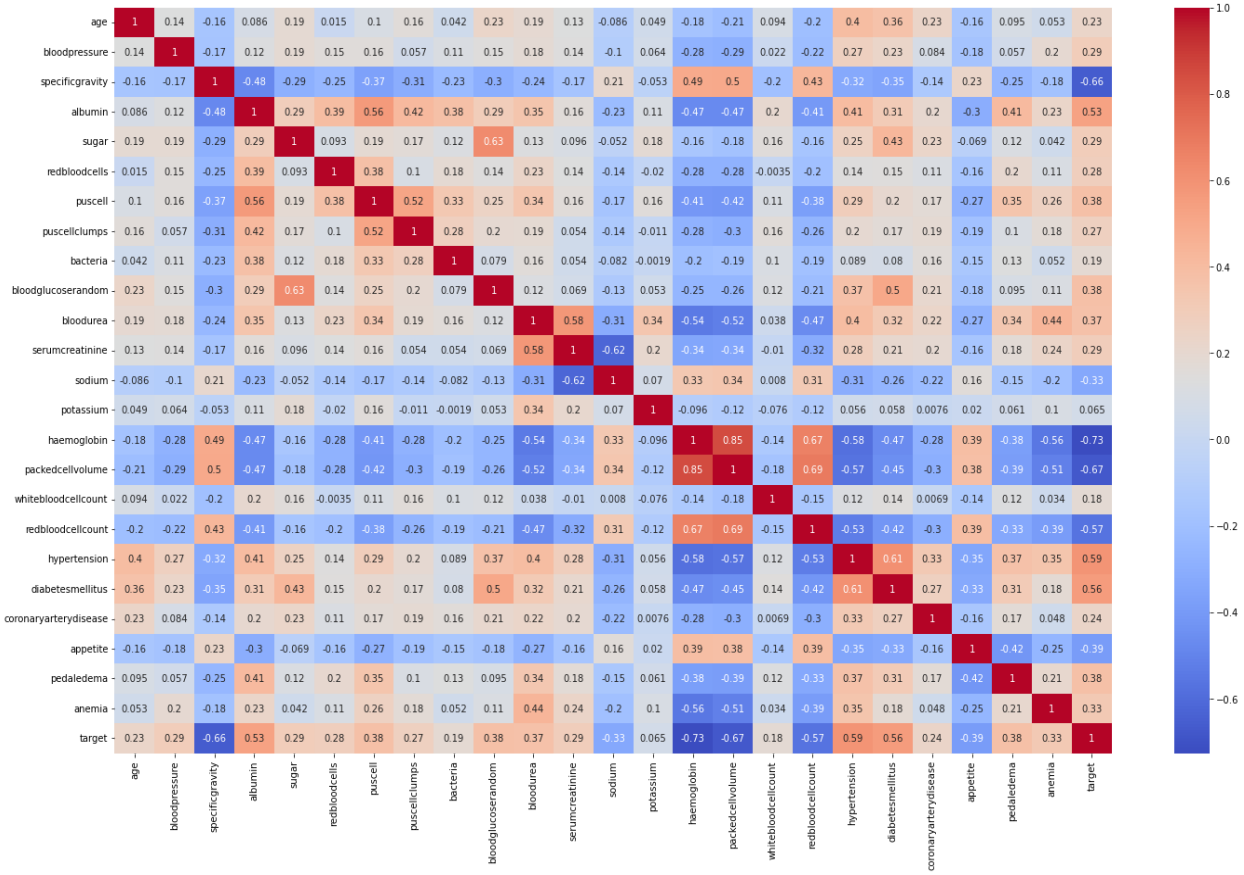
data.corr()

