INTERNSHIP PROJECT REPORT

Submitted to

VISVESVARAYA TECHONOLOGICAL UNIVERSITY

“Jnana Sangama”, Belagavi- 590 018

A PROJECT REPORT ON

“HEART DESEASE PREDICTION”

Under the Guidance of

Mrs. Kanimozhi G

Submitted by

VANDANA MOHAN GOUDA 4SU17CS115

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING SDM INSTITUTE OF TECHNOLOGY

(Affiliated to VTU, Belagavi and Approved by AICTE, New Delhi)

UJIRE - 574 240, KARNATAKA, INDIA.

CERTIFICATE

This is to certify that the **Project Work** entitled **"HEART DESEASE PREDICTION"** is a bonafide work carried out by **VANDANA MOHAN GOUDA 4SU17CS115**.student of Final year B.E, in partial fulfilment for the award of degree of **Bachelor of Engineering in ComputerScience and Engineering** of the **Visvesvaraya Technological University,Belagavi**. During the academic year **2020-2021**.It is certified that all corrections and suggestions indicated for **Internship** have been incorporated in the report. The project report has been approved, as it satisfies the academic requirements in respect of Project work prescribed for the said degree.

ABSTRACT

Over the last decade heart disease is the main reason for death in the world. Almost one person dies of Heart disease about every minute in India alone. In order to lower the number of deaths from heart diseases, there has to be a fast and efficient detection technique. Decision Tree is one of the effective data mining methods till this date. The algorithm used in this project is namely are Decision Tree, k-nearest neighbours algorithm (KNN).Heart disease defines several healthcare conditions that are vast in nature which is related to the heart and has many basic causes that affect the entire body. The existing data of heart disease patients from heart dataset taken from kaggle is used to make a test and clearance to the performance of decision tree and KNN algorithms.These datasets consist of 303 instances and 14 attributes.This study's goal is to predict the heart disease in single patient and multipatients.In this project, of machine learning algorithm for predicting the heart diseases of a person. The dataset for this project is manually take the values for single patients and automatically taken from the dataset values for multipatients. With this project, we have shown that the raw data from the heart dataset actually increases the accuracy of the model using decision tree and KNN machine learning algorithm.

TABLE OF CONTENTS:

Title Names

WiseTech Source Pvt. Ltd

About **WiseTech Source Pvt. Ltd**

WiseTech Source Pvt. Ltd is a new venture of Wise group, specializing in Offshore Development Centre (ODC) for companies who want to reduce costs without spending time on searching and hiring Skilled Professionals and Project Management. At WiseTech Source, we build software development professionals with modern tech stacks. We open new paths for Job seekers, Educational Institutions, Professionals and Industries with our diverse opportunities. We fulfill Tech requirements with our best-in- class enhancement platform and the expertise of Software Developers.

Our Services :

Software Development Nurturing Program

WiseTech Source Aims To Build A Strong Career Foundation With Excellent Environment, Skilled Technical Experts And Hands-On Experience.

Institutions

We Help Pre-Graduates Upgrade to Leading Technology Skills and Proficiency for High-Growth Careers in Software Industry.

Corporate

Platform At WiseTech Source, We match your tech requirement with ready to deploy professionals and or automation software solutions

We Offer a Full Range of Software Solutions, Products and Services!

Our products simplify the process and aid the customers to effectively manage and reduce operational costs.

We enhance productivity and thereby the company's revenue

-We integrate complex models in our apps and process using Artificial Intelligence

-Machine learning, Natural Language Processing, Robotics, Neural networks and forming the backbone to our agile innovations

-We extensively use .Net Core, MVC, Angular, Java, Python which form the building blocks

-We use fastest evolving Business Intelligence and Data Visualization tools like Tableau and R

Our Mission: To develop superior skilled workforce and generate ample opportunities Whether you're a Job-seeker, an employer, or an educational Institution, you can help us in our mission to Empower Job- seekers and Generate Ample Workforce Opportunities. Together we can fill IT Jobs by 2025 ...20%

Our Vision: WiseTech Source is an Offshore Development Center who pursue to bridge new opportunities for Young Grads, Professionals, Educational Institutions and and Corporates.

BASIC STUDY

The dataset consists of desease and non desease patients data.The details are as follows:

Total Dataset:303 instances and 14 attributes Train

size: 70% from total

Test size: 30%

First we need to load all libraries before proceeding further

INTRODUCTION

In this project, we have presented a system which is suitable for real-time heart diseases prediction and can be used by the single users and multiple users who have heart disease. Different from many other systems it is able to Predict the heart desease.The diagnosis system of the system is able to predict the heart disease by using ML. I am going to use Decision tree classifier and KNN for a Machine Learning Algorithm to classify data into desease or no desease classes. In this file am using the dataset taken from kaggle directly and the data needs to be processed and converted to arrays for final dataset to work on it! More than 1 million cases per year (India), data always required. So am using the multiple patients data directly which in this case it’s indeed and also in the perspective of specialist too. Well there are more than 120+ types in this kept is to classify desease or no desease. The datasets consists of 303 instances and 14 attributes .algorithms and the prediction results are based on the heart disease dataset instance.we ran experiments with some popular algorithms like KNN and Decision Tree.The experiment was carried out with the holdout test and the accuracyof the proposed system was 84% achieved with the KNN.

PROBLEM STATEMNT

Heart disease can be managed effectively with a combination of lifestyle changes, medicine and, in some cases, surgery. With the right treatment, the symptoms of heart disease can be reduced and the functioning of the heart improved. The predicted results can be used to prevent and thus reduce cost for surgical treatment and other expensive.The ‘target’ dataset has 2 class labels ‘0’ and ‘1’.here 0 represents the no desease and 1 represents the desease.as we can see from above Given the data, we need to classify them into one of these classes.

PROBLEM SOULTION

The overall objective of my work will be to predict accurately with few tests and attributes the presence of heart disease. Attributes considered form the primary basis for tests and give accurate results more.Here we will classify the dataset based on the prediction of data. When user selects an heart dataset this project gives the results as whether it is have desease or no desease .

