

EARLY STAGE DIABETES PREDICTION USING MACHINE LEARNING TECHNIQUES & DJANGO

*A Major Project report submitted in partial fulfilment of requirements in the VIII
semester for the award of degree of*

**Bachelor of Technology In
Computer Science and Engineering**

By

GOLLAPALLI BHAVYA MOULIKA

(Reg No: 17131A0562)

JAHNAVI SANJANA PATNALA

(Reg No: 17131A0577)

KESAVADAS SRI SAI THARUN

(Reg No: 17131A0599)

KURLI VINEETHA

(Reg No: 17131A05B7)

Under the esteemed guidance of

**Dr. G. Satya Keerthi
Assistant Professor**



**COLLEGE OF ENGINEERING
(AUTONOMOUS)**

GAYATRI VIDYA PARISHAD COLLEGE OF ENGINEERING (AUTONOMOUS)

Approved by AICTE, New Delhi and Affiliated to JNTU-Kakinada

Re-accredited by NAAC with “A” Grade with CGPA of 3.47/4.00

Madhurawada, Visakhapatnam-530 048

Gayatri Vidya Parishad College of Engineering (Autonomous)



COLLEGE OF ENGINEERING
(AUTONOMOUS)

CERTIFICATE

This is to certify that the project thesis entitled “**EARLY STAGE DIABETES PREDICTION USING MACHINE LEARNING TECHNIQUES & DJANGO**”, being submitted by: **GOLLAPALLI BHAVYA MOULIKA** (Reg No: 17131A0562), **JAHNAVI SANJANA PATNALA** (Reg No: 17131A0577), **KESAVADAS SRI SAI THARUN** (Reg No: 17131A0599), **KURLI VINEETHA** (Reg No: 17131A05B7)

in partial fulfilment for the award of the Degree of **Bachelor of Technology in Computer Science and Engineering** to the Jawaharlal Nehru Technological University Kakinada, Kakinada is a record of bonafide work carried out under my guidance and supervision. The results embodied in this project thesis have not been submitted to any other University or Institute for the award of any Degree or Diploma.

Guide

Dr. G. Satya Keerthi
Assistant Professor
Department of CSE
GVPCE(A)

Head of the Department

Dr. P. Krishna Subba Rao
Professor & HOD
Department of CSE
GVPCE(A)

DECLARATION

We hereby declare that this main project entitled “**EARLY STAGE DIABETES PREDICTION, USING MACHINE LEARNING TECHNIQUES & DJANGO**” is a bonafide work done by us, Batch-36 and submitted to **Department of Computer Science and Engineering, Gayatri Vidya Parishad College of Engineering (Autonomous), Visakhapatnam**, in partial fulfilment for the award of the degree of B. Tech is of our own and it is not submitted to any other university or has been published any time before.

GOLLAPALLI BHAVYA MOULIKA (17131A0562)

JAHNAVI SANJANA PATNALA (17131A0577)

KESAVADAS SRI SAI THARUN (17131A0599)

KURLI VINEETHA (17131A05B7)

PLACE: VISAKHAPATNAM

DATE: 03-07-2021

ACKNOWLEDGEMENT

We would like to take this opportunity to extend our hearty gratitude to our esteemed institute "**Gayatri Vidya Parishad College of Engineering (Autonomous)**" where we got the platform to fulfill our cherished desire.

We express our sincere thanks to **Dr. A.B. KOTESWARA RAO**, principal, Gayatri Vidya Parishad College of Engineering (Autonomous), for his support and encouragement during the course of this project.

We pay our deep sense of gratitude to **Dr. P. KRISHNA SUBBA RAO**, Head of the Department of Computer Science and Engineering, Gayatri Vidya Parishad College of Engineering (Autonomous) for his constant support and encouragement.

We are obliged to **Dr. G. SATYA KEERTHI**, Assistant Professor, Department of Computer Science and Engineering, who has been our guide, whose valuable suggestions, guidance and comprehensive assistance helped us a lot in realizing the project.

We also thank **Dr. SITHA KUMARI**, Associate Professor, Project Coordinator, Department of Computer Science and Engineering, for guiding us throughout the project and helping us in completing the project efficiently.

Lastly, we are grateful to all our friends, for their relentless support in augmenting the value of work, our family, for being considerate and appreciative throughout.

ABSTRACT

Diabetes is a condition that impairs the body's ability to process blood glucose and is one of the most lethal diseases in the world. Patients need to visit a diagnostic center, to get the reports after consultation, which needs heavy investment of time and accuracy. But now, due to the growth of machine learning methods, we have got the flexibility to search out and answer the current issue. Age, obesity, lack of exercise, hereditary diabetes, living style, bad diet, high blood pressure, etc. can cause Diabetes. People having diabetes have a high risk of diseases like heart disease, kidney disease, stroke, eye problem, nerve damage, etc. The aim of this project is to develop a system, which might predict the diabetic risk level of a patient. We will be performing exploratory data analysis on the data collected and then using various machine learning algorithms to train the model and furthermore, with the help of Django framework, we will be creating a webpage that takes input from the user and then predicts whether a person is diabetic or not. The dataset which we have used in this project contains the signs and symptoms data of newly diabetic or would-be diabetic patients. This has been collected using direct questionnaires from the patients of Sylhet Diabetes Hospital in Sylhet, Bangladesh and approved by doctors.

CONTENTS

CERTIFICATE OF ORIGINALITY	2
ACKNOWLEDGEMENT	4
ABSTRACT	5
1. INTRODUCTION	8
1.1 OBJECTIVE	09
1.2 ABOUT THE PROJECT	09
1.3 AN OVERVIEW OF TECHNOLOGY, FRAMEWORK USED	10
1.3.1 DATA SCIENCE	10
1.3.2 MACHINE LEARNING	10
1.3.2.1 KNN CLASSIFIER	11
1.3.2.2 SVM CLASSIFIER	12
1.3.2.3 LOGISTIC REGRESSION	13
1.3.2.4 RANDOM FOREST	14
1.3.3 DJANGO	15
1.4 SOFTWARE DESCRIPTION	16
1.4.1 PYTHON	16
1.4.2 BENEFITS OF PYTHON	16
2. LITERATURE SURVEY	17
2.1 EXISTING SYSTEM	17
2.1.1 DIAGNOSIS BY TRADITIONAL WAY	17
2.1.2 PREDICTION USING DATA MINING	17
3. DRAWBACKS OF EXISTING SYSTEM	18
4. PROPOSED SYSTEM	19
4.1 K NEIGHBORS CLASSIFIER	19
5. SOFTWARE REQUIREMENT ANALYSIS	21
5.1 PROBLEM STATEMENT	21
5.2 SOFTWARE REQUIREMENT ANALYSIS	21
5.2.1 FUNCTIONAL REQUIREMENTS	21
5.2.2 NON - FUNCTIONAL REQUIREMENTS	23

5.3 SYSTEM REQUIREMENTS	24
5.3.1 SOFTWARE REQUIREMENTS	24
5.3.2 HARDWARE REQUIREMENTS	24
6. SYSTEM DESIGN	25
6.1 ARCHITECTURE	25
6.2 PROCESS FLOW DIAGRAM	26
6.3 CLASS DIAGRAM	27
6.4 SEQUENCE DIAGRAM	28
6.5 CONTROL FLOW DIAGRAM	29
7. IMPLEMENTATION OF WORK	30
7.1 OUTLINE OF FILES USED IN DJANGO	30
7.2 SAMPLE CODE TEMPLATE OF DJANGO	30
7.3 OUTLINE OF PACKAGES IN MACHINE LEARNING	39
7.4 SAMPLE CODE TEMPLATE OF MACHINE LEARNING	40
8. RESULTS	45
9. SAMPLE TEST CASES	50
10. CONCLUSION	54
11. FUTURE ENHANCEMENTS	55
12. BIBLIOGRAPHY	56

1. INTRODUCTION

Diabetes is a condition when our body isn't able to take up Sugar (Glucose) into its cells and use it for energy, which results in buildup of extra sugar. Considering the current scenario, in developing countries like India, Diabetic Mellitus (DM) has become a very severe disease. Predictions have been made that by 2045 it may boost upto one in every three people. This is a serious issue we need to deal with. The chronic disease of diabetes results in action when there is a vast increase in the blood glucose concentration. This is a major cause for other problems and diseases such as kidney diseases and heart related problems. Many unhealthy eating habits and lack of proper body exercises also cause diabetic pre behavior. It has been stated by the WHO that the total count of people suffering from diabetes has vastly increased over the past three years. Diabetic Mellitus (DM) is classified as a Non-Communicable Disease (NCD) and many people are suffering from it.

There are 2 types of Diabetes:

Type 1: Upto 10% of people who have Diabetes, have Type-1. It is usually diagnosed in children and young adults. People with type-1 diabetes need to take insulin every day.

Type 2: It is the most common and 90% of people who have diabetes, have type-2. It occurs usually in middle aged to older people. In this type, the body doesn't make enough insulin and is called insulin-resistant diabetes.

Early prediction of diseases like diabetes can be controlled and save human life. To accomplish this, this work explores prediction of diabetes by taking various attributes related to diabetes disease. For this purpose we use the Pima Indian Diabetes Dataset, we apply various Machine Learning classification Techniques to predict diabetes.

1.1 OBJECTIVE

In this project, we will be predicting whether a person is a type-2 diabetic or not, as it is the most common and a serious health concern. Machine Learning is a method that is used to train computers or machines explicitly. Various Machine Learning Techniques provide efficient results to collect Knowledge by building various classification models from collected dataset. Such collected data can be useful to predict diabetes. Various techniques of Machine Learning are capable of prediction.

1.2 ABOUT THE PROJECT

In this project, to predict Diabetes at an early stage, we have trained various machine learning algorithms like KNN, Logistic regression, SVM, Naïve bayes --classifier, Random forest. After building the model with one of these classifiers, we have designed an UI to take inputs from the user and make a prediction of his/her diabetic condition.

The following features have been provided to help us predict whether a person is diabetic or not:

- **Pregnancies:** Number of times pregnant
- **Glucose:** Plasma glucose concentration over 2 hours in an oral glucose tolerance test
- **BloodPressure:** Diastolic blood pressure (mm Hg)
- **SkinThickness:** Triceps skin fold thickness (mm)
- **Insulin:** 2-Hour serum insulin (mu U/ml)
- **BMI:** Body mass index (weight in kg/(height in m)²)
- **DiabetesPedigreeFunction:** Diabetes pedigree function (a function which scores likelihood of diabetes based on family history)
- **Age:** Age (years)
- **Outcome:** Class variable (0 if non-diabetic, 1 if diabetic)

1.3 AN OVERVIEW OF TECHNOLOGY, FRAMEWORK USED

Artificial intelligence (AI) refers to the simulation of human intelligence in machines that are programmed to think like humans and mimic their actions. AI systems work by ingesting large amounts of labeled training data, analyzing the data for correlations and patterns, and using these patterns to make predictions about future states. A subset of artificial intelligence is machine learning, which learns from and adapts to new data without being assisted by humans.

1.3.1. DATA SCIENCE

Data Science is a blend of various tools, algorithms, and machine learning principles with the goal to discover hidden patterns from the raw data. Data Science is primarily used to make decisions and predictions making use of predictive causal analytics, prescriptive analytics (predictive plus decision science) and machine learning. It involves a plethora of disciplines and expertise areas to produce a holistic, thorough and refined look into raw data and also refers to the process of extracting clean information to formulate actionable insights.

1.3.2 MACHINE LEARNING

Machine learning is an application of artificial intelligence (AI) that provides systems the ability to automatically learn and improve from experience without being explicitly programmed. Machine learning focuses on the development of computer programs that can access data and use it to learn for themselves.

The practice of machine learning involves taking data, examining patterns and developing some sort of prediction about future outcomes. By feeding an algorithm more data over time, we can sharpen the machine learning model's predictions.

The various ML models which we have used for training the model are as follows:

1.3.2.1 K-NEAREST NEIGHBORS CLASSIFIER

K-nearest neighbors (KNN) algorithm is a type of supervised ML algorithm which can be used for both classification as well as regression predictive problems. However, it is mainly used for classification of predictive problems in industry. The KNN algorithm assumes that similar things exist in close proximity. In other words, similar things are near to each other.

Predictions are made for a new instance (x) by searching through the entire training set for the K most similar instances (the neighbors) and summarizing the output variable for those K instances.

To determine which of the K instances in the training dataset are most similar to a new input a distance measure is used. For real-valued input variables, the most popular distance measure is Euclidean distance. Euclidean distance is calculated as the square root of the sum of the squared differences between a new point (x) and an existing point (x_i) across all input attributes j .

The following two properties would define KNN well –

- **Lazy learning algorithm** – KNN is a lazy learning algorithm because it does not have a specialized training phase and uses all the data for training while classification. All training data used in the testing phase, which makes the training faster.
- **Non-parametric learning algorithm** – KNN is also a non-parametric learning algorithm because it doesn't assume anything about the underlying data.

1.3.2.2 SUPPORT VECTOR MACHINE

The objective of the support vector machine algorithm is to find a hyperplane in an N-dimensional space (N — the number of features) that distinctly classifies the data points. In the SVM algorithm, we plot each data item as a point in n-dimensional space (where n is the number of features you have) with the value of each feature being the value of a particular coordinate.

The goal of the SVM algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data points in the correct category in the future. This best decision boundary is called a **hyperplane**.

SVM chooses the extreme points/vectors that help in creating the hyperplane. These extreme cases are called **support vectors**, and hence the algorithm is termed as Support Vector Machine.

SVM can be of two types:

- **Linear SVM:** Linear SVM is used for linearly separable data, which means if a dataset can be classified into two classes by using a single straight line, then such data is termed as linearly separable data, and classifier is used called as Linear SVM classifier.
- **Non-linear SVM:** Non-Linear SVM is used for non-linearly separated data, which means if a dataset cannot be classified by using a straight line, then such data is termed as non-linear data and classifier used is called as Non-linear SVM classifier.

