A PROJECT REPORT

on

"DIABETES PREDICTION SYSYTEM"

Submitted to

KIIT Deemed to be University

In Partial Fulfillment of the Requirement for the Award of

BACHELOR'S DEGREE IN
INFORMATION TECHNOLOGY

BY

GEETANSH VERMA	1729201
SAURABH KUMAR	1729218
SINDHUSUTA MOHANTY	1729222
ANSUMAN SAHOO	1729242

UNDER THE GUIDANCE OF
PROF. Dr. TANMOY MAITRA

SCHOOL OF COMPUTER ENGINEERING

KALINGA INSTITUTE OF INDUSTRIAL TECHNOLOGY

BHUBANESWAR, ODISHA - 751024

May 2020

A PROJECT REPORT

on

"DIABETES PREDICTION SYSYTEM"

Submitted to

KIIT Deemed to be University

In Partial Fulfillment of the Requirement for the Award of

BACHELOR'S DEGREE IN
INFORMATION TECHNOLOGY

BY

GEETANSH VERMA	**1729201**
SAURABH KUMAR	**1729218**
SINDHUSUTA MOHANTY	**1729222**
ANSUMAN SAHOO	**1729242**

UNDER THE GUIDANCE OF
PROF. Dr. TANMOY MAITRA

SCHOOL OF COMPUTER ENGINEERING

KALINGA INSTITUTE OF INDUSTRIAL TECHNOLOGY

BHUBANESWAE, ODISHA -751024

May 2020

KIIT Deemed to be University

School of Computer Engineering

Bhubaneswar, ODISHA 751024

CERTIFICATE

This is certify that the project entitled

"DIABETES PREDICTION SYSYTEM"

submitted by

GEETANSH VERMA	**1729201**
SAURABH KUMAR	**1729218**
SINDHUSUTA MOHANTY	**1729222**
ANSUMAN SAHOO	**1729242**

is a record of bonafide work carried out by them, in the partial fulfilment of the requirement for the award of Degree of Bachelor of Engineering (Computer Science & Engineering OR Information Technology) at KIIT Deemed to be university, Bhubaneswar. This work is done during year 2019-2020, under our guidance.

Date: 28/11/2020

(**Prof. TANMOY MAITRA**)
Project Guide

Acknowledgements

We are profoundly grateful to Prof. Tanmoy Maitra for his expert guidance and continuous encouragement throughout to see that this project rights its target since its commencement to its completion. His constant guidance has helped us complete the project within the given time. Last but not the least, we would like to thank our fellow project mates for helping us out with all the problem that we faced and making our experience enjoyable.

GEETANSH VERMA
SAURABH KUMAR
SINDHUSUTA MOHANTY
ANSUMAN SAHOO

ABSTRACT

Diabetes is considered as one of the deadliest and chronic diseases which causes an increase in blood sugar. Many complications occur if diabetes remains untreated and unidentified. The tedious identifying process results in visiting of a patient to a diagnostic center and consulting doctor. But the rise in machine learning approaches solves this critical problem.

The motive of this study is to design a model which can prognosticate the likelihood of diabetes in patients with maximum accuracy. Therefore six machine learning classification algorithms namely Decision Tree, KNN ,Logistic regrassion ,Naive Bayes,Random Forest,SVM are used in this experiment to detect diabetes at an early stage. Experiments are performed on Pima Indians Diabetes Database (PIDD) which is sourced from UCI machine learning repository. [1]

Keywords: Diabetes, Decision Tree, KNN, Logistic regrassion, Naive Bayes

Random forest, SVM

Contents

1.Introduction
2.Literature Survey 2
3.Software Requirements Specification 3
3.1System Requirement…… 3
3.2Technical Requirement…… 3
3.3 Libraries Requirement…….. 4
4.Requirement Analysis 5
4.1 Logical Dataset…… 5
4.2 Parameter…… 5
5.System Design
5.1.MODEL DIAGRAM 6
5 .2System Testing 7
6.Test Cases and Test Results…… 8
6.1Project Planning 9
7. Project Planning Included…… 9
8.Implementation 10
8.1Collect the Dataset……. 10
8.2Load the dataset…… 10
8.3Describe the Dataset…… 10
8.4Analysis of the dataset for cleaning …… 10
8.5Split the dataset for training and testing……. 11
8.6 Train the dataset using algorithms ……..
8.7Compare the accuracy of all algorithms 11-14
8.8Create GUI for the model…….. 14-18
9.Hyper parameter Optimization 19
9.1Hyper parameter Optimization of KNN 20
9.2 Hyper parameter Optimization of Decision tree 20
9.3 Hyper parameter Optimization of Logistic Regression 21
9.4 Hyper parameter Optimization of Naive Bayes 22
9.5 Hyper parameter Optimization of Random Forest 22
9.6 Hyper parameter Optimization of SVM 23
10. Screen shots of Project 23
10.1Data Entry page…….. 23
10.2After clicking on KNN button…. 23
10.3 After clicking on Logistic regression button…. 24
10.4After clicking on Naive Bayes button…. 24
10.5After clicking on Decision tree button…. 25
10.6After clicking on compare button…. 25
10.7After entering data and clicking on predict button…. 26
10.8After clicking on Reset button…. 26
11.Conclusion and Future Scope 27

11.1Conclusion 27
11.2 Future Scope 27
11.3 References 28

List of Figures

1.1 MODEL DIAGRAM..... 06

Chapter 1

Introduction

Now days from health care industries large amount of data is generating. This data contains lots of useful information that can be used to take efficient medical decisions. Hence it is necessary to store and process such data to extract knowledge from it and by using that take efficient decisions. In current situations the numbers of people suffering from Diabetes are increasing per day. Diabetic Mellitus (DM) belongs to the family of Non Communicable Diseases (NCD). It is one of the dangerous diseases as it has long term impacts on patient's body. Diabetic Mellitus (DM) is categorized into three types. First is Type 1 DM which is called as Insulin - Dependent Diabetes Mellitus (IDDM). Type 1 DM is caused when patient's body is unable to produce insulin and requires injecting insulin externally to the patient. The second type of DM is Type 2 DM which is called as Non-Insulin-Dependent Diabetes Mellitus (NIDDM). This type of diabetes is caused when patient's body cell are not able to use insulin properly. Third type of DM is gestational diabetes which is caused in pregnant women due to the development of high blood sugar without the knowledge of Pre-diagnosis of Diabetes. This type of DM is very dangerous and can lead to Type 2 DM. As numbers of patients of this serious disease are increasing day by day, the size of diabetic data set is also increasing. With the utilization of machine learning algorithm, it can be possible to analyze the diabetes data and accordingly provide early diagnosis and better treatment. Machine learning is the set of different methods that can be used to find patterns from the dataset and then use those patterns to predict future conditions or to make efficient decisions under some conditions. Machine learning can make decisions independently at its own. Machine learning is categorized into to two types, supervised learning and unsupervised learning.

Chapter 2

Literature Survey

Orabi et al. in designed a system for diabetes prediction, whose main aim is the prediction ofdiabetes a candidateissufferingataparticularage. Theproposedsystemisdesignedbasedon the concept of machine learning, by applying decision tree. Obtained results were satisfactory as the designed system works well in predicting the diabetes incidents at a particular age, with higher accuracy using Decision tree,

Pradhan etal in usedGenetic programming (GP) for the training and testing of the database for prediction of diabetes by employing Diabetes data set which is sourced from UCI repository. Results achieved using Genetic Programming , gives optimal accuracy as compared to other implemented techniques. There can be significant improve in accuracy by taking less time for classifier generation. It proves to be useful for diabetes prediction at low cost.

Rashid et al. in designed a prediction model with two sub-modules to predict diabetes-chronic disease.ANN(ArtificialNeuralNetwork)isusedinthefirstmoduleandFBS(FastingBlood Sugar) is used in the second module. Decision Tree is used to detect the symptoms of diabetes on patients health.

Nongyao et al. in applied an algorithm which classifies the risk of diabetes mellitus. To fulfill the objective au- thor has employed four following renowned machine learning classification methods namely Decision Tree, Artificial Neural Networks, Logistic Regression and Naive Bayes. For improving the robustness of designed model Bagging and Boosting techniques are used. [3]

Chapter 3

Software Requirements Specification

3.1 System Requirement

Operating system:- Linux or Windows 10

RAM requirement:- 8 GB.

