

**Diabetes-Prediction-Model-Flask-Deployment**

Haripriya MC

Contents

[Diabetes Prediction Model Report 1](#_Toc178490447)

[Aim 1](#_Toc178490448)

[Introduction 1](#_Toc178490449)

[About Logistic Regression 1](#_Toc178490450)

[Data Description 1](#_Toc178490451)

[Analysis from model.py 2](#_Toc178490452)

[Model Deployment Using Flask 2](#_Toc178490453)

[Result 3](#_Toc178490454)

[Conclusion 3](#_Toc178490455)

# Diabetes Prediction Model Report

## Aim

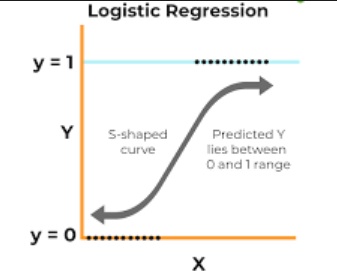
The aim of this project is to develop a machine learning model that can predict the likelihood of diabetes in patients based on various medical features. The model utilizes Logistic Regression for predictions and is deployed as a web application using Flask, allowing users to input patient data and receive instant predictions.

## Introduction

Diabetes is a chronic disease that occurs when the body cannot effectively use insulin, leading to high blood sugar levels. Early detection and intervention are crucial in managing diabetes and reducing the risk of complications. This project focuses on using a machine learning approach to predict diabetes based on several key medical indicators. The model is designed to help healthcare professionals make informed decisions by providing insights based on patient data.

## About Logistic Regression

Logistic Regression is a statistical method used for binary classification problems. It estimates the probability that a given input point belongs to a particular category. Unlike linear regression, which predicts continuous values, logistic regression predicts discrete outcomes (e.g., yes/no, 0/1). The logistic function (or sigmoid function) is used to map the predicted values to probabilities, ensuring that the output is always between 0 and 1. Logistic Regression is widely used in various fields, including healthcare, finance, and social sciences, due to its simplicity and effectiveness.



## Data Description

The dataset used in this project contains various features related to patient health. The key features are as follows:

gender: Gender of the patient. Encoded as Male (0) and Female (1).

age: Age of the patient in years.

hypertension: Indicates the presence of hypertension (0 = No, 1 = Yes).

heart\_disease: Indicates the presence of heart disease (0 = No, 1 = Yes).

smoking\_history: Smoking status of the patient. Encoded as Never (0), Former (1), Current (2), and Unknown (3).

bmi: Body Mass Index, a measure of body fat based on height and weight.

HbA1c\_level: Glycated hemoglobin level, indicating average blood glucose levels over the past three months.

blood\_glucose\_level: Blood glucose level measured at the time of testing.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| gender | age | hypertension | heart\_disease | smoking\_history | bmi | HbA1c\_level | blood\_glucose\_level | diabetes |
| Female | 80 | 0 | 1 | never | 25.19 | 6.6 | 140 | 0 |
| Female | 54 | 0 | 0 | No Info | 27.32 | 6.6 | 80 | 0 |
| Male | 28 | 0 | 0 | never | 27.32 | 5.7 | 158 | 0 |
| Female | 36 | 0 | 0 | current | 23.45 | 5 | 155 | 0 |
| Male | 76 | 1 | 1 | current | 20.14 | 4.8 | 155 | 0 |
| Female | 20 | 0 | 0 | never | 27.32 | 6.6 | 85 | 0 |
| Female | 44 | 0 | 0 | never | 19.31 | 6.5 | 200 | 1 |
| Female | 79 | 0 | 0 | No Info | 23.86 | 5.7 | 85 | 0 |
| Male | 42 | 0 | 0 | never | 33.64 | 4.8 | 145 | 0 |
| Female | 32 | 0 | 0 | never | 27.32 | 5 | 100 | 0 |
| Female | 53 | 0 | 0 | never | 27.32 | 6.1 | 85 | 0 |
| Female | 54 | 0 | 0 | former | 54.7 | 6 | 100 | 0 |

## Analysis from model.py

The training process of the model involves the following key steps:

1. \*\*Data Cleaning\*\*: The dataset was cleaned to handle missing or incorrect values, ensuring data integrity.