1.3.2.3 LOGISTIC REGRESSION CLASSIFIER

Logistic regression is one of the most popular Machine Learning algorithms, which comes under the Supervised Learning technique. It is used for predicting the categorical dependent variable using a given set of independent variables. Logistic regression uses the concept of predictive modeling as regression; therefore, it is called logistic regression, but is used to classify samples:

Therefore, it falls under the classification algorithm.

- Logistic regression predicts the output of a categorical dependent variable. Therefore the outcome must be a categorical or discrete value. It can be either Yes or No, 0 or 1, True or False, etc. but instead of giving the exact value as 0 and 1, **it gives the probabilistic values which lie between 0 and 1.**
- It is much similar to Linear Regression except that how they are used. Linear Regression is used for solving Regression problems, whereas **Logistic regression is used for solving the classification problems.**
- In Logistic regression, instead of fitting a regression line, we fit an "S" shaped logistic function, which predicts two maximum values (0 or 1).
- The curve from the logistic function indicates the likelihood of something such as whether the cells are cancerous or not, a mouse is obese or not based on its weight, etc.
- Logistic Regression is a significant machine learning algorithm because it has the ability to provide probabilities and classify new data using continuous and discrete datasets.
- Logistic Regression can be used to classify the observations using different types of data and can easily determine the most effective variables used for the classification.

1.3.2.4 RANDOM FOREST CLASSIFIER

Random Forest is a popular machine learning algorithm that belongs to the supervised learning technique. It can be used for both Classification and Regression problems in ML. It is based on the concept of **ensemble learning**, which is a process of combining multiple classifiers to solve a complex problem and to improve the performance of the model. Random Forest works in two-phase first is to create the random forest by combining N decision trees, and second is to make predictions for each tree created in the first phase.

As the name suggests, *"Random Forest is a classifier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset."* Instead of relying on one decision tree, the random forest takes the prediction from each tree and based on the majority votes of predictions, and it predicts the final output. The greater number of trees in the forest leads to higher accuracy and prevents the problem of overfitting.

Random forests (RF) are basically a bag containing 'n' Decision Trees (DT) having a different set of hyper-parameters and trained on different subsets of data. Since the random forest combines multiple trees to predict the class of the dataset, it is possible that some decision trees may predict the correct output, while others may not. But together, all the trees predict the correct output. Therefore, below are two assumptions for a better Random forest classifier:

- There should be some actual values in the feature variable of the dataset so that the classifier can predict accurate results rather than a guessed result.
- The predictions from each tree must have very low correlations.

1.3.3 DJANGO

Django is a high-level python web- framework that encourages rapid development and clean, pragmatic design. Built by experienced developers, it takes care of much of the hassle of Web development, so you can focus on writing your app without needing to reinvent the wheel. It's free and open source.

- **Ridiculously fast**

Django was designed to help developers take applications from concept to completion as quickly as possible.

- **Fully loaded**

Django includes dozens of extras you can use to handle common Web development tasks. Django takes care of user authentication, content administration, site maps, RSS feeds, and many more tasks — right out of the box.

- **Reassuringly secure**

Django takes security seriously and helps developers avoid many common security mistakes, such as SQL injection, cross-site scripting, cross-site request forgery and clickjacking. Its user authentication system provides a secure way to manage user accounts and passwords.

- **Exceedingly scalable**

Some of the busiest sites on the planet use Django's ability to quickly and flexibly scale to meet the heaviest traffic demands.

- **Incredibly versatile**

Companies, organizations and governments have used Django to build all sorts of things — from content management systems to social networks to scientific computing platforms.

1.4 SOFTWARE DESCRIPTION

1.4.1 PYTHON

Python is a multi-paradigm programming language. Object-oriented programming and structured programming are fully supported, and many of its features support functional programming and aspect-oriented programming (including by metaprogramming and metaobjects. Many other paradigms are supported via extensions, including design by contract and logic programming. Python uses dynamic typing and a combination of reference counting and a cycle-detecting garbage collector for memory management. It also features dynamic name resolution (late binding), which binds method and variable names during program execution.

Python's developers strive to avoid premature optimization, and reject patches to non-critical parts of CPython that would offer marginal increases in speed at the cost of clarity. When speed is important, a Python programmer can move time-critical functions to extension modules written in languages such as C, or use PyPy, a just-in-time compiler. Cython is also available, which translates a Python script into C and makes direct C-level API calls into the Python interpreter. An important goal of Python's developers is keeping it fun to use. Python's design offers some support for functional programming in the Lisp tradition. It has filter, map, and reduce functions, list comprehensions, dictionaries, sets, and generator expressions. The standard library has two modules (itertools and functools) that implement functional tools borrowed from Haskell and Standard ML.

1.4.2 BENEFITS OF PYTHON

- Presence of Third-Party Modules
- Extensive Support Libraries
- Open Source and Community Development
- Learning Ease and Support Available

2. LITERATURE SURVEY

2.1 EXISTING SYSTEM

The analysis of related work gives results on various healthcare datasets, where analysis and predictions were carried out using various methods and techniques. Various prediction models have been developed and implemented by various researchers using variants of data mining techniques. Various traditional methods, based on physical and chemical tests, are available for diagnosing diabetes.

2.1.1 DIAGNOSIS OF DIABETES (TRADITIONAL WAY)

Current practice in hospitals is to collect required information for diabetes diagnosis through various tests, and appropriate treatment is provided based on diagnosis.

The various tests include the following;

Glycated hemoglobin (A1C) test: This blood test, which doesn't require fasting, indicates your average blood sugar level for the past two to three months. It measures the percentage of blood sugar attached to hemoglobin, the oxygen-carrying protein in red blood cells.

Oral glucose tolerance test. For this test, you fast overnight, and the fasting blood sugar level is measured. Then you drink a sugary liquid, and blood sugar levels are tested periodically for the next two hours.

2.1.2 PREDICTION USING DATA MINING

The existing instances of datasets consist of only two groups i.e., blood tests and urine tests. Implementation is done by the WEKA tool.

3. DRAWBACKS OF EXISTING SYSTEM

One can imagine how much waste of time can occur if we keep on physically visiting the hospitals every day for one same purpose all along the days and is a time taking process, as it needs a heavy investment of time, to get the reports. Also, Early stage prediction cannot be possible, as it is a challenging task for medical practitioners.

In the existing method, the classification and prediction accuracy are not so high, by Classifying the data using the WEKA tool, as it only has fewer parameters and it can only be performed only on small datasets. If you have a large data set you will most likely run in to out of memory exception, as the processing is limited by memory.

4. PROPOSED SYSTEM

Machine learning is considered to be a dire need of today's situation in order to eliminate human efforts by supporting automation with minimum flaws. Not only in the current and ongoing generations require remote healthcare but the oncoming and almost all of the future generations are totally going to be dependent on the digital smart technology to take necessary steps regarding health. The involvement of machine technical learning algorithms and smart medical sensors in the system produces a huge impact over the world.

In this project, we will be proposing a predictive model, for diabetes detection, using machine learning algorithms and for better classification of diabetes we included few external factors.

Predictive Analysis (Supervised learning) incorporates a variety of machine learning algorithms, data mining techniques and statistical methods that uses current and past data to find knowledge and predict future events.

Thus, with the help of available medical sensors, we can test the various attributes which are used in the project, like, Insulin, Blood Pressure, Glucose levels, Pregnancies, BMI, Skin thickness, Age, Diabetes pedigree function etc., and by using various Machine learning techniques, we can build the predictive model, for predicting the diabetic condition of a person.

4.1 K NEIGHBORS CLASSIFIER

K-nearest neighbors (KNN) algorithm uses 'feature similarity' to predict the values of new datapoints which further means that the new data point will be assigned a value based on how closely it matches the points in the training set. We can understand its working with the help of following steps –

- Step 1 – For implementing any algorithm, we need a dataset. So during the first step of KNN, we must load the training as well as test data.

- Step 2 – Next, we need to choose the value of K i.e. the nearest data points. K can be any integer.
- Step 3 – For each point in the test data we –
 - a) Calculate the distance between test data and each row of training data with the help of any of the methods namely: Euclidean, Manhattan or Hamming distance. The most commonly used method to calculate distance is Euclidean.
 - b) Now, based on the distance value, sort them in ascending order.
 - c) Next, it will choose the top K rows from the sorted array.
 - d) Now, it will assign a class to the test point based on the most frequent class of these rows.
- Step 4 – End

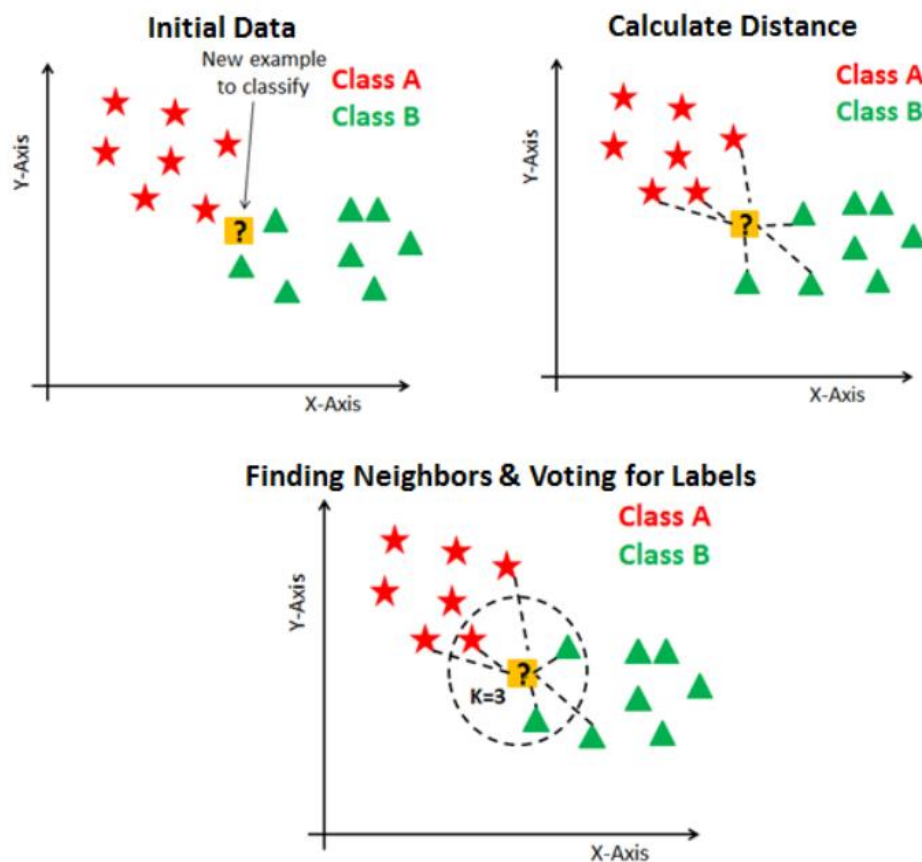


Fig 4.1: KNN Classifier

5. SOFTWARE REQUIREMENT ANALYSIS

5.1 PROBLEM STATEMENT

Scientists have been making the prediction that, by 2045, diabetes may boost up to one in every three people. This is a serious issue we need to deal with. The chronic disease of diabetes, results in action when there is a vast increase in the blood glucose concentration. Hence, an Early stage prediction would be crucial in order to help better treatment of a patient with a diabetic state. The objective is to build a predictive system with some medical field attributes that will be useful for patients and diabetes doctors in identifying diabetes. The proposed system is an automation for the detection of diabetes by using the old diabetes patient's data.

5.2 SRS DOCUMENT

A Software Requirements Specification (SRS) is a complete description of the include of the system to be developed. It includes a set of use cases that describe all the interactions the users will have with the software. Use cases are also known as functional requirements. In addition to use cases, the SRS also contains non-functional requirements. Non-functional requirements are requirements which impose constraints on the design or implementation (such as performance engineering requirements, quality standards, or design constraints).

5.2.1 FUNCTIONAL REQUIREMENTS

A functional requirement defines a function of a system or its component. A function is described as a set of inputs, the behavior, and outputs. How the system implements functional requirements is detailed in the system design. In some cases, a requirement requirements analyst generates use cases after gathering and validating a set of functional requirements. Each use case illustrates behavioral scenarios through one or more functional requirements.

1. NUMPY

Numpy is a Python package. It stands for 'Numerical Python'. It is a library consisting of multidimensional array objects and a collection of routines for processing of arrays. Numeric, the ancestor of NumPy, was developed by Jim Hugunin. Another package Num array was also developed, having some additional functionalities. In 2005, Travis Oliphant created the NumPy package by incorporating the features of Numarray into Numeric package. There are many contributors to this open source project.

Operations using NumPy:

1. Mathematical and logical operations on arrays.
2. Fourier transforms and routines for shape manipulation.
3. Operations related to linear algebra. NumPy has in-built functions for linear algebra and random number generation.

2. PANDAS

Pandas has been one of the most popular and favorite data science tools used in Python programming language for data analysis. Data is unavoidably messy in real world. And Pandas is seriously a game changer when it comes to cleaning, transforming, manipulating and analyzing data.

3. SKLEARN

Scikit-learn is one the most popular ML libraries. It supports many supervised and unsupervised learning algorithms. It adds a set of algorithms for common machine learning and data mining tasks, including clustering, regression and classification.

4. SEABORN

Seaborn is a library for making statistical graphics in Python. It builds on top of matplotlib and integrates closely with pandas data structures. Seaborn helps

you explore and understand your data.

5. MATPLOTLIB

Matplotlib is one of the most popular Python packages used for data visualization. It is a cross-platform library for making 2D plots from data in arrays. It provides an object-oriented API that helps in embedding plots in applications using Python GUI toolkits such as PyQt, WxPython Tkinter. It can be used in Python shells, Google Colab notebook and web application servers also.

5.2.2 NON-FUNCTIONAL REQUIREMENTS

In systems engineering and requirements engineering, a non-functional requirement is a requirement that specifies the criteria that can be used to judge the operation of a system, rather than specific behaviors. Non-functional requirements are often called qualities of a system or constraints or non-behavioral requirements.

