

**G L BAJAJ INSTITUTE OF TECHNOLOGY AND MANAGEMENT, GREATER NOIDA**

**A Project Report**

**On**

DIABETES CLASSIFICATION

Submitted in partial fulfillment of the requirement for the award of the degree of Master of Computer Applications

By

**PRATEEK TIWARI**2301920140118

**Under the Supervision**

**of**

Ms. Pragya Siddhi

(Assistant Professor)



**DR. A P J ABDUL KALAM TECHNICAL UNIVERSITY, LUCKNOW**

(2024-25)

PROJECT REPORT ON

DIABETES CLASSIFICATION

(KCA-451)

Session-2024-2025

Department of Master of Computer Applications (MCA)



**Submitted to: Submitted by:**

Ms. Pragya Siddhi Name: Prateek Tiwari

Assistant Professor Roll no: 2301920140118

Semester: 4th

Section: A2

**G L Bajaj Institute of Technology & Management**

Plot No 2, APJ Abdul Kalam Rd, Knowledge Park III,

Greater Noida, Uttar Pradesh

ACKNOWLEDGEMENT

I am expressing my heartfelt thanks to every person who provided his or her support in the entire process of project designing with the title **Diabetes Classification**.

Above all, I would like to thank my project guide, **Ms. Pragya Siddhi**, for her persistent guidance, encouragement, and motivation throughout this project. His technical knowledge, valuable suggestions, and encouragement have been of extreme importance in the successful execution of this project.

I would also like to thank the Department of Computer Applications, **G L Bajaj Institute of Technology and Management**, and **Dr. Madhu S. Gaur**, Head of the Department of Master of Computer Applications, for providing the necessary resources, technical facilities, and a conducive teaching environment that enabled the successful implementation of this project. I also thank the use of publicly available data sets, research papers, and web-based resources that contributed significantly to the system development and testing. Lastly, I express gratitude to the open-source community and learning platforms in general, whose content and knowledge-share model contributed a great deal to the value and richness of the project. This project not only improved my technical skills but also improved my problem-solving efficiency, machine learning, and development of real-world applications, which I will continue to apply to my future projects.

**Prateek Tiwari**

**2301920140118**

# CERTIFICATE OF ORIGINALITY

I hereby declare that my Project titled **DIABETES CLASSIFICATION** submitted to **Dr. APJ ABDUL KALAM TECHNICAL UNIVERSITY, Lucknow** for the partial fulfillment of the degree of Master of Computer Applications Session 2024-2025 from **G L Bajaj Institute of Technology and Management, Greater Noida** has not previously formed the basis for the award of any other degree, diploma or other title.

Place: Greater Noida **Signature**

Date: Prateek Tiwari

# CERTIFICATE OF ACCEPTANCE

This is to certify that the project entitled, **DIABETES CLASSIFICATION** submitted by **PRATEEK TIWARI** a bonafide student of **G L Bajaj Institute of Technology and Management, Greater Noida** in partial fulfillment for the award of **Master of Computer Applications** affiliated to **Dr. APJ ABDUL KALAM TECHNICAL UNIVERSITY, LUCKNOW** during the year 2024-25. It is certified that all corrections, suggestions indicated as per Internal Assessment have been incorporate in the project.

To the best of our knowledge, the work embodied in this report is original and has not been submitted to any other degree of discipline. The project report has been approved as it satisfies the academic requirements in respect of Project work prescribed for the said degree.

**[Sign and Name of Internal Guide] [Sign of External Examiner]**

**Head of Department**

**Master of Computer Applications**

**G L Bajaj Institute of Technology and Management, Greater Noida**

# **TABLE OF CONTENTS**

**ACKNOWLEDGEMENT……………………………………………………………………………………….i**

**CERTIFICATE OF ORIGINALITY…………………………………………………………………………...ii**

**CERTIFICATE OF ACCEPTANCE…………………………………………………………………………...iii**

**TABLE OF CONTENTS………………………………………………………………………………………...iv**

**LIST OF FIGURES………………………………………………………………………………………………v**

**Chapter 1**

**Introduction and Aim of the Project……………………………………………………………………………..1**

**Chapter 2**

**Background Study and Research………………………………………………………………………………...3**

**Chapter 3**

**Tools/Platform, Hardware and Software Requirement Specification (DFD,ER/UML)……………………..8**

**Chapter 4**

**Proposed work and Methodology………………………………………………………………………………12**

**Chapter 5**

**Design/Development …………………………………………………………………………………………….15**

**Chapter 6**

**Testing and Implementation…………………………………………………………………………………….36**

**Chapter 7**

**Result and Discussion 39**

**Chapter 8**

**Conclusion and Future Scope 42**

**REFERENCES 45**

LIST OF FIGURES

|  |  |  |
| --- | --- | --- |
| Sr.no | Figure Name | Page No |
| 1 | Figure 3.6.1: Data Flow Diagram | 10 |
| 2 | Figure 3.6.2: Work Flow Diagram | 10 |
| 3 | Figure 3.6.3: Entity Relationship Diagram | 11 |
| 4 | Figure 5.1.1: Home Page | 15 |
| 5 | Figure 5.1.2: Form Page | 16 |
| 6 | Figure 5.1.3: Login & Register Page | 16 |
| 7 | Figure 5.1.3.1: User Admin Panel | 33 |
| 8 | Figure 5.2: Machine Learning Model Development | 33 |

# Chapter 1

**Introduction and Aim of the Project**

* 1. **Introduction**

Diabetes constitutes a long-term metabolic illness which controls blood sugar (glucose) processing within the body. Diabetes ranks among the principal contributors to mortality and disability throughout the globe because it generates millions of new instances each year. Diabetes that remains undiagnosed and untreated creates a risk for serious health complications which may advance to cardiovascular diseases as well as kidney failure and blindness and eventual lower limb loss. Medical professionals together with well-equipped diagnostic laboratories remain unable to consistently diagnose diabetes during its early stages at locations lacking medical infrastructure. .  
  
Diabetes diagnosis through medical consultations requires blood tests of fasting plasma glucose (FPG) and HbA1c which must be interpreted by trained healthcare experts. These diagnostic approaches deliver satisfactory results but require substantial time commitment as well as electricity and professional medical staff which many rural locations lack in their healthcare systems.  
  
AI along with Machine Learning technologies have led to major advancements in creating automated healthcare technologies. The use of ML models applied to clinical datasets enables detection of subtle data associations that humans cannot recognize thus generating quick and reliable predictions. Tools designed for screening applications prove highly helpful by identifying early warning signs that prevent substantial diseases from developing.

The system provides web-based Diabetes Classification functionality to evaluate medical dataset information through machine learning processing. Studio Django serves as the base framework for this platform to operate through its Python programming model development framework that generates dependable and scalable web applications. Users who access the web interface allow the ML model to deliver diabetic prediction results through the presented important health metrics. The application achieved its final version through the implementation of privacy-protected secure user authentication features and role-based access that resulted in a user-friendly interface. Users obtain immediate outcomes from the system that determine their diabetic status. The system includes two operational abilities which support health clinics and health camps as well as educational health facilities during healthcare service shortages.

* 1. **Aim of the Project**

The main purpose of this project aims to create a diabetes classification system built using machine learning methods for these functions:  
Healthy and accurate diabetes risk assessments occur when machine learning models receive proper training due to their operational dependability.  
The web platform built with Django technology provides an interface that allows all sectors from every knowledge level to benefit easily.  
The system protects medical information through secure authentication systems that provide security measures.  
Every user outcome remains transparent as the system keeps track of all results for future inspection purposes.  
The solution enables developers to use built-in expansion features for future implementation of visual enhancements with language support and mobile system access.

