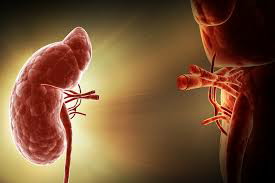
**Chronic Kidney Disease(CKD) using ML**

****

Team Members:(Team-24)

**K. Rupak Teja Reddy**

**T.Sai Ram Akshay**

**V.C. Manikanta Gupta**

**M. Sudha Kiran**

**I. Asish**

In Association with: Smartbridge

Trainee Name: Pradeepthi Duggaraju

Mentor Support: 1)Nikhil Gurram

2)Ram Mohan Bethi

**Index**

1.Introduction

1.1 Overview

1.2 Purpose

2. Literature Survey

2.1 Existing Problem

2.2 Proposed Solution

3. Theoretical Analysis

3.1 Block Diagram

3.2 Hardware/Software Designing

4. Experimental Investigations

5. Flowchart

6. Result

7. Advantages & disadvantages

8. Applications

9. Conclusion

10. Future Scope

11. Bibliography

12. Appendix

1. Introduction:-

1.1 Overview:-

Our kidneys are vital organs that clean our blood and affect many different bodily functions. Various medical conditions can cause the kidneys to stop working properly for a short time – for example, if blood isn’t pumped around the body properly or if an enlarged prostate or kidney stones make urine build up in the kidneys.

But if the kidneys still aren’t working at full capacity more than three months later or are permanently damaged, then the problem is considered to be chronic kidney disease. The most common causes in adults are diabetes and high blood pressure.

Older people are especially prone to developing chronic kidney disease. It often goes unnoticed for a long time because poorly functioning kidneys don’t cause any problems at first.

Treatment mostly aims to stop the kidney disease from getting worse, or at least delay the process. It is also important to treat complications such as anemia, acidosis or changes in bone metabolism. If the kidneys fail completely, dialysis or a kidney transplant can prolong the person’s life.

1.2 Purpose:-

Technological development, including machine learning, has a huge impact on health through an effective analysis of various chronic diseases for more accurate diagnosis and successful treatment. Kidney disease is a major chronic disease associated with aging, hypertension, and diabetes, affecting people 60 and over. Its major cause is the malfunctioning of the kidney in disposing toxins from the blood. This study analysis chronic kidney disease using machine learning techniques based on a chronic kidney disease (CKD) dataset from the UCI machine learning data warehouse. CKD is detected using the Apriori association technique for 400 instances of chronic kidney patients with 10-fold-cross-validation testing, and the results are compared across a number of classification algorithms and regression techniques. The dataset is pre-processed by completing and normalizing missing data. The most relevant features are selected from the dataset for improved accuracy and reduced training time. The results for selected features of the dataset indicate 99% detection accuracy for CKD. The identified technique is further tested using test and train data samples to predict their CKD.