- **Performance**

The performance is measured in terms of the output generated by the model. Accuracy test results are the outputs generated by the model.

- **Cost**

The cost of the system is affordable because of usage of open source software.

- **Flexibility**

This model can work on any system that has a python IDE with dataset, and all the required libraries.

- **Robustness**

Robustness is the ability of the system to cope with error during execution and cope with erroneous inputs. The system should be able to handle errors during the execution process.

- **Platform compatibility**

This Project is compatible with Desktops and Laptops.

5.3 SYSTEM REQUIREMENTS

To be used efficiently, all computer software needs certain hardware components or other software resources to be present on a computer. These prerequisites are known as (computer) system requirements and are often used as guidelines as opposed to an absolute rule. Most software defines two sets of system requirements: minimum and recommended. With increasing demand for higher processing power and resources in newer versions of software, system requirements tend to increase over time. Industry analysts suggest that this trend plays a bigger part in driving upgrades in existing computer systems than technological advancements.

5.3.1 SOFTWARE REQUIREMENTS

Operating system	: Windows
Programming Language	: Python
IDE	: Google Colab
Packages	: Numpy, Pandas, Matplotlib, SkLearn

5.3.2 HARDWARE REQUIREMENTS

Processor	: Intel/ARM processor
RAM	: 8GB
Disc space	: 100 GB

6. SYSTEM DESIGN

The System design consists of various designs, which we have implemented in our Diabetes prediction system, using machine learning and Django. This system has been built with various designs such as architecture diagram, sequence diagram, class diagram, process flow and control-flow diagrams.

6.1 ARCHITECTURE DIAGRAM

An architectural diagram is a diagram of a system that is used to abstract the overall outline of the software system and the relationships, constraints, and boundaries between components. It is an important tool as it provides an overall view of the physical deployment of the software system and its evolution roadmap.

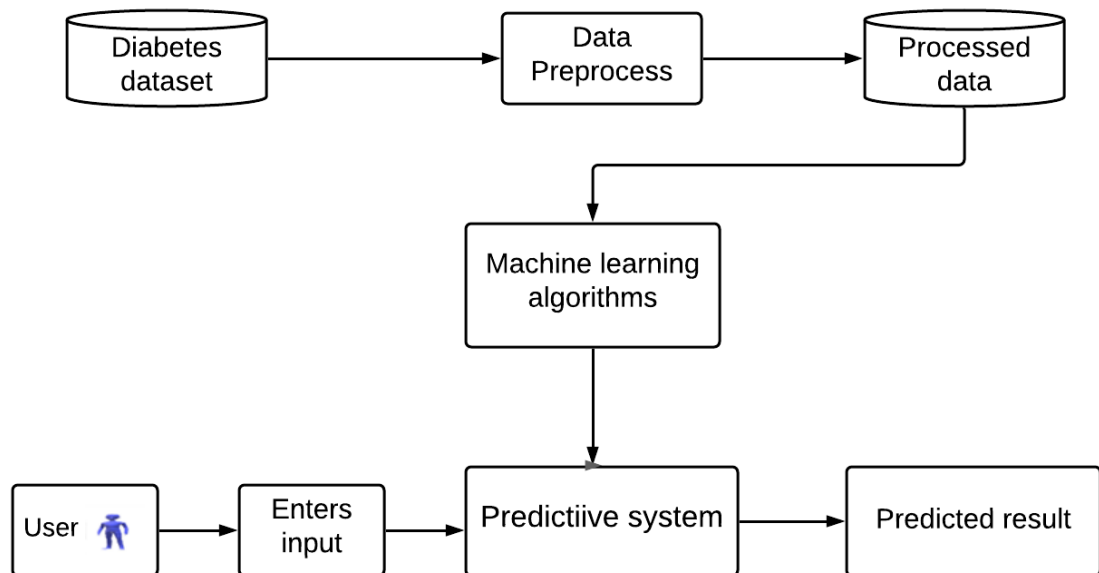


Fig 6.1: System Architecture

6.2 PROCESS-FLOW DIAGRAM

Process flows (or process diagrams) provide a visual overview or workflow diagram of all the tasks and relationships involved in a process and explains how a system works. The process flow diagram can greatly improve any process improvement project by increasing understanding the flow of information, people, and resources. The more data the map incorporates into the design the more beneficial the map will be to your efforts.

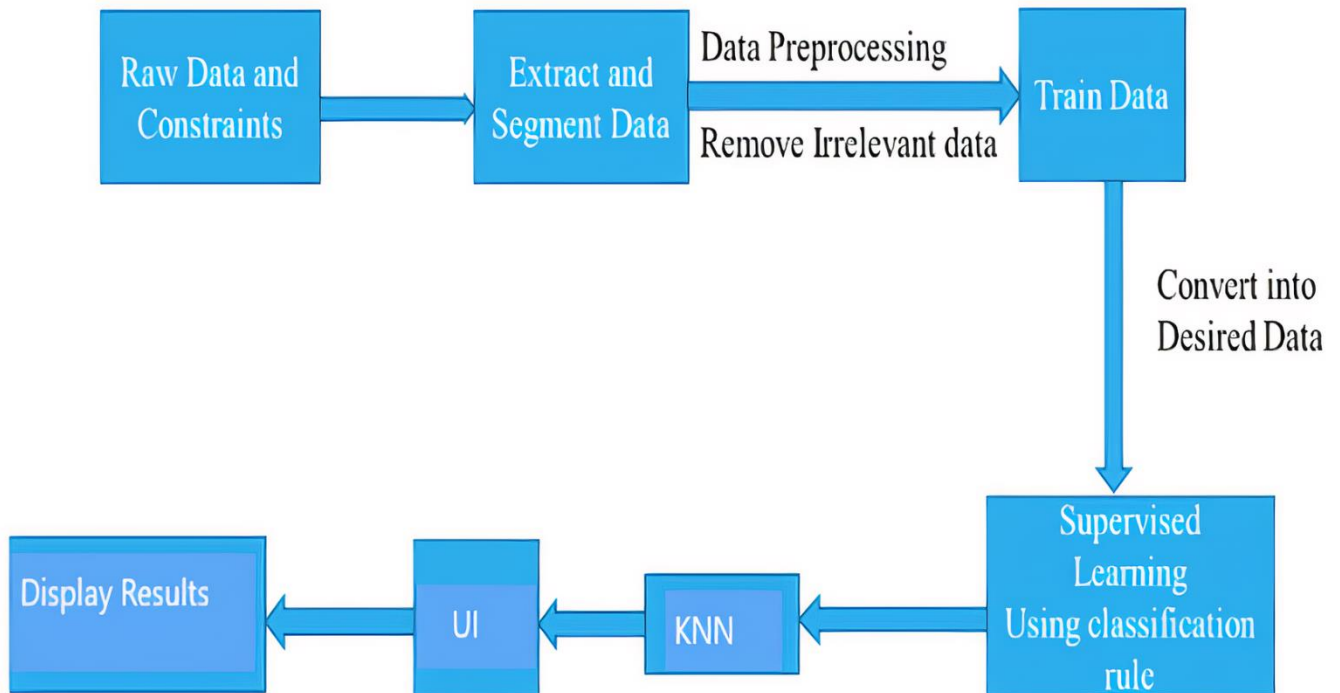


Fig 6.2: Process Flow Diagram

6.3 CLASS DIAGRAM

A class diagram is the basic building block of object-oriented modelling. A Class diagram consist information about all the classes that is used and all the related datasets, and all the other necessary attributes and their relationships with other entities, all these information is necessary inorder to use the concept of the prediction, where the user inputs the values. The figure below depicts the class diagram of our model.

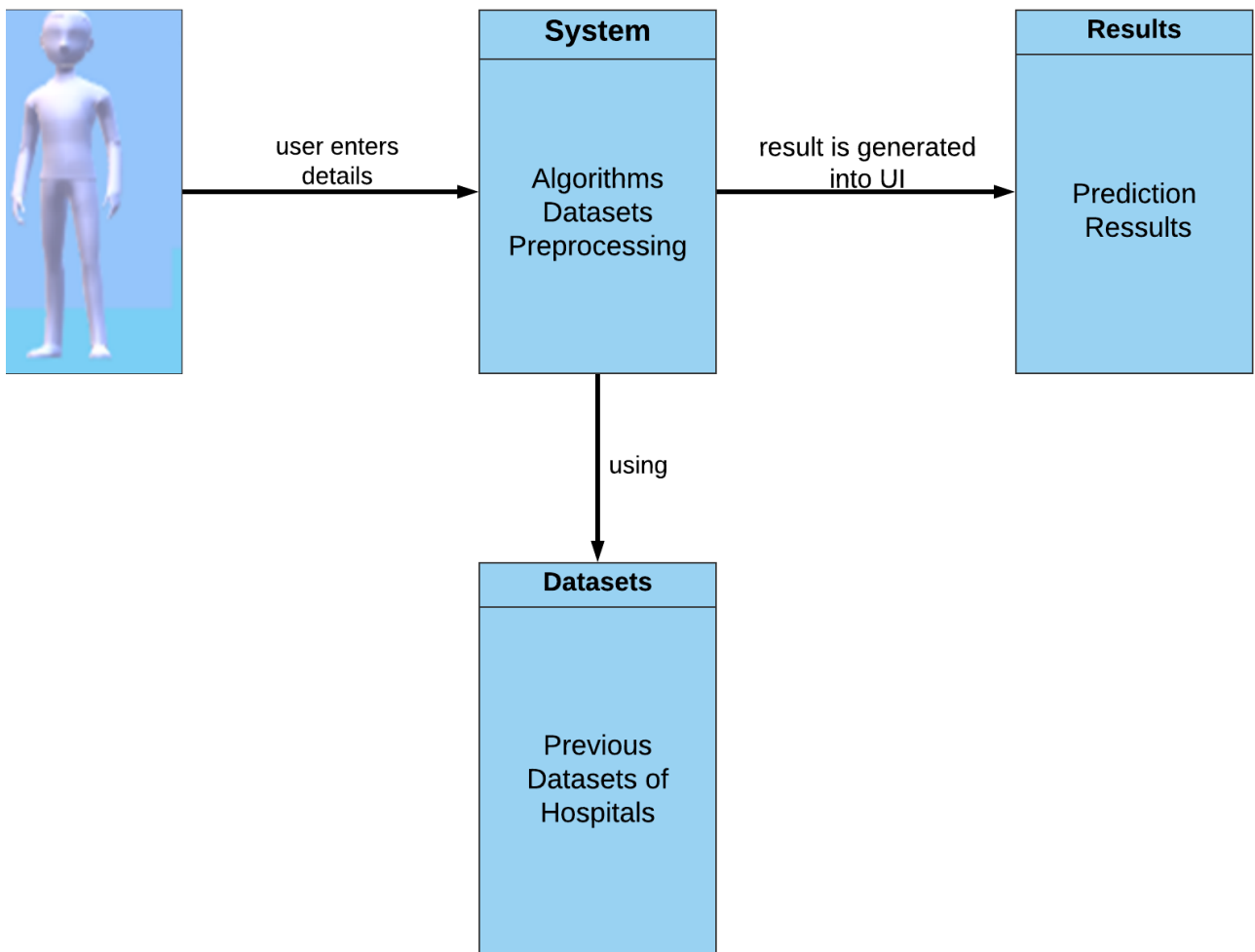


Fig 6.3: Class Diagram

6.4 SEQUENCE DIAGRAM

A sequence diagram shows object interactions arranged in time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario. The horizontal axis shows the elements involved in the interaction and the vertical axis shows the progress of time or the ordering of messages. The figure below shows the sequence diagram of our model.

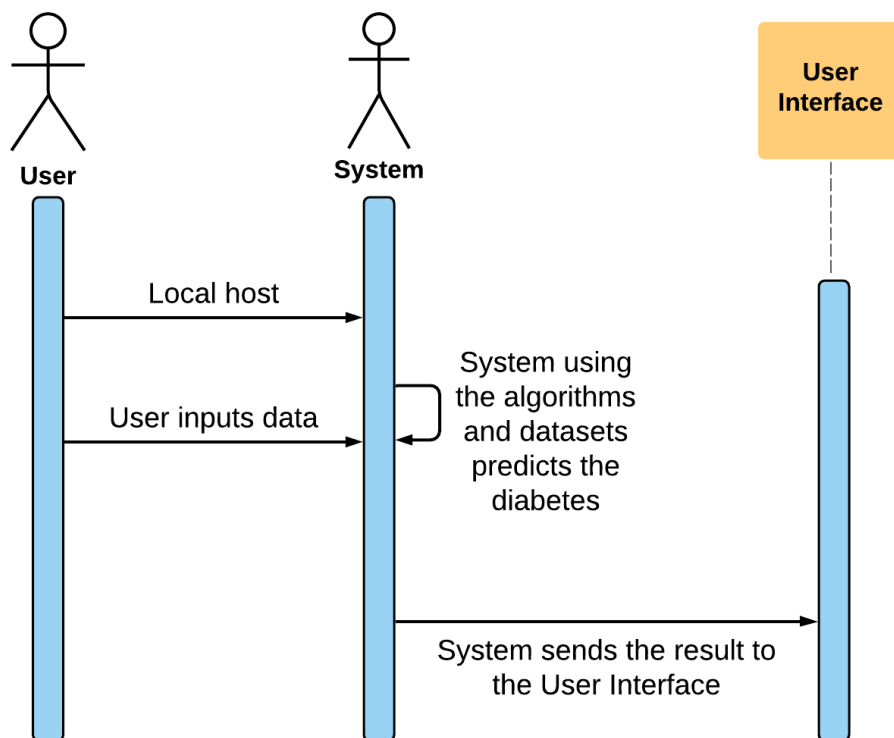


Fig 6.4: Sequence Diagram

6.5 CONTROL FLOW DIAGRAM

A control flow diagram helps us understand the details of a process. It shows us where the control starts and ends and where it may branch off in another direction, given certain situations. A control flow diagram for our model is shown in the figure below.

