

A PROJECT REPORT
on
“DIABETES PREDICTION SYSTEM”

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
APRIL 2021

A PROJECT REPORT
on
“DIABETES PREDICTION SYSTEM”

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
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**

This is been recorded as a bonafide work carried out by them, in the partial fulfillment 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: 14/04/2021

(Prof. TANMOY MAITRA)
Project Guide

Acknowledgements

We are extremely grateful to Prof. Tanmoy Maitra for his insightful guidance and continual encouragement throughout to ensure that this project maintains its target from its inception to its completion. His constant advice helped us to complete the project within the required time frame. Last but not the least, we would like to thank our project colleagues for helping us solve all the problems we faced and for providing us with a pleasant experience.

GEETANSH VERMA

SAURABH KUMAR

SINDHUSUTA MOHANTY

ANSUMAN SAHOO

ABSTRACT

Diabetes is taken into account together of the deadliest and chronic diseases which causes a rise in blood glucose. There are numerous complications if diabetes is not treated and not identified. The time-consuming identification process leads to a patient's visit to a medical diagnosis and consultation centre. But the increase in machine learning approaches solves this critical problem.

The purpose of this study is to develop a model that can predict the probability of diabetes in patients with the greatest precision. Therefore 6 machine learning classification algorithms namely Decision Tree, KNN ,Logistic regression ,Naive Bayes,Random Forest,SVM are utilized in this experiment to detect diabetes at an early stage. Experiments are conducted using the Pima Indian Diabetes (PIDD) database from the UCI Machine Learning Repository[1].

Keywords of the content: **Diabetes, Decision Tree, KNN, Logistic regression,**

Naive BayesRandom forest, SVM

Contents

1. Introduction	
2. Literature Survey	2
3. Software Requirements Specification	3
3.1 System Requirement.....	3
3.2 Technical 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
6.1 Project Planning	9
7. Project Planning Included.....	9
8. Implementation	10
8.1 Collect the Dataset.....	10
8.2 Load the dataset.....	10
8.3 Describe the Dataset.....	10
8.4 Analysis of the dataset for cleaning	10
8.5 Split the dataset for training and testing.....	11
8.6 Train the dataset using algorithms	
8.7 Compare the accuracy of all algorithms	11-14
8.8 Create GUI for the model.....	14-18
9. Hyper parameter Optimization	19
9.1 Hyper 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
10.1 Data Entry page.....	23
10.2 After clicking on KNN button....	23
10.3 After clicking on Logistic regression button....	24
10.4 After clicking on Naive Bayes button....	24
10.5 After clicking on Decision tree button....	25
10.6 After clicking on compare button....	25
10.7 After entering data and clicking on predict button....	26
10.8 After clicking on Reset button....	26
11. Conclusion and Future Scope	27

11.1 Conclusion	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 great deal of knowledge is generating. This data contains many useful information which will be wanted to take efficient medical decisions. Hence it's 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 individuals affected by Diabetes are increasing per day. Diabetic Mellitus (DM)

belongs to the family of Non Communicable Diseases (NCD). It's one among the damaging diseases because it has future impacts on patient's body. Diabetic Mellitus (DM) is categorized into three types. First is Type 1 DM which is named as Insulin - Dependent DM (IDDM). Type 1 DM is caused when patient's body is unable to supply insulin and requires injecting insulin externally to the patient. The second sort of DM is Type 2 DM which is named as type II diabetes Mellitus (NIDDM), this sort of diabetes is caused when patient's body cell aren't ready to use insulin properly. Third sort of DM is gestational diabetes which is caused in pregnant women thanks to the event of high blood glucose without the knowledge of Pre-diagnosis of Diabetes. This sort of DM is extremely dangerous and may cause Type 2 DM. As numbers of patients of this serious disease are increasing day by day, the dimensions of diabetic data set is additionally increasing. With the use of machine learning algorithm, it is often possible to research the diabetes data and accordingly provide early diagnosis and better treatment.

Machine learning is that the set of various methods which will be wanted to find patterns from the dataset then use those patterns to predict future conditions or to form efficient decisions under some conditions. Machine learning can make decisions independently at its own.

Machine learning is categorized into 2 types, supervised learning and unsupervised learning.

Chapter 2

Literature Survey

Orabi et al. have developed a diabetes prediction system, the primary purpose of which is the prediction of diabetes that an applicant is suffering at a particular age. The proposed systems are based on the concept of automatic learning, through the application of the decision tree. Obtained results were satisfactory because the designed system works well in predicting the diabetes incidents at a specific age, with higher accuracy using Decision tree.

Pradhan et al. used Genetic programming (GP) for the training and testing of the database for predict of diabetes by using Diabetes data set which is generated from UCI repository. The results obtained through genetic programming offer optimum precision in relation to the other techniques used. Significant improvement in accuracy can occur by taking less time to generate classifiers. It helps in the prediction of low-cost diabetes.