ROM:-256 GB

System Type:- 64-bit Operating System

x-64 based processor

Processor:-Intel® Core™ i5-7200 CPU 2.50GHz 2.71GHz

To obtain efficient results, more computational power is favourable .

3.2 Technical Requirements

Python3 : Python is an interpreted high-level programming language for general-purpose programming. Created by GuidovanRossum andfirst released in 1991, It has a design philosophy that emphasizes code readability, notably using significant white space. It provides constructs that enable clear programming on both small and large scales.

PyCharm :- It is developed by Jet brains,which is an Integrated Development Environment used in computer programming and specifically for the Python language.

Jupyter Notebook :-TheJupyter Notebook is an open-source web application that allows us to create and also share document that contains live code, equations, visualizations and narrative text.

Spyder: - Spyder is an open source cross-platform integrated development environment for scientific programming in the Python language. Spyder integrates NumPy, SciPy, Matplotlib and IPython, as well as other open source software.

3.3 Libraries Requirement:

Pandas: - *It is a software library written for the Python programming language fordata manipulation and analysis.It also offers datastructures and operations for manipulating numerical tables and time series.*

Matplotlib: - *It is a Python 2D plotting library which produces publication quality figures in a variety of hard copy formats and interactive environments across platforms.*

NumPy: - It *is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays.*

Scikit-learn *: It is a library in Python that provides many unsupervised and supervised learning algorithms.*

The functionality that scikit-learn provides include:

Regression, including Linear and Logistic Regression
Classification, ***including K-Nearest Neighbors***
Clustering, ***including K-Means and K-Means++***
Model selection
Preprocessing, ***including Min-Max Normalization***

Tkinter:- Tkinter is the standard GUI library for Python. Python when combined with Tkinter provides a fast and easy way to create GUI applications. Tkinter provides a powerful object-oriented interface to the Tk GUI toolkit.

Creating a GUI application using Tkinter is an easy task. All we need to do is perform the following steps .

1. We should Import the *Tkinter* module.
2. Then after we should create the GUI application main window.
3. Then we should add one or more of the above-mentioned widgets to the GUI application.
4. Finally,we sholud Enter the main event loop to take action against each event triggered by the user.

Chapter 4 Requirement Analysis

4.1 Logical Dataset:

- Experiments are performed on *Pima Indians Diabetes Database (PIDD*) which is sourced from UCI machine learning repository.
- *PIDD:*This dataset is originally from the National Institute of Diabetes and Digestive and Kidney Diseases. The objective of the dataset is to diagnostically predict whether or not a patient has diabetes, based on certain diagnostic measurements included in the dataset. Several constraints were placed on the selection of these instances from a larger database. In particular, all patients here are females at least 21 years old of Pima Indian heritage.

The dataset contains 8 attributes and 1 class variable. This data set contains total 768 diabetic and non-diabetic women records whose age is above 21 years. All attributes in data set are numeric. For this work the dataset is considered as complete data set having no missing value in it.

4.2 Parameter :

1. **Pregnancies:** No. of times pregnant
2. **Glucose:** Plasma Glucose Concentration a 2 hour in an oral glucose tolerance test (mg/dl) 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 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)
9. **Outcome:** Class Variable (0 or 1) where '0' denotes patient is not having diabetes and '1' denotes patient having diabetes The dependent variable is whether the patient is having diabetes or not.More than 70% Pima Indian population is suffering from the diabetes.
10. ***4.3 Data Cleaning*** will take place as data has got lot of missing values. Handling missing values can be done either by replacing null values with mode or mean or replacing the null value with a random variable.

Chapter 5 -System Design

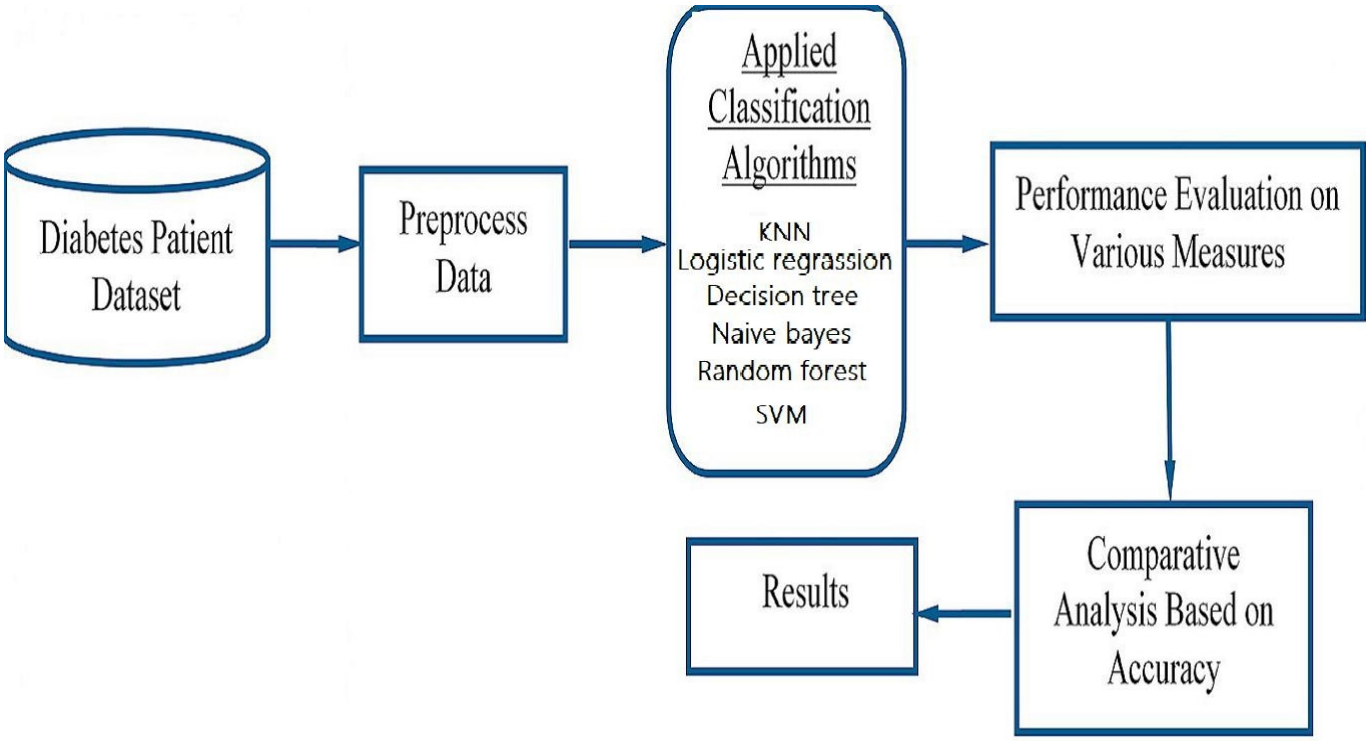


Fig :1.1(model diagram)

To perform the experimentation Pima Indians Diabetes Dataset has been used.The dataset contains 8 attributes and 1 class variable .This data set contains total 768 diabetic and non-diabetic women records whose age is above 21 years.Class variable (0 or 1).0-It indicates False Diabetic Test and1-It indicates True Diabetic Test .First we load the dataset as a CSV file and identify each column then process the dataset by finding a mean value and perform some cleaning operation.Divided the whole data into 8:2 for train and test respectively.

Train and test of dataset done over the all four algorithm i.e DESCISION TREE,KNN,LOGISTIC REGRASSION ,NAIVE BAYES ,Random Forest and Support Vector Machine. From the testing part we analyze the performance of each and every algorithm in percent .Most of the case we got SVM gives us the highest accuracy as compared to other so finally we pick it.

Finally it provides a interface to the user ,where a user has to provide the value of the parameter that she has and our machine will take that value and start the process from the algorithm comparison and use that which has maximum accuracy . Input data will take to test and provides the best possible output.

We are also trying to test some other algorithm in future so that our machine can perform more accurately and achieve the goal of system. [6]

Chapter 6

System Testing

The validity of program was checked by inserting various input parameters and based upon the input the system does the classification into: person will have diabetes or will not have diabetes in future.

