

EARLY DETECTION OF CHRONIC KIDNEY DISEASE USING MACHINE LEARNING

PROJECT REPORT

Submitted by

Sudharsana Suresh

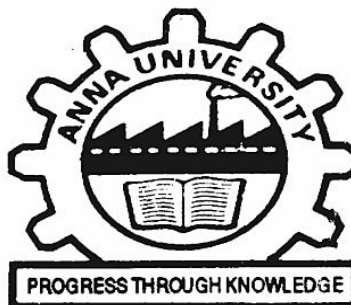
Naveen U

Geethanjali B

Nithik N

Bharath B

(PNT2022TMID36039)



**Department of Information Technology
Madras Institute of Technology
Anna University, Chennai- 600 044**

CONTENTS

CHAPTER NO	TITTLE	PAGE NO
1	INTRODUCTION	
1.1	Project Overview	4
1.2	Purpose	4
2	LITERATURE SURVEY	
2.1	Existing problem	5
2.2	References	5
2.3	Problem Statement Definition	7
3	IDEATION & PROPOSED SOLUTION	
3.1	Empathy Map Canvas	8
3.2	Ideation & Brainstorming	9
3.3	Proposed Solution	9
3.4	Problem Solution fit	12
4	REQUIREMENT ANALYSIS	
4.1	Functional requirement	14
4.2	Non-Functional requirements	14
5	PROJECT DESIGN	
5.1	Data Flow Diagram	15
5.2	Solution & Technical Architecture	16
5.3	User Stories	17
6	PROJECT PLANNING & SCHEDULING	
6.1	Sprint Planning & Estimation	18
6.2	Sprint Delivery Schedule	18
6.3	Reports from JIRA	19

7	CODING & SOLUTIONING	
7.1	Feature 1	20
7.2	Feature 2	29
7.3	Database Schema	29
8	TESTING	
8.1	Test Cases	30
8.2	User Acceptance Testing	30
9	RESULTS	
9.1	Deployment in IBM Cloud	31
9.2	Prediction Results	32
9.3	Performance Metrics	33
10	CONCLUSION	35
11	FUTURE SCOPE	35
12	APPENDIX	36
	Source Code	
	GitHub & Project Demo Link	

CHAPTER 1

INTRODUCTION

1.1 Project Overview

Prediction of chronic kidney disease is one of the most crucial challenges in healthcare analytics. The most fascinating and difficult jobs in daily life because millions of people die each year due to lack of access to inexpensive treatment for chronic kidney disease (CKD), which affects one third of the adult population. If chronic kidney disease is addressed early on, it may be cured. The major goal of the project is to use diagnostic measurements like albumin and blood pressure to quickly, accurately, and painlessly determine if a patient has chronic kidney disease or not. Based on the information provided by the model, suitable treatment can then be administered.

1.2 Purpose of the Project

The project's goal is to warn medical professionals of kidney illness early on, ensuring a quick recovery or kidney disease prevention. The goal of this project is to use machine learning to develop a model for the early diagnosis of chronic kidney disease. Integration of the output into Flask is present. The user input on the numerous factors required to decide on the early identification of renal illness is collected through the front end designed in HTML. The IBM cloud is using the similar model

CHAPTER 2

LITREATURE SURVEY

2.1 Existing problem

In many countries today, kidney disease is discovered in its late stages, resulting in the loss of precious lives. There aren't many ways to spot them in the beginning. The majority of user information is still unconfirmed, making it challenging to identify bogus users. The application's user interface is not intuitive, and in order to engage with it, a user needs a device running the Android operating system and a working internet connection.

2.2 References

- [1] B. Gudeti, S. Mishra, S. Malik, T. F. Fernandez, A. K. Tyagi and S. Kumari, "A Novel Approach to Predict Chronic Kidney Disease using Machine Learning Algorithms," 2020 4th International Conference on Electronics, Communication and Aerospace Technology (ICECA), 2020, pp. 1630-1635, doi: 10.1109/ICECA49313.2020.9297392.
- [2] A. Charleonnann, T. Fufaung, T. Niyomwong, W. Chokchueypattanakit, S. Suwannawach and N. Ninchawee, "Predictive analytics for chronic kidney disease using machine learning techniques," 2016 Management and Innovation Technology International Conference (MITicon), 2016, pp. MIT-80-MIT-83, doi: 10.1109/MITICON.2016.8025242.
- [3] A. J. Aljaaf et al., "Early Prediction of Chronic Kidney Disease Using Machine Learning Supported by Predictive Analytics," 2018 IEEE Congress on Evolutionary Computation (CEC), 2018, pp. 1-9, doi: 10.1109/CEC.2018.8477876.
- [4] R. A. Alassaf et al., "Pre-emptive Diagnosis of Chronic Kidney Disease Using

Machine Learning Techniques," 2018 International Conference on Innovations in Information Technology (IIT), 2018, pp. 99-104, doi: 10.1109/INNOVATIONS.2018.8606040.

[5] A. Sobrinho, A. C. M. D. S. Queiroz, L. Dias Da Silva, E. De Barros Costa, M. Eliete Pinheiro and A. Perkusich, "Computer-Aided Diagnosis of Chronic Kidney Disease in Developing Countries: A Comparative Analysis of Machine Learning Techniques," in IEEE Access, vol. 8, pp. 25407-25419, 2020, doi: 10.1109/ACCESS.2020.2971208.

[6] Y. Amirgaliyev, S. Shamiluulu and A. Serek, "Analysis of Chronic Kidney Disease Dataset by Applying Machine Learning Methods," 2018 IEEE 12th International Conference on Application of Information and Communication Technologies (AICT), 2018, pp. 1-4, doi: 10.1109/ICAICT.2018.8747140.

[7] J. Qin, L. Chen, Y. Liu, C. Liu, C. Feng and B. Chen, "A Machine Learning Methodology for Diagnosing Chronic Kidney Disease," in IEEE Access, vol. 8, pp. 20991-21002, 2020, doi: 10.1109/ACCESS.2019.2963053.

[8] M. U. Emon, A. M. Imran, R. Islam, M. S. Keya, R. Zannat and Ohidujjaman, "Performance Analysis of Chronic Kidney Disease through Machine Learning Approaches," 2021 6th International Conference on Inventive Computation Technologies (ICICT), 2021, pp. 713-719, doi: 10.1109/ICICT50816.2021.9358491.

[9] T. M. Rahman, S. Siddiqua, S. E. Rabby, N. Hasan and M. H. Imam, "Early Detection of Kidney Disease Using ECG Signals Through Machine Learning Based Modelling," 2019 International Conference on Robotics,Electrical and Signal Processing Techniques (ICREST), 2019, pp. 319-323, doi: 10.1109/ICREST.2019.8644354.

[10] W. H. S. D. Gunarathne, K. D. M. Perera and K. A. D. C. P.

Kahandawaarachchi, "Performance Evaluation on Machine Learning Classification Techniques for Disease Classification and Forecasting through Data Analytics for Chronic Kidney Disease (CKD)," 2017 IEEE 17th International Conference on Bioinformatics and Bioengineering (BIBE), 2017, pp. 291-296, doi: 10.1109/BIBE.2017.00-39.

2.3 Problem Statement

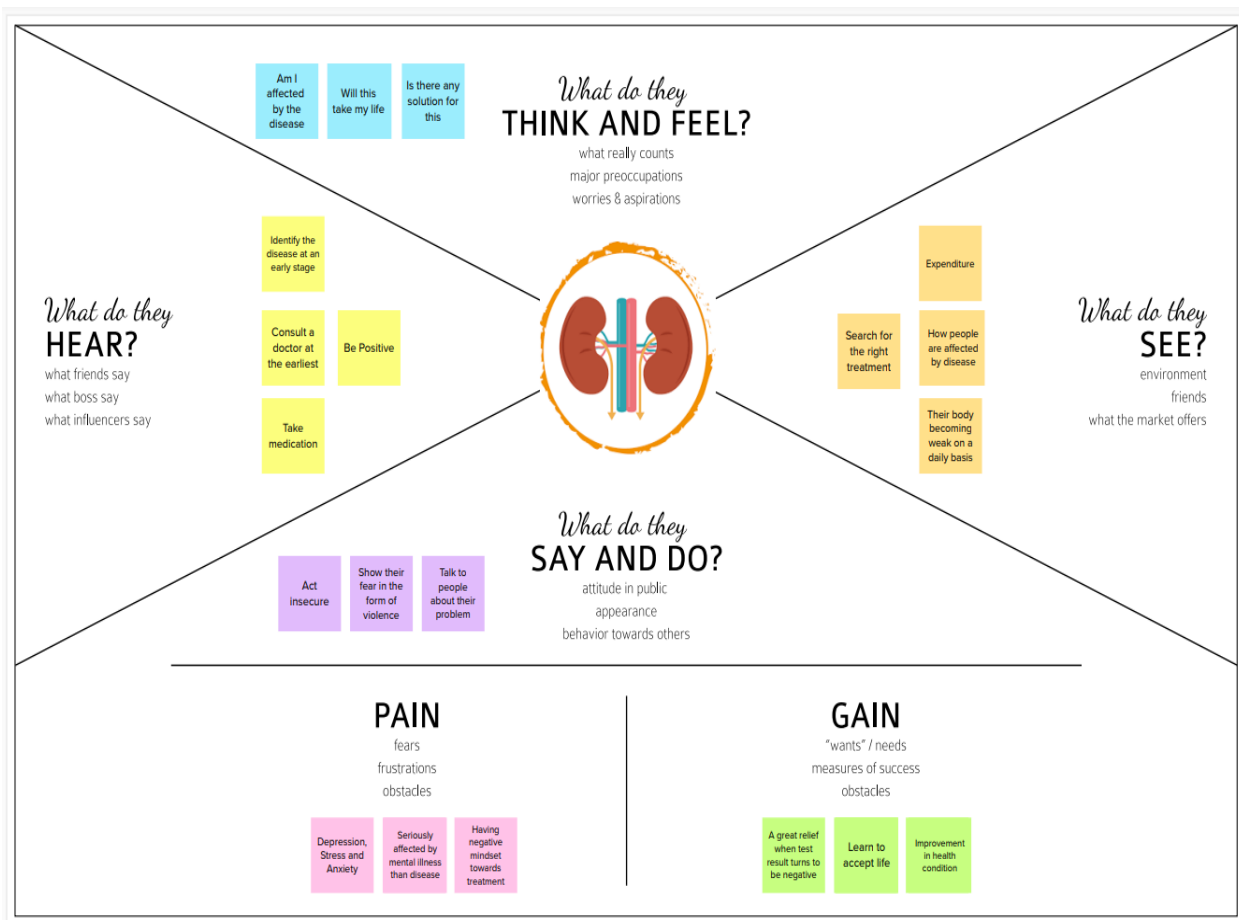
Chronic Kidney Disease (CKD) is a dangerous illness that, if detected in time, can be cured. Most people aren't aware that the many medical tests we have for different reasons could tell us crucial things about kidney diseases. In order to determine whether aspects of a variety of medical tests might contain information that is relevant to the disease, they are examined. The research suggests that doing so allows us to determine the severity of the issue, and we use this knowledge to develop a machine learning model that predicts chronic kidney disease. Early treatment of chronic kidney disease may result in a cure. With the aid of diagnostic data such as Blood Pressure (Bp), Albumin, and other factors, the main objective of this study is to more rapidly and reliably determine whether a patient has chronic kidney disease (Al).