Rashid et al. In designed a prediction model with two sub-modules to predict diabetics-chronic. Disease ANN issued within the first module and FBS(Fasting Blood Sugar) is employed within the second module. Decision Tree is employed to detect the symptoms of diabetes on patients health.

Nongyao et al. in applied an algorithm which classifies the danger of DM . to satisfy the target author 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 OS

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 Guido van Rossum and first released in 1991, it's a design philosophy that emphasizes code readability, notably using significant white space. It provides constructs that enable clear programming on both small and enormous scales.

PyCharm :- It is developed by Jet brains, which is associated with an integrated development environment employed in programming and specifically for the Python language.

Jupyter Notebook :- It is developed by Jet brains, which is an integrated development environment employed in creating mental models and specifically for the Python language.

Spyder : - It is developed by Jet brains, which is an integrated development environment employed in creating mental models and specifically for the Python language.

3.1 Libraries Requirement:

Pandas: - It is developed by Jet brains, which is AN Integrated Development surroundings employed in programing and specifically for the Python language.

Matplotlib: -It is developed by Jet brains, which is associate degree IntegratedDevelopment surroundings utilized in programing and specifically for the Python language.

NumPy: - It is a library for the Python programming language , adding support for giant , multi-dimensional arrays and matrices, along side an outsized collection of high-level mathematical functions to work on these arrays.

Scikit-learn : *A library in Python that gives many unsupervised and supervised learning algorithms.*

The useful 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 that the standard GUI library for Python. Python when combined with Tkinter provides a quick and straightforward thanks to create GUI applications.

Tkinter provides a strong object-oriented interface to the Tk GUI toolkit.

Creating a GUI application using Tkinter is a simple task. All we'd like to try to to is perform the subsequent steps .

1. We should import the *Tkinter* module in code.
2. Then after we should create the GUI application main window.
3. Then we should include one or more of the above-mentioned widgets to the GUI application.
4. Finally,we should Enter the most event loop to require 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 created from UCI machine learning repository.

PIDD: This dataset is originally from the National Institute of Diabetes and Digestive and Kidney Diseases. the target of the dataset is to diagnostically predict whether or not a patient has diabetes, supported certain diagnostic measurements included within the dataset. Several constraints were placed on the choice of those instances from a bigger database. especially , all patients here are females a minimum of 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 dataset are numeric. For this work the dataset is taken into account 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 named impaired glucose tolerance. this is often called "pre- diabetes." It means you're at increased risk of developing diabetes over time. A glucose level of 200 mg/dL (11.1 mmol/L) or higher is employed to diagnose diabetes.

3.Blood Pressure:

Blood Pressure(mmHg): In case Diastolic B.P > 90 implies Tall B.P(High Probability of Diabetes) Diastolic B.P < 60 implies low B.P (Less Likelihood of Diabetes)

2. Skin Thickness: Triceps Skin Fold Thickness (mm) – A value used to estimate body fat. Normal Triceps SkinFold Thickness in women is 23mm. Greater thickness leads towards obesity and chances of diabetes also increases.

3. Insulin: Two-Hour Serum Insulin (mu U/ml) Normal Insulin Level 16-166 mIU/L Values above this range can be distress.

4. 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.

5. Diabetes Pedigree Function: It provides information about diabetes past in relatives and genetic relationship of those relatives with patients. Higher Pedigree Function means patient is more likely to possess diabetes.

6. Age (years)

7. 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 Seventy percent(70%) Pima Indian population is suffering from the diabetes.

