ANALYSIS AND FUTURE RISK PREDICTION OF DIABETES MELLITUS

A PROJECT REPORT

Submitted by

PRAFFULLA KUMAR DUBEY [Reg No: RA1711003030211]
UDBHAV NARYANI [Reg No: RA1711003030230]

Under the guidance of

Ms. Medhavi Malik

(Assistant Professor, Department of Computer Science & Engineering)

in partial fulfillment for the award of the degree

of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE & ENGINEERING

of

FACULTY OF ENGINEERING AND TECHNOLOGY

Delhi NCR Campus, Modinagar, Ghaziabad (U.P.)

JUNE 2021

SRM INSTITUTE OF SCIENCE & TECHNOLOGY

(Under Section 3 of UGC Act, 1956)

BONAFIDE CERTIFICATE

Certified that this project report titled "**ANALYSIS AND FUTURE RISK PREDICTION OF DIABETES MELLITUS**" is the bonafide work of "**PRAFFULLA KUMAR DUBEY [Reg No: RA1711003030211], UDBHAV NARYANI [Reg No: RA1711003030230]**", who carried out the project work under my supervision. Certified further, that to the best of my knowledge the work reported herein does not form any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

SIGNATURE

Ms. Medhavi Malik
GUIDE
Assistant Professor
Dept. of Computer Science & Engineering

SIGNATURE

Dr. R.P. Mahapatra
HEAD OF THE DEPARTMENT
Dept. of Computer Science & Engineering

Signature of the Internal Examiner

Signature of the External Examiner

ABSTRACT

The number of patients suffering from diabetes in India are increasing rapidly. One of the major reasons behind this can be that most of the population does not take regular check-ups for diabetes. Hence they are unable to take proper precautions and have a risk of getting diabetes in the future. The current methodologies followed by the doctors are manual and can be time consuming, hence we have come up with a solution for this problem, by providing a model in which the user can enter their health report details and know their future risk of having diabetes and if required they can consult a doctor for the same. In this project, we propose to analyse the Pima Indian diabetes dataset and develop an intelligent system for predicting future risk of diabetes using supervised Machine Learning (ML) algorithms. We would be coding in python using Jupyter Notebook for analysis, training and testing of our model. We would be deploying our prediction model on localhost using Spyder Integrated Development Environment (IDE).

ACKNOWLEDGEMENTS

We would like to express our sincere gratitude to our guide, Ms. Medhavi Malik for her guidance, useful advice, valuable support throughout the course of the project and for giving us a clear understanding which enabled us in successful completion of our project.

Praffulla Kumar Dubey

Udbhav Naryani

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABBREVIATIONS

AI	Artificial Intelligence
ML	Machine Learning
DL	Deep Learning
SVM	Support Vector Machine
MLP	Multilayer Perceptron
kNN	K-Nearest Neighbour
BMI	Body Mass Index
GUI	Graphical User Interface
IDE	Integrated Development Environment
ANN	Artificial Neural Networks
NN	Neural Networks
RNN	Recurrent Neural Networks
CNN	Convolution Neural Networks
DNN	Deep Neural Networks

LIST OF SYMBOLS

ρ	Step Function
A	Activation Function
W	Weight provided to model
I	Binary Input
N	Total Amount of weights
P	Pre-activation function
$P(T\|K)$	Posterior Probability
$P(K\|T)$	Likelihood Probability
$P(T)$	Prior Probability
$P(K)$	Marginal Probability

CHAPTER 1

INTRODUCTION

1.1 Introduction

Presently in the world, there are various chronic diseases that are distributed throughout the developed as well as developing countries. Diabetes mellitus, which is these days called as diabetes is such one of these chronic diseases and it is responsible for cutting human life at an early age. It is a group of metabolic disease which marks high blood sugar levels over a prolonged period. It has been observed that diabetes disease is on a rapid increase in Asian country like India. Various health sectors are working to predict these chronic diseases in future hence saving the human life. Exercise plays an important role in recovering diabetes and if a diabetic person eats healthy food and does exercise regularly then there are chances of recovery for them. Various human body parts like eye, heart, nerves and kidney can be damaged by diabetes disease. Currently, in order to diagnose diabetes, the doctors take patients sample of blood and measure the sugar concentration that is present in their blood. This methodology followed by the doctors is time consuming and some other features are like blood pressure levels, insulin levels, age of the patient and Body Mass Index (BMI). There are chances for a person to be diabetic in future if an ancestor of the patient shows a presence of diabetes. Diabetes of two types, Type 1 and Type 2 diabetes. At present there are no intrusive techniques to predict whether the patient has type one diabetes. Type two diabetes does not depends on the insulin levels and it has been observed that it is more common in Pima Indians than it is in any other population on earth. Type 2 diabetes can be cured if it is detected at an early age.

1.2 Machine Learning

The application of Artificial Intelligence (AI) that helps a machine in improving and learning from experiences is known as ML. The main objective of ML is accessing data

and learning from the accessed data using computer programs. A model is trained on the past data and is used for prediction of future data. This model can be stored and deployed in order to build powerful applications.

1.2.1 Support Vector Machine

Support Vector Machine (SVM) is a recent ML algorithm also termed as SVM. SVM algorithms are very closely related to Deep Learning (DL) Neural Networks (NN) algorithm "Multilayer Perceptron (MLP)". SVM converts the actual data into a dataset with higher dimensions using non-linear mapping. SVM can be used for classification as well as numerical prediction. SVM has a margin on either sides of a hyperplane and this hyperplane is used to separate data into two different groups. The expected generalization error can be minimized by increasing the margin and increasing the distance between instances on either sides and the hyperplane. Sub-sections must be numbered as shown in this text.

1.2.2 k-Nearest Neighbours

K-Nearest Neighbour (kNN) provides a large set of rules which can be used to divide the data into groups. This algorithm makes an assumption that similar things exist close to each other. It finds distance between different data items when plotted on a graph. It finds K samples close to an element. Then the most frequent label is assigned to the entire group. As the dataset is small kNN could be well suited to solve our problem.

1.2.3 Naïve Bayes Classifier

Naïve Bayes algorithm is a sequential probabilistic algorithm which performs prediction and classification by using execution and estimation using the Bayes Theorem shown in equation 1.1. In order to find relationships and patterns between diabetes and medical records of a person the problem of quantity of data arises. Data available to solve this problem is limited. Naïve Bayes algorithm could be well suited because it

requires less data for training and it can perform well with numeric data.

$$P(T|K) = P(K|T)P(T)/P(K) \tag{1.1}$$

where,

- $P(T|K)$ is **Posterior Probability**: Probability of hypothesis A on the observed event B.
- $P(K|T)$ is **Likelihood Probability**: Probability of evidence given that probability of hypothesis is true.
- $P(T)$ is **Prior Probability**: Probability of hypothesis before observing the evidence.
- $P(K)$ is **Marginal Probability**: Probability of evidence.

1.2.4 Random Forest Classifier

Random Forest Algorithm uses bagging method to train multiple decision trees. It makes up a model of many different decision trees as a single model in order to maximize the overall result. Each decision tree gives out an outcome and the outcome with the highest frequency is used as the overall outcome of the model as shown in figure 1.1. It is usually used for both classifier based and regression based problems. It adds randomness to our model while forming the trees. Its goal is not to find the best feature when we split a node but finding the most important feature from randomly generated feature sets. This diversity results in formation of a good model.