	age	bloodpressure	specificgravity	albumin	sugar	redbloodcells	puscell	puscellclumps	bacteria	bloodglucoserandom	...	packedcellvolume	whitebloodcellcount	redbloodce
age	1.000000	0.136316	-0.160374	0.085918	0.186750	0.014904	0.102286	0.157401	0.042427	0.230731	...	-0.212796	0.093794	-0.
bloodpressure	0.136316	1.000000	-0.166980	0.123518	0.189561	0.150384	0.156231	0.056808	0.110164	0.150180	...	-0.289237	0.022177	-0.
specificgravity	-0.160374	-0.166980	1.000000	-0.479962	-0.292053	-0.253894	-0.365353	-0.306426	-0.231704	-0.299413	...	0.496434	-0.195068	0.
albumin	0.085918	0.123518	-0.479962	1.000000	0.287751	0.394844	0.561713	0.417868	0.377935	0.293150	...	-0.473446	0.202920	-0.
sugar	0.186750	0.189561	-0.292053	0.287751	1.000000	0.092940	0.190062	0.168091	0.119399	0.627002	...	-0.181285	0.156649	-0.
redbloodcells	0.014904	0.150384	-0.253894	0.394844	0.092940	1.000000	0.377394	0.102948	0.184402	0.138615	...	-0.279875	-0.003471	-0.
puscell	0.102286	0.156231	-0.365353	0.561713	0.190062	0.377394	1.000000	0.520118	0.330401	0.247665	...	-0.418435	0.106733	-0.
puscellclumps	0.157401	0.056808	-0.306426	0.417868	0.168091	0.102948	0.520118	1.000000	0.275082	0.195223	...	-0.296580	0.163215	-0.
bacteria	0.042427	0.110164	-0.231704	0.377935	0.119399	0.184402	0.330401	0.275082	1.000000	0.079162	...	-0.189420	0.104748	-0.
bloodglucoserandom	0.230731	0.150180	-0.299413	0.293150	0.627002	0.138615	0.247665	0.195223	0.079162	1.000000	...	-0.258200	0.119881	-0.
bloodurea	0.194291	0.180841	-0.244995	0.347418	0.126897	0.233935	0.344501	0.189094	0.161676	0.118859	...	-0.523091	0.038362	-0.
serumcreatinine	0.133985	0.143184	-0.171998	0.161310	0.096434	0.135660	0.158038	0.054429	0.053959	0.068886	...	-0.338611	-0.010333	-0.
sodium	-0.086040	-0.100705	0.210791	-0.225612	-0.051758	-0.139037	-0.171171	-0.138816	-0.082156	-0.130569	...	0.343555	0.008010	0.
potassium	0.049399	0.063667	-0.052935	0.107893	0.177396	-0.020473	0.155434	-0.011416	-0.001944	0.052732	...	-0.117795	-0.075563	-0.
haemoglobin	-0.178308	-0.279303	0.490699	-0.474399	-0.156807	-0.280996	-0.411707	-0.276645	-0.204860	-0.254435	...	0.847490	-0.137978	0.
packedcellvolume	-0.212796	-0.289237	0.496434	-0.473446	-0.181285	-0.279875	-0.418435	-0.296580	-0.189420	-0.258200	...	1.000000	-0.175226	0.
whitebloodcellcount	0.093794	0.022177	-0.195068	0.202920	0.156649	-0.003471	0.106733	0.163215	0.104748	0.119881	...	-0.175226	1.000000	-0.
redbloodcellcount	-0.203199	-0.219317	0.432222	-0.405092	-0.161994	-0.196534	-0.381949	-0.263810	-0.192212	-0.214094	...	0.693473	-0.153776	1.
hypertension	0.395073	0.266901	-0.323643	0.406057	0.254268	0.140538	0.291719	0.195623	0.089046	0.367816	...	-0.569377	0.116827	-0.
diabetesmellitus	0.364306	0.226489	-0.351016	0.308101	0.430514	0.145646	0.201032	0.165236	0.080070	0.503254	...	-0.449981	0.144101	-0.
coronaryarterydisease	0.231419	0.084135	-0.135814	0.200957	0.229301	0.111493	0.172295	0.188029	0.162395	0.207020	...	-0.296691	0.006923	-0.
appetite	-0.156581	-0.175054	0.230975	-0.303145	-0.069216	-0.160868	-0.274985	-0.189688	-0.149126	-0.177285	...	0.382685	-0.142714	0.
pedaledema	0.094772	0.056902	-0.253803	0.411080	0.116442	0.199285	0.350227	0.104356	0.134732	0.094806	...	-0.388729	0.123381	-0.
anemia	0.052938	0.195134	-0.184155	0.229556	0.042464	0.107625	0.260566	0.175861	0.052208	0.112449	...	-0.514410	0.034059	-0.
target	0.227842	0.293693	-0.659504	0.531562	0.294555	0.282642	0.375154	0.265313	0.186871	0.379321	...	-0.673129	0.177571	-0.

mlt.figure(figsize=(24,14))

sns.heatmap(data.corr(), annot=True, cmap='coolwarm')

mlt.show()



#We clearly see that 'pcv' and 'hemo' feature has 85% multicollinearity

#So we remove one of the feature. i.e pcv

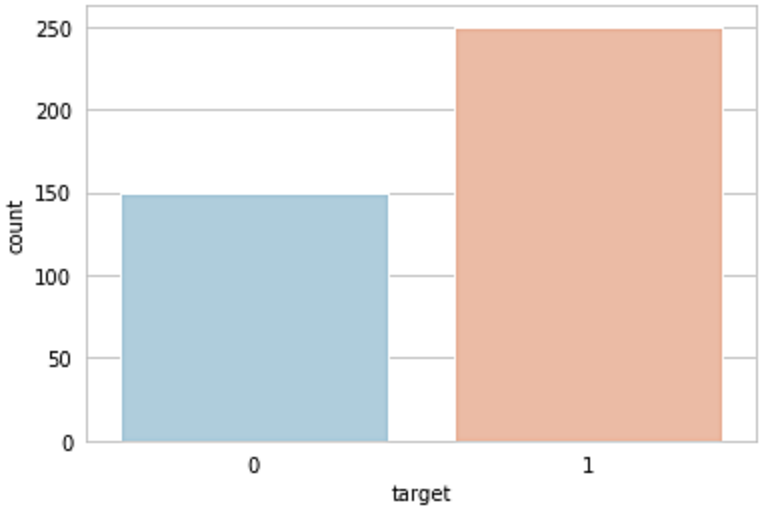
```
data.drop('packedcellvolume', axis=1, inplace=True)
```

data

	age	bloodpressure	specificgravity	albumin	sugar	redbloodcells	puscell	puscellclumps	bacteria	bloodglucoserandom	...	haemoglobin	whitebloodcellcount	redbloodcellcount	hypertension	diabetesme
0	48.0	80.0	1.020	1.0	0.0	0.0	0.0	0.0	0.0	121.0	...	15.4	7800.0	5.2	1.0	
1	7.0	50.0	1.020	4.0	0.0	0.0	0.0	0.0	0.0	121.0	...	11.3	6000.0	4.8	0.0	
2	62.0	80.0	1.010	2.0	3.0	0.0	0.0	0.0	0.0	423.0	...	9.6	7500.0	4.8	0.0	
3	48.0	70.0	1.005	4.0	0.0	0.0	1.0	1.0	0.0	117.0	...	11.2	6700.0	3.9	1.0	
4	51.0	80.0	1.010	2.0	0.0	0.0	0.0	0.0	0.0	106.0	...	11.6	7300.0	4.6	0.0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
395	55.0	80.0	1.020	0.0	0.0	0.0	0.0	0.0	0.0	140.0	...	15.7	6700.0	4.9	0.0	
396	42.0	70.0	1.025	0.0	0.0	0.0	0.0	0.0	0.0	75.0	...	16.5	7800.0	6.2	0.0	
397	12.0	80.0	1.020	0.0	0.0	0.0	0.0	0.0	0.0	100.0	...	15.8	6600.0	5.4	0.0	
398	17.0	60.0	1.025	0.0	0.0	0.0	0.0	0.0	0.0	114.0	...	14.2	7200.0	5.9	0.0	
399	58.0	80.0	1.025	0.0	0.0	0.0	0.0	0.0	0.0	131.0	...	15.8	6800.0	6.1	0.0	

