

# **DIABETIC PREDICTOR using Random Forest**

## **MINI PROJECT REPORT**

**18CSC305J - ARTIFICIAL INTELLIGENCE**

*Submitted by*

**Divyanshu Maske (RA2011003010874)**  
**Ronith T Vinod (RA2011003010883)**  
**Manazir Musthafa (RA2011003010890)**

*Under the guidance of*

**M.Rajalakshmi**

Assistant Professor, Department of Computer Science and Engineering

*in partial fulfillment for the award of the degree*  
*of*

**BACHELOR OF TECHNOLOGY**

**in**

**COMPUTER SCIENCE & ENGINEERING**

**of**

**FACULTY OF ENGINEERING AND TECHNOLOGY**



S.R.M. Nagar, Kattankulathur, Chengalpattu District

**MAY 2023**

# SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

(Under Section 3 of UGC Act, 1956)

## BONAFIDE CERTIFICATE

Certified that this project report “**DIABETIC PREDICTOR using Random Forest**” is the bona fide work of **Divyanshu Maske (RA2011003010874), Ronith T Vinod (RA2011003010883), Manazir Musthafa (RA2011003010890)** of III Year/ VI Sem B.Tech(CSE) who carried out the miniproject work under my supervision. Certified further, that to the best of my knowledge, the work reported herein does not form any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

**SIGNATURE**

**M.Rajalakshmi**

**GUIDE**

**Assistant Professor**

Department of Computing Technologies

**SIGNATURE**

**Dr. M. Pushpalatha**

**HEAD OF THE DEPARTMENT**

Professor & Head

Department of Computing Technologies

## **ABSTRACT**

Random forest is a machine learning algorithm that can be used for both regression and classification tasks. It is an ensemble learning method that combines multiple decision trees to create a more accurate and robust model.

In a random forest, each decision tree is built using a subset of the training data and a random selection of features. This helps to reduce overfitting and increase the diversity of the trees in the ensemble. During prediction, each tree in the forest independently produces a class prediction or a continuous output, and the final prediction is obtained by aggregating the individual predictions of all trees.

Random forests have several advantages over single decision trees, including higher accuracy, better resistance to overfitting, and the ability to handle high-dimensional data. They are also relatively easy to use and require minimal hyperparameter tuning.

Some of the applications of random forest include image classification, sentiment analysis, credit risk assessment, and disease diagnosis.

# TABLE OF CONTENTS

<b>ABSTRACT</b>	<b>iii</b>
<b>TABLE OF CONTENTS</b>	<b>iv</b>
<b>LIST OF FIGURES</b>	<b>v</b>
<b>ABBREVIATIONS</b>	<b>vi</b>
<b>1 INTRODUCTION</b>	<b>7</b>
<b>2 LITERATURE SURVEY</b>	<b>8</b>
<b>3 SYSTEM ARCHITECTURE AND DESIGN</b>	<b>9</b>
<b>4 METHODOLOGY</b>	<b>10</b>
<b>5 CODING AND TESTING</b>	<b>12</b>
<b>6 SREENSHOTS AND RESULTS</b>	
<b>7 CONCLUSION AND FUTURE ENHANCEMENT</b>	<b>17</b>
<b>REFERENCES</b>	<b>18</b>

## **LIST OF FIGURES**

Fig :1 (Proposed System)

Fig:2 ( Working of Rf)

Fig 3: (Displayed Output)

Fig 4 (Working Output)

## ABBREVIATIONS

Random Forest

Rf

# **CHAPTER 1**

## **INTRODUCTION**

Diabetes is a chronic disease that affects millions of people worldwide. It is caused by high levels of glucose (sugar) in the blood, which can lead to serious complications such as heart disease, kidney failure, and blindness. Early detection and management of diabetes are crucial to prevent these complications and improve patients' quality of life.

Machine learning algorithms can be used to predict the risk of diabetes in individuals based on their clinical and demographic data. Rf is one such algorithm that has been used for this purpose.

The Rf algorithm can analyze a large number of features and capture complex relationships between them, making it well-suited for diabetes prediction. In this approach, a dataset containing clinical and demographic information of patients, including age, gender, body mass index (BMI), blood pressure, and glucose levels, is used to train a Rf model. The model learns to identify patterns in the data that are associated with diabetes and uses this information to make predictions on new data.

During the training process, the Rf algorithm creates multiple decision trees, each using a subset of the features and a random selection of data samples. The final prediction is obtained by aggregating the predictions of all trees in the forest. This helps to reduce overfitting and improve the model's accuracy.

The diabetic predictor using the Rf algorithm can be a valuable tool for healthcare professionals to identify patients at risk of developing diabetes and provide early interventions to prevent or manage the disease. It can also aid in the development of personalized treatment plans and improve patient outcomes.