CODE

######################### Project Code ################

```
import pymysql
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from tkinter import PhotoImage
from tkinter import filedialog
import os
from tkinter import messagebox,ttk
import tkinter.font as font
import tkinter as tk
from tkinter import Label
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
import seaborn as sns
from sklearn.metrics import confusion_matrix
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
from sklearn.metrics import precision_score,recall_score,f1_score
import re
from pandastable import Table,TableModel
from sklearn.ensemble import ExtraTreesClassifier


fln = None
class Toplevel1:
   def cancel(self):
      msg = messagebox.askyesno("Login Portal ","Are you sure, you want
to cancel login?")
      if msg:
         root1.destroy()

   def login(self):
      name = self.Entry1.get()
      passwd = self.Entry2.get()
      if name == "" and passwd == "":
         messagebox.showinfo("Login portal","Please Enter your Username
and Password!!")
      try:
         con =
pymysql.connect(host='localhost',user='root',password="root",database='
test')
         cur=con.cursor()
         cur.execute("select * from register where email='"+name+"';")
      row=cur.fetchone()
      print(row,"row")
      if name!="":
        if row:
          cur.execute("select password from register where email
='"+name+"';")
          pas=cur.fetchone()
          print(pas)
          con.commit()
          if passwd in pas:
            messagebox.showinfo("Login portal","The Login is
successful!")
            root1.configure(command=s())
          if passwd not in pas:
            messagebox.showerror("Login portal","Invalid Password!!")
        else:
          messagebox.showerror("Login portal","Invalid Username!!")
        con.close()
    except Exception as e:
      e

  def clear(self):
    self.email.delete(0,END)
    self.password.delete(0,END)
    self.answer.delete(0,END)
    self.cmb_quest.current(0)

  def insert(self):
    pattern = r"([\w\.-]+)@([\w\.]+)(\.[\w\.]+)"
    if not re.match(pattern,self.email.get()):
      messagebox.showerror("error","Invalid email!!")
      self.clear()
    flag = 0
    while self.email.get():
      passw = self.password.get()
      if passw =="":
        messagebox.showinfo("info","Password connot b emty!")
        break
      elif (len(passw)<8):
        flag = -1
        messagebox.showinfo("info","Password length shd't b less than
8!")
        break
      elif not re.search("[a-z]", passw):
        flag = -1
```

```
            messagebox.showinfo("info","should contain atleat one small
letter!")
            break
         elif not re.search("[A-Z]", passw):
            flag = -1
            messagebox.showinfo("info","should contain atleat one capital
letter!")
            break
         elif not re.search("[0-9]", passw):
            flag = -1
            messagebox.showinfo("info","should contain atleat one number!")
            break
         elif self.answer.get() == "":
            messagebox.showinfo("info","select your question and fill the
answer")
            flag=-1
            break
         else:
            flag = 1
            messagebox.showinfo("info","Good Password!")
            break
         if flag==-1:
            messagebox.showwarning("warning!","please enter strong
password contains..a-zA-Z0-9")
            self.clear()
      if flag == 1:
         try:
            con =
pymysql.connect(host='localhost',user='root',password="root",database='
test')
            cur=con.cursor()
            cur.execute("select * from register where
email='"+self.email.get()+"';")
            notvalid = cur.fetchone()
            flag1=1
            if notvalid:
               messagebox.showerror("Error","User already Exists!!")
               flag1 = 0
               self.clear()
         except e:
            e
         if flag1:
            sql = "INSERT INTO register (email, password,answer) VALUES
(%s, %s,%s)"
            val = (self.email.get(),self.password.get(),self.answer.get())
            vv=cur.execute(sql,val)
            print("1",vv)
            con.commit()
            con.close()
            messagebox.showinfo("info","registration successfull!!\nThank
You now you can proceed to login")
            self.clear()




   def registration(self):
      self.root3 = Toplevel()
      self.root3.title("Registration Form")
      self.root3.resizable(0,0)
      self.root3.geometry("350x500+480+150")
      self.root3.configure(bg='white')
      self.root3.focus_force()
      self.root3.grab_set()

      t=Label(self.root3,text="Registration",font=("times new
roman",20,"bold"),bg="white",fg="red").place(x=0,y=10,relwidth=1)


      question = Label(self.root3,text="Security Question",font=("times new
roman",15,"bold"),bg="white",fg="black").place(x=50,y=260)

      self.cmb_quest = ttk.Combobox(self.root3,font=("times new
roman",13),state='readonly',justify=CENTER)
      self.cmb_quest['values']=("Select","Your Pet Name","Your Birth
Place","Your Best Friend","Your Fantasy")
      self.cmb_quest.place(x=50,y=290,width=250)
      self.cmb_quest.current(0)

      answer = Label(self.root3,text="Answer",font=("times new
roman",15,"bold"),bg="white",fg="black").place(x=50,y=340)
      self.answer = Entry(self.root3,font=("times new
roman",15),bg='white')
      self.answer.place(x=50,y=370,width=250)


      email = Label(self.root3,text="Email",font=("times new
roman",15,"bold"),bg="white",fg="black").place(x=50,y=100)
      self.email = Entry(self.root3,font=("times new roman",15),bg='white')
      self.email.place(x=50,y=130,width=250)
```

```
        password = Label(self.root3,text="Password",font=("times new
roman",15,"bold"),bg="white",fg="black").place(x=50,y=180)
        self.password = Entry(self.root3,show="*",font=("times new
roman",15),bg='white')
        self.password.place(x=50,y=210,width=250)

#       global flag=0

        btn_change =
Button(self.root3,text="Register",bg="green",fg="white",font=("times new
roman",15,"bold"),command=self.insert).place(x=130,y=410)

    def reset(self):
        flag=0
        while True:
            passw = self.new_password.get()
            if passw =="":
                messagebox.showinfo("info","Password connot b emty!")
                break
            elif (len(passw)<8):
                flag = -1
                messagebox.showinfo("info","Password length shd't b less than
8!")
                break
            elif not re.search("[a-z]", passw):
                flag = -1
                messagebox.showinfo("info","should contain atleat one small
letter!")
                break
            elif not re.search("[A-Z]", passw):
                flag = -1
                messagebox.showinfo("info","should contain atleat one capital
letter!")
                break
            elif not re.search("[0-9]", passw):
                flag = -1
                messagebox.showinfo("info","should contain atleat one number!")
                break
            elif self.txt_answer.get() == "":
                messagebox.showinfo("info","select your question and fill the
answer")
                flag=-1
                break
            elif self.txt_answer.get()!="":
```

```
            flag=-1
            con = 
pymysql.connect(host='localhost',user='root',password="root",database='
test')
            cur=con.cursor()
            cur.execute("select answer from register where answer 
='"+self.txt_answer.get()+"';")
            row = cur.fetchone()
            if row:
               flag = 1
               messagebox.showinfo("info","Good Password!")
               break
            else:
               messagebox.showerror("Error","answer not matched!")
               self.txt_answer.delete(0,END)
               self.new_password.delete(0,END)
               self.cmb_quest.current(0)
            con.commit()
            con.close()
            break


      if flag == 1:
         try:
            con = 
pymysql.connect(host='localhost',user='root',password="root",database='
test')
            cur=con.cursor()
            cur.execute("update register SET 
password='"+self.new_password.get()+"' where email 
='"+self.Entry1.get()+"';")
            con.commit()
            con.close()
            self.txt_answer.delete(0,END)
            self.new_password.delete(0,END)
            self.cmb_quest.current(0)
            messagebox.showinfo("info","password updated 
successfully!!\nThank You now you can proceed to login")
         except Exception as e:
            e


   def forgot_password(self):
      con = 
pymysql.connect(host='localhost',user='root',password="root",database='
test')
        cur=con.cursor()
        cur.execute("select * from register where 
email='"+self.Entry1.get()+"';")
        row = cur.fetchone()
        con.commit()
        con.close()
        if row == None and self.Entry1.get()!="":
            messagebox.showerror("Error:","user doesn't exists!!")
            self.Entry1.delete(0,END)
            return
        if self.Entry1.get() == "":
            messagebox.showerror("Error","Please enter the valid username to 
reset your password!")
        else:
            self.root2 = Toplevel()
            self.root2.title("Forget Password")
            self.root2.geometry("350x400+480+150")
            self.root2.configure(bg='white')
            self.root2.focus_force()
            self.root2.grab_set()

            t=Label(self.root2,text="Forget Password",font=("times new 
roman",20,"bold"),bg="white",fg="red").place(x=0,y=10,relwidth=1)



            question = Label(self.root2,text="Security Question",font=("times 
new roman",15,"bold"),bg="white",fg="gray").place(x=50,y=100)

            self.cmb_quest = ttk.Combobox(self.root2,font=("times new 
roman",13),state='readonly',justify=CENTER)
            self.cmb_quest['values']=("Select","Your Pet Name","Your Birth 
Place","Your Best Friend","Your Fantasy")
            self.cmb_quest.place(x=50,y=130,width=250)
            self.cmb_quest.current(0)

            txt_answer = Label(self.root2,text="Answer",font=("times new 
roman",15,"bold"),bg="white",fg="gray").place(x=50,y=180)
            self.txt_answer = Entry(self.root2,font=("times new 
roman",15),bg='lightgray')
            self.txt_answer.place(x=50,y=210,width=250)

            new_password = Label(self.root2,text="New 
password",font=("times new 
roman",15,"bold"),bg="white",fg="gray").place(x=50,y=260)
```

```
        self.new_password = Entry(self.root2,show="* ",font=("times new roman",15),bg='lightgray')
        self.new_password.place(x=50,y=290,width=250)

        btn_change = Button(self.root2,text="Reset Password",command=self.reset,bg="green",fg="white",font=("times new roman",15,"bold")).place(x=90,y=340)