CHAPTER 3

IDEATION & PROPOSED SOLUTION

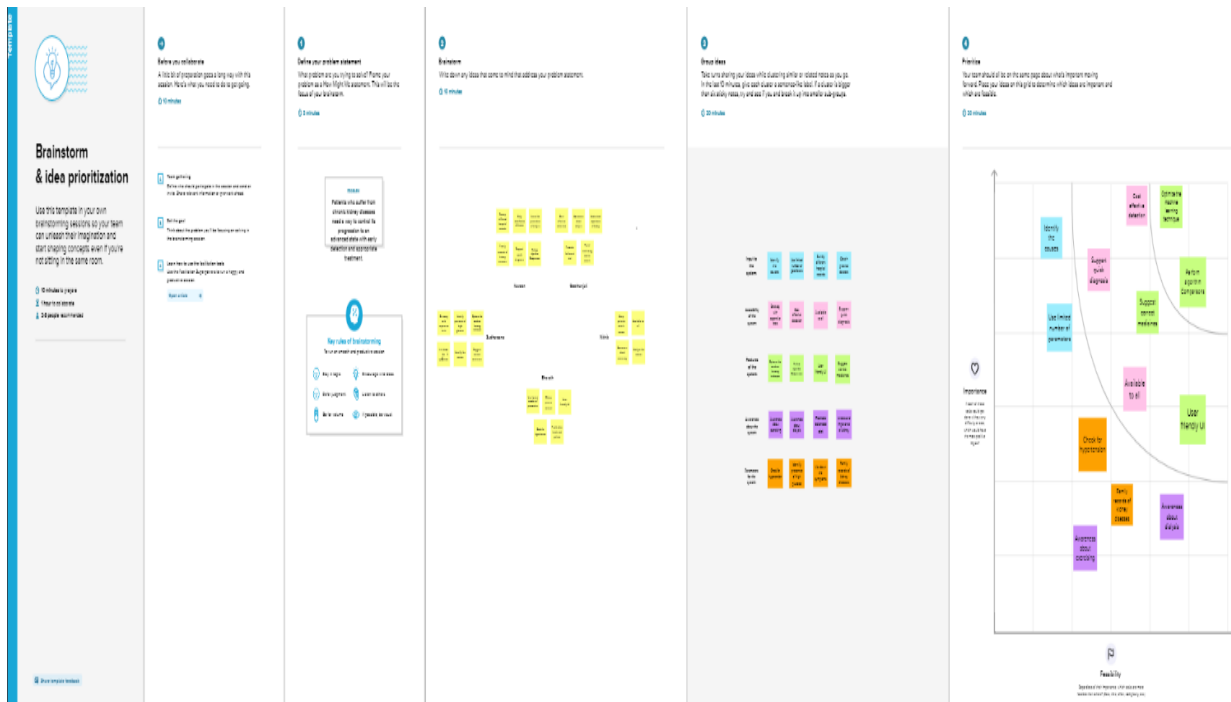
3.1 Empathy map

Empathy maps are a helpful tool used by designers to not only evaluate user behaviour but also to visually communicate their findings to colleagues, uniting the team around a shared understanding of the user. Empathy maps work best when utilised early in the design process in user-centered design.



3.2 Ideation & Brain Storming

Ideation is a general term that refers to the process of coming up with and expressing new ideas. It is an imaginative thought that seeks to resolve a dilemma or offer a more effective means of carrying out an action. It includes creating fresh concepts, upgrading existing ones, and figuring out how to put fresh concepts into action. Brainstorming is the most frequently practiced form of ideation. The intention of brainstorming is to leverage the collective thinking of the group, by engaging with each other, listening, and building on other ideas.



3.3 Proposed Solution

S.No	Parameter	Description

1	Problem Statement (Problem to be solved)	Patients who suffer from chronic kidney diseases need a way to control its progression to an advanced state with early detection and appropriate treatment. Machine learning has advanced to the point that it is now possible to look through patient medical information and identify chronic kidney disease in its early stages.
2	Idea / Solution description	Since certain data are missing, the initial step is to perform pre-processing by cleaning the dataset, along with scaling and normalisation of values. The next step is to use dimensionality reduction to identify the key features in the dataset and to remove any irrelevant ones. To accomplish early detection of chronic kidney disease utilising the indicated key traits, a decision tree model must be fitted.
3	Novelty / Uniqueness	<ul style="list-style-type: none"> • An indicator of how well the kidneys is working is the amount of a waste product called creatinine in the blood. By examining this data, early kidney disease can be identified by detecting deviations from the norm. • In the case of healthcare

		management products, it is especially important to have a UI that is very user-friendly and open to everyone.
4	Social Impact / Customer Satisfaction	The primary goal of this application is early prediction, and appropriate treatments may be able to prevent or delay the disease's progression to an advanced state.
5	Business Model (Revenue Model)	<ul style="list-style-type: none"> ● The suggested strategy has the potential to generate income from direct patients as payment for the development of immediate outcomes. ● It can also collaborate with the healthcare sector to generate revenue from patients who come in for kidney disease diagnosis.
6	Scalability of the Solution	<ul style="list-style-type: none"> ● The dimensionality reduction process can be adjusted to produce precise predictions with an increase in the features taken into account. ● The accuracy of many models can be compared in order to determine which is best. ● It can be used for a variety of illnesses in addition to chronic disorders.

3.4 Problem Solution Fit

Problem-Solution Fit happens when there is proof that customers are interested in particular tasks, challenges, and benefits. You've established that a problem exists and created a value offer that takes into account the tasks, challenges, and gains of your clients at this point. A problem-solution-fit occurs when such a solution is discovered and a business develops a strategy that, from a variety of angles, offers a game changer for customers. However, if businesses miss evaluating the Problem-Solution Fit they developed, they face a risk of finding that no one wants their solution, which is unfortunate considering the effort and money invested.

Define CS, fit into CC	1. CUSTOMER SEGMENT(S) CS Patients that face mild to severe symptoms ranging from unusual fatigue, high blood pressure, malaise to insufficient urine production, high levels of creatinine, kidney failure; that maybe an indication of a serious health issue like chronic kidney disease prediction.	6. CUSTOMER CONSTRAINTS CC i. Although free, the web program works on computers, smartphones, and other electronic gadgets, which may be out of reach for the less fortunate members of the society. ii. Requires recent blood/urine test results, making this a requirement for the machine learning model before it can offer a forecast.	5. AVAILABLE SOLUTIONS AS The primary treatments are lifestyle modifications to keep you as healthy as possible, medication to manage related issues like high blood pressure and high cholesterol, and dialysis. None of these options focuses on early kidney disease detection using data from specific human body testing. All primary therapies may be avoided by quickly completing an early diagnostic.	Explore AS, differentiate
	2. JOBS-TO-BE-DONE / PROBLEMS J&P The following jobs are to be done: i. Identify the most important diagnostic data that can cause chronic kidney disease ii. Create an ML model that can predict the presence of chronic kidney disease iii. Design an interactive, simple and freely available UI for communicating with the patients.	9. PROBLEM ROOT CAUSE RC Kidney disease is most frequently brought on by diabetes. However, obesity and heart disease can also contribute to the harm that results in renal failure. Long-term functional decline can also be brought on by problems with the urinary system and inflammation in various kidney regions.	7. BEHAVIOUR BE First, it is assumed that the patient would undergo a few tests and provide the required results as input to the frontend of the created system. Based on this data, the machine learning model predicts the future. The fact that the application is free to use makes it incredibly beneficial to users.	
Identify strong TR & EM	3. TRIGGERS TR Patients are encouraged to get a kidney function test if they experience symptoms that point to potential renal issues. These signs and symptoms may include: unusual nausea and vomiting; blood in urine (hematuria) and painful urination (dysuria).	10. YOUR SOLUTION SL Patients with chronic kidney disease require a means to prevent its development into a severe condition by early detection and effective treatment. With the advancement of machine learning, it is now able to search through patient medical records and spot chronic kidney disease in its early stages. The system successfully resolves the aforementioned issue without charging a fee by combining the machine learning model with an intuitive UI.	8. CHANNELS of BEHAVIOUR CH 8.1. ONLINE In order for the machine learning model to produce predictions, the patients are required to provide the appropriate health check test results into the online application. 8.2. OFFLINE In order to complete the required health examination, patients must visit laboratories or hospitals, from which the information can be entered into the web application.	Identify strong TR & EM
	4. EMOTIONS: BEFORE / AFTER EM Patients experience a rush of terror prior to interacting with the suggested system. They will feel relieved and acquire a diagnosis after seeing the results.			

CHAPTER 4

REQUIREMENT ANALYSIS

4.1 Functional Requirements

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Registration	Registration through Form for new users
FR-2	User Login	Login through credentials for existing users
FR-3	User Requirements	Store past records Generate report for presence of CKD Diagnostic remedies for symptoms
FR-4	User Entry	Input form for pre-diagnostic test results
FR-5	Business Requirements	Quick diagnosis for CKD
FR-6	User Feedback	Allows users to submit feedback through a form

4.2 Non Functional Requirements

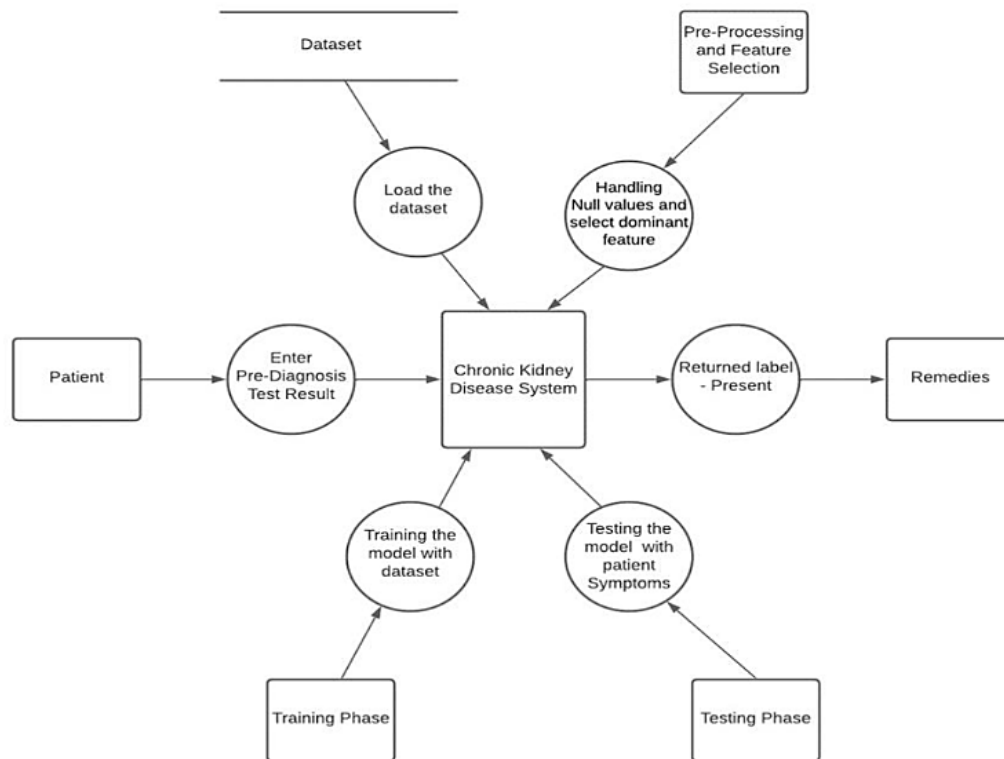
FR No.	Non-Functional Requirement	Description
NFR-1	Usability	Simple user-friendly interface for communication
NFR-2	Security	Safeguard the details shared by users and maintain confidentiality
NFR-3	Reliability	Diagnosis based on probability predicted by ML model must be reliable
NFR-4	Performance	Reduction in overall time taken for diagnosis
NFR-5	Availability	Available at any time to users from various places
NFR-6	Scalability	Needs to support numerous users at once