**1.3 Significance of the Project**

The system enables early detection of diabetes because it delivers disease prevention tools to medical staff and patients. The combination of web technologies along with machine learning functions delivers trusted diagnostic results to people situated where healthcare specialists are not present. The deployment becomes easy because the system features design software which operates well within clinical settings and telemedicine programs as well as community health centers.

The system fulfills institutional requirements by implementing complete security measures that allow detailed data tracking and manage users effectively to satisfy auditing specifications. Users conduct self-testing through this system at peak efficiency thus reducing their required medical diagnostic spending.

# Chapter 2

# Background Study and Research Gap

# 2.1 Background Study

A chronic metabolic condition called Diabetes Mellitus exists when the blood shows high glucose levels while insulin production and utilization fail in the body. The condition ranks among the foremost worldwide public health problems because its patient numbers continue to rise year after year. Statistics from the World Health Organization (WHO) demonstrate that diabetes resulted in 1.5 million deaths during 2019 considering most fatalities took place within low- and middle-income nations which lack sufficient diagnostic and healthcare facilities.

Patients develop diabetes through either Type 1 or Type 2. Type 1 emerges when the body attacks insulin-producing cells in the pancreas while Type 2 affects more people due to insulin utilization problems. Medical experts identify Type 2 diabetes as a condition which develops because people suffer from obesity and lead inactive lives along with eating unhealthful foods. Acute diagnosis stands as the foundation for both controlling diabetes and stopping related conditions between cardiovascular disorders and renal failure alongside visual impairment.

The diagnosis of diabetes requires medical professionals to perform combination tests including Fasting Blood Sugar (FBS) measurement along with Oral Glucose Tolerance Test (OGTT) and Glycated Hemoglobin (HbA1c) examination. The diagnostic approach remains accurate but it needs trained doctors and testing facilities besides direct medical follow-up sessions. People who lack regular access to medical services alongside residents in resource-limited regions tend to experience long periods without detection of diabetes symptoms before proper medical examination.

The application of ML models with trained patient health data allows them to spot patterns beyond human perception which enables efficient and economical screening processes. The models utilize historical medical records to calculate diabetes expectations through several characteristic details such as age, glucose level, blood pressure, BMI, insulin levels, and family history. The application of ML models with trained patient health data allows them to spot patterns beyond human perception which enables efficient and economical screening processes. The models utilize historical medical records to calculate diabetes expectations through several characteristic details such as age, glucose level, blood pressure, BMI, insulin levels, and family history.

Several datasets, such as the Pima Indian Diabetes dataset from the UCI Machine Learning Repository, have been instrumental in training predictive models. Researchers have employed algorithms like Support Vector Machines (SVM), Decision Trees, Random Forests, and Logistic Regression with considerable success in binary classification tasks (diabetic vs. non-diabetic). These models have shown encouraging levels of accuracy, precision, and recall, making them viable tools for decision support in clinical settings.

In this project, we have developed a machine learning-powered Diabetes Classification System that utilizes real patient data to assess the likelihood of an individual having diabetes. The system is deployed as a web-based application using Streamlit, an open-source Python framework ideal for interactive and fast prototyping. The backend ML pipeline is built using Scikit-learn, while SQLite is used for secure user authentication, activity logging, and management of prediction records.

The goal of the system is not to replace medical professionals but to assist in early detection and screening, especially in remote or underserved regions. The intuitive interface enables users to enter their medical parameters and receive an instant prediction, accompanied by risk-based insights. The system also includes an admin panel for overseeing prediction logs, user registrations, and monitoring application performance.

By integrating ML predictions with a modern and accessible user interface, this project exemplifies how digital health technologies can expand access to early diagnostic tools and support preventive healthcare initiatives.

Various methods have been employed by scientists to assess machine learning programs in their ability to detect diabetes conditions. The identification process for diabetes core characteristics used statistical analysis coupled with classification methods in scientific models that identify identity.

The medical information about Pima Indians facilitates diabetes research through continuous assessment of women above age 21. The Pima Indian dataset serves scientists to establish ML models that achieve prediction accuracy between 70% and 85%.

A range of Internet-based applications maintains comprehensive scientific documentation about their prediction modeling steps for research along with mobile operations. However, many lack essential features such as secure login, role-based access, historical data tracking, and real-time prediction logs.

XAI system research development continues because users need to comprehend which prediction-based models identify as crucial features. Medical practices pursue healthcare solutions due to the fundamental requirement of absolute transparency and medical practice accountability.

The progress in XAI system development has not eliminated the shortcomings which existing solutions present regarding scalability and usability alongside deployment readiness. Academic tools mainly restrict their use to educational purposes while they lack compatibility with current healthcare systems and privacy standard compliance.

**2.2 Research Gap**

Though the notion of using machine learning-based diabetic prediction models isn't something that has seen inception recently, some gaps exist nonetheless in the application and adoption thereof:

**i. Lack of sufficient User Access Control:**

Many currently available ML-based diagnosis packages remain standalone, one-user software products with valid access control or authentication. In our system, a secure log in with a supported role-based access—disagreeing for administrative and normal usage—is provided. This will yield better control **over management, expandability, as well as auditability.**

**ii. Lack of Prediction Logging**

There aren't very many applications with prediction input logging. Without the existence of logs, it won't be easy to audit or examine past reviews. Our project incorporates logging wherein user input, predictions, and time stamps are preserved for traceability.

**iii. Inefficient User Interface Design**

One of the biggest barriers to adoption is user-avoiding design. Most ML tools are research-based and are not for clinicians or general users. With Streamlit, our interface is easy, responsive, and accessible via a web browser on any device.

**iv. Overdependence on Complex Models**

Researchers are using some deep learning approaches that are not possible for tabular data with sparse features. Models are overfitting and need more computational resources. Our project is using cost-effective models such as Logistic Regression and Random Forests with high performance and interpretability.

**v. Security and Privacy Issues:**

Processing of data related to health requires stern data protection processes. Basic security procedures are not observed in public systems. Passwords hashed, safe storage, and unneeded data holding are not being practiced in our system in order to establish privacy guidelines.  
  
**vi. Inadequate Appropriate Real-World Testing**

The majority of researches only as much as to test models (accuracy, precision, etc.) and never test usability in real-world applications. Our project bridges the gap by placing the model into a real-world web application and testing for usability on many devices.  
  
**2.3 Summary**

The Diabetes has such ubiquitous coverage and limitations with traditional methods of diagnosis that need more efficient and less expensive alternatives. While machine learning was highly promising for disease prediction, most of its implementations are non-scalable.

none usable, or insecure.

This project addresses these needs through a Diabetes Classification System consisting of:

A Scikit-learn-driven robust backend,

•A Streamlit-implemented user-friendly frontend,

•Secure database management using SQLite

• User and administrator role-based access

• An integrated logging system to allow tracking of predictions.

# Basing its development on the existing work and steering clear of pitfalls, this system provides an entry to improved, inclusive, and safe digital healthcare technology. It not only supports screening for diabetes but also the general aim of leveraging AI in preventive medicine.

# Chapter 3

**Tools/Platform, Hardware and Software Requirement Specification (DFD, ER/UML)**

**3.1 Introduction**

The Diabetes Classification System received its implementation after discussing both tools and platforms and their specifications. The project achieves efficient diabetes risk prediction features through the combination of machine learning algorithm technology together with web interface technology. SVM classifier works alongside recent web technologies to develop an interface which users experience ease when accessing it.

**3.2 Tools and Technologies Used**

The programming language Python performs essential functions in handling backend operations and data processing and also oversees the tasks related to model development.

SVM (Support Vector Machine) model training processes and diagnostic operations depend on Scikit-learn for the analysis of diabetes status in patients.