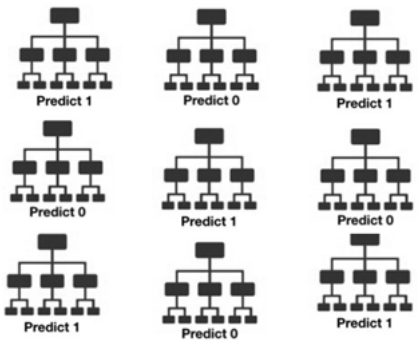

Figure 1.1: Random Forest

1.2.5 Decision Tree

Decision Tree is a flowchart structured like a tree. It is used for predicting and classifying data values by representing them as nodes of a tree. The root node of this tree is used to distinguish and separate instances with different attribute values. The final leaf node represents the class label assigned. The goal is to predict the value or class of a given set of input variable with the help of decision rules formed from training data.

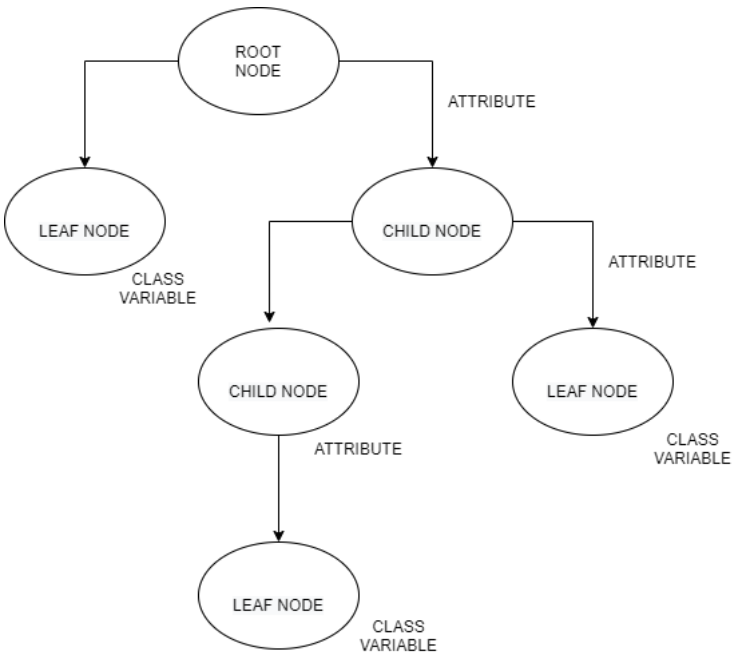

Figure 1.2: Decision Tree

1.2.6 Logistic Regression

Logistic Regression is a type of supervised ML algorithm that basically uses the logistic function in order to perform binary classification of the dependent variables. The result is computed based on the response received from every variable of the "odds ratio" of the event of interest observed. It uses an equation which is used to find the conditional probability of an outcome on specified predictor values.

1.3 Deep Learning

Recently, various ML models developed are being used for forecasting the results or in a simpler term these models are being used for prediction of results. These models have become very famous in the medical diagnosis field also and medical diagnosis can be termed as one of the main problems in medical applications. There has been a significant research for improving the accuracy of different NN for medical datasets as NN improve the medical diagnosis models. There are many NN like Recurrent Neural Networks (RNN), Convolution Neural Networks (CNN) and Artificial Neural Networks (ANN). Artificial Neural Networks also termed as ANN contain various layers of nodes with computing data and these nodes are capable of processing information. These ANN can also find non linearities that are not originating as inputs and they are able to form clusters, association, approximate functions and forecasting, and helps to perform multi factorial analysis to make complex patterns, where we have less prior information. Today, ANN are being preferred over other models because the non-linearity that they posses allow the data to be fitted more accurately, as ANN are noise-insensitive hence this leads to an accurate prediction irrespective of the measurement errors, ANN helps in fast processing and are hardware failure tolerant because of the presence of high parallelism, in the response to the changing environment ANN's learning and the adaptability permits the models to update the model's internal structure with response to the change in the environment. Frank Rosenblatt in 1957 introduced Perceptron and a perceptron learning rule based on the original MCP neuron was proposed by him. A perceptron can be defined as a NN that is capable of doing some computations to detect various features or business intelligence in the data that has been provided as an input whereas a MLP also termed as MLP is a type of feedforward ANN and it utilizes backpropagation for training which is a supervised learning technique.

1.3.1 Perceptron

The perceptron is defined as an algorithm for binary classifiers of supervised learning algorithms where a binary classifier is a function which is used for deciding whether an input provided by the user is part of a specific group or not. It is a form of a linear

classification algorithm where a prediction is made on a linear predictor function by combining feature vectors with different set of weights. Perceptron is one of the simplest and first ANN. It is similar to logistic regression algorithm. It can learn separation feature space for two class classification tasks, although unlike logistic regression, it uses stochastic gradient descent optimization algorithm for learning purpose and perceptron does not predict calibrated probabilities.

Perceptron Algorithm

In the model as shown in figure 1.3, there are binary inputs (I1, I2, ..., IN) and same N is the total amount of weights (W1, W2, ..., WN) provided to the model. All the inputs are now multiplied and then summed up together. This sum is called as pre-activation function and denoted as "P".

$$P = \sum_{i=1}^{n} Weight_i I_i = Weight^T I$$

(1.2)

Now we apply the activation function and this produces an activation which is termed as "A" and this activation function can also be termed as "step function".

$$\rho(y) = 1, y >= 0 \tag{1.3}$$

$$\rho(y) = 0, y < 0 \tag{1.4}$$

The model generates an output 1 if the input value is either 0 or greater whereas, our output value is 0 if we have an input value is lesser than 0.

$$A = \rho(Weight^T I) \tag{1.5}$$

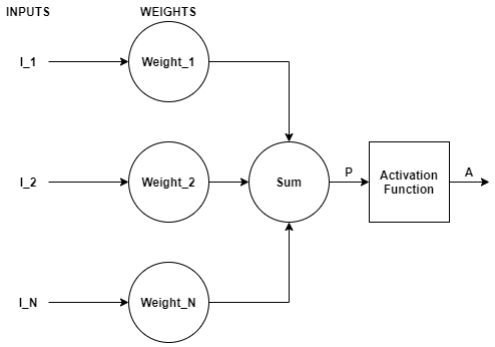

Figure 1.3: Perceptron

1.3.2 Multi-Layer Perceptron

The MLP or MLP algorithm is the most used NN based algorithm. MLP is considered as a type of a feedforward based artificial NN. Generally, the signals transmitted from the network are unidirectional i.e., from input to output. As there is no loop in MLP, hence output does not affect the neuron and this architecture of MLP is termed as feed-forward architecture. There are layers present in MLP that do not have a direct connection to the environment and these layers are called as hidden layers. The input layer is the layer that provides input to the model. The non-linear activation function provides all the strength to MLP. We can use any of the non-linear functions except for polynomial function and most commonly used functions are single pole sigmoid and bipolar sigmoid function.