2. \*\*Feature Encoding\*\*: Categorical features such as gender and smoking history were converted into numerical format to facilitate model training.

3. \*\*Splitting the Dataset\*\*: The dataset was divided into training and test sets to evaluate the model's performance. Typically, an 80-20 split was used, where 80% of the data was used for training and 20% for testing.

4. \*\*Model Training\*\*: The Logistic Regression model was trained using the training set. The model learned the relationship between the features and the target variable (diabetes).

5. \*\*Model Evaluation\*\*: The model's performance was assessed using the test set to check its accuracy and predictive power.  
  
# Importing the libraries

import numpy as np

import pandas as pd

import pickle

from sklearn.linear\_model import LogisticRegression

from sklearn.model\_selection import train\_test\_split

# Load the new dataset (replace 'new\_dataset.csv' with the actual path to your new CSV file)

dataset = pd.read\_csv(r'D:\My-Projects\Diabetes\_Prediction\templates\diabetes\_prediction\_dataset.csv')

# Handle missing or 'No Info' values in smoking\_history by replacing them with a default value, like 'unknown'

dataset['smoking\_history'].replace('No Info', 'unknown', inplace=True)

# Filling missing values for BMI, HbA1c\_level, and blood\_glucose\_level if any

dataset['bmi'].fillna(dataset['bmi'].mean(), inplace=True)

dataset['HbA1c\_level'].fillna(dataset['HbA1c\_level'].mean(), inplace=True)

dataset['blood\_glucose\_level'].fillna(dataset['blood\_glucose\_level'].mean(), inplace=True)

# Converting categorical data (gender, smoking\_history) into numerical format

dataset['gender'] = dataset['gender'].map({'Male': 0, 'Female': 1})

dataset['smoking\_history'] = dataset['smoking\_history'].map({'never': 0, 'former': 1, 'current': 2, 'unknown': 3})

# Check if there are any remaining missing values

print("Number of missing values in each column:")

print(dataset.isnull().sum())

# Splitting the features (X) and target variable (y)

X = dataset[['gender', 'age', 'hypertension', 'heart\_disease', 'smoking\_history', 'bmi', 'HbA1c\_level', 'blood\_glucose\_level']]

y = dataset['diabetes']  # Target variable

# Check for any missing values in X before fitting the model

if X.isnull().any().any():

    print("There are missing values in X. Filling them with column means.")

    X = X.fillna(X.mean())  # Fill any remaining NaN values with the mean of the respective columns

# Splitting Training and Test Set

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=0)

# Training the model using Logistic Regression

model = LogisticRegression()

# Fitting the model

model.fit(X\_train, y\_train)

# Saving the trained model to disk

pickle.dump(model, open('diabetes\_model.pkl', 'wb'))

# Loading the model and making a prediction (for testing purposes)

loaded\_model = pickle.load(open('diabetes\_model.pkl', 'rb'))

sample\_data = [[1, 80, 0, 1, 0, 25.19, 6.6, 140]]  # Sample input data

print(loaded\_model.predict(sample\_data))

## Model Deployment Using Flask

Flask is a lightweight web application framework for Python that allows for the easy deployment of web applications. In this project, Flask was used to create a simple web interface where users can input patient data and receive predictions from the trained model. The following steps were involved in the deployment:

1. \*\*Creating the Flask App\*\*: A Flask app was initialized to handle web requests and serve the user interface.