CHAPTER 5

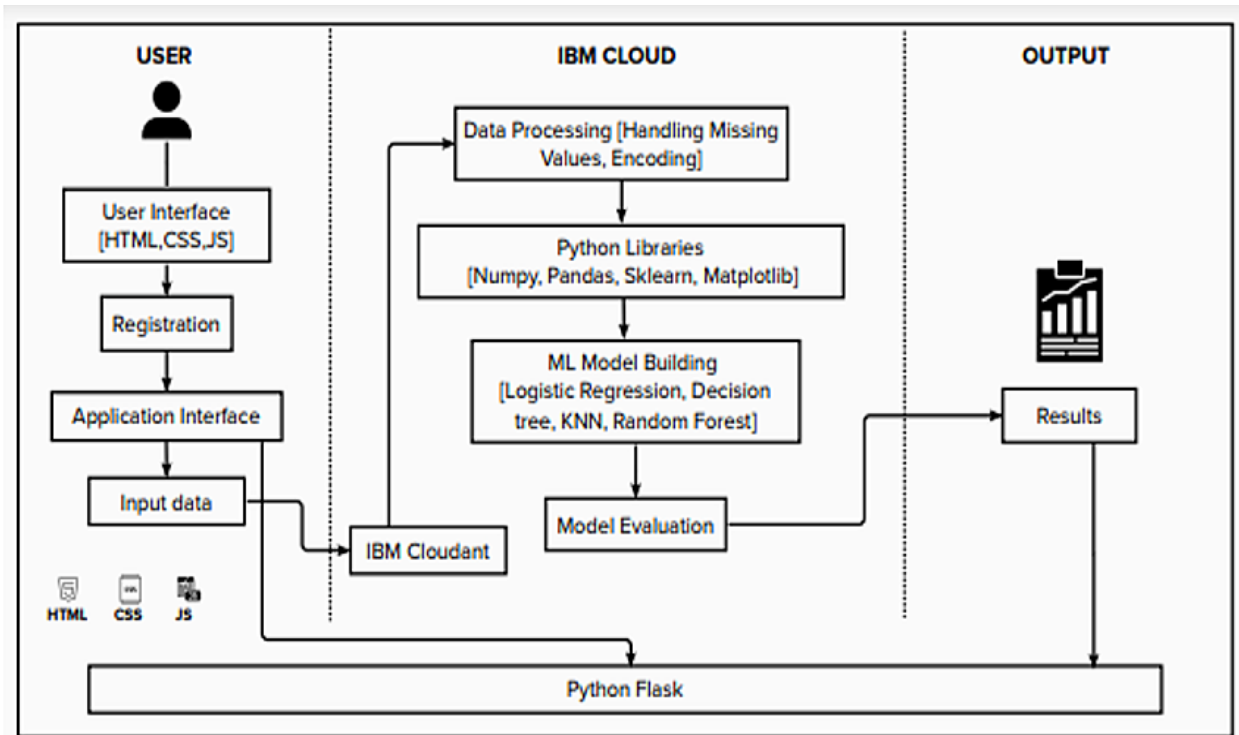
PROJECT DESIGN

5.1 Data Flow Diagram

A data flow diagram is a visualization tool used to illustrate the flow of processes in a company or a specific project within it. It highlights the movement of information as well as the sequence of steps or events required to complete a work task.



5.2 Solution & Technical Architecture



S. No	Component	Description	Technology
1.	User Interface	An Interface for the user to interact with the prediction model.	HTML, CSS, JavaScript
2.	User Registration	User can register in the web application	HTML forms
3.	Disease Prediction	The user enters the data which is given as input to model to predict the disease.	Machine Learning with Python.
4.	Update Prediction result	The result of disease prediction is updated in the Web UI for the user to know the output.	Python.
5.	Database	Relational database structure to store the user data	MYSQL.
6.	Cloud Database	Database services on IBM cloud.	IBM Cloudant.
7.	Machine Learning Model	To predict the chronic kidney disease (CKD) with various input parameters.	Random Forest, KNN, Decision tree, Logistic Regression.
8.	Infrastructure (Server / Cloud)	Application Deployment on Cloud	IBM Cloud.

Application Characteristics

S. No	Characteristics	Description	Technology
1.	Open-Source Frameworks	The python open-source frameworks are used to build the web application as well as to build Machine Learning model.	Python Flask, NumPy, Scikit-Learn etc.
2.	Scalable Architecture	The 3-tier architecture used with a separate user interface, application tier and data tier make it easily scalable.	IBM Watson Studio.
3.	Availability	The web application is highly available as it is deployed in cloud.	IBM Cloud.
4.	Performance	The performance of the website is improved with caching and security.	IBM Cloud Internet Services.

5.3 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 diagnosis tool using my email and password	I can access my account / dashboard	High	Sprint-1
		USN-2	As a user, I will receive confirmation email on registering for the diagnosis tool	I will receive confirmation email	High	Sprint-1
		USN-3	As a user, I can register for the application through my Gmail	I can register and access the dashboard with my Gmail Login	Low	Sprint-4
	Login	USN-4	As a user, I can log into the application by entering my credentials	I can login and access past records	High	Sprint-1
	Dashboard	USN-5	As a user, I can see my past records and activities	I can access the functionalities diagnosing tool	High	Sprint-3
	Entry form	USN-6	As a user, I must enter my pre-diagnostic test results	I can use the form to input test results	High	Sprint-2
	Report	USN-7	As a user, I can view the report generated by the tool	I can view negative/ positive results produced after diagnosis	High	Sprint-3
	Remedies	USN-8	As a user, I will receive remedies to treat my symptoms	I can cure my symptoms with the remedies suggested	Medium	Sprint-3
Customer Care Executive	Queries	USN-9	As a customer care executive, I must assist users that face problems through Q&A	I will provide 24/7 support for the tool	Low	Sprint-4
	Feedback	USN-10	As a customer care executive, I should get input for the tool's enhancement from users	I must work on improving tool's performance	Low	Sprint-4
Administrator	Feature importance	USN-11	As an administrator, I should identify the most significant factors that lead to CKD based on the present trend	I must identify important features	High	Sprint-2
	Train model	USN-12	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

CHAPTER 6

PROJECT PLANNING AND SCHEDULING


6.1 Sprint Planning and Estimation

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Registration	USN-1	As a user, I can register for the diagnosis tool using my email and password	7	High	Team Lead Member 1
Sprint-1		USN-2	As a user, I will receive confirmation email on registering for the diagnosis tool	6	High	Member 1 Member 4
Sprint-4		USN-3	As a user, I can register for the application through my Gmail	6	Low	Member 2 Member 3
Sprint-1	Login	USN-4	As a user, I can log into the application by entering my credentials	6	High	Team Lead Member 1
Sprint-3	Dashboard	USN-5	As a user, I can see my past records and activities	6	High	Team Lead Member 3
Sprint-2	Entry form	USN-6	As a user, I must enter my pre-diagnostic test results	7	High	Team Lead Member 1
Sprint-3	Report	USN-7	As a user, I can view the report generated by the tool	7	High	Member 2 Team Lead
Sprint-3	Remedies	USN-8	As a user, I will receive remedies to treat my symptoms	6	Medium	Member 1 Member 2
Sprint-4	Queries	USN-9	As a customer care executive, I must assist users that face problems through Q&A	6	Low	Member 2 Member 3
Sprint-4	Feedback	USN-10	As a customer care executive, I should get input for the tool's enhancement from users	7	Low	Member 3 Member 4
Sprint-2	Feature importance	USN-11	As an administrator, I should identify the most significant factors that lead to CKD based on the present trend	6	High	Member 2 Member 4
Sprint-2	Train model	USN-12	As an administrator, I must use the most suitable ML model for detection of CKD	6	High	Team Lead Member 4

6.2 Sprint Delivery Schedule

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	29 Oct 2022
Sprint-2	20	6 Days	31 Oct 2022	05 Nov 2022	20	05 Nov 2022
Sprint-3	20	6 Days	07 Nov 2022	12 Nov 2022	20	12 Nov 2022
Sprint-4	20	6 Days	14 Nov 2022	19 Nov 2022	20	19 Nov 2022

6.3 Reports From JIRA

 Jira Software

Your work ▾

Projects ▾


Filters ▾

Dashboards ▾


People ▾


Apps ▾

Create


 CKD predictor
Software project


PLANNING


 Roadmap

 Board

DEVELOPMENT

 Project pages


 Add shortcut


 Project settings

You're in a team-managed project
[Learn more](#)

Projects / CKD predictor

CP





TO DO 7 ISSUES

USN-3 Register using Gmail account
☒ CP-3

USN-5 View Dashboard
☒ CP-5

USN-7 View diagnosis report
☒ CP-7

USN-8 Remedies for diagnosis
☒ CP-8

USN-9 Customer Q&A
☒ CP-9

USN-10 Customer feedback
☒ CP-10

USN-12 Select optimal model
☒ CP-12

IN PROGRESS 3 ISSUES

USN-11 Feature selection for model
☒ CP-11


USN-2 Confirmation Email on registration
☒ CP-2

USN-6 Entry form for pre-diagnostic test
☒ CP-6

DONE 2 ISSUES ✓

USN-1 Registration Page
☒ CP-1 ✓

USN-4 Login page for registered users
☒ CP-4 ✓



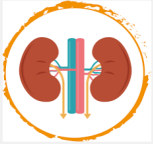
CHAPTER 7

CODING AND SOLUTIONING

7.1 Feature 1: Pre-diagnostic Form

The user can enter the results of the pre-diagnostic tests that have previously been performed on this page. This pre-diagnosis form contains the top ten characteristics determined by feature selection using Extra Tree Classifier on the dataset including 24 different features. The model predicts the likelihood of chronic kidney disease based on the values input.