1. Literature Survey:-

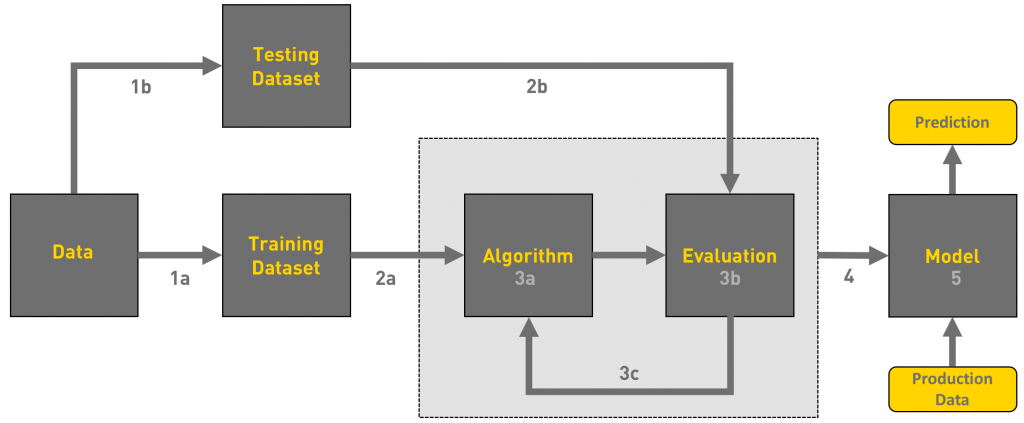
* 1. Existing Problem:-

Progressive CKD is linked to several complications with higher prevalence and intensity at lower levels of kidney function, which interact with each other and these complications contribute to high morbidity and mortality and poor quality of life. Some of these complications can be readily defined and quantified (cardiovascular disease, hypertension, anemia, mineral bone disorder, volume overload, electrolytes, and acid-base abnormalities) and may require a specific management approach, for example, the prescription of erythropoiesis-stimulating agents to correct anemia. Other less well-defined complications with a less distinct pathogenesis, such as anorexia, fatigue, cachexia, pruritus, nausea, and sexual dysfunction, may manifest as complex symptoms often associated with advanced CKD. The major problem faced by the CKD affected persons was not diagnosed until final stages.

2.2 Proposed Solution:-

With the use of Machine Learning Model, there will be no limitation of the complexity increasing number of variables. This Model and train and test the given factors which causes Chronic Kidney Disease and with the best performing machine learning model it can effortlessly predict the result of the patient with much higher accuracy than traditional methods.

1. Theoretical Analysis:-
   1. Block Diagram:-



3.2 Hardware / Software designing:-

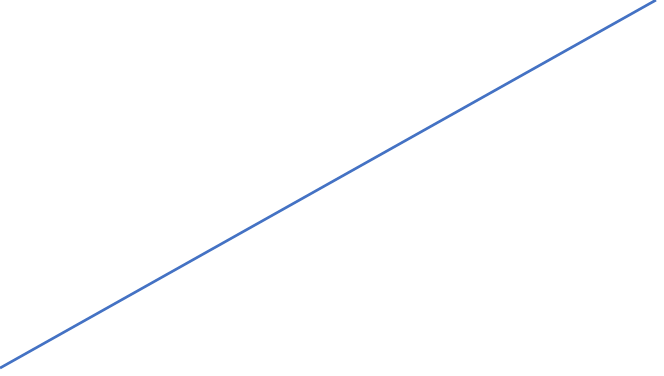
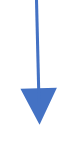
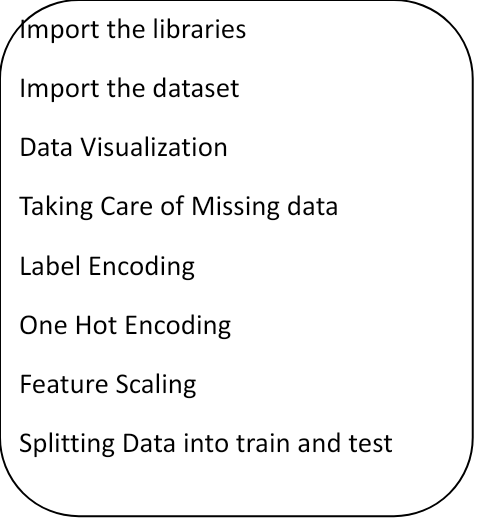
Python, Python Web Frame Works, Python for Data Analysis, Python for Data Visualization, Data Pre-processing Techniques, Machine Learning, Regression Algorithms

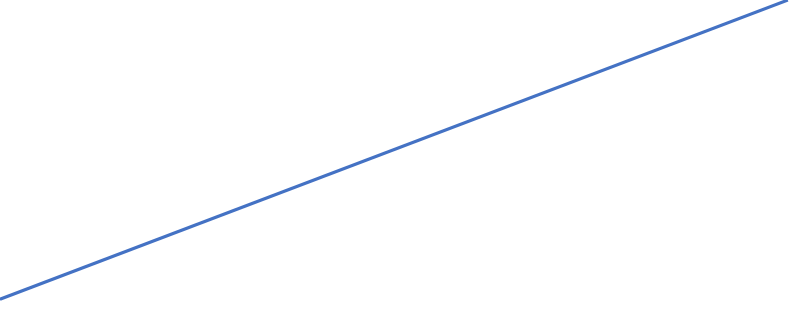
1. Experimental Investigations:

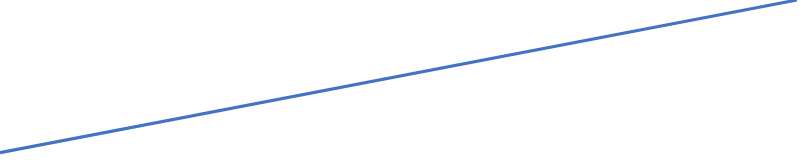
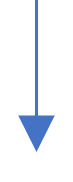
During our investigation, we got to know all the required parameters to predict whether the person is suffering from Chronic Kidney Disease or not and we also analysed different models and concluded the best model for predicting the output.

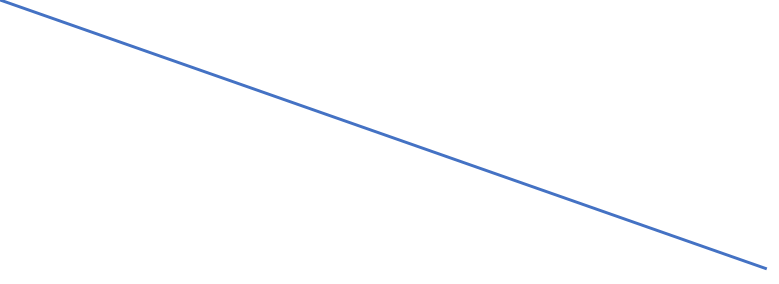
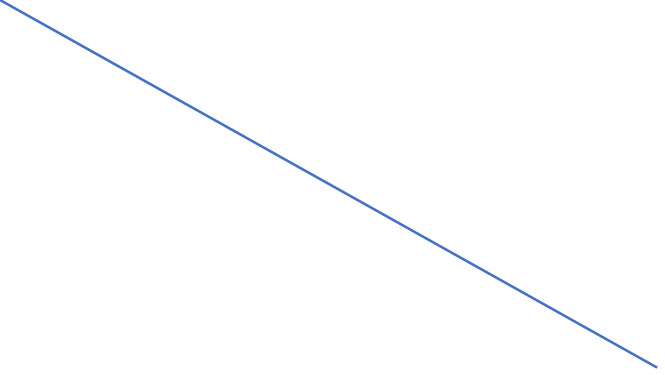
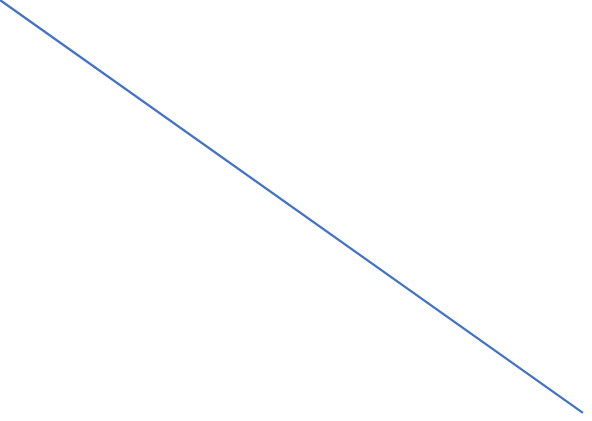
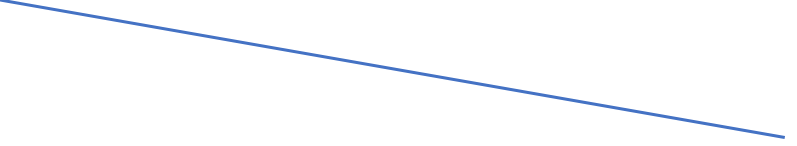
1. Flowchart:-