400 rows × 24 columns

```
sns.set_style('whitegrid')
sns.countplot(x='target',data=data,palette='RdBu_r')
mlt.show()
```



Split the data into dependent and independent variables

```
x = data.iloc[:, :-1]
y = data.iloc[:, -1]
print(x)
print(y)
print(x.shape)
print(y.shape)
```

Output exceeds the size limit. Open the full output data in a text editor

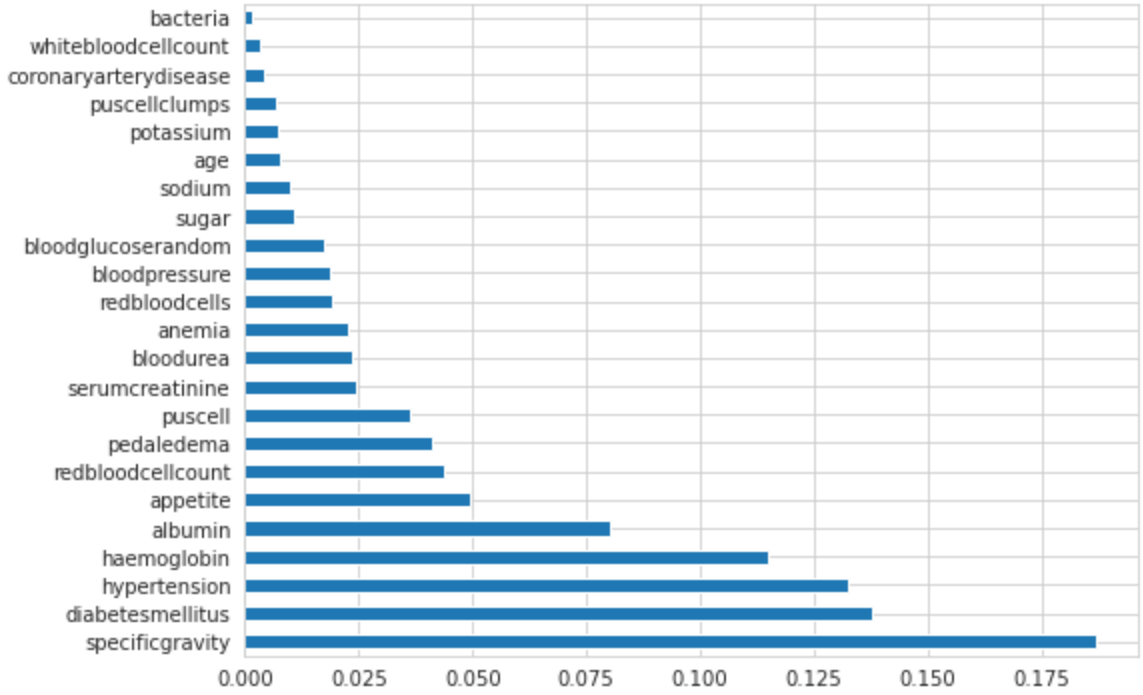
```
     age  bloodpressure  specificgravity  albumin  sugar  redbloodcells  \
0    48.0           80.0            1.020      1.0    0.0            0.0
1     7.0           50.0            1.020      4.0    0.0            0.0
2    62.0           80.0            1.010      2.0    3.0            0.0
3    48.0           70.0            1.005      4.0    0.0            0.0
4    51.0           80.0            1.010      2.0    0.0            0.0
..    ...            ...              ...      ...    ...            ...
395  55.0           80.0            1.020      0.0    0.0            0.0
```

```
396  42.0         70.0         1.025     0.0    0.0          0.0
397  12.0         80.0         1.020     0.0    0.0          0.0
398  17.0         60.0         1.025     0.0    0.0          0.0
399  58.0         80.0         1.025     0.0    0.0          0.0

     puscell  puscellclumps  bacteria  bloodglucoserandom  ...  potassium  \
0        0.0            0.0       0.0               121.0  ...        4.4
1        0.0            0.0       0.0               121.0  ...        4.4
2        0.0            0.0       0.0               423.0  ...        4.4
3        1.0            1.0       0.0               117.0  ...        2.5
4        0.0            0.0       0.0               106.0  ...        4.4
..       ...            ...       ...                 ...  ...        ...
395      0.0            0.0       0.0               140.0  ...        4.9
396      0.0            0.0       0.0                75.0  ...        3.5
397      0.0            0.0       0.0               100.0  ...        4.4
398      0.0            0.0       0.0               114.0  ...        4.9
399      0.0            0.0       0.0               131.0  ...        3.5
...
399    0
Name: target, Length: 400, dtype: int64
(400, 23)
(400,)
```

Feature Importance

```
from sklearn.ensemble import ExtraTreesClassifier
import matplotlib.pyplot as plt
model=ExtraTreesClassifier()
model.fit(x,y)
```

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



Train and Test split

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test=
train_test_split(x,y,test_size=0.3,random_state=33)
print(' x_train.shape : ',x_train.shape)
print(' y_train.shape : ',y_train.shape)
print(' x_test.shape : ',x_test.shape)
print(' y_test.shape : ',y_test.shape)
x_train.shape :  (280, 23)
 y_train.shape :  (280,)
 x_test.shape :  (120, 23)
```

y_test.shape : (120,)

Model Training

```
# Importing Performance Metrics:
from sklearn.metrics import accuracy_score, confusion_matrix,
classification_report, precision_score, recall_score, f1_score