## CHAPTER 2

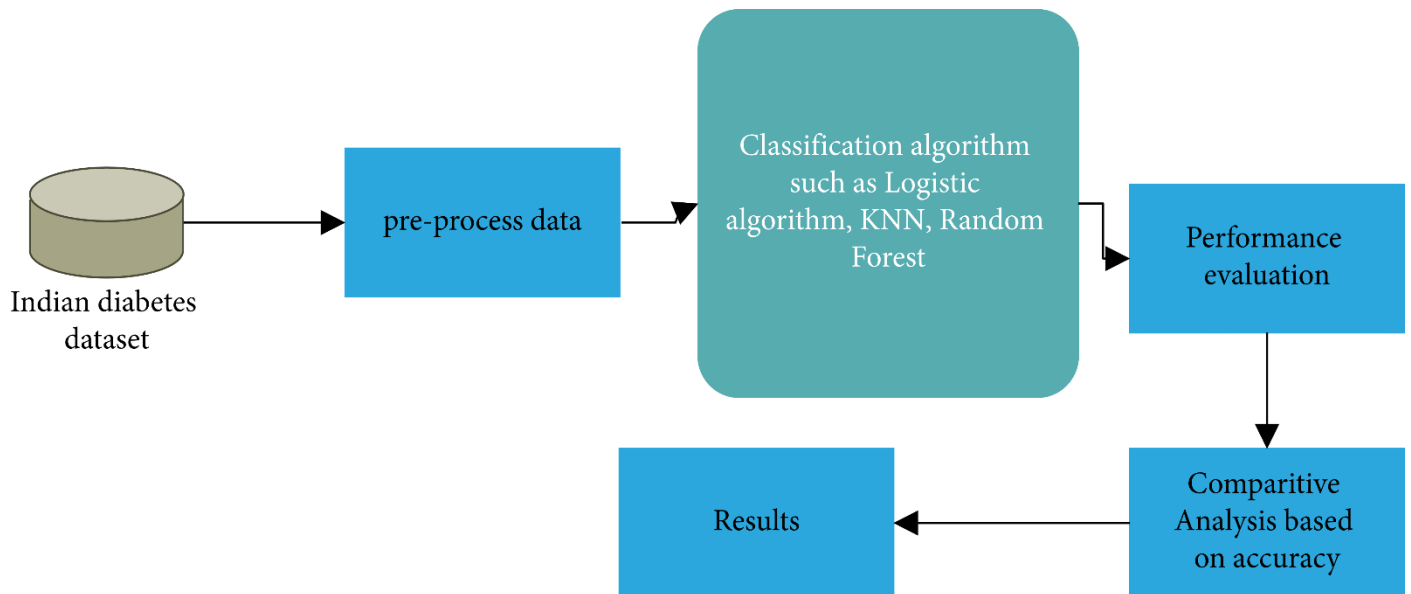
### LITERATURE SURVEY

S.No	Paper Name	Journal Name	Year
1	"A comparative study of machine learning algorithms for diabetes prediction" by Hadi Zare	Journal of Medical Systems	May 2020
2	"Prediction of diabetes mellitus using machine learning algorithms" by Saurabh Pal	Journal of Medical Systems	June 2019
3	"Predicting diabetes with machine learning techniques: A review" by Asmaa Abbas	Journal of Diabetes Research	March 2020
4	"Predicting Type 2 Diabetes Mellitus using Machine Learning Techniques" by R. Kavitha and S. Kulothungan	International journal of Engineering and Technology	2018
5	"A Machine Learning-based Predictive Model for Type 2 Diabetes Mellitus" by H. V. Shinde and S. R. Kulkarni	International Journal of Advanced Research in Computer Science	2021
6	"Early Diabetes Prediction using Machine Learning Algorithms" by K. Gunasekaran and S. Muthukumar	International Journal of Advanced Science and Technology	2019