    def __init__(self, top=None):
        '''This class configures and populates the toplevel window.
           top is the toplevel containing window.'''
        _bgcolor = '#d9d9d9'  # X11 color: 'gray85'
        _fgcolor = '#000000'  # X11 color: 'black'
        _compcolor = '#d9d9d9' # X11 color: 'gray85'
        _ana1color = '#d9d9d9' # X11 color: 'gray85'
        _ana2color = '#ececec' # Closest X11 color: 'gray92'
        font14 = "-family {Showcard Gothic} -size 12"
        font15 = "-family {Showcard Gothic} -size 9"
        font18 = "-family {Segoe UI Symbol} -size 9 -weight bold"

        top.geometry("470x475+520+239")
        top.minsize(148, 1)
        top.maxsize(1924, 1055)
        top.resizable(False, False)
        top.title("Login Portal")
        top.configure(borderwidth="5")
        top.configure(background="orange")
        top.configure(cursor="arrow")

        self.Label1 = tk.Label(top)
        self.Label1.place(relx=0.106, rely=0.232, height=56, width=122)
        self.Label1.configure(background="orange")
        self.Label1.configure(disabledforeground="#a3a3a3")
        self.Label1.configure(font=font18)
        self.Label1.configure(foreground="#000000")
        self.Label1.configure(text='''Email''')

        self.Label2 = tk.Label(top)
        self.Label2.place(relx=0.106, rely=0.4, height=36, width=122)
        self.Label2.configure(background="orange")
        self.Label2.configure(disabledforeground="#a3a3a3")
        self.Label2.configure(font=font18)
        self.Label2.configure(foreground="#000000")
        self.Label2.configure(text='''Password''')
        self.Entry1 = tk.Entry(top)
        self.Entry1.place(relx=0.404, rely=0.253,height=34,
relwidth=0.37,width=50)
        self.Entry1.configure(background="orange")
        self.Entry1.configure(borderwidth="3")
        self.Entry1.configure(disabledforeground="#a3a3a3")
        self.Entry1.configure(font="TkFixedFont")
        self.Entry1.configure(foreground="#000000")
        self.Entry1.configure(insertbackground="black")

        self.Entry2 = tk.Entry(top,show="*")
        self.Entry2.place(relx=0.404, rely=0.4,height=34,
relwidth=0.37,width=50)
        self.Entry2.configure(background="orange")
        self.Entry2.configure(borderwidth="3")
        self.Entry2.configure(disabledforeground="#a3a3a3")
        self.Entry2.configure(font="TkFixedFont")
        self.Entry2.configure(foreground="#000000")
        self.Entry2.configure(insertbackground="black")

        self.Label3 = tk.Label(top)
        self.Label3.place(relx=0.0, rely=0.021, height=66, width=472)
        self.Label3.configure(background="orange")
        self.Label3.configure(disabledforeground="#a3a3a3")
        self.Label3.configure(font=font14)
        self.Label3.configure(foreground="#000000")
        self.Label3.configure(text='''Login''')

        self.Button1 = tk.Button(top)
        self.Button1.place(relx=0.213, rely=0.653, height=43, width=116)
        self.Button1.configure(activebackground="#ececec")
        self.Button1.configure(activeforeground="#000000")
        self.Button1.configure(background="orange")
        self.Button1.configure(borderwidth="5")
        self.Button1.configure(disabledforeground="#a3a3a3")
        self.Button1.configure(font=font15)
        self.Button1.configure(foreground="red")
        self.Button1.configure(highlightbackground="#d9d9d9")
        self.Button1.configure(highlightcolor="black")
        self.Button1.configure(pady="0")
        self.Button1.configure(text='''Cancel''')
        self.Button1.configure(command=self.cancel)

        self.Button2 = tk.Button(top)
        self.Button2.place(relx=0.596, rely=0.653, height=43, width=106)
        self.Button2.configure(activebackground="#ececec")
        self.Button2.configure(activeforeground="#000000")
        self.Button2.configure(background="orange")
        self.Button2.configure(borderwidth="5")
        self.Button2.configure(cursor="hand2")
        self.Button2.configure(disabledforeground="#a3a3a3")
        self.Button2.configure(font=font15)
        self.Button2.configure(foreground="green")
        self.Button2.configure(highlightbackground="#d9d9d9")
        self.Button2.configure(highlightcolor="black")
        self.Button2.configure(pady="0")
        self.Button2.configure(text='''Login''')
        self.Button2.configure(command=self.login)

        self.Button3 = tk.Button(top)
        self.Button3.place(relx=0.574, rely=0.547, height=23, width=166)
        self.Button3.configure(activebackground="orange")
        self.Button3.configure(activeforeground="#000000")
        self.Button3.configure(background="#FFFFFF")
        self.Button3.configure(borderwidth="0")
        self.Button3.configure(disabledforeground="#a3a3a3")
        self.Button3.configure(foreground="red")
        self.Button3.configure(highlightbackground="#FFFFFF")
        self.Button3.configure(highlightcolor="red")
        self.Button3.configure(pady="0")
        self.Button3.configure(text='''Forget Password?''')
        self.Button3.configure(command=self.forgot_password)

        self.Button4 = tk.Button(top)
        self.Button4.place(relx=0.15, rely=0.547, height=23, width=166)
        self.Button4.configure(activebackground="orange")
        self.Button4.configure(activeforeground="#000000")
        self.Button4.configure(background="#FFFFFF")
        self.Button4.configure(borderwidth="0")
        self.Button4.configure(disabledforeground="#a3a3a3")
        self.Button4.configure(foreground="blue")
        self.Button4.configure(highlightbackground="#FFFFFF")
        self.Button4.configure(highlightcolor="red")
        self.Button4.configure(pady="0")
        self.Button4.configure(text='''Register New Account?''')
        self.Button4.configure(command=self.registration)
```

```
def main():
   global val, w, root1
   root1 = tk.Tk()
   top = Toplevel1(root1)
   root1.mainloop()
def main1():
   q = messagebox.askyesnocancel('heart desease prediction',
message='Do You Wish To Logout!')
   if q:
      root.destroy()
      main()
   else:
      s()

main()

def New_window():
   new = Toplevel(root)
   new.geometry('350x350')

   new.configure(bg="plum1")
   new.title("More Info..")

   lbn = Label(new,text="Data
Information",background='plum4',justify='left',font=20,foreground='white
')
   lbn.config(font=("Times", 44))
   lbn.pack(pady=5,padx=5)
#    new.resizable(False,False)
   lb = tk.Label(new,text="New Windoww")
   lb.pack(side=tk.LEFT)
   lb.place(relx = 0.2,rely = 0.18,anchor ='nw')
   class TestApp(Frame):
      """Basic test frame for the table"""
      def __init__(self, parent=None):
         self.parent = parent
         Frame.__init__(self)
         self.main = new
         self.main.geometry('600x400+200+100')
         self.main.title('Table app')
         f = Frame(self.main)
         f.pack(fill=BOTH,expand=1)
         df = TableModel.getSampleData()
         pt = Table(f, dataframe=dat,
                         showtoolbar=True, showstatusbar=True)
```

```
        pt.show()
        return


    app = TestApp()
        #launch the app
    app.mainloop()

def create_plot():
    cm = confusion_matrix(y_test,pred1)

    plot, ax = plt.subplots(figsize=(11, 9))

    sns.heatmap(cm,cmap='YlGnBu',annot=True)

    return plot


def exit():
    v =messagebox.askokcancel(title='Message', message='Are Sure You
Want To Exit?')
    if v:
        root.destroy()

def clear_all():
    Entry1.delete(0,END)
    Entry2.delete(0,END)
    Entry3.delete(0,END)
    Entry4.delete(0,END)

def InsertBlob():
    name= Entry1.get()
    age = Entry2.get()
    gender=Entry3.get()
    date=Entry4.get()
    flag3=0
    while True:
        if fln == None:
            messagebox.showinfo("info","please include an image! ")
            break
        elif name == "":
            messagebox.showinfo("info","please enter the name it cannot be
empty ")
            break
        elif not re.match(r"[a-zA-Z]",name):
            flag3=-1
            messagebox.showinfo("info","please enter the name properly only
alphabets! pls")
            break
        elif age == "":
            messagebox.showinfo("info","please enter the age it cannot be
empty ")
            break
        elif not re.match(r"[0-9]*[0-9]+",age):
            flag3=-1
            messagebox.showinfo("info","please enter the age properly only
numbers! pls")
            break
        elif gender == "":
            messagebox.showinfo("info","please enter the gender it cannot be
empty ")
            break
        elif not re.match(r"[mf|MF]",gender):
            flag3=-1
            messagebox.showinfo("info","please enter the gender properly only
M|m for male and F|f for female pls")
            break
        elif date == "":
            messagebox.showinfo("info","please enter the date it cannot be
empty ")
            break
        elif not re.match(r"^(0?[1-9]|[12][0-9]|3[01])[\/\-](0?[1-
9]|1[012])[\/\-]\d{4}$",date):
            messagebox.showinfo("info","please enter the date correctly
(dd/mm/yyyy)")
            break
        else:
            flag3=1
            break

    def RetrieveBlob():
    sql = "select * from records;"
    cur.execute(sql)
    cur.commit()


def s():
    def mod():
        def heart_disease(new_input):
            df=pd.read_csv("heart.csv")
            df.count().isnull()
            df.describe()
            x= df.drop('target',axis=1)
            y=df['target']
            model= ExtraTreesClassifier()
            model.fit(x,y)
            feat_importances = pd.Series(model.feature_importances_,
index=x.columns)
            randomforest_classifier=
RandomForestClassifier(n_estimators=100)
            score=cross_val_score(randomforest_classifier,x,y,cv=15)
            acc=score.mean()
            op=model.predict(new_input)
            com=[op,acc]
            return com
        #root=Tk()
        #Label(root, text = 'HEART DESEASE PREDICTION', font =(
        #'USING MAHINE-LEARNING', 15)).pack(side = TOP, pady = 10)

    # Creating a photoimage object to use image


    # here, image option is used to
    # set image on button
        #Button(root, text = 'Click Me !', image = photo).pack(side = TOP)
        window = Tk()

    # Creating a photoimage object to use image


    # here, image option is used to
    # set image on button
        #Button(window, text = 'Click Me !', image = photo).pack(side = TOP)







        window.title("Predicting-Heart-Disease")
        window.geometry('650x500')
        #photo = PhotoImage(file = "HERAT.png")
        lbl = Label(window, text="Enter the following parameters")
        lbl.grid(column=2, row=0)
```