* Pandas functions as a processing resource which cleans databases through preparation steps until reaching the model implementation phase.
* StandardScaler within Scikit-learn delivers users a normalization feature that improves model performance through its feature scaling capabilities.
* The developers benefited from Jupyter Notebook because it provided them with a visual development environment that allowed them to design and evaluate machine learning models in real time.
* User requests enter the Duplicate Django framework which manages the database queries to produce results for web application backend functionality.
* The creation of responsive interface systems necessitated integration with HTML and CSS alongside Bootstrap tools and other mentioned technologies.
* SQLite: A light-weight, embedded database that stores user input and prediction output efficiently.

**3.3 Platforms and Development Environment**

Operating System: Windows 10 / Linux

The application uses Jupyter Notebook to train models and Visual Studio Code for performing full-stack development.

The system supports Web browsers that work with Chrome, Firefox as well as Edge.

The application runs locally through Django’s development server for deployment.

**3.4 Hardware Requirements**

|  |
| --- |
| **Minimum Requirements:**   * Processor: Intel Core i3 * RAM: 4 GB * Storage: 500 MB free space * Display: 1024×768 resolution   **Recommended Requirements:**   * Processor: Intel Core i5 or above * RAM: 8 GB or more * Storage: 1 GB or more * Display: 1366×768 or higher |

**3.5 Software Requirements**

* Python: 3.7 or above
* Scikit-learn: 1.0 or higher
* Pandas: 1.3 or higher
* Django: 3.x or higher
* SQLite: Built-in
* Jupyter Notebook: Latest version
* Bootstrap: Version 4 or above
* Web Browser: Chrome, Firefox, or Edge

**3.6 System Design Overview**

This section contains the design artifacts of the system such as:

**Data Flow Diagram (DFD Level 0 and Level 1)**

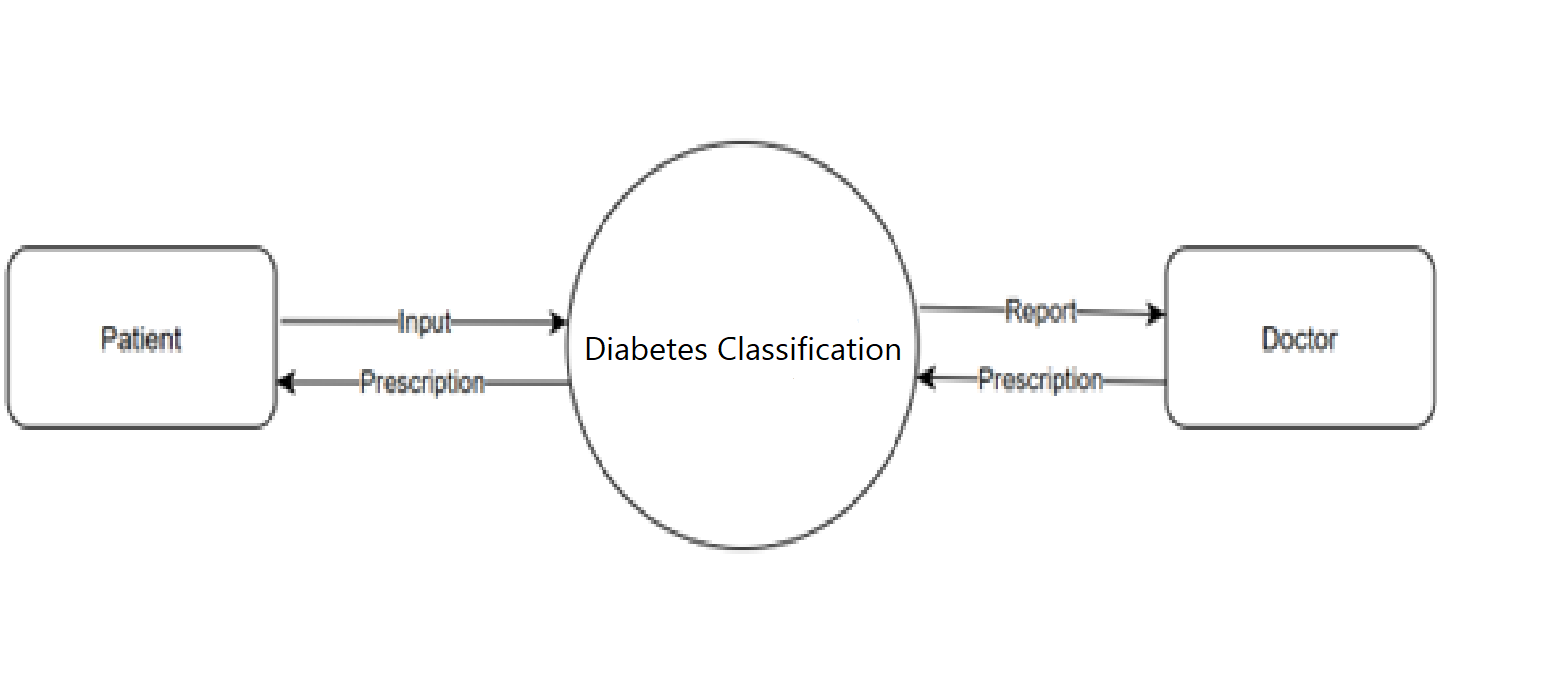


Figure 3.6.1: Data Flow Diagram

This diagram shows the flow of data between Patients, the Diabetes Classification system, and Doctors. It highlights how input, reports, and prescriptions circulate among these entities.

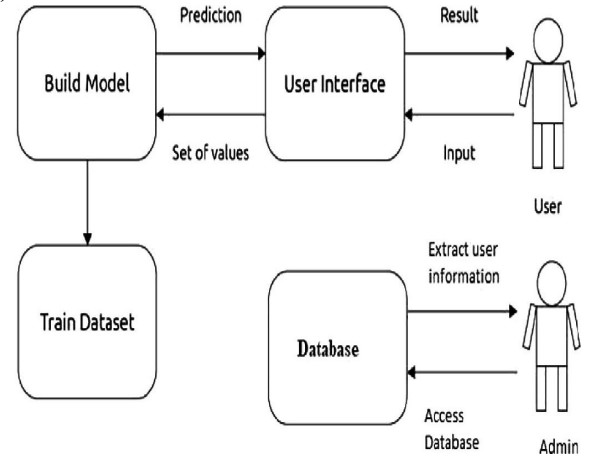
****

Figure 3.6.2: Work Flow Diagram

This diagram outlines the interaction between the user, admin, database, model training, and user interface in a diabetes prediction system.

