

# **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**



**GAYATRI VIDYA PARISHAD COLLEGE OF ENGINEERING (AUTONOMOUS)**

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

Reccredited by NAAC with “A” Grade with CGPA of 3.47/4.00

Madhurawada, Visakhapatnam-530 048

**Gayatri Vidya Parishad 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**  
**JAHNAVI SANJANA PATNALA**  
**KESAVADAS SRI SAI THARUN**  
**KURLI VINEETHA**

**(Reg No: 17131A0562)**  
**(Reg No: 17131A0577)**  
**(Reg No: 17131A0599)**  
**(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 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 VINEEHA (17131A05B7)

PLACE: VISAKHAPATNAM

DATE: 02-07-2021

## 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 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

<b>1. INTRODUCTION</b>	<b>1</b>
1.1 OBJECTIVE	1
1.2 ABOUT THE PROJECT	1
1.3 AN OVERVIEW OF TECHNOLOGY, FRAMEWORK USED	2
1.3.1 MACHINE LEARNING	2
1.3.1.1 KNN CLASSIFIER	2
1.3.2 DJANGO	3
<b>2. LITERATURE SURVEY</b>	<b>4</b>
2.1 EXISTING SYSTEM	4
2.1.1 PREDICTION USING TRADITIONAL WAY, DATA MINING	4
<b>3. DRAWBACKS OF EXISTING SYSTEM</b>	<b>5</b>
<b>4. PROPOSED SYSTEM</b>	<b>6</b>
<b>5. SOFTWARE REQUIREMENT ANALYSIS</b>	<b>7</b>
5.1 PROBLEM STATEMENT	7
5.2 SOFTWARE REQUIREMENT ANALYSIS	7
5.2.1 FUNCTIONAL REQUIREMENTS	7
5.2.2 NON - FUNCTIONAL REQUIREMENTS	8
5.3 SYSTEM REQUIREMENTS	9
5.3.1 SOFTWARE REQUIREMENTS	9
5.3.2 HARDWARE REQUIREMENTS	9
<b>6. SYSTEM DESIGN</b>	<b>10</b>
6.1 ARCHITECTURE	10
6.2 PROCESS FLOW DIAGRAM	10
6.3 CLASS DIAGRAM	11
6.4 SEQUENCE DIAGRAM	12
6.5 CONTROL FLOW DIAGRAM	13
<b>7. IMPLEMENTATION OF WORK</b>	<b>14</b>
7.1 OUTLINE OF FILES USED IN DJANGO	14
7.2 SAMPLE CODE TEMPLATE OF DJANGO	14
7.3 OUTLINE OF PACKAGES IN MACHINE LEARNING	16

7.4 SAMPLE CODE TEMPLATE OF MACHINE LEARNING	17
<b>8. RESULTS</b>	<b>20</b>
<b>9. SAMPLE TEST CASES</b>	<b>22</b>
<b>10. CONCLUSION</b>	<b>24</b>
<b>11. BIBLIOGRAPHY</b>	<b>25</b>

# **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. Diabetic Mellitus (DM) is classified as a Non-Communicable Disease (NCD) and many people are suffering from it.

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, we have trained various machine learning algorithms like KNN, Logistic regression, SVM, Naïve bayes classifier, Random forest. We have selected KNN as our main algorithm, to build the predictive system using KNN. After building the model, we have developed a webpage, using Django, which takes the inputs from the user and checks whether he/she is diabetic or not.



## **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.

### **1.3.1 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.

#### **1.3.1.1 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 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.

KNN has the following basic steps:

1. Calculate distance
2. Find closest neighbors
3. Vote for labels

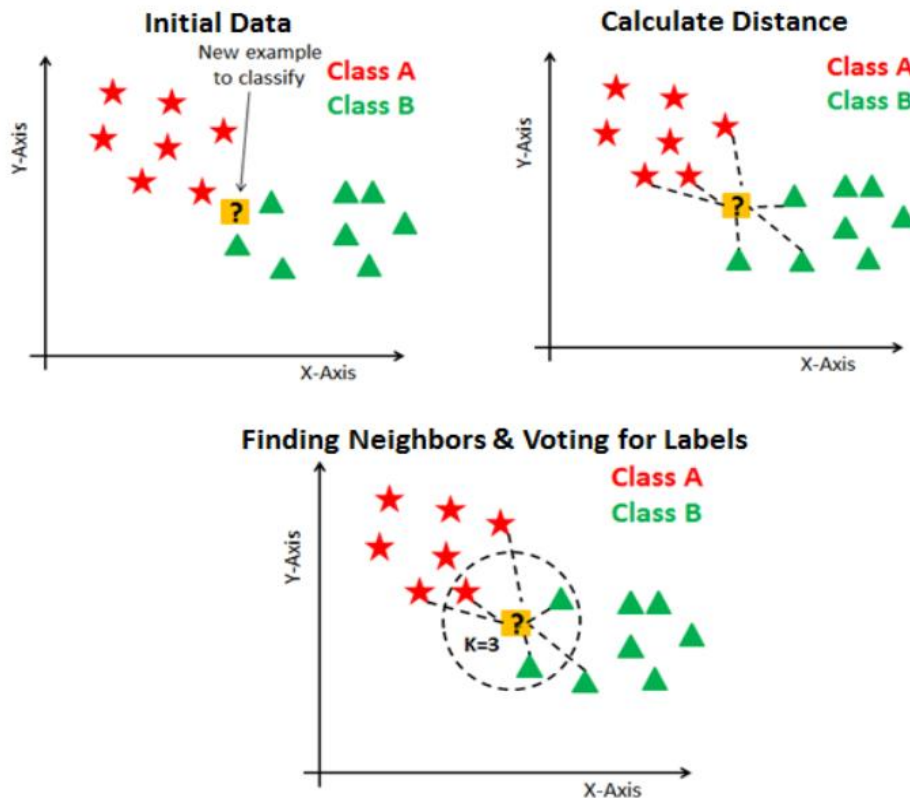


Fig a: Steps in KNN algorithm

### 1.3.2 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.

## **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.

#### **2.1.1 PREDICTION USING DATA MINING (TRADITIONAL WAY)**

The existing instances of datasets consist of only two groups i.e., blood tests and urine tests. Implementation is done by the WEKA tool, to classify the data. Current practice in hospitals is to collect required information for diabetes diagnosis through various tests, and appropriate treatment is provided based on diagnosis. Various traditional methods, based on physical and chemical tests, are available for diagnosing diabetes.

### **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. In the existing method, the classification and prediction accuracy are not so high and is a time taking process, as it needs a heavy investment of time, to get the reports.

Early stage prediction cannot be possible, as it is a challenging task for medical practitioners. Classifying the data using the WEKA tool can only be performed only on small datasets.

## 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.

## **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. This is a major cause for other problems and diseases such as kidney diseases and heart 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. 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 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