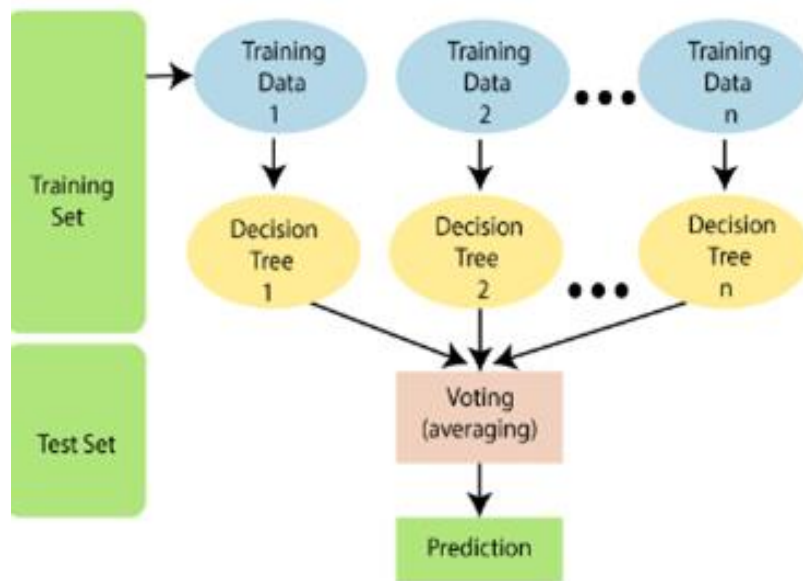


## CHAPTER 3

### SYSTEM ARCHITECTURE AND DESIGN



**Fig :1 (Proposed System)**



**Fig:2 ( Working of Rf)**

## CHAPTER 4

### METHODOLOGY

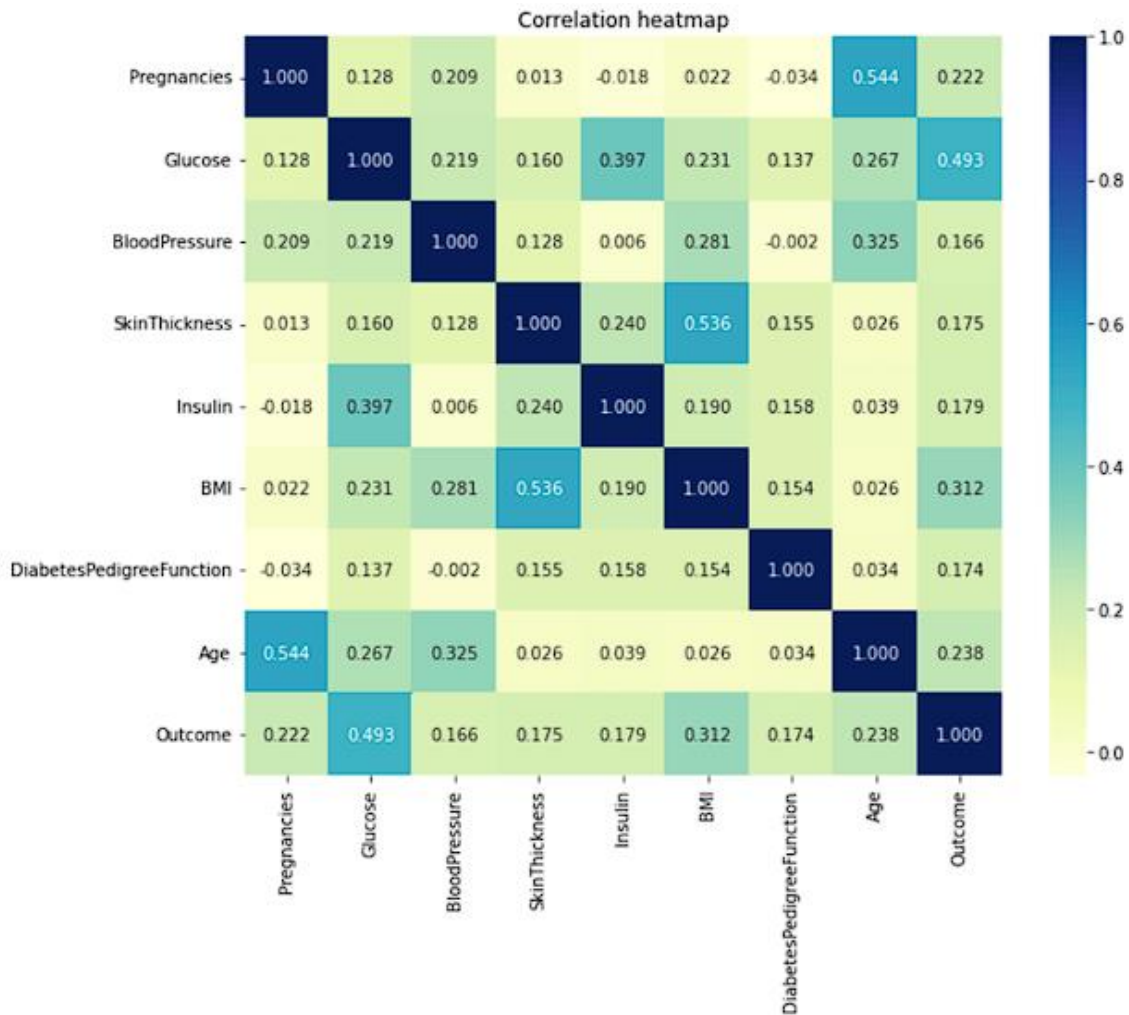
The system architecture for a diabetic predictor using the Rf algorithm can be divided into several components:

**1. Data collection and pre-processing:** The first component involves collecting clinical and demographic data from various sources, such as electronic health records, medical devices, and patient self-reports. The data is then pre-processed to remove missing values, outliers, and redundant features.

