DIABETES PREDICTION AND PATIENT CLUSTERING

PROJECT REPORT

FOR

Data Mining

Embedded Project

WINTER SEMESTER – 2018-19

Submitted by:

Gaurav Kumar Singh – 16BCE0242

VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

SCOPE

VELLORE INSTITUTE OF TECHNOLOGY

VELLORE – 632014 TAMIL NADU

INDIA

Abstract

Diabetes is one of the deadliest diseases in the world. It is not only a disease but also creator of different kinds of diseases like heart attack, blindness etc. The normal identifying process is that patients need to visit a diagnostic center, consult their doctor, and sit tight for a day or more to get their reports. Cause of Diabetes vary depending on the genetic makeup, family history, ethnicity, health etc. Diabetes & pre-diabetes is diagnosed by blood test. The aim of this project is predict whether a person has diabetes or not and if the person is suffering from diabetes then cluster the subject according to the severity of the disease. The program will be helpful for doctors to identify patients suffering from diabetes. Moreover individuals too can find the outcome by giving inputs they know. Further predicting the disease early leads to treating the patient before it becomes critical.

Contents

1. Introduction

Diabetes mellitus (DM), commonly referred to as diabetes, is a group of metabolic disorders characterized by high blood sugar levels over a prolonged period. Symptoms of high blood sugar include frequent urination, increased thirst, and increased hunger. If left untreated, diabetes can cause many complications. Acute complications can include diabetic ketoacidosis, hyperosmolar hyperglycemic state, or death. Serious long-term complications include cardiovascular disease, stroke, chronic kidney disease, foot ulcers, and damage to the eyes. Diabetes is due to either the pancreas not producing enough insulin, or the cells of the body not responding properly to the insulin produced. As of 2017, an estimated 425 million people had diabetes worldwide, with type 2 DM making up about 90% of the cases. This represents 8.8% of the adult population, with equal rates in both women and men. Trend suggests that rates will continue to rise. Diabetes at least doubles a person's risk of early death. In 2017, diabetes resulted in approximately 3.2 to 5.0 million deaths. The global economic cost of diabetes related health expenditure in 2017 was estimated at US$727 billion. In the United States, diabetes cost nearly US$245 billion in 2012.

2. Problem Statement

Diabetes is one of the deadliest diseases in the world. It is not only a disease but also creator of different kinds of diseases like heart attack, blindness etc. The normal identifying process is that patients need to visit a diagnostic center, consult their doctor, and sit tight for a day or more to get their reports.

Problem Statement 1: Classify the patients into diabetic or non-diabetic

Problem Statement 2: Cluster diabetic patients into severe or not severe.

3. Aim of the Project

Given the problem statement, the aim of this project is to identify whether the patient has diabetes or not based on diagnostic measurements and if the patient is suffering from diabetes then cluster him/her according to the severity of the disease.

4. Literature Survey

4.1. Dataset Used

The dataset used has been obtained from UCI Machine Learning Repository having 769 records of Female Patients exclusively. This dataset is originally from the National Institute of Diabetes and Digestive and Kidney Diseases. The objective is to predict based on diagnostic measurements whether a patient has diabetes or not. This dataset has 769 samples of diabetic and healthy individuals. In particular, all patients here are females of at least 21 years of age. The diabetes dataset is credited to UCI machine

learning database repository. The dataset consist of 769 samples, out of which 500 are non-diabetic while 269 are diabetic people.

4.2. Attribute Details

The dataset has total 9 attributes out of which 8 are independent variables and one is the dependent variable i.e. target variable which determines whether patient is having diabetes or not. There are 8 independent variables:

1. **Pregnancies:** No. of times pregnant

2. **Glucose:** Plasma Glucose Concentration a 2 hour in an oral glucose tolerance test (mg/dl)

Plasma Glucose Test	Normal	Prediabetes	Diabetes
2 hour post-prandial	Below 7.8 mmol/l Below 140 mg/dl	7.8 to 11.0 mmol/l 140 to 199 mg/dl	11.1 mmol/l or more 200 /dl or more

A 2-hour value between 140 and 200 mg/dL (7.8 and 11.1 mmol/L) is called impaired glucose tolerance. This is called "pre-diabetes." It means you are at increased risk of developing diabetes over time. A glucose level of 200 mg/dL (11.1 mmol/L) or higher is used to diagnose diabetes.