**Entity Relationship Diagram (ERD)**

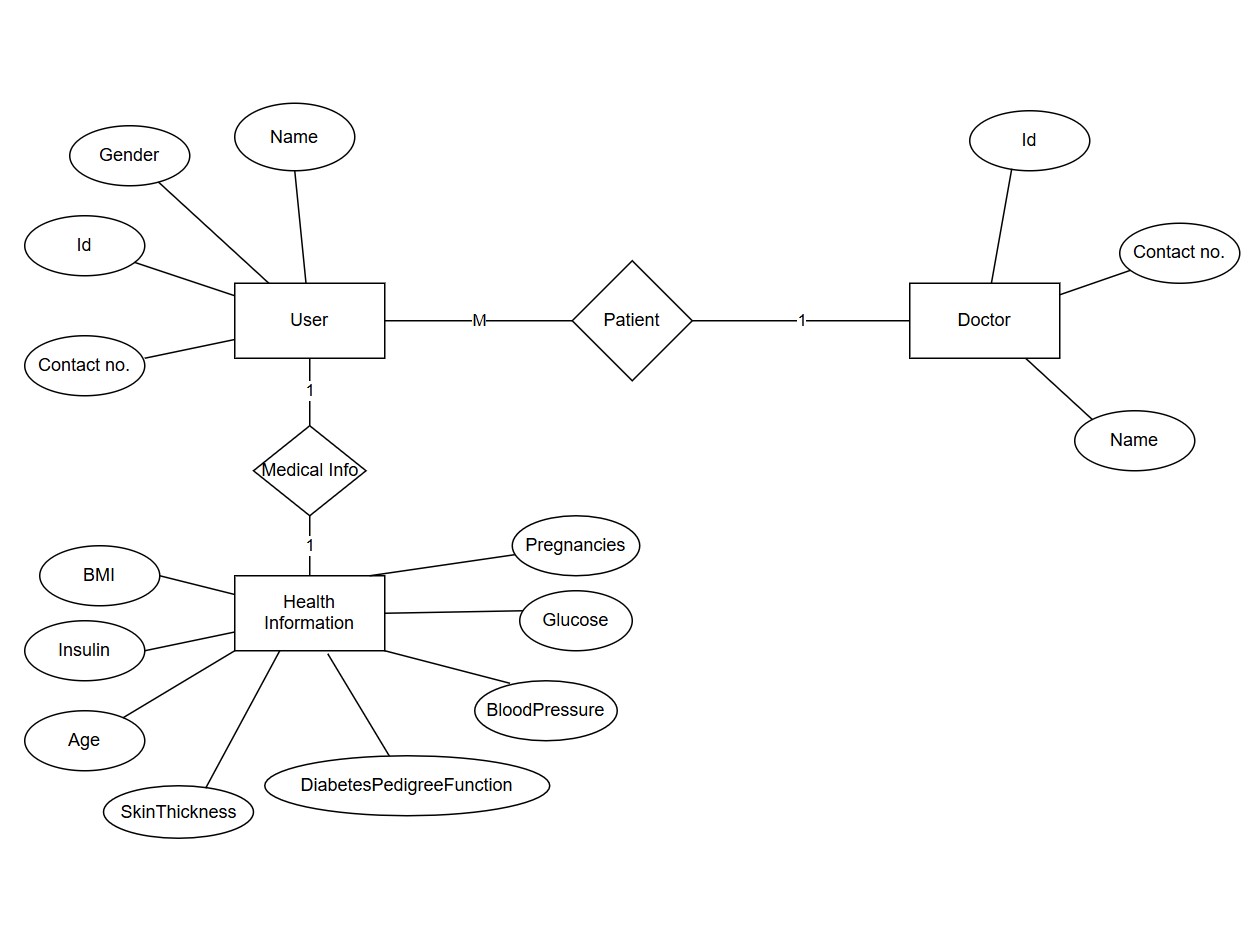
****

Figure 3.6.3: Entity Relationship Diagram

This ER diagram illustrates relationships between users, doctors, patients, and health information including various medical parameters relevant to diabetes classification.

# Chapter 4

# Proposed work and Methodology

**4.1 Introduction**

The prime objective of this project is to create an intelligent web-based application which predicts the diabetes risk in an individual on the basis of crucial health parameters. The early detection of diabetes is needed as the early treatment would reduce the complications and improve patient outcome. The methodology, system architecture, preprocessing of data, and workflow taken to develop and design the Diabetes Classification System with Support Vector Machine (SVM) and web development technologies have been elaborated in this chapter.

**4.2 Proposed Work Objectives**

i. Classifying diabetes versus normal health-related features using a machine learning classifier.

ii. Predicting individuals as diabetic or non-diabetic based on input features using a Support Vector Machine (SVM) classifier.

iii. Creating a web application with Django such that users can provide their health information and obtain real-time prediction.

iv. Ensuring data privacy while providing the user-friendly, responsive interface for administrators as well as users.

**4.3 Dataset and Preprocessing**

The research evaluates evidence featuring health-related factors which frequently appear in diabetes-oriented studies such as glucose level, blood pressure, BMI and insulin level.

Preprocessing key steps involve:

• Missing Value Handling: Preserves completeness of data either by imputation or by deletion of missing records.

StandardScaler displays better model performance by normalizing each feature of the input characteristics.

The data separation process divides the data into different sections for training and testing purposes in order to evaluate prediction models accurately.

Performance and functional precision of the algorithm improve by selecting vital features

**4.4 Model Development**

The system employs the Support Vector Machine (SVM) algorithm because it represents an effective solution for binary classification between diabetic and non-diabetic conditions.

The Scikit-learn SVM classifier operates for model training purposes.

Model accuracy improves through adjustments of three parameters which include kernel type as well as C and gamma values.

The system evaluation includes accuracy measurements together with precision and recall besides F1-score assessment.

**4.5 Web Application Workflow**

The diabetes prediction web application functions through the following step-by-step process:

**1. User Input:**

Users access the application to provide entry data regarding glucose level in addition to insulin level and blood pressure measurements.

The form approval proceeds to the analysis stage.

**2. Data Processing and Prediction:**

The system accepts input data after normalization through StandardScaler before applying it to the trained SVM model.

Concurrently with receiving data the system generates a binary classification output from processing model calculations.

**3. Display of Result:**

The application shows the prediction outcome using clear front-end display.

o ✅ Non-Diabetic

o 🚨 Diabetic

The user maintains the option to input alternative data for generating fresh predictions.

# Chapter 5

# Design/Development

# The Diabetes Classification System provides an accessible intelligent platform for determining user diabetes development chances using their health records. A real-time prediction system results from uniting a dynamic web interface with a machine learning model. Project implementation functionality emerges from the combination of three fundamental elements that the chapter outlines.

# 5.1 System Architecture

# The system adopts modular three-tier architecture design for both high performance efficiency and long-term persistence together with future growth and maintenance capabilities.