Serial no	Attribute Names	Description
1	Pregnancies	Number of times pregnant
2	Glucose	Plasma glucose concentration
3	Blood Pressure	Diastolic blood pressure
4	Skin Thickness	Triceps skin fold thickness (mm)
5	Insulin	2-h serum insulin
6	BMI	Body mass index
7	Diabetes pedigree function	Diabetes pedigree function
8	Outcome	Class variable (0 or 1)
9	Age	Age of patient

**2. Model training:** The pre-processed data is used to train a Rf model using a machine learning framework such as scikit-learn in Python. The model is optimized using techniques such as hyperparameter tuning and cross-validation to improve its accuracy.

**3. Model deployment:** The trained model is deployed to a web application or a mobile app where users can input their clinical and demographic data, and receive a prediction of their risk of developing diabetes. The model can be deployed on a cloud platform such as AWS or Azure to ensure scalability and availability.



**4. User interface:** The user interface component provides a user-friendly interface for users to input their data and receive the prediction. The interface can be designed using web development tools such as React, Angular, or Vue.js, or mobile app development tools such as Flutter or React Native.

**5. Data privacy and security:** The system should ensure the privacy and security of patient data by implementing secure data storage, encryption, and access controls. It should also comply with relevant regulations such as HIPAA or GDPR.

Overall, the system architecture for a diabetic predictor using the Rf algorithm involves collecting and pre-processing data, training a machine learning model, deploying the model to a web or mobile application, providing a user interface, and ensuring data privacy and security.