Multilayer Perceptron Algorithm

In the model as shown in figure 1.4, feed forward type of network is implemented. The main objective of feed forward network is to provide approximation for any function $T()$. Let for a classifier $q = T * (I)$ that is mapping input I to output q, then MLP finds the approximation by mapping $q = T(I;\theta)$ and it learns the best parameters. MLP consists of many functions together. A MLP network with 4 layers will have the following equation:

$$T(I) = T(4)(T(3)(T(2)(T(1)(I))) \tag{1.6}$$

and each layer is represented as:

$$q = A(WeightIP + Bias) \tag{1.7}$$

where,

- A = activation function
- Weight = weights in the layers
- I = the input in form of vectors
- Bias = the bias vector

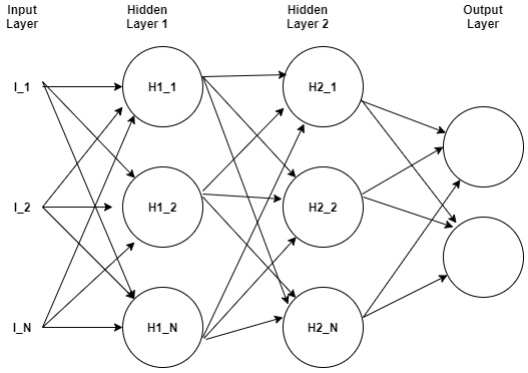

Figure 1.4: Multilayer Perceptron

1.4 Hybrid Model

Hybrid Models are an ensemble of different ML algorithms. They can be used for taking advantage of efficiency of different models on particular sets of data which will in turn improve the accuracy of overall prediction model.

We used stacking to design our Hybrid Model. There are two layers in the model, **base layer (level 0)** and the **meta layer (level 1)**. In base layer we will be using algorithms which gave good accuracy after testing. After testing all the algorithms above we found out kNN, SVM and Decision Tree gave the highest accuracy. In meta layer we used Logistic Regression as our problem is classification based.

CHAPTER 2

LITERATURE SURVEY

2.1 Approach

Literature Survey can be defined as an objective approach or critical analysis of the relevant research being required for the execution of the project. It is the summary of the already done research. Its main goal is to let the readers update themselves with the literature on a topic of research and form the basis for another goal, such as enhancement of future research of the same domain.

Literature Survey is an important aspect for carrying out any research oriented project because if we do have limited time for conducting a research, then doing literature survey gives an overview of the topic. For a project to be successful, research plays an important role.

For our project, we surveyed many online journals and research papers but we only found ten papers that were relevant to our project. These papers were relevant as most of them used the exact same dataset that we have used in our project. We proposed to use ML and DL algorithms so the research papers that we surveyed mostly had ML and DL algorithms and the papers that used any other algorithms were not taken into consideration by us for our project.

2.2 Case Studies

Barik et al. (2021) in their paper "Analysis of Prediction Accuracy of Diabetes Using Classifier and Hybrid Machine Learning Techniques" used XGBoost and Random Forest algorithm and they did not perform data cleaning on the dataset. They achieved the highest accuracy of 74.1 % with XGBoost algorithm. Balaji et al. (2017) in their paper "Optimal predictive analytics of pima diabetics using deep learning" used only RNN for

training and testing their models. Mohebbi et al. (2017) in their paper "A deep learning approach to adherence detection for type 2 diabetics" used CNN and the dataset used by the authors was very small. N. and SriPreethaa (2019) in their paper "Diabetes prediction in healthcare systems using machine learning algorithms on hadoop clusters" used used only Random Forest Algorithm on Hadoop Cluster and the authors discussed the information gain methods for feature selection but they did not mention the pre-processing steps used. Mercaldo et al. (2017) in their paper "Diabetes mellitus affected patients classification and diagnosis through machine learning techniques" used Hoeffding Tree, JRip, BayesNet, Random Forest. The authors used WEKA tool for analysis and prediction but they did not mention the pre-processing steps and the highest accuracy achieved by the authors was 76 % using Hoeffding Tree. Lekha and M. (2018) in their paper "Real-time non-invasive detection and classification of diabetes using modified convolution neural network" used only RNN for training and testing their models. The dataset used by them was very small in size. Kandhasamy and Balamurali (2015) in their paper "Performance analysis of classifier models to predict diabetes mellitus" used J48, kNN and Random Forest algorithm for prediction and they got 73.8 % accuracy with J48 algorithm. The authors did not mention the data pre-processing techniques and methodology used by them. Iyer et al. (2015) in their paper "Diagnosis of diabetes using classification mining techniques" used only two algorithms i.e., Decision Tree and Naive Bayes algorithms and the authors got an accuracy of 79.5% from Naive Bayes classifier. Pre-processing and dataset transformation was done using WEKA tool. Zou et al. (2018) in their "Predicting diabetes mellitus with machine learning techniques." paper used Decision Tree, NN and Random Forest and algorithms. They achieved highest accuracy of 76% with NN. MATLAB tool was used for model creation by the authors. Nanda et al. (2011) in their paper "Prediction of gestational diabetes mellitus by maternal factors and bio-markers at 11 to 13 weeks" used only Logistic Regression for prediction and the accuracy of their model was not mentioned.

CHAPTER 3

SYSTEM ANALYSIS

One in every two Indians suffering from diabetes are unaware of their condition, this could be because of lack of awareness because of this there is a need to predict the future risk of getting diabetes so that proper precautions can be taken before its too late.

3.1 Proposed work

In order to solve the above problem we propose to perform data analysis on the Pima India Diabetes dataset and design a ML algorithm which can be used to predict the future risk of getting diabetes. We will use Jupyter notebooks for analysis and prediction and Spyder IDE for Graphical User Interface (GUI).

3.2 Objective

- Analyse and gain insights from the collected data.
- Develop a diabetes prediction model using supervised ML/DL algorithms.
- Predict future risk of diabetes.
- Develop a user friendly GUI.

3.3 Methodology

The technique of gathering, processing and analyzes data i.e., combine programming, mathematics and statistics domain to extract insights and knowledge from the data is termed as Data Science. This technique is considered to be an important technique for a project as it improves the performance prediction of ML algorithms used i.e., help in predicting the outcome by providing input to the machine. The methodology followed is explained below:

- **Data Import:** It is the step where we collect the dataset. The collected dataset is then split into train data and test data for the prediction model. The dataset can be collected from various resources like online or data stored in various databases. It is important to select the correct dataset because our prediction model is trained using the dataset and if use incorrect model then we would not receive the required results from our trained ML model.
- **Data Cleaning:** Data cleaning plays an important role as the data is collected from various resources and hence the data in coarse in nature. This coarse data is termed as "dirty data". The dirty data may contain some values that might not be required for the model training purpose hence that should be handled and we handle unrequired data in the data cleaning step. Also, the dataset may contain null values in it. It becomes important to handle null values because we can not use the dataset with null values. We can replace the null values with either mean, median or mode values.
- **Data Visualization:** In this technique we visualize the data present in the dataset by plotting various graphs. This technique makes analysis of data easy and helps us to find co-relation among various attributes present in the dataset.
- **Data Analysis:** Analyze the results of various graphs and gain knowledge about the different attributes of the dataset.
- **Select ML/DL Algorithm:** Select the best ML/DL algorithm for creation of prediction model.
- **Prediction Model:** Create the prediction model for predicting diabetes.
- **Design User Interface:** Develop a GUI for the prediction model to be easily used by users.

3.4 Proposed Algorithm

- **Step I:** Collect Diabetes Dataset
- **Step II:** Perform Preprocessing on the data
  - Drop useless columns
  - Fill in missing values with class median.
- **Step III:** Split the data into Training set and Testing set.
- **Step IV:** Create ensemble model
  - **Level 0:** Stack kNN, SVM and Decision Tree in Base Layer.
  - **Level 1:** Make Logistic Regression Model in Meta Layer from the outcomes of base layer.