8. 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

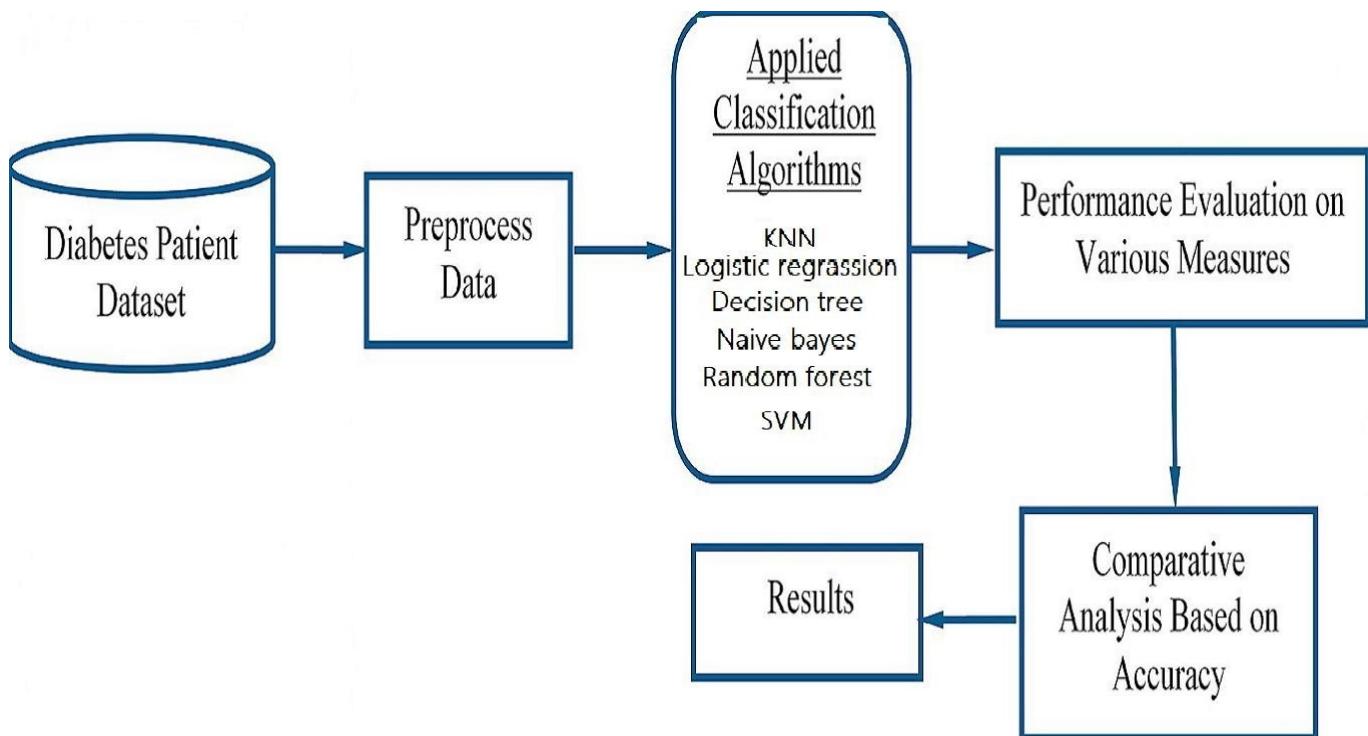


Fig :1.1(model diagram)

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 been 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 been 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)

	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 been 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)
T03				

School of Computer Engineering, KIIT, BBSR

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 IMPLEMENTATION

8.1 Collect the dataset:

At first the dataset was collected from the UCI Repository and was saved as csv (comma 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.

```

9  #dataset cleaning and transformation
10 #cleaning blood pressure column
11 median_bp = pima['bp'].median()
12 pima['bp'] = pima['bp'].replace(to_replace=0, value
13 #cleaning skin thickness column
14 median_skin = pima['skin'].median()
15 pima['skin'] = pima['skin'].replace(to_replace=0,
16 #cleaning insulin column
17 median_insulin = pima['insulin'].median()
18 pima['insulin'] = pima['insulin'].replace(to_replace
19 =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.

```

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

```

to split the dataset

8.6

Train the dataset using algorithms

Confusion matrix

Quality metrics must be considered when assessing the adequacy of classifiers. True positives (TP) are positive cases that the classifier correctly labelled, while true negatives (TN) are negative cases that the classifier correctly labelled. False positives (FP) are negative cases that have been incorrectly labelled, while false negatives (FN) are positive cases that have been incorrectly labelled.

[7]

$$\text{Accuracy} = (\text{TN}) / (\text{TP} + \text{TN} + \text{FP} + \text{FN})$$

Using KNN Algorithm

11

The system was thus trained using KNN Algorithm

```

26 #APPLYING KNN ALGORITHM
27 from sklearn.neighbors import KNeighborsClassifier
28 knnaccuracy=0#global
29 KNN=KNeighborsClassifier()#global
30 #user-defined function for implementation of KNN
   algorithm to the dataset
31 - def knn():
32     '''Apply KNN algorithm to data set'''
33
34     global knnaccuracy
35     global KNN
36     KNN=KNeighborsClassifier(n_neighbors=9)
37     KNN.fit(X_train,Y_train)
38     Y_pred=KNN.predict(X_test)
39     #confusion matrix of KNN algorithm
40     from sklearn import metrics
41     confusion=metrics.confusion_matrix(Y_test,Y_pred)
42     print(confusion)
43     TP=confusion[1,1]#True Positive
44     TN=confusion[0,0]#True Negative
45     FP=confusion[0,1]#False Positive
46     FN=confusion[1,0]#False Negative
47     knnaccuracy=round(((TP+TN)/(TP+TN+FP+FN)),2)*100
48     #printing KNN algorithm's accuracy for our model
49     m.showinfo(title="Accuracy",message="Accuracy
      is"+str(knnaccuracy)+"%")
50

```

Training the model using KNN Algorithm

Using Logistic Regression Algorithm

The system was then trained using Logistic Regression Algorithm.