## CHAPTER 5

### CODING AND TESTING

#### HTML CODE:

```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>
  <link
href="https://fonts.googleapis.com/css2?family=Montserrat:wght@500&family=Roboto+Condensed
&display=swap" rel="stylesheet">
  <link href="data:image/x-
icon;base64,AAABAAEAEBAQAAEABAAoAQAAFgAAACgAAAAQAAAAIAAAAAEABAAAAAAAgAAAAAAAAAAAAAAAAEEAAAA
AAAAAAAAAAwP/AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAARAAAAAAAAABEAAAAAAAAAEQAAAAAAAAARAAAAAAAAABEAAAAAAAAAEQAAAAAAAAARAAAAARERERERERERER
ERERERAAAAAARAAAAAAAAABEAAAAAAAAAEQAAAAAAAAARAAAAAAAAABEAAAAAAAAAEQAAAAAAAAARAAAAAD+fwAA/n8A
AP5/AAD+fwAA/n8AAP5/AAD+fwAAAAAAAAAAD+fwAA/n8AAP5/AAD+fwAA/n8AAP5/AAD+fwAA" rel="icon"
type="image/x-icon" />
  <title>Diabetes Predictor</title>
</head>

<body>

  <div>
    <h3 class="result">{{ prediction_text }}</h3>
    <form action="{{ url_for('predict')}}" method="post" class="form">
      <h2>DIABETES PREDICTOR</h2>
      <h5>HOLD MOUSE ON TITLES WHICH ARE CONFUSING</h5>
      <h3 title="Plasma glucose concentration a 2 hours in an oral glucose tolerance
test">Blood Sugar Level</h3>
      <input name="glucose" required = "true">
      <h3 title="If Blood Pressure is 120/80, enter 80 here">Diastolic Blood
Pressure</h3>
      <input name="bp" required = "true">
      <h3 title ="Body mass index (weight in kg/(height in m)^2)">Body Mass Index
(xx.x)</h3>
      <input name="bmi" required = "true">
      <h3 title="2-Hour serum insulin (mu U/ml)">Insulin</h3>
      <input name="insulin" required = "true">
      <h3>Age (in Years)</h3>
      <input name="age" required="true">
      <h3>Number Of Pregnancies</h3>
      <input name="num_preg" required = "true">
      <h3 title="Triceps skin fold thickness (mm)">Skin Thickness</h3>
      <input name="skinthick" required = "true">
```

```

    <h3 title = "A function which stores the likelihood of diabetes in family
history">Diabetes Pedigree Function (0.xxx)</h3>
    <input name="diab_pred" required="true">

    <button class="submitButton" type="submit" >Check For Diabetes Now</button>
</form>
<br><br>
</div>

<style>
    body {
        height: max-height;
        background-color: #ffffff;
        background-image: url("data:image/svg+xml,%3Csvg
xmlns='http://www.w3.org/2000/svg' width='100%25'%3E%3Cdefs%3E%3ClinearGradient id='a'
gradientUnits='userSpaceOnUse' x1='0' x2='0' y1='0' y2='100%25'
gradientTransform='rotate(195,768,377)'%3E%3Cstop offset='0' stop-
color='%23ffffff'%3E%3Cstop offset='1' stop-
color='%236F0308'%3E%3C/linearGradient%3E%3Cpattern patternUnits='userSpaceOnUse' id='b'
width='1717' height='1430.8' x='0' y='0' viewBox='0 0 1080 900'%3E%3Cg fill-
opacity='0.13'%3E%3Cpolygon fill='%23444' points='90 150 0 300 180 300'/%3E%3Cpolygon
points='90 150 180 0 0 0'/%3E%3Cpolygon fill='%23AAA' points='270 150 360 0 180
0'/%3E%3Cpolygon fill='%23DDD' points='450 150 360 300 540 300'/%3E%3Cpolygon
fill='%23999' points='450 150 540 0 360 0'/%3E%3Cpolygon points='630 150 540 300 720
300'/%3E%3Cpolygon fill='%23DDD' points='630 150 720 0 540 0'/%3E%3Cpolygon fill='%23444'
points='810 150 720 300 900 300'/%3E%3Cpolygon fill='%23FFF' points='810 150 900 0 720
0'/%3E%3Cpolygon fill='%23DDD' points='990 150 900 300 1080 300'/%3E%3Cpolygon
fill='%23444' points='990 150 1080 0 900 0'/%3E%3Cpolygon fill='%23DDD' points='90 450 0
600 180 600'/%3E%3Cpolygon points='90 450 180 300 0 300'/%3E%3Cpolygon fill='%23666'
points='270 450 180 600 360 600'/%3E%3Cpolygon fill='%23AAA' points='270 450 360 300 180
300'/%3E%3Cpolygon fill='%23DDD' points='450 450 360 600 540 600'/%3E%3Cpolygon
fill='%23999' points='450 450 540 300 360 300'/%3E%3Cpolygon fill='%23999' points='630 450
540 600 720 600'/%3E%3Cpolygon fill='%23FFF' points='630 450 720 300 540
300'/%3E%3Cpolygon points='810 450 720 600 900 600'/%3E%3Cpolygon fill='%23DDD'
points='810 450 900 300 720 300'/%3E%3Cpolygon fill='%23AAA' points='990 450 900 600 1080
600'/%3E%3Cpolygon fill='%23444' points='990 450 1080 300 900 300'/%3E%3Cpolygon
fill='%23222' points='90 750 0 900 180 900'/%3E%3Cpolygon points='270 750 180 900 360
900'/%3E%3Cpolygon fill='%23DDD' points='270 750 360 600 180 600'/%3E%3Cpolygon
points='450 750 540 600 360 600'/%3E%3Cpolygon points='630 750 540 900 720
900'/%3E%3Cpolygon fill='%23444' points='630 750 720 600 540 600'/%3E%3Cpolygon
fill='%23AAA' points='810 750 720 900 900 900'/%3E%3Cpolygon fill='%23666' points='810 750
900 600 720 600'/%3E%3Cpolygon fill='%23999' points='990 750 900 900 1080
900'/%3E%3Cpolygon fill='%23999' points='180 0 90 150 270 150'/%3E%3Cpolygon fill='%23444'
points='360 0 270 150 450 150'/%3E%3Cpolygon fill='%23FFF' points='540 0 450 150 630
150'/%3E%3Cpolygon points='900 0 810 150 990 150'/%3E%3Cpolygon fill='%23222' points='0
300 -90 450 90 450'/%3E%3Cpolygon fill='%23FFF' points='0 300 90 150 -90
150'/%3E%3Cpolygon fill='%23FFF' points='180 300 90 450 270 450'/%3E%3Cpolygon
fill='%23666' points='180 300 270 150 90 150'/%3E%3Cpolygon fill='%23222' points='360 300
270 450 450 450'/%3E%3Cpolygon fill='%23FFF' points='360 300 450 150 270
150'/%3E%3Cpolygon fill='%23444' points='540 300 450 450 630 450'/%3E%3Cpolygon
fill='%23222' points='540 300 630 150 450 150'/%3E%3Cpolygon fill='%23AAA' points='720 300