3. **Blood Pressure: Diastolic Blood Pressure(mmHg)**
   If Diastolic B.P > 90 means High B.P (High Probability of Diabetes)

   Diastolic B.P < 60 means low B.P (Less Probability of Diabetes)

4. **Skin Thickness:** Triceps Skin Fold Thickness (mm):
   A value used to estimate body fat. Normal Triceps Skinfold Thickness in women is 23mm. Higher thickness leads to obesity and chances of diabetes increases.

5. **Insulin:** 2-Hour Serum Insulin (mu U/ml)

	Normal Insulin Level
2 Hours After Glucose	16-166 mIU/L

Values above this range can be alarming.

6. **BMI:** Body Mass Index (weight in kg/ height in m2) Body Mass Index of 18.5 to 25 is within the normal range

BMI between 25 and 30 then it falls within the overweight range. A BMI of 30 or over falls within the obese range.

7. **Diabetes Pedigree Function:** It provides information about diabetes history in relatives and genetic relationship of those relatives with patients. Higher Pedigree Function means patient is more likely to have diabetes.

8. **Age (years)**

The dependent variable is whether the patient is having diabetes or not.

9. **Outcome:** Class Variable (0 or 1) where '0' denotes patient is not having diabetes and '1' denotes patient having diabetes

5. Design and Implementation

5.1. Prediction/Classification

A model is made by combining 3 classification algorithms. Predicted outcome is taken from each classifier and the outcome in the best of three is taken as the final outcome like a voting mechanism. The 3 algorithms used are:

- **ID3 Decision Tree:** In decision tree learning, ID3 (Iterative Dichotomiser 3) is an algorithm invented by Ross Quinlan used to generate a decision tree from a dataset. ID3 is the precursor to the C4.5 algorithm, and is typically used in the machine learning and natural language processing domains. The ID3 algorithm is used by training on a data set to produce a decision tree which is stored in memory. At runtime, this decision tree is used to classify new test cases (feature vectors) by traversing the decision tree using the features of the datum to arrive at a leaf node. The class of this terminal node is the class the test case is classified as.

- **Naïve Bayes Classifier:** In machine learning, naive Bayes classifiers are a family of simple "probabilistic classifiers" based on applying Bayes' theorem with strong (naive) independence assumptions between the features. Naive Bayes classifiers are highly scalable, requiring a number of parameters linear in the number of variables (features/predictors) in a learning problem. Maximum-likelihood training can be done by evaluating a closed-form expression: which takes linear time, rather than by expensive iterative approximation as used for many other types of classifiers.

- **KNN Classifier:** In pattern recognition, the k-nearest neighbours algorithm (k-NN) is a non-parametric method used for classification and regression. In both cases, the input consists of the k closest training examples in the feature space. The output depends on whether k-NN is used for classification or regression. In k-NN classification, the output is a class membership. An object is classified by a plurality

vote of its neighbours, with the object being assigned to the class most common among its k nearest neighbours (k is a positive integer, typically small). If k = 1, then the object is simply assigned to the class of that single nearest neighbour.

5.2. Clustering

After the Classification is done and the outcome is Diabetes positive, then the subject is clustered into one of the 2 clusters:

1. Diabetes not severe at the moment
2. Diabetes severe and immediate medical help required

The clustering is only done by taking the attributes Glucose Level and insulin level into consideration since they are the most vital attributes for a diabetes patient.