# RandomForestClassifier:
from sklearn.ensemble import RandomForestClassifier
from sklearn import metrics
RandomForest = RandomForestClassifier()
RandomForest = RandomForest.fit(x_train,y_train)

# Predictions:
y_pred = RandomForest.predict(x_test)

# Performance:
q=round(metrics.precision_score(y_test,y_pred)*100,2)
w=round(metrics.accuracy_score(y_test,y_pred)*100,2)
e=round(metrics.recall_score(y_test,y_pred)*100,2)
r=round(metrics.f1_score(y_test,y_pred)*100,2)

print('\nPrecision: ' ,str(q))
print('Accuracy: ' ,str(w))
print('Recall: ' ,str(e))
print('F1-score: ' ,str(r))
print(classification_report(y_test,y_pred))
```

#display confusion matrix
print('\nConfusion Matrix: \n' ,metrics.confusion_matrix(y_test,y_pred))
#plot confusion matrix
sns.heatmap(metrics.confusion_matrix(y_test,y_pred), annot = True)
plt.show()

```
...
Precision:  96.88
Accuracy:  98.33
Recall:  100.0
F1-score:  98.41
              precision    recall  f1-score   support

           0       1.00      0.97      0.98        58
           1       0.97      1.00      0.98        62

    accuracy                           0.98       120
   macro avg       0.98      0.98      0.98       120
weighted avg       0.98      0.98      0.98       120


Confusion Matrix:
 [[56  2]
 [ 0 62]]
```

#K-Nearest Neighbour
from sklearn import metrics
from sklearn.neighbors import KNeighborsClassifier

k_nearest_neighbour_model = KNeighborsClassifier()
k_nearest_neighbour_model=k_nearest_neighbour_model.fit(x_train,y_train)
y_pre= k_nearest_neighbour_model.predict(x_test)

#display KNN classification Metrices for SVM
a=round(metrics.precision_score(y_test,y_pre)*100,2)
b=round(metrics.accuracy_score(y_test,y_pre)*100,2)
c=round(metrics.recall_score(y_test,y_pre)*100,2)
d=round(metrics.f1_score(y_test,y_pre)*100,2)

print('\nPrecision: ' ,str(a))

```
print('Accuracy: ' ,str(b))
print('Recall: ' ,str(c))
print('F1-score: ' ,str(d))
#display classification report for KNN
print('\nClassification Report:\n' ,metrics.classification_report(y_test,y_pre))

#display confusion matrix
print('\nConfusion Matrix: \n' ,metrics.confusion_matrix(y_test,y_pre))
#plot confusion matrix
sns.heatmap(metrics.confusion_matrix(y_test,y_pre), annot = True)
plt.show()
```

```
Precision:  80.0
Accuracy:  70.83
Recall:  58.06
F1-score:  67.29

Classification Report:
                precision    recall  f1-score   support

             0       0.65      0.84      0.74        58
             1       0.80      0.58      0.67        62

    accuracy                           0.71       120
   macro avg       0.73      0.71      0.70       120
weighted avg       0.73      0.71      0.70       120


Confusion Matrix:
 [[49  9]
 [26 36]]
```

```
#LogisticRegression
from sklearn.linear_model import LogisticRegression

logreg = LogisticRegression()

x = data.iloc[:,:-1]
```

```
y = data['target']

x_train, x_test, y_train, y_test = train_test_split(x,y, stratify = y, shuffle = True)

logreg.fit(x_train,y_train)
```

```
/opt/conda/envs/Python-3.9/lib/python3.9/site-packages/sklearn/linear_model/_logistic.py:814: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
LogisticRegression()
```

```
print("Training score")
logreg.score(x_train,y_train)
```

```
Training score
0.9533333333333334
```

```
print("testing accuracy")
logreg.score(x_test,y_test)
```

```
testing accuracy
```

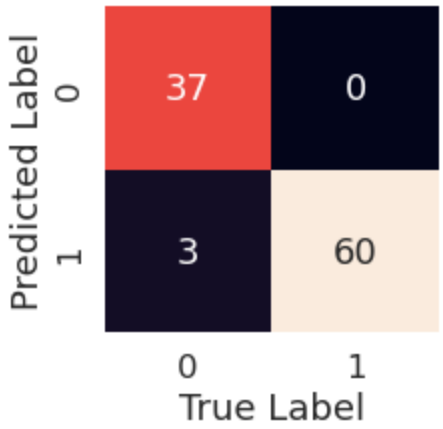
0.97

```
from sklearn import metrics
Y_pred = logreg.predict(x_test)
s=round(metrics.precision_score(y_test,Y_pred)*100,2)
f=round(metrics.accuracy_score(y_test,Y_pred)*100,2)
g=round(metrics.recall_score(y_test,Y_pred)*100,2)
h=round(metrics.f1_score(y_test,Y_pred)*100,2)

print('\nPrecision: ' ,str(s))
print('Accuracy: ' ,str(f))
print('Recall: ' ,str(g))
print('F1-score: ' ,str(h))
```

```
Precision:  100.0
Accuracy:  97.0
Recall:  95.24
F1-score:  97.56
```

```
sns.set(font_scale=1.5)
def plot_conf_mat(y_test,y_preds):

    fig,ax=plt.subplots(figsize=(3,3))
    ax=sns.heatmap(confusion_matrix(y_test,y_preds),annot=True,cbar=False)
    plt.xlabel("True Label")
    plt.ylabel("Predicted Label")
```

```
log_pred = logreg.predict(x_test)
plot_conf_mat(y_test, log_pred)
```




```
#Decision tree
from sklearn.tree import DecisionTreeClassifier

#initialise the decision tree Model
decision_tree_model = DecisionTreeClassifier(random_state = 0)

decision_tree_model=decision_tree_model.fit(x_train,y_train)
ypred=decision_tree_model.predict(x_test)
#display KNN classification Metrices for Decision Tree
z=round(metrics.precision_score(y_test,ypred)*100,2)
v=round(metrics.accuracy_score(y_test,ypred)*100,2)
n=round(metrics.recall_score(y_test,ypred)*100,2)
m=round(metrics.f1_score(y_test,ypred)*100,2)

print('\nPrecision: ' ,str(z))
print('Accuracy: ' ,str(v))
```

```
print('Recall: ' ,str(n))
print('F1-score: ' ,str(m))