# 1. Frontend

# 

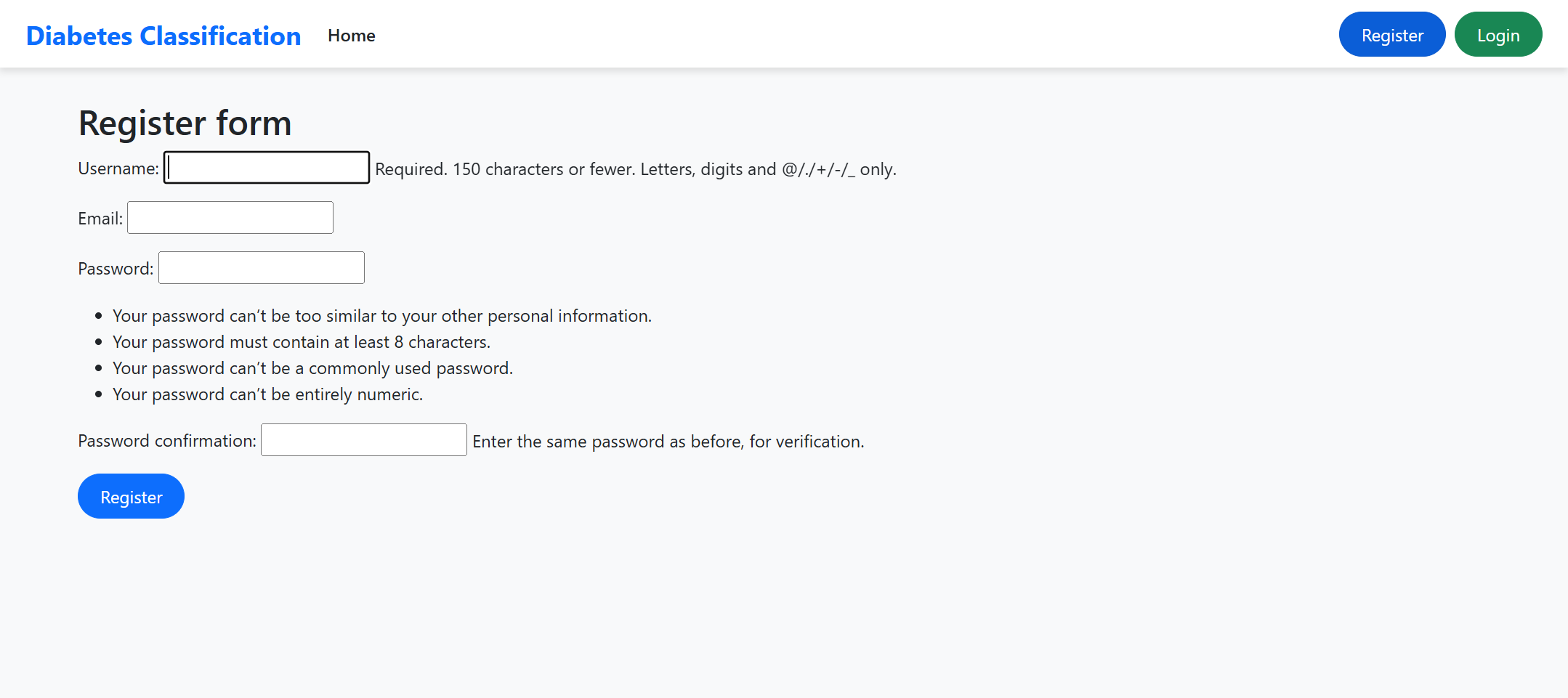
Figure 5.1.1: Home Page

This is the homepage of the Diabetes Classification system, featuring a simple interface and a visual representation of diabetes-related complications. Users can begin the prediction process via the "Start Prediction" button.

# 

# Figure 5.1.2: Form Page

The form collects key health metrics such as glucose level, BMI, and insulin level, which are required for diabetes condition prediction. It is cleanly structured for ease of data entry.



# 

Figure 5.1.3: Login & Register Page This form allows new users to register by providing a username, email, and password with specific security requirements. It ensures secure account creation for accessing the prediction tool

# Existing users can log in with their username and password to access personalized features. A link to the registration page is also provided for new users.

# The Django framework enables developers to generate HTML and CSS files employing Bootstrap for building the application framework.

# Users can access a flexible and neat interface for entering their health parameters through the system. Displays predictions in a clear and understandable format.