6.1 Test Cases and Test Results

Test ID	Test Case Title	Test Condition	System Behavior	Expected Result
T01	Pregnant=2 Glucose=200 Blood pressure=140 Skin=19 Insulin=140 Body mass index=23 Pedigree=1 Age=34	After comparison SVM is found to have highest accuracy. Therefore here SVM has being used	Knn-78.0% Decision tree-71.0% Naive bayes-79.0% Logistic regression-80% Random forest - 79.0% SVM - 82.0%	You have no diabetes (0)
T02	Pregnant=1 Glucose=140 Blood pressure=65 Skin=21 Insulin=110 Body mass index=19 Pedigree=0 Age=29	After comparison SVM is found to have highest accuracy. Therefore here SVM has being used	Knn-78.0% Decision tree-71.0% Naive bayes-79.0% Logistic regression-80% Random forest - 79.0% SVM - 82.0%	You have no diabetes (0)

T03	Pregnant=4 Glucose=187 Blood pressure=110 Skin=17 Insulin=47 Body mass index=18 Pedigree=2 Age=25	After comparison SVM is found to have highest accuracy. Therefore here SVM has being used	Knn-78.0% Decision tree-71.0% Naive bayes-79.0% Logistic regression-80% Random forest - 79.0% SVM - 82.0%	You have Diabetes or may get soon (1)

Chapter 7

Project Planning

7.1 PROJECT PLANNING INCLUDED

-Project purpose

-Business and project goals and objectives

-Scope and expectations

-Assumptions and constraints

-Project management approach

-Ground rules for the project

-Project budget

-Project timeline

-The conceptual design of new technology

Chapter 8

Implementation

DESCRIPTION OF EACH SEGMENT OF CODE IMPLIMENTATION

8.1 Collect the dataset:

At first the dataset was collected from the UCI Repository and was saved as csv (coma separated value) file.

8.2 Load the dataset:

Then the dataset was loaded into the system through the code using read_csv() present in ***pandas*** package.

```
#Load the dataset
import pandas as pd
from tkinter import messagebox as m
col_names=['pregnant','glucose','bp','skin','insulin','bmi','pedigree','age'
    ,'label']
pima=pd.read_csv("D:\pima.csv",names=col_names)
```

to load the dataset

8.3 Describe the dataset:

Then the dataset was described (i.e. the names were given to the columns for easy access) through the code.

```
#describe the dataset
feature=['pregnant','glucose','bp','skin','insulin','bmi','pedigree','age']
```

to describe the dataset

8.4 Analysis of the dataset for cleaning purpose:

Then the analysis of the dataset was done. Some anomalies were found during this process which was mandatory to correct to achieve an accurate result. Thus, the mandatory columns (like blood pressure, skin thickness, insulin) which should not have 0 attribute were cleaned and replaced with the median value of the respective whole column.

```
#dataset cleaning and transformation
#cleaning blood pressure column
median_bp = pima['bp'].median()
pima['bp'] = pima['bp'].replace(to_replace=0, value
    =median_bp)
#cleaning skin thickness column
median_skin = pima['skin'].median()
pima['skin'] = pima['skin'].replace(to_replace=0,
    value=median_skin)
#cleaning insulin column
median_insulin = pima['insulin'].median()
pima['insulin'] = pima['insulin'].replace(to_replace
    =0, value=median_insulin)
```

to clean the dataset

8.5 Split the dataset for training and testing:

Then the dataset was split in ratio 8:2 for training and testing purpose respectively. The dataset was split through the code using train_test_split() which was present in model_selection module of ***sklearn*** package.

```
#splitting the dataset
from sklearn.model_selection import train_test_split
X_train,X_test,Y_train,Y_test=train_test_split(X,Y
    ,test_size=0.1,random_state=0)
```

to split the dataset

8.6 Train the dataset using algorithms

Confusion matrix

To check the adequacy of classifiers performance measures need to be taken into account. True positives (TP) refers to the positive cases that were correctly labeled by the classifier, while true negatives (TN) are the negative cases that were correctly labeled by the classifier. False positives (FP) are the negative cases that were incorrectly labeled, while false negatives (FN) are the positive cases that were incorrectly labeled . Evaluation measures of classifiers such as accuracy was calculated. The accuracy of the classifier on a given test data set is the percentage of test cases that are correctly classified by classifier. [7]

Accuracy=(TN)/(TP+TN+FP+FN)

Using KNN Algorithm

The system was thus trained using KNN Algorithm.

```
#APPLYING KNN ALGORITHM
from sklearn.neighbors import KNeighborsClassifier
knnaccuracy=0#global
KNN=KNeighborsClassifier()#global
#user-defined function for implementation of KNN
    algorithm to the dataset
def knn():
    '''Apply KNN algorithm to data set'''

    global knnaccuracy
    global KNN
    KNN=KNeighborsClassifier(n_neighbors=9)
    KNN.fit(X_train,Y_train)
    Y_pred=KNN.predict(X_test)
    #confusion matrix of KNN algorithm
    from sklearn import metrics
    confusion=metrics.confusion_matrix(Y_test,Y_pred
        )
    print(confusion)
    TP=confusion[1,1]#True Positive
    TN=confusion[0,0]#True Negative
    FP=confusion[0,1]#False Positive
    FN=confusion[1,0]#False Negative
    knnaccuracy=round(((TP+TN)/(TP+TN+FP+FN)),2)*100
    #printing KNN algorithm's accuracy for our model
    m.showinfo(title="Accuracy",message="Accuracy
        is"+str(knnaccuracy)+"%")

```

Training the model using KNN Algorithm

Using Logistic Regression Algorithm

The system was then trained using Logistic Regression Algorithm.

```
#APPLYING LOGISTIC REGRESSION
from sklearn.linear_model import LogisticRegression
LOGREG=LogisticRegression()#global
logaccuracy=0#global
#user-defined function for implementation of
    Logistic Regression to the dataset
def logreg():
    '''Apply Logistic Regression to Data Set'''

    global logaccuracy
    global LOGREG
    LOGREG=LogisticRegression()
    LOGREG.fit(X_train,Y_train)
    Y_pred=LOGREG.predict(X_test)
    #confusion matrix of Logistic Regression
    from sklearn import metrics
    confusion=metrics.confusion_matrix(Y_test,Y_pred
        )
    print(confusion)
    TP=confusion[1,1]#True Positive
    TN=confusion[0,0]#True Negative
    FP=confusion[0,1]#False Positive
    FN=confusion[1,0]#False Negative
    logaccuracy=round(((TP+TN)/(TP+TN+FP+FN)),2)*100
    #printing Logistic Regression's accuracy for our
        model
    m.showinfo(title="Accuracy",message="Accuracy
        is"+str(logaccuracy)+"%")

```

Training the model using Logistic Regression

Using Naive Bayes Algorithm

The system was then trained using Gaussian Naïve Bayes Algorithm.

```
#APPLYING NAIVE BAYES ALGORITHM
from sklearn.naive_bayes import GaussianNB
naiveaccuracy=0
NAIVE=GaussianNB

#user-defined function for implementation of Naive Bayes Classifier
    Algorithm to the dataset
def naive():
    '''Apply Naive Bayes Classifier Algorithm to Data Set'''

    global naiveaccuracy
    global NAIVE
    NAIVE=GaussianNB()
    NAIVE.fit(X_train,Y_train)
    Y_pred=NAIVE.predict(X_test)

    #confusion matrix of Naive Bayes algorithm
    from sklearn import metrics
    confusion=metrics.confusion_matrix(Y_test,Y_pred)
    TP=confusion[1,1]#True Positive
    TN=confusion[0,0]#True Negative
    FP=confusion[0,1]#False Positive
    FN=confusion[1,0]#False Negative
    naiveaccuracy=round(((TP+TN)/(TP+TN+FP+FN)),2)*100
    #printing Naive Bayes algorithm's accuracy for our model
    m.showinfo(title="Accuracy",message="Accuracy is"+str(naiveaccuracy)+"%"
        )
```