- **Step V:** Find Predictive Outcome using Testing set on the model using equations 3.1 and 3.2.

$$Ensemble\_Model.fit(train\_x\_data, train\_y\_data) \tag{3.1}$$

$$Predictive\_Outcome = model.predict(test\_x\_data) \tag{3.2}$$

- **Step VI:** Calculate accuracy achieved by the model.

3.5 Dataset Specification

The dataset collected for this project is the Pima India diabetes dataset. This dataset is provided by data world. This dataset consists of medical records of women above the age of twenty-one years. It contains attributes like glucose and insulin which are relevant for prediction of diabetes. Descriptions of each attribute present in the dataset is given below in table 3.1

Table 3.1: Dataset Specification

S.no	**Attribute**	**Description**
1.	Pregnancies	Number of times pregnant.
2.	Glucose	Plasma glucose concentration a 2 hours in an oral glucose tolerance test.
3.	BloodPressure	Pressure of circulating blood against the walls of blood vessels (mm Hg).
4.	SkinThickness	Triceps skin fold thickness (mm).
5.	Insulin	Hormone that lowers the level of glucose in blood (mu U/ml).
6.	BMI	Body mass index (weight in kg/(height in m)^2).
7.	DiabetesPedigree Function	A function which scores likelihood of diabetes based on family history.
8.	Age	Age (in years).
9.	Outcome	Class variable (0 or 1), 0 = Non Diabetic 1 = Diabetic
10.	Doctor	Name of Doctor in-charge.
11.	Hospital	Hospital Name.

CHAPTER 4

SYSTEM DESIGN

4.1 Architectural Design

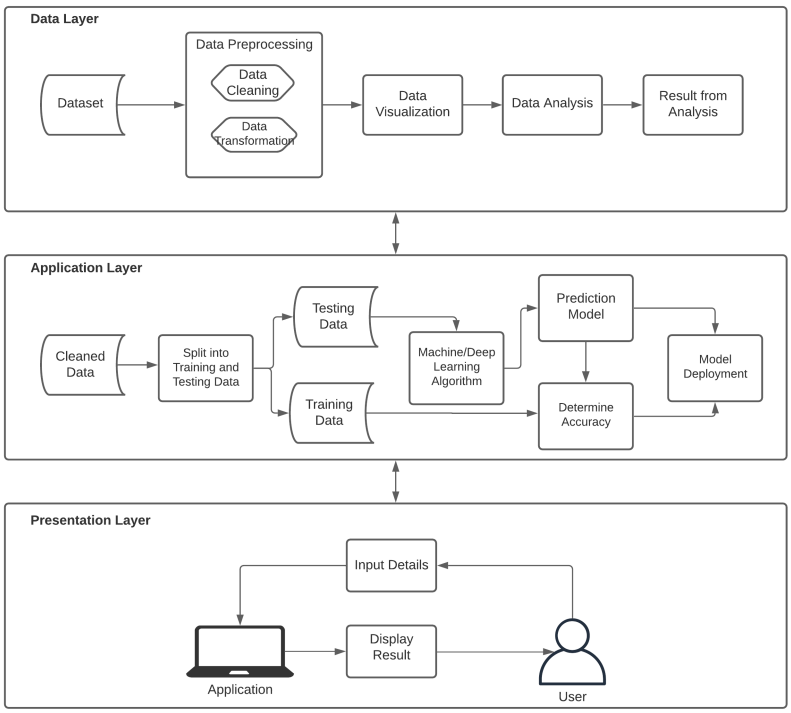

Figure 4.1: Architectural Design

The architectural design for the overall system is divided into 3 layers as shown in figure 4.1 above :

- **Data Layer:** Pre-process the data and gain knowledge from it.
- **Application Layer:** Create, train and test the diabetes prediction model.
- **Presentation Layer:** Create a GUI for the user to use the diabetes prediction model.

4.2 Use Case Diagram

There are two types of users in our system as shown in figure 4.2:

- **Admin:** Admin gathers, cleans and analyzes the data. They also develop the prediction model and user interface.
- **End User:** End user inputs the required information into the system i.e. the health details. Then they can submit the details to the server and view the prediction results.

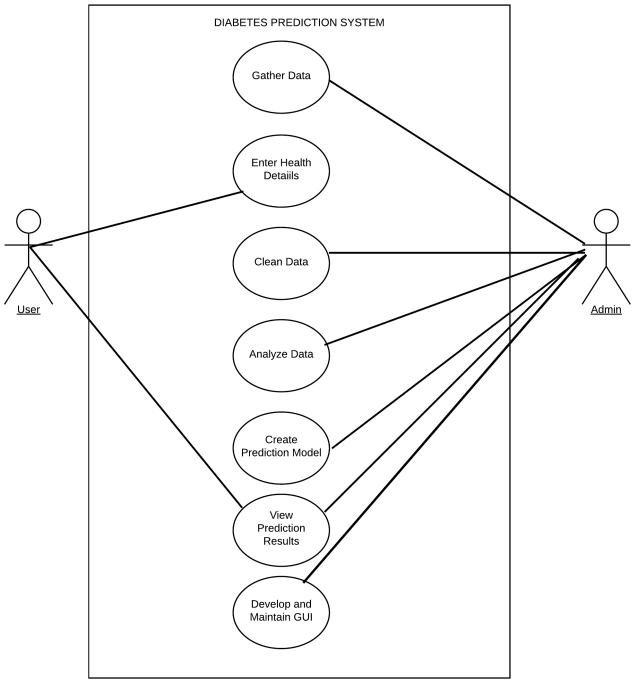

Figure 4.2: Use Case Diagram

4.3 Data Flow Diagram

Data flow diagram shows the flow of data and processes in the system before and at the time of evaluation. First in order to make the gathered data useful it is pre-processed. Then data visualisation is performed in order to better understand the data. Then the data is used to train the ML models. The models are tested and the model with highest accuracy is selected for prediction. This prediction model is saved and used with the

GUI as shown in figure 4.3.

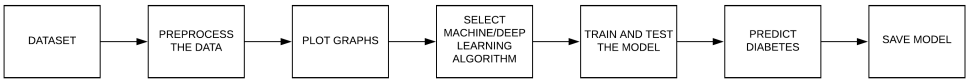

Figure 4.3: Data Flow Diagram

4.4 Sequence Diagram

Sequence diagram is used to depict how the different steps take place in the system. It also helps in identifying the order of different steps and actions taking place in the system. In the beginning the end user interacts with the GUI. During this interaction user opens the application, enters health details and submits the data. The GUI transfers the collected information to prediction model. The model predicts the result and sends it to the GUI which in turn displays the prediction result. Finally the user views the prediction result. The sequence diagram for the system is shown below in figure 4.4

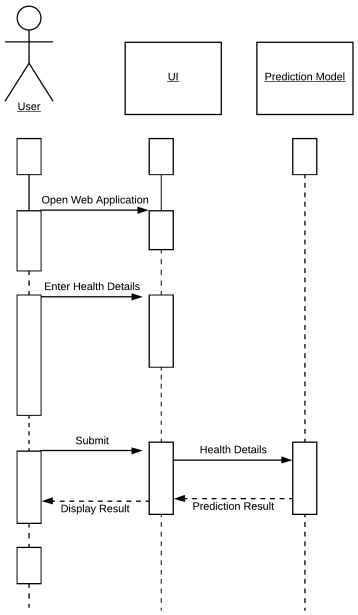

Figure 4.4: Sequence Diagram

CHAPTER 5

CODING AND MODEL TESTING

In this project we would perform analysis and prediction on Diabetes dataset. Our main goal would be to extract information from the data and get a high accuracy from our prediction model. First, we will explore the dataset which would in-turn help us with data pre-processing and cleaning. After pre-processing and cleaning we will visualise and analyse the data in order to extract meaningful information. Finally we will design a prediction model with the aim to maximize accuracy.