- **K-Means Clustering:** K-means clustering is a method of vector quantization, originally from signal processing, that is popular for cluster analysis in data mining. K-means clustering aims to partition n observations into k clusters in which each observation belongs to the cluster with the nearest mean, serving as a prototype of the cluster. This results in a partitioning of the data space into cells. The problem is computationally difficult (NP-hard); however, efficient heuristic algorithms converge quickly to a local optimum. These are usually similar to the expectation-maximization algorithm for mixtures of Gaussian distributions via an iterative refinement approach employed by both k-means and Gaussian mixture modelling. They both use cluster centers to model the data; however, k-means clustering tends to find clusters of comparable spatial extent, while the expectation-maximization mechanism allows clusters to have different shapes.

5.3. GUI Interface

Rather than using a simple CUI program, a GUI interface has been used for collecting inputs and well as displaying the output. GUI interface for python can be implemented using tKinter. It is a standard Python interface to the Tk GUI toolkit shipped with Python. Python with tkinter outputs the fastest and easiest way to create the GUI applications.

5.4. Tools and Libraries Used Used

5.4.1 Spyder

Spyder is a powerful scientific environment written in Python, for Python, and designed by and for scientists, engineers and data analysts. It offers a unique combination of the advanced editing, analysis, debugging, and profiling functionality of a comprehensive development tool with the data exploration, interactive execution, deep inspection, and beautiful visualization capabilities of a scientific package.

5.4.2 Sklearn

Scikit-learn (formerly scikits.learn) is a free software machine learning library for the Python programming language. It features various classification, regression and clustering algorithms including support vector machines, random forests, gradient boosting, k-means and DBSCAN, and is designed to interoperate with the Python numerical and scientific libraries NumPy and SciPy.

Functions Used:

sklearn.cluster.KMeans(): Clustering using Kmeans method.

tree.DecisionTreeClassifier(): Classifying using decision tree method.

sklearn.neighbors.KNeighborsClassifier(): Classifying using knn method.

sklearn.naive_bayes.GaussianNB(): Classifying using Gaussian naïve bayes method.