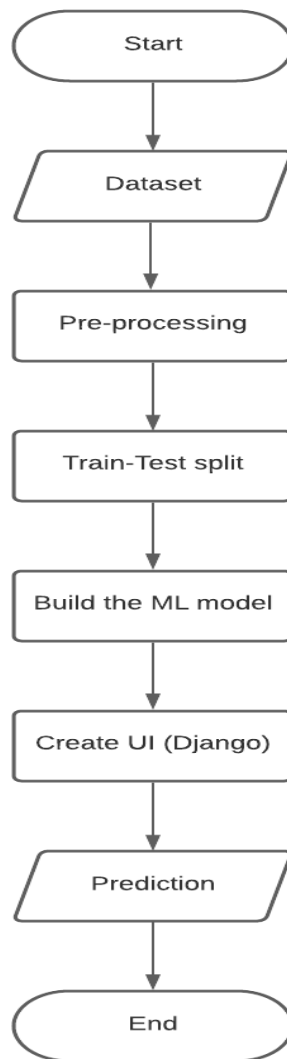


Fig 6.5: Control Flow Diagram

7. IMPLEMENTATION OF WORK

7.1 A BRIEF OUTLINE OF THE FILES USED (DJANGO)

- The file 'Home.html' is a HTML file that contains the Front-end, for our user interface. It acts as a homepage for our system.
- The file 'Predict.html' is also a HTML file, that contains the Front-end and displays the attributes, from where the user has to input the values, in order for the system to make predictions.
- The file 'URLs.py' is a Python file that acts as a back-end for our system, where it is responsible for Linking the front-end and bank-end parts. It is also responsible for redirecting the pages from one to another.
- The file 'Views.py' is also a python file, responsible for back-end operations. It contains the machine learning model, which we have built and thus responsible for our system to make predictions, based on the model.

7.2 SAMPLE CODE TEMPLATE

#Home.html

<h1>

WELCOME TO DIABETES PREDICTION SYSTEM

</h1>

<form action="predict">

<input type="submit" value="Get started">

</form>

i) **home:**

```
<header>
<div class="container">
<div class="diabetes">
<a href="" class="logo">Diabetes Prediction System</a>
</div>
<div class="navbar">
<ul>
<li><a href="#home" class="active">Home</a></li>
<li><a href="#about">About</a></li>
<li><a href="#contact">Contact</a></li>
</ul>
</div>
</div>
</header>
<section class="home" id="home">
<div class="container">
<div class="home-content">
<div class="block">
<div align='center'>
<h1> DIABETES PREDICTION SYSTEM</h1>
<form action="predict">
  <input type="submit" value="Get started">
</form>
</div>
</div>
</div>
</div>
</div>
</section>
```

ii) **about:**

```
<!--About section-->
<section class="about-us" id="about">
<div class="container">
<div class="row">
<div class="section-title">
<h1>About Us</h1>
</div>
</div>
<div class="row">
<div class="about-content">
<div class="row">
<div class="text">
<p><strong>DiabetesPrediction System</strong> aims at predicting Diabetes at an
Early stage, So that Early Diagnosis and necessary treatment would be possible.With
the help of Diabetes Prediction System,a person can check his/her Diabetic condition
and need not to visit a diagnostic center anymore,to get diagnosed.</p>
</div>
</div>
</div>
</div>
</div>
</section>
```


iii) **contact:**

```
<!--contact section-->
<section class="contact-us" id="contact">
<div class="container">
<div class="row">
<div class="section-title text-center">
  <h1>Contact Us</h1>
</div>
</div>
<div class="row">
<div class="contact-form">
<div class="row">
<div class="text">
  <h2>Drop Us a line</h2>
  <p>We are here to answer any question you may have</p>
</div>
</div>
<div class="row space-between">
<div class="col-6">
<input type="text" class="form-control" name="" placeholder="Name">
</div>
<div class="col-6">
<input type="text" class="form-control" name="" placeholder="Email">
</div>
</div>
<div class="row">
<div class="col-12">
<textarea class="form-control" placeholder="Your Comment"></textarea>
</div>
```

```

</div>
<div class="row">
<div class="button text-center">
  <a href="">Get in Touch</a>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</section>

```

iv) **social-media:**

```

<footer>
<div class="container">
<div class="logo">
  Diabetes Prediction System
</div>
<div class="social">
<a href="https://www.instagram.com/"><i class="fa fa-instagram"></i></a>
<a href="https://twitter.com/?lang=en"><i class="fa fa-twitter"></i></a>
<a href="https://www.facebook.com/"><i class="fa fa-facebook"></i></a>
<a href="https://www.youtube.com/channel/"><i class="fa fa-youtube"></i></a>
</div>
</div>
</footer>
<section class="copyright">
<p>© Copyright Diabetes Prediction System 2021. All Rights Reserved.</p>
</div>
</section>

```

#Predict.html

```
<div align='center' class="main">
<h1>Please enter the following information:</h1>
<form id = "frm1" action="result">
<style>
  tr,td
  {
  width:150px;
  text-align:center;
  padding:5px;
  }
</style>
<table>
  <tr>
    <td align="right">Pregnancies:</td>
    <td align="left"><input type="text" name="n1"></td>
  </tr>
  <tr>
    <td align="right">Glucose:</td>
    <td align="left"><input type="text" name="n2"></td>
  </tr>
  <tr>
    <td align="right">Bloodpressure:</td>
    <td align="left"><input type="text" name="n3"></td>
  </tr>
  <tr>
    <td align="right">Skin Thickness:</td>
    <td align="left"><input type="text" name="n4"></td>
  </tr>
```

```

<tr>
  <td align="right">Insulin:</td>
  <td align="left"><input type="text" name="n5"></td>
</tr>

<tr>
  <td align="right">BMI:</td>
  <td align="left"><input type="text" name="n6"></td>
</tr>

<tr>
  <td align="right">Diabetes Pedigree function:</td>
  <td align="left"><input type="text" name="n7"></td>
</tr>

<tr>
  <td align="right">Age:</td>
  <td align="left"><input type="text" name="n8"></td>
</tr>
</table>

<input type="submit">

</form>

<p></p>

<h1>Result: { {result2} }</h1>

</div>

```

#Urls.py

```
urlpatterns = [  
    path('admin/', admin.site.urls),  
    path("", views.home),  
    path("predict/", views.predict),  
    path("predict/result", views.result),  
]
```

#Views.py

```
def home(request):  
    return render(request, 'home.html')  
def predict(request):  
    return render(request, 'predict.html')  
def result(request):  
    knn = KNeighborsClassifier()  
  
    knn.fit(X_train, y_train)  
    result1=""  
    if pred==[1]:  
        result1="Positive"  
    else:  
        result1="Negative"  
  
    return render(request, "predict.html", {"result2":result1 })
```

#Settings.py

```
ROOT_URLCONF = 'mainprj.urls'
```

```
TEMPLATES = [
```

```
{
```

```
    'BACKEND': 'django.template.backends.django.DjangoTemplates',
```

```
    'DIRS': [os.path.join(BASE_DIR, 'templates')]
```

```
# Static files (CSS, JavaScript, Images)
```

```
# https://docs.djangoproject.com/en/2.2/howto/static-files/
```

```
STATIC_URL = '/static/'
```

```
STATICFILES_DIRS= (
```

```
    os.path.join(BASE_DIR, 'static'),
```

```
)
```

```
STATIC_ROOT=os.path.join(os.path.dirname(BASE_DIR), 'static')
```

7.3 A BRIEF OUTLINE OF PACKAGES (MACHINE LEARNING)

7.3.1 Train-Test Split

The train-test split is a technique for evaluating the performance of a machine learning algorithm. It can be used for classification or regression problems and can be used for any supervised learning algorithm. The procedure involves taking a dataset and dividing it into two subsets.

Train Dataset: Used to fit the machine learning model.

Test Dataset: Used to evaluate the fit machine learning model.

7.3.2 The StandardScaler Transform

Scaling of Features is an essential step in modeling the algorithms with the datasets. the data obtained contains features of various dimensions and scales altogether. Different scales of the data features affect the modeling of a dataset adversely. It leads to a biased outcome of predictions in terms of misclassification error and accuracy rates. Thus, it is necessary to Scale the data prior to modeling. Python sklearn library offers us with StandardScaler() function to standardize the data values into a standard format.

7.3.3 The KNeighborsClassifier

When we trained the KNN on training data, it took the following steps for each data sample:

- Calculate the distance between the data sample and every other sample with the help of a method such as Euclidean.
- Sort these values of distances in ascending order.
- Choose the top K values from the sorted distances.
- Assign the class to the sample based on the most frequent class in the above K values.

7.4 SAMPLE CODE TEMPLATES

IMPORTING LIBRARIES:

First of all, we will import the following libraries in order for our system to work.



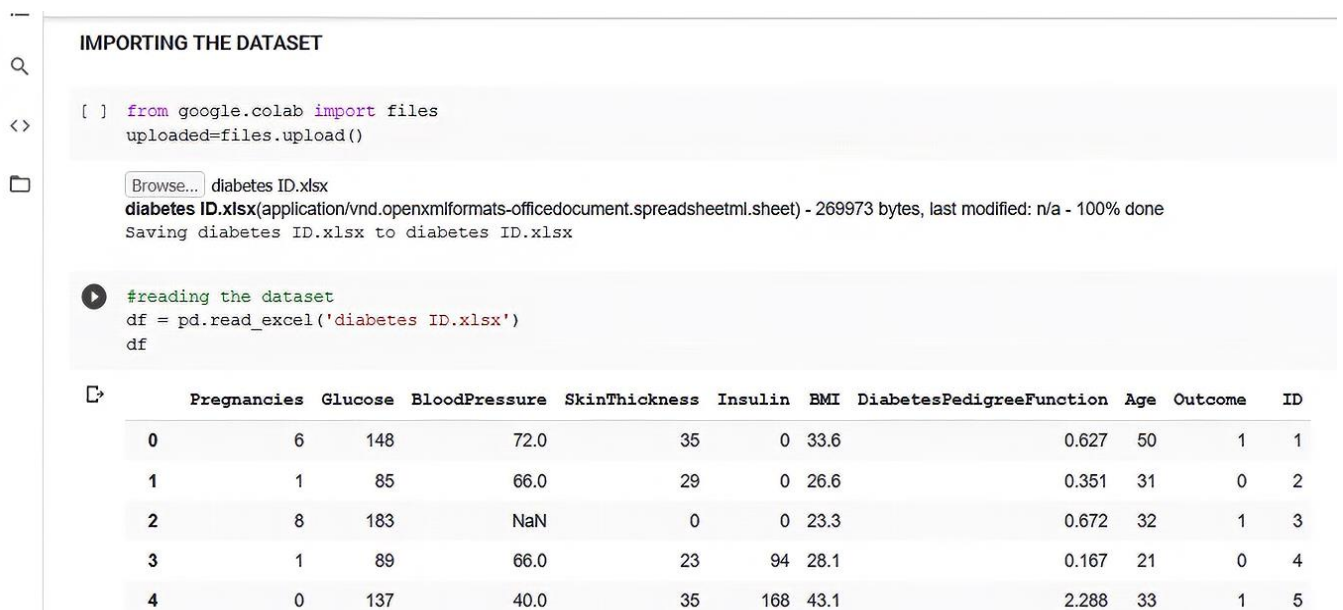
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import sklearn.impute

import warnings
warnings.filterwarnings('ignore')

sns.set()
%matplotlib inline
```

DATA DESCRIPTION:

Next, We have our data saved in a xlsx file called `diabetes ID.xlsx`. We first read our dataset into a pandas dataframe called `df`, and then we show the records from our dataset.



```
from google.colab import files
uploaded=files.upload()

#reading the dataset
df = pd.read_excel('diabetes ID.xlsx')
df
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome	ID
0	6	148	72.0	35	0	33.6	0.627	50	1	1
1	1	85	66.0	29	0	26.6	0.351	31	0	2
2	8	183	NaN	0	0	23.3	0.672	32	1	3
3	1	89	66.0	23	94	28.1	0.167	21	0	4
4	0	137	40.0	35	168	43.1	2.288	33	1	5

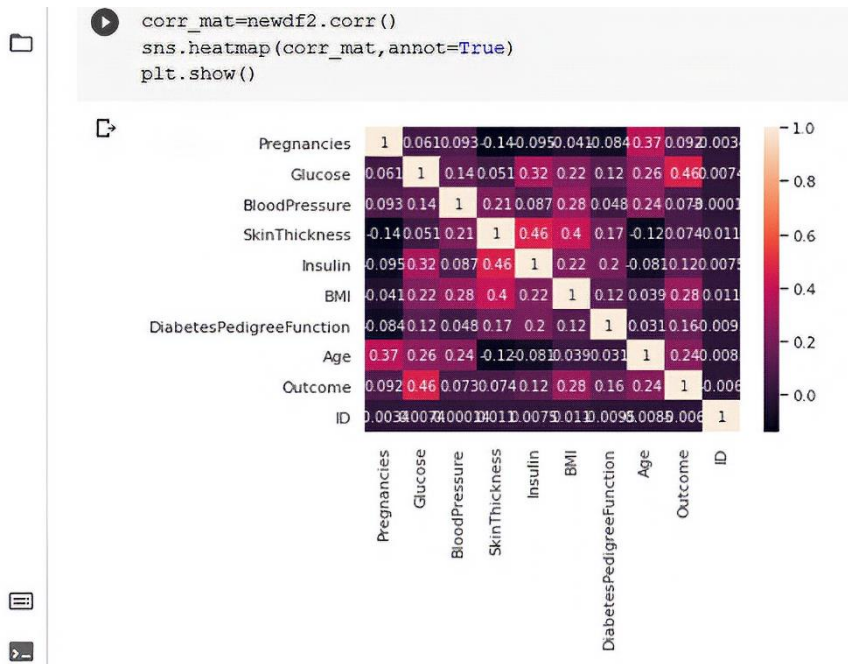
DATA EXPLORATION:

Let us now explore our data set to get a feel of what it looks like and get some insights about it. Let's start by finding correlation of every pair of features (and the outcome variable), and visualize the correlations using a heatmap.