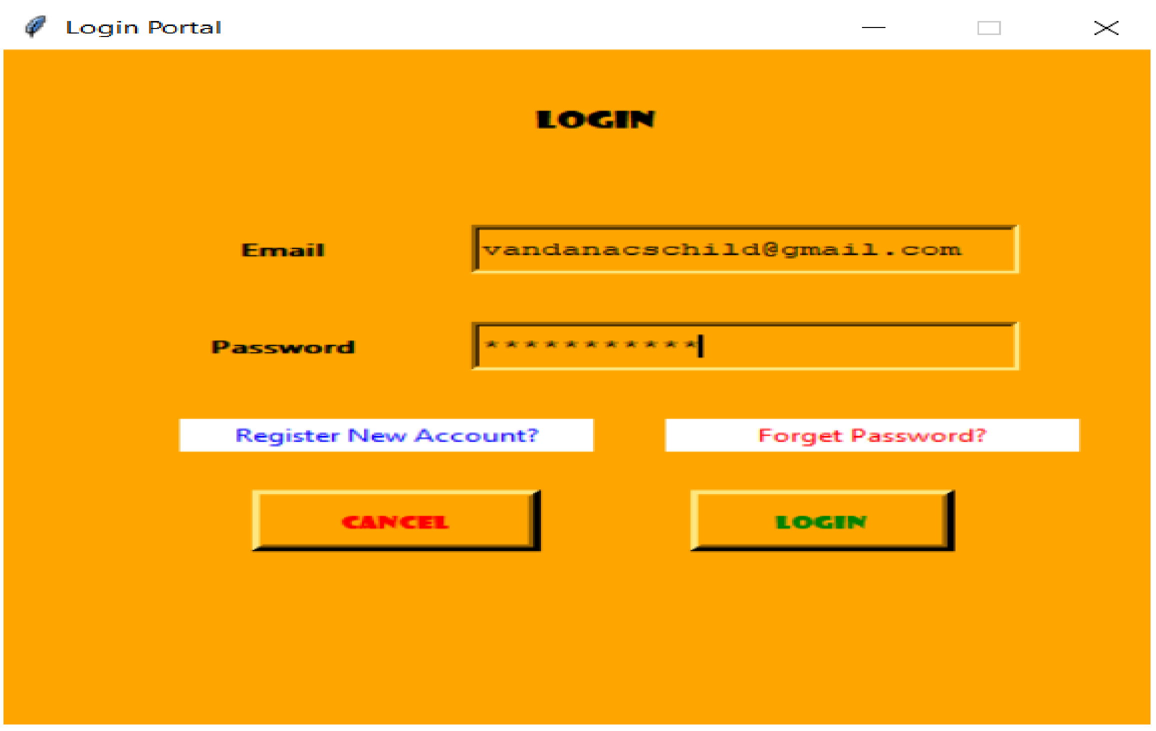
```
    lbl = Label(window, text=" AGE")
    lbl.grid(column=1, row=1)
    age = Entry(window,width=20)
    age.grid(column=3, row=1)
    age.focus()


    lbl = Label(window, text=" SEX (m=1,f=0)")
    lbl.grid(column=1, row=2)
    sex = Entry(window,width=20)
    sex.grid(column=3, row=2)
    sex.focus()



    lbl = Label(window, text=" chest pain level (0-3)")
    lbl.grid(column=1, row=3)
    cp = Entry(window,width=20)
    cp.grid(column=3, row=3)
    cp.focus()



    lbl = Label(window, text=" resting blood pressure  (in mm Hg )")
    lbl.grid(column=1, row=4)
    bps = Entry(window,width=20)
    bps.grid(column=3, row=4)
    bps.focus()



    lbl = Label(window, text="cholestoral in mg/dl")
    lbl.grid(column=1, row=5)
    chol = Entry(window,width=20)
    chol.grid(column=3, row=5)
    chol.focus()

    lbl = Label(window, text="(fasting blood sugar > 120 mg/dl) (1 = true;
0 = false)")
    lbl.grid(column=1, row=6)
    fbs = Entry(window,width=20)
    fbs.grid(column=3, row=6)
    fbs.focus()
    lbl = Label(window, text="resting electrocardiographic results")
    lbl.grid(column=1, row=7)
    ecg = Entry(window,width=20)
    ecg.grid(column=3, row=7)
    ecg.focus()

    lbl = Label(window, text="maximum heart rate achieved")
    lbl.grid(column=1, row=8)
    mhr = Entry(window,width=20)
    mhr.grid(column=3, row=8)
    mhr.focus()

    lbl = Label(window, text="exercise induced angina (1 = yes; 0 = no)")
    lbl.grid(column=1, row=9)
    eia = Entry(window,width=20)
    eia.grid(column=3, row=9)
    eia.focus()

    lbl = Label(window, text="ST depression induced by exercise relative
to rest")
    lbl.grid(column=1, row=10)
    st = Entry(window,width=20)
    st.grid(column=3, row=10)
    st.focus()

    lbl = Label(window, text="the slope of the peak exercise ST segment")
    lbl.grid(column=1, row=11)
    slope = Entry(window,width=20)
    slope.grid(column=3, row=11)
    slope.focus()

    lbl = Label(window, text="number of major vessels (0-3) colored by
flourosopy")
    lbl.grid(column=1, row=12)
    ca = Entry(window,width=20)
    ca.grid(column=3, row=12)
    ca.focus()

    lbl = Label(window, text=" A blood disorder called thalassemia \n (3 =
normal; 6 = fixed defect; 7 = reversable defect)")
    lbl.grid(column=1, row=13)
    thal = Entry(window,width=20)
    thal.grid(column=3, row=13)
    thal.focus()
    def clicked():

new_input=[age.get(),sex.get(),cp.get(),bps.get(),chol.get(),fbs.get(),ecg.get
(),mhr.get(),eia.get(),st.get(),slope.get(),ca.get(),thal.get()]
    #    listToStr = ' ,'.join(map(str, new_input))
    #    inp.configure(text= "you entered: "+listToStr )
        new_input= np.reshape(new_input,(1,-1))
        d=heart_disease(new_input)
        ans=d[0]
  #        score=cross_val_score(randomforest_classifier,x,y,cv=15)
        sc=str(d[1]*100)

        if(ans==1):
            s='you have a heart Disease'
        else:
            s="you are in perfect health"
        ot.configure(text="Result: " + s + "\n(accuracy ="+ sc +"% )" )



    btn = Button(window, text="submit",command=clicked)
    btn.grid(column=3, row=15)


    ot = Label(window, text="",font=("Arial Bold", 10))
    ot.grid(column=1, row=17)


    window.mainloop()


  def getthefile():
    global fln,dat
    fln = filedialog.askopenfilename(initialdir=os.getcwd(),title="Select
Image File",filetypes=(("Microsoft Office Excel Comma Separated Values
File","*.csv"),("ALL Files","*.*")))
    dat = pd.read_csv(fln)
    head, tail = os.path.split(fln)

    lbl2 = Label(root,text='"'+tail+'file is selected! from "'+head+'" path in
your system! \n \n "',background='#d9d9d9')
    lbl2.place(relx=0.114, rely=0.324, height=25, width=42)

    lbl2.pack(pady=11)
```

```
   root1.destroy()
   global root
   root = Tk()
   frm = Frame(root,background='white')
   frm.pack(side=BOTTOM,padx=15,pady=15)

   f = Frame(root,background='black')
   f.pack(side=TOP,padx=10,pady=10)

   global myFont
   myFont= font.Font(family='Helvetica', size=10, weight='bold')


   ##
   _bgcolor = '#d9d9d9'  # X11 color: 'gray85'
   _fgcolor = '#000000'  # X11 color: 'black'
   _compcolor = '#d9d9d9' # X11 color: 'gray85'
   _ana1color = '#d9d9d9' # X11 color: 'gray85'
   _ana2color = '#ececec' # Closest X11 color: 'gray92'
   font12 = "-family {Yu Gothic} -size 19 -underline 1"
   font13 = "-family {Segoe UI} -size 19"
   font14 = "-family {Showcard Gothic} -size 12"
   font15 = "-family {Showcard Gothic} -size 9"
   font18 = "-family {Segoe UI Symbol} -size 9 -weight bold"

   Label1 = tk.Label(root)
   Label1.place(relx=0.307, rely=0.086, height=171, width=742)
   Label1.configure(background="#d9d9d9")
   Label1.configure(borderwidth="20")
   Label1.configure(disabledforeground="#a3a3a3")
   Label1.configure(font=font12)
   Label1.configure(foreground="#000000")
   Label1.configure(highlightcolor="#646464646464")
   Label1.configure(text='''Heart Disease Prediction''')

#    ##

   btn4 =
Button(frm,text="LOGOUT",command=main1,cursor='hand2',height=3,wi
dth=20)
   btn4['font']=myFont
   btn4.pack(side=tk.RIGHT,padx=10)
   btn = Button(frm,text="Multiple
Prediction",command=getthefile,cursor='hand2',height=3,width=20)
   btn['font']=myFont
   btn.pack(side=tk.LEFT,padx=10)

   #btn2 = Button(frm,text="Single
Prediction",command=mod,cursor='hand2',height=3,width=20)
   #btn2['font']=myFont
   #btn2.pack(side=tk.LEFT,padx=10)



   btn3 = Button(frm,text="File
View",command=New_window,cursor='hand2',height=3,width=20)
   btn3['font']=myFont
   btn3.pack(side=tk.LEFT,padx=10)

   def process():
      try:
         if True:
            global y_test,pred1,pred
            x= data.drop('target',axis=1)
            y=data['target']
            model= ExtraTreesClassifier()
            model.fit(x,y)
            feat_importances = pd.Series(model.feature_importances_,
index=x.columns)
            randomforest_classifier=
RandomForestClassifier(n_estimators=100)
#              x_train , x_test , y_train , y_test =
train_test_split(data.drop(['target'],axis=1,),data['target'],test_size=0.2,ran
dom_state=0)
#              tr = RandomForestClassifier(n_estimators=100)
#              tr.fit(x_train,y_train)
#              pred1 = tr.predict(x_test)
            pred =  model.predict(dat)
            print(pred)
            print(pred.ndim)
            dat['res'] = pred
            print(dat)
            target_names = ['nodisease','disease']
            dat['res'] = dat.res.apply(lambda x :target_names[x])
   #              global score
   #              score = tr.score(x_test,y_test)*100
   #              lbl = tk.Label(root,text="prediction :"+str(int(pred))+"\n
accuracy
:"+str(round(score,2)),justify='center',foreground='white',background='gr
een')
   #            lbl.config(font=("Times", 44))
   #            lbl.pack(pady=10,padx=10)
   #            lbl.place(relx = 0.3,
   #                     rely = 0.8,
   #                     anchor ='sw')
            messagebox.showwarning(title='FYI', message='Succesfully
Predicted Please Look into MoreInfo...for update!')
            fln = None

      except Exception as e:
         print(e)
         messagebox.showwarning(title='FYI', message='Please insert
data!\n   Go to Broswer')


   btn =
Button(frm,text="Predict",command=process,cursor='hand2',height=3,wi
dth=20)
   btn['font']=myFont
   btn.pack(side=tk.TOP,pady=10,padx=10)




   root.title("HEART DESEASE PREDICTION")
   root.geometry("500x500")
   root.configure(bg='#d9d9d9')
   root.mainloop()


   heart=pd.read_csv("heart1.csv")
   heart

 info = ["age","1: male, 0: female","chest pain type, 1: typical angina, 2:
atypical angina, 3: non-anginal pain,       4: asymptomatic","resting blood
pressure"," serum cholestoral in mg/dl","fasting blood sugar > 120
mg/dl","resting electrocardiographic results (values 0,1,2)"," maximum
heart rate achieved","exercise induced angina","oldpeak = ST depression
induced by exercise relative to rest","the slope of the peak exercise ST
segment","number of major vessels (0-3) colored by flourosopy","thal: 3 =
normal; 6 = fixed defect; 7 = reversable defect"]
```

```
for i in range(len(info)):
print(heart.columns[i]+":\t\t\t"+info[i])

heart['target']

heart.groupby('target').size()

heart.groupby('target').sum()

heart.shape

heart.size

heart.describe()

heart.info()

heart.hist(figsize=(14,14))
plt.show()

plt.bar(x=heart['sex'],height=heart['age'])
plt.show()

numeric_columns=['trestbps','chol','thalach','age','oldpeak']
sns.pairplot(heart[numeric_columns])

#Storing in X and y

X,y=heart.loc[:,:'thal'],heart.loc[:,'target']
X
Y

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
X=heart.drop(['target'],axis=1)
X

X_train,X_test,y_train,y_test=train_test_split(X,y,random_state=10,test_size
=0.3,shuffle=True)
X_test
y_test

Catagory=['No....but i pray you get Heart Disease or at leaset Corona Virus
Soon...','Yes you have Heart Disease....RIP in Advance']
```

```
from sklearn.tree import DecisionTreeClassifier
dt=DecisionTreeClassifier(max_depth=7)
dt.fit(X_train,y_train)

prediction=dt.predict(X_test)
accuracy_dt=accuracy_score(y_test,prediction)*100
accuracy_dt

print("Accuracy on training set: {:.3f}".format(dt.score(X_train, y_train)))
print("Accuracy on test set: {:.3f}".format(dt.score(X_test, y_test)))
y_test
prediction
X_DT=np.array([[52 ,1, 1,128,205,1,1,184,0,0.0,2,0,2]])
X_DT_prediction=dt.predict(X_DT)
X_DT_prediction[0]
print(Catagory[int(X_DT_prediction[0])])

# KNN
#data prrocesing
sc=StandardScaler().fit(X_train)
X_train_std=sc.transform(X_train)
X_test_std=sc.transform(X_test)
X
X_test_std

from sklearn.neighbors import KNeighborsClassifier
knn=KNeighborsClassifier(n_neighbors=4)#here n neibhour used to
increase the accuracyy
knn.fit(X_train_std,y_train)
prediction_knn=knn.predict(X_test_std) #30% x-test 70% trainstd
accuracy_knn=accuracy_score(y_test,prediction_knn)*100
accuracy_knn
print("Accuracy on training set: {:.3f}".format(knn.score(X_train, y_train)))
print("Accuracy on test set: {:.3f}".format(knn.score(X_test, y_test)))
data=pd.read_csv('heart.csv')
data.head()
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3)#20% test
data remaining 80% train data
x_train.head()
k_range=range(1,26) #explan:range 1 to 26 how much neibhour i want
scores={}
scores_list=[]

for k in k_range:
    knn=KNeighborsClassifier(n_neighbors=k)
    knn.fit(X_train_std,y_train)
    prediction_knn=knn.predict(X_test_std)
    scores[k]=accuracy_score(y_test,prediction_knn)#actual n predicted
data
    scores_list.append(accuracy_score(y_test,prediction_knn))
scores

X_knn=np.array([[52 ,1, 1,128,205,1,1,184,0,0.0,2,0,2]])
X_knn_std=sc.transform(X_knn)
X_knn_prediction=dt.predict(X_knn)
X_knn_std
(X_knn_prediction[0])
print(Catagory[int(X_knn_prediction[0])])

algorithms=['Decision Tree','KNN']
scores=[accuracy_dt,accuracy_knn]
sns.set(rc={'figure.figsize':(15,7)})
plt.xlabel("Algorithms")
plt.ylabel("Accuracy score")
sns.barplot(algorithms,scores)
```