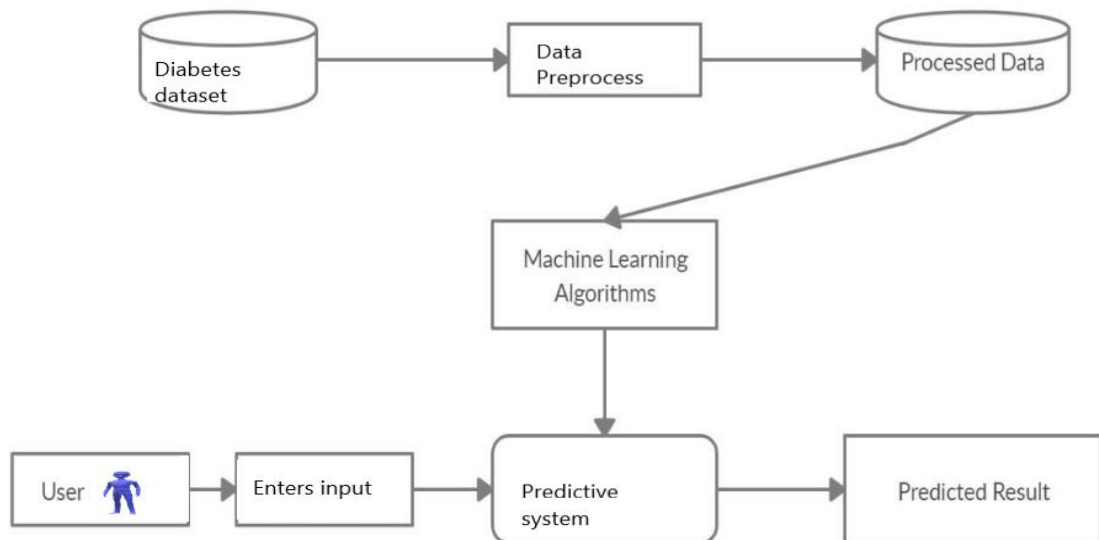
### **5.3.2 HARDWARE REQUIREMENTS**

Processor	: Intel/ARM processor
RAM	: 8GB
Disc space	: 500 GB

## 6. SYSTEM DESIGN

### 6.1 ARCHITECTURE DIAGRAM

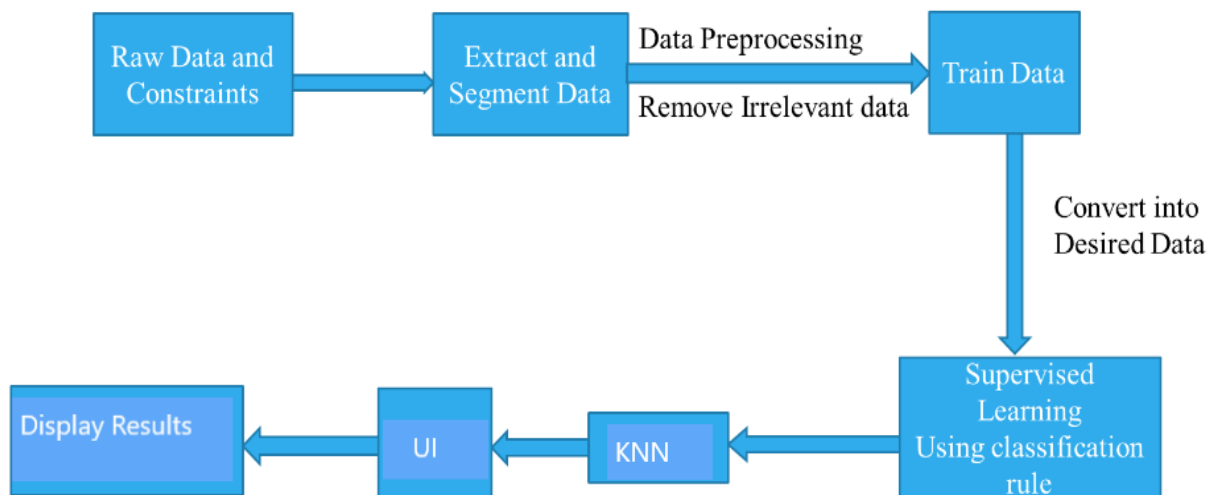
An architectural diagram is a diagram of a system that is used to abstract the overall outline of the software system and the relationships, constraints, and boundaries between components. It is an important tool as it provides an overall view of the physical deployment of the software system and its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



**Fig b: System Architecture**

### 6.2 PROCESS-FLOW DIAGRAM

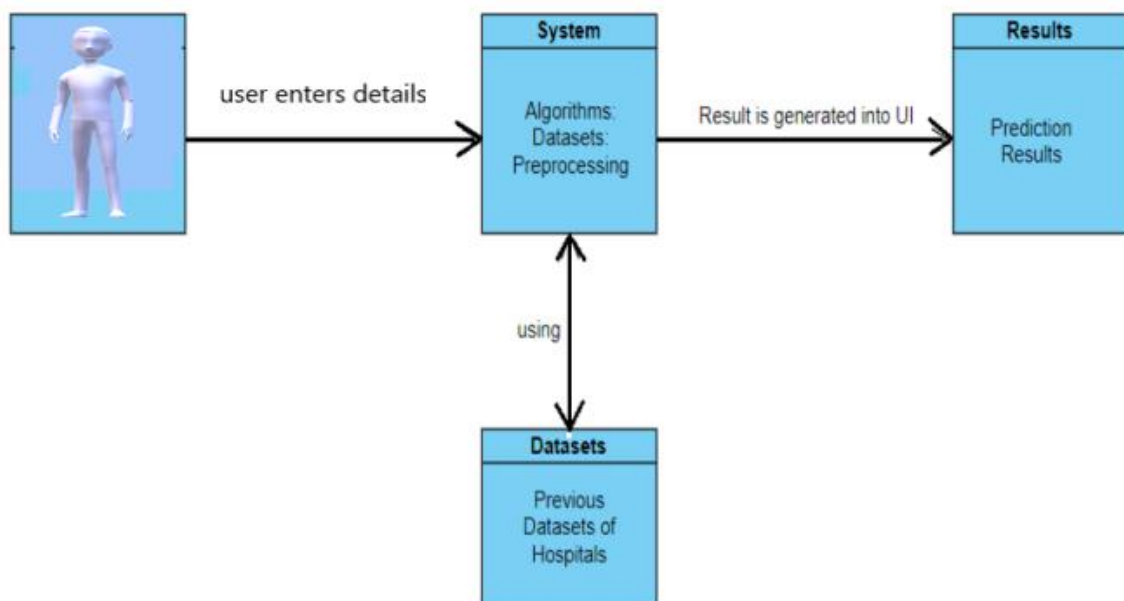
Process flows (or process diagrams) provide a visual overview or workflow diagram of all the tasks and relationships involved in a process and explains how a system works. The process flow diagram can greatly improve any process improvement project by increasing understanding the flow of information,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



**Fig c: Process Flow Diagram**

### 6.3 CLASS DIAGRAM

A class diagram is the basic building block of object oriented modelling. It describes the structure of a system in terms of its classes, attributes, operations and the relationship among the objects. The figure below depicts the class diagram of our model.