#display classification report for Decision Tree
print('\nClassification Report:\n' ,metrics.classification_report(y_test,ypred))

#display confusion matrix
print('\nConfusion Matrix: \n' ,metrics.confusion_matrix(y_test,ypred))
#plot confusion matrix
sns.heatmap(metrics.confusion_matrix(y_test,ypred), annot = True)
plt.show()
```

```
...
      Precision:  96.61
      Accuracy:  93.0
      Recall:  91.94
      F1-score:  94.21

      Classification Report:
                    precision    recall  f1-score   support

                 0       0.88      0.95      0.91        38
                 1       0.97      0.92      0.94        62

          accuracy                           0.93       100
         macro avg       0.92      0.93      0.93       100
      weighted avg       0.93      0.93      0.93       100


      Confusion Matrix:
       [[36  2]
       [ 5 57]]
```

```
#Precision
precision = [q,a,s,z]
algorithms = ["Randomforest","K-Nearest Neighbors","Logistic Regression","Decision Tree"]
```

```
for i in range(len(algorithms)):
    print("The Precision score achieved using "+algorithms[i]+" is:
"+str(precision[i])+" %")
```

The Precision score achieved using Randomforest is: 96.88 %
The Precision score achieved using K-Nearest Neighbors is: 80.0 %
The Precision score achieved using Logistic Regression is: 100.0 %
The Precision score achieved using Decision Tree is: 100.0 %

```
#Accuracy
accuracy = [w,b,f,v]
algorithms = ["Randomforest","K-Nearest Neighbors","Logistic
Regression","Decision Tree"]

for i in range(len(algorithms)):
    print("The accuracy score achieved using "+algorithms[i]+" is:
"+str(accuracy[i])+" %")
```

The accuracy score achieved using Randomforest is: 98.33 %
The accuracy score achieved using K-Nearest Neighbors is: 70.83 %
The accuracy score achieved using Logistic Regression is: 97.0 %
The accuracy score achieved using Decision Tree is: 97.0 %

```
#Recall
recall = [e,c,g,n]
algorithms = ["Randomforest","K-Nearest Neighbors","Logistic
Regression","Decision Tree"]
```

```
for i in range(len(algorithms)):
    print("The recall score achieved using "+algorithms[i]+" is:
"+str(recall[i])+" %")
```

```
The recall score achieved using Randomforest is: 100.0 %
The recall score achieved using K-Nearest Neighbors is: 58.06 %
The recall score achieved using Logistic Regression is: 95.24 %
The recall score achieved using Decision Tree is: 95.24 %
```

```
#F1-score
score = [r,d,h,m]
algorithms = ["Randomforest","K-Nearest Neighbors","Logistic
Regression","Decision Tree"]

for i in range(len(algorithms)):
    print("The F1 score achieved using "+algorithms[i]+" is: "+str(score[i])+"
%")
```

```
The F1 score achieved using Randomforest is: 98.41 %
The F1 score achieved using K-Nearest Neighbors is: 67.29 %
The F1 score achieved using Logistic Regression is: 97.56 %
The F1 score achieved using Decision Tree is: 97.56 %
```

Based on the above table, it can be noted that Random Forest Classifier has the highest accuracy (99%), highest recall (100%), highest precision (98.4%) and highest F1-score (99.1%) of all the other algorithms.

Pickling the model file for deployment

```
from sklearn.ensemble import RandomForestClassifier
RandomForest= RandomForestClassifier()
```

```
RandomForest =RandomForest.fit(x_train,y_train)

from sklearn.model_selection import cross_val_score
cv = cross_val_score(RandomForest,x,y,cv=5)
np.mean(cv)
0.9925

import pickle
# Creating a pickle file for the classifier
filename = 'RandomForest.pkl'
pickle.dump(RandomForest, open(filename, 'wb'))

pwd
'/home/wsuser/work'
```

Deployment

```
from ibm_watson_machine_learning import APIClient
import json
import numpy as np
```

Authenticate and Set Space

```
wml_credentials={
   "apikey":"a-qUIpHZUwdZyhqTXKdDX9PKskdMto90F2YK-FEe1VTw",
   "url":"https://us-south.ml.cloud.ibm.com"
}
```

```
wml_client=APIClient(wml_credentials)
wml_client.spaces.list()
Note: 'limit' is not provided. Only first 50 records will be displayed if the
number of records exceed 50
------------------------------------  -------------  ------------------------
ID                                    NAME           CREATED
4cc08e3f-9703-49f7-b9fb-c59023271670  ckd_Prediction  2022-11-
15T04:47:52.214Z
------------------------------------  -------------  ------------------------