RESULT

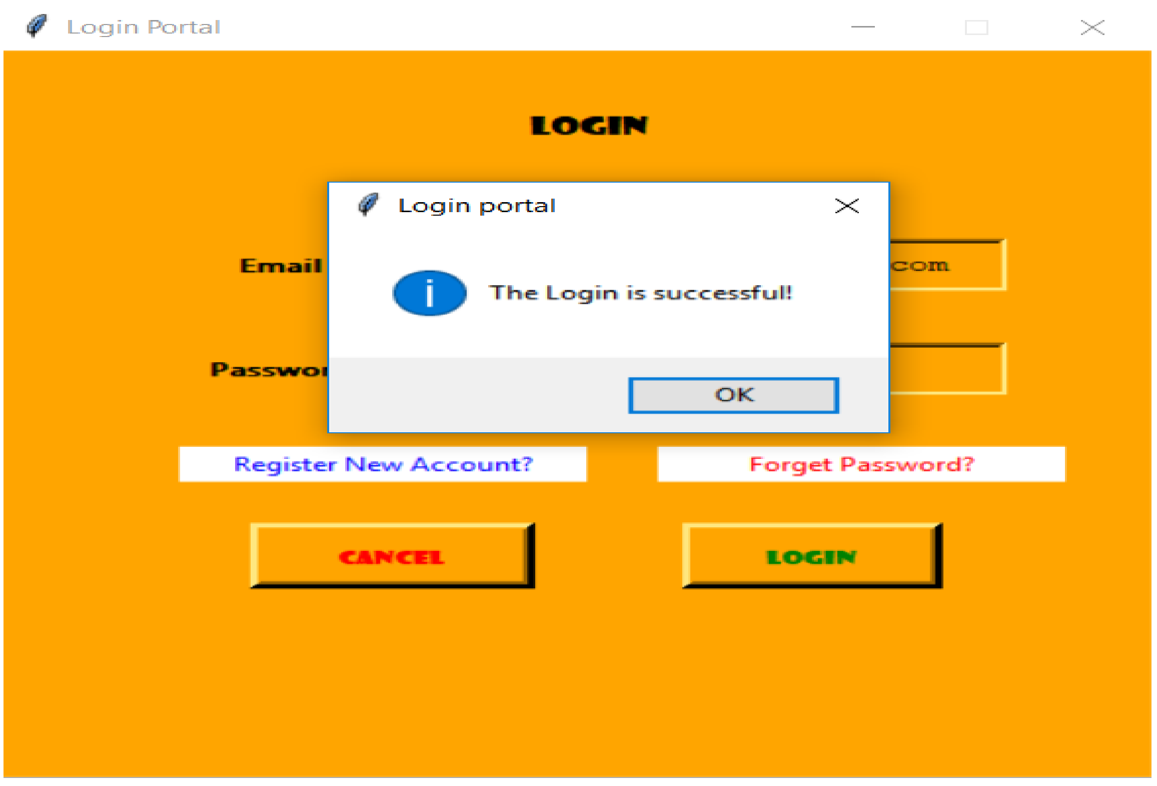

Login Portal

LOGIN

Email vandanacschild@gmail.com

Password ************

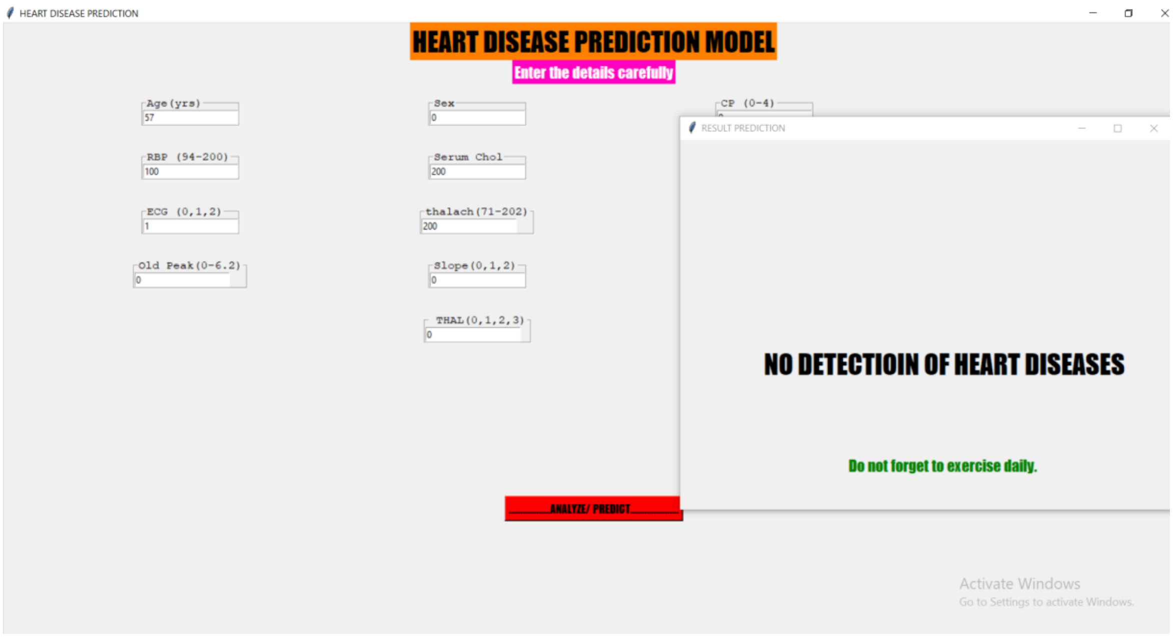
Login page


Login Portal

LOGIN

Login portal

The Login is successful!