**Fig d: 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.

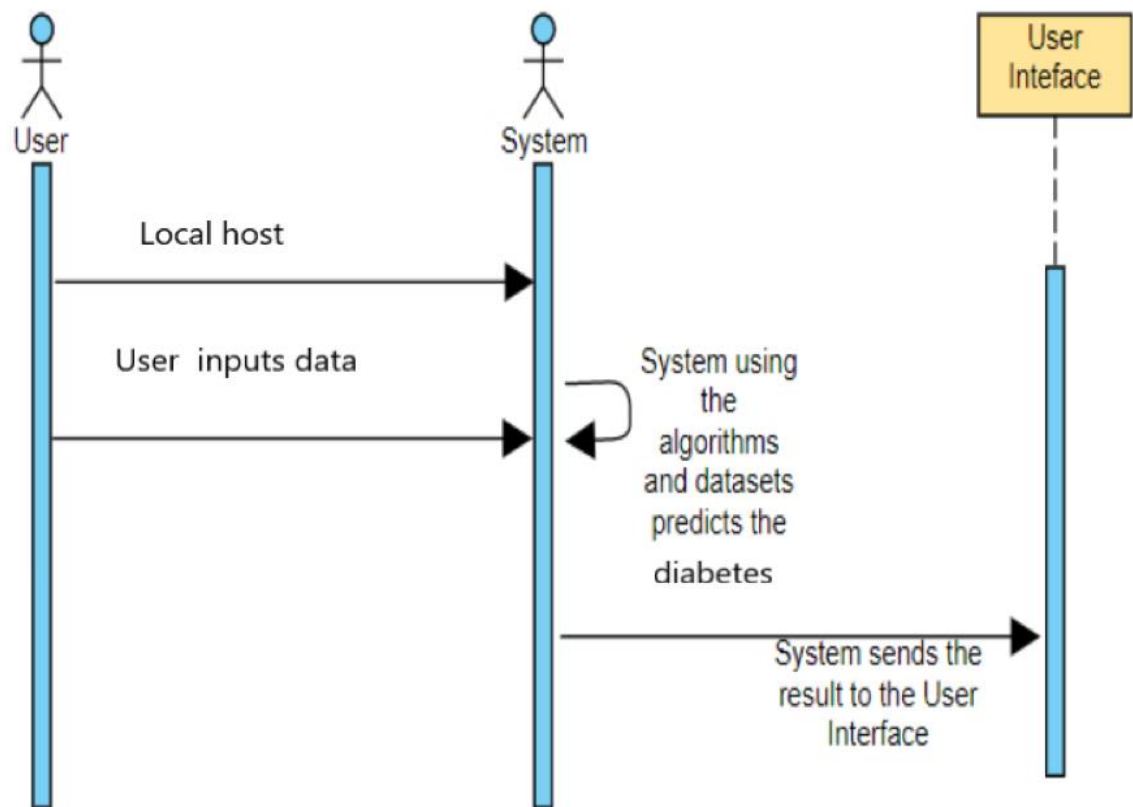
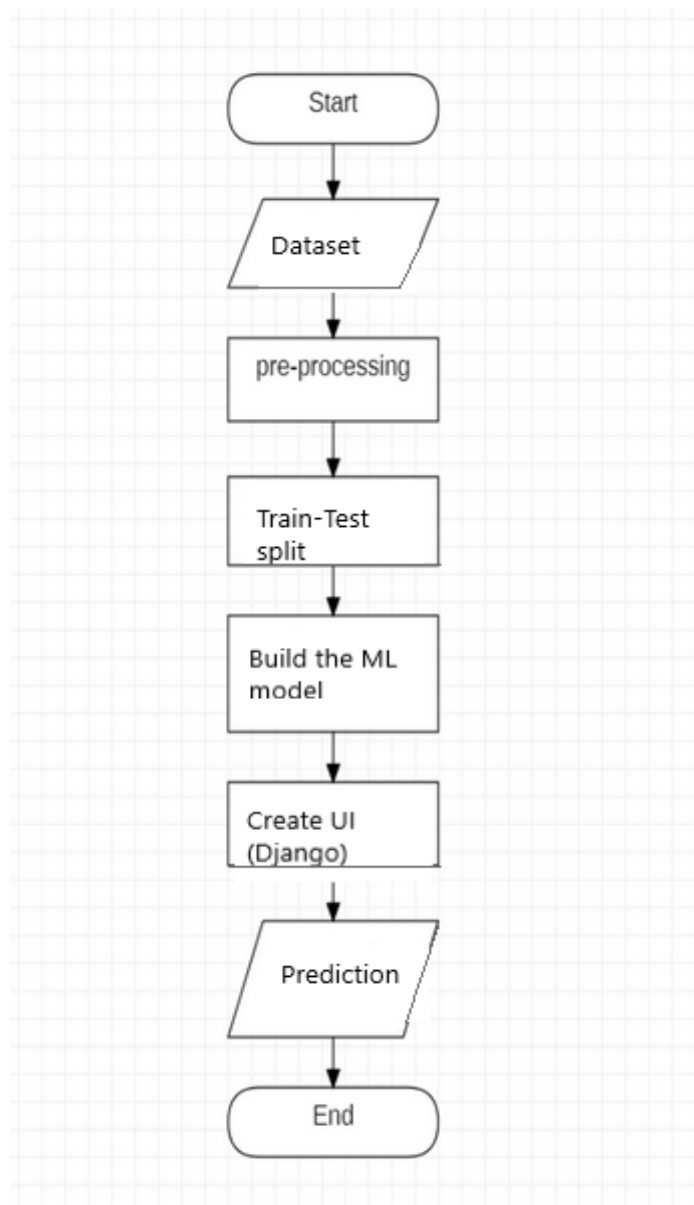