6. Code

```
import numpy as np
import pandas as pd
from sklearn import tree
from sklearn.naive_bayes import GaussianNB
from sklearn.neighbors import KNeighborsClassifier
from sklearn.cluster import KMeans
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
df = pd.read_csv("diabetes.csv")
x=df.iloc[:, :-1].values
y=df.iloc[:, 8].values


clf = tree.DecisionTreeClassifier(criterion='entropy')
gnb = GaussianNB()
knn=KNeighborsClassifier(n_neighbors=5)
knn=knn.fit(x,y)
clf = clf.fit(x, y)
```

```
gnb=gnb.fit(x, y)

X=np.array(pd.read_csv('d2.csv'))

kmeans = KMeans(n_clusters=2, random_state=0, init=np.array([[0,0],[189,846]]), n_init=1).fit(X)

from tkinter import *

from PIL import ImageTk, Image



root=Tk()

root.title("Diabetes Predictor")

root.geometry("600x450")


def acc():

    X_train,X_test,y_train,y_test=train_test_split(x,y,test_size=0.25)

    clf1 = tree.DecisionTreeClassifier(criterion='entropy').fit(X_train,y_train)

    gnb1 = GaussianNB().fit(X_train,y_train)

    knn1=KNeighborsClassifier(n_neighbors=5).fit(X_train,y_train)

    y1pred=clf1.predict(X_test)

    y2pred=gnb1.predict(X_test)

    y3pred=knn1.predict(X_test)


    e1=accuracy_score(y_test,y1pred)

    e2=accuracy_score(y_test,y2pred)

    e3=accuracy_score(y_test,y3pred)


    a1.configure(state='normal')

    a1.delete(0, END)

    s1="Decision Tree Accuracy: "+str(round(e1*100,2))+"%"

    a1.insert(END,s1)

    a1.configure(state='disabled')
```

```
    a2.configure(state='normal')
    a2.delete(0, END)
    s2="Naive Bayes Accuracy: "+str(round(e2*100,2))+"%"
    a2.insert(END,s2)
    a2.configure(state='disabled')

    a3.configure(state='normal')
    a3.delete(0, END)
    s3="KNN Accuracy: "+str(round(e3*100,2))+"%"
    a3.insert(END,s3)
    a3.configure(state='disabled')




def action():
    preg=int(ip2.get())
    glu=int(ip3.get())
    blood=int(ip4.get())
    skin=int(ip5.get())
    insulin=int(ip6.get())
    bmi=float(ip7.get())
    pdf=float(ip8.get())
    age=int(ip9.get())

    ans1=clf.predict([[preg , glu, blood, skin, insulin, bmi, pdf, age]])
    ans2=gnb.predict([[preg , glu, blood, skin, insulin, bmi, pdf, age]])
    ans3=knn.predict([[preg , glu, blood, skin, insulin, bmi, pdf, age]])

    pos=0
```

```
    neg=0
    if(ans1==[[0]]):
        neg=neg+1
    else:
        pos=pos+1
    if(ans2==[[0]]):
        neg=neg+1
    else:
        pos=pos+1
    if(ans3==[[0]]):
        neg=neg+1
    else:
        pos=pos+1


    if(neg>pos):
        resultbox.configure(state='normal')
        resultbox.delete(0, END)
        resultbox.insert(END,"Diabetes Negative. You don't have diabetes")
        resultbox.configure(state='disabled')
    else:
        imp=kmeans.predict([[glu,insulin]])
        if(imp==[[0]]):
            resultbox.configure(state='normal')
            resultbox.delete(0, END)