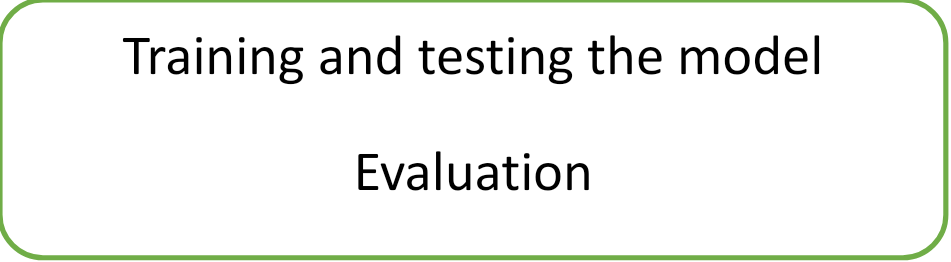


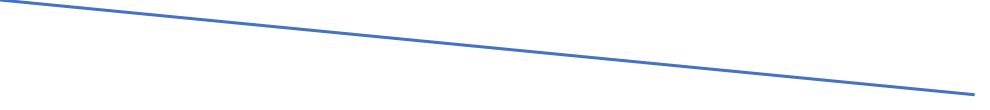
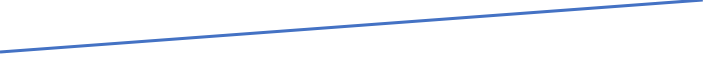


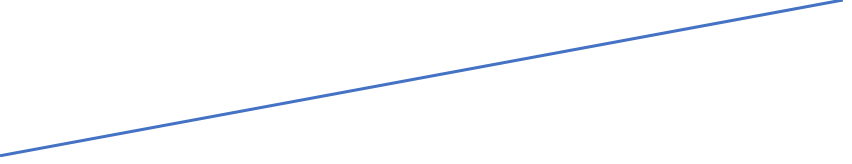


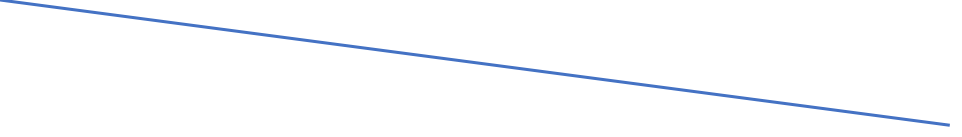












1. Result:-

We have analysed the data and used Machine Learning to predict the Compressive Strength of Concrete. We have used Linear Regression and Decision Tree Classifier to make predictions and compared their performance. Decision Tree Classifier has highest accuracy and is a good choice for this problem. Decision Tree Classifier trains the model with subsets of data sampled from the training data, this will make our model more robust and accurate.

1. Advantages and disadvantages:-

Advantages:-

* Easy model building with less formal statistical knowledge required.
* Higher accuracy can reduces errors in wrong prediction of result.
* Easy user interface with straight forward prediction.

Disadvantages:-

* The several factors of the patient needs to be accurately found out before any prediction of result.
* Clinical interpretations of model parameters is diﬃcult.

1. Applications:-

* Can predict the results of the patient using the inputs provided within short duration of time.

1. Conclusion:-

* Best alternative to state whether the patient has CKD positive or negative.
* Further improvement is required.

1. Future Scope:-

* Trying and experimenting with other classiﬁers and diﬀerent regression techniques to improve the result.
* This model can predict the outcome with many different inputs within seconds. The model will save a lot of time to cure within the initial stage of CKD.

1. Biblography:-

<https://www.kaggle.com/mansoordaku/ckdisease?select=kidney_disease.csv>

-Data from kidney\_disease.csv dataset.

1. Appendix:-

1. Source Code for Data Preprocessing:-

# import the libraries

import pandas as pd

import numpy as np

# Reading the dataset

dataset = pd.read\_csv(r"C:\Users\rupak\Downloads\Rsip2020-Chronic Kidney Disease(CKD)\dataset.csv")