Fig e: Sequence Diagram

## 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 f: 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="let's get started">
</form>
```

#### **#Predict.html**

```
<h1>Please enter the following information: </h1>
<form action="result">
    <table>
        <tr>
            <td align="right">Various attributes: </td>
            <td align="left"><input type="text" name="n1"></td>
        </tr>
    </table>
    <input type="submit">
```

```

        </form>
        Result: {{result2}}
#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 })

```

The above code snippets are just an overview of developing an user interface.

## 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


### 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 × 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	EMI	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:

#### 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)

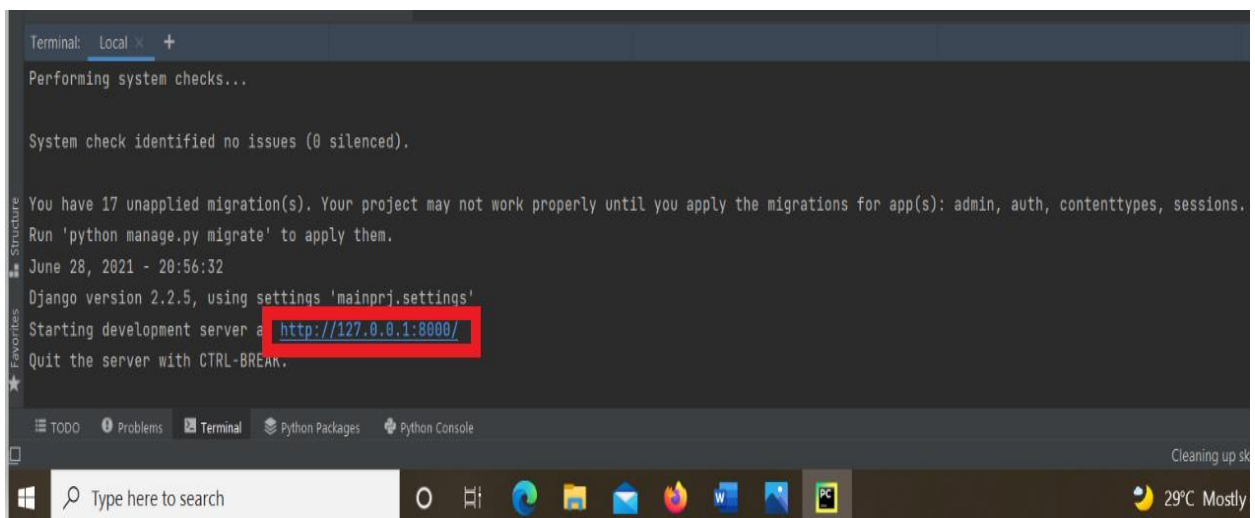
y_pred = knn.predict(X_test)

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

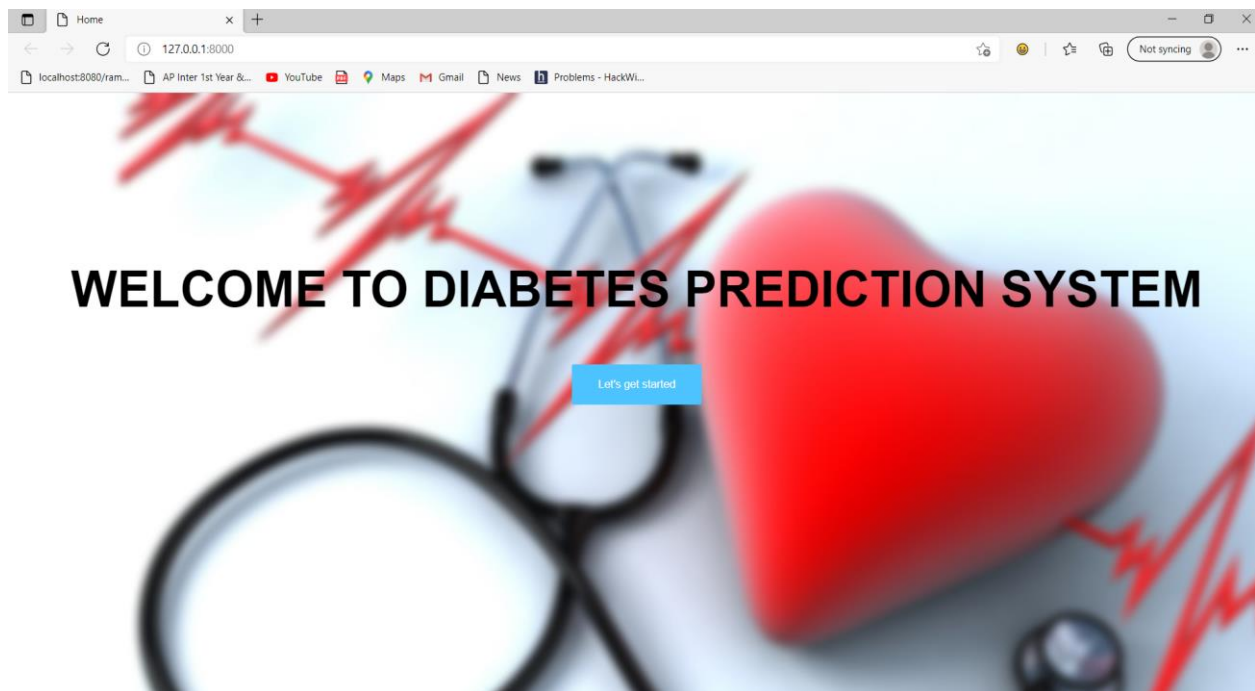
print(f"Training Accuracy of KNN Model is {knn_train_acc}")
print(f"Test Accuracy of KNN Model is {knn_test_acc}")
```