HEART DISEASE PREDICTION

HEART DISEASE PREDICTION MODEL

Enter the details carefully

Age (yrs): 57	Sex: 0	CP (0-4)
RBP (94-200): 100	Serum Chol: 200	
ECG (0,1,2): 1	thalach (71-202): 200	
Old Peak (0-6.2): 0	Slope (0,1,2): 0	
	THAL (0,1,2,3): 0	

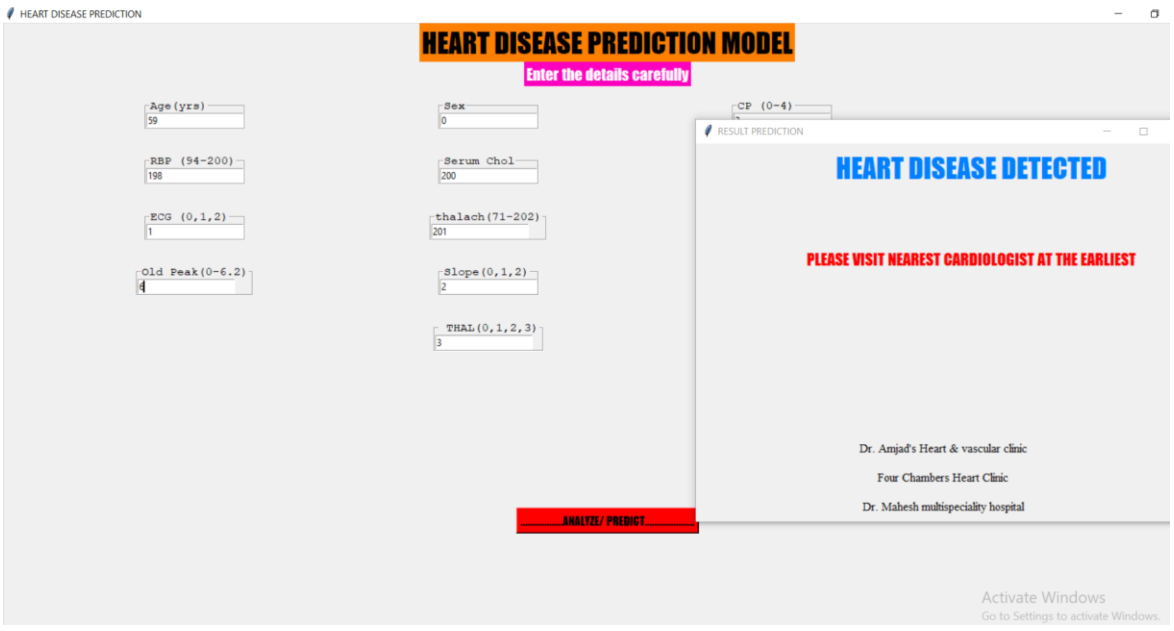
RESULT PREDICTION

NO DETECTIOIN OF HEART DISEASES

Do not forget to exercise daily.

Activate Windows
Go to Settings to activate Windows.

For single patient



HEART DISEASE PREDICTION

HEART DISEASE PREDICTION MODEL

Enter the details carefully

Age (yrs): 59	Sex: 0	CP (0-4)
RBP (94-200): 198	Serum Chol: 200	
ECG (0,1,2): 1	thalach (71-202): 201	
Old Peak (0-6.2): 6	Slope (0,1,2): 2	
	THAL (0,1,2,3): 3	

RESULT PREDICTION

HEART DISEASE DETECTED

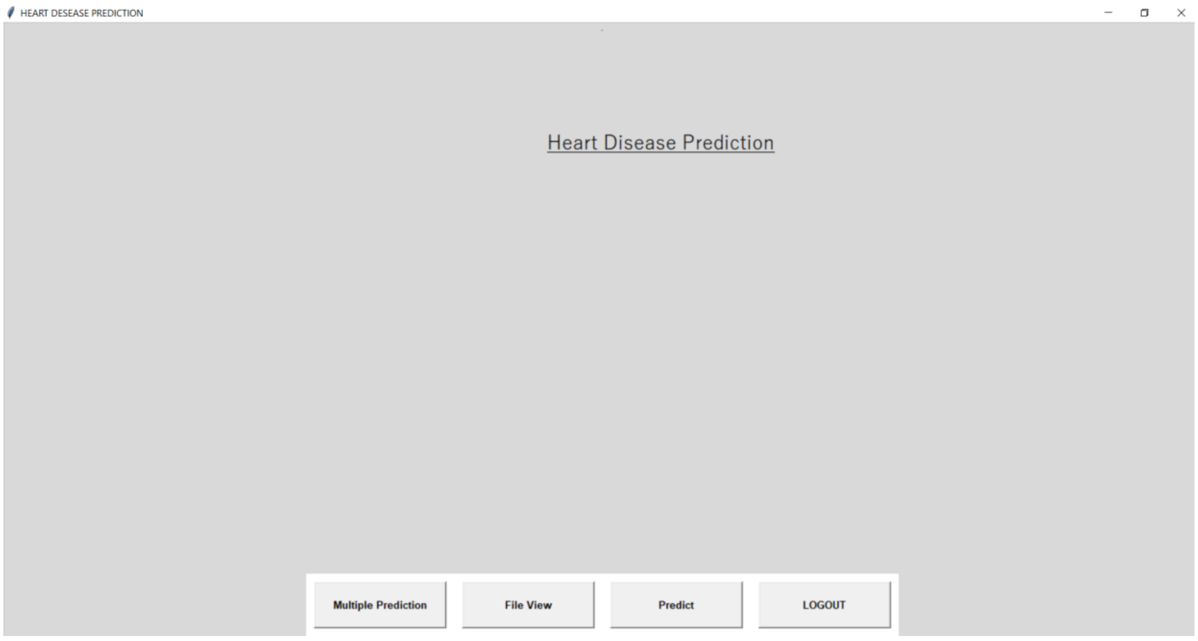
PLEASE VISIT NEAREST CARDIOLOGIST AT THE EARLIEST

Dr. Amjad's Heart & vascular clinic

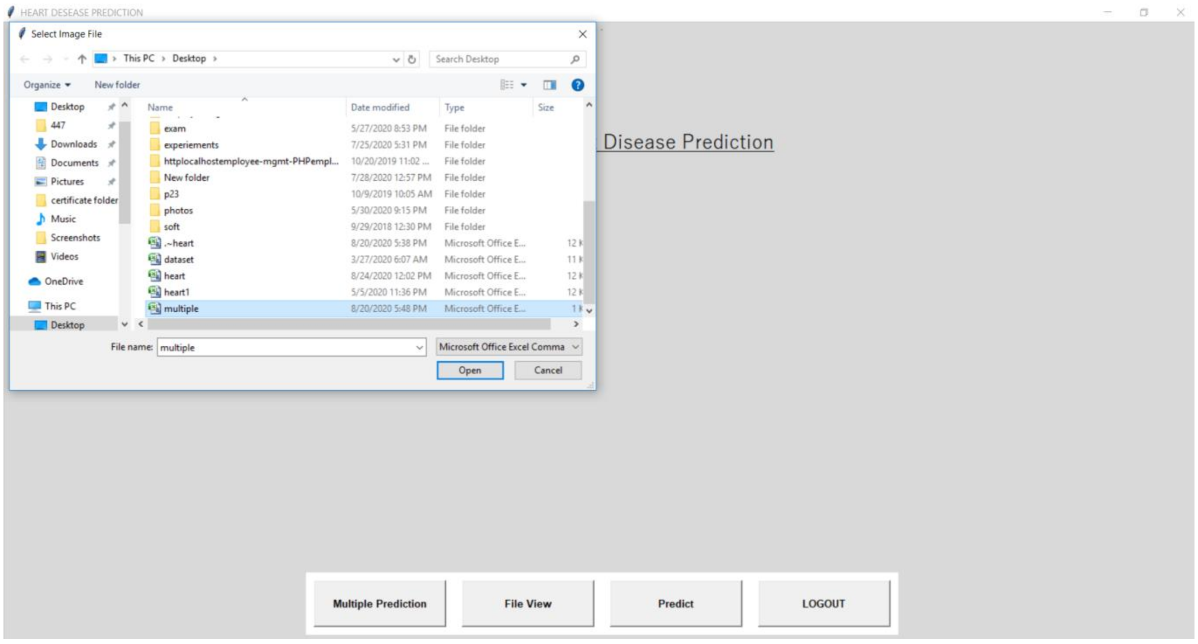
Four Chambers Heart Clinic

Dr. Mahesh multispeciality hospital

Activate Windows
Go to Settings to activate Windows.