            resultbox.insert(END,"Diabetes positive. Medical Assistance not required at moment.")
            resultbox.configure(state='disabled')
        else:
            resultbox.configure(state='normal')
            resultbox.delete(0, END)
```

```
        resultbox.insert(END,"Diabetes positive. Medical Assistance required as soon as possible.")
        resultbox.configure(state='disabled')



C = Canvas(root, bg="blue", height=250, width=300)
filename = PhotoImage(file = "bg1.png")
background_label = Label(root, image=filename)
background_label.place(x=0, y=0, relwidth=1, relheight=1)

label1=Label(root,text="  Diabetes Predictor", bg='#e3f4fc',font=("Helvetica", 36))
label1.grid(row=1,columnspan=2)

label2=Label(root,text="Enter the no. of pregnancies: ", bg='#e3f4fc')
label2.grid(row=3,column=0,sticky=E,padx=20,pady=5)

label3=Label(root,text="Enter the glucose level: ", bg='#e3f4fc')
label3.grid(row=4,column=0,sticky=E,padx=20,pady=5)

label4=Label(root,text="Enter the blood pressure: ", bg='#e3f4fc')
label4.grid(row=5,column=0,sticky=E,padx=20,pady=5)

label5=Label(root,text="Enter the skin thickness in mm: ", bg='#e3f4fc')
label5.grid(row=6,column=0,sticky=E,padx=20,pady=5)

label6=Label(root,text="Enter the insulin level: ", bg='#e3f4fc')
label6.grid(row=7,column=0,sticky=E,padx=20,pady=5)

label7=Label(root,text="Enter the BMI(Body Mass Index): ", bg='#e3f4fc')
label7.grid(row=8,column=0,sticky=E,padx=20,pady=5)
```

```
label8=Label(root,text="Enter the diabetes pedigree function: ", bg='#e3f4fc')
label8.grid(row=9,column=0,sticky=E,padx=20,pady=5)

label9=Label(root,text="Enter your age: ", bg='#e3f4fc')
label9.grid(row=10,column=0,sticky=E,padx=20,pady=5)

ip2=Entry(root,width=20)
ip2.grid(row=3,column=1,sticky=W)

ip3=Entry(root,width=20)
ip3.grid(row=4,column=1,sticky=W)

ip4=Entry(root,width=20)
ip4.grid(row=5,column=1,sticky=W)

ip5=Entry(root,width=20)
ip5.grid(row=6,column=1,sticky=W)

ip6=Entry(root,width=20)
ip6.grid(row=7,column=1,sticky=W)

ip7=Entry(root,width=20)
ip7.grid(row=8,column=1,sticky=W)

ip8=Entry(root,width=20)
ip8.grid(row=9,column=1,sticky=W)

ip9=Entry(root,width=20)
ip9.grid(row=10,column=1,sticky=W)
```

```
resultbox=Entry(root,width=60)
resultbox.grid(row=12,columnspan=2,sticky=E,padx=97,pady=7)
resultbox.configure(state='disabled')

button1=Button(root, text="Predict",command=action,width=10)
button1.grid(row=11,column=0,sticky=E, padx=100,pady=5)

button2=Button(root, text="Test Model",command=acc,width=10)
button2.grid(row=11,column=1,sticky=E,padx=100,pady=5)

a1=Entry(root,width=60)
a1.grid(row=15,columnspan=2,sticky=E,padx=97)
a1.configure(state='disabled')

a2=Entry(root,width=60)
a2.grid(row=16,columnspan=2,sticky=E,padx=97)
a2.configure(state='disabled')

a3=Entry(root,width=60)
a3.grid(row=17,columnspan=2,sticky=E,padx=97)
a3.configure(state='disabled')
C.grid()
root.mainloop()
```