CORRELATION MATRIX

```
[ ] newdf2.corr()
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome	ID
Pregnancies	1.000000	0.061308	0.092751	-0.137195	-0.095281	-0.040976	-0.084208	0.370447	0.091583	-0.003406
Glucose	0.061308	1.000000	0.139151	0.051246	0.322342	0.224655	0.119880	0.255922	0.459708	0.007405
BloodPressure	0.092751	0.139151	1.000000	0.213673	0.087209	0.283341	0.048272	0.237038	0.073023	-0.000141
SkinThickness	-0.137195	0.051246	0.213673	1.000000	0.458872	0.401681	0.171326	-0.118707	0.074318	0.010850
Insulin	-0.095281	0.322342	0.087209	0.458872	1.000000	0.219661	0.199175	-0.080712	0.122046	0.007480
BMI	-0.040976	0.224655	0.283341	0.401681	0.219661	1.000000	0.117294	0.038922	0.283706	0.011370
DiabetesPedigreeFunction	-0.084208	0.119880	0.048272	0.171326	0.199175	0.117294	1.000000	0.031367	0.155530	-0.009510
Age	0.370447	0.255922	0.237038	-0.118707	-0.080712	0.038922	0.031367	1.000000	0.236826	-0.008482
Outcome	0.091583	0.459708	0.073023	0.074318	0.122046	0.283706	0.155530	0.236826	1.000000	-0.005971
ID	-0.003406	0.007405	-0.000141	0.010850	0.007480	0.011370	-0.009510	-0.008482	-0.005971	1.000000



In the above heatmap, brighter colors indicate more correlation. As we can see from the table and the heatmap, glucose levels, age, BMI and Diabetes pedigree function all have slightly high correlation with the outcome variable.

Stage-1: Data preprocessing

a) Data cleaning:

```
df.isna().sum() #Check for missing data

newdf2=df.fillna(method='bfill') #Replace missing data
newdf2.describe() #Describing the data
newdf2.duplicated().sum() #check for duplicate records
print(newdf2['Pregnancies'].quantile(0.50))
print(newdf2['Pregnancies'].quantile(0.95))
newdf2['Pregnancies']=np.where(newdf2['Pregnancies']>6,3,newdf2['Pregnancies']) #Removing Outliers
newdf2.describe()
```

So, the final data, after Data cleaning, looks like this :



	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome	ID
0	6	148	72.0	35	0	33.6	0.627	50	1	1
1	1	85	66.0	29	0	26.6	0.351	31	0	2
2	3	183	66.0	0	0	23.3	0.672	32	1	3
3	1	89	66.0	23	94	28.1	0.167	21	0	4
4	0	137	40.0	35	168	43.1	2.288	33	1	5
...
4763	2	75	64.0	24	55	29.7	0.370	33	0	4764
4764	3	179	72.0	42	130	32.7	0.719	36	1	4765
4765	6	85	78.0	0	0	31.2	0.382	42	0	4766
4766	0	129	110.0	46	130	56.0	0.319	26	1	4767
4767	2	81	72.0	15	76	30.1	0.547	25	0	4768

4768 rows x 10 columns

b) Dimensionality reduction:

```
newdf3=newdf2.drop('ID',axis=1) #Dimensionality reduction
```

Q

<>

📄

🔍

DROPPING UNIMPORTANT COLUMNS

```
newdf3=newdf2.drop('ID',axis=1)  
newdf3
```

🔍

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72.0	35	0	33.6	0.627	50	1
1	1	85	66.0	29	0	26.6	0.351	31	0
2	3	183	66.0	0	0	23.3	0.672	32	1
3	1	89	66.0	23	94	28.1	0.167	21	0
4	0	137	40.0	35	168	43.1	2.288	33	1
...
4763	2	75	64.0	24	55	29.7	0.370	33	0
4764	3	179	72.0	42	130	32.7	0.719	36	1
4765	6	85	78.0	0	0	31.2	0.382	42	0
4766	0	129	110.0	46	130	56.0	0.319	26	1
4767	2	81	72.0	15	76	30.1	0.547	25	0

📄

4768 rows x 9 columns

c) Data transformation:

```
scaler = StandardScaler()  
X_train = scaler.fit_transform(X_train) #Feature  
scaling  
X_test = scaler.transform(X_test)  
  
scaler.fit_transform(df[['Age','Insulin']])
```

```
▶ scaler.fit_transform(df[['Age','Insulin']])  
  
array([[ 1.43355141, -0.72069667],  
       [-0.17953892, -0.72069667],  
       [-0.09463943, -0.72069667],  
       ...,  
       [ 0.75435548, -0.72069667],  
       [-0.60403638,  0.4502942 ],  
       [-0.68893587, -0.03611739]])
```

Stage-2: Test-Train split

```
#splitting data into test and train

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(
X, y, test_size = 0.2)
```

Stage-3: Building a model

```
#fitting data to the model

from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix
knn = KNeighborsClassifier()
knn.fit(X_train, y_train)
y_pred = knn.predict(X_test)

knn_train_acc = accuracy_score(y_train, knn.predict(
X_train))
```

Stage-4: Linking ML code with Django

```
def home(request):
    return render(request, 'home.html')
def predict(request):
    return render(request, 'predict.html')
def result(request):
    knn = KNeighborsClassifier()

    knn.fit(X_train, y_train)
    result1=""
    if pred==[1]:
        result1="Positive"
    else:
        result1="Negative"

    return render(request, "predict.html", {"result2":result1 })
```

8. RESULTS

The main aim of this project was to design and implement a Diabetes Prediction System Using Machine Learning and Django and it has been achieved successfully.

In this project, we have created a user interface, from which a person can check his diabetic condition, based on some medical parameters.

Our model has achieved an accuracy of 96 % and can be used to check whether a person is diabetic or not.

KNN Classifier Train-Test Accuracy:

To build the system, we have used KNN Classifier and We can see below, that the model achieved a Train accuracy of 99 % and a Test accuracy of 96 %

1) K NEIGHBOURS CLASSIFIER (KNN)

```
#fitting data to thhe model
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix
knn = KNeighborsClassifier()
knn.fit(X_train, y_train)

y_pred = knn.predict(X_test)

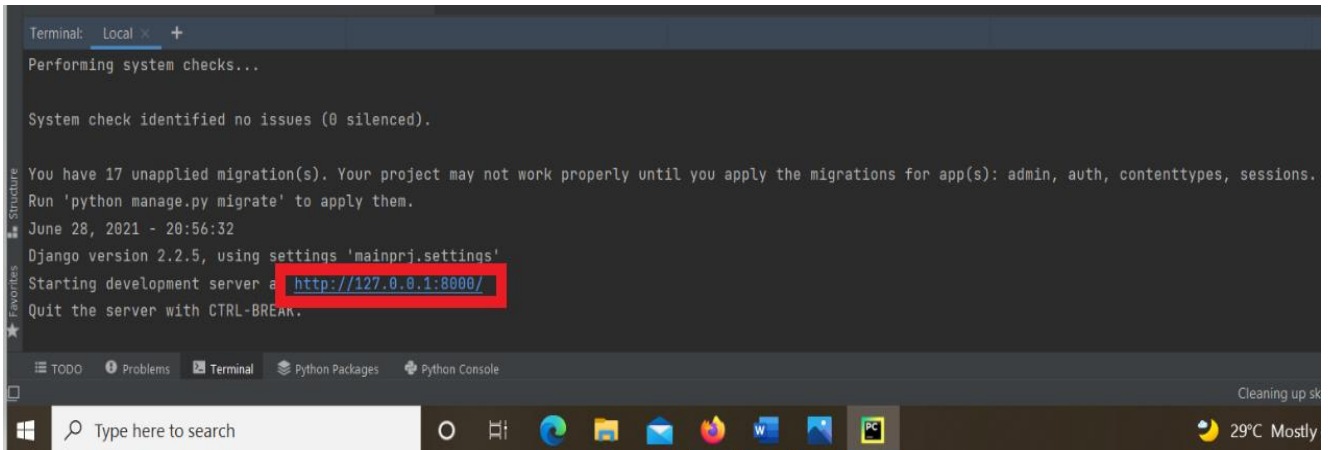
knn_train_acc = accuracy_score(y_train, knn.predict(X_train))
knn_test_acc = accuracy_score(y_test, y_pred)

print(f"Training Accuracy of KNN Model is {knn_train_acc}")
print(f"Test Accuracy of KNN Model is {knn_test_acc}")
```

```
➞ Training Accuracy of KNN Model is 0.9902988987939172
   Test Accuracy of KNN Model is 0.9591194968553459
```

Launching the Local Host, with Django:

After setting up the environment and running the files, which consists of both Front-end and Back-end, We have launched the Local host, through where we will be redirected to the Diabetes Prediction System. Below we can see the Local host;



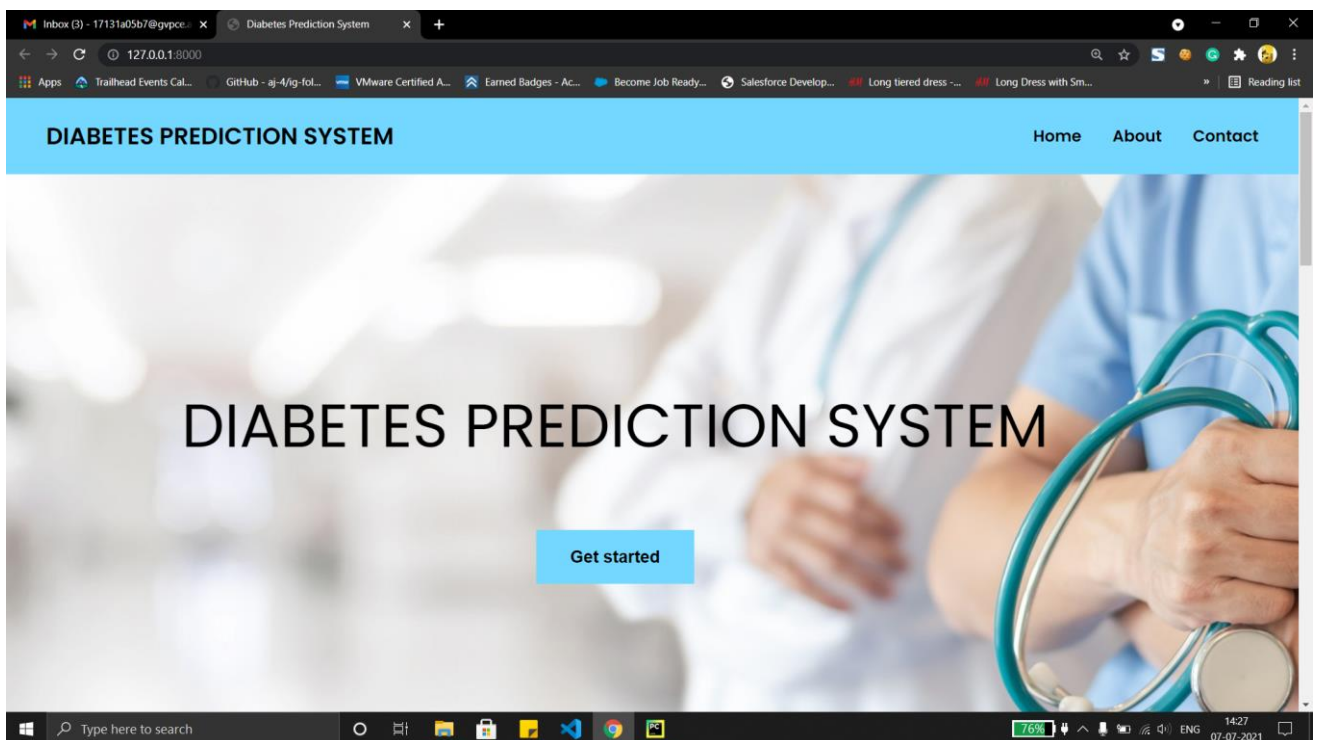
```
Terminal: Local x +
Performing system checks...

System check identified no issues (0 silenced).