Training the model using Gaussian Naïve Bayes

Using Decision Tree Algorithm

The system was then trained using Decision Tree Algorithm.

```
#APPLYING DECISION TREE ALGORITHM
from sklearn.tree import DecisionTreeClassifier
dtreeaccuracy=0
DTREE=DecisionTreeClassifier()

#user-defined function for implementation of Decision tree to the dataset
def dtree():
    '''Apply Decision Tree Algorithm to Data Set'''

    global dtaccuracy
    global DTREE
    DTREE=DecisionTreeClassifier(random_state=0)
    DTREE.fit(X_train,Y_train)
    Y_pred=DTREE.predict(X_test)
    from sklearn import metrics
    confusion=metrics.confusion_matrix(Y_test,Y_pred)

    #confusion matrix of Decision Tree algorithm
    TP=confusion[1,1]#True Positive
    TN=confusion[0,0]#True Negative
    FP=confusion[0,1]#False Positive
    FN=confusion[1,0]#False Negative
    dtaccuracy=round(((TP+TN)/(TP+TN+FP+FN)),2)*100
    #printing Decision Tree algorithm's accuracy for our model
    m.showinfo(title="Accuracy",message="Accuracy is"+str(dtaccuracy)+"%")
```

Training the model using Decision Tree Algorithm

Using Random forest algorithm

```
def random():
    '''Apply Random Forest Classifier to dataset'''

    global rfaccuracy
    global random
    random=RandomForestClassifier(random_state=10)
    random.fit(X_train,Y_train)
    Y_pred=random.predict(X_test)
    from sklearn import metrics
    confusion=metrics.confusion_matrix(Y_test,Y_pred)
    TP=confusion[1,1]
    TN=confusion[0,0]
    FP=confusion[0,1]
    FN=confusion[1,0]
    sensitivity=TP/(TP+FN)
    specificity=TN/(TN+FP)
    rfaccuracy=round(((TP+TN)/(TP+TN+FP+FN)),2)*100
    m.showinfo(title="Accuracy",message="Accuracy is"+str(rfaccuracy)+"%")
```

Using Support vector machine algorithm

```
from sklearn.svm import SVC
svmaccuracy=0
def SVM():
    '''Apply SVM Classifier to dataset'''

    global svmaccuracy
    global SVM
    SVM=SVC(kernel='linear', C=0.01)
    SVM.fit(X_train,Y_train)
    Y_pred=SVM.predict(X_test)
    try:
        l=SVM.predict([[float(vpreg.get()),float(vglucose.get()),float(vbp.get()),float(vskin.get()),float
        if l==0:
            m.showinfo(title="Diabetes Prediction",message="You Have No Diabetes")
        else:
            m.showinfo(title="Diabetes Prediction",message="You have Diabetes or may get soon")
    except:
        reset()
        btnpredict.configure(command=predicts)
        x=0
        vpreg.set()




    from sklearn import metrics
    confusion=metrics.confusion_matrix(Y_test,Y_pred)
    TP=confusion[1,1]
    TN=confusion[0,0]
    FP=confusion[0,1]
    FN=confusion[1,0]
    sensitivity=TP/(TP+FN)
    specificity=TN/(TN+FP)
    svmaccuracy=round(((TP+TN)/(TP+TN+FP+FN)),2)*100
    m.showinfo(title="Accuracy",message="Accuracy is"+str(svmaccuracy)+"%")
```

8.7 Compare the accuracy of all the Algorithm

After finding out the accuracy of both the algorithms, they were compared so as to find the optimized algorithm. And accuracy of both the algorithms were displayed in graphical representation.

```
#user-defined function to compare accuracy of all the algorithms and display
    in form of graph
def compare():
    '''compare the 4 algorithm by using bar'''
    result=[knnaccuracy,logaccuracy,naiveaccuracy,dtaccuracy]
    clf=['KNN','LOG REG','NAIVEBAYES','DTREE']
    import matplotlib.pyplot as plt
    plt.bar(clf,result)
    plt.ylabel("accuracy")
    plt.xlabel("Model")
    plt.title("Model acuracy comparasion")
    plt.show()
```

To compare all the algorithm and display it as a graph

8.8 Create GUI for the model

Then the Graphical User Interfaces for the whole model were model were created.

```
#GUI
from tkinter import *
w=Tk()
w.geometry("1200x800")
w.title("Diabetes Prediction System")
w.resizable(1,0)
vpreg=StringVar()
vglucose=StringVar()
vbp=StringVar()
vskin=StringVar()
vinsulin=StringVar()
vbmi=StringVar()
vpedegree=StringVar()
vage=StringVar()
```

```
#user-defined function to be called when predict button is clicked
def predict():
    ''' It will predict the status of a new patient'''

    global KNN
    global LOGREG
    global NAIVE
    global DTREE
    global knnaccuracy
    global logaccuracy
    global naiveaccuracy
    global dtaccuracy

#Handling errors
    try:
        a=float(vpreg.get())
        b=float(vglucose.get())
        c=float(vbp.get())
        d=float(vskin.get())
        e=float(vinsulin.get())
        f=float(vbmi.get())
        g=float(vpedegree.get())
        h=float(vage.get())
        #x=1
        #validate()

    except:
        reset()
```

To create space for the GUI

Function of Predict button

```
#Logic of predict function
    if knnaccuracy>=logaccuracy and knnaccuracy>=naiveaccuracy and knnaccuracy>=dtaccuracy:
        k=KNN.predict([[a,b,c,d,e,f,g,h]])
        if k==0:
            m.showinfo(title="Diabetes Prediction",message="You Have No Diabetes")
        else:
            m.showinfo(title="Diabetes Prediction",message="You have Diabetes or may get soon")
    elif logaccuracy>=knnaccuracy and logaccuracy>=naiveaccuracy and logaccuracy>=dtaccuracy:
        l=LOGREG.predict([[a,b,c,d,e,f,g,h]])
        if l==0:
            m.showinfo(title="Diabetes Prediction",message="You Have No Diabetes")
        else:
            m.showinfo(title="Diabetes Prediction",message="You have Diabetes or may get soon")

    elif naiveaccuracy>=knnaccuracy and naiveaccuracy>=logaccuracy and naiveaccuracy>=dtaccuracy:
        n=NAIVE.predict([[a,b,c,d,e,f,g,h]])
        if n==0:
            m.showinfo(title="Diabetes Prediction",message="You Have No Diabetes")
        else:
            m.showinfo(title="Diabetes Prediction",message="You have Diabetes or may get soon")

    else:
        d=DTREE.predict([[a,b,c,d,e,f,g,h]])
        if d==0:
            m.showinfo(title="Diabetes Prediction",message="You Have No Diabetes")
        else:
            m.showinfo(title="Diabetes Prediction",message="You have Diabetes or may get soon")
```

Logic of Predict button

```
151 #user-defined function to reset all the input values
        to NULL
152 #to be called when reset button is clicked
153 def reset():
154     vpreg.set("")
155     vglucose.set("")
156     vbp.set("")
157     vskin.set("")
158     vinsulin.set("")
159     vbmi.set("")
160     vpedegree.set("")
161     vage.set("")
```

Function of Reset button

```
#Designing
#load the image
img=PhotoImage(file="D:/diabetesimage.png")
#Label Image
lblimage=Label(w,image=img)
lblimage.grid(row=1,column=1,rowspan=9)
#Label title
labeltitle=Label(w,text="Enter Your Details!!!!",fg
    ='red',font=('arial',20,'bold'))
labeltitle.grid(row=1,column=2,columnspan=2)
#Label pregnant
labelpreg=Label(w,text="Pregnant",font=('arial',20
    ,'bold'))
labelpreg.grid(row=2,column=2)
entrypreg=Entry(w,font=('arial',20,'bold'
    ),textvariable=vpreg)
entrypreg.grid(row=2,column=3)
#Label glucose
labelglucose=Label(w,text="Glucose",font=('arial',20
    ,'bold'))
labelglucose.grid(row=3,column=2)
entryglucose=Entry(w,font=('arial',20,'bold'
    ),textvariable=vglucose)
entryglucose.grid(row=3,column=3)
#Label bp
labelbp=Label(w,text="Blood Pressure",font=('arial'
    ,20,'bold'))
labelbp.grid(row=4,column=2)
entrybp=Entry(w,font=('arial',20,'bold'
    ),textvariable=vbp)
entrybp.grid(row=4,column=3)
```