[Home](#) [Contact](#) [About](#)



Chronic Kidney Disease Prediction

CKD Prediction

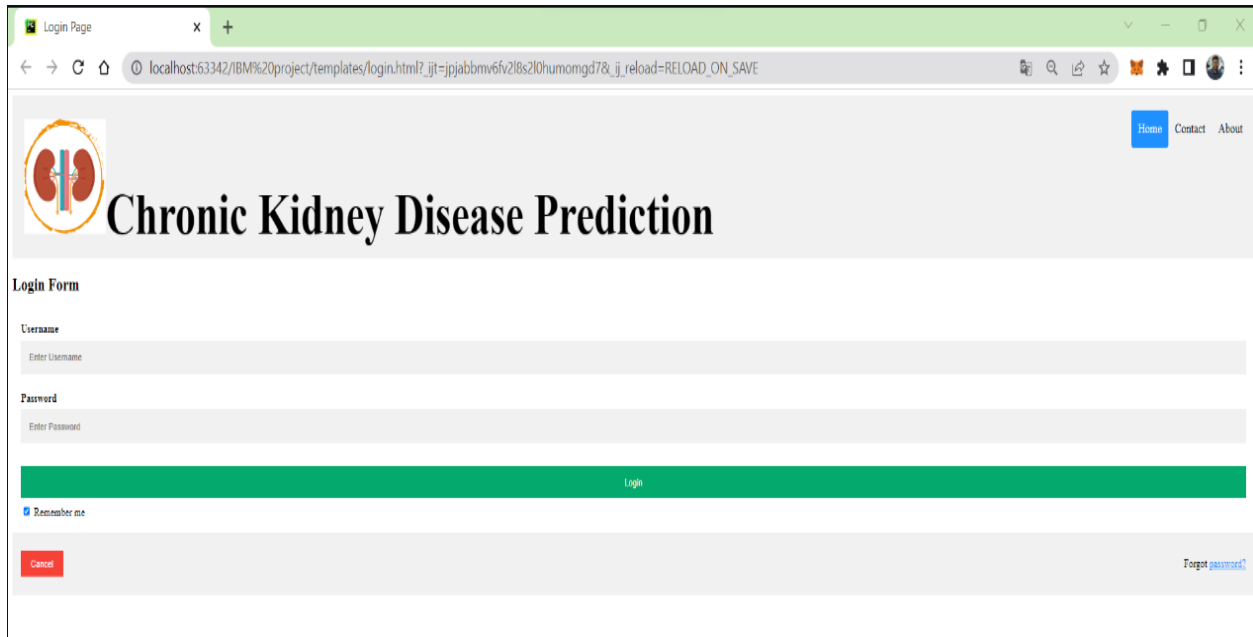
Please fill in this form for generating report

Name	<input type="text" value="sudhu"/>
Specific Gravity	<input type="text" value="1.020"/>
Sugar Level (0-5)	<input type="text" value="4"/>
Albumin (0-4)	<input type="text" value="2"/>
Albumin (0-4)	<input type="text" value=""/>
Potassium (in mEq/L)	<input type="text" value="4.4"/>
Haemoglobin	<input type="text" value="15.4"/>
Packed Cell Volume	<input type="text" value="39"/>
Appetite	<input type="text" value="Good"/>
Hypertension	<input type="text" value="Yes"/>
Diabetes Mellitus	<input type="text" value="Yes"/>
Anemia	<input type="text" value="Yes"/>
<input type="button" value="Predict"/>	

7.1.1 Login & Registration Form

Login Page

localhost:63342/IBM%20project/templates/login.html?_ijt=jpjabbm6fv2l8s2l0humomgd78&_ij_reload=RELOAD_ON_SAVE



Home Contact About

Chronic Kidney Disease Prediction

Login Form

Username
Enter Username

Password
Enter Password

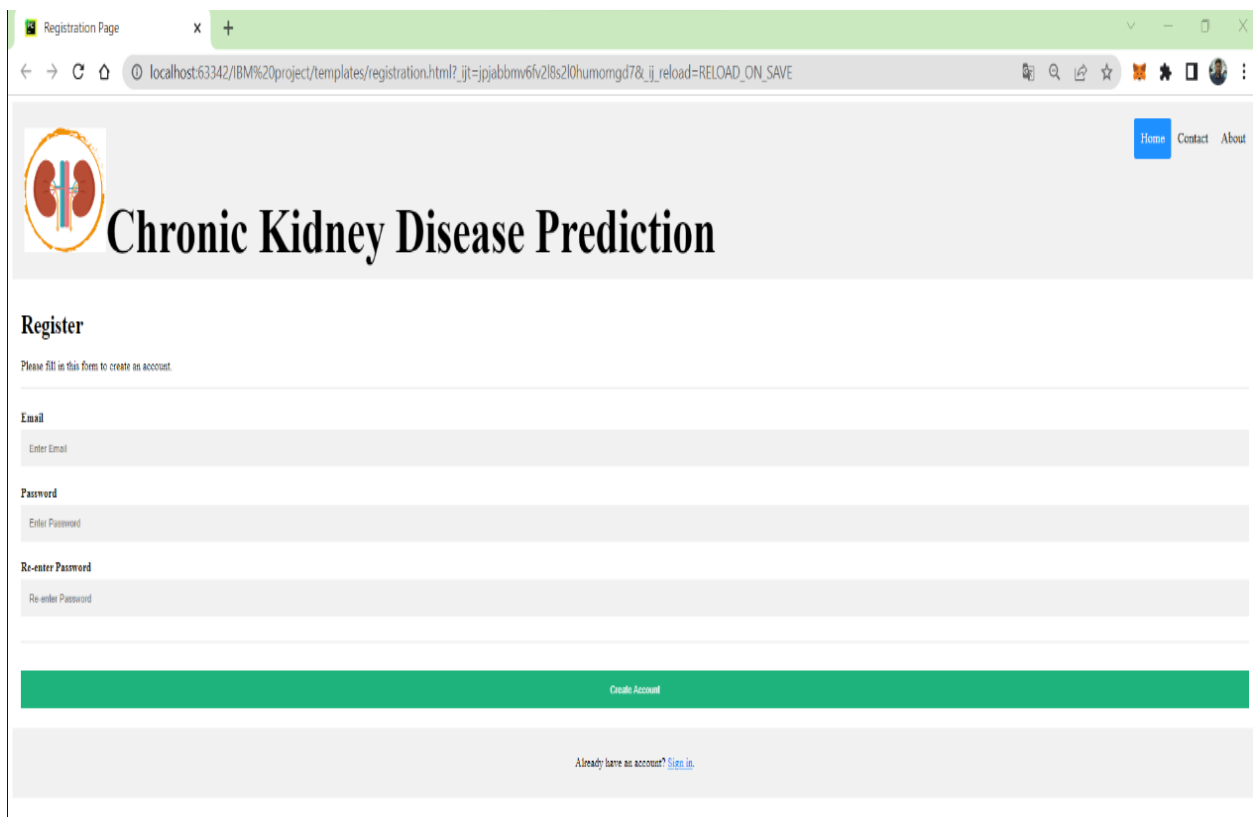
Login

☒ Remember me

Cancel [Forgot password?](#)

Registration Page

localhost:63342/IBM%20project/templates/registration.html?_ijt=jpjabbm6fv2l8s2l0humomgd78&_ij_reload=RELOAD_ON_SAVE



Home Contact About

Chronic Kidney Disease Prediction

Register

Please fill in this form to create an account.

Email
Enter Email

Password
Enter Password

Re-enter Password
Re-enter Password

Create Account

Already have an account? [Sign in](#)

7.1.2 Methodologies

7.1.2.1 Dataset Information

The dataset contains 26 features and 401 records. The data columns are of the type float,int and String. The features that are present in the data set include:

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

```
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')
```

7.1.2.2 Data Cleaning

The selected dataset was examined. By collecting the dataset's feature summary, it was possible to comprehend the various data kinds that were there. Floating values were specifically converted from numerical columns. The most commonly present value, or mode, was used to fill in both the numerical and category columns with missing data. The category columns were then subjected to label encoding, which made it easier for the machine learning model to function.

```
#filling missing values with mode for categorical attributes
df['rbc'].fillna('normal',inplace=True)
df['pc'].fillna('normal',inplace=True)
df['pcc'].fillna('notpresent',inplace=True)
df['ba'].fillna('notpresent',inplace=True)
df['htn'].fillna('no',inplace=True)
df['dm'].fillna('no',inplace=True)
df['cad'].fillna('no',inplace=True)
df['appet'].fillna('good',inplace=True)
df['pe'].fillna('no',inplace=True)
df['ane'].fillna('no',inplace=True)
```

```
#filling missing values with median for numerical attributes
for col in num_cols:
    df[col]=df[col].fillna(df[col].median())
```

```
#label encoding for target class
df['classification']=df['classification'].map({'ckd':1,'notckd':0})
```

The dataset is divided into 80:20 ratio where 80% is for training data and 20% for testing data.

```
X_train,X_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=0)
print("X_train size {} , X_test size {}".format(X_train.shape,X_test.shape))
```

7.1.2.3 Model Building

Gradient Boosting Classifier

```
from sklearn.ensemble import GradientBoostingClassifier
gb = GradientBoostingClassifier()
gb.fit(X_train, y_train)

# accuracy score, confusion matrix and classification report of gradient boosting classifier
gb_acc = accuracy_score(y_test, gb.predict(X_test))

print(f"Training Accuracy of Gradient Boosting Classifier is {accuracy_score(y_train, gb.predict(X_train))}")
print(f"Test Accuracy of Gradient Boosting Classifier is {gb_acc*100} \n")
```

Accuracy

```
Training Accuracy of Gradient Boosting Classifier is 1.0
Test Accuracy of Gradient Boosting Classifier is 96.66666666666667
```

Stochastic Gradient Boosting

```
sgb = GradientBoostingClassifier(max_depth = 4, subsample = 0.90, max_features = 0.75, n_estimators = 200)
sgb.fit(X_train, y_train)

# accuracy score, confusion matrix and classification report of stochastic gradient boosting classifier

sgb_acc = accuracy_score(y_test, sgb.predict(X_test))

print(f"Training Accuracy of Stochastic Gradient Boosting is {accuracy_score(y_train, sgb.predict(X_train))*100}")
print(f"Test Accuracy of Stochastic Gradient Boosting is {sgb_acc*100} \n")
```

Accuracy

```
Training Accuracy of Stochastic Gradient Boosting is 100.0
Test Accuracy of Stochastic Gradient Boosting is 96.66666666666667
```


Extra Tree Classifier

```
# accuracy score, confusion matrix and classification report of extra tree classifier
etc = ExtraTreesClassifier()
etc.fit(X_train, y_train)

etc_acc = accuracy_score(y_test, etc.predict(X_test))

print(f"Training Accuracy of Extra Trees Classifier is {accuracy_score(y_train, etc.predict(X_train))*100}")
print(f"Test Accuracy of Extra Trees Classifier is {etc_acc*100} \n")

print(f"Confusion Matrix :- \n{confusion_matrix(y_test, etc.predict(X_test))}\n")
print(f"Classification Report :- \n {classification_report(y_test, etc.predict(X_test))}")
```

Accuracy

Training Accuracy of Extra Trees Classifier is 100.0
Test Accuracy of Extra Trees Classifier is 99.16666666666667

K-Nearest Neighbour

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report

knn = KNeighborsClassifier()
knn.fit(X_train, y_train)

# accuracy score, confusion matrix and classification report of knn
knn_acc = accuracy_score(y_test, knn.predict(X_test))
print(f"Training Accuracy of KNN is {accuracy_score(y_train, knn.predict(X_train))*100}")
print(f"Test Accuracy of KNN is {knn_acc*100} \n")

print(f"Confusion Matrix :- \n{confusion_matrix(y_test, knn.predict(X_test))}\n")
print(f"Classification Report :- \n {classification_report(y_test, knn.predict(X_test))}")
```

Accuracy

Training Accuracy of KNN is 94.64285714285714
Test Accuracy of KNN is 91.66666666666666

Decision Tree Classifier

```
from sklearn.tree import DecisionTreeClassifier

dtc = DecisionTreeClassifier()
dtc.fit(X_train, y_train)

# accuracy score, confusion matrix and classification report of decision tree

dtc_acc = accuracy_score(y_test, dtc.predict(X_test))

print(f"Training Accuracy of Decision Tree Classifier is {accuracy_score(y_train, dtc.predict(X_train))*100}")
print(f"Test Accuracy of Decision Tree Classifier is {dtc_acc*100} \n")
```

Accuracy

```
Training Accuracy of Decision Tree Classifier is 100.0
Test Accuracy of Decision Tree Classifier is 95.0
```

Random Forest Classifier

```
from sklearn.ensemble import RandomForestClassifier

rd_clf = RandomForestClassifier(criterion = 'entropy', max_depth = 11, max_features = 'auto', min_samples_leaf = 2, min_samples_split = 3, n_estimator
rd_clf.fit(X_train, y_train)