You have 17 unapplied migration(s). Your project may not work properly until you apply the migrations for app(s): admin, auth, contenttypes, sessions.
Run 'python manage.py migrate' to apply them.
June 28, 2021 - 20:56:32
Django version 2.2.5, using settings 'mainprj.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
```

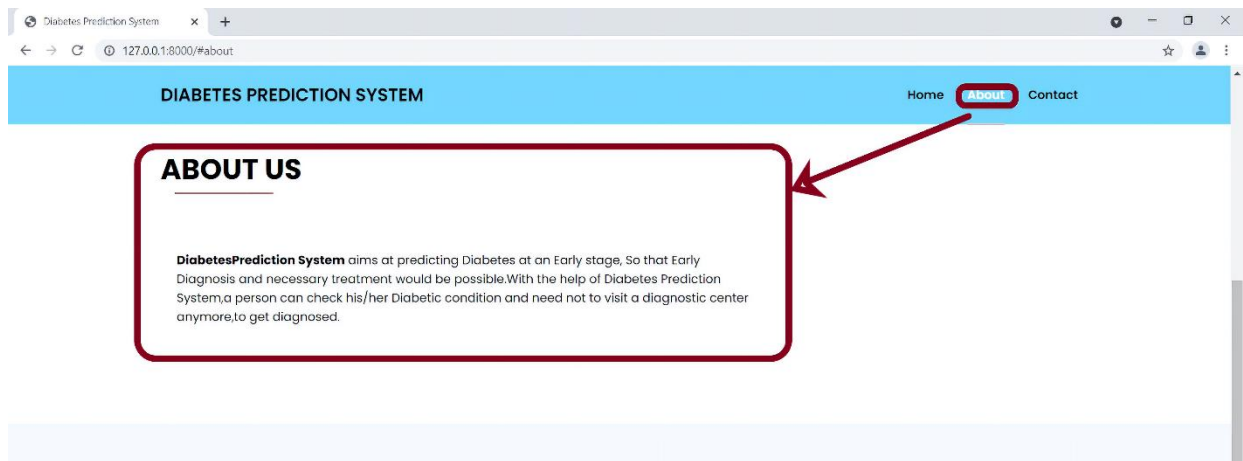
HOME_PAGE of Diabetes Prediction System:

The home page of our system consists of Elements like 'Home', 'About', 'Contact', as shown in the below image.



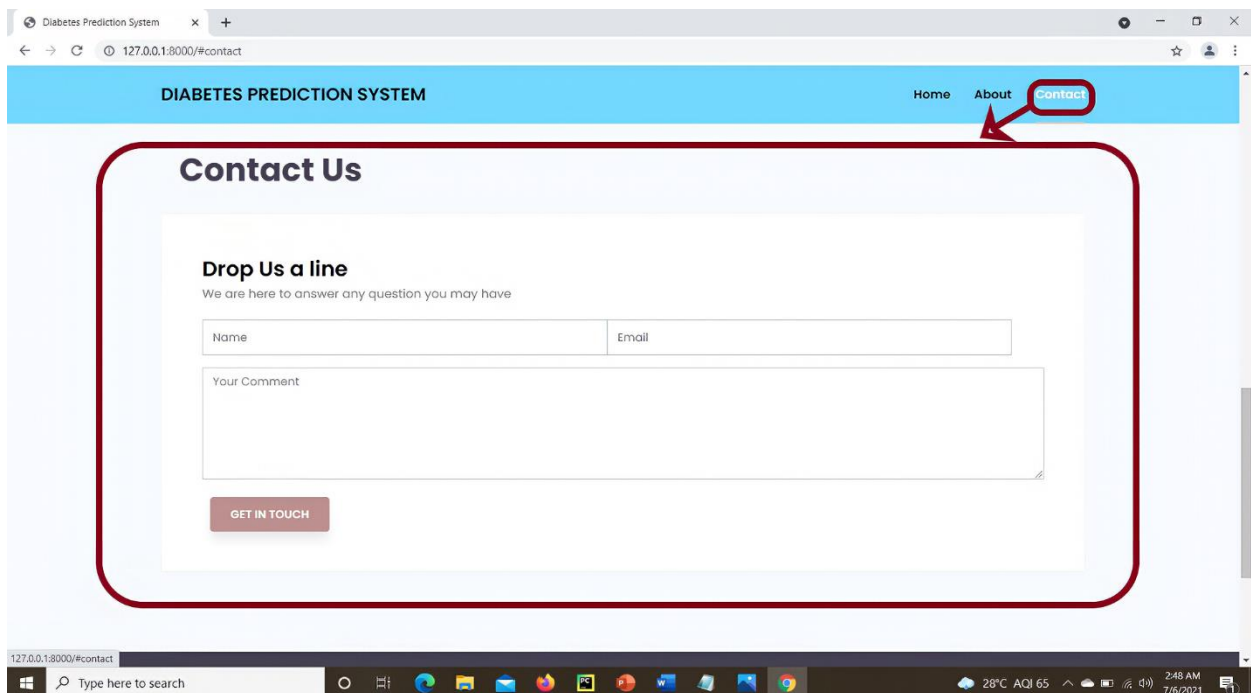
a) About:

When the user clicked on 'Home', he/she will be returned back to the home page, 'About' consists of some information regarding our 'Diabetes Prediction System', which is shown below.



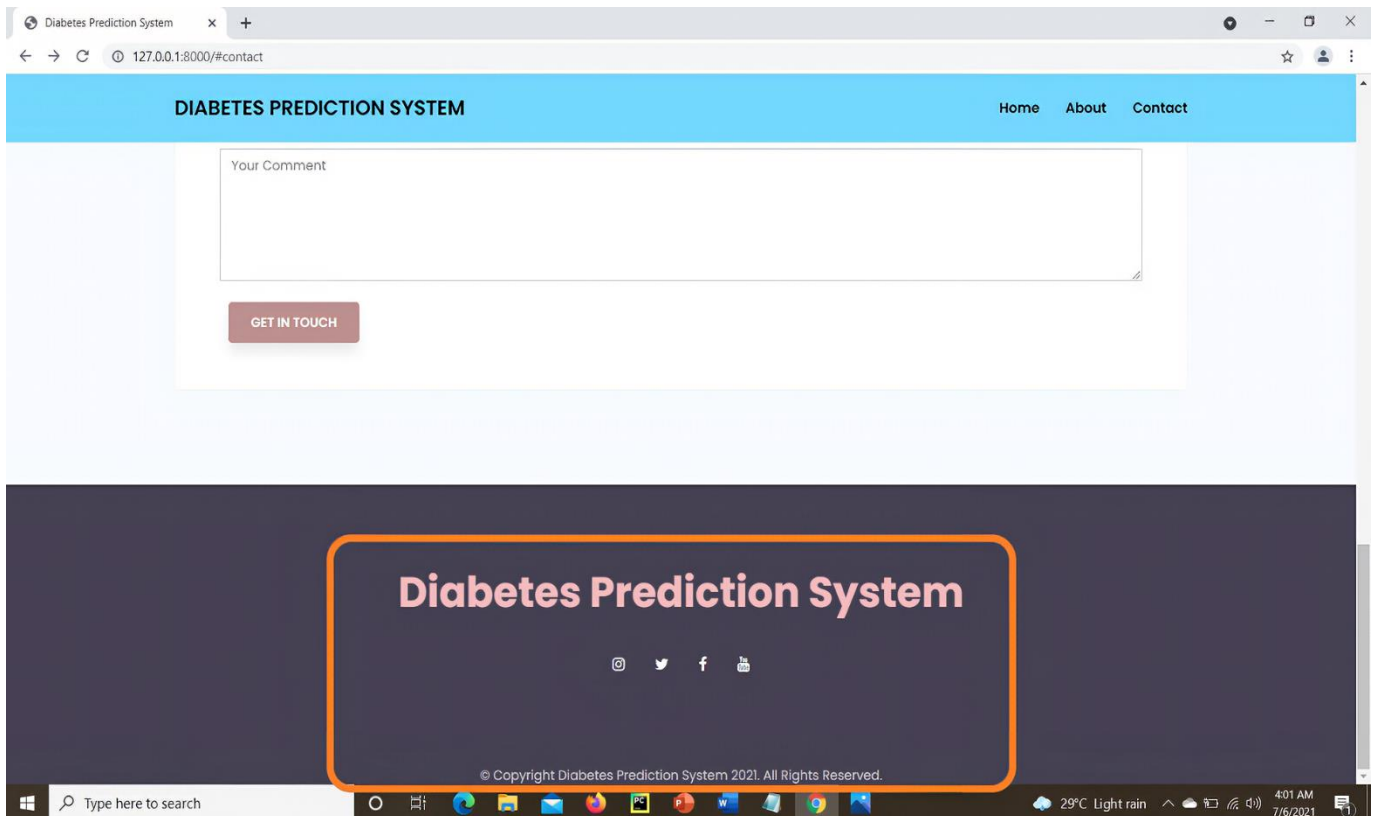
b) Contact:

When a user has some queries or need some help, he/she can reach out to the helpdesk of our system, which is shown below.



c) Social media:

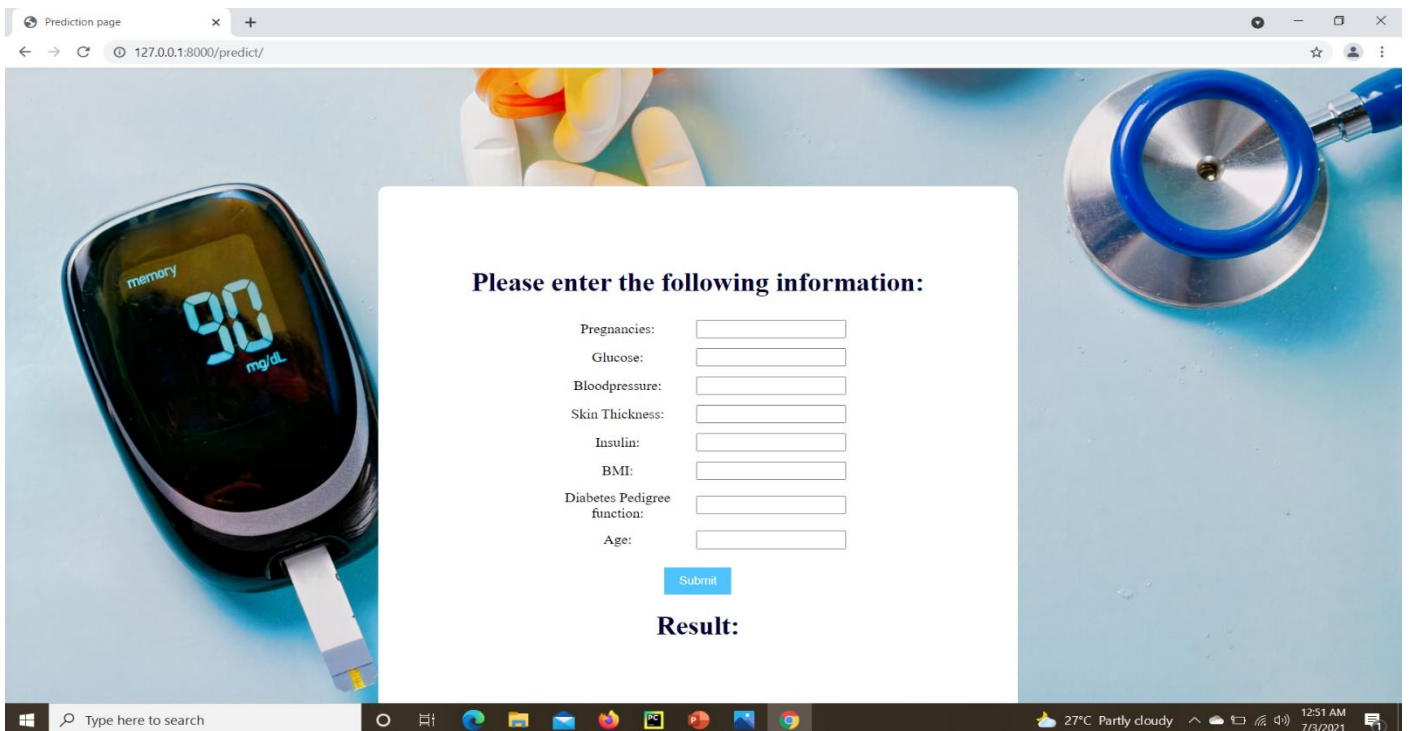
We have also added some social media sites, which helps the user to connect to the system administrators, in case of any queries.



The main reason behind adding all these blocks to our home page is to enrich the user experience and to provide a hassle-free environment, while using our 'Diabetes prediction system'. While the user accesses our system, they will seek for help, if they are facing any trouble via the 'Contact', using our system and can also learn about the importance of Early stage prediction of Diabetes.

PREDICT_PAGE of Diabetes Prediction System:

When the user want to check his/her Diabetic condition, after clicking on the ‘Get Started’ button, the user will be redirected to the Predict page, as shown in the below figure;



Prediction page

127.0.0.1:8000/predict/

Please enter the following information:

Pregnancies:

Glucose:

Bloodpressure:

Skin Thickness:

Insulin:

BMI:

Diabetes Pedigree function:

Age:

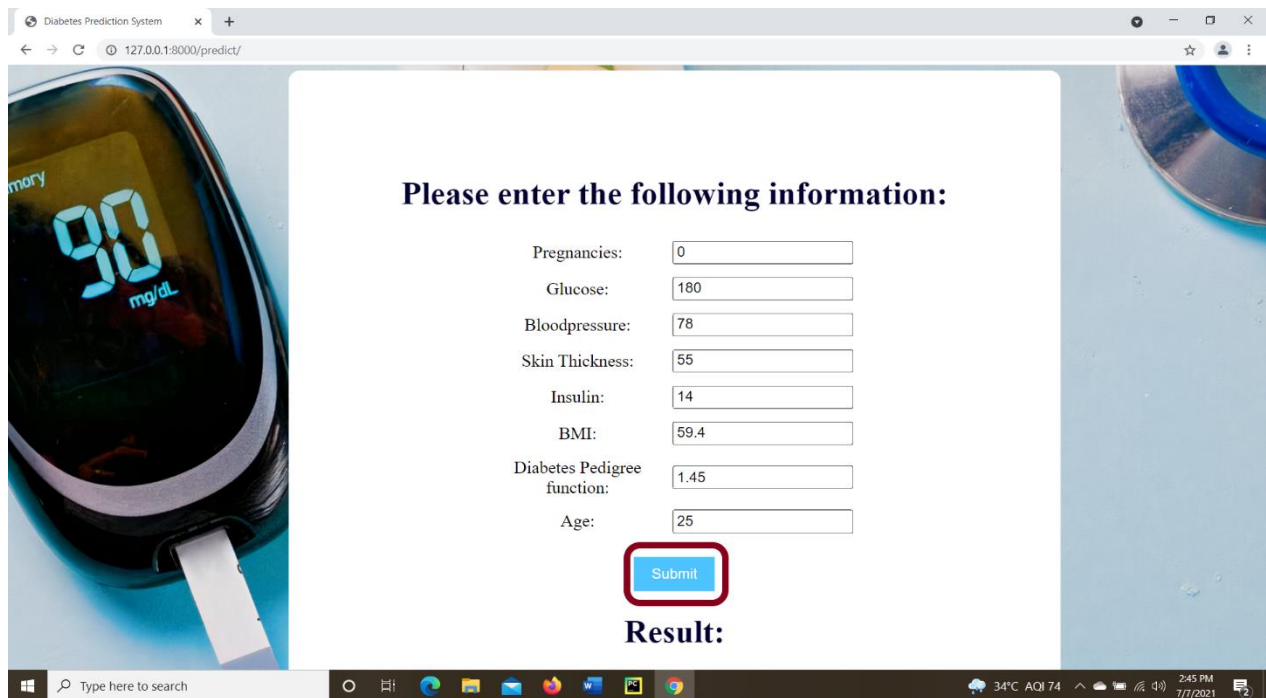
Result:

Here, the user will give the inputs, based on his/her medical conditions and can check whether he/she is Diabetic or not. ‘Positive’ indicates the user has Diabetes and ‘Negative’ indicates the user is not Diabetic.

9. SAMPLE TEST CASES

The Following is the demonstration of some of the test cases:

Case 1: When a person 'A' input the values, the system has to predict "POSITIVE"

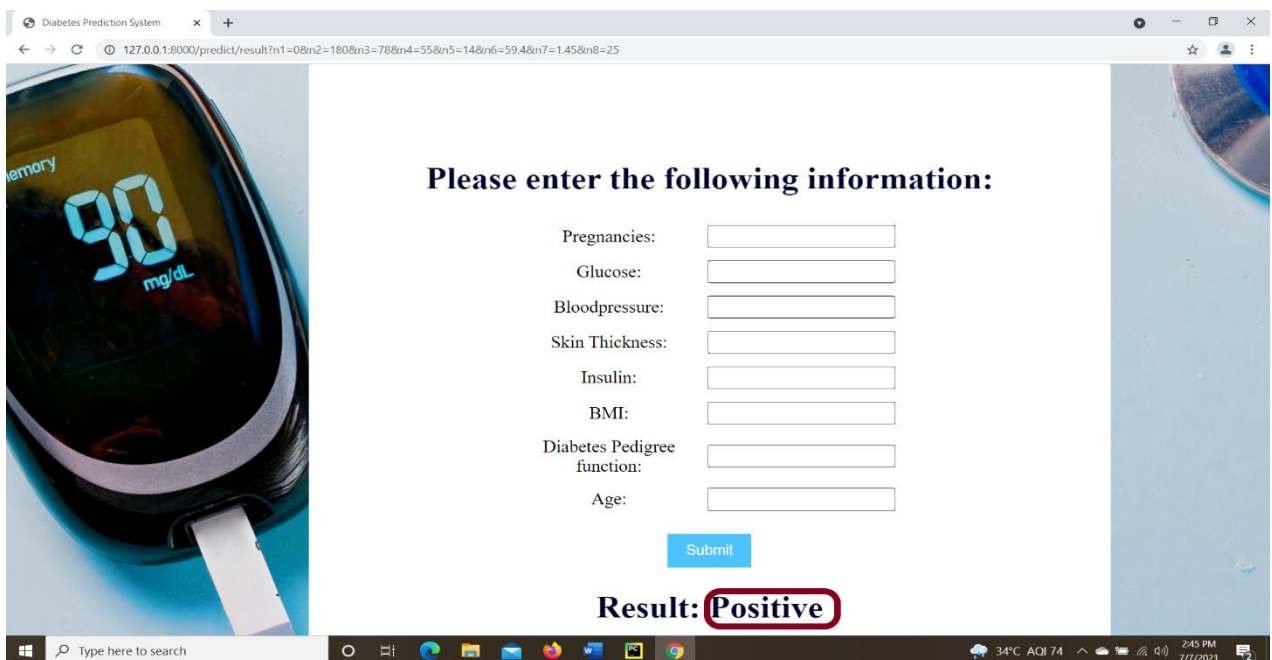


The screenshot shows a web browser window titled "Diabetes Prediction System" with the URL "127.0.0.1:8000/predict/". The page features a background image of a glucose meter on the left. The main content area has the heading "Please enter the following information:" followed by a list of input fields with pre-filled values: Pregnancies: 0, Glucose: 180, Bloodpressure: 78, Skin Thickness: 55, Insulin: 14, BMI: 59.4, Diabetes Pedigree function: 1.45, and Age: 25. A blue "Submit" button is located below the fields. Below the button, the word "Result:" is displayed. The Windows taskbar at the bottom shows the search bar, several application icons, and system status information including 34°C, AQI 74, and the date 7/7/2021.