Designing the page

```
#Label skin thickness
labelskin=Label(w,text="Skin",font=('arial',20
    ,'bold'))
labelskin.grid(row=5,column=2)
entryskin=Entry(w,font=('arial',20,'bold'
    ),textvariable=vskin)
entryskin.grid(row=5,column=3)
#Label insulin
labelinsulin=Label(w,text="Insulin",font=('arial',20
    ,'bold'))
labelinsulin.grid(row=6,column=2)
entryinsulin=Entry(w,font=('arial',20,'bold'
    ),textvariable=vinsulin)
entryinsulin.grid(row=6,column=3)
#Label bmi
labelbmi=Label(w,text="Body Mass Index",font
    =('arial',20,'bold'))
labelbmi.grid(row=7,column=2)
entrybmi=Entry(w,font=('arial',20,'bold'
    ),textvariable=vbmi)
entrybmi.grid(row=7,column=3)
```

Designing the page cont..

```
#Label pedigree
labelpedigree=Label(w,text="Pedegree",font=('arial'
    ,20,'bold'))
labelpedigree.grid(row=8,column=2)
entrypedigree=Entry(w,font=('arial',20,'bold'
    ),textvariable=vpedegree)
entrypedigree.grid(row=8,column=3)
#Label age
labelage=Label(w,text="Age",font=('arial',20,'bold'
    ))
labelage.grid(row=9,column=2)
entryage=Entry(w,font=('arial',20,'bold'
    ),textvariable=vage)
entryage.grid(row=9,column=3)

#Button Predict
btnpredict=Button(w,text="Predict",bg='yellow',width
    =10,relief='groove',\
                  font=('arial',20,'bold'),fg
                      ='green',command=predict)
btnpredict.grid(row=10,column=2)
#Button Reset
btnreset=Button(w,text="Reset",bg='yellow',width=10
    ,relief='groove',\
                font=('arial',20,'bold'),fg='green'
                    ,command=reset)
btnreset.grid(row=10,column=3)
```

```
#Button KNN
btnknn=Button(w,text="  KNN  ",bg='cyan',font=('arial',20,'bold'),command
    =knn)
btnknn.grid(row=10,column=1)

#Button Logistic Regression
btnlogreg=Button(w,text="Logistic Regression",bg='cyan',font=('arial',20
    ,'bold'),command=logreg)
btnlogreg.grid(row=11,column=1)

#Button Naive Bayes
btndt=Button(w,text="Naive Bayes",bg='cyan',font=('arial',20,'bold'),commanc
    =naive)
btndt.grid(row=12,column=1)

#Button Decision Tree
btndt=Button(w,text="Decision Tree",bg='cyan',font=('arial',20,'bold'
    ),command=dtree)
btndt.grid(row=13,column=1)

#Button Compare
btncompare=Button(w,text="Compare",bg='cyan',border=10,font=('arial',20
    ,'bold'),\
                  command=compare)
btncompare.grid(row=14,column=1)

#infinite loop-mainloop
w.mainloop()
```

Designing the page cont..

Chapter 9:
Hyperparameter optimization of algorithms:

1. Hyper parameter optimization of KNN:

```
def knn():
    '''Apply KNN algorithm to data set'''

    global knnaccuracy
    global KNN
    KNN=KNeighborsClassifier(n_jobs=-1)
    params = {'n_neighbors':[5,6,7,8,9,10],
          'leaf_size':[1,2,3,5],
          'weights':['uniform', 'distance'],
          'algorithm':['auto', 'ball_tree','kd_tree','brute'],
          'n_jobs':[-1]}
    model1 = GridSearchCV(KNN, param_grid=params, n_jobs=1)
    model1.fit(X_train,Y_train)
    print("Best Hyper Parameters:\n",model1.best_params_)
    print("best accuracy of knn is",model1.best_score_)
    prediction=model1.predict(X_test)

##############################################################################
                            #confusion matrix
#it is used to check the performance of the algorithm
#find sensitivity,Specificity and Accuracy
##############################################################################
    from sklearn import metrics
    confusion=metrics.confusion_matrix(Y_test,prediction)
    print(confusion)
    TP=confusion[1,1]#True Positive
    TN=confusion[0,0]#True Negative
    FP=confusion[0,1]#False Positive
    FN=confusion[1,0]#False Negative
    sensitivity=TP/(TP+FN)#When actual value is positive how many prediction True
    print("sensitivity is",sensitivity)
    specificity=TN/(TN+FP)#When actual value is negative how many prediction True
    print("specificity is",specificity)
    knnaccuracy=round(((TP+TN)/(TP+TN+FP+FN)),2)*100
    m.showinfo(title="Accuracy",message="Accuracy is"+str(knnaccuracy)+"%")
    print(metrics.accuracy_score(prediction,Y_test))
```

2. Hyper parameter optimization of Decision tree:

```
dtreeaccuracy=0
def dtree():
    from sklearn.tree import DecisionTreeClassifier
    global dtaccuracy
    global DTREE
    DTREE=DecisionTreeClassifier(random_state=1234)
    params = {'max_features': ['auto', 'sqrt', 'log2'],
          'min_samples_split': [2,3,4,5,6,7,8,9,10,11,12,13,14,15],
          'min_samples_leaf':[1,2,3,4,5,6,7,8,9,10,11],
          'random_state':[123]}
    model1 = GridSearchCV(DTREE, param_grid=params,n_jobs=-1)
    model1.fit(X_train,Y_train)
    #The best hyper parameters set
    print("Best Hyper Parameters:",model1.best_params_)
    print("best accuracy of decision tree is",model1.best_score_)
    Y_pred=model1.predict(X_test)
    from sklearn import metrics
    confusion=metrics.confusion_matrix(Y_test,Y_pred)
    print(confusion)
    TP=confusion[1,1]#True Positive
    TN=confusion[0,0]#True Negative
    FP=confusion[0,1]#False Positive
    FN=confusion[1,0]#False Negative
    sensitivity=TP/(TP+FN)#When actual value is positive how many prediction True
    print("sensitivity is",sensitivity)
    specificity=TN/(TN+FP)#When actual value is negative how many prediction True
    print("specificity is",specificity)
    dtaccuracy=round(((TP+TN)/(TP+TN+FP+FN)),2)*100
    m.showinfo(title="Accuracy",message="Accuracy is"+str(dtaccuracy)+"%")
    print(metrics.accuracy_score(Y_pred,Y_test))
```

3. Hyper parameter optimization of Logistic Regression:

```
from sklearn.linear_model import LogisticRegression
#LOGREG=LogisticRegression()
logaccuracy=0
def logreg():
    '''Apply Logistic Regression to Data Set'''

    global logaccuracy
    global LOGREG
    LOGREG=LogisticRegression(solver='liblinear',multi_class='auto')
    LRparam_grid = {
    'C': [0.001, 0.01, 0.1, 1, 10, 100, 1000],
    'penalty': ['l2'],
    'max_iter': list(range(100,800,100)),
    'solver': ['newton-cg', 'lbfgs', 'liblinear', 'sag', 'saga']
}
    LR_search = GridSearchCV(LOGREG, param_grid=LRparam_grid, refit = True, verbose = 3, cv=None)
    LR_search.fit(X_train,Y_train)
    print("best hyper parameters are", LR_search.best_params_)
    print("best accuracy of logistic regression is",LR_search.best_score_)
    Y_pred=LR_search.predict(X_test)
    from sklearn import metrics
    confusion=metrics.confusion_matrix(Y_test,Y_pred)
    print(confusion)
    TP=confusion[1,1]#True Positive
    TN=confusion[0,0]#True Negative
    FP=confusion[0,1]#False Positive
    FN=confusion[1,0]#False Negative
    sensitivity=TP/(TP+FN)#When actual value is positive how many prediction True
    print("sensitivity is",sensitivity)
    specificity=TN/(TN+FP)#When actual value is negative how many prediction True
    print("specificity is",specificity)
    logaccuracy=round(((TP+TN)/(TP+TN+FP+FN)),2)*100
    m.showinfo(title="Accuracy",message="Accuracy is"+str(logaccuracy)+"%")

    print(metrics.accuracy_score(Y_test,Y_pred))
```