5.1 Coding

Import the Required Libraries

We are using "pandas" and "numpy" to store and manipulate data. "seaborn" and matplotlib are used for data visualization. The "missingno" library is used to find out the missing data so that it can be handled properly. "train_test_split", "metrics" and "accuracy_score" are used for training and testing of our model.

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
import missingno as msno
from sklearn.model_selection import train_test_split
from sklearn import metrics
from sklearn.metrics import accuracy_score
```

Data Cleaning

In our dataset the columns that have ’0’ are null values, hence for cleaning purpose we are replaced the ’0’ values in columns with ’NaN’.

```
df1[['Glucose','BloodPressure','SkinThickness','Insulin','BMI']] =
                                    df1[['Glucose','BloodPressure','
                                    SkinThickness','Insulin','BMI']].
                                    replace(0,np.NaN)
```

The columns "Doctor" and "Hospital" are not relevant to our project as they cannot be used to predict the future risk of getting diabetes.

```
df1.drop(['Doctor'], axis = 1, inplace = True)
df1.drop(['Hospital'],axis = 1, inplace = True)
```

As there are a lot of missing values in the dataset, we handled them by replacing them with the "class median".

```
def outcome_median(column_name): #function to find median
    median = df1[df1[column_name].notnull()]
    median = median[[column_name, 'Outcome']].groupby(['Outcome'])[[
                                        column_name]].median().
                                        reset_index()
    return median
df1.loc[(df1['Outcome'] == 0 )&(df1['Attribute_Name'].isnull()),'
                                    Attribute_Name'] = outcome_median(
                                    'Attribute_Name')
```

Data Analysis

We plotted a "heatmap" and a "pairplot" in order to analyze the relationships between different attributes present in the dataset.

```
sns.heatmap(df1.corr(), annot = True)
plt.title('Heatmap for the Dataset')
sns.pairplot(data = df1, hue = 'Outcome')
```

Prepare Data for Training and Testing

For training and testing of our models we splitted the data into two in such a way that 80% data would be used for training and 20% would be used for testing. We will have constant random state so that the model gives consistent results every time it runs.

```
X_train,X_test,y_train,y_test=train_test_split(df1.drop('Outcome',
                                    axis = 1),df1['Outcome'],test_size
                                     = 0.20,random_state = 101)
```

k-Nearest Neighbour Algorithm

```
from sklearn.neighbors import KNeighborsClassifier # Importing
                                    libraries
from sklearn import neighbors
n = KNeighborsClassifier() # Making the model
n.fit(X_train,y_train) # Fitting training data to the model
y_expect = y_test
y_pred1 = n.predict(X_test) # Testing the model
accuracy_KNN = accuracy_score(y_test, y_pred1) # Calculating Accuracy
print(accuracy_KNN*100)
```

After implementing the kNN algorithm it was found that the accuracy of model is best at n = 5 (default) i.e. 88.31 %.

Logistic Regression

```
from sklearn.linear_model import LogisticRegression # Importing
                                    libraries
LRModel = LogisticRegression()  # Making the model
```

```
LRModel.fit(X_train,y_train) # Fitting training data to the model
y_expect = y_test
y_pred4 = LRModel.predict(X_test) # Testing the model
accuracy_LR = accuracy_score(y_test, y_pred4) # Calculating Accuracy
print(accuracy_LR*100)
```

After implementing the Logistic Regression algorithm it was found that the accuracy of model 78.57 %.

Decision Tree

```
from sklearn import tree # Importing libraries
from sklearn.tree import DecisionTreeClassifier
clf = tree.DecisionTreeClassifier(random_state=0, max_depth=2) #
                                    Making the model
clf.fit(X_train,y_train) # Fitting training data to the model
y_expect = y_test
y_pred5 = clf.predict(X_test)
accuracy_DT = accuracy_score(y_test, y_pred5) # Calculating Accuracy
print(accuracy_DT*100)
```

After implementing the Decision Tree algorithm it was found that the accuracy of model is 85.71 %.

Random Forest

```
from sklearn.ensemble import RandomForestClassifier # Importing
                                    libraries
ranfor = RandomForestClassifier(n_estimators = 11, criterion = '
                                    entropy', random_state = 42) #
                                    Making the model
ranfor.fit(X_train, y_train) # Fitting training data to the model
y_expect = y_test
y_pred6 = ranfor.predict(X_test)
accuracy_ranfor = accuracy_score(y_test, y_pred6) # Calculating
                                    Accuracy
```

```
print(accuracy_ranfor*100)
```

After implementing the Random Forest algorithm it was found that the accuracy of model is 83.76 %.

Naïve Bayes Classifier

```
from sklearn.naive_bayes import GaussianNB # Importing libraries
nb = GaussianNB() # Making the model
nb.fit(X_train, y_train) # Fitting training data to the model
y_expect = y_test
y_pred7 = nb.predict(X_test)
accuracy_nb = accuracy_score(y_test, y_pred7) # Calculating Accuracy
print(accuracy_nb*100)
```

After implementing the Naive Bayes algorithm it was found that the accuracy of model is 77.27 %.

Support Vector Machine Algorithm

```
from sklearn import svm # Importing libraries
SM = svm.SVC() # Making the model
SM.fit(X_train, y_train) # Fitting training data to the model
y_expect = y_test
y_pred8 = SM.predict(X_test)
accuracy_SVM = accuracy_score(y_test, y_pred8) # Calculating Accuracy
print(accuracy_SVM*100)
```

After implementing the SVM algorithm it was found that the accuracy of model is 87.01 %.

Perceptron Algorithm

```
from sklearn.linear_model import Perceptron # Importing libraries
p = Perceptron(random_state=42,max_iter=10,tol=0.001) # Making the
                                                     model
p.fit(X_train, y_train) # Fitting training data to the model
```

```
print("Accuracy on test set: {:.2f}".format(p.score(X_test, y_test)))
                                    # Calculating Accuracy
```

After implementing perceptron algorithm the accuracy was found out to be 67%.

Multilayer Perceptron

```
from sklearn.neural_network import MLPClassifier # Importing
                                    libraries
mlp = MLPClassifier(random_state=42) # Making the model
mlp.fit(X_train, y_train) # Fitting training data to the model
print("Accuracy on test set: {:.2f}".format(mlp.score(X_test, y_test)
                                    )) # Calculating Accuracy
```

After implementing multilayer perceptron algorithm the accuracy was found out to be 84%.

Hybrid Model