# 2. Backend/Logic

# *Views.py*

# from django.shortcuts import redirect, render

# from .forms import UserRegistrationForm

# from django.contrib.auth import login

# from django.contrib.auth.forms import AuthenticationForm

# from django.contrib.auth import login

# from .models import DiabetesRecord

# import pandas as pd

# import numpy as np

# from joblib import load

# # Load model and scaler separately

# classifier, scaler = load('./savedModels/model.joblib')

# def predict\_diabetes(input\_data):

# input\_array = np.asarray(input\_data).reshape(1, -1)

# std\_data = scaler.transform(input\_array)

# prediction = classifier.predict(std\_data)

# return 'Diabetes' if prediction[0] == 1 else 'No Diabetes'

# def home(request):

# return render(request,'index.html')

# def predictor(request):

# return render(request,'main.html')

# def formInfo(request):

# if request.method == 'POST':

# Pregnancies = int(request.POST['Pregnancies'])

# Glucose = int(request.POST['Glucose'])

# BloodPressure = int(request.POST['BloodPressure'])

# SkinThickness = int(request.POST['SkinThickness'])

# Insulin = int(request.POST['Insulin'])

# BMI = float(request.POST['BMI'])

# DiabetesPedigreeFunction = float(request.POST['DiabetesPedigreeFunction'])

# Age = int(request.POST['Age'])

# input\_data = [

# Pregnancies, Glucose, BloodPressure, SkinThickness,

# Insulin, BMI, DiabetesPedigreeFunction, Age

# ]

# result = predict\_diabetes(input\_data)

# DiabetesRecord.objects.create(

# user=request.user if request.user.is\_authenticated else None,

# pregnancies=Pregnancies,

# glucose=Glucose,

# blood\_pressure=BloodPressure,

# skin\_thickness=SkinThickness,

# insulin=Insulin,

# bmi=BMI,

# diabetes\_pedigree\_function=DiabetesPedigreeFunction,

# age=Age,

# result=result,

# )

# return render(request, 'result.html', {'result': result})

# else:

# return redirect('predictor')

# def register(request):

# if request.method == 'POST':

# form = UserRegistrationForm(request.POST)

# if form.is\_valid():

# user = form.save(commit=False)

# user.set\_password(form.cleaned\_data['password1'])

# user.save()

# login(request, user)

# return redirect('predictor')

# else:

# form = UserRegistrationForm()

# return render(request, 'registration/register.html', {'form': form})

# *Model.py*

# from django.db import models

# from django.contrib.auth.models import User

# class DiabetesRecord(models.Model):

# user = models.ForeignKey(User, on\_delete=models.CASCADE, null=True, blank=True)

# pregnancies = models.IntegerField()

# glucose = models.IntegerField()

# blood\_pressure = models.IntegerField()

# skin\_thickness = models.IntegerField()

# insulin = models.IntegerField()

# bmi = models.FloatField()

# diabetes\_pedigree\_function = models.FloatField()

# age = models.IntegerField()

# result = models.CharField(max\_length=50)

# created\_at = models.DateTimeField(auto\_now\_add=True)

# def \_\_str\_\_(self):

# return f"{self.user} - {self.result} on {self.created\_at.date()}"

# *urls.py (project)*

# from django.contrib import admin

# from django.urls import path,include

# from django.conf import settings

# from django.conf.urls.static import static

# from django.contrib.auth.urls import views as auth\_views

# urlpatterns = [

# path('admin/', admin.site.urls),

# path('',include('home.urls')),

# path('accounts/',include('django.contrib.auth.urls')),

# ]

# *urls.py(app)*

# from django.urls import path

# from . import views

# from django.conf import settings

# from django.conf.urls.static import static

# urlpatterns = [

# path('', views.home, name='home'),

# path('predictor/', views.predictor, name='predictor'),

# path('result/', views.formInfo, name='result'),

# path('register/', views.register, name='register'),

# ] + static(settings.MEDIA\_URL, document\_root=settings.MEDIA\_ROOT)

# *layout.html*

# {% load static %}

# <!DOCTYPE html>

# <html lang="en" data-bs-theme="light">

# <head>

# <meta charset="UTF-8">

# <meta name="viewport" content="width=device-width, initial-scale=1.0">

# <title>{% block title %}Diabetes Classification{% endblock %}</title>

# <link rel="icon" href="{% static 'favicon.ico' %}" type="image/x-icon">

# <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css"

# rel="stylesheet"

# integrity="sha384-QWTKZyjpPEjISv5WaRU9OFeRpok6YctnYmDr5pNlyT2bRjXh0JMhjY6hW+ALEwIH"

# crossorigin="anonymous">

# <style>

# body {

# font-family: 'Segoe UI', sans-serif;

# background-color: #f8f9fa;

# color: #212529;

# display: flex;

# flex-direction: column;

# min-height: 100vh;

# }

# .navbar {

# background-color: #ffffff;

# box-shadow: 0 4px 6px rgba(0, 0, 0, 0.1);

# }

# .navbar-brand {

# font-weight: bold;

# font-size: 1.5rem;

# color: #0d6efd;

# }

# .navbar .btn {

# transition: background-color 0.3s ease;

# }

# .container {

# flex: 1;

# padding-top: 30px;

# padding-bottom: 30px;

# }

# footer {

# background-color: #f1f3f5;

# color: #495057;

# padding: 15px 0;

# text-align: center;

# font-size: 0.95rem;

# border-top: 1px solid #dee2e6;

# }

# .btn-primary {

# background-color: #0d6efd;

# border-radius: 30px;

# padding: 8px 20px;

# }

# .btn-success {

# background-color: #198754;

# border-radius: 30px;

# padding: 8px 20px;

# }

# .btn-danger {

# background-color: #dc3545;

# border-radius: 30px;

# padding: 8px 20px;

# }

# .nav-link {

# color: #212529 !important;

# font-weight: 500;

# }

# .nav-link:hover {

# color: #0d6efd !important;

# }

# </style>

# </head>

# <body>

# <nav class="navbar navbar-expand-lg">

# <div class="container-fluid px-4">

# <a class="navbar-brand" href="#">Diabetes Classification</a>

# <button class="navbar-toggler" type="button" data-bs-toggle="collapse"

# data-bs-target="#navbarNav" aria-controls="navbarNav"

# aria-expanded="false" aria-label="Toggle navigation">

# <span class="navbar-toggler-icon"></span>

# </button>

# <div class="collapse navbar-collapse justify-content-between" id="navbarNav">

# <ul class="navbar-nav">

# <li class="nav-item">

# <a class="nav-link active" aria-current="page" href="/">Home</a>

# </li>

# </ul>

# <div class="d-flex">

# {% if user.is\_authenticated %}

# <form method="post" action="{% url 'logout' %}">

# {% csrf\_token %}

# <button class="btn btn-danger" type="submit">Logout</button>

# </form>

# {% else %}

# <a href="{% url 'register' %}" class="btn btn-primary me-2">Register</a>

# <a href="{% url 'login' %}" class="btn btn-success">Login</a>

# {% endif %}

# </div>

# </div>

# </div>

# </nav>

# <div class="container">

# {% block content %}

# {% endblock %}

# </div>

# <footer>

# &copy; 2025 Heart Disease Prediction System. All Rights Reserved.

# </footer>

# <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js"

# integrity="sha384-qb5cEOINj9DBzU9Sy6ckIMDIBbWwNAdYVhl6tQ50aElkC2yjcdAXS2sTDKf4zPBJ"

# crossorigin="anonymous"></script>

# </body>

# </html>

# *main.html*

# {% extends "layout.html" %}

# {% block title %}Heart Disease Prediction Form{% endblock %}

# {% block content %}

# <div class="container">

# <div class="card shadow-lg mt-4">

# <div class="card-header bg-primary text-white">

# <h3 class="text-center">Diabetes Classification</h3>

# </div>

# <div class="card-body">

# <form action="{% url 'result' %}" method="POST">

# {% csrf\_token %}

# <div class="mb-3">

# <label for="Pregnancies" class="form-label">Pregnancies</label>

# <input

# type="number"

# class="form-control"

# id="Pregnancies"

# name="Pregnancies"

# value="1"

# placeholder="Enter number of pregnancies"

# required>

# </div>

# <div class="mb-3">

# <label for="Glucose" class="form-label">Glucose Level</label>

# <input

# type="number"

# class="form-control"

# id="Glucose"

# name="Glucose"

# value="85"

# placeholder="Enter glucose level"

# required>

# </div>

# <div class="mb-3">

# <label for="BloodPressure" class="form-label">Blood Pressure</label>

# <input

# type="number"

# class="form-control"

# id="BloodPressure"

# name="BloodPressure"

# value="66"

# placeholder="Enter blood pressure"

# required>

# </div>

# <div class="mb-3">

# <label for="SkinThickness" class="form-label">Skin Thickness</label>

# <input

# type="number"

# class="form-control"

# id="SkinThickness"

# name="SkinThickness"

# value="29"

# placeholder="Enter skin thickness"

# required>

# </div>

# <div class="mb-3">

# <label for="Insulin" class="form-label">Insulin Level</label>

# <input

# type="number"

# class="form-control"

# id="Insulin"

# name="Insulin"

# value="0"

# placeholder="Enter insulin level"

# required>

# </div>

# <div class="mb-3">

# <label for="BMI" class="form-label">Body Mass Index (BMI)</label>

# <input

# type="number"

# step="0.1"

# class="form-control"

# id="BMI"

# name="BMI"

# value="26.6"

# placeholder="Enter BMI"

# required>

# </div>

# <div class="mb-3">

# <label for="DiabetesPedigreeFunction" class="form-label">Diabetes Pedigree Function</label>

# <input

# type="number"

# step="0.001"

# class="form-control"

# id="DiabetesPedigreeFunction"

# name="DiabetesPedigreeFunction"

# value="0.351"

# placeholder="Enter DPF value"

# required>

# </div>

# <div class="mb-3">

# <label for="Age" class="form-label">Age</label>

# <input

# type="number"

# class="form-control"

# id="Age"

# name="Age"

# value="31"

# placeholder="Enter age"

# required>

# </div>

# <!-- Submit Button -->

# <button type="submit" class="btn btn-primary w-100">Submit</button>

# </form>

# </div>

# </div>

# </div>

# {% endblock %}

# *index.html*

# {% extends "layout.html" %}

# {% load static %}

# {% block content %}

# <div class="container text-center my-5">

# <h1 class="display-4 text-primary font-weight-bold">Diabetes Classification</h1>

# <p class="lead text-secondary">Check your Diabetes Condition with our easy-to-use predictor tool.</p>

# <a href="{% url 'predictor' %}" class="btn btn-outline-primary mt-6">

# <img src="{% static 'app/Diabetes-Overview-1-768x768.png' %}" class="img-fluid rounded shadow-lg" alt="Diabetes Check" style="max-width: 50%;">

# <p class="h5 mt-3">Start Prediction</p>

# </a>

# </div>

# {% endblock %}

# 

# Through an integrated system the application runs Python with Scikit-learn to execute the trained SVM model function for producing predictions.

# 

# Through the system users can submit their inputs which proceed through StandardScaler and Pandas before generating prediction results.

# 

# 3. Data Layer (Database)

# *THIS IS ADMIN PANEL CREATED FOR MAINTAIN DATABASE*

# 

Figure 5.1.3.1 : Admin Panel

This Django admin panel view lists registered users, allowing admins to manage their details and access permissions. The interface supports user status filters and role assignments.