Pregnancies:	<input type="text" value="0"/>
Glucose:	<input type="text" value="180"/>
Bloodpressure:	<input type="text" value="78"/>
Skin Thickness:	<input type="text" value="55"/>
Insulin:	<input type="text" value="14"/>
BMI:	<input type="text" value="59.4"/>
Diabetes Pedigree function:	<input type="text" value="1.45"/>
Age:	<input type="text" value="25"/>

Result:

INPUT-1



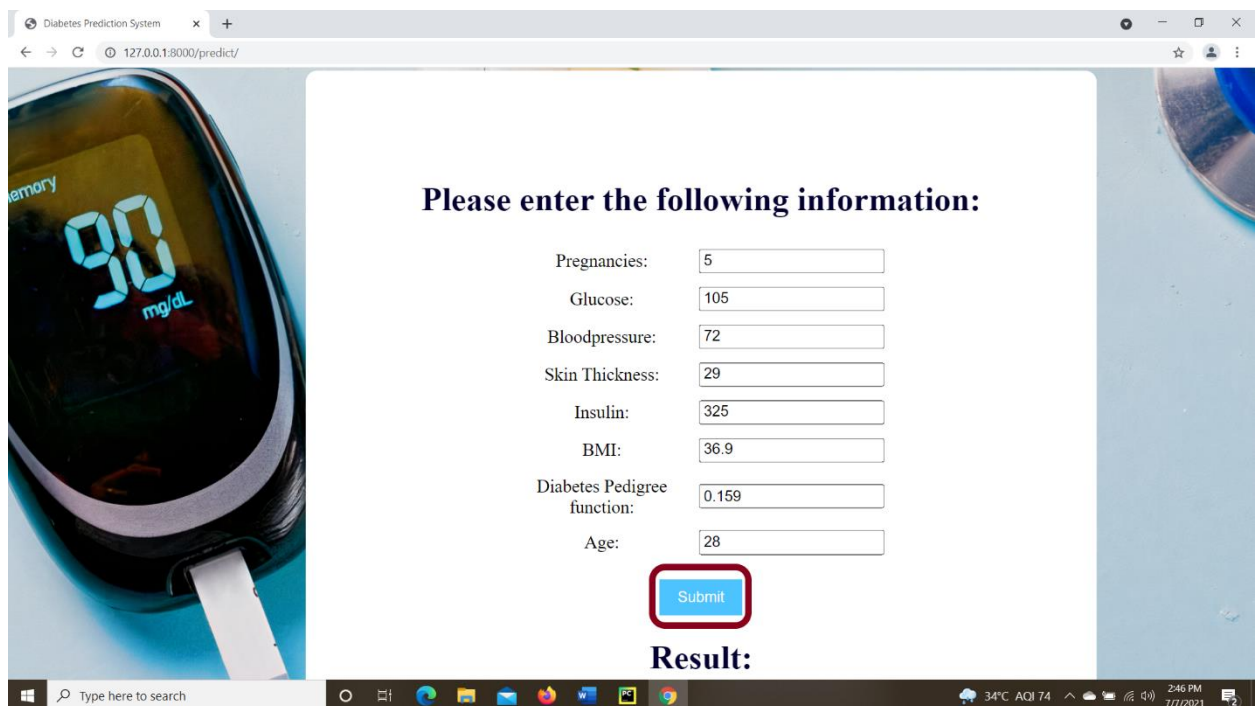
This screenshot shows the same web browser window after the form has been submitted. The URL in the address bar now includes a long query string: "127.0.0.1:8000/predict/result?n1=0&n2=180&n3=78&n4=55&n5=14&n6=59.4&n7=1.45&n8=25". The input fields are now empty. The "Submit" button remains. Below it, the "Result:" label is followed by the word "Positive", which is enclosed in a red rounded rectangle. The Windows taskbar at the bottom is identical to the previous screenshot.

Pregnancies:	<input type="text"/>
Glucose:	<input type="text"/>
Bloodpressure:	<input type="text"/>
Skin Thickness:	<input type="text"/>
Insulin:	<input type="text"/>
BMI:	<input type="text"/>
Diabetes Pedigree function:	<input type="text"/>
Age:	<input type="text"/>

Result: Positive

OUTPUT-1

Case 2: When a person ‘B’ input the values, the system has to predict “NEGATIVE”

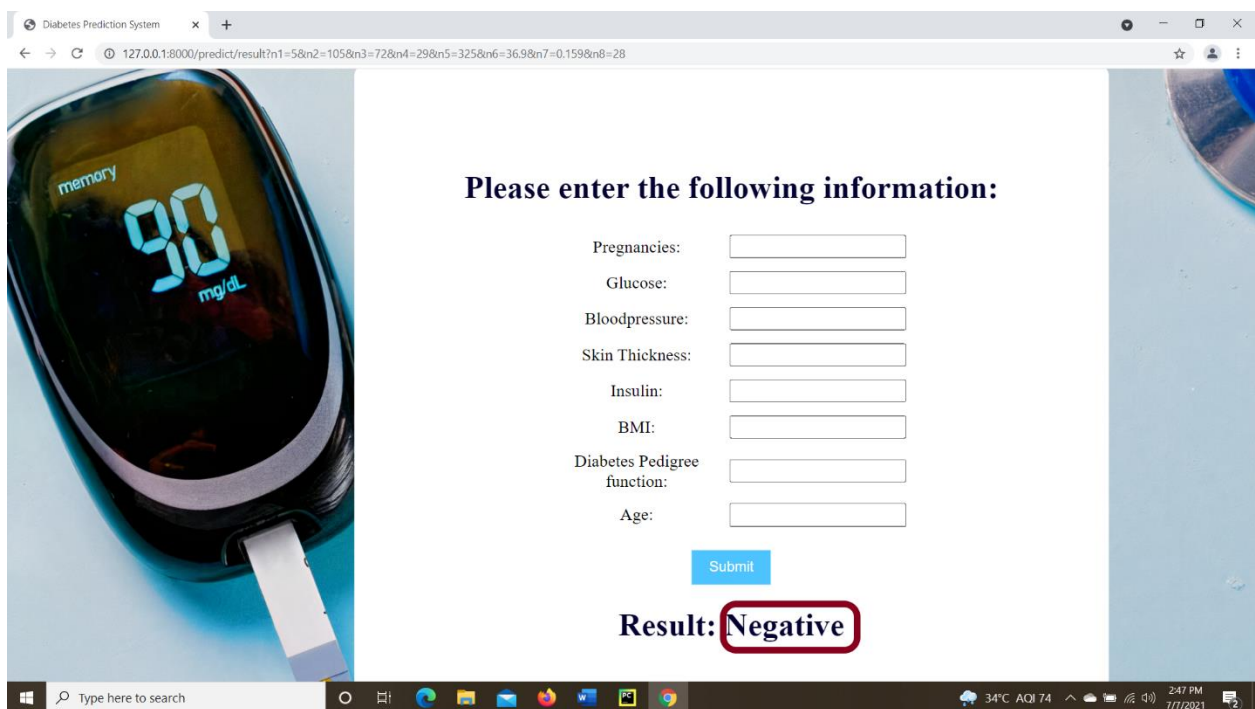


The screenshot shows a web browser window titled "Diabetes Prediction System" with the URL "127.0.0.1:8000/predict/". On the left is a background image of a glucose meter displaying "90 mg/dL". The main content area has the heading "Please enter the following information:" followed by a form with the following fields and values:

Field	Value
Pregnancies:	5
Glucose:	105
Bloodpressure:	72
Skin Thickness:	29
Insulin:	325
BMI:	36.9
Diabetes Pedigree function:	0.159
Age:	28

Below the form is a blue "Submit" button, which is highlighted with a red rectangle. Underneath the button, the text "Result:" is visible.

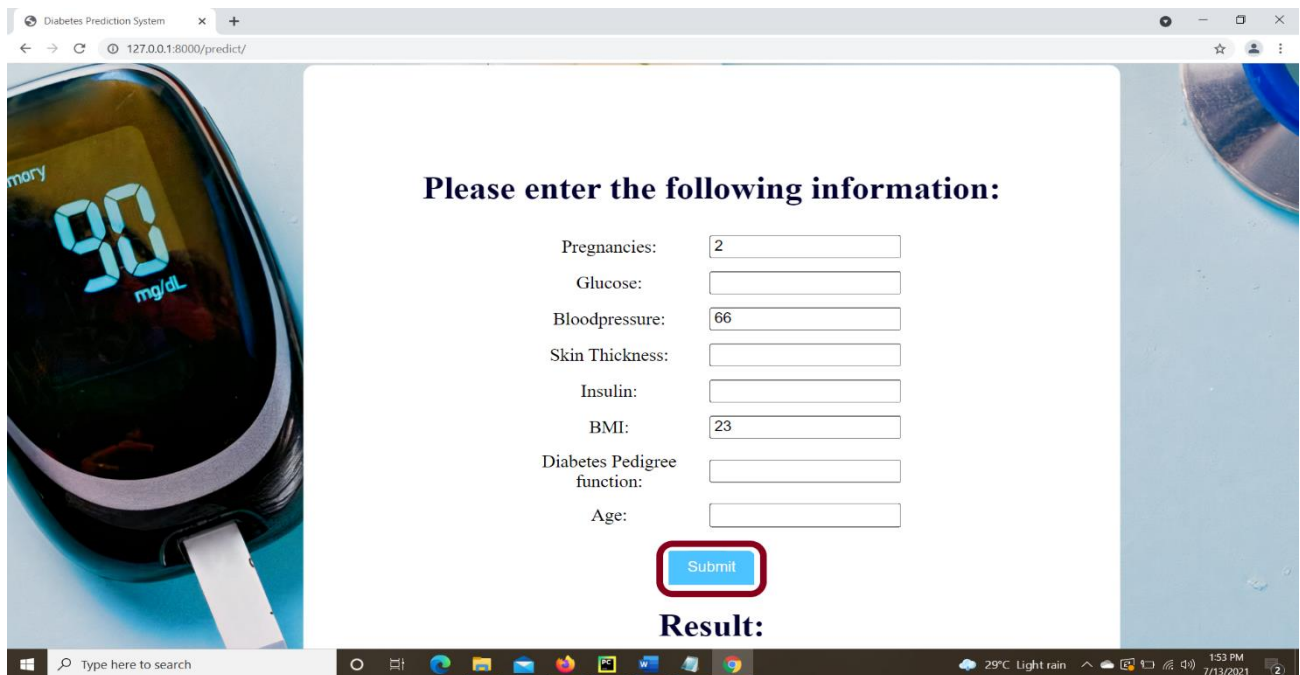
INPUT-2



The screenshot shows the same web browser window, but the URL now includes the prediction result: "127.0.0.1:8000/predict/result?n1=5&n2=105&n3=72&n4=29&n5=325&n6=36.9&n7=0.159&n8=28". The input form fields are now empty. The "Submit" button is still present. Below the button, the text "Result:" is followed by the word "Negative", which is highlighted with a red rectangle.

OUTPUT-2

Case 3: When a person does not input all the values, the system has to display a prompt, as shown below:

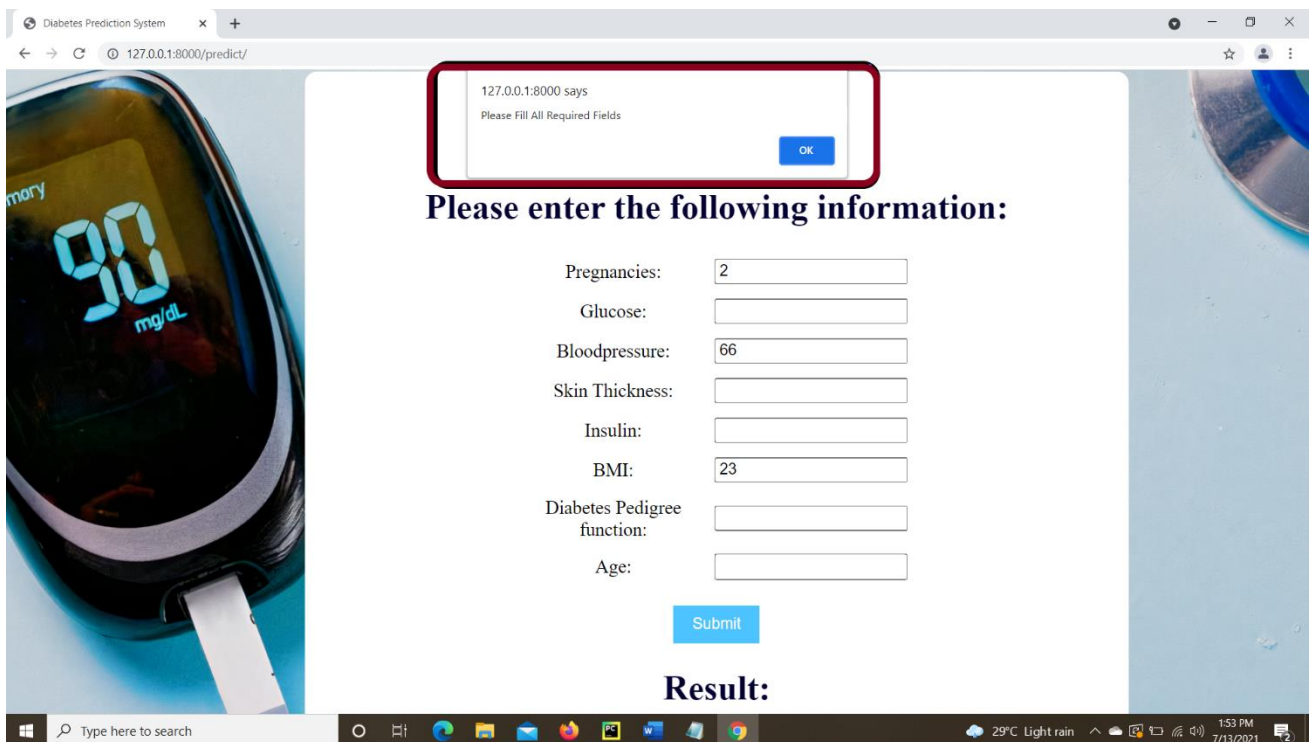


The screenshot shows a web browser window titled "Diabetes Prediction System" with the URL "127.0.0.1:8000/predict/". The page displays a form titled "Please enter the following information:" with the following fields:

- Pregnancies:
- Glucose:
- Bloodpressure:
- Skin Thickness:
- Insulin:
- BMI:
- Diabetes Pedigree function:
- Age:

A red box highlights the "Submit" button. Below the form, the text "Result:" is visible. The background of the page features a digital glucose meter showing "90 mg/dL".