4. Hyper parameter Optimization of Naive Bayes:

```
naiveaccuracy=0
def naive():
    from sklearn.naive_bayes import BernoulliNB
    global naiveaccuracy
    global NAIVE
    NAIVE=BernoulliNB()
    params = {'alpha': [0.01, 0.1, 0.5, 1.0, 10.0],
         }

    bernoulli_nb_grid = GridSearchCV(NAIVE, param_grid=params, n_jobs=-1, cv=5, verbose=5)
    bernoulli_nb_grid.fit(X_train,Y_train)
    print(bernoulli_nb_grid.best_params_)
    print(" best accuracy of naive bayes is",bernoulli_nb_grid.best_score_)
    Y_pred=bernoulli_nb_grid.predict(X_test)

    from sklearn import metrics
    confusion=metrics.confusion_matrix(Y_test,Y_pred)
    print(confusion)
    TP=confusion[1,1]#True Positive
    TN=confusion[0,0]#True Negative
    FP=confusion[0,1]#False Positive
    FN=confusion[1,0]#False Negative
    sensitivity=TP/(TP+FN)#When actual value is positive how many prediction True
    print("sensitivity is",sensitivity)
    specificity=TN/(TN+FP)#When actual value is negative how many prediction True
    print("specificity is",specificity)

    naiveaccuracy=round(((TP+TN)/(TP+TN+FP+FN)),2)*100
    m.showinfo(title="Accuracy",message="Accuracy is"+str(naiveaccuracy)+"%")
    print(metrics.accuracy_score(Y_test,Y_pred))
```

5. Hyper parameter Optimization of Random Forest:

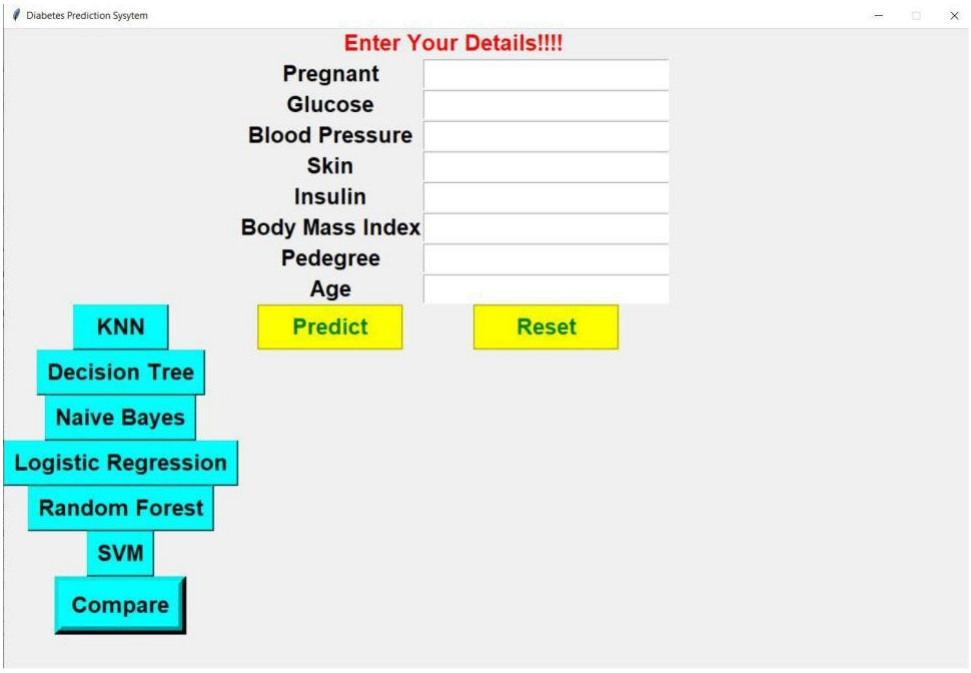
```
from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import RandomForestClassifier
rfaccuracy=0
def random():
#making the instance
    global rfaccuracy
    global random
    model=RandomForestClassifier()
    #hyper parameters set
    params = {'criterion':['gini','entropy'],
              'n_estimators':[10,15,20,25,30],
              'min_samples_leaf':[1,2,3],
              'min_samples_split':[3,4,5,6,7],
              'random_state':[123],
              'n_jobs':[-1]}
    #Making models with hyper parameters sets
    model1 = GridSearchCV(model, param_grid=params, n_jobs=-1)
    #learning
    model1.fit(X_train,Y_train)
    #The best hyper parameters set
    print("Best Hyper Parameters:\n",model1.best_params_)
    #Prediction
    Y_pred=model1.predict(X_test)
    #importing the metrics module
    from sklearn import metrics
    confusion=metrics.confusion_matrix(Y_test,Y_pred)
    TP=confusion[1,1]
    TN=confusion[0,0]
    FP=confusion[0,1]
    FN=confusion[1,0]
    sensitivity=TP/(TP+FN)
    specificity=TN/(TN+FP)
    rfaccuracy=round(((TP+TN)/(TP+TN+FP+FN)),2)*100
    m.showinfo(title="Accuracy",message="Accuracy is"+str(rfaccuracy)+"%")
```

6. Hyper parameter Optimization of SVM:

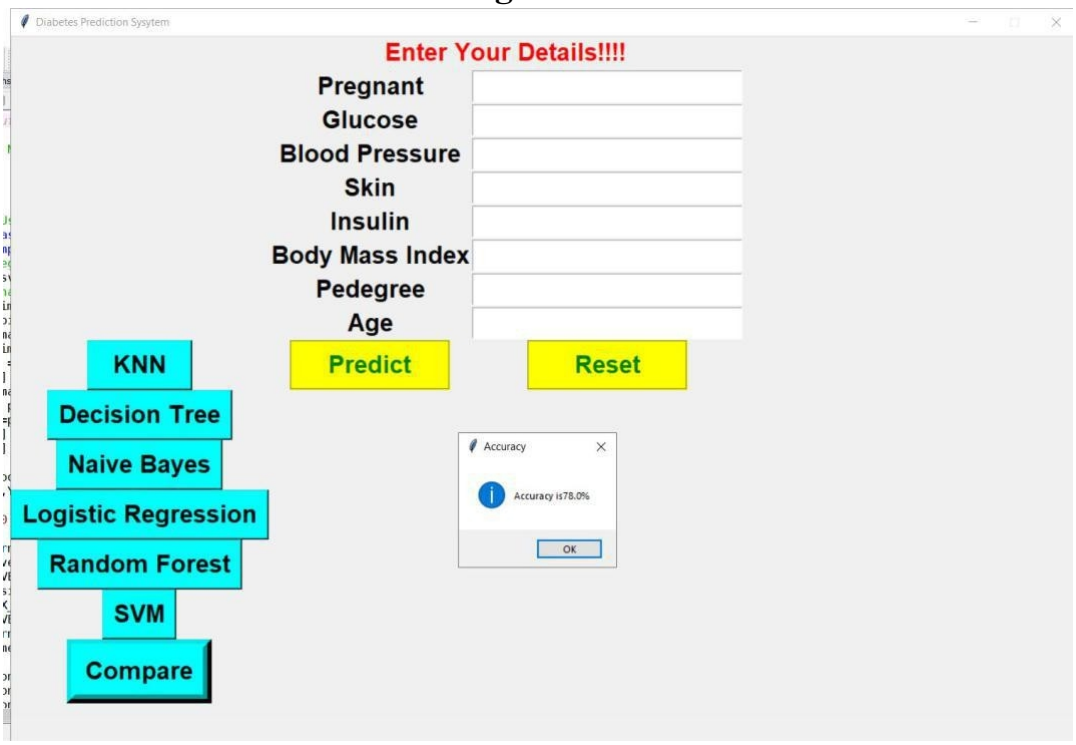
```
from sklearn.svm import SVC
from sklearn import svm
svmaccuracy=0
def SVM():
    '''Apply SVM Classifier to dataset'''

    global svmaccuracy
    global SVM
    SVM=svm.SVC()
    params = {'C': [6,7,8,9,10,11,12],
          'kernel': ['linear','rbf']}
    model1 = GridSearchCV(SVM, param_grid=params, n_jobs=-1)
    model1.fit(X_train,Y_train)
    print("Best Hyper Parameters of SVM are:\n",model1.best_params_)
    print("best accuracy of SVM is",model1.best_score_)
    Y_pred=model1.predict(X_test)
    from sklearn import metrics
    confusion=metrics.confusion_matrix(Y_test,Y_pred)
    TP=confusion[1,1]
    TN=confusion[0,0]
    FP=confusion[0,1]
    FN=confusion[1,0]
    sensitivity=TP/(TP+FN)
    specificity=TN/(TN+FP)
    svmaccuracy=round(((TP+TN)/(TP+TN+FP+FN)),2)*100
    m.showinfo(title="Accuracy",message="Accuracy is"+str(svmaccuracy)+"%")
```