SPACE_ID="4cc08e3f-9703-49f7-b9fb-c59023271670"

wml_client.set.default_space(SPACE_ID)
'SUCCESS'
wml_client.software_specifications.list(500)
Output exceeds the size limit. Open the full output data in a text editor
------------------------------  --------------------------------  ----
NAME                            ASSET_ID                          TYPE
default_py3.6                   0062b8c9-8b7d-44a0-a9b9-46c416adcbd9  base
kernel-spark3.2-scala2.12       020d69ce-7ac1-5e68-ac1a-31189867356a
base
pytorch-onnx_1.3-py3.7-edt      069ea134-3346-5748-b513-49120e15d288
base
scikit-learn_0.20-py3.6         09c5a1d0-9c1e-4473-a344-eb7b665ff687  base
spark-mllib_3.0-scala_2.12      09f4cff0-90a7-5899-b9ed-1ef348aebdee  base
pytorch-onnx_rt22.1-py3.9       0b848dd4-e681-5599-be41-b5f6fccc6471
```

base
ai-function_0.1-py3.6 0cdb0f1e-5376-4f4d-92dd-da3b69aa9bda base
shiny-r3.6 0e6e79df-875e-4f24-8ae9-62dcc2148306 base
tensorflow_2.4-py3.7-horovod 1092590a-307d-563d-9b62-4eb7d64b3f22
base
pytorch_1.1-py3.6 10ac12d6-6b30-4ccd-8392-3e922c096a92 base
tensorflow_1.15-py3.6-ddl 111e41b3-de2d-5422-a4d6-bf776828c4b7
base
autoai-kb_rt22.2-py3.10 125b6d9a-5b1f-5e8d-972a-b251688ccf40
base
runtime-22.1-py3.9 12b83a17-24d8-5082-900f-0ab31fbfd3cb base
scikit-learn_0.22-py3.6 154010fa-5b3b-4ac1-82af-4d5ee5abbc85 base
default_r3.6 1b70aec3-ab34-4b87-8aa0-a4a3c8296a36 base
pytorch-onnx_1.3-py3.6 1bc6029a-cc97-56da-b8e0-39c3880dbbe7
base
kernel-spark3.3-r3.6 1c9e5454-f216-59dd-a20e-474a5cdf5988 base
pytorch-onnx_rt22.1-py3.9-edt 1d362186-7ad5-5b59-8b6c-9d0880bde37f
base
tensorflow_2.1-py3.6 1eb25b84-d6ed-5dde-b6a5-3fbdf1665666 base
spark-mllib_3.2 20047f72-0a98-58c7-9ff5-a77b012eb8f5 base
tensorflow_2.4-py3.8-horovod 217c16f6-178f-56bf-824a-b19f20564c49
base
runtime-22.1-py3.9-cuda 26215f05-08c3-5a41-a1b0-da66306ce658
base
do_py3.8 295addb5-9ef9-547e-9bf4-92ae3563e720 base
...
pytorch-onnx_rt22.2-py3.10-edt f8a05d07-e7cd-57bb-a10b-23f1d4b837ac
base
scikit-learn_0.19-py3.6 f963fa9d-4bb7-5652-9c5d-8d9289ef6ad9 base

```
tensorflow_2.4-py3.8          fe185c44-9a99-5425-986b-59bd1d2eda46  base
------------------------------  ------------------------------------  ----
```

Save and Deploy the Model

```
import sklearn
sklearn.__version__
'1.0.2'

MODEL_NAME='ckd'
DEPLOYMENT_NAME='ckd_Prediction'
DEMO_MODEL=RandomForest

#set Python version
software_spec_uid=wml_client.software_specifications.get_id_by_name('run
time-22.1-py3.9')
#setup model meta
model_props={
    wml_client.repository.ModelMetaNames.NAME:MODEL_NAME,
    wml_client.repository.ModelMetaNames.TYPE:'scikit-learn_1.0',

wml_client.repository.ModelMetaNames.SOFTWARE_SPEC_UID:software
_spec_uid
}
#Save Model
model_details=wml_client.repository.store_model(
```

```
    model=DEMO_MODEL,
    meta_props=model_props,
    training_data=x_train,
    training_target=y_train
)
model_details
```

Output exceeds the size limit. Open the full output data in a text editor


```
{'entity': {'hybrid_pipeline_software_specs': [],
  'label_column': 'target',
  'schemas': {'input': [{'fields': [{'name': 'age', 'type': 'float64'},
      {'name': 'bloodpressure', 'type': 'float64'},
      {'name': 'specificgravity', 'type': 'float64'},
      {'name': 'albumin', 'type': 'float64'},
      {'name': 'sugar', 'type': 'float64'},
      {'name': 'redbloodcells', 'type': 'float64'},
      {'name': 'puscell', 'type': 'float64'},
      {'name': 'puscellclumps', 'type': 'float64'},
      {'name': 'bacteria', 'type': 'float64'},
      {'name': 'bloodglucoserandom', 'type': 'float64'},
      {'name': 'bloodurea', 'type': 'float64'},
      {'name': 'serumcreatinine', 'type': 'float64'},
      {'name': 'sodium', 'type': 'float64'},
      {'name': 'potassium', 'type': 'float64'},
      {'name': 'haemoglobin', 'type': 'float64'},
      {'name': 'whitebloodcellcount', 'type': 'float64'},
      {'name': 'redbloodcellcount', 'type': 'float64'},
      {'name': 'hypertension', 'type': 'float64'},
      {'name': 'diabetesmellitus', 'type': 'float64'},
```

```
      {'name': 'coronaryarterydisease', 'type': 'float64'},
      {'name': 'appetite', 'type': 'float64'},
      {'name': 'pedaledema', 'type': 'float64'},
      {'name': 'anemia', 'type': 'float64'}],
...
  'name': 'ckd',
  'owner': 'IBMid-66400458ES',
  'resource_key': 'd587232d-3c2d-4949-a2f0-3abab5ef7305',
  'space_id': '4cc08e3f-9703-49f7-b9fb-c59023271670'},
 'system': {'warnings': []}}
model_id=wml_client.repository.get_model_id(model_details)
model_id
'4de3ea4c-4e9d-4b77-b461-3ce44654f34d'
#set Meta
deployment_props={

wml_client.deployments.ConfigurationMetaNames.NAME:DEPLOYMENT
_NAME,
    wml_client.deployments.ConfigurationMetaNames.ONLINE:{}
}
#Deploy
deployment=wml_client.deployments.create(
artifact_uid=model_id,
meta_props=deployment_props
)
```

##

Synchronous deployment creation for uid: '4de3ea4c-4e9d-4b77-b461-3ce44654f34d' started

##

initializing
Note: online_url is deprecated and will be removed in a future release. Use serving_urls instead.

ready

Successfully finished deployment creation, deployment_uid='3aadc0e4-27aa-434f-9875-9abaed603306'

Flask Integration with cloud Deployment

app.py