#Taking Care of Missing Data

dataset.isnull().any()

dataset['age'].fillna(dataset['age'].median(),inplace=True)

dataset['Blood Pressure'].fillna(dataset['Blood Pressure'].mode()[0],inplace=True)

dataset['Specific Gravity'].fillna(dataset['Specific Gravity'].median(),inplace=True)

dataset['Albumin'].fillna(dataset['Albumin'].mode()[0],inplace=True)

dataset['Sugar'].fillna(dataset['Sugar'].median(),inplace=True) dataset['Red Blood Cells'].fillna(dataset['Red Blood Cells'].mode()[0],inplace=True)

dataset['Pus Cell'].fillna(dataset['Pus Cell'].mode()[0],inplace=True)

dataset['Pus Cell clumps'].fillna(dataset['Pus Cell clumps'].mode()[0],inplace=True)

dataset['Bacteria'].fillna(dataset['Bacteria'].mode()[0],inplace=True)

dataset['Blood Glucose Random'].fillna(dataset['Blood Glucose Random'].median(),inplace=True)

dataset['Blood Urea'].fillna(dataset['Blood Urea'].mode()[0],inplace=True)

dataset['Serum Creatinine'].fillna(dataset['Serum Creatinine'].median(),inplace=True)

dataset['Sodium'].fillna(dataset['Sodium'].mode()[0],inplace=True)

dataset['Potassium'].fillna(dataset['Potassium'].median(),inplace=True)

dataset['Hemoglobin'].fillna(dataset['Hemoglobin'].mode()[0],inplace=True)

dataset['Packed Cell Volume'].unique()

p=dataset['Packed Cell Volume'].mode()[0]

p

dataset['Packed Cell Volume'].replace(['\t?','\t43'],[41,41],inplace=True)

dataset['Packed Cell Volume'].fillna(dataset['Packed Cell Volume'].mode()[0],inplace=True)

dataset['White Blood Cell Count'].unique()

p=dataset['White Blood Cell Count'].mode()[0]

p

dataset['White Blood Cell Count'].replace(['\t?','\t8400'],[9800,9800],inplace=True)

dataset['White Blood Cell Count'].fillna(dataset['White Blood Cell Count'].mode()[0],inplace=True)

dataset['Red Blood Cell Count'].unique()

q=dataset['Red Blood Cell Count'].mode()[0]

q

dataset['Red Blood Cell Count'].replace(['\t?'],[5.2],inplace=True)

dataset['Red Blood Cell Count'].fillna(dataset['Red Blood Cell Count'].mode()[0],inplace=True)

dataset['Hypertension'].fillna(dataset['Hypertension'].mode()[0],inplace=True)

dataset['Appetite'].fillna(dataset['Appetite'].mode()[0],inplace=True)

dataset['Pedal Edema'].fillna(dataset['Pedal Edema'].mode()[0],inplace=True)

dataset['Anemia'].fillna(dataset['Anemia'].mode()[0],inplace=True)

dataset.isnull().any()

#Label Encoder

from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()

dataset['Red Blood Cells'] = le.fit\_transform(dataset['Red Blood Cells'])

dataset['Pus Cell'] = le.fit\_transform(dataset['Pus Cell'])

dataset['Pus Cell clumps'] = le.fit\_transform(dataset['Pus Cell clumps'])

dataset['Bacteria'] = le.fit\_transform(dataset['Bacteria'])

dataset['Hypertension'] = le.fit\_transform(dataset['Hypertension'])

dataset['Appetite'] = le.fit\_transform(dataset['Appetite'])

dataset['Pedal Edema'] = le.fit\_transform(dataset['Pedal Edema'])

dataset['Anemia'] = le.fit\_transform(dataset['Anemia'])

x = dataset.iloc[:,0:22].values

y = dataset.iloc[:,22].values

x.shape

#One Hot Encoding

from sklearn.preprocessing import OneHotEncoder

one = OneHotEncoder()

c= one.fit\_transform(x[:,19:20]).toarray()

x = np.delete(x,19,axis = 1)

x

x = np.concatenate((c,x),axis = 1)

x.shape

y.shape

#Splitting data into train and test

from sklearn.model\_selection import train\_test\_split

x\_train,x\_test,y\_train,y\_test = train\_test\_split(x,y, test\_size =0.2,random\_state = 0)

#Feature Scaling

from sklearn.preprocessing import StandardScaler

sc = StandardScaler()

x\_train = sc.fit\_transform(x\_train)

x\_test = sc.fit\_transform(x\_test)