# accuracy score, confusion matrix and classification report of random forest

rd_clf_acc = accuracy_score(y_test, rd_clf.predict(X_test))

print(f"Training Accuracy of Random Forest Classifier is {accuracy_score(y_train, rd_clf.predict(X_train))*100}")
print(f"Test Accuracy of Random Forest Classifier is {rd_clf_acc*100} \n")
```

Accuracy

```
Training Accuracy of Random Forest Classifier is 99.28571428571429
Test Accuracy of Random Forest Classifier is 96.66666666666667
```

Ada Boost Classifier

```
from sklearn.ensemble import AdaBoostClassifier

ada = AdaBoostClassifier(base_estimator = dtc)
ada.fit(X_train, y_train)

# accuracy score, confusion matrix and classification report of ada boost

ada_acc = accuracy_score(y_test, ada.predict(X_test))

print(f"Training Accuracy of Ada Boost Classifier is {accuracy_score(y_train, ada.predict(X_train))*100}")
print(f"Test Accuracy of Ada Boost Classifier is {ada_acc*100} \n")
```

Accuracy

```
Training Accuracy of Ada Boost Classifier is 100.0
Test Accuracy of Ada Boost Classifier is 97.5
```

XGBoost

```
from xgboost import XGBClassifier

xgb = XGBClassifier(objective = 'binary:logistic', learning_rate = 0.5, max_depth = 5, n_estimators = 150)
xgb.fit(X_train, y_train)

# accuracy score, confusion matrix and classification report of xgboost

xgb_acc = accuracy_score(y_test, xgb.predict(X_test))

print(f"Training Accuracy of XgBoost is {accuracy_score(y_train, xgb.predict(X_train))*100}")
print(f"Test Accuracy of XgBoost is {xgb_acc*100} \n")
```

Accuracy

```
Training Accuracy of XgBoost is 100.0
Test Accuracy of XgBoost is 96.66666666666667
```

LGBM Classifiers

```
from lightgbm import LGBMClassifier

lgbm = LGBMClassifier(learning_rate = 1)
lgbm.fit(X_train, y_train)

# accuracy score, confusion matrix and classification report of lgbm classifier

lgbm_acc = accuracy_score(y_test, lgbm.predict(X_test))

print(f"Training Accuracy of LGBM Classifier is {accuracy_score(y_train, lgbm.predict(X_train))*100}")
print(f"Test Accuracy of LGBM Classifier is {lgbm_acc*100} \n")
```

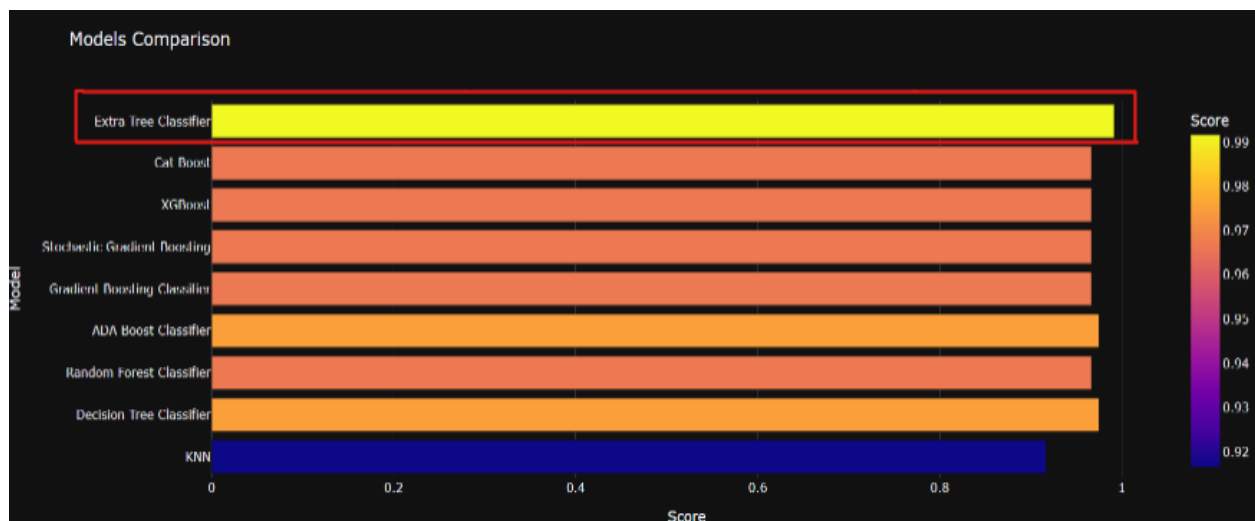
Accuracy

```
Training Accuracy of LGBM Classifier is 100.0
Test Accuracy of LGBM Classifier is 96.66666666666667
```

Model Comparison

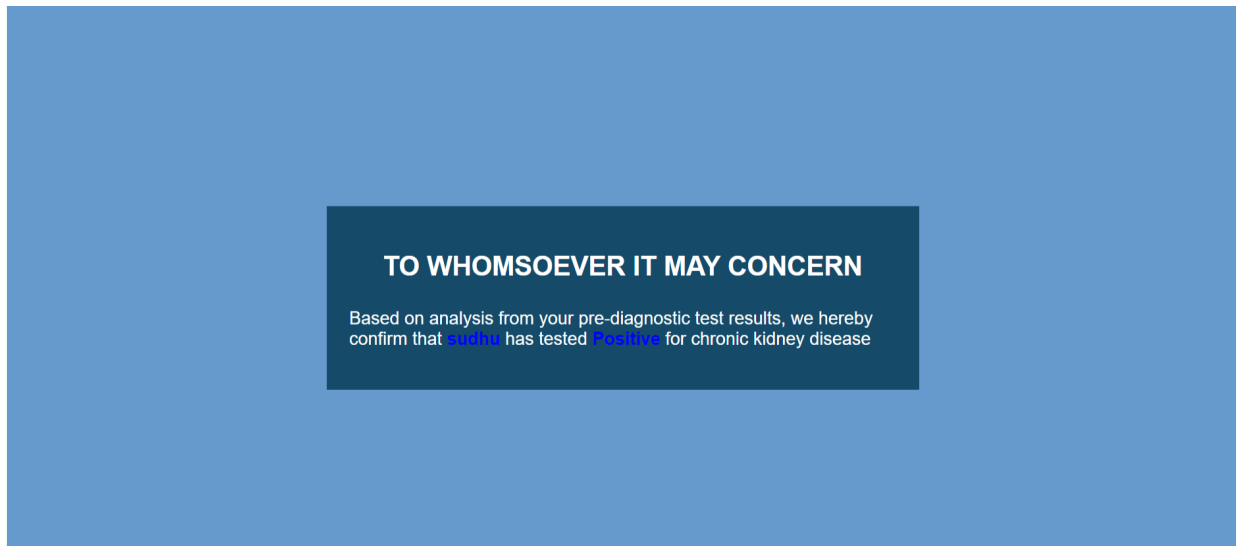
```
models = pd.DataFrame({
    'Model' : [ 'KNN', 'Decision Tree Classifier', 'Random Forest Classifier', 'ADA Boost Classifier',
                'Gradient Boosting Classifier', 'Stochastic Gradient Boosting', 'XGBoost', 'Cat Boost', 'Extra Tree Classifier'],
    'Score' : [knn_acc, dtc_acc, rd_clf_acc, ada_acc, gb_acc, sgb_acc, xgb_acc, cat_acc, etc_acc]
})

models.sort_values(by = 'Score', ascending = False)
```



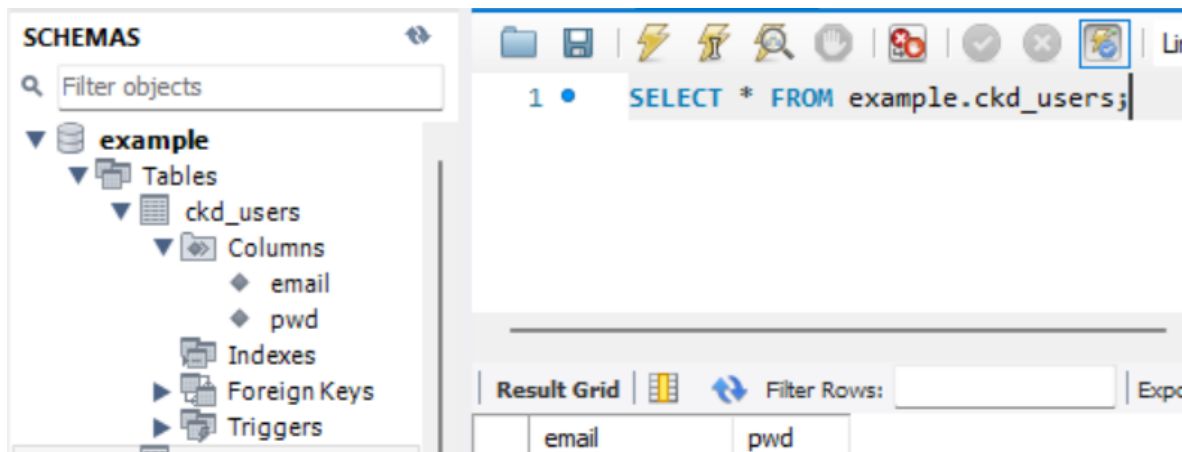
7.2 Feature 2: Report Generation

The forecast outcome is shown on this page. The classifier model receives the user-inputted values from the previous page as input. The user is shown the forecast result in the form of a report.



7.3 Database Schema

The inputs from the website's login page are stored in a database that is made using MySQL. It creates a table called "ckd users" with the columns "email" and "password" in it. When a new user logs in, they can create an account and have their information saved in the database.



CHAPTER 8

TESTING

8.1 Test Cases

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

Resolution	Severity 1	Severity 2	Severity 3	Severity 4	Subtotal
By Design	2	2	0	1	5
Duplicate	1	0	1	0	2
External	2	1	0	1	4
Fixed	1	2	0	0	3
Not Reproduced	0	0	1	0	1
Skipped	0	0	1	0	1
Won't Fix	0	1	0	1	2
Totals	6	6	3	3	18

8.2 User Acceptance Testing

This report shows the number of test cases that have passed, failed and untested.

Section	Total Cases	Not Tested	Fail	Pass
Login Page	2	0	0	2
Pre-diagnostic Form	4	0	0	4
Final Report Output	4	0	0	4

CHAPTER 9

RESULTS

9.1 Deployment in IBM Cloud

The screenshot shows the IBM Cloud console interface for the 'ckdModel' project. The 'Deployments' tab is selected, displaying a table with one deployment entry. The table has columns for Name, Type, Status, Asset, and Last modified. The deployment 'ckd' is in an 'Online' state with a 'Deployed' status. A file upload dialog is open on the right, prompting the user to drop files or browse for files to upload. Below the dialog, a message states: 'Stay on the page until upload completes. Incomplete uploads are cancelled.'

Name	Type	Status	Asset	Last modified
(ip) ckd	Online	Deployed	ckd	1 minute ago Sudharsana Suresh (You)


Integrating Flask with Scoring EndPoint