7. Screenshots

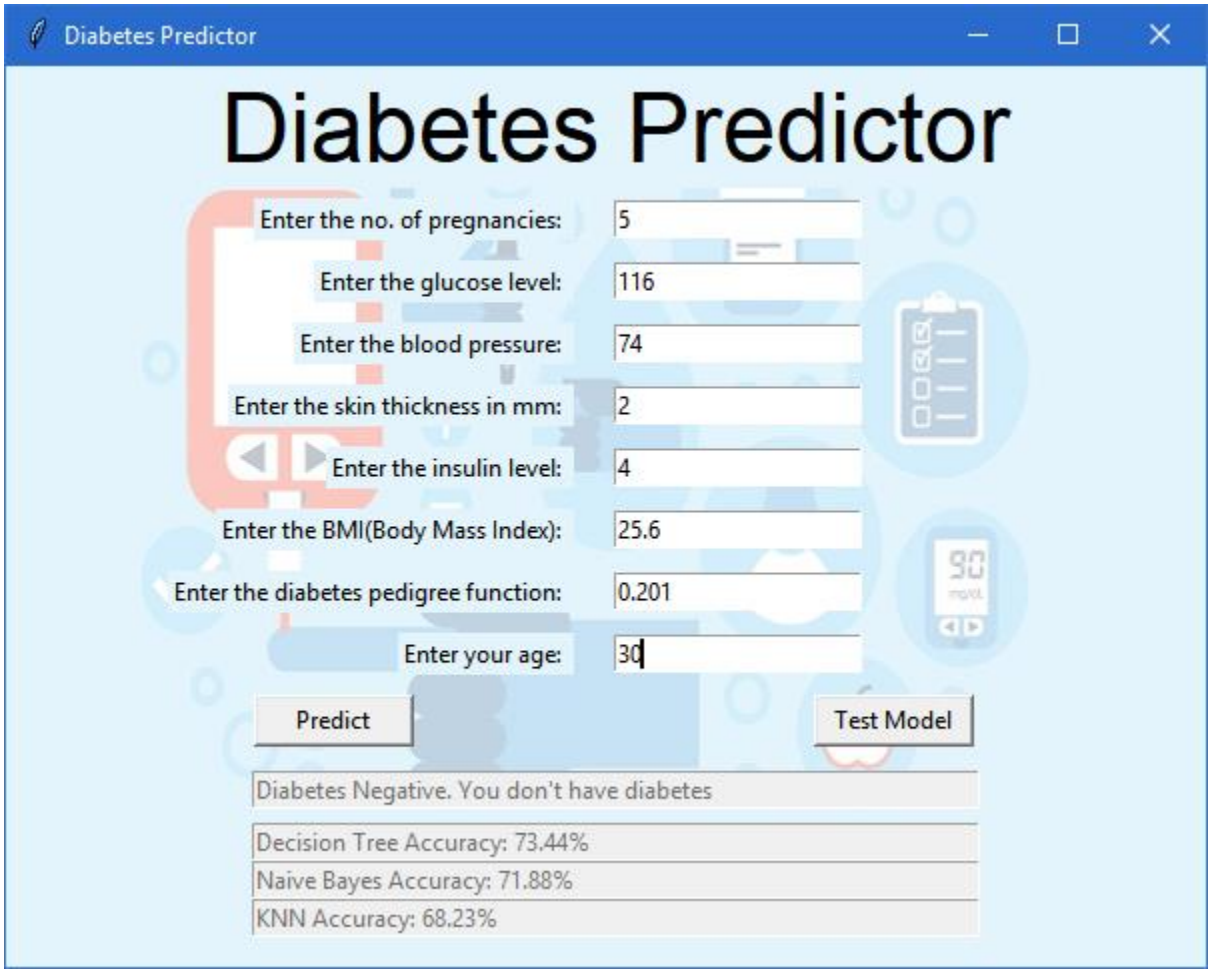


Figure 1: Diabetes Negative Case

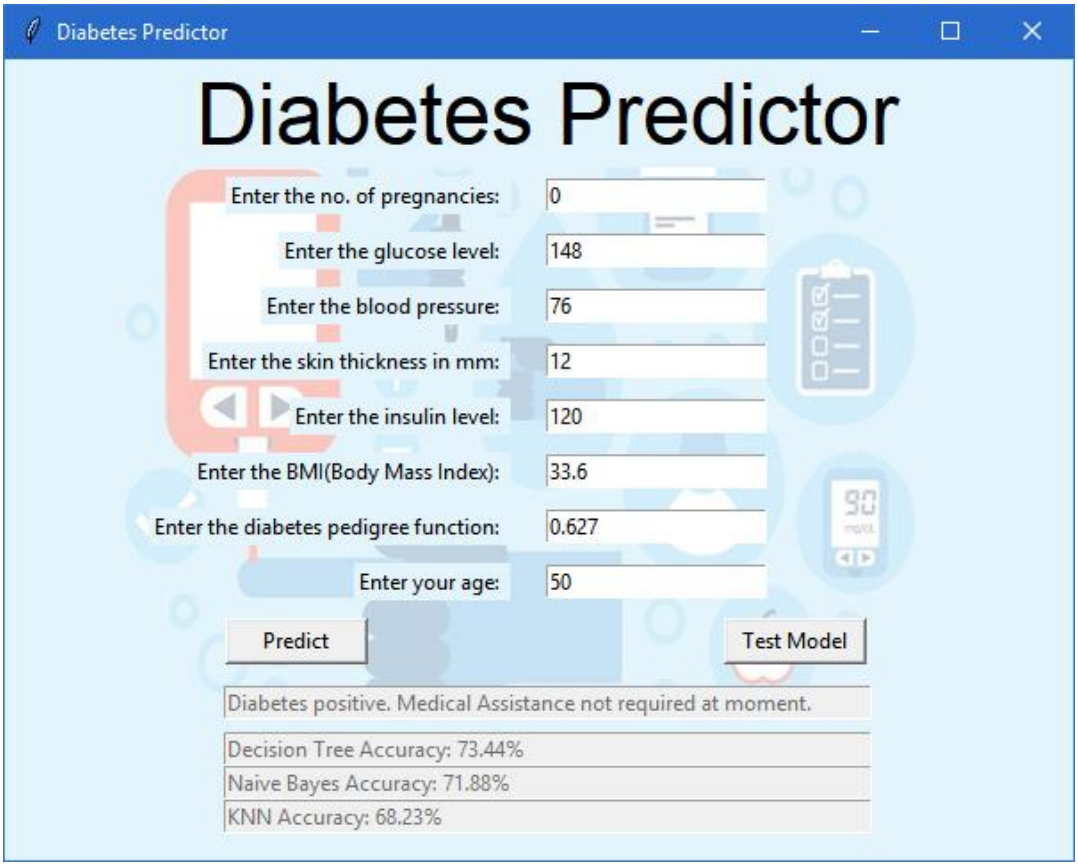


Figure 2: Diabetes Positive Case, but the disease is not severe

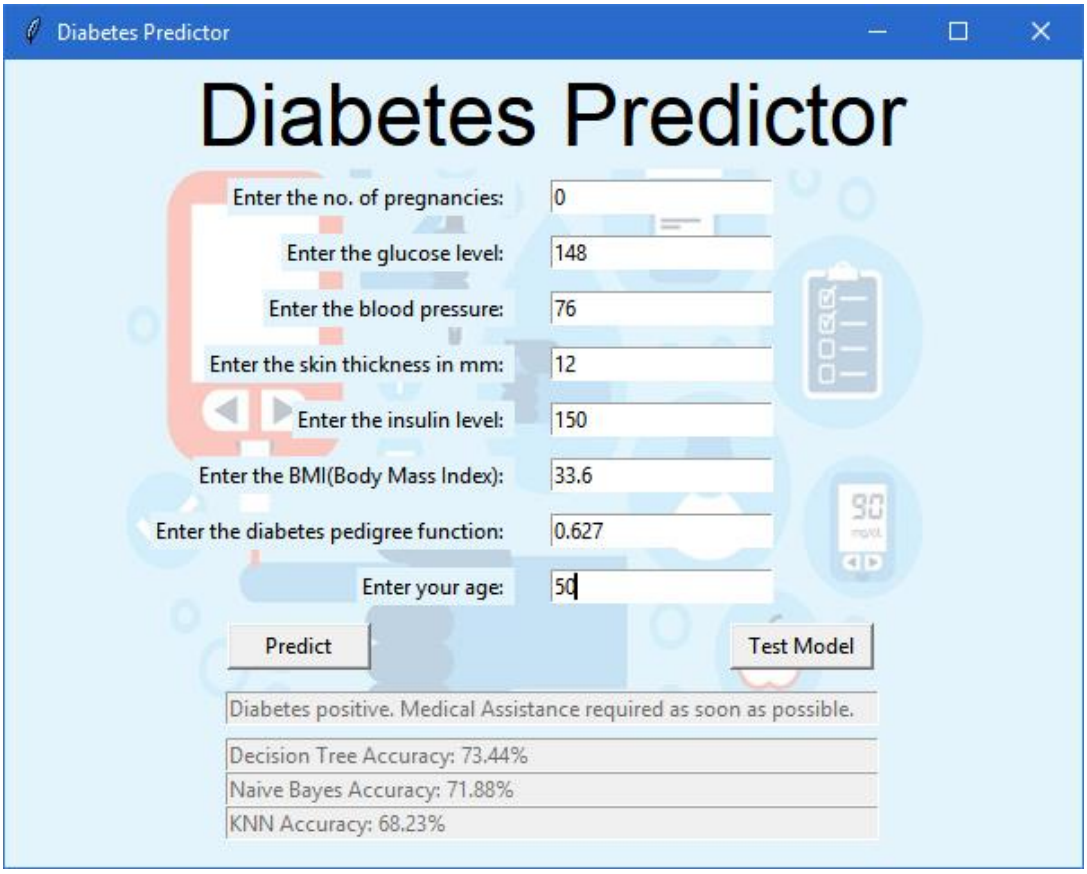
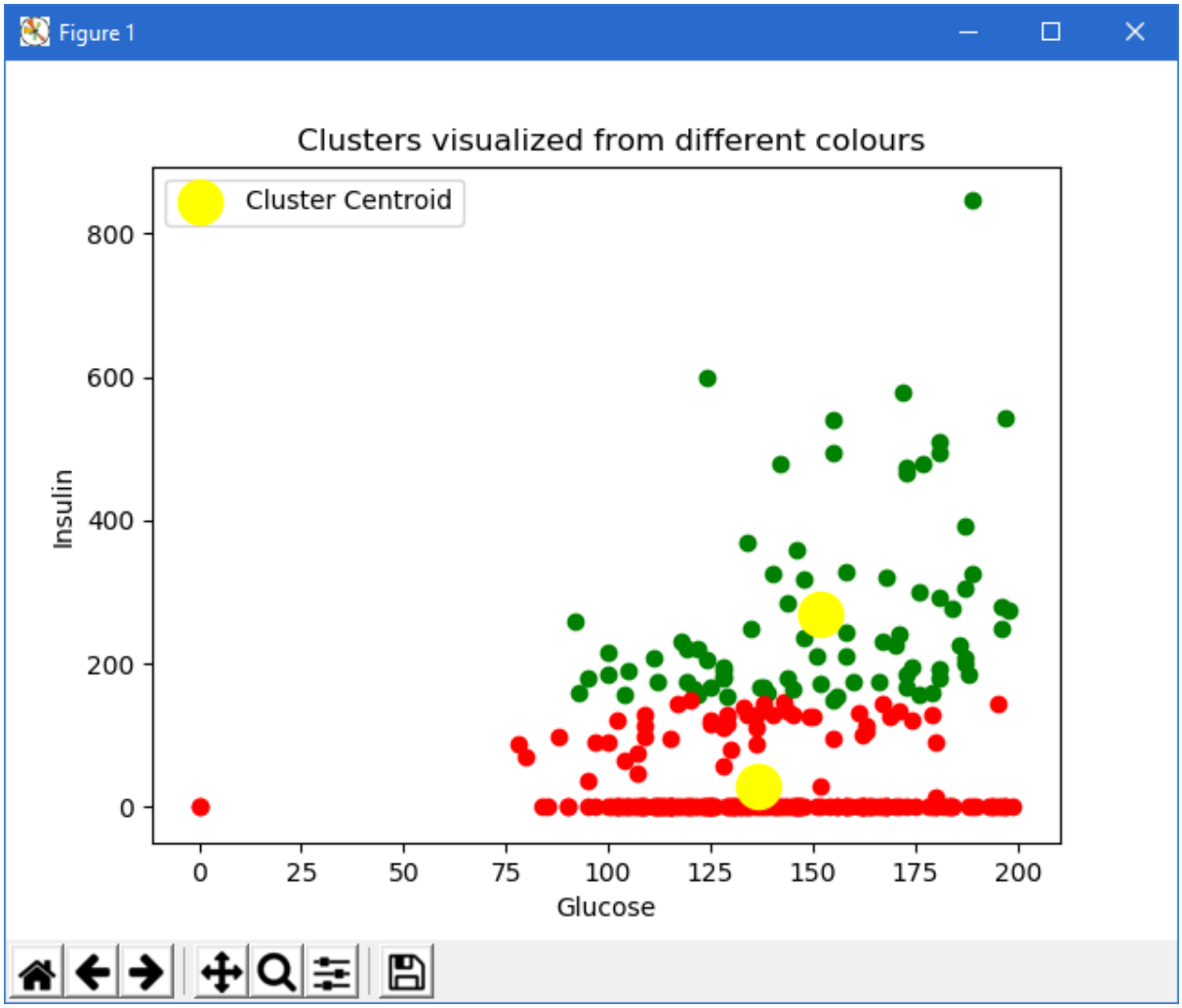


Figure 3: Diabetes Positive Case but the disease is severe

8. Clustering Results



The figure given above shows the clustering of all the diabetic patients in the dataset into severe or not severe according to the attribute Glucose and Insulin level. The red cluster indicates the not severe cluster and the green cluster indicates the severe cluster.

9. Conclusion

This project will help in dealing with the diabetes problem and further predicting the disease early leads to treating the patient before it becomes critical.

10. References

[1] https://en.wikipedia.org/wiki/Diabetes_mellitus
[2] https://en.wikipedia.org/wiki/ID3_algorithm
[3] https://en.wikipedia.org/wiki/Naive_Bayes_classifier
[4] https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm
[5] https://en.wikipedia.org/wiki/K-means_clustering
[6] https://en.wikipedia.org/wiki/Tkinter