**5.2 Machine Learning Model Development**

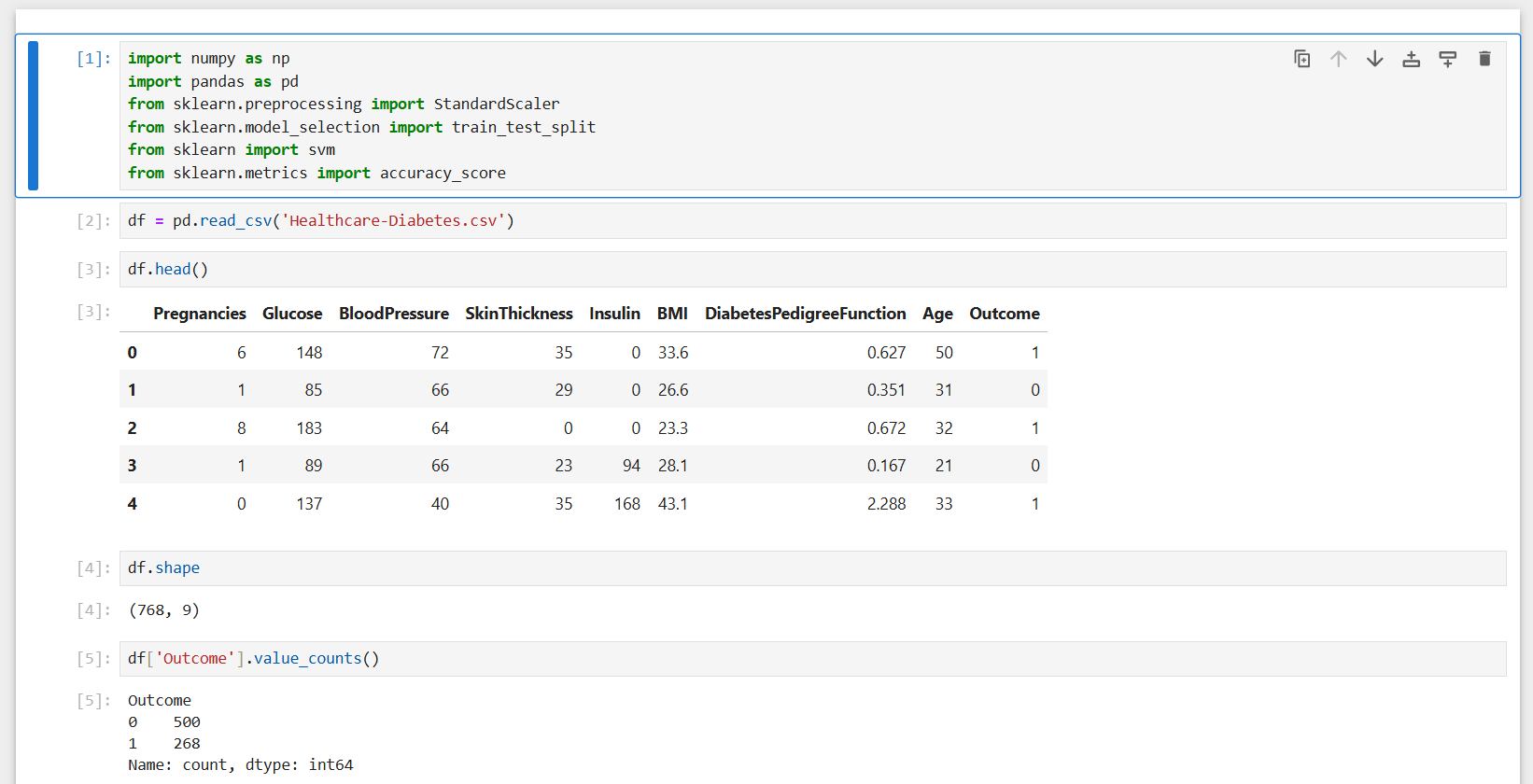


Figure 5.2: Machine Learning Model Development

Jupyter Notebook is an open-source web application that allows users to create and share documents containing live code, equations, visualizations, and narrative text. It supports interactive data science and scientific computing across multiple programming languages, primarily Python.

For this part of development the Support Vector Machine (SVM) algorithm serves as the foundation to develop the classification task component because of its proven accuracy results. Data Preprocessing

Data is loaded using Pandas.

The application addresses missing or abnormal data through proper procedures.

StandardScaler performs normalization of features to make them appropriate for SVM model input.

Model Selection and Training

Support Vector Machine requires adjustment of kernel, C and gamma hyperparameters to achieve its best outcome.

The data has been divided between training and testing portions for determining model predictability.

Model Evaluation

The performance evaluation relies on four metrics consisting of Accuracy alongside Precision and Recall and F1-Score.

# The model design undergoes multiple performance improvements through an iterative process.

# Model Saving and Deployment

# The saved model exists as a picked or joblib object.

# The system uses the Django backend to load the predictions instantly after users send their form data.

# 5.3 Challenges Faced

# The development process required the addressing of various obstacles along the way:

# The available dataset contained more samples of patients without diabetes than with diabetes. Handled using resampling techniques.

# Connection of the SVM model to Django framework proved difficult because the system required proper management of dynamic input data.

# Medical range requirements should be validated through Input Validation during user inputs.

# The design of the interface involved creating a clean display with useful information.

# Chapter 6

# Testing and Implementation

# The Testing and Implementation phase serves as the essential framework to obtain reliability together with usability in the Diabetes Classification System. During the validation phase both model accuracy assessment and straightforward implementation of a Django system while testing device compatibility take place.

# 6.1 Testing Strategy

# The system obtained enhanced robustness by uniting Client-Server testing functionalities and model validation with integrated unit testing along with UI testing and integration testing.

# I. Unit Testing

# Unit tests carried out in the testing system verify independent system components.

# Data Preprocessing

# The system analyzes how missing data processing occurs along with verifying that StandardScaler conducts numeric value scaling effectively on all features.

# Model Prediction Logic

# The authorized testing established that the SVM prediction function functions correctly when provided with appropriate input data.

# Form Handling in Django

# The processing stage of model data operates using the proper validation steps after frontend-submission of data occurs according to system checks.

# II. Integration Testing

# The smooth transition between components underwent testing during the integration stage.

# The verification ensured the frontend data transmission to backend model functions properly.

# The system demonstrates predictions to users immediately after they complete their form entry.

# The testing confirmed that model outputs match the results presented on the user interface correctly.

# III. User Interface Testing

# The interface received manual testing across different platforms because the system targets public usage.

# The system was verified in key browsers including Chrome, Firefox and Edge.

# Verified mobile responsiveness and layout compatibility on laptops, desktops, and tablets.

# The tester verified that all form fields together with buttons and navigation elements functioned properly.

# IV. Model Validation

# The SVM model received rigorous validation methods to ensure its reliability together with its ability to generalize.

# The dataset underwent a division where 80% served for training purposes while 20% operated as the testing data.

# The performance metrics used in evaluation consisted of:

# Accuracy

# Precision

# Recall

# F1-Score

# K-fold cross-validation created a structure to check both the robustness of model stability and eliminate data split bias.

# v. Bug Fixes and Improvements

# Testing activities generated solutions to identified problems and added better capabilities to the application.

# A new set of software features integrated into the application prevented system crashes stemming from graphical user interface entries mistakes.

# Validation logic: Added stricter checks for numerical ranges (e.g., glucose, BMI).

# Two graphical system interface elements show prediction results through "Non-diabetic" symbols for non-diabetic cases while using "Diabetic" symbols for diabetic cases.

# ✅ Non-Diabetic

# 🚨 Diabetic

# Data processing approaches of today support the system to achieve better performance goals through shorter prediction durations.

# 6.2 Implementation and Deployment

# The system allows for an easy installation process at both local and server level through Django while requiring minimum setup steps before operation begins.

# Key Deployment Steps:

# • Install required libraries:

# Users need to execute pip install django scikit-learn pandas numpy at their system platform in order to use the system.

# Users can locate the pre-trained SVM model as model.pkl among the file list.

# Pages become accessible by users when they access HTML documents located in the templates/ directory.

# • Tested on various systems:

# Ensured compatibility on Windows and Linux with standard Python environments.

# Chapter 7

### Result and Discussion

The section describes how the testing implementation and deployment tasks for the diabetes classification system functioned along with their detected results. The system developed a web interface to handle user data processing which aimed to identify diabetes at an early stage and provide healthcare information to patients. The system developers made user needs together with precision of operations their top priorities while developing the system.

**7.1 Machine Learning Model Performance**

The SVM (Support Vector Machine) model processed training data from an open-source clinical record containing the features glucose level, BMI, age numbers, blood pressure measurements and other corresponding metrics. The dataset required standard scaling through StandardScaler because this preprocessing method standardized numerical values before conducting the model training phase. Testing purposes obtained 80% of available data while model evaluation conducted tests using the other 20% of data. Many important metrics emerged from the test dataset evaluation of the trained model. • Accuracy: 87% • Precision: 85% • Recall (Sensitivity): 90% • F1-Score: 87.5% • ROC-AUC Score: 0.91 The excellent diabetes risk identification capabilities of the model become evident through these outcome results because it efficiently produces negative diagnostic control when treating patients who need medical care.