```
from flask import Flask,render_template,request
import numpy as np
import requests

# NOTE: you must manually set API_KEY below using information
retrieved from your IBM Cloud account.
API_KEY = "a-qUIpHZUwdZyhqTXKdDX9PKskdMto90F2YK-FEe1VTw"
token_response = requests.post('https://iam.cloud.ibm.com/identity/token',
data={"apikey":
 API_KEY, "grant_type": 'urn:ibm:params:oauth:grant-type:apikey'})
mltoken = token_response.json()["access_token"]

header = {'Content-Type': 'application/json', 'Authorization': 'Bearer ' +
mltoken}

app = Flask(__name__)

@app.route('/')

def formpg():
    return render_template('main.html')

@app.route('/predict', methods=['GET','POST'])
```

```
def predict():
    if request.method == 'POST':
        age =request.form.get('age')
        bp = request.form.get('bp')
        sg = request.form.get('sg')
        al = request.form.get('al')
        su = request.form.get('su')
        rbc = request.form.get('rbc')
        pc = request.form.get('pc')
        pcc = request.form.get('pcc')
        ba = request.form.get('ba')
        bgr = request.form.get('bgr')
        bu = request.form.get('bu')
        sc = request.form.get('sc')
        sod = request.form.get('sod')
        pot = request.form.get('pot')
        hemo = request.form.get('hemo')
        wc = request.form.get('wc')
        rc = request.form.get('rc')
        htn = request.form.get('htn')
        dm = request.form.get('dm')
        cad = request.form.get('cad')
        appet = request.form.get('appet')
        pe = request.form.get('pe')
        ane = request.form.get('ane')
```

```
    data =
[[int(age),float(bp),float(sg),float(al),float(su),float(rbc),float(pc),float(pcc),fl
oat(ba),float(bgr),float(bu),float(sc),float(sod),float(pot),float(hemo),float(wc
),float(rc),float(htn),float(dm),float(cad),float(appet),float(pe),float(ane)]]
    # NOTE: manually define and pass the array(s) of values to be scored in
the next line
    payload_scoring = {"input_data": [{"fields":
[age,bp,sg,al,su,rbc,pc,pcc,ba,bgr,bu,sc,sod,pot,hemo,wc,rc,htn,dm,cad,appet,
pe,ane], "values": data}]}

    response_scoring = requests.post('https://us-
south.ml.cloud.ibm.com/ml/v4/deployments/3aadc0e4-27aa-434f-9875-
9abaed603306/predictions?version=2022-11-15', json=payload_scoring,
    headers={'Authorization': 'Bearer ' + mltoken})
    print("Scoring response")
    pred=response_scoring.json()
    output=pred['predictions'][0]['values'][0][0]
    print(output)
    return render_template("result.html",prediction=output)


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

Front-end Application Development

`main.html`

```
<!DOCTYPE html>

<html lang="en" dir="ltr">

<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Chronic Kidney Disease Predictor</title>
  <link rel="stylesheet" type="text/css" href="{{ url_for('static',
filename='style.css') }}">
  <script src="https://kit.fontawesome.com/5f3f547070.js"
crossorigin="anonymous"></script>
  <link
href="https://fonts.googleapis.com/css2?family=Pacifico&display=swap"
rel="stylesheet">
</head>

<body>



  <div class="container">
    <h2 class='container-heading'><span class="heading_font">Chronic
Kidney Disease Predictor</span></h2>
    <div class='description'>
      <p>A Web Application that predicits chances of having Chronic Kidney
Disease or not, Built with the help of Flask /p><br>
```

```
        <p>(Note:This model is 98.33% accurate)</p>
    </div>
  </div>

  <!-- Text Area -->
  <div class="ml-container">
    <form action="{{ url_for('predict')}}" method="POST">


        <label for="age">Age</label>
        <input type="text" id="age" name="age" placeholder="Your
age..."><br>

        <label for="bp">Blood Pressure</label>
        <input type="text" id="bp" name="bp" placeholder="A number in range
[50-180] mmHg"><br>

        <label for="sg">Specific Gravity</label>
        <input type="text" id="sg" name="sg" placeholder="A number must be
(1.01 , 1.02 , 1.005 , 1.015 ,1.025) "><br>

        <label for="al">Albumin</label>
        <input type="text" id="al" name="al" placeholder="A number in range
0 to 5"><br>

        <label for="su">Sugar</label>
        <input type="text" id="su" name="su" placeholder="A number in range
0 to 5"><br>
```

```
<label for="rbc">Red Blood Cells</label>
<input type="text" id="rbc" name="rbc" placeholder="Normal-0 , abnormal-1"><br>

<label for="pc">Puscell</label>
<input type="text" id="pc" name="pc" placeholder="Normal-0 , abnormal-1"><br>

<label for="pcc">Puscell Clumps</label>
<input type="text" id="pcc" name="pcc" placeholder="Not Present-0 , Present-1"><br>

<label for="ba">Bacteria</label>
<input type="text" id="ba" name="ba" placeholder="Not Present-0 , Present-1"><br>

<label for="bgr">Blood Glucose Random</label>
<input type="text" id="bgr" name="bgr" placeholder="A number in range (22 to 490)"><br>

<label for="bu">Blood Urea</label>
<input type="text" id="bu" name="bu" placeholder="A number in range (1.5 to 391)"><br>

<label for="sc">Serum Creatinine</label>
<input type="text" id="sc" name="sc" placeholder="A number in range (0.4 to 76)"><br>

<label for="sod">Sodium</label>
```

```
        <input type="text" id="sod" name="sod" placeholder="A number in range (4.5 to 163)"><br>

        <label for="pot">Potassium</label>
        <input type="text" id="pot" name="pot" placeholder="A number in range (2.5 to 47)"><br>

        <label for="hemo">Hemoglobin</label>
        <input type="text" id="hemo" name="hemo" placeholder="A number in range (3.1 to 17.8)"><br>

        <label for="wc">WhiteBloodCell Count</label>
        <input type="text" id="wc" name="wc" placeholder="A number in range (3800 to 21600)"><br>

        <label for="rc">RedBloodCell Count</label>
        <input type="text" id="rc" name="rc" placeholder="A number in range (2.1 to 6.5)"><br>

        <label for="htn">Hypertension</label>
        <input type="text" id="htn" name="htn" placeholder="Yes-1 , No-0"><br>

        <label for="dm">Diabetes Mellitus</label>
        <input type="text" id="dm" name="dm" placeholder="Yes-1 , No-0"><br>

        <label for="cad">Coronary Artery Disease</label>
        <input type="text" id="cad" name="cad" placeholder="Yes-1 , No-
```

```
0"><br>

        <label for="appet">Appetite</label>
        <input type="text" id="appet" name="appet" placeholder="Good-1 ,
Poor-0"><br>

        <label for="pe">Pedaledema</label>
        <input type="text" id="pe" name="pe" placeholder="Yes-1 , No-
0"><br>

        <label for="ane">Anemia</label>
        <input type="text" id="ane" name="ane" placeholder="Yes-1 , No-
0"><br>



      <input type="submit" class="my-cta-button" value="Predict">
    </form>
  </div>

</body>

</html>
```