```
* Running on all addresses (0.0.0.0)
  WARNING: This is a development server. Do not use it in a production deployment.
* Running on http://127.0.0.1:8080
* Running on http://192.168.29.18:8080 (Press CTRL+C to quit)
* Serving Flask app 'app' (lazy loading)
* Environment: production
  WARNING: This is a development server. Do not use it in a production deployment.
  Use a production WSGI server instead.
* Debug mode: off
Scoring response
Positive
```

9.2 Prediction Results

9.2.1 Data Entry Form

[Home](#) [Contact](#) [About](#)



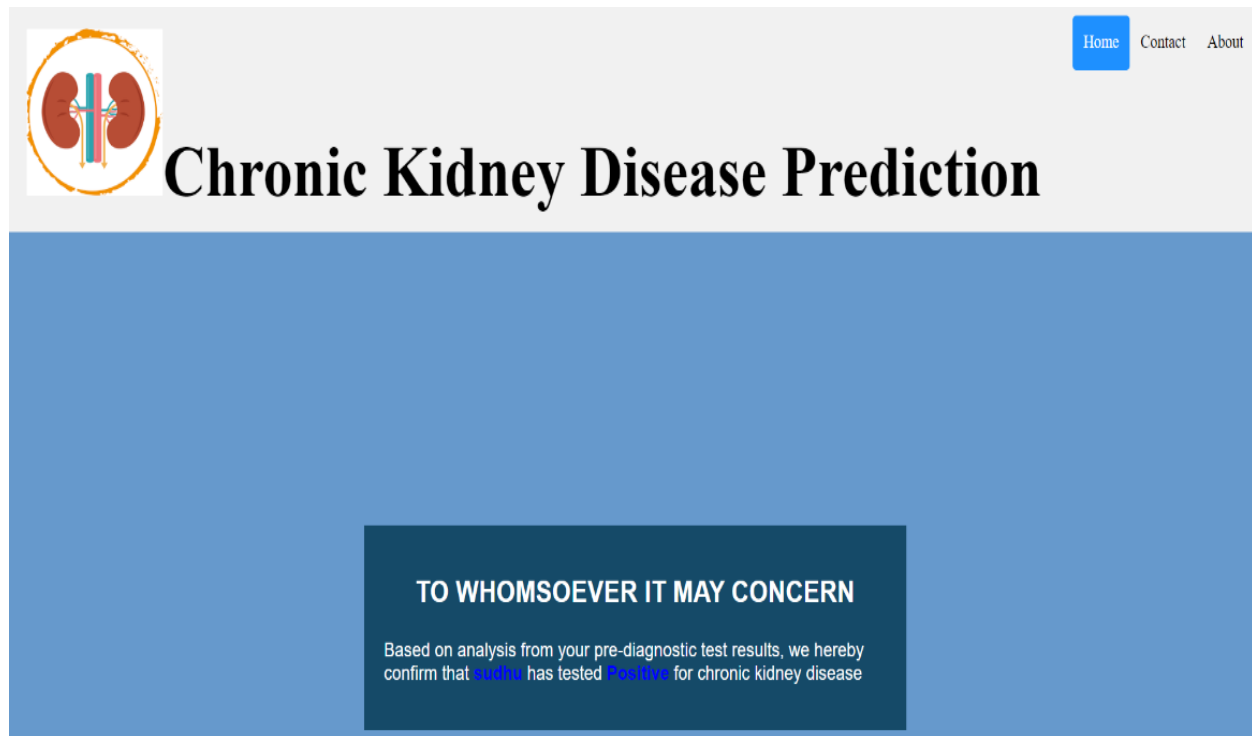
Chronic Kidney Disease Prediction

CKD Prediction

Please fill in this form for generating report

Name	<input type="text" value="sudhu"/>
Specific Gravity	<input type="text" value="1.020"/>
Sugar Level (0-5)	<input type="text" value="4"/>
Albumin (0-4)	<input type="text" value="2"/>
Potassium (in mEq/L)	<input type="text" value="4.4"/>
Haemoglobin	<input type="text" value="15.4"/>
Packed Cell Volume	<input type="text" value="39"/>
Appetite	<input type="text" value="Good"/>
Hypertension	<input type="text" value="Yes"/>
Diabetes Mellitus	<input type="text" value="Yes"/>
Anemia	<input type="text" value="Yes"/>
<input type="button" value="Predict"/>	

9.2.2 Output



The image shows a web interface for "Chronic Kidney Disease Prediction". At the top left is a logo of two kidneys. At the top right are navigation links: "Home" (highlighted in blue), "Contact", and "About". The main title "Chronic Kidney Disease Prediction" is centered below the logo. Below the title is a large blue rectangular area. In the center of this blue area is a dark blue box containing the text "TO WHOMSOEVER IT MAY CONCERN". Below this box, in smaller text, it says "Based on analysis from your pre-diagnostic test results, we hereby confirm that **sudhu** has tested **Positive** for chronic kidney disease".

9.3 Performance Metrics

Model	Accuracy
Gradient Boosting Classifier	Training Accuracy: 1.0 Testing Accuracy: 96.66
Stochastic Gradient Boosting	Training Accuracy: 100.0 Testing Accuracy: 96.66
Extra Tree Classifier	Training Accuracy: 100.0 Testing Accuracy: 99.16
K Neighest Neighbour	Training Accuracy: 94.64 Testing Accuracy: 91.66

Decision Tree Classifier	Training Accuracy: 96.78 Testing Accuracy: 97.5
Random Forest Classifier	Training Accuracy: 99.28 Testing Accuracy: 96.66
Ada Boost Classifier	Training Accuracy: 100.0 Testing Accuracy: 97.5
XGBoost	Training Accuracy: 100.0 Testing Accuracy: 96.66
Cat Boost Classifier	Training Accuracy: 98.92 Testing Accuracy: 96.66
LGBM Classifier	Training Accuracy: 100.0 Testing Accuracy: 96.66

CHAPTER 10

CONCLUSION

The benefit of using this approach is that because the prediction process goes much more quickly, doctors may start treating patients with CKD as soon as possible and classify bigger patient populations in a shorter amount of time. We would want to work with larger datasets in the future or compare the results of this dataset with another dataset that contains the same information because the dataset utilized in this paper only has 400 cases. Additionally, using the proper information, we attempt to determine whether a person with this condition has a higher likelihood of developing chronic risk factors like hypertension, a family history of renal failure, and diabetes in order to reduce the incidence of CKD. For both professionals and patients, making an early prognosis is essential to preventing and delaying the progression of chronic renal disease to kidney failure.

CHAPTER 11

FUTURE SCOPE

In such chronic disorders, data mining algorithms can be employed to pinpoint crucial indicators of disease suffering. When people are struggling with the cost of doing multiple tests, data prediction can be used as an affordable and precise option. The algorithm assists in choosing a subset of tests that would provide comparable accuracy rather than doing all of them, saving a significant amount of money. Because it is free, Boruta Analysis aids in medical diagnosis, which might be costly otherwise. It expedites and makes the diagnostic more affordable for the patients. The problems are stopped from getting worse by early detection. It can be used for a variety of ailments as well as countless other things and is not only restricted to treating chronic illnesses.

CHAPTER 12

APPENDIX

11.1. Source code

login.html

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Login Page</title>
<style><!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Login Page</title>
<style>
```

```
input[type=text], input[type=password] {
    width: 100%;
    padding: 12px 20px;
    margin: 8px 0;
    display: inline-block;
    border: 1px solid #ccc;
    box-sizing: border-box;
}
```

```
button {
```

```
background-color: #04AA6D;
color: white;
padding: 14px 20px;
margin: 8px 0;
border: none;
cursor: pointer;
width: 100%;
}
```

```
button:hover {
  opacity: 0.8;
}
```

```
.cancelbtn {
  width: auto;
  padding: 10px 18px;
  background-color: #f44336;
}
```

```
.imgcontainer {
  text-align: center;
  margin: 24px 0 12px 0;
}
```

```
img.avatar {
  width: 40%;
  border-radius: 50%;
}
```

```
.container {
  padding: 16px;
}
```

```
span.psw {
  float: right;
  padding-top: 16px;
}

@media screen and (max-width: 300px) {
  span.psw {
    display: block;
    float: none;
  }
  .cancelbtn {
    width: 100%;
  }
}

.header {
  overflow: hidden;
  background-color: #f1f1f1;
  padding: 20px 10px;
}

.header a {
  float: left;
  color: black;
  text-align: center;
  padding: 12px;
  text-decoration: none;
  font-size: 18px;
  line-height: 25px;
  border-radius: 4px;
}
```

```
.header a.logo {
  font-size: 75px;
  font-weight: bold;
  align: center;
}

.header a:hover {
  background-color: #ddd;
  color: black;
}

.header a.active {
  background-color: dodgerblue;
  color: white;
}

.header-right {
  float: right;
}

@media screen and (max-width: 500px) {
  .header a {
    float: none;
    display: block;
    text-align: left;
  }
  .header-right {
    float: none;
  }
}

* {box-sizing: border-box}
```

```
.container {  
  padding: 16px;  
}  
  
input[type=text], input[type=password] {  
  width: 100%;  
  padding: 15px;  
  margin: 5px 0 22px 0;  
  display: inline-block;  
  border: none;  
  background: #f1f1f1;  
}  
  
input[type=text]:focus, input[type=password]:focus {  
  background-color: #ddd;  
  outline: none;  
}  
  
hr {  
  border: 1px solid #f1f1f1;  
  margin-bottom: 25px;  
}  
  
.registerbtn {  
  background-color: #04AA6D;  
  color: white;  
  padding: 16px 20px;  
  margin: 8px 0;  
  border: none;  
  cursor: pointer;  
  width: 100%;
```



```

    opacity: 0.9;
}

.registerbtn:hover {
    opacity:1;
}

a {
    color: dodgerblue;
}

.signin {
    background-color: #f1f1f1;
    text-align: center;
}
</style>
</head>
<body>

<form action="/action_page.php" method="post">
    <div class="header">

        <a href="#default" class="logo">Chronic Kidney Disease Prediction</a>
        <div class="header-right">
            <a class="active" href="#home">Home</a>
            <a href="#contact">Contact</a>
            <a href="#about">About</a>
        </div>
    </div>
    <h2>Login Form</h2>
    <div class="container">

```

```

<label for="uname"><b>Username</b></label>
<input type="text" placeholder="Enter Username" name="uname" required>

<label for="psw"><b>Password</b></label>
<input type="password" placeholder="Enter Password" name="psw" required>

<button type="submit">Login</button>
<label>
  <input type="checkbox" checked="checked" name="remember"> Remember
me
</label>
</div>

<div class="container" style="background-color:#f1f1f1">
  <button type="button" class="cancelbtn">Cancel</button>
  <span class="psw">Forgot <a href="#">password?</a></span>
</div>
</form>
</body>
</html>

```

format.css

```

.imgcontainer {
  text-align: center;
  margin: 24px 0 12px 0;
}

.header {
  overflow: hidden;
  background-color: #f1f1f1;
  padding: 20px 10px;
}