```
from sklearn.ensemble import StackingClassifier # Importing library
def get_stacking(): # Function to make the base layer(level 0) for
                                    the model
    level0 = list()
    level0.append(('knn', KNeighborsClassifier()))
    level0.append(('svm', svm.SVC()))
    level0.append(('dt', DecisionTreeClassifier()))
    level1 = LogisticRegression()
    model = StackingClassifier(estimators=level0, final_estimator=
                                        level1, cv=None)
    return model
HybridModel = get_stacking() # Making the model
HybridModel.fit(X_train,y_train) # Fitting training data to the model
y_expect = y_test
y_pred9 = HybridModel.predict(X_test)
print(metrics.classification_report(y_expect,y_pred9))
```

After implementing hybrid model algorithm the accuracy was found out to be 90.6%

Saving the Model

```
import pickle
filename = 'hybrid_model.pkl'
pickle_out = open(filename, 'wb')
pickle.dump(HybridModel, pickle_out)
pickle_out.close()
```

Making GUI for the Model

```
import streamlit as st # Importing libraries
import pickle
pickle_in = open('hybrid_model.pkl','rb')
classifier = pickle.load(pickle_in) # Loading the model
#Creating GUI
st.title('Diabetes Prediction Application')
name = st.text_input("Name:")
pregnancy = st.number_input("Enter Number of pregnancy:")
glucose = st.number_input("Plasma Glucose Concentration:")
bp = st.number_input("Blood pressure (in mm Hg):")
skin = st.number_input("Triceps skin fold thickness (in mm):")
insulin = st.number_input("2-Hour serum insulin: ")
bmi = st.number_input("Body mass index (weight in kg/(height in m)^2)
                                    :")
dpf = st.number_input("Family History of Diabetes (0 = no, 1 = yes):"
                                    )
age = st.number_input("Age:")
submit = st.button('Press to Predict')
# Predicting outcome and displaying result
if submit:
    prediction = classifier.predict([[pregnancy, glucose, bp, skin,
                                      insulin, bmi, dpf, age]])
    if prediction == 0:
        st.write('Congratulations ',name,' you do not have a risk of
                                          being diabetic at present.
                                          ')
    else:
```

```
        st.write('Sorry ',name,' you have a risk of being diabetic.
                                                Please consult the doctor
                                                as soon as possible')
```

5.2 Testing

5.2.1 Working of Model

Testing is done by manually providing input in order to check whether the model is working or not.

Model predicted that the person has low risk of being diabetic (denoted by "0" in figure 5.1) which is correct according to the dataset.

```
x4 = ['Pregnancies','Glucose','BloodPressure','SkinThickness','Insulin','BMI','DiabetesPedigreeFunction','Age']
data4 = [4,130,79,35,120,33.6,0.627,50]
paitentid_HM = pd.DataFrame([data4],columns = x4)
paitentid_HM.head()
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age
0	4	130	79	35	120	33.6	0.627	50

```
#HybridModel
predictions_diabetes4 = HybridModel.predict(paitentid_HM)
print(predictions_diabetes4)
```

```
[0]
```

Figure 5.1: Manual Test I

Model predicted that the person has a risk of being diabetic (denoted by "1" in figure 5.2) which is correct according to the dataset.

```
x5 = ['Pregnancies','Glucose','BloodPressure','SkinThickness','Insulin','BMI','DiabetesPedigreeFunction','Age']
data5 = [6,288,72,35,150,33.6,0.627,50]
paitentid_HM1 = pd.DataFrame([data3],columns = x5)
paitentid_HM1.head()
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age
0	6	288	72	35	150	33.6	0.627	50

```
#HybridModel
predictions_diabetes5 = HybridModel.predict(paitentid_HM1)
print(predictions_diabetes5)
```