```

51 #APPLYING LOGISTIC REGRESSION
52 from sklearn.linear_model import LogisticRegression
53 LOGREG=LogisticRegression()#global
54 logaccuracy=0#global
55 #user-defined function for implementation of
   Logistic Regression to the dataset
56 - def logreg():
57     '''Apply Logistic Regression to Data Set'''
58
59     global logaccuracy
60     global LOGREG
61     LOGREG=LogisticRegression()
62     LOGREG.fit(X_train,Y_train)
63     Y_pred=LOGREG.predict(X_test)
64     #confusion matrix of Logistic Regression
65     from sklearn import metrics
66     confusion=metrics.confusion_matrix(Y_test,Y_pred)
67     print(confusion)
68     TP=confusion[1,1]#True Positive
69     TN=confusion[0,0]#True Negative
70     FP=confusion[0,1]#False Positive
71     FN=confusion[1,0]#False Negative
72     logaccuracy=round(((TP+TN)/(TP+TN+FP+FN)),2)*100
73     #printing Logistic Regression's accuracy for our
      model
74     m.showinfo(title="Accuracy",message="Accuracy
      is"+str(logaccuracy)+"%")
75

```

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 NAIIVE
    NAIIVE=GaussianNB()
    NAIIVE.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(vage.get())]])
        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=[]
        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 accuracy 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 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 NAIIVE
    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

```

150
151 #user-defined function to reset all the input values
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("")
162

```

Function of Reset button 16

```

162 #Designing
163 #Load the image
164 img=PhotoImage(file="D:/diabetesimage.png")
165 #Label Image
166 lblimage=Label(w,image=img)
167 lblimage.grid(row=1,column=1,rowspan=9)
168 #Label title
169 labeltitle=Label(w,text="Enter Your Details!!!!",fg
170 = 'red',font=('arial',20,'bold'))
171 labeltitle.grid(row=1,column=2,columnspan=2)
172 #Label pregnant
173 labelpreg=Label(w,text="Pregnant",font=('arial',20
174 , 'bold'))
175 labelpreg.grid(row=2,column=2)
176 entrypreg=Entry(w,font=('arial',20,'bold'
177 ),textvariable=vpreg)
178 entrypreg.grid(row=2,column=3)
179 #Label glucose
180 labelglucose=Label(w,text="Glucose",font=('arial',20
181 , 'bold'))
182 labelglucose.grid(row=3,column=2)
183 entryglucose=Entry(w,font=('arial',20,'bold'
184 ),textvariable=vglucose)
185 entryglucose.grid(row=3,column=3)
186 #Label bp
187 labelbp=Label(w,text="Blood Pressure",font=('arial'
188 ,20,'bold'))
189 labelbp.grid(row=4,column=2)
190 entrybp=Entry(w,font=('arial',20,'bold'
191 ),textvariable=vbp)
192 entrybp.grid(row=4,column=3)

```

Designing the page

```

186 #Label skin thickness
187 labelskin=Label(w,text="Skin",font=('arial',20
188 , 'bold'))
189 labelskin.grid(row=5,column=2)
190 entryskin=Entry(w,font=('arial',20,'bold'
191 ),textvariable=vskin)
192 entryskin.grid(row=5,column=3)
193 #Label insulin
194 labelinsulin=Label(w,text="Insulin",font=('arial',20
195 , 'bold'))
196 labelinsulin.grid(row=6,column=2)
197 entryinsulin=Entry(w,font=('arial',20,'bold'
198 ),textvariable=vinsulin)
199 entryinsulin.grid(row=6,column=3)
200 #Label bmi
201 labelbmi=Label(w,text="Body Mass Index",font
202 =('arial',20,'bold'))
203 labelbmi.grid(row=7,column=2)
204 entrybmi=Entry(w,font=('arial',20,'bold'
205 ),textvariable=vbmi)
206 entrybmi.grid(row=7,column=3)

```

Designing the page cont..

```

201 #Label pedigree
202 labelpedigree=Label(w,text="Pedegree",font=('arial',
20,'bold'))
203 labelpedigree.grid(row=8,column=2)
204 entrypedigree=Entry(w,font=('arial',20,'bold'),
),textvariable=vpedegree)
205 entrypedigree.grid(row=8,column=3)
206 #Label age
207 labelage=Label(w,text="Age",font=('arial',20,'bold'))
208 labelage.grid(row=9,column=2)
209 entryage=Entry(w,font=('arial',20,'bold'),
),textvariable=vage)
210 entryage.grid(row=9,column=3)
211
212 #Button Predict
213 btnpredict=Button(w,text="Predict",bg='yellow',width
=10,relief='groove',
214 [ | | | | font=('arial',20,'bold'),fg
='green',command=predict)
215 btnpredict.grid(row=10,column=2)
216 #Button Reset
217 btnreset=Button(w,text="Reset",bg='yellow',width=10
,relief='groove',
218 [ | | | font=('arial',20,'bold'),fg='green'
,command=reset)
219 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'),command
=native)
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 NAIIVE
    NAIIVE=BernoulliNB()
    params = {'alpha': [0.01, 0.1, 0.5, 1.0, 10.0],
               }