```

```
}
```

```
.header a {  
  float: left;  
  color: black;  
  text-align: center;  
  padding: 12px;  
  text-decoration: none;  
  font-size: 18px;  
  line-height: 25px;  
  border-radius: 4px;  
}
```

```
.header a.logo {  
  font-size: 75px;  
  font-weight: bold;  
  align: center;  
}
```

```
.header a:hover {  
  background-color: #ddd;  
  color: black;  
}
```

```
.header a.active {  
  background-color: dodgerblue;  
  color: white;  
}
```

```
.header-right {  
  float: right;
```

```
}

@media screen and (max-width: 500px) {
  .header a {
    float: none;
    display: block;
    text-align: left;
  }
  .header-right {
    float: none;
  }
}

* {box-sizing: border-box}

.container {
  padding: 16px;
}

input[type=text], input[type=password] {
  width: 100%;
  padding: 15px;
  margin: 5px 0 22px 0;
  display: inline-block;
  border: none;
  background: #f1f1f1;
}

input[type=text]:focus, input[type=password]:focus {
  background-color: #ddd;
  outline: none;
}
```

```
hr {
  border: 1px solid #f1f1f1;
  margin-bottom: 25px;
}

.registerbtn {
  background-color: #04AA6D;
  color: white;
  padding: 16px 20px;
  margin: 8px 0;
  border: none;
  cursor: pointer;
  width: 100%;
  opacity: 0.9;
}

.registerbtn:hover {
  opacity: 1;
}

a {
  color: dodgerblue;
}

.signin {
  background-color: #f1f1f1;
  text-align: center;
}
```

registration.html

```
<!DOCTYPE html>
<html lang="en">
```

```
<head>
  <meta charset="UTF-8">
  <title>Registration Page</title>
</head>
<style>
.imgcontainer {
  text-align: center;
  margin: 24px 0 12px 0;
}

.header {
  overflow: hidden;
  background-color: #f1f1f1;
  padding: 20px 10px;
}

.header a {
  float: left;
  color: black;
  text-align: center;
  padding: 12px;
  text-decoration: none;
  font-size: 18px;
  line-height: 25px;
  border-radius: 4px;
}

.header a.logo {
  font-size: 75px;
  font-weight: bold;
  align: center;
}
```

```
.header a:hover {
  background-color: #ddd;
  color: black;
}

.header a.active {
  background-colour: dodgerblue;
  color: white;
}

.header-right {
  float: right;
}

@media screen and (max-width: 500px) {
  .header a {
    float: none;
    display: block;
    text-align: left;
  }
  .header-right {
    float: none;
  }
}

* {box-sizing: border-box}

.container {
  padding: 16px;
}
```

```
input[type=text], input[type=password] {  
    width: 100%;  
    padding: 15px;  
    margin: 5px 0 22px 0;  
    display: inline-block;  
    border: none;  
    background: #f1f1f1;  
}  
  
input[type=text]:focus, input[type=password]:focus {  
    background-color: #ddd;  
    outline: none;  
}  
  
hr {  
    border: 1px solid #f1f1f1;  
    margin-bottom: 25px;  
}  
  
.registerbtn {  
    background-color: #04AA6D;  
    color: white;  
    padding: 16px 20px;  
    margin: 8px 0;  
    border: none;  
    cursor: pointer;  
    width: 100%;  
    opacity: 0.9;  
}  
  
.registerbtn:hover {  
    opacity: 1;
```



```

}

a {
  color: dodgerblue;
}

.signin {
  background-color: #f1f1f1;
  text-align: center;
}
</style>
<body>
<div></div>
<div class="header">

  <a href="#default" class="logo">Chronic Kidney Disease Prediction</a>
  <div class="header-right">
    <a class="active" href="#home">Home</a>
    <a href="#contact">Contact</a>
    <a href="#about">About</a>
  </div>
</div>

<form action="action_page.php">
  <div class="container">
    <h1>Register</h1>
    <p>Please fill in this form to create an account.</p>
    <hr>

    <label for="email"><b>Email</b></label>
    <input type="text" placeholder="Enter Email" name="email" id="email"

```

```
required>
```

```
<label for="psw"><b>Password</b></label>
```

```
<input type="password" placeholder="Enter Password" name="psw" id="psw"
required>
```

```
<label for="psw-repeat"><b>Re-enter Password</b></label>
```

```
<input type="password" placeholder="Re-enter Password" name="psw-repeat"
id="psw-repeat" required>
```

```
<hr>
```

```
<button type="submit" class="registerbtn">Create Account</button>
```

```
</div>
```

```
<div class="container signin">
```

```
<p>Already have an account? <a href="#">Sign in</a>.</p>
```

```
</div>
```

```
</form>
```

```
</body>
```

```
</html>
```

Data cleaning.ipynb

```
import numpy as np
```

```
import pandas as pd
```

```
import matplotlib.pyplot as plt
```

```
import seaborn as sns
```

```
import warnings
```

```
warnings.filterwarnings('ignore')
```

```
df=pd.read_csv(r'/content/drive/MyDrive/Dataset_med_rec/kidney_disease.csv')
```

```
df.info()
```

```
df['pcv']=df['pcv'].apply(lambda x:x if type(x)==type(3.5) else
```

```

x.replace('\t43','43').replace('\t?', 'Nan'))

# cleaning "WC"
df['wc']=df['wc'].apply(lambda x:x if type(x)==type(3.5) else
x.replace('\t?', 'Nan').replace('\t6200','6200').replace('\t8400','8400'))

# cleaning "RC"
df['rc']=df['rc'].apply(lambda x:x if type(x)==type(3.5) else x.replace('\t?', 'Nan'))

# cleaning "dm"
df['dm']=df['dm'].apply(lambda x:x if type(x)==type(3.5) else
x.replace('\tno','no').replace('\tyes','yes').replace(' yes','yes'))

# cleaning "CAD"
df['cad']=df['cad'].apply(lambda x:x if type(x)==type(3.5) else x.replace('\tno','no'))

# cleaning "Classification"
df['classification']=df['classification'].apply(lambda x:x if type(x)==type(3.5) else
x.replace('ckd\t','ckd'))

#explicitly converting numerical columns
mistyped=[['pcv','rc','wc']]
for i in mistyped:
    df[i]=df[i].astype('float')

#categorical columns
cat_cols=list(df.select_dtypes('object'))
cat_cols

#numerical columns
num_cols=list(df.select_dtypes(['int64','float64']))
num_cols

```

```

#handling missing data
df.isnull().sum().sort_values(ascending=False)

#filling missing values with mode for categorical attributes
df['rbc'].fillna('normal',inplace=True)
df['pc'].fillna('normal',inplace=True)
df['pcc'].fillna('notpresent',inplace=True)
df['ba'].fillna('notpresent',inplace=True)
df['htn'].fillna('no',inplace=True)
df['dm'].fillna('no',inplace=True)
df['cad'].fillna('no',inplace=True)
df['appet'].fillna('good',inplace=True)
df['pe'].fillna('no',inplace=True)
df['ane'].fillna('no',inplace=True)

#filling missing values with median for numerical attributes
for col in num_cols:
    df[col]=df[col].fillna(df[col].median())
df.isna().sum().sort_values(ascending=False)
df['classification'].unique()

#label encoding for target class
df['classification']=df['classification'].map({'ckd':1,'notckd':0})

#label encoding for categorical attributes
df['rbc']=df['rbc'].map({'normal':0,'abnormal':1})
df['pc']=df['pc'].map({'normal':0,'abnormal':1})
df['pcc']=df['pcc'].map({'notpresent':0,'present':1})
df['ba']=df['ba'].map({'notpresent':0,'present':1})
df['htn']=df['htn'].map({'no':0,'yes':1})
df['dm']=df['dm'].map({'no':0,'yes':1})

```

```

df['cad']=df['cad'].map({'no':0,'yes':1})
df['pe']=df['pe'].map({'no':0,'yes':1})
df['ane']=df['ane'].map({'no':0,'yes':1})
df['appet']=df['appet'].map({'good':0,'poor':1})
from sklearn.model_selection import train_test_split
x=df.drop('classification',axis=1)#independent
y=df['classification']#dependent
X_train,X_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=0)
print("X_train size {} , X_test size {}".format(X_train.shape,X_test.shape))

```

Model_Building_On_IBM

```

import warnings
from decimal import Decimal
import numpy as np
import seaborn as sns
import pandas as pd
from sklearn.ensemble import ExtraTreesClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score,confusion_matrix,classification_report
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import RandomForestClassifier
from xgboost import XGBClassifier
import matplotlib.pyplot as plt
import os, types
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='****',
    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 = 'ckdprediction-donotdelete-pr-kvcxes1swicxqv'
object_key = 'processed_kidney.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 = pd.read_csv(body)
df.head()

# Separating the dependent and independent variables
y = df['classification']
X = df.drop(['classification','id'], axis = 1)
X.head()

# Feature Selection using Extra Tree Classifier

# Building the model

```

```

extra_tree_forest = ExtraTreesClassifier(n_estimators = 5,criterion ='entropy',
max_features = 2)

# Training the model
extra_tree_forest.fit(X, y)

# Computing the importance of each feature
feature_importance = extra_tree_forest.feature_importances_

# Normalizing the individual importances
feature_importance_normalized = np.std([tree.feature_importances_ for tree in
extra_tree_forest.estimators_],
axis = 0)

# Plotting a Bar Graph to compare the models
plt.figure(figsize=(50,20))
plt.bar(X.columns, feature_importance_normalized)

plt.xlabel('Feature Labels')
plt.ylabel('Feature Importances')
plt.title('Comparison of different Feature Importances')
plt.show()

# Identifying top 10 features for pre-diagnosis test

feature_scores=pd.DataFrame(extra_tree_forest.feature_importances_,columns=['S
core'],index=X.columns).sort_values(by='Score',ascending=False)
top10_feature = feature_scores.nlargest(n=10, columns=['Score'])
plt.figure(figsize=(20,6))

```

```

print(top10_feature.index)
g = sns.barplot(x=top10_feature.index, y=top10_feature['Score'])
p = plt.title('Top 10 Features with Extra Tree Classifier')
p = plt.xlabel('Feature name')
p = plt.ylabel('Extra Tree score')
p = g.set_xticklabels(g.get_xticklabels(), horizontalalignment='right')

```

```

top10_feature.index

```

```

X.columns

```

```

X.columns

```

```

for ele in X.columns:
    if ele not in top10_feature.index:
        X = X.drop(ele, axis = 1)

```

```

X.head()

```

```

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.30,
random_state = None)

```

```

from sklearn.ensemble import GradientBoostingClassifier
gb = GradientBoostingClassifier()
gb.fit(X_train, y_train)

```

```

# accuracy score, confusion matrix and classification report of gradient boosting
classifier
gb_acc = accuracy_score(y_test, gb.predict(X_test))

```

```

print(f"Training Accuracy of Gradient Boosting Classifier is

```



```

{accuracy_score(y_train, gb.predict(X_train))}")
print(f"Test Accuracy of Gradient Boosting Classifier is {gb_acc*100} \n")

print(f"Confusion Matrix :- \n{confusion_matrix(y_test, gb.predict(X_test))}\n")
print(f"Classification Report :- \n {classification_report(y_test,
gb.predict(X_test))}")

# Stochastic Gradient Boosting (SGB)

sgb = GradientBoostingClassifier(max_depth = 4, subsample = 0.90, max_features
= 0.75, n_estimators = 200)
sgb.fit(X_train, y_train)

# accuracy score, confusion matrix and classification report of stochastic gradient
boosting classifier

sgb_acc = accuracy_score(y_test, sgb.predict(X_test))

print(f"Training Accuracy of Stochastic Gradient Boosting is
{accuracy_score(y_train, sgb.predict(X_train))*100}")
print(f"Test Accuracy of Stochastic Gradient Boosting is {sgb_acc*100} \n")

print(f"Confusion Matrix :- \n{confusion_matrix(y_test, sgb.predict(X_test))}\n")
print(f"Classification Report :- \n {classification_report(y_test,
sgb.predict(X_test))}")

# **Extra Tree Classifier**

# accuracy score, confusion matrix and classification report of extra tree classifier
etc = ExtraTreesClassifier()