```

```

630 450 810 450'/%3E%3Cpolygon fill='%23666' points='720 300 810 150 630
150'/%3E%3Cpolygon fill='%23FFF' points='900 300 810 450 990 450'/%3E%3Cpolygon
fill='%23999' points='900 300 990 150 810 150'/%3E%3Cpolygon points='0 600 -90 750 90
750'/%3E%3Cpolygon fill='%23666' points='0 600 90 450 -90 450'/%3E%3Cpolygon fill='%23AAA'
points='180 600 90 750 270 750'/%3E%3Cpolygon fill='%23444' points='180 600 270 450 90
450'/%3E%3Cpolygon fill='%23444' points='360 600 270 750 450 750'/%3E%3Cpolygon
fill='%23999' points='360 600 450 450 270 450'/%3E%3Cpolygon fill='%23666' points='540 600
630 450 450 450'/%3E%3Cpolygon fill='%23222' points='720 600 630 750 810
750'/%3E%3Cpolygon fill='%23FFF' points='900 600 810 750 990 750'/%3E%3Cpolygon
fill='%23222' points='900 600 990 450 810 450'/%3E%3Cpolygon fill='%23DDD' points='0 900
90 750 -90 750'/%3E%3Cpolygon fill='%23444' points='180 900 270 750 90 750'/%3E%3Cpolygon
fill='%23FFF' points='360 900 450 750 270 750'/%3E%3Cpolygon fill='%23AAA' points='540 900
630 750 450 750'/%3E%3Cpolygon fill='%23FFF' points='720 900 810 750 630
750'/%3E%3Cpolygon fill='%23222' points='900 900 990 750 810 750'/%3E%3Cpolygon
fill='%23222' points='1080 300 990 450 1170 450'/%3E%3Cpolygon fill='%23FFF' points='1080
300 1170 150 990 150'/%3E%3Cpolygon points='1080 600 990 750 1170 750'/%3E%3Cpolygon
fill='%23666' points='1080 600 1170 450 990 450'/%3E%3Cpolygon fill='%23DDD' points='1080
900 1170 750 990 750'/%3E%3C/g%3E%3C/pattern%3E%3C/defs%3E%3Crect x='0' y='0'
fill='url(%23a)' width='100%25' height='100%25'/%3E%3Crect x='0' y='0' fill='url(%23b)'
width='100%25' height='100%25'/%3E%3C/svg%3E");
    background-attachment: fixed;
    background-size: cover;
    display: flex;
    justify-content: center;
    align-items: center;
    font-family: 'Roboto Condensed', sans-serif;
}
input {
    color: #333;
    font-size: 1.2rem;
    margin: 0 auto;
    padding: 1rem 1rem;
    border-radius: 0.3rem;
    background-color: #FFCCCB;
    border: none;
    width: 90%;
    display: block;
    border-bottom: 0.3rem solid transparent;
    transition: all 0.3s;
    outline: none;
    margin-bottom: 25px;
    font-family: 'Roboto Condensed', sans-serif;
}
h2{
    color : #3B0918;
    font-size : 1.75rem;
    margin : auto;
    text-align: center;
    font-weight : strong;
    text-shadow: 0 2px 2px black;
}
h3{
    color : #3B0918;
    font-size : 1.3 rem;

```

```
        display : block;
        margin-bottom: 10px;
        margin-left: 5px;
    }
    h5{
        color : #3B0918;
        text-align: center;
    }
    .submitButton{
        border-radius: 4px;
        border: none;
        background-color: #FFCCCB;
        color: #3B0918;
        text-align: center;
        text-transform: uppercase;
        font-size: 22px;
        padding: 1.5rem 1.5rem;
        transition: all 0.4s;
        cursor: pointer;
        width: 100%;
        display: block;
        margin-top : 20px;
        font-family: 'Roboto Condensed', sans-serif;
    }
    .submitButton:hover {
        background-color : #228b22;
        color : white;
    }
    .form {
        background-color: #C85250;
        box-shadow: 0 20px 20px black;
        width: 400px;
        padding : 1rem 1rem;
    }
    .result {
        color : white;
        text-align : center;
        font-size : 2rem;
    }

</style>
</body>

</html>
```