```
[1]
```

Figure 5.2: Manual Test II

5.2.2 Testing Efficiency of the Model

We tested the efficiency of the model shown in figure 5.3 by calculating the below mentioned parameters.

	precision	recall	f1-score	support
0	0.90	0.95	0.93	125
1	0.90	0.81	0.85	67
accuracy			0.90	192
macro avg	0.90	0.88	0.89	192
weighted avg	0.90	0.90	0.90	192

Figure 5.3: Efficiency of the Model

Accuracy

Accuracy of a model represents the amount of correct predictions for the testing dataset. It can be calculated by ***ModelAccuracy = TrueNegatives + TruePositives / (TrueNegatives + FalseNegatives + TruePositives + FalsePositives***).
For our model the accuracy came out to be 90.6%.

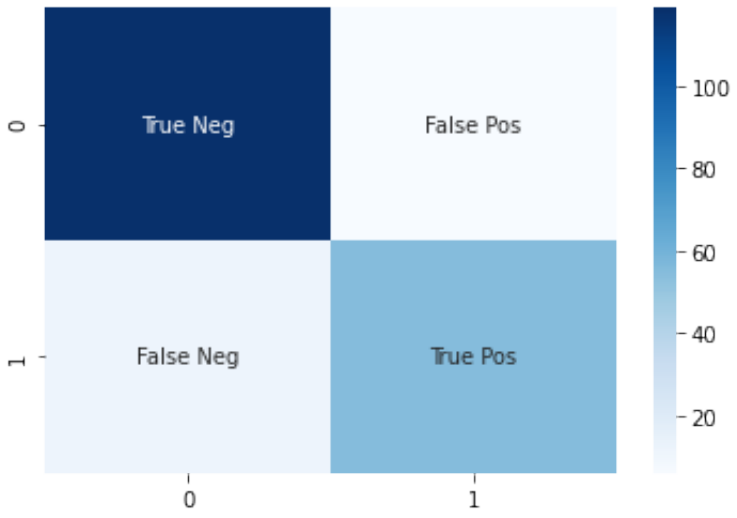

Figure 5.4: Confusion Matrix

Precision

Precision of a model is the part of relevant predictions (true positives) in all the predictions that were made for a certain class. It can be calculated by ***ModelPrecision = Truepositives / (TruePositives + FalsePositives***).
The precision for are model is 90% for both the classes.

Recall

Recall of a model is the part of predictions which were made to predict a class with respect to all the predictions that truly belong to that class. It can be calculated by ***ModelRecall = Truepositives / (TruePositives + FalseNegatives)***.
Recall for our model is 95% for outcome negative and 81% for outcome positive.

f1-Score

f1-Score is used to represent the balance between precision and recall. It can be calculated by ***Modelf1-Score = 2 *((ModelPrecision * ModelRecall) / (precission + recall))***.
f1-Score of our model is 0.93 for negative and 0.85 for positive class label.

Support

Support signifies the frequency of occurrences of each class label predicted by the model. For our model the support was 125 for negative and 67 for positive class label.

CHAPTER 6

RESULTS

This chapter contains the GUI sample that have been created for the project and it contains various screenshots that showcases the results of our project.

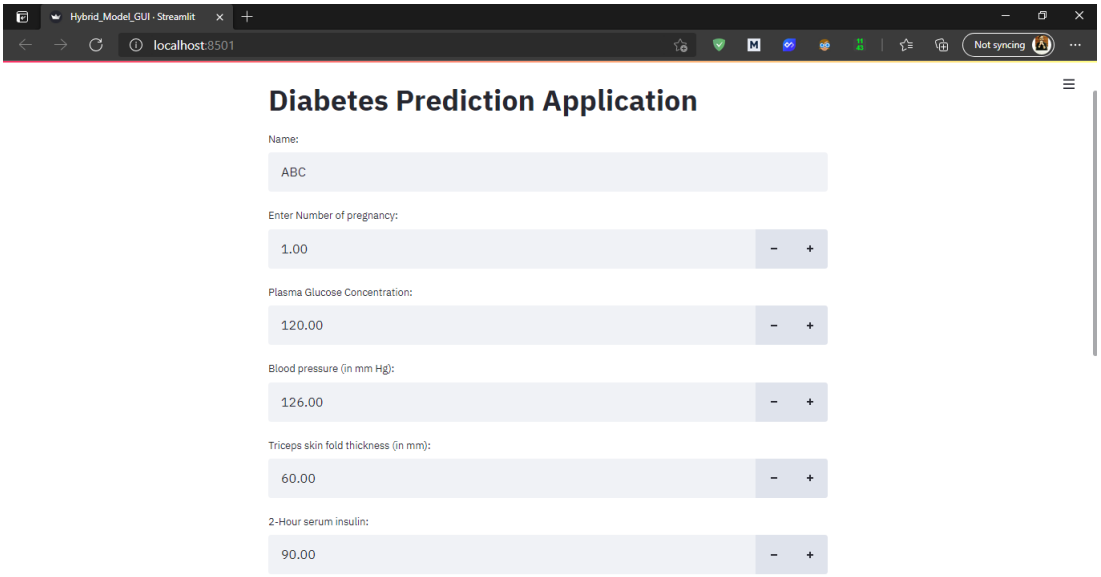

Figure 6.1: Result I (a)

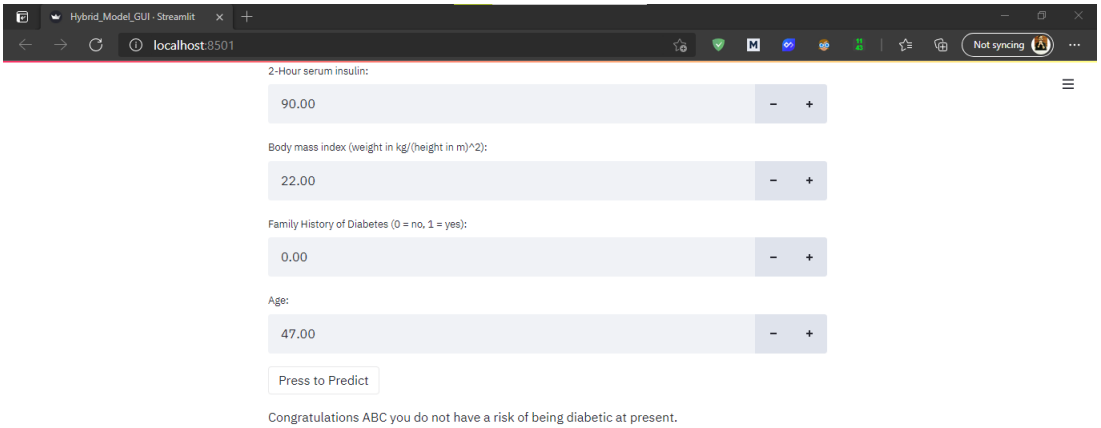

Figure 6.2: Result I (b)

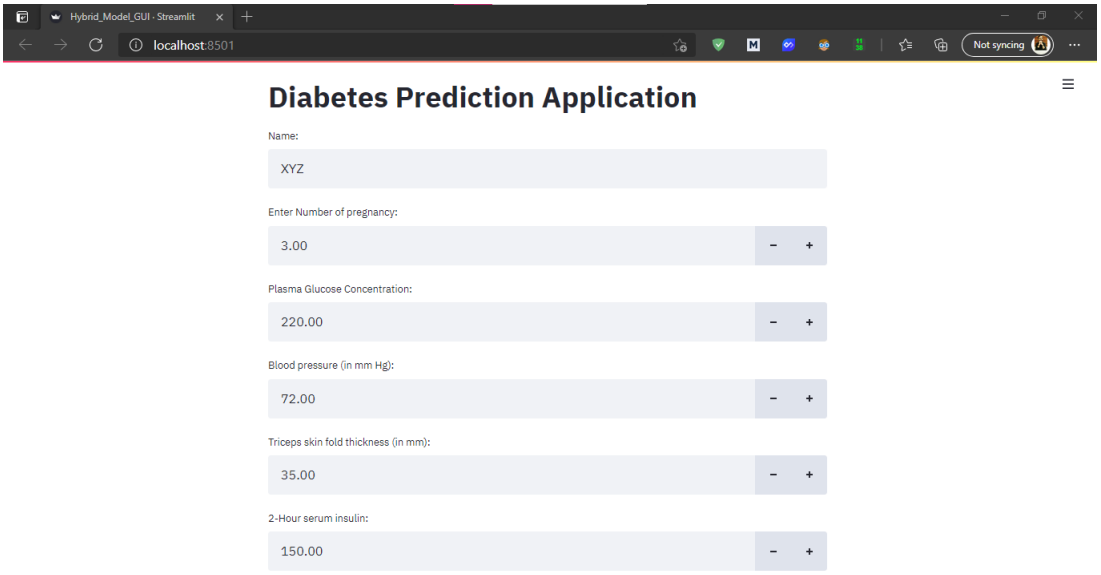

Figure 6.3: Result II (a)

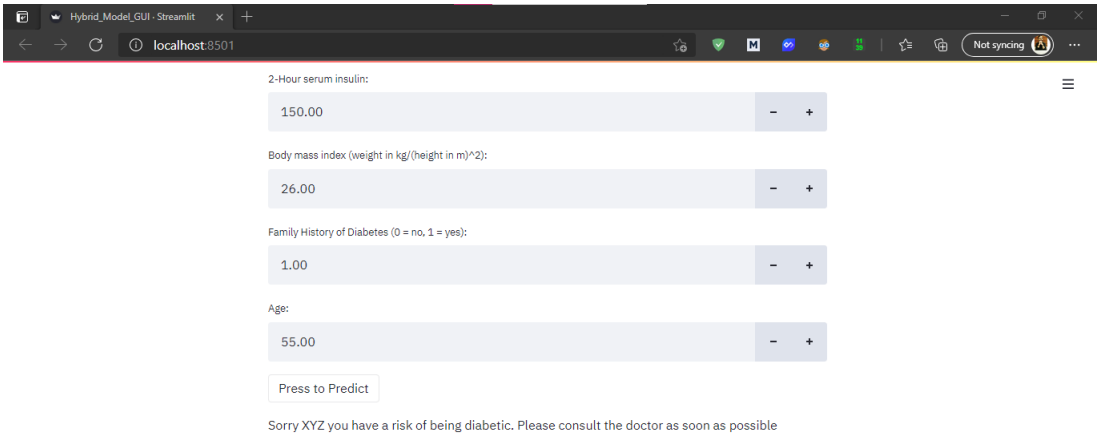

Figure 6.4: Result II (b)

CHAPTER 7

CONCLUSION

In order to analyze the propinquity betwixt the non-identical attributes present in the collected dataset, we plotted a heatmap as shown in figure 7.1. From the heatmap we can see that "glucose" and "insulin" are highly correlated with diabetes. We can also see that "BMI" and "skin thickness" have a high correlation with each other. There is a correlation between "age" and "pregnancies". "insulin" and "glucose" are also strongly correlated with each other.

In this project, we have implemented different ML and DL algorithms as mentioned earlier. Upon testing our trained model, accuracy of Perceptron came out to be 67% which is the lowest accuracy achieved. Naïve Bayes gave an accuracy of 77.2%. An accuracy of 83.7% was obtained by using Random Forest. MLP with scaled input for training was 83.9% accurate. 78.5% accuracy was obtained by using Logistic Regression algorithm, we used Logistic Regression in the meta layer (level 1) of our ensemble model. Top three accuracies were achieved 88.3% by kNN, 87.0% by SVM and 85.7% by Decision Tree, we used these three algorithms in the base layer (level 0). The output from these layers was fed to the meta layer to train our ensemble hybrid model. We achieved an accuracy of 90.6% with our ensemble model.