```

```

etc.fit(X_train, y_train)

etc_acc = accuracy_score(y_test, etc.predict(X_test))

print(f"Training Accuracy of Extra Trees Classifier is {accuracy_score(y_train,
etc.predict(X_train))*100}")
print(f"Test Accuracy of Extra Trees Classifier is {etc_acc*100} \n")

print(f"Confusion Matrix :- \n{confusion_matrix(y_test, etc.predict(X_test))}\n")
print(f"Classification Report :- \n {classification_report(y_test,
etc.predict(X_test))}")

# **KNN**

from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, confusion_matrix,
classification_report

knn = KNeighborsClassifier()
knn.fit(X_train, y_train)

# accuracy score, confusion matrix and classification report of knn
knn_acc = accuracy_score(y_test, knn.predict(X_test))
print(f"Training Accuracy of KNN is {accuracy_score(y_train,
knn.predict(X_train))*100}")
print(f"Test Accuracy of KNN is {knn_acc*100} \n")

print(f"Confusion Matrix :- \n{confusion_matrix(y_test, knn.predict(X_test))}\n")
print(f"Classification Report :- \n {classification_report(y_test,
knn.predict(X_test))}")

```

```

# **Decision Tree Classifier**

from sklearn.tree import DecisionTreeClassifier

dtc = DecisionTreeClassifier()
dtc.fit(X_train, y_train)

# accuracy score, confusion matrix and classification report of decision tree

dtc_acc = accuracy_score(y_test, dtc.predict(X_test))

print(f"Training Accuracy of Decision Tree Classifier is {accuracy_score(y_train,
dtc.predict(X_train))*100}")
print(f"Test Accuracy of Decision Tree Classifier is {dtc_acc*100} \n")

print(f"Confusion Matrix :- \n{confusion_matrix(y_test, dtc.predict(X_test))}\n")
print(f"Classification Report :- \n {classification_report(y_test,
dtc.predict(X_test))}")

# hyper parameter tuning of decision tree

from sklearn.model_selection import GridSearchCV
grid_param = {
    'criterion' : ['gini', 'entropy'],
    'max_depth' : [3, 5, 7, 10],
    'splitter' : ['best', 'random'],
    'min_samples_leaf' : [1, 2, 3, 5, 7],
    'min_samples_split' : [1, 2, 3, 5, 7],
    'max_features' : ['auto', 'sqrt', 'log2']
}

```

```

grid_search_dtc = GridSearchCV(dtc, grid_param, cv = 5, n_jobs = -1, verbose =
1)
grid_search_dtc.fit(X_train, y_train)

# best estimator
dtc = grid_search_dtc.best_estimator_

# accuracy score, confusion matrix and classification report of decision tree
dtc_acc = accuracy_score(y_test, dtc.predict(X_test))

print(f"Training Accuracy of Decision Tree Classifier is {accuracy_score(y_train,
dtc.predict(X_train))*100}")
print(f"Test Accuracy of Decision Tree Classifier is {dtc_acc*100} \n")

print(f"Confusion Matrix :- \n{confusion_matrix(y_test, dtc.predict(X_test))}\n")
print(f"Classification Report :- \n {classification_report(y_test,
dtc.predict(X_test))}")

# **Random Forest Classifier**

from sklearn.ensemble import RandomForestClassifier

rd_clf = RandomForestClassifier(criterion = 'entropy', max_depth = 11,
max_features = 'auto', min_samples_leaf = 2, min_samples_split = 3, n_estimators
= 130)
rd_clf.fit(X_train, y_train)

# accuracy score, confusion matrix and classification report of random forest

rd_clf_acc = accuracy_score(y_test, rd_clf.predict(X_test))

```

```
print(f"Training Accuracy of Random Forest Classifier is {accuracy_score(y_train,
rd_clf.predict(X_train))*100}")
print(f"Test Accuracy of Random Forest Classifier is {rd_clf_acc*100} \n")
```

```
print(f"Confusion Matrix :- \n{confusion_matrix(y_test,
rd_clf.predict(X_test))}\n")
print(f"Classification Report :- \n {classification_report(y_test,
rd_clf.predict(X_test))}")
```

```
# ADA Boost Classifier
```

```
from sklearn.ensemble import AdaBoostClassifier
```

```
ada = AdaBoostClassifier(base_estimator = dtc)
ada.fit(X_train, y_train)
```

```
# accuracy score, confusion matrix and classification report of ada boost
```

```
ada_acc = accuracy_score(y_test, ada.predict(X_test))
```

```
print(f"Training Accuracy of Ada Boost Classifier is {accuracy_score(y_train,
ada.predict(X_train))*100}")
print(f"Test Accuracy of Ada Boost Classifier is {ada_acc*100} \n")
```

```
print(f"Confusion Matrix :- \n{confusion_matrix(y_test, ada.predict(X_test))}\n")
print(f"Classification Report :- \n {classification_report(y_test,
ada.predict(X_test))}")
```

```
# XGBoost
```

```

from xgboost import XGBClassifier

xgb = XGBClassifier(objective = 'binary:logistic', learning_rate = 0.5, max_depth
= 5, n_estimators = 150)
xgb.fit(X_train, y_train)

# accuracy score, confusion matrix and classification report of xgboost

xgb_acc = accuracy_score(y_test, xgb.predict(X_test))

print(f"Training Accuracy of XgBoost is {accuracy_score(y_train,
xgb.predict(X_train))*100}")
print(f"Test Accuracy of XgBoost is {xgb_acc*100} \n")

print(f"Confusion Matrix :- \n{confusion_matrix(y_test, xgb.predict(X_test))}\n")
print(f"Classification Report :- \n {classification_report(y_test,
xgb.predict(X_test))}")

get_ipython().system('pip install catboost')

# **Cat Boost Classifier**

from catboost import CatBoostClassifier

cat = CatBoostClassifier(iterations=10)
cat.fit(X_train, y_train)

cat_acc = accuracy_score(y_test, cat.predict(X_test))

print(f"Training Accuracy of Cat Boost Classifier is {accuracy_score(y_train,

```

```

cat.predict(X_train))*100}")
print(f"Test Accuracy of Cat Boost Classifier is {cat_acc*100} \n")

print(f"Confusion Matrix :- \n{confusion_matrix(y_test, cat.predict(X_test))}\n")
print(f"Classification Report :- \n {classification_report(y_test,
cat.predict(X_test))}")

# **LGBM Classifier**

from lightgbm import LGBMClassifier

lgbm = LGBMClassifier(learning_rate = 1)
lgbm.fit(X_train, y_train)

# accuracy score, confusion matrix and classification report of lgbm classifier

lgbm_acc = accuracy_score(y_test, lgbm.predict(X_test))

print(f"Training Accuracy of LGBM Classifier is {accuracy_score(y_train,
lgbm.predict(X_train))*100}")
print(f"Test Accuracy of LGBM Classifier is {lgbm_acc*100} \n")

print(f"{confusion_matrix(y_test, lgbm.predict(X_test))}\n")
print(classification_report(y_test, lgbm.predict(X_test)))

# **Models Comparison**

models = pd.DataFrame({
    'Model' : [ 'KNN', 'Decision Tree Classifier', 'Random Forest Classifier', 'ADA
Boost Classifier',

```

```

        'Gradient Boosting Classifier', 'Stochastic Gradient Boosting', 'XGBoost',
        'Cat Boost', 'Extra Tree Classifier'],
        'Score' : [knn_acc, dtc_acc, rd_clf_acc, ada_acc, gb_acc, sgb_acc, xgb_acc,
        cat_acc, etc_acc]
    })

```

```

models.sort_values(by = 'Score', ascending = False)

```

```

import plotly.express as px
px.bar(data_frame = models, x = 'Score', y = 'Model', color = 'Score', template =
'plotly_dark',
        title = 'Models Comparison')

```

```

get_ipython().system('pip install ibm_watson_machine_learning')

```

```

from ibm_watson_machine_learning import APIClient
wml_credentials = {
    "url" : "https://us-south.ml.cloud.ibm.com",
    "apikey" : "*****"
}
client = APIClient(wml_credentials)

```

```

def guid_from_space_name(client, space_name):
    space = client.spaces.get_details()
    return(next(item for item in space['resources'] if item['entity']['name'] ==
space_name)['metadata']['id'])

```

```

space_uid = guid_from_space_name(client, 'ckdModel')
print("sSpace UID = "+space_uid)

```

```

client.set.default_space(space_uid)

```



```
client.software_specifications.list(limit=200)
```

```
software_spec_uid = client.software_specifications.get_uid_by_name("runtime-22.1-py3.9")
```

```
software_spec_uid
```

```
model_details = client.repository.store_model(model = etc, meta_props = {  
    client.repository.ModelMetaNames.NAME: "ckd",  
    client.repository.ModelMetaNames.TYPE:"scikit-learn_1.0",  
    client.repository.ModelMetaNames.SOFTWARE_SPEC_UID:  
software_spec_uid  
}  
)
```

```
model_id = client.repository.get_model_uid(model_details)
```

```
model_id
```

Integrating flask with end score

app.py

```
from flask import Flask,render_template,request
```

```
#import pickle
```

```
import requests
```

```
# NOTE: you must manually set API_KEY below using information retrieved  
from your IBM Cloud account.
```

```
API_KEY = "v6zXMCaTXBv8fJlMVEucGxo5uWRpDfXWGxqBpPn2P_Xy"
```

```
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}
```

```
#model=pickle.load(open('CKDmodel.pkl','rb'))
```

```
app=Flask(__name__)
```

```
@app.route('/')
```

```
def index():
```

```
    return render_template('index.html')
```

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

```
def predict_satisfaction():
```

```
    v1=float(request.form.get('sg'))
```

```
    v2=int(request.form.get('su'))
```

```
    v3=int(request.form.get('ab'))
```

```
    v4=float(request.form.get('pot'))
```

```
    v5=float(request.form.get('hemo'))
```

```
    v6=int(request.form.get('pcv'))
```

```
    v7=request.form.get('ap')
```

```
    v7 = 0 if 'Good' else 1
```

```
    v8=request.form.get('ht')
```

```
    v8 = 0 if 'No' else 1
```

```
    v9=request.form.get('dm')
```

```
    v9=0 if 'No' else 1
```

```
    v10=request.form.get('ae')
```

```
    v10=0 if 'No' else 1
```

```
    name=request.form.get('name')
```

```
    total = [[v1,v3,v2,v4,v5,v6,v8,v9,v7,v10]]
```

```
    # NOTE: manually define and pass the array(s) of values to be scored in the next line
```

```
    payload_scoring = {"input_data": [{"field": ["sg", "ab", "su", "pot", "hemo",
```

```

"pcv", "ht", "dm","ap", "ae"]],
        "values": total}}}]
    response_scoring = requests.post(
        'https://us-south.ml.cloud.ibm.com/ml/v4/deployments/2abfa46a-b72d-4d9c-
8165-8a20f83abe0d/predictions?version=2022-11-20',
        json=payload_scoring,
        headers={'Authorization': 'Bearer ' + mltoken})
    print("Scoring response")
    d = {0: "Negative", 1: "Positive"}
    pred=response_scoring.json()
    res=pred['predictions'][0]['values'][0][0]
    print(d[int(res)])
    return render_template('report.html', res=d[int(res)], pname=name)
'''

#prediction
result=model.predict(np.array([v1,v2,v3,v4,v5,v6,v7,v8,v9,v10]).reshape(1,10))
d={0:"Negative",1:"Positive"}
prediction_result=d[result[0]]
return render_template('report.html',res=prediction_result,pname=name)
'''

if __name__=='__main__':
    app.run(host='0.0.0.0',port=8080)

```

11.2. Github link:

<https://github.com/IBM-EPBL/IBM-Project-2050-1658424611>

11.3. Demo link:

<https://drive.google.com/file/d/1X6udxq0Kya8h-PT1DCwZH4DR5AAhILbj/view?usp=sharing>