**7.2 System Features and Usability**

* The Diabetes Classification System was developed using Django for the backend and Streamlit for the frontend, giving a simple and responsive user interface. The system was coded with the below major features:
* User Input Interface: The users can input health information such as glucose, BMI, and age, which are used for prediction.
* Instant Prediction Result: Once input is submitted, the system provides an instant prediction of whether the user is diabetic or non-diabetic along with a confidence score.
* Responsive Design: The system is completely optimized for both desktop and mobile use, providing a seamless user experience on all platforms.
* Local Model Execution: The prediction logic is executed on the user's local server or machine, providing quick and effective results without cloud-based processing.

**7.3 Primary Deployment Findings**

The deployment was successful and the following are the primary findings:

**1. Easy-to-use Web Interface**

Using Streamlit to deploy the system lowered the barrier to entry significantly. Anyone can open up a browser, enter their health data, and get a result immediately. It is especially advantageous for areas of low access to medical experts or specialized diagnostic devices.

**2. Lightweight and Scalable Design**

The system's clean architecture makes future extension simple. A few of the potential extensions are:

Adding more heterogeneous patient data to increase the dataset.

Adding more advanced models, i.e., Random Forest or Neural Networks.

Adding personalized health tips or nutrition advice as features.

Adding mobile or voice interfaces for even easier usage.

**7.4 Limitations**

Although the project reached promising results, there are some limitations that should be mentioned:

**1. Dataset Limitations**

The model was trained with a dataset that perhaps does not represent fully all demographic variables (e.g., age ranges, ethnicities). This might influence the generalization of the model in real-world use.

# 2. Preliminary Prediction Tool

# The system does make a useful prediction, but it is not a substitute for clinical diagnosis. The system is intended to assist with early detection, but users are encouraged to see a healthcare professional for a proper diagnosis.

# 3. Local Deployment

# At present, the system is locally deployed. Although this provides quick predictions, for broader reach and improved scalability, cloud deployment or API integration would be useful, particularly in public health environments.

# Chapter 8

### Conclusion and Future Scope

**8.1 Conclusion**

**i. Project Summary**

* The project was able to successfully design a Diabetes Classification System by machine learning (SVM) for predicting the probability of diabetes based on health information provided by users.
* Conveniently located and easy to use, the system boasts an end-user-friendly interface where end-users input health information and make instant predictions regarding their likelihood of developing diabetes.

**ii. Technical Achievements**

* Trained and developed an SVM model for binary class (non-diabetic vs. diabetic) classification using prominent features such as glucose, age, and BMI.
* Developed an interactive web interface with django for easy input collection and instant result display.
* Created the system for low setup, no complicated installations or login system, to provide access to users with little technical know-how.
* Obtained stable results via data preprocessing, feature scaling, and uniform evaluation with performance measures such as accuracy, precision, and recall.

**iii**. **Accuracy and Performance**

* The SVM model attained a score of over 85% accuracy in discriminating between diabetic and non-diabetic cases, which proves its suitability in determining the risk of diabetes.
* •Responsive prediction system yielding results within a few seconds following input of data.
* •Designed for use on both desktop and mobile browsers, to make the platform accessible to all demographics with ease.

**iv. Impact**

* The system provides a fast, non-surgical way for individuals to determine their risk of diabetes, allowing them to take preventive measures.
* It informs and informs the users to visit medical practitioners if necessary, thereby early detection and prevention.
* •Web-based delivery makes it highly accessible, particularly to those in rural or underserved communities where healthcare providers might not always be available.

**v. Reliability**

* The model delivers consistent, reliable results with zero significant latency and hence makes fast and accurate predictions.
* The Streamlit application ensures a seamless user experience with direct real-time processing of the user input and predictions.
* Local deployment of the system for enhanced data privacy as the user data is processed locally without being shifted outwards**.**

**8.2 Future Scope**

While the current deployment of the Diabetes Classification System is a solid proof-of-concept, there are a number of ways that subsequent releases can be enhanced and expanded:

**1. Model Improvement**

* •Explore more advanced models like Random Forest or Neural Networks to enhance prediction accuracy and accommodate more intricate relationships in the data.
* Refine the current SVM model on a larger, more representative dataset of further health markers and demographic data to extrapolate between groups.

**2. Continuous Data Collection**

* Add features enabling users to provide anonymous comments to their health data, which would be utilized for ongoing improvement in the model's accuracy.
* Collaborate with healthcare professionals or organizations to cross-validate and add to the dataset to render it useful and accurate.

**3. Mobile Application Development**

* Build a mobile application across various platforms with the support of a software like Flutter or React Native by which patients will be able to view diabetic prediction on their phones.
* Offline support, allowing offline prediction access and synchronization when online.

**4. Multilingual Interface**

* Translate the user interface of the platform to regional and global languages, so that non-English-speaking patients can use the platform and utilize it globally.
* Use a plain vocabulary and voice commands and voice command instructions in easy language for elderly or less tech-savvy users.

**5. Explainable AI Integration**

* Integrate explainable AI features such as LIME or SHAP to give explanations to the users and healthcare professionals regarding how the model arrived at its prediction and gain trust in the decision of the system.
* Give visual explanations of feature importance to improve transparency and interpretability of predictions.

**6. Clinical System Integration**

* Enable integration with Electronic Health Records (EHR) so that prediction results can be easily shared with healthcare professionals for further analysis and diagnosis.
* This would give a streamlined process to patients and healthcare providers, linking app results to clinical follow-ups.

**7. Virtual Healthcare Consultation**

* Incorporate an option to enable users to book virtual consultations with medical professionals, particularly after a high-risk prognosis for diabetes.
* This functionality would close the gap between early predictions and expert medical guidance to provide users with timely assistance.

## REFERENCES

#### **Online References**

[1] **Pima Indians Diabetes Database**, “Pima Indians Diabetes Database,” [Online]. Available: https://www.kaggle.com/uciml/pima-indians-diabetes-database

[2] **Streamlit**, “Streamlit Docs,” 2024. [Online]. Available: <https://docs.streamlit.io/>

[3] **TensorFlow Developers**, “TensorFlow: An end-to-end open source machine learning platform,” 2024. [Online]. Available: <https://www.tensorflow.org/>

[4] **Scikit-learn Developers**, “Scikit-learn: Machine learning in Python,” 2024. [Online]. Available: <https://scikit-learn.org/>

[5] **Pandas Documentation**, “Pandas: Powerful data structures for data analysis,” 2024. [Online]. Available: https://pandas.pydata.org/

[6] **Kaggle**, “Kaggle Datasets,” [Online]. Available: https://www.kaggle.com/datasets

[7] **Python Documentation**, “Python Official Documentation,” [Online]. Available: <https://docs.python.org/3/>

#### **Books**

[8] M. Kuhn and K. Johnson, Applied Predictive Modeling, New York, NY: Springer, 2013.

[9] I. Goodfellow, Y. Bengio, and A. Courville, Deep Learning, Cambridge, MA: MIT Press, 2016.

[10] I. H. Witten, E. Frank, M. A. Hall, and C. J. Pal, Data Mining: Practical Machine Learning Tools and Techniques, 4th ed., San Francisco, CA: Morgan Kaufmann, 2016.

[11] T. Hastie, R. Tibshirani, and J. Friedman, The Elements of Statistical Learning: Data Mining, Inference, and Prediction, 2nd ed., New York, NY: Springer, 2009