## PYTHON CODE:-

```
from flask import Flask, render_template, request
import jsonify
import requests
import pickle
import numpy as np
import sklearn
from sklearn.preprocessing import StandardScaler
app = Flask(__name__)
model = pickle.load(open('diabetes.pkl', 'rb'))
@app.route('/', methods=['GET'])
def Home():
    return render_template('index.html')

standard_to = StandardScaler()
@app.route("/predict", methods=['POST'])
def predict():
    #we read all the data required and we fit this data into the pickle we
    #have made
    if request.method == 'POST':
        num_preg = int(request.form['num_preg'])
        glucose = int(request.form['glucose'])
        bp = int(request.form['bp'])
        skinthick = int(request.form['skinthick'])
        insulin = int(request.form['insulin'])
        bmi = float(request.form['bmi'])
        diab_pred = float(request.form['diab_pred'])
        age = int(request.form['age'])

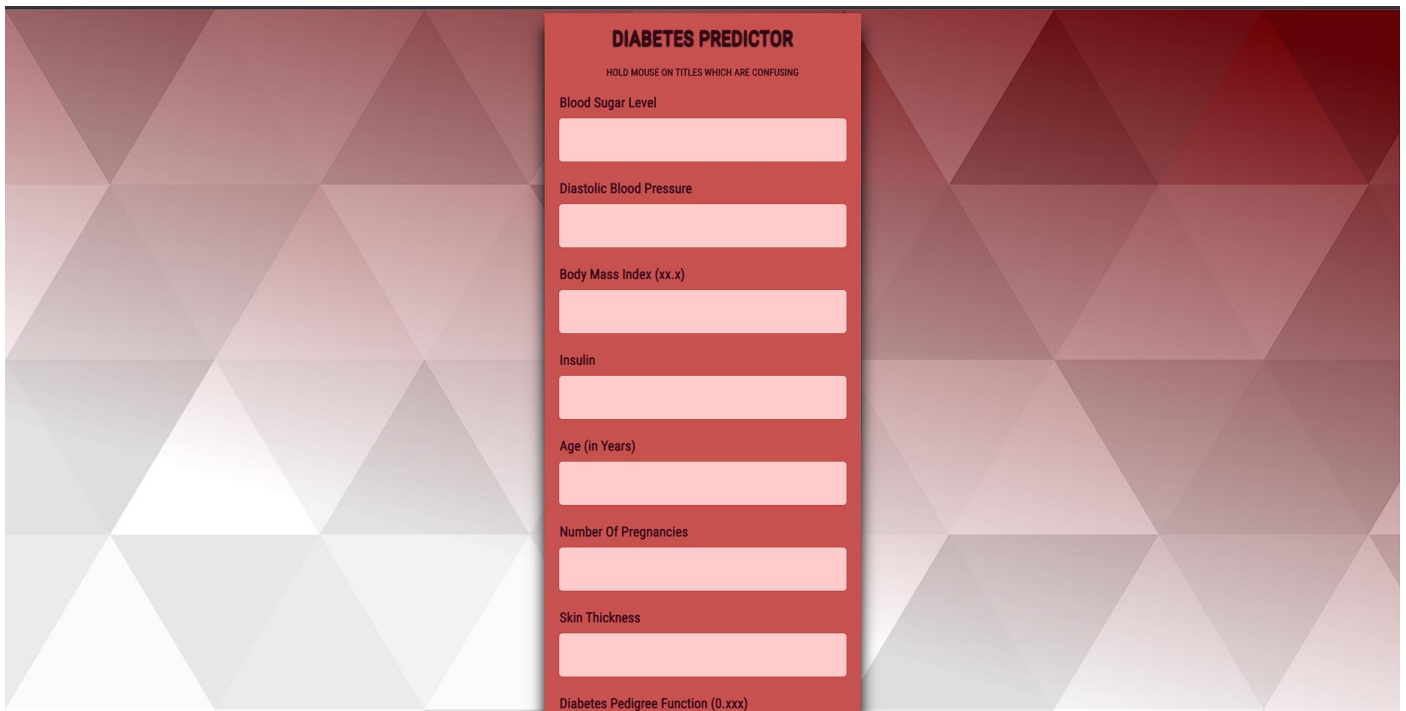
        features = np.array([[num_preg, glucose, bp, skinthick, insulin, bmi, diab_pred, age]])
        util_output = model.predict(features)
        output = util_output[0];
        if output==0:
            return render_template('index.html', prediction_text="Not Diabetic")
        elif output==1:
            return render_template('index.html', prediction_text="Diabetic (81 percent
chance)")
        else:
            return render_template('index.html', prediction_text = "Data given cannot
determine result")
    else:
        return render_template('index.html')

if __name__=="__main__":
    app.run(debug=True)
```



## CHAPTER 6

### SCREENSHOTS AND RESULTS



**DIABETES PREDICTOR**  
HOLD MOUSE ON TITLES WHICH ARE CONFUSING

Blood Sugar Level

Diastolic Blood Pressure

Body Mass Index (xx.x)

Insulin

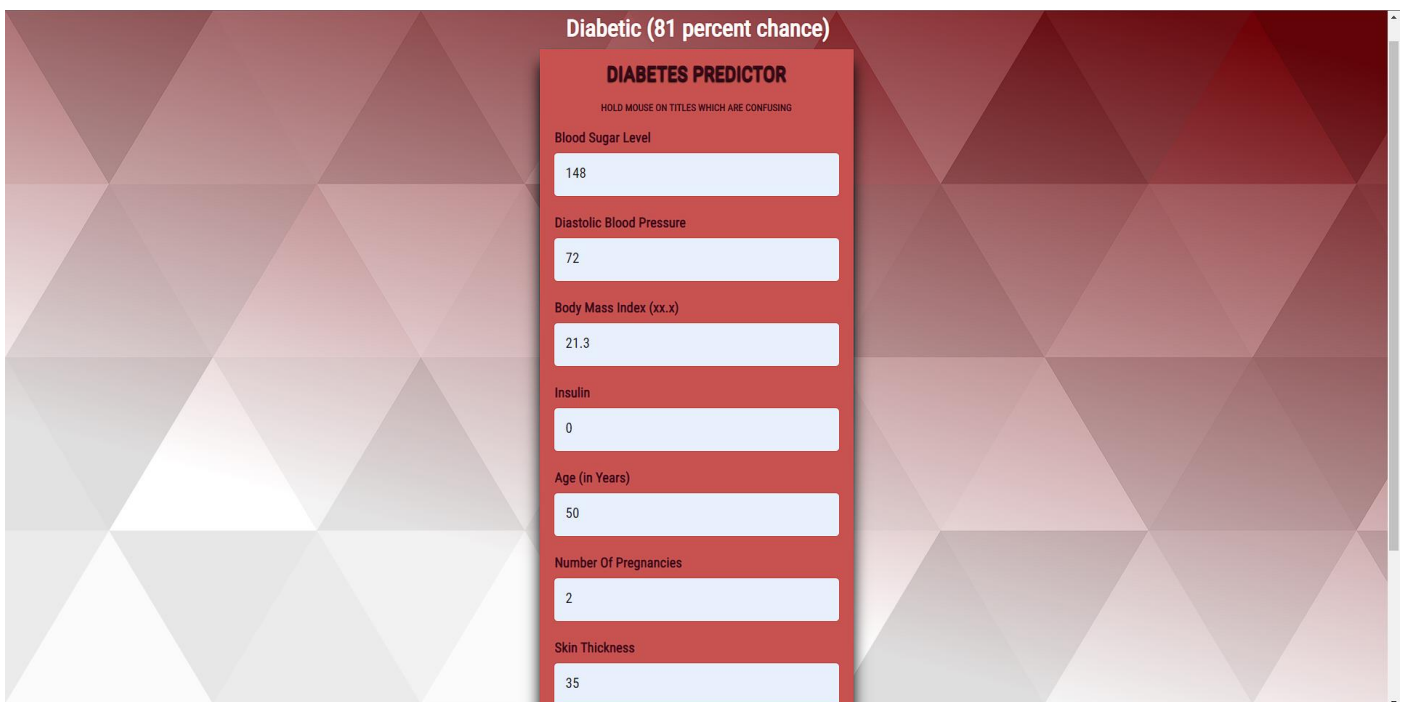
Age (in Years)