result.html

```
<!DOCTYPE html>

<html lang="en" dir="ltr">
<head>
```

```
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Chronic Kidney Disease Predictor</title>
<link rel="shortcut icon" href="{{ url_for('static', filename='diabetes-favicon.ico') }}">
<link rel="stylesheet" type="text/css" href="{{ url_for('static', filename='style.css') }}">
<script src="https://kit.fontawesome.com/5f3f547070.js" crossorigin="anonymous"></script>
<link href="https://fonts.googleapis.com/css2?family=Pacifico&display=swap" rel="stylesheet">
</head>

<body>

<!-- Website Title -->
<div class="container">
<h2 class='container-heading'><span class="heading_font">kidney Disease Predictor</span></h2>
<div class='description'>
<p>A Machine Learning Web App, Built with Flask</p>
</div>
</div>
<!-- Result -->
<div class="results">
{% if prediction==1 %}
<h1>Prediction: <span class='danger'>Oops! You have Chances of getting Chronic Kidney Disease.</span></h1>
```

```
{% elif prediction==0 %}
<h1>Prediction: <span class='safe'>Great! You DON'T have chances of getting Chronic Kidney Disease.</span>

{% endif %}
</div>

</body>
</html>
```

Style.css

```
html{
height: 100%;
margin: 0;
}

body{
font-family: Arial, Helvetica,sans-serif;
  text-align: center;
margin: 0;
padding: 0;
width: 100%;
height: 100%;
display: flex;
flex-direction: column;
}
```

```
/* Website Title */
.container{
    padding: 30px;
position:relative;
background: linear-gradient(45deg, #161616, #383436, #161616);
background-size: 500% 500%;
animation: change-gradient 10s ease-in-out infinite;
}
@keyframes change-gradient {
0%{
background-position: 0 50%;
}
50%{
background-position: 100% 50%;
}
100%{
background-position: 0 50%; }
}

.container-heading{
    margin: 0;
}

.heading_font{
    color: #ffffff;
font-family: 'Pacifico', cursive;
font-size: 35px;
font-weight: normal;
}
```

```
.description p{
color: #ffffff;
    font-style: italic;
    font-size: 14px;
    margin: -5px 0 0;
}

/* Text Area */
.ml-container{
    margin: 30px 0;
flex: 1 0 auto;

}

.form {
text-align: center;
width: 250px;
height: 25px;
margin-bottom: 5px;

}

input[type=text], select {
    width:60%;
    padding: 12px 20px;
    margin: 8px 0;
    display: inline-block;
    border: 1px solid #ccc;
```

```
    border-radius: 4px;
    box-sizing: border-box;
}

label {
display: inline block;
width: 200px;
font-weight: bold;
text-align: center;
float: left;

  }

/* Predict Button */
.my-cta-button{
background: #f9f9f9;
border: 2px solid #000000;
    border-radius: 1000px;
    box-shadow: 3px 3px #8c8c8c;
margin-top: 10px;
    padding: 10px 36px;
    color: #000000;
    display: inline-block;
font: italic bold 20px/1 "Calibri", sans-serif;
    text-align: center;
}

.my-cta-button:hover{
    color: #141414;
```

```
    border: 2px solid #46424b;
}

.my-cta-button:active{
    box-shadow: 0 0;
}

/* Contact */
.contact-icon{
color: #ffffff;
padding: 7px;
}

.contact-icon:hover{
color: #8c8c8c;
}

/* Footer */
.footer{
flex-shrink: 0;
position:relative;
padding: 20px;
background: linear-gradient(45deg, #161616, #383436, #161616);
background-size: 500% 500%;
animation: change-gradient 10s ease-in-out infinite;
}

.footer-description{
color: #ffffff;
```

```
margin: 0;
font-size: 12px;
}

/* Result */
.results{
   padding: 30px 0 0;
flex: 1 0 auto;
}

.danger{
   color: #ff0000;
}

.safe{
   color: green;
}
```

GitHub & Project Demo Link

github link:https://github.com/IBM-EPBL/IBM-Project-9898-1659082801

Demo link: https://drive.google.com/file/d/1XZ6gZQsDG7-HAysDLfFJ_fz9zR-cFOJB/view?usp=drivesdk