#Data Visualization

import matplotlib.pyplot as plt

%matplotlib inline

plt.plot(x\_train,y\_train,'g')

plt.xlabel("this is x\_train")

plt.ylabel("this is y\_train")

plt.title("Chronic Kidney Disease")

plt.plot(dataset['Pedal Edema'],dataset['Class'],'g')

plt.xlabel("Pedal Edema")

plt.ylabel("Anemia")

plt.title("Chronic Kidney Disease")

plt.bar(dataset['Pedal Edema'],dataset['Class'],color = "green")

plt.xlabel("x-axis")

plt.ylabel("y-axis")

fig = plt.figure(figsize = (10,10))

ax1 = fig.add\_subplot(2,2,1)

ax2 = fig.add\_subplot(2,2,2)

ax3 = fig.add\_subplot(2,2,3)

ax1.plot(dataset['Pedal Edema'],dataset['Class'])

ax2.bar(dataset['Pedal Edema'],dataset['Class'])

ax3.scatter(dataset['Pedal Edema'],dataset['Class'])

import seaborn as sns

sns.scatterplot(x = "Hemoglobin",y = "Sugar" ,data = dataset )

sns.lineplot(x="Blood Pressure",y="Hypertension",data = dataset)

sns.distplot(dataset['Blood Glucose Random'])

sns.barplot(x = "Pus Cell",y = "Pus Cell clumps",data = dataset)

sns.boxplot(x = "Albumin",y="Sugar", data = dataset)

sns.catplot(x="White Blood Cell Count",y="Hemoglobin",data = dataset)

sns.catplot(x="White Blood Cell Count",y="Hemoglobin",hue = 'Class',data = dataset)

sns.heatmap(dataset.corr(),annot = True)

1. Source Code for Multi-Linear Regression:-

from sklearn.linear\_model import LinearRegression

mlr = LinearRegression()

mlr.fit(x\_train,y\_train)

y\_pred = mlr.predict(x\_test)

y\_pred

x\_test

from sklearn.metrics import r2\_score

accuracy = r2\_score(y\_test,y\_pred)

accuracy

mlr.predict([[0,0,0,0,1,0,1,0,0,0,0,0,1,0,0,45,70,0,1,1,0,0,120,50,4,110,4,10,40,8000,4,0,1,1]])

1. Source Code for Data Tree Classifier:-

from sklearn.preprocessing import StandardScaler

sc = StandardScaler()

x\_train = sc.fit\_transform(x\_train)

x\_test = sc.fit\_transform(x\_test)

from sklearn.tree import DecisionTreeClassifier

dtc = DecisionTreeClassifier (random\_state = 0)

dtc.fit(x\_train,y\_train)

import pickle

pickle.dump(dtc,open('decisiontreeclassifier.pkl','wb'))

y\_pred = dtc.predict(x\_test)

y\_pred

y\_test

from sklearn.metrics import accuracy\_score

accuracy\_score(y\_pred,y\_test)

from sklearn.metrics import confusion\_matrix

cm = confusion\_matrix(y\_test,y\_pred)

cm

import sklearn.metrics as metrics

fpr,tpr , threshold = metrics.roc\_curve(y\_test,y\_pred)

roc\_auc = metrics.auc(fpr,tpr)

import matplotlib.pyplot as plt

plt.title("roc")

plt.plot(fpr,tpr,'b',label = 'auc = %0.2f'%roc\_auc)

plt.legend(loc = 'lower right')

plt.plot([0,1],[0,1],'r--')

plt.xlim([0,1])

plt.ylim([0,1])

plt.ylabel('tpr')

plt.xlabel('fpr')

y = dtc.predict(sc.transform([[0,0,1,40,70,0,4,3,1,1,0,0,120,50,4,110,4,10,40,8000,4,0,1,1]]))

y

1. Source code for HTML file:-

<html>

<head>

</head>

<style>

body {

background-image: url("https://moscowfamilyeye.com/wp-content/uploads/2018/03/Medical-Researchers-in-a-Lab.jpeg");

background-repeat: no-repeat;

background-attachment: fixed;

background-size: cover;

}

</style>

<body>

<center>

<h1><b style="color:brown">Chronic Kidney Disease Prediction Application</b></h1>