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

### Launching the Local Host, with Django:



## HOME\_PAGE of Diabetes Prediction System:



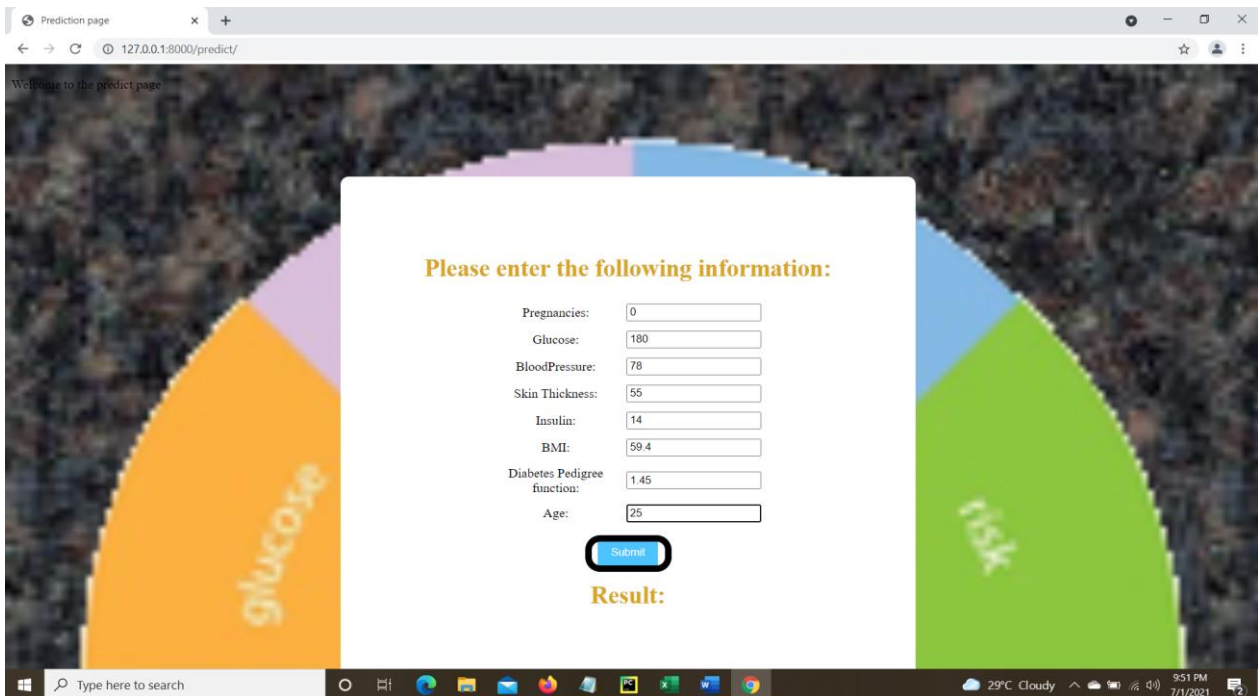
## PREDICT\_PAGE of Diabetes Prediction System:

A screenshot of a web browser displaying the predict page of a Diabetes Prediction System. The browser's address bar shows the URL "127.0.0.1:8000/predict/". The page has a dark, textured background with a large, colorful circular graphic on the left side. The graphic is divided into four segments: orange (labeled "glucose"), purple, blue, and green (labeled "risk"). In the center of the page is a white rectangular form with the heading "Please enter the following information:". Below the heading are several input fields for the following information: Pregnancies, Glucose, Bloodpressure, Skin Thickness, Insulin, BMI, Diabetes Pedigree function, and Age. A blue "Submit" button is located below the input fields. Below the "Submit" button is the text "Result:". A "Restore pages" dialog box is visible in the top right corner of the browser window, indicating that Microsoft Edge closed while some pages were open.

## 9. SAMPLE TEST CASES

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

**Case 1: When a diabetic person input the values, the system has to predict “POSITIVE”**



Prediction page

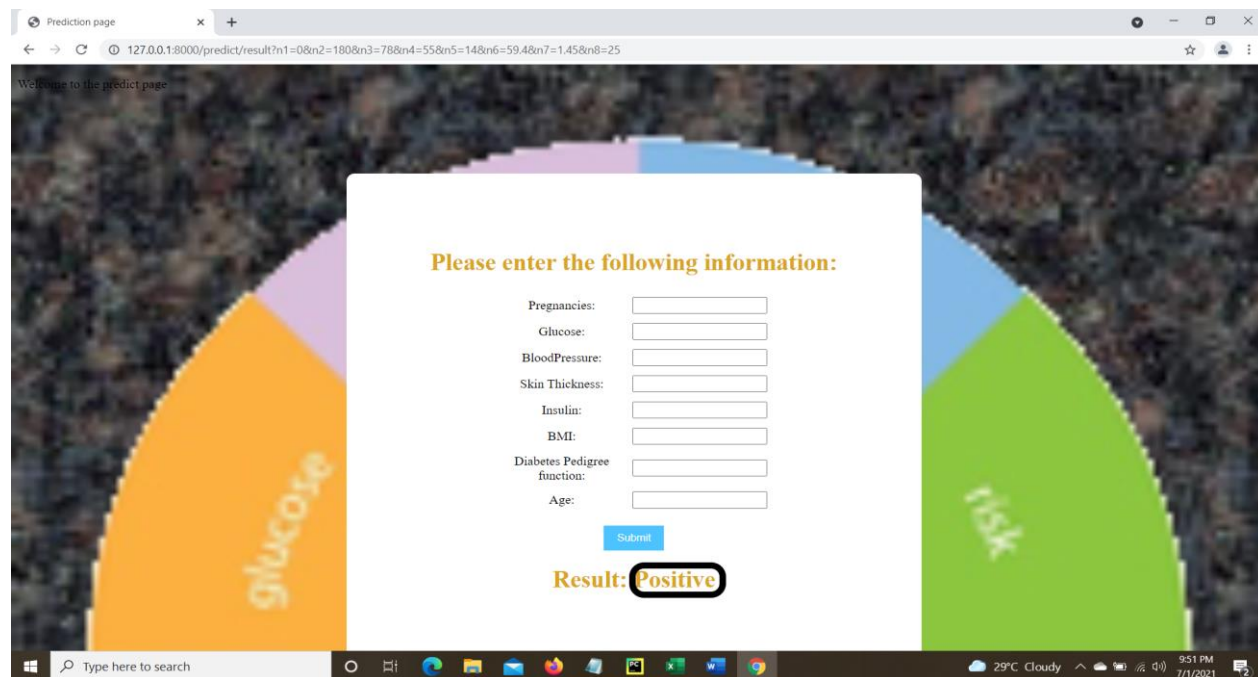
127.0.0.1:8000/predict/

Welcome to the predict page

Please enter the following information:

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:



Prediction page

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

Welcome to the predict page

Please enter the following information:

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**



## Case 2: When a non- diabetic person input the values, the system has to predict “NEGATIVE”

Prediction page

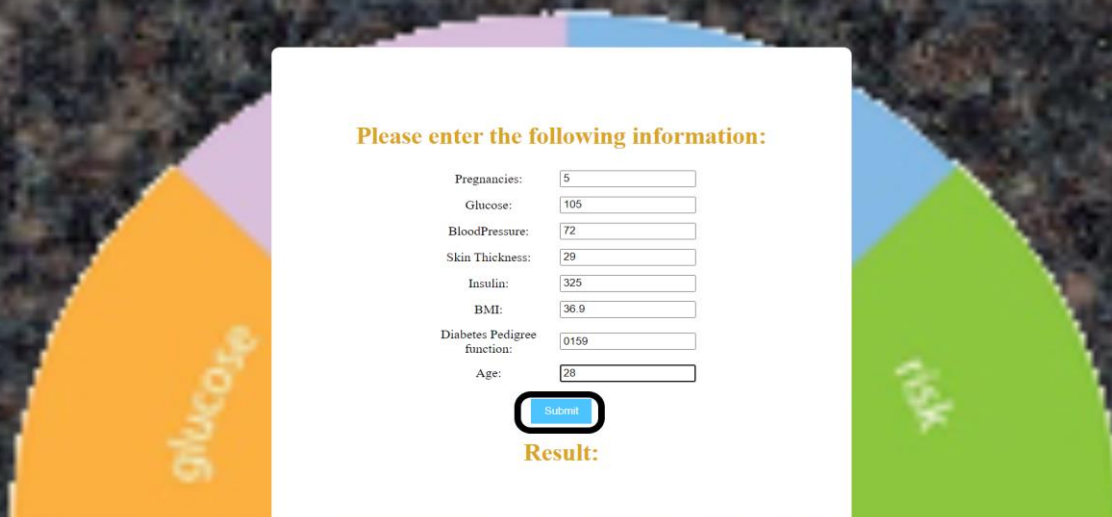
127.0.0.1:8000/predict/

Welcome to the predict page

Please enter the following information:

Pregnancies:	<input type="text" value="5"/>
Glucose:	<input type="text" value="105"/>
BloodPressure:	<input type="text" value="72"/>
Skin Thickness:	<input type="text" value="29"/>
Insulin:	<input type="text" value="325"/>
BMI:	<input type="text" value="36.9"/>
Diabetes Pedigree function:	<input type="text" value="0.159"/>
Age:	<input type="text" value="28"/>

Result:



Prediction page

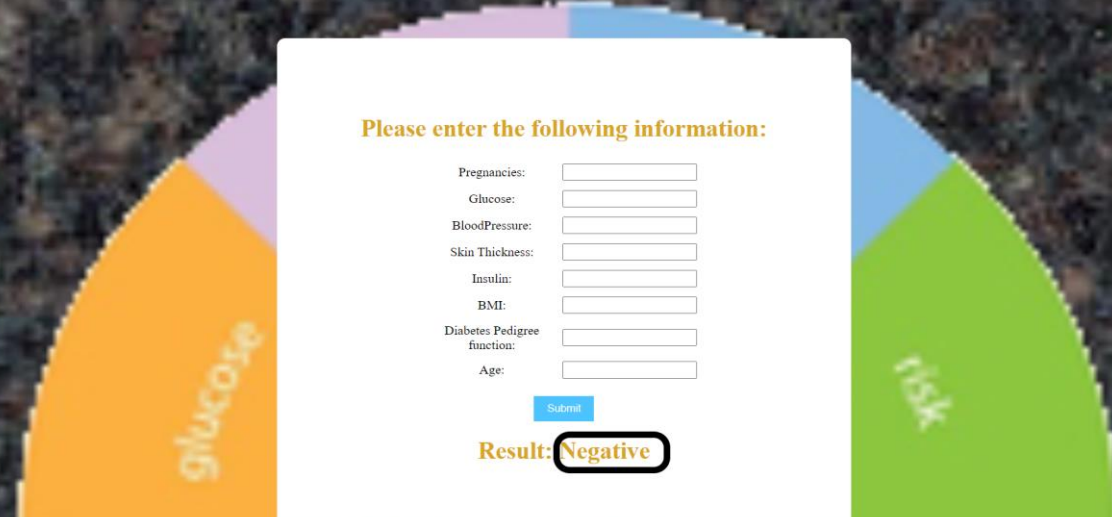
127.0.0.1:8000/predict/result?n1=5&n2=105&n3=72&n4=29&n5=325&n6=36.9&n7=0.159&n8=28

Welcome to the predict page

Please enter the following information:

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: **Negative**



## **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. 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>