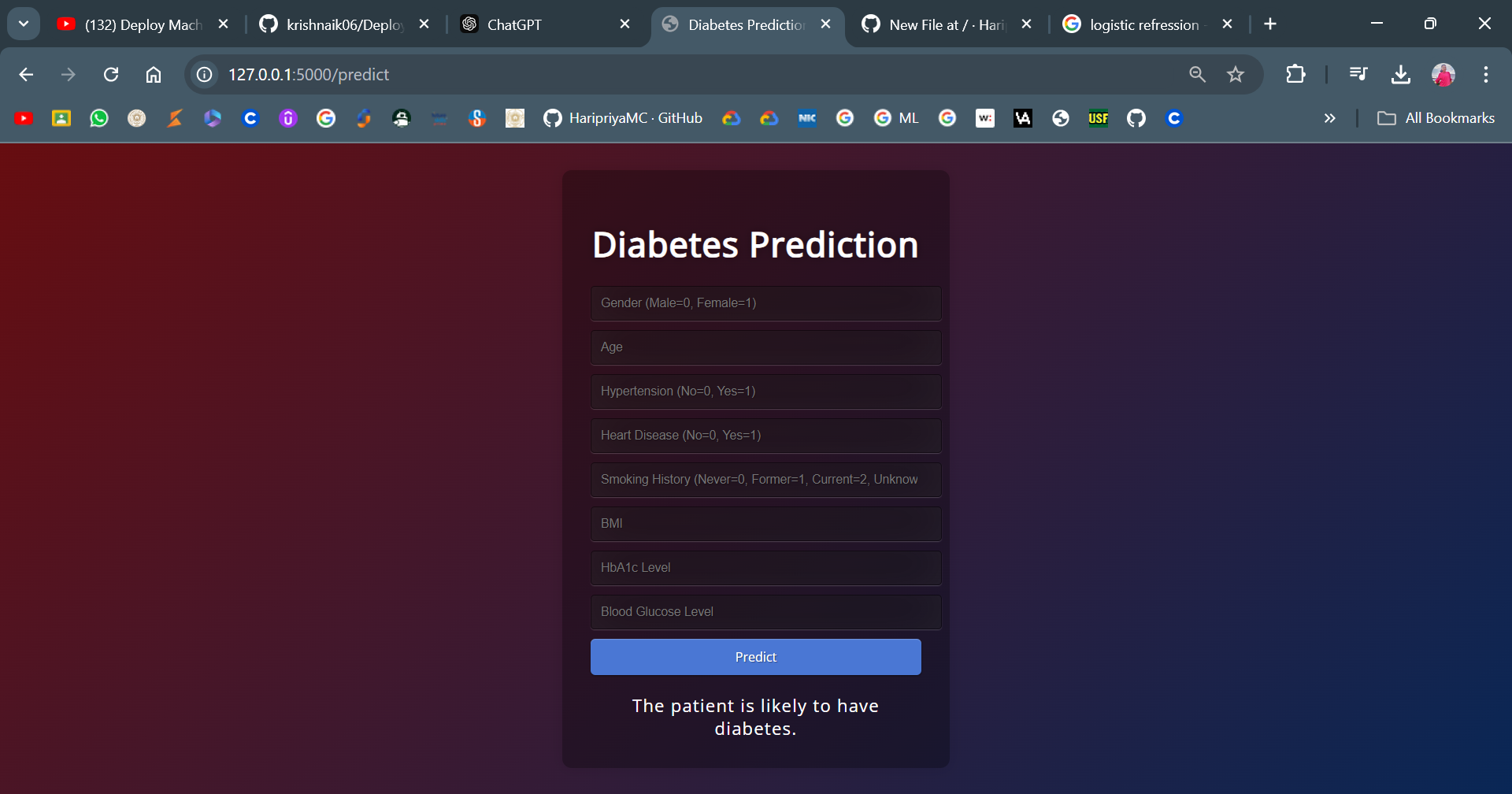
2. \*\*Defining Routes\*\*: Specific routes were defined to render the homepage and handle predictions. The '/' route renders the homepage, while the '/predict' route processes the input data and returns the prediction.

3. \*\*Integrating the Model\*\*: The trained Logistic Regression model was loaded and integrated into the Flask app, allowing the app to make predictions based on user input.

4. \*\*Running the Server\*\*: The Flask server was run locally, enabling users to access the application through their web browser.

## Result

Upon successful deployment, users can access the web application and input various medical parameters. The model predicts whether the patient is likely to have diabetes based on the input data. The result is displayed on the web page, allowing for immediate feedback. If the model predicts a high risk of diabetes, the application informs the user; if the risk is low, it provides reassurance.



## Conclusion

This project successfully demonstrates the use of machine learning techniques, specifically Logistic Regression, for predicting diabetes based on medical indicators. The integration of Flask allows for a user-friendly web interface, making the model accessible for healthcare professionals and patients. Future improvements could involve using more sophisticated models and incorporating additional features for better accuracy and predictive power.