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

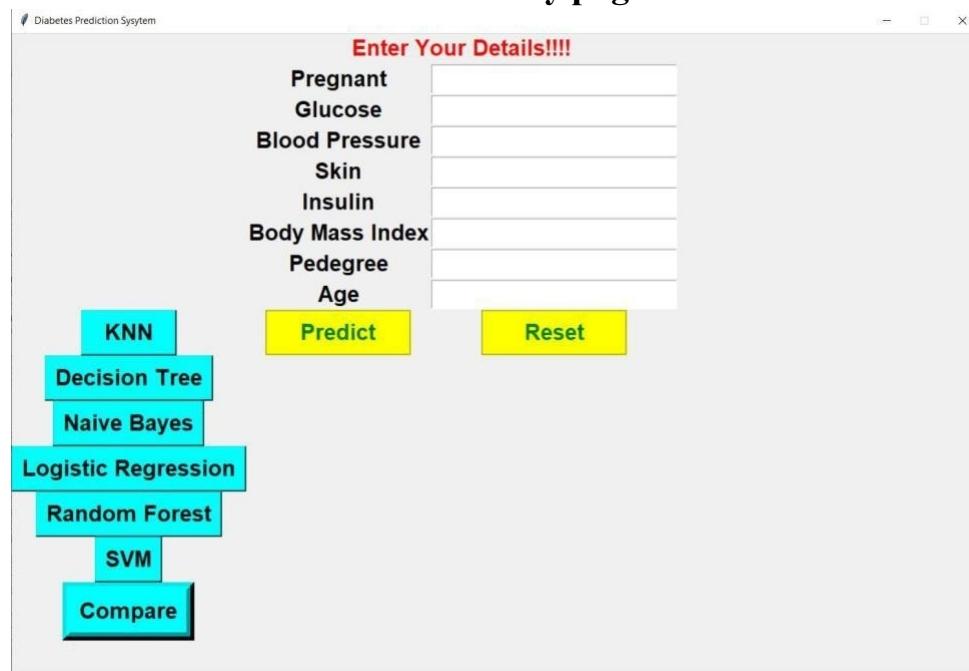
```
from sklearn.svm import SVC
from sklearn import svm
svmaccuracy=0
def SVM():
    '''Apply SVM Classifier to dataset'''

    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

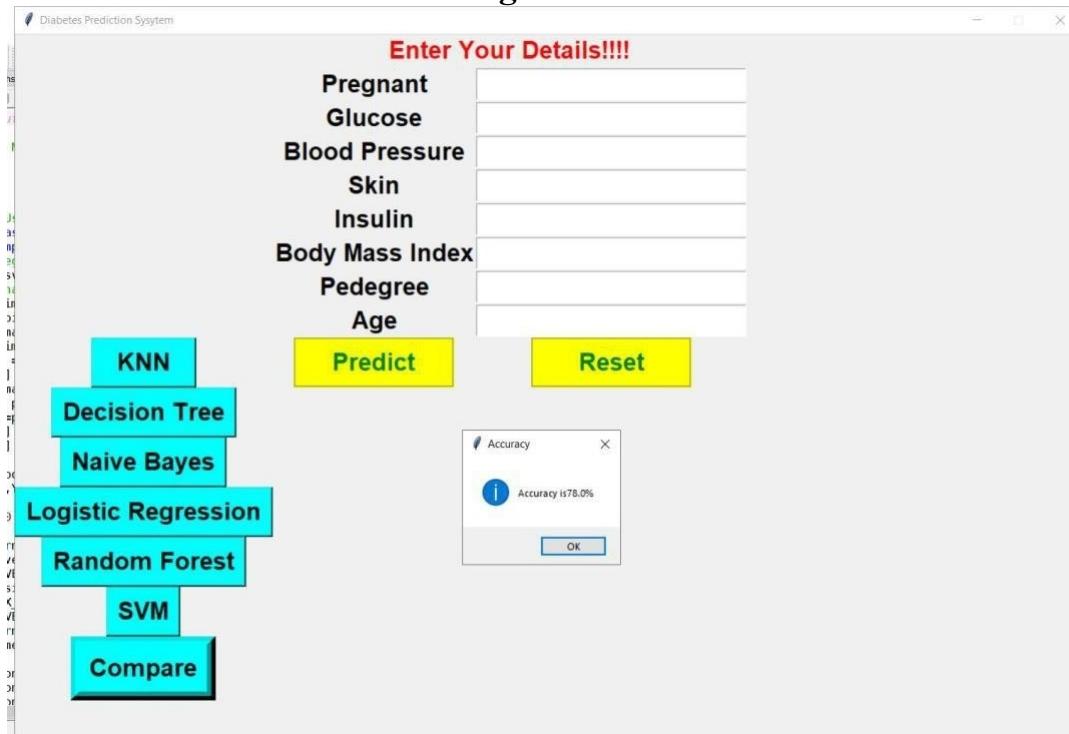
Screen shots of Project