<form action = "/login" method = "post">

<label for = "Appetite">Choose the Appetite condition</label>

<select name ="ap">

<option value ="good">good</option>

<option value ="poor">poor</option>

<option value ="moderate">moderate</option>

<select/>

<br>

<p> Enter age</p>

<p><input type = "text" name = "ag"/></p>

<p> Enter blood pressure</p>

<p><input type = "text" name = "bp"/></p>

<p> Enter specific gravity</p>

<p><input type = "text" name = "sg"/></p>

<p> Enter Albumin value</p>

<p><input type = "text" name = "al"/></p>

<p> Enter sugar value</p>

<p><input type = "text" name = "su"/></p>

<p>Choose RBC condition</p>

<p><input type = "text" name = "rbc"/></p>

<p>Choose Pus Cell condition</p>

<p><input type = "text" name = "pc"/></p>

<p>Choose Pus Cell clamps</p>

<p><input type = "text" name = "pcc"/></p>

<p>Bacteria</p>

<p><input type = "text" name = "b"/></p>

<p> Enter Blood Glucose Random value</p>

<p><input type = "text" name = "bgr"/></p>

<p> Enter Blood Urea value</p>

<p><input type = "text" name = "bu"/></p>

<p> Enter Serum Creatinine value</p>

<p><input type = "text" name = "sc"/></p>

<p> Enter Sodium value</p>

<p><input type = "text" name = "so"/></p>

<p> Enter Potassium value</p>

<p><input type = "text" name = "po"/></p>

<p> Enter Hemoglobin value</p>

<p><input type = "text" name = "he"/></p>

<p> Enter Packed Cell Volume value</p>

<p><input type = "text" name = "pcv"/></p>

<p> Enter White Blood Cell Count</p>

<p><input type = "text" name = "wbcc"/></p>

<p> Enter Red Blood Cell Count</p>

<p><input type = "text" name = "rbcc"/></p>

<p>Hypertension</p>

<p><input type = "text" name = "hy"/></p>

<p>Pedal Edema</p>

<p><input type = "text" name = "pe"/></p>

<p>Anemia</p>

<p><input type = "text" name = "an"/></p>

<p><input type="Submit" name="Submit" style="background-color:Green"/></p>

</center>

</form>

<b>{{label}}</b>

</body>

</html>

1. Source code for Flask application:-

from flask import Flask , render\_template ,request

import pickle

app = Flask(\_\_name\_\_)

model = pickle.load(open('decisiontreeclassifier.pkl','rb'))

@app.route('/')

def hello\_world():

return render\_template('index.html')

@app.route('/login', methods = ["POST"])

def login():

ap= request.form["ap"]

if(ap== "good"):

ap1,ap2,ap3 = 1,0,0

if(ap== "moderate"):

ap1,ap2,ap3 = 0,1,0

if(ap== "poor"):

ap1,ap2,ap3 = 0,0,1

ag = request.form["ag"]

bp = request.form["bp"]

sg = request.form["sg"]

al = request.form["al"]

su = request.form["su"]

rbc = request.form["rbc"]

pc = request.form["pc"]

pcc = request.form["pcc"]

b = request.form["b"]

bgr = request.form["bgr"]

bu = request.form["bu"]

sc = request.form["sc"]

so = request.form["so"]

po = request.form["po"]

he = request.form["he"]

pcv = request.form["pcv"]

wbcc = request.form["wbcc"]

rbcc = request.form["rbcc"]

hy = request.form["hy"]

pe = request.form["pe"]

an = request.form["an"]

total = [[ap1,ap2,ap3,int(ag),int(bp),float(sg),int(al),int(su),int(rbc),int(pc),int(pcc),int(b),int(bgr),int(bu),float(sc),int(so),float(po),float(he),int(pcv),int(wbcc),float(rbcc),int(hy),int(pe),int(an)]]

q = model.predict(total)

print(q)

if(q[0]==0):

mesg="negative"

if(q[0]==1):

mesg="positive"

return render\_template('index.html',label="The result is:"+str(mesg))

if \_\_name\_\_=='\_\_main\_\_':

app.run(debug = True)