Chapter 9
Screen shots of Project

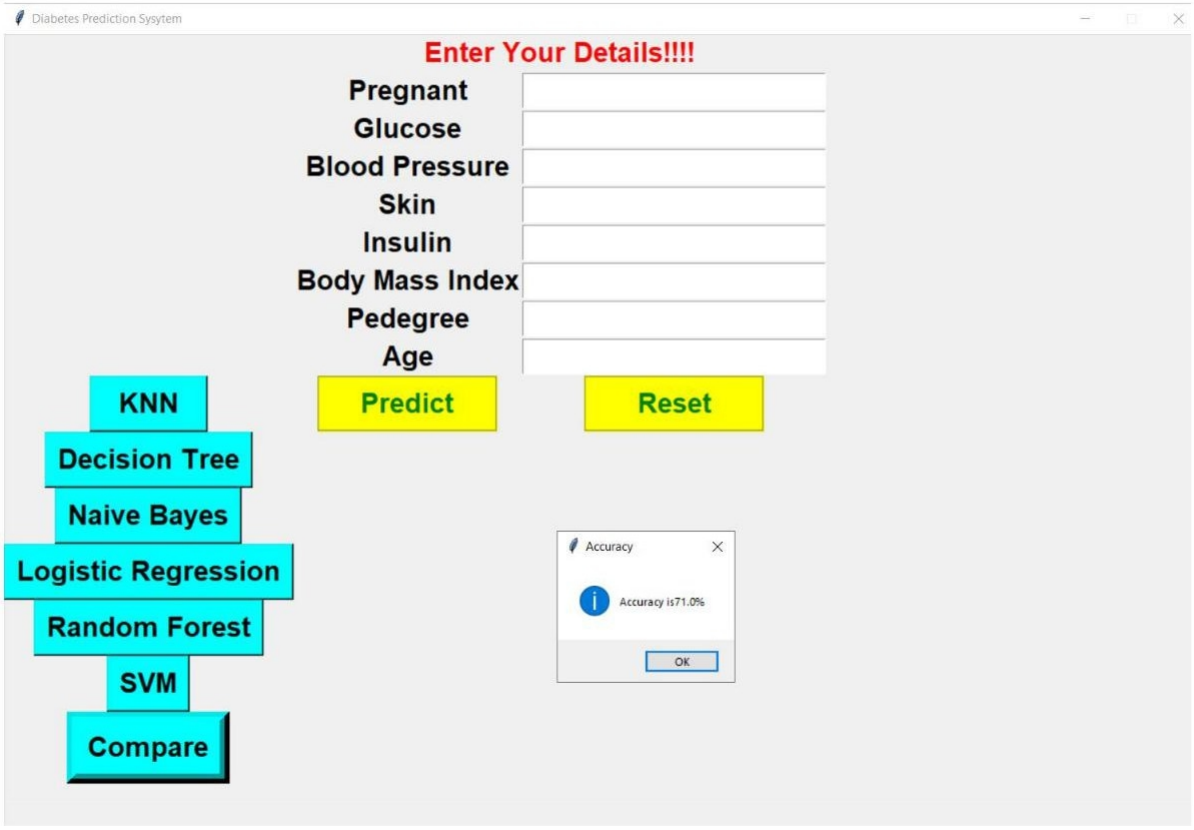
9.1 Data Entry page



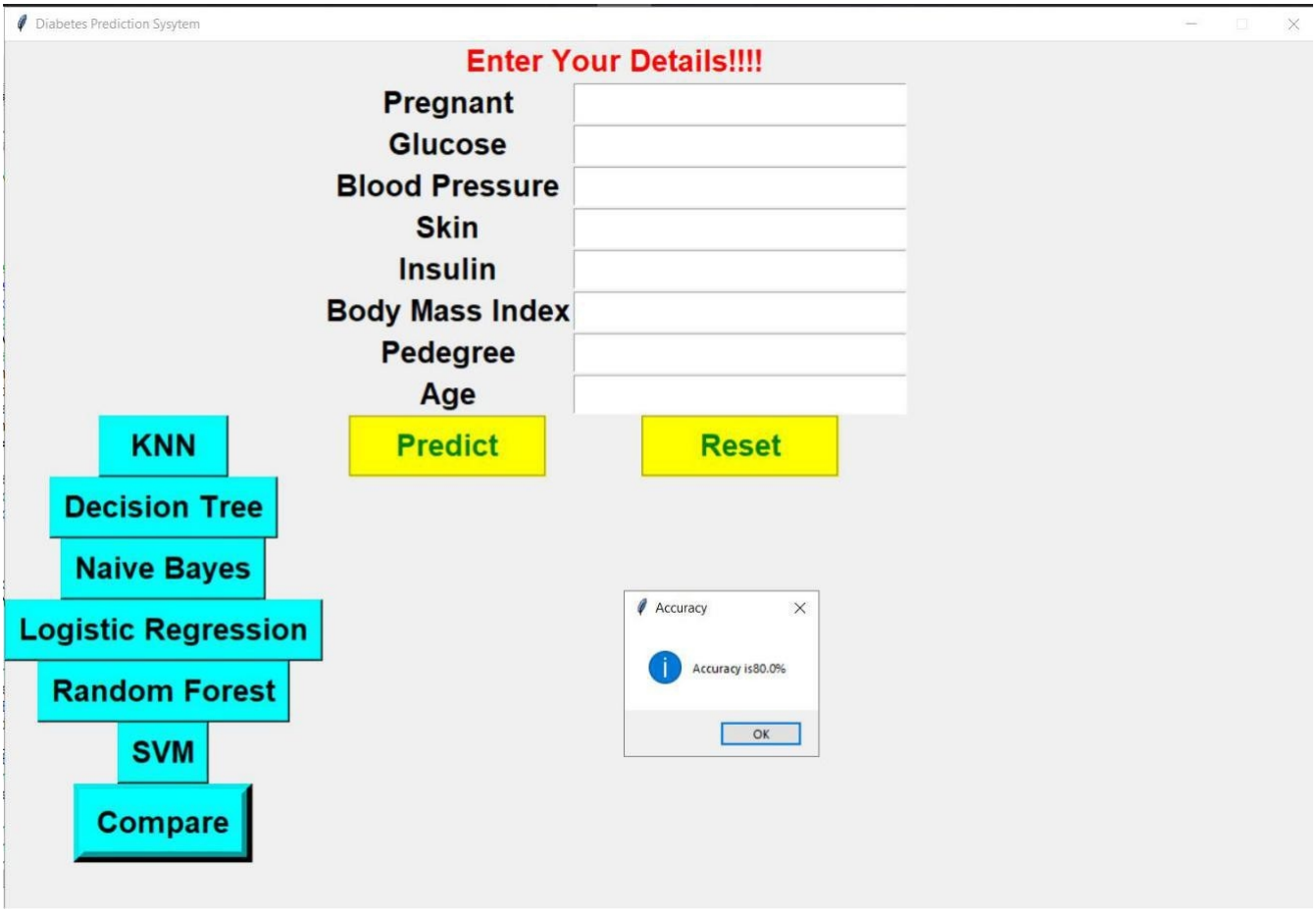
9.2 After clicking on the KNN button



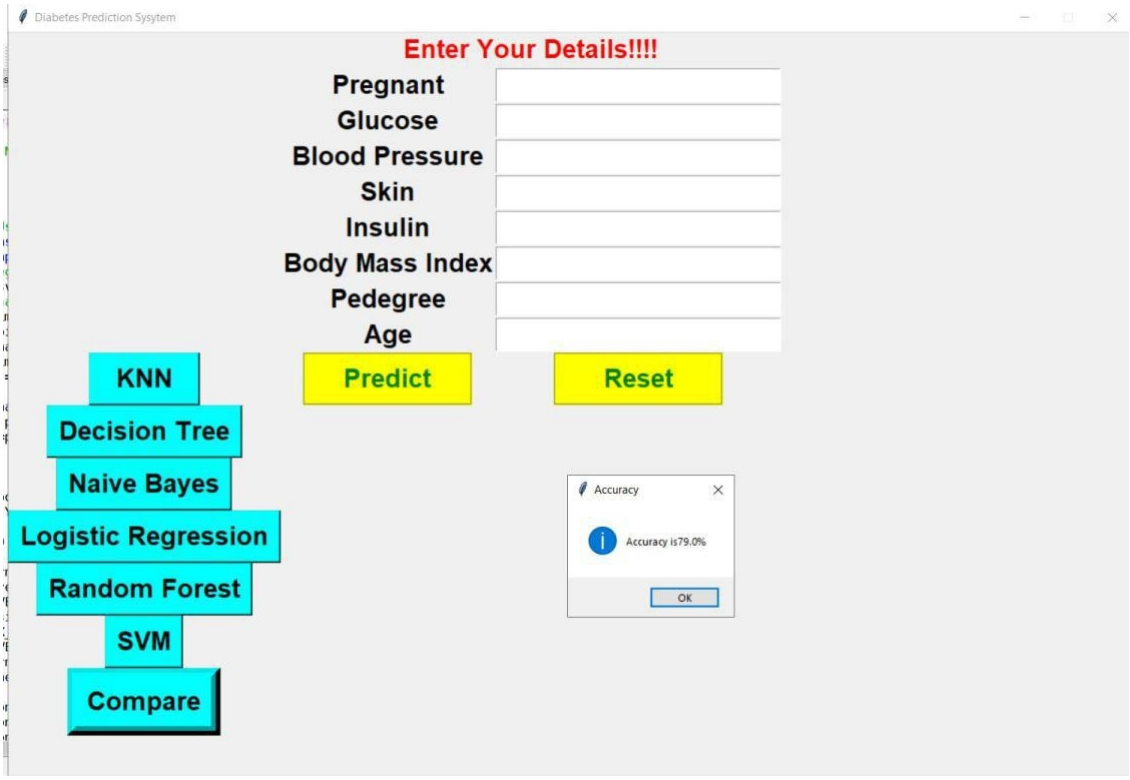
9.3 After clicking on the Logistic Regression button



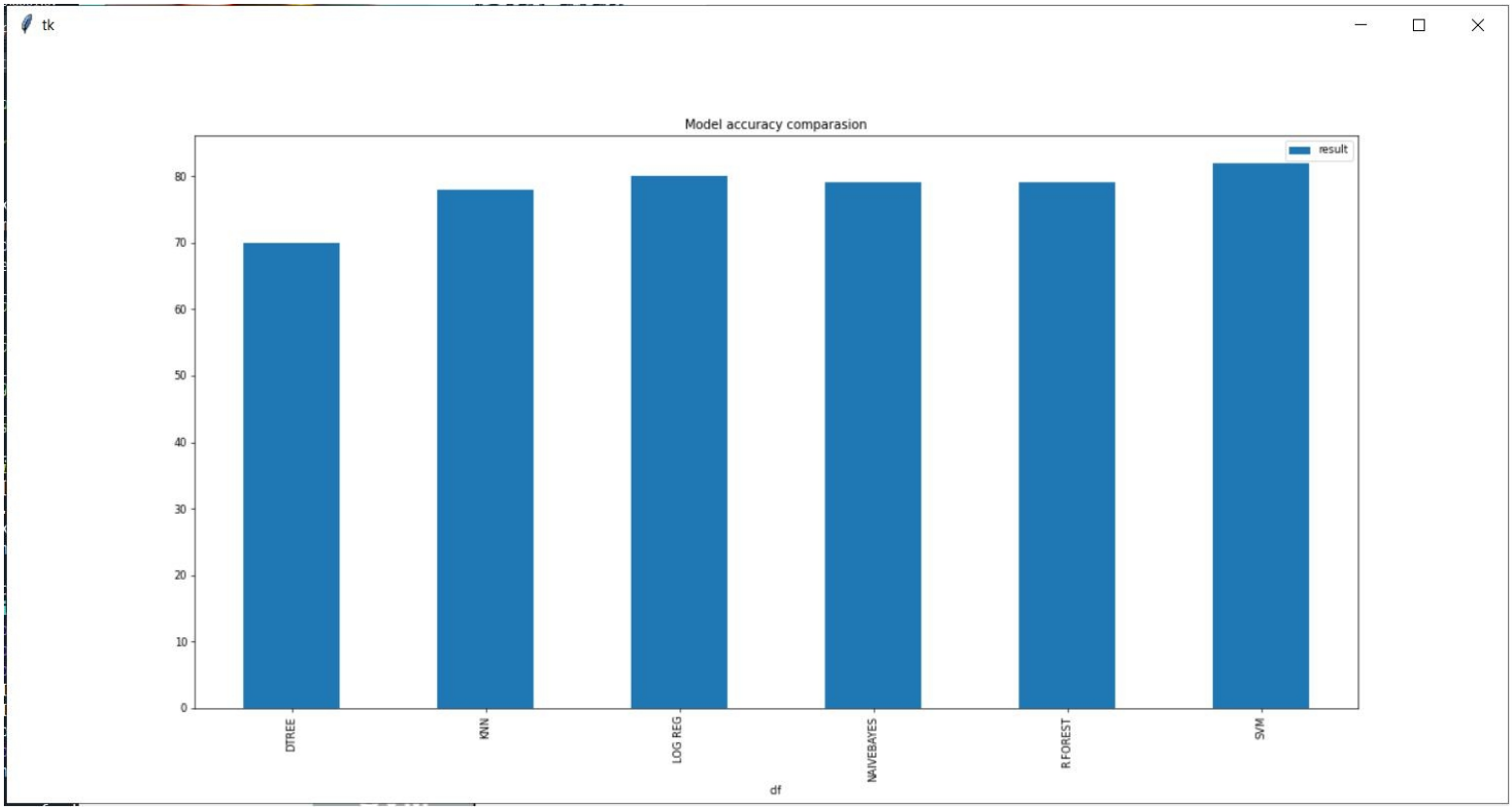
9.4 After clicking on the Naive Bayes button



9.5 After clicking on the Decision Tree button



9.6 After clicking on the Compare button



9.7 After entering the data and clicking on the Predict button

9.8 Then after clicking on the Reset button

Chapter 11

Conclusion and Future Scope

10.1 FUTURE SCOPE

We tried and optimize every algorithm and we found out Logistic regression algorithm best suitable for over application.

But this is just not the end. We can also implement other classification algorithms to obtain more accurate and optimized result.

10.2 CONCLUSION

Machine learning algorithms in the medical field extracts different hidden patterns from the medical data. They can be used for the analysis of important clinical parameters, prediction of various diseases, forecasting tasks in medicine, extraction of medical knowledge, therapy planning support and patient management. A number of algorithms were proposed for the prediction and diagnosis of diabetes. These algorithms provide more accuracy than the available traditional systems. We tried and optimize every algorithm and we found out Logistic regression algorithm best suitable for over application.

References

1. UCI Machine Learning Repository- Center for Machine Learning and Intelligent System, http://archive.ics.uci.edu/ml/

2. Gaganjot Kaur, Amit Chhabra, "Improved J48 Classification Algorithm for the Prediction of Diabetes", International Journal of Computer Applications (0975 – 8887) Volume 98 – No.22, July 2014.

3. Machine Learning tutorials and examples https://www.toptal.com/machine-learning/machine- learningtheory-an-introductory-primer .

4. Sudip Mandal, Goutam Saha, Rajat K. Pal, "A Comparative Study on Disease Classification using Different Soft Computing Techniques",The SIJ Transactions on Computer Science Engineering & its Applications (CSEA), Vol. 2, No. 3, May 2014.

5. Classification Techniques" IJISET - International Journal of Innovative Science, Engineering & Technology, Vol. 1 Issue 6, August 2014.

6. http://www.diabetes.ca/about-diabetes/types-of-diabetes

7. http://www.diabetes.org/diabetes-basics/type

8. G. F. Cooper. A simple constraint-based algorithm for efficiently mining observational databases for causal relationships. Machine Learning, 1(2):203–224, 1997