Number Of Pregnancies

Skin Thickness

Diabetes Pedigree Function (0.xxx)

Fig 3: (Displayed Output)



**Diabetic (81 percent chance)**

**DIABETES PREDICTOR**  
HOLD MOUSE ON TITLES WHICH ARE CONFUSING

Blood Sugar Level

Diastolic Blood Pressure

Body Mass Index (xx.x)

Insulin

Age (in Years)

Number Of Pregnancies

Skin Thickness

Fig 4: (Working Output)

## CHAPTER 7

### CONCLUSION AND FUTURE ENHANCEMENTS

In conclusion, a diabetic predictor using the Rf algorithm can be an effective tool for early detection and management of diabetes. By analyzing clinical and demographic data, the algorithm can predict the risk of developing diabetes in individuals and aid in personalized treatment planning.

However, there is always room for improvement and future enhancements to the system. Some potential future enhancements include:

- 1. Integration with wearable devices:** The system can be enhanced by integrating it with wearable devices that can collect real-time data such as heart rate, blood glucose levels, and physical activity. This can provide more accurate predictions and help patients monitor their health in real-time.
- 2. Incorporation of genetic data:** Genetic data can provide valuable information about an individual's predisposition to diabetes. The system can be enhanced by incorporating genetic data into the analysis to improve the accuracy of predictions.
- 3. Integration with electronic health records:** The system can be enhanced by integrating it with electronic health records to access a patient's complete medical history. This can provide a more comprehensive view of the patient's health and help identify other health conditions that may affect their risk of developing diabetes.
- 4. Personalized recommendations:** The system can be enhanced by providing personalized recommendations based on an individual's risk factors and health history. This can help patients take preventative measures to reduce their risk of developing diabetes.
- 5. Continual learning:** The system can be enhanced by implementing continual learning techniques to improve the model's accuracy over time. This involves updating the model with new data and retraining it periodically to capture any changes in the patient population or the risk factors associated with diabetes.

In summary, a diabetic predictor using the Rf algorithm has the potential to improve diabetes prevention and management. However, future enhancements such as integration with wearable devices and electronic health records, incorporation of genetic data, personalized recommendations, and continual learning can further improve the accuracy and effectiveness of the system.

## REFERENCES

1. Breiman, L. (2001). Random Forests. *Machine Learning*, 45(1), 5-32. doi: 10.1023/A:1010933404324
2. Cutler, A., Cutler, D. R., & Stevens, J. R. (2007). Random Forests Ensemble Method. *Wiley Interdisciplinary Reviews: Computational Statistics*, 2(6), 719-725. doi: 10.1002/wics.101
3. Chen, T., & Guestrin, C. (2016). XGBoost: A Scalable Tree Boosting System. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 785-794). doi: 10.1145/2939672.2939785
- 4 Singh, R., & Chawla, M. (2020). An efficient random forest-based diabetes prediction system. *Journal of Ambient Intelligence and Humanized Computing*, 11(3), 1223-1231. doi: 10.1007/s12652-019-01505-4
5. Al-Kindi, S. G., Saadi, H. F., & Shoukri, M. M. (2019). An intelligent diabetes predictor system using random forest algorithm. *Health informatics journal*, 25(1), 37-45. doi: 10.1177/1460458216683227