9.1 Data Entry page



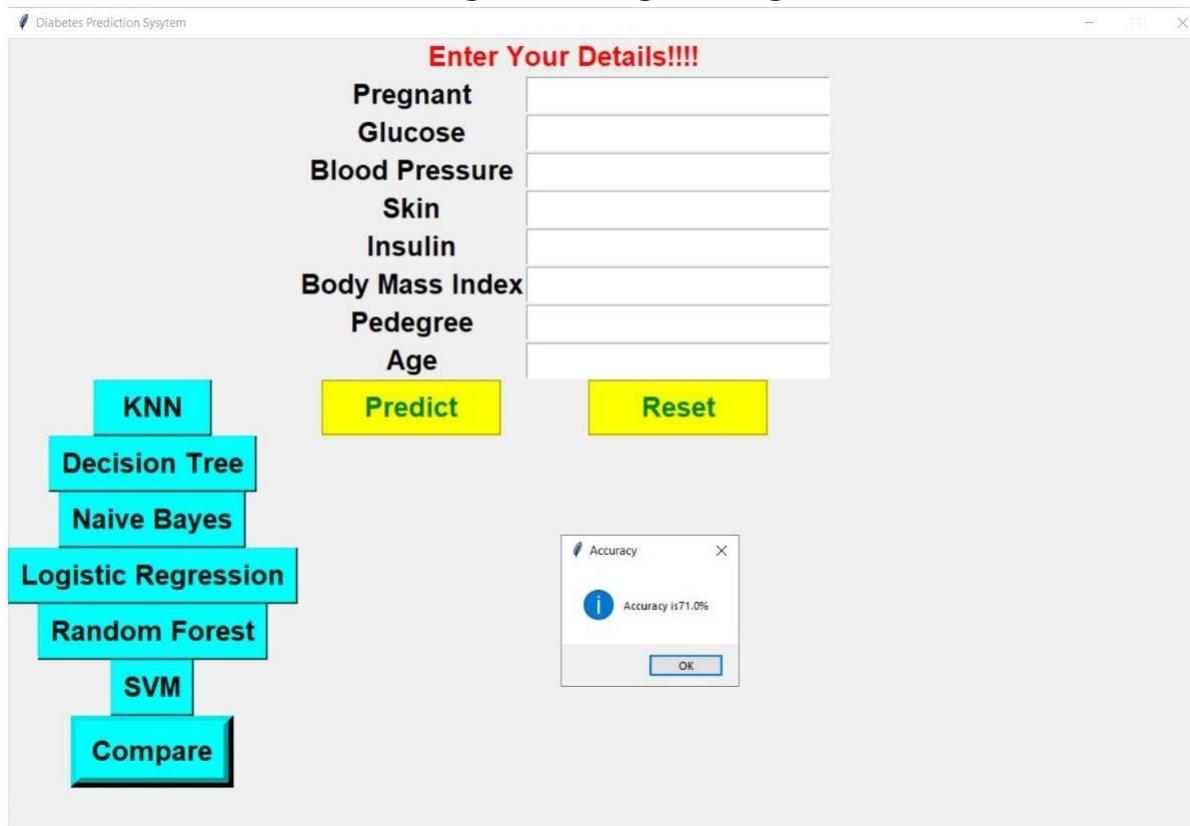
22

9.2 After clicking on the KNN button

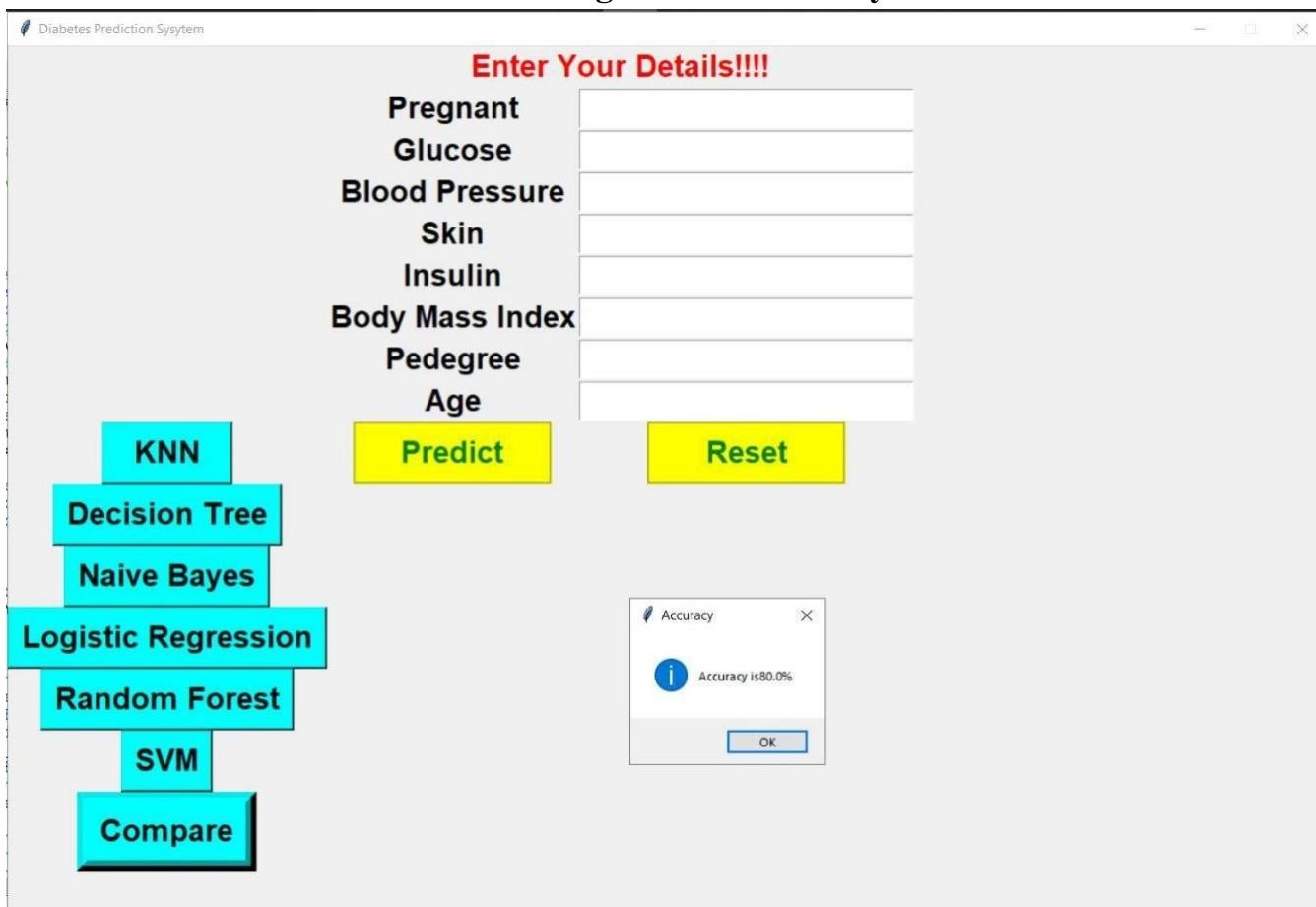


23

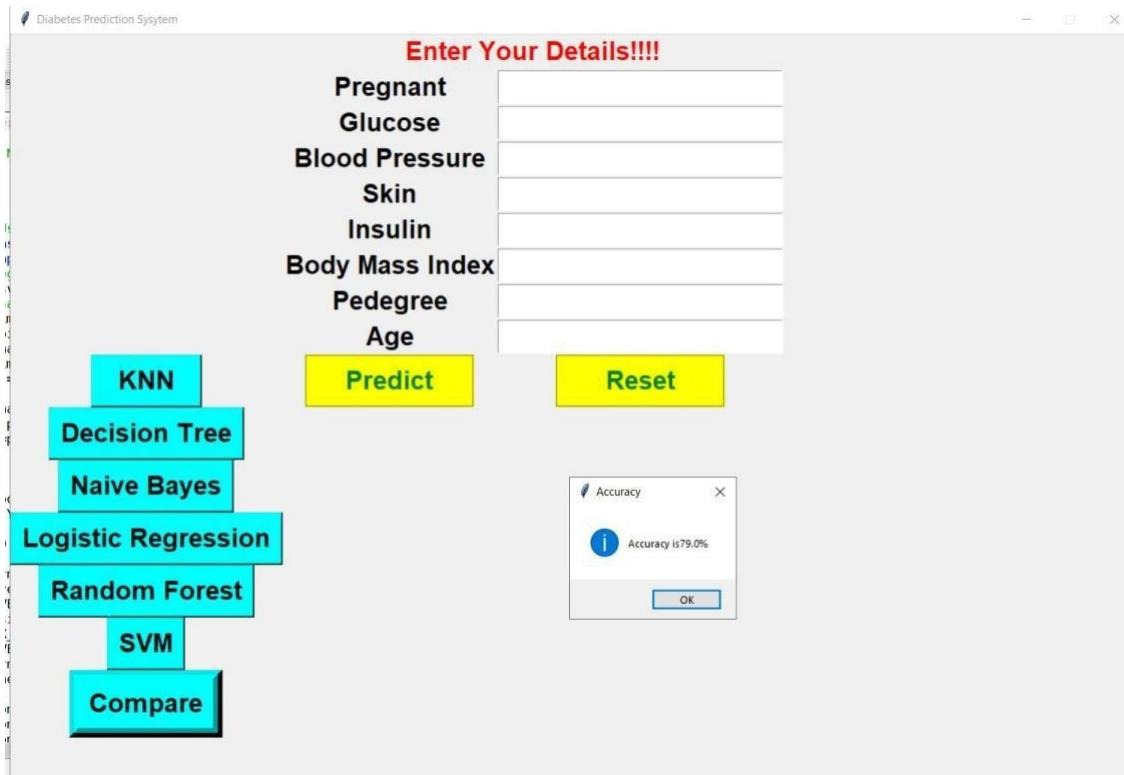
9.3 After clicking on the Logistic Regression button



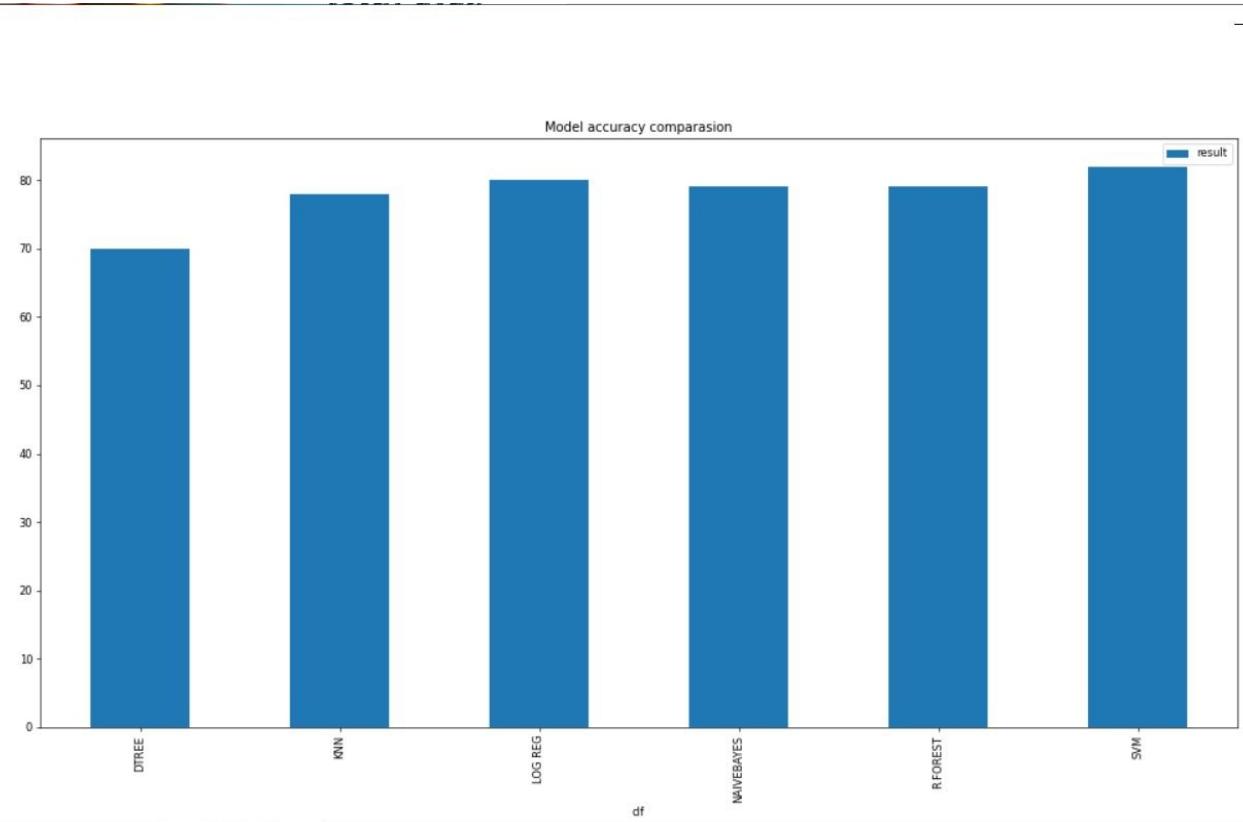
9.4 After clicking on the Naive Bayes button



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 tested and optimized each algorithm and found out that SVM algorithm best suited for more 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 retrieve different hidden models of medical data. They can be used to analyse important clinical parameters, predicting various diseases, medical task forecasting, medical knowledge retrieval, therapeutic 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 that we Acknowledged SVM algorithm best suitable for over application.

References

1. UCI Machine Learning Repository- Center for Machine Learningand 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-learning-theory-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