INPUT-3



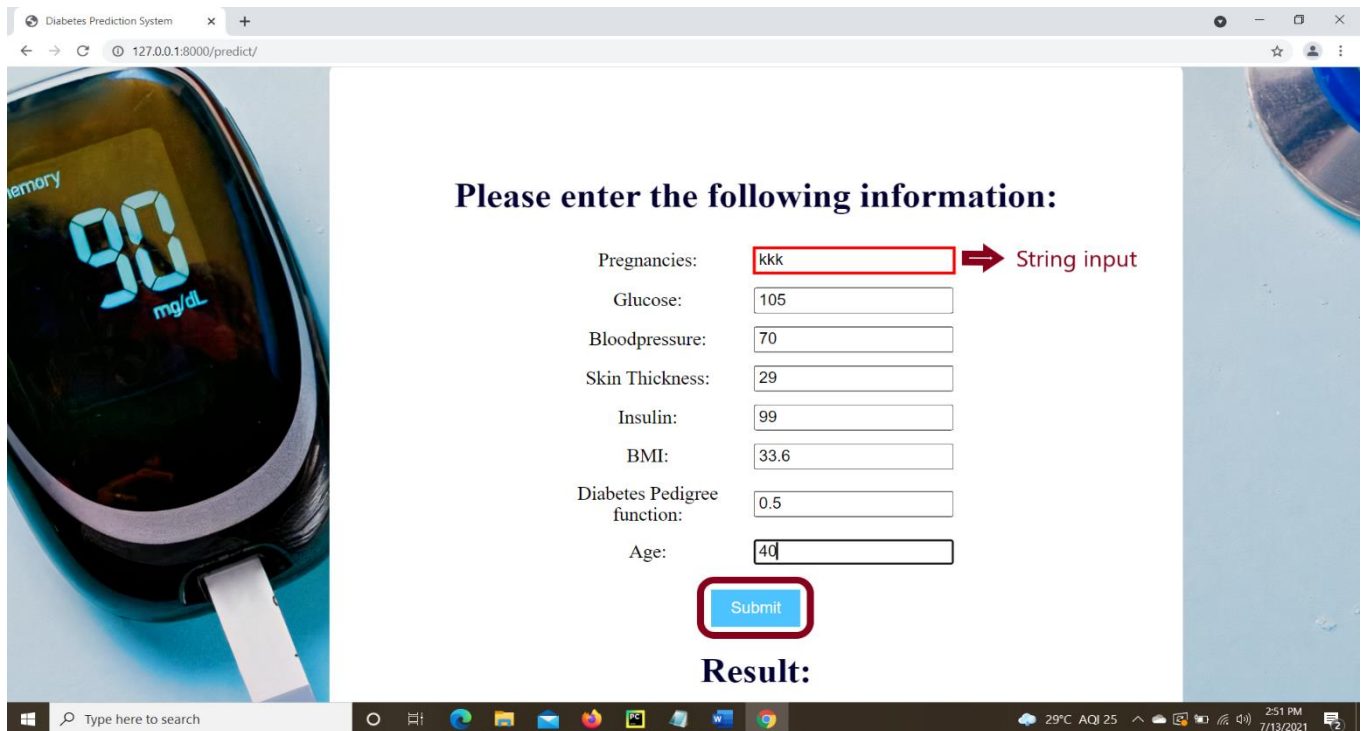
The screenshot shows the same web browser window as before, but with an error message displayed in a red box at the top of the form area. The message reads:

127.0.0.1:8000 says
Please Fill All Required Fields

Below the error message, the "Please enter the following information:" form is visible, with the same fields as in the previous screenshot. The "Submit" button is still highlighted with a red box. The background of the page features a digital glucose meter showing "90 mg/dL".

OUTPUT-3

Case 4: When a person inputs a string / character, the system has to display a prompt, as shown below:



Diabetes Prediction System

Please enter the following information:

Pregnancies: ➔ String input

Glucose:

Bloodpressure:

Skin Thickness:

Insulin:

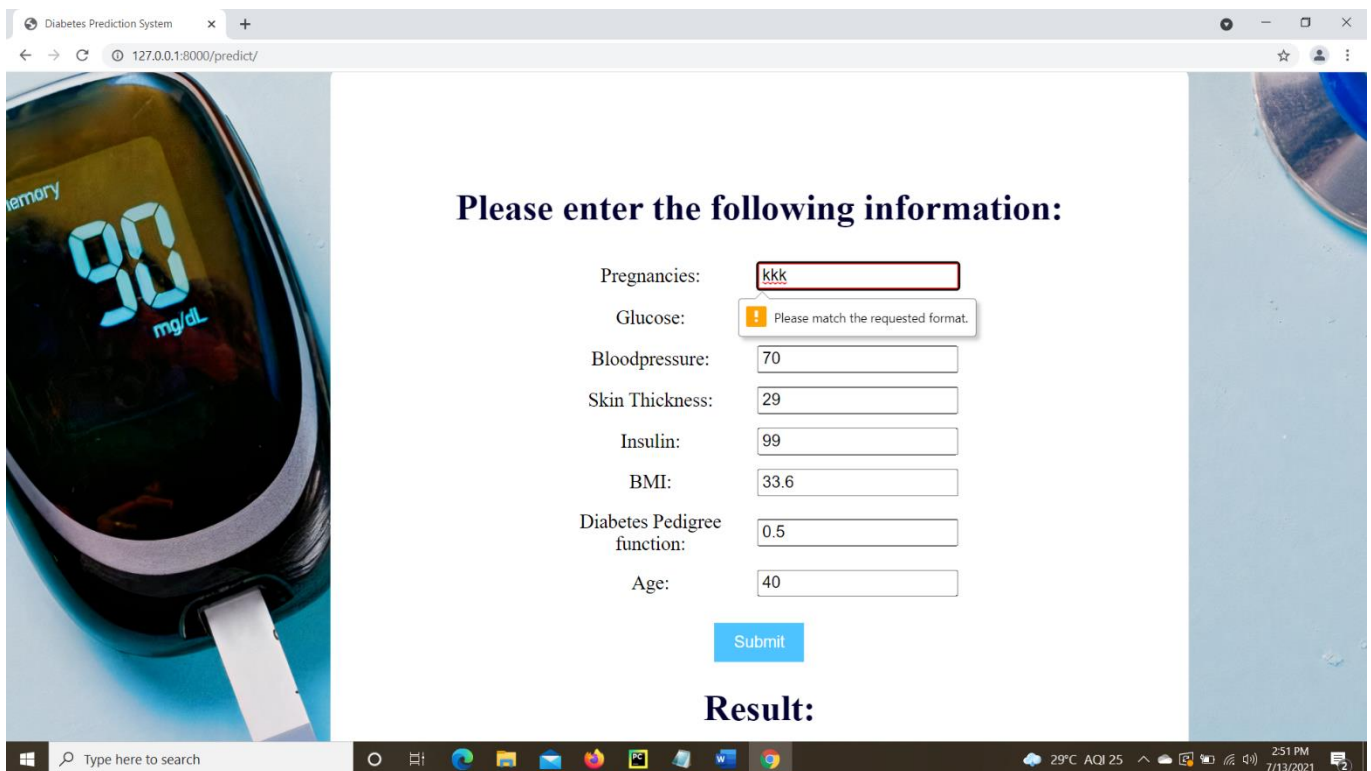
BMI:

Diabetes Pedigree function:

Age:

Result:

INPUT-4



Diabetes Prediction System

Please enter the following information:

Pregnancies:

Glucose: Please match the requested format.

Bloodpressure:

Skin Thickness:

Insulin:

BMI:

Diabetes Pedigree function:

Age:

Result:

OUTPUT-4

10. CONCLUSION

Early stage prediction of diabetes is quite a challenging task, for medical Practitioners. Diabetes can be controlled if it is predicted earlier. To achieve this goal, we will do early prediction of Diabetes. and Early prediction of diabetes could lead to a better treatment of the disease and reduce the risk of hospitalization.

The system, which we have built is useful to physicians, to predict diabetes in the initial stages. So, conventional treatments and solutions may be given to the patients. System used some of the techniques like ML for the prediction, so that to get the more precise results.

11. FUTURE ENHANCEMENTS

While we achieved a successful implementation, significant improvements can be made by addressing several key issues.

- First, a much larger dataset should be designed to improve the model's generality.
- Next, the model has the capacity to improve further by addition of some other medical parameters.

12. BIBLIOGRAPHY

- [1] <https://archive.ics.uci.edu/ml/machine-learning-databases/pima-indians-diabetes/>
- [2] <https://www.sciencedirect.com/science/article/pii/S0169716120300225>
- [3] <https://towardsdatascience.com/machine-learning-an-introduction-23b84d51e6d0#3add>
- [4] <https://docs.djangoproject.com/en/3.2/intro/tutorial01/>
- [5] <https://builtin.com/data-science>
- [6] “Performance Analysis of Machine Learning Techniques to Predict Diabetes Mellitus” Md Faisal Faruque, Asaduzzaman, Iqbal H. Sarker, IEEE 2019.
- [7] “A Comprehensive Exploration to the Machine Learning Techniques for Diabetes Identification” Sidong Wei1, Xuejiao Zhao, Chunyan Miao Shanghai Jiao Tong University, China.
- [8] “Classifying Medical Data to Predict the Likelihood of Disease” by Razan Paul, Abu Sayed Md. Latiful Hoque, IEEE 2010.
- [9] "Analyzing Feature Importances for Diabetes Prediction using Machine Learning", by Debadri Dutta, Debpriyo Paul, Parthajeet Ghosh, IEEE 2018.