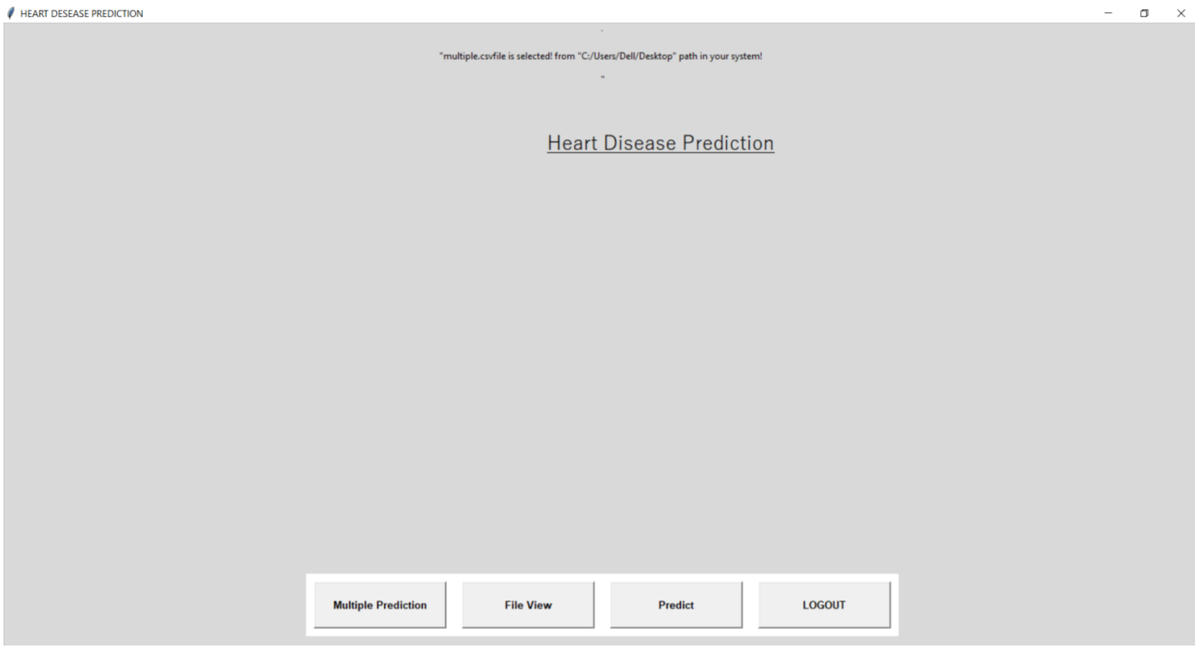
For Multiple Patients



Name	Date modified	Type	Size
exam	5/27/2020 8:53 PM	File folder	
experiements	7/25/2020 5:31 PM	File folder	
httplocalhostemployee-mgmt-PHPempl...	10/20/2019 11:02 ...	File folder	
New folder	7/28/2020 12:57 PM	File folder	
p23	10/9/2019 10:05 AM	File folder	
photos	5/30/2020 9:15 PM	File folder	
soft	9/29/2018 12:30 PM	File folder	
.~heart	8/20/2020 5:38 PM	Microsoft Office E...	12 K
dataset	3/27/2020 6:07 AM	Microsoft Office E...	11 K
heart	8/24/2020 12:02 PM	Microsoft Office E...	12 K
heart1	5/5/2020 11:36 PM	Microsoft Office E...	12 K
multiple	8/20/2020 5:48 PM	Microsoft Office E...	1 K

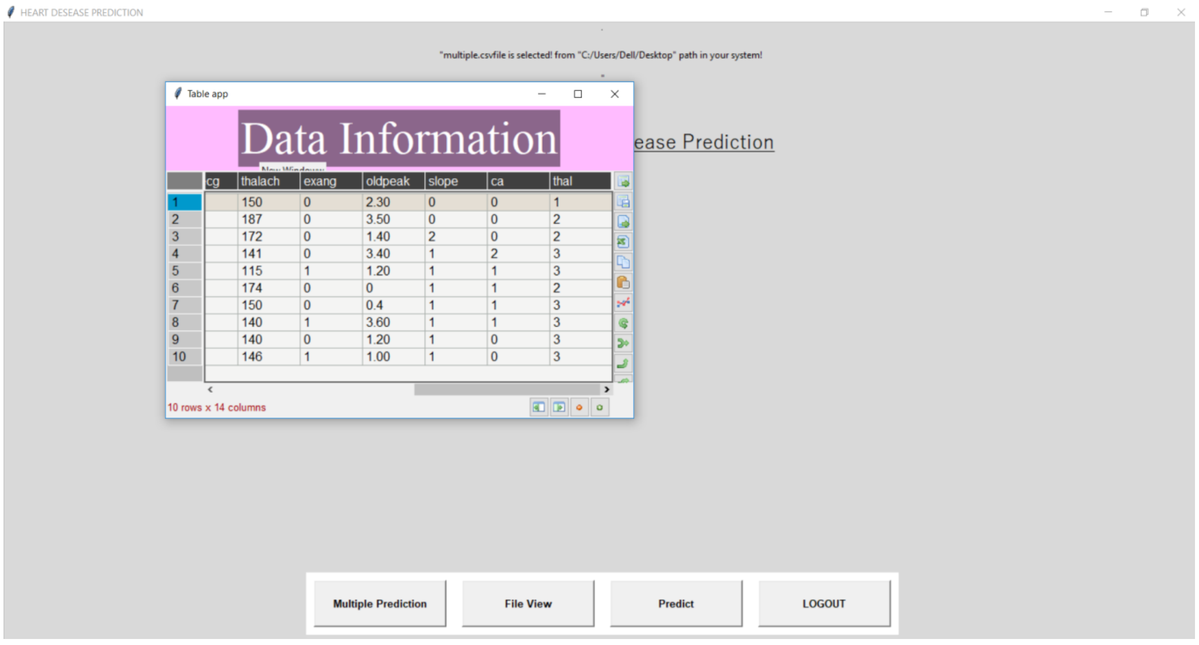
File name: multiple

Microsoft Office Excel Comma


HEART DESEASE PREDICTION

"multiple.csvfile is selected! from "C:/Users/Dell/Desktop" path in your system!

Heart Disease Prediction


HEART DESEASE PREDICTION

"multiple.csvfile is selected! from "C:/Users/Dell/Desktop" path in your system!

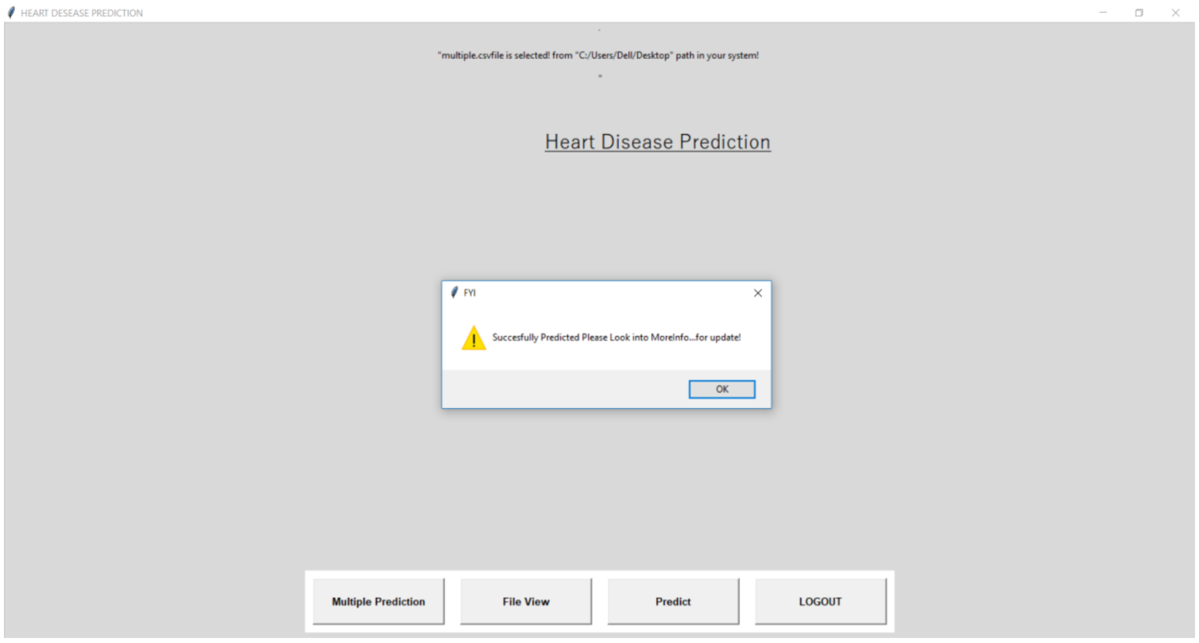
ease Prediction

Table app

Data Information

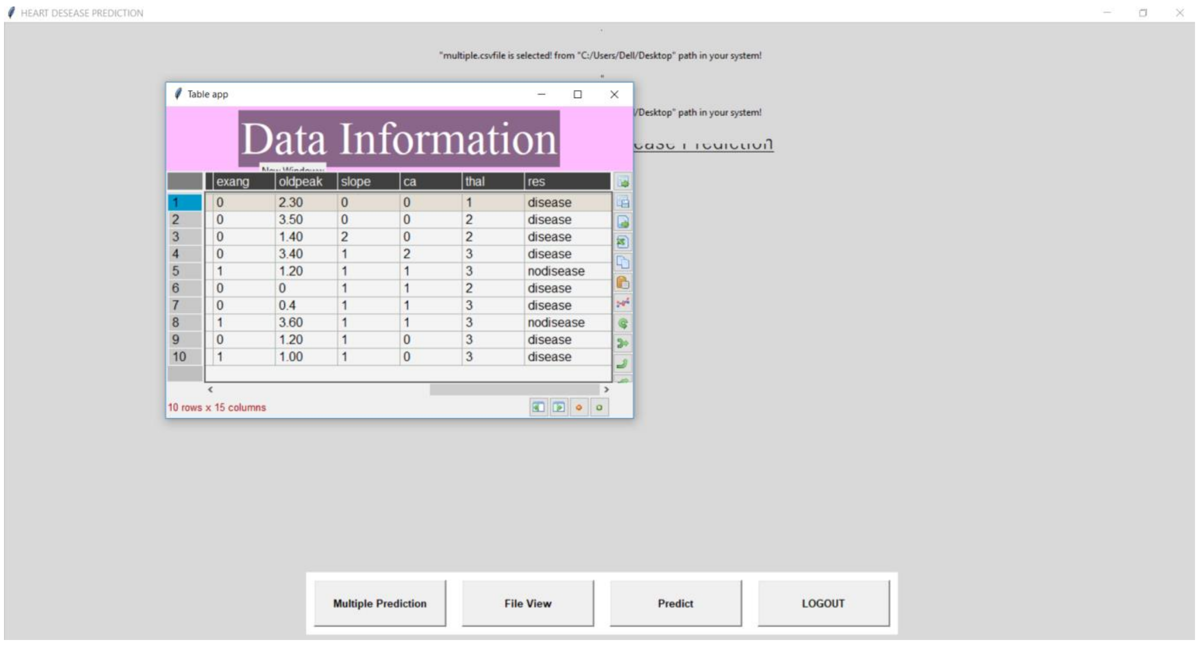
	cg	thalach	exang	oldpeak	slope	ca	thal
1		150	0	2.30	0	0	1
2		187	0	3.50	0	0	2
3		172	0	1.40	2	0	2
4		141	0	3.40	1	2	3
5		115	1	1.20	1	1	3
6		174	0	0	1	1	2
7		150	0	0.4	1	1	3
8		140	1	3.60	1	1	3
9		140	0	1.20	1	0	3
10		146	1	1.00	1	0	3

10 rows x 14 columns



"multiple.csvfile is selected! from "C:/Users/Dell/Desktop" path in your system!