The above mentioned accuracy are shown in figure 7.2.

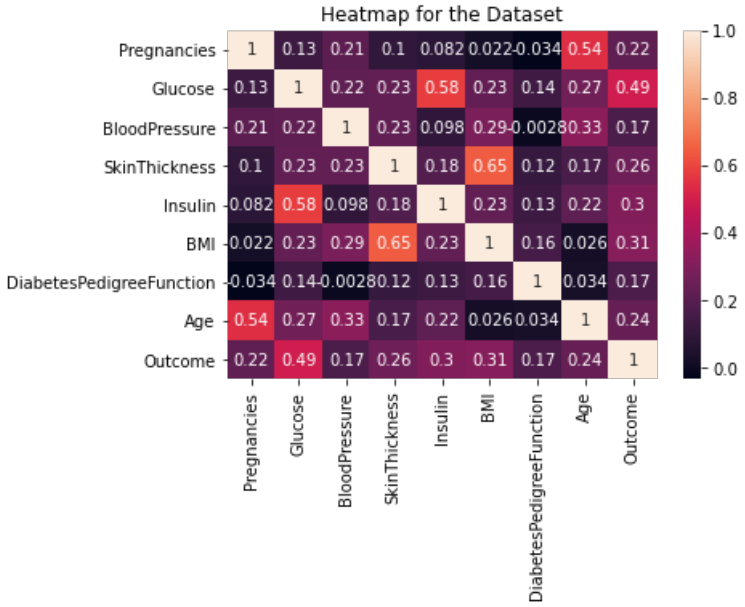

Figure 7.1: Heatmap

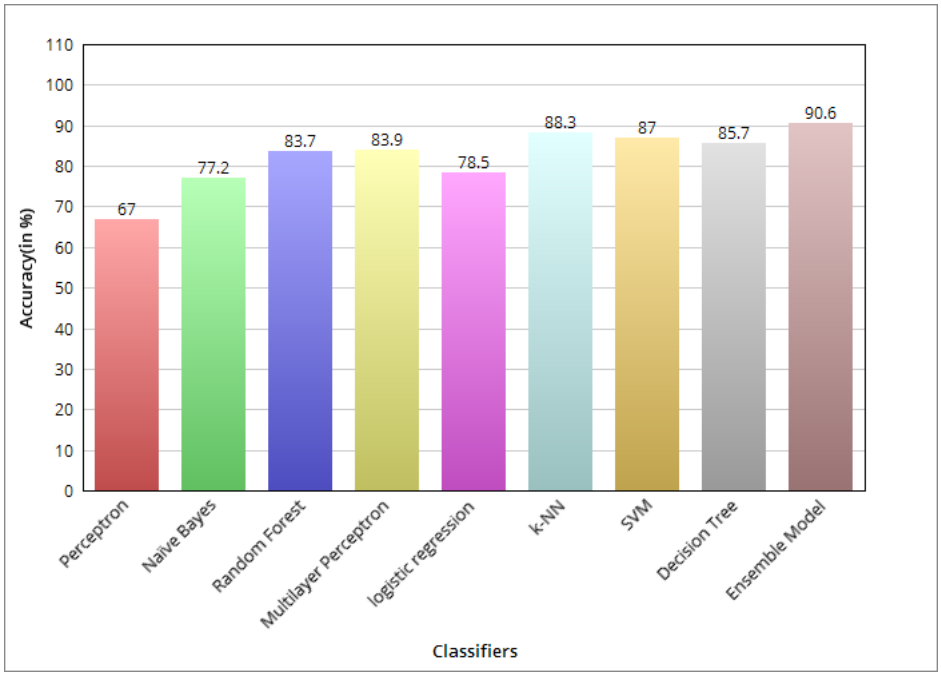

Figure 7.2: Accuracy Graph for Prediction Model

CHAPTER 8

FUTURE ENHANCEMENT

Only restriction faced during this project included access to the finite data in the dataset. The next step to improve the accuracy of model created for prediction is to apply the various techniques mentioned in this research paper using different or dataset larger than the dataset used. Also, we can use attributes like daily food habits and exercise habits of persons for training the model as these attributes can affect whether a person has a risk of being diabetic or not. If Deep Neural Networks (DNN) (Keras, Thaeno and Tensorflow) and unsupervised learning algorithms are implemented on a larger dataset, then the accuracy of the prediction model can be improved to a great extent. We can develop similar prediction models for various diseases like heart disease, cancer, brain tumors, asthma and so on.

REFERENCES

1. Balaji, H., Iyengar, N., and Caytiles, R. D. (2017). "Optimal predictive analytics of pima diabetics using deep learning." *International journal of database theory and application*, 10, 47–62.

2. Barik, S., Mohanty, S., Mohanty, S., and Singh, D. (2021). *Analysis of Prediction Accuracy of Diabetes Using Classifier and Hybrid Machine Learning Techniques*, 399–409.

3. Iyer, A., Jeyalatha, s., and Sumbaly, R. (2015). "Diagnosis of diabetes using classification mining techniques." *International Journal of Data Mining Knowledge Management Process*, 5, 1–14.

4. Kandhasamy, J. p. and Balamurali, S. (2015). "Performance analysis of classifier models to predict diabetes mellitus." *Procedia Computer Science*, 47, 45–51.

5. Lekha, S. and M., S. (2018). "Real-time non-invasive detection and classification of diabetes using modified convolution neural network." *IEEE Journal of Biomedical and Health Informatics*, 22(5), 1630–1636.

6. Mercaldo, F., Nardone, V., and Santone, A. (2017). "Diabetes mellitus affected patients classification and diagnosis through machine learning techniques." *Procedia Computer Science*, 112, 2519–2528.

7. Mohebbi, A., Aradóttir, T., Johansen, A., Bengtsson, H., Fraccaro, M., and Mørup, M. (2017). "A deep learning approach to adherence detection for type 2 diabetics." *Proceedings of 2017 39th Annual International Conference of the Ieee Engineering in Medicine and Biology Society*, 2017 39th Annual International Conference of the Ieee Engineering in Medicine and Biology Society (embc), United States. IEEE, 2896–9. 2017 39th Annual International Conference of the Ieee Engineering in Medicine and Biology Society, EMBC 2017 ; Conference date: 11-07-2017 Through 15-07-2017.

8. N., Y. and SriPreethaa, K. (2019). "Diabetes prediction in healthcare systems using machine learning algorithms on hadoop cluster." *Cluster Computing*, 22.

9. Nanda, S., Savvidou, M., Syngelaki, A., Akolekar, R., and Nicolaides, K. (2011). "Prediction of gestational diabetes mellitus by maternal factors and biomarkers at 11 to 13 weeks." *Prenatal Diagnosis*, 31(2), 135 – 141.

10. Zou, Q., Qu, K., Luo, Y., Yin, D., Ju, Y., and Tang, H. (2018). "Predicting diabetes mellitus with machine learning techniques." *Frontiers in Genetics*, 9, 515.