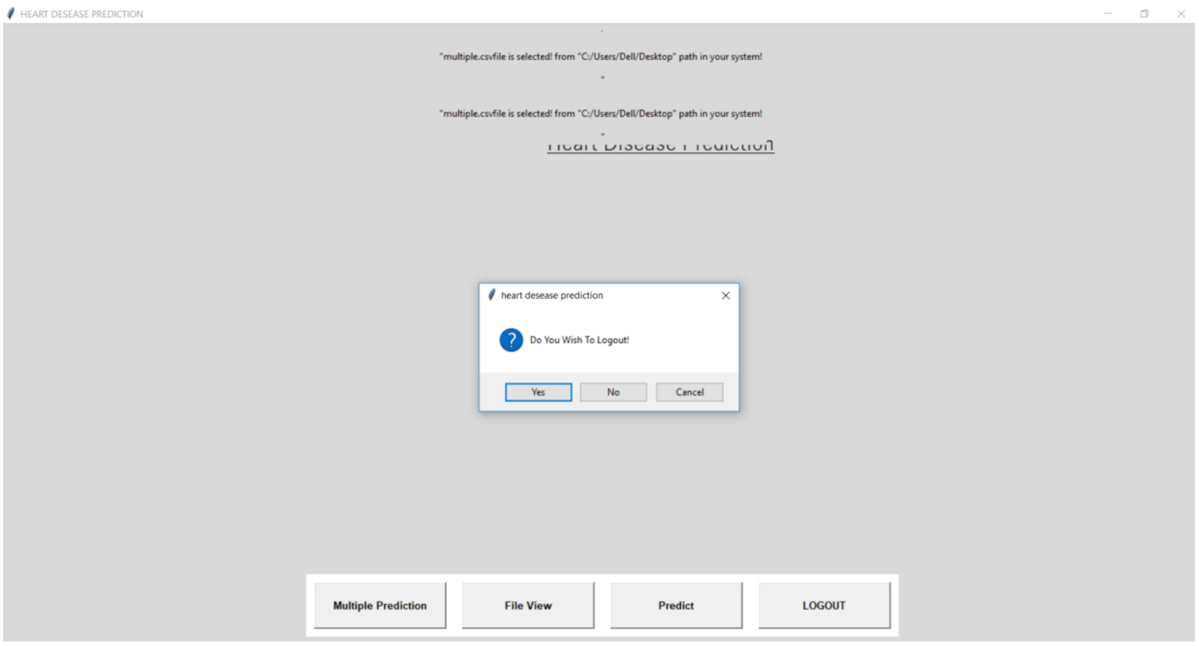
Succesfully Predicted Please Look into MoreInfo...for update!



"multiple.csvfile is selected! from "C:/Users/Dell/Desktop" path in your system!

	exang	oldpeak	slope	ca	thal	res
1	0	2.30	0	0	1	disease
2	0	3.50	0	0	2	disease
3	0	1.40	2	0	2	disease
4	0	3.40	1	2	3	disease
5	1	1.20	1	1	3	nodisease
6	0	0	1	1	2	disease
7	0	0.4	1	1	3	disease
8	1	3.60	1	1	3	nodisease
9	0	1.20	1	0	3	disease
10	1	1.00	1	0	3	disease

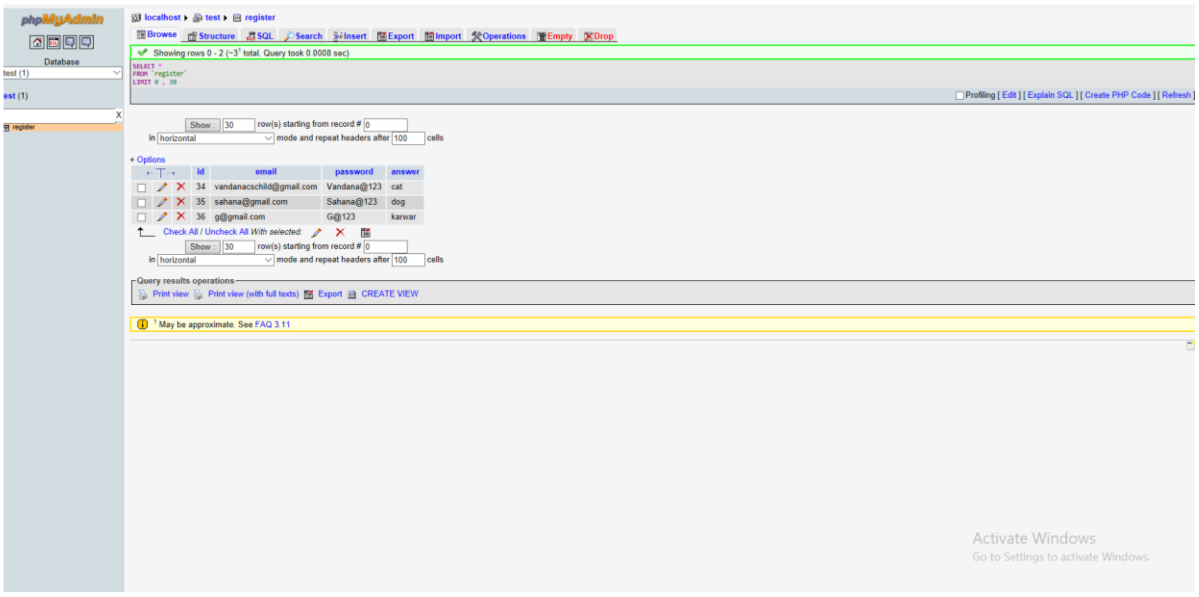
10 rows x 15 columns



"multiple.csvfile is selected! from "C:/Users/Dell/Desktop" path in your system!

"multiple.csvfile is selected! from "C:/Users/Dell/Desktop" path in your system!

Do You Wish To Logout!



Showing rows 0 - 2 (~3[1] total, Query took 0.0008 sec)

```
SELECT *
FROM `register`
LIMIT 0 , 30
```

Profiling [Edit] [Explain SQL] [Create PHP Code] [Refresh]

Show : 30 row(s) starting from record # 0
in horizontal mode and repeat headers after 100 cells

+ Options

id	email	password	answer
34	vandanacschild@gmail.com	Vandana@123	cat
35	sahana@gmail.com	Sahana@123	dog
36	g@gmail.com	G@123	karwar

Check All / Uncheck All *With selected:*

Show : 30 row(s) starting from record # 0
in horizontal mode and repeat headers after 100 cells

Query results operations
Print view Print view (with full texts) Export CREATE VIEW

[1] May be approximate. See FAQ 3.11

Db connectivity

CONCLUSION AND SCOPE FOR THE FUTURE WORK.

In a nutshell, this internship has been an excellent and rewarding experience. I can conclude that there have been a lot I've learnt from my work. Needless to say, the technical aspects of the work I've done are not flawless and could be improved provided enough time. As someone with no prior experience with ML whatsoever I believe my time spent in research and discovering it was well worth it and contributed to finding an acceptable solution to build a fully functional application.Two main things that I've learned the importance of are time-management skills and self motivation.Heart disease is the leading cause of death in the world over the past 10 years (World Health Organization 2007). The European Public Health Alliance reported that heart attacks, strokes and other circulatory diseases account for 41% of all deaths (European Public Health Alliance 2010). Several different symptoms areassociated with heart disease, which makes it difficult to diagnose it quicker and better. Working on heart disease patients databases can be compared to real-life application. Doctors knowledge to assign the weight to each attribute. More weight is assigned to the attribute having high impact on disease prediction. Therefore it appears reasonable to try utilizing the knowledge and experience of several specialists collected in databases towards assisting the Diagnosis process. It also provides healthcare professionals an extra source of knowledge for making decisions.. This project presents a new model that enhances the KNN accuracy in identifying heart disease patients. Clinical decisions are often made based on doctors' intuition and experience rather than on the knowledge rich data hidden in the database.This practice leads to unwanted biases, errors and excessive medical costs which affects the quality of service provided to patients.AI will proposed that integration of clinical decision support with computer-based patient records could reduce medical errors, enhance patient safety, decrease unwanted practice variation, and improve patient outcome. This suggestion is promising as data modelling have the potential to generate a knowledge-rich environment.In the future improve the accuracy level using other Ml algorithm around 86-95%.but we have more features in dataset so we can go for deep learning techniques.In future its helpful for peoples who are suffering from heart desease.as there would consider the each and every data and learn the same and help us to judge with loss too.

REFFERENCES

https://www.researchgate.net/publication/308160606_Heart_Disease_Prediction_System_Using_Data_Mining_and_Hybrid_Intelligent_Techniques_A_Review

https://www.researchgate.net/publication/334612815_HEART_DISEASE_PREDICTION_SYSTEM_HDPS

https://www.geeksforgeeks.org/machine-learning/

https://www.geeksforgeeks.org/python-gui-tkinter/

https://www.zdnet.com/article/what-is-machine-learning-everything-you-need-to-know/

45