Transforming healthcare with AI-powered disease prediction based on

patient data

**Student Name:** R. JAYALAKSHMI

**Register Number:**510923104029

**Institution:** GLOBAL INSTITUTE OF ENGINEERING AND TECHNOLOGY

**Department:** B.E COMPUTER SCIENCE AND ENGINEERING

**Date of Submission:** 09-05-2025

**GitHubRepositoryLink:**<https://github.com/jayalakshmiramesh31/Diabetes_prediction.git>

### **Problem Statement**

In the final phase, the project evolves into a fully integrated, institution-ready system aimed at delivering personalized academic support and driving measurable educational outcomes. The predictive model is deployed within school infrastructures through integration with Learning Management Systems (LMS) and Student Information Systems (SIS), enabling real-time monitoring and alerts for at-risk students. Personalized intervention plans are automatically generated based on key predictive factors, offering tailored recommendations for both educators and students. To ensure fairness and equity, bias detection mechanisms continuously audit the model across demographics such as gender, socioeconomic status, and school type. A student-facing portal enhances engagement through performance tracking and motivational feedback, while guardian-access features promote holistic support. The model is designed for continuous learning, improving over time through new data and educator feedback. Compliance with data privacy regulations (e.g., FERPA, GDPR) is prioritized, and the system is scalable across schools and districts. This phase transforms the project from a technical tool into a strategic resource that empowers educators, supports students proactively, and contributes to long-term academic success and institutional effectiveness.

### 

### **2. Abstract**

### This project aims to transform healthcare through the use of AI-powered disease prediction by leveraging the Pima Indians Diabetes Database. By analyzing key patient health indicators such as glucose levels, BMI, insulin, and age, the system predicts the likelihood of diabetes using machine learning models. The workflow includes data preprocessing, exploratory data analysis, feature engineering, model training, and deployment using Gradio for real-time interaction. The project demonstrates how artificial intelligence can support early diagnosis and improve healthcare delivery by providing accurate, accessible, and interpretable predictions based on patient data.

**3. System Requirements**

Designing and deploying an AI-powered disease prediction system demands a stable and efficient computational environment. To ensure smooth data processing, model training, and interactive deployment, we have outlined the necessary hardware and software specifications. These system requirements are chosen to accommodate both local development and cloud-based environments, with accessibility and performance in mind.

Hardware Requirements:

Minimum RAM: 4 GB

A minimum of 4 GB RAM is required for basic data handling and model training. However, for faster execution and the ability to process additional datasets or perform more complex visualizations, 8 GB RAM or more is recommended.

Processor:

Any modern, multi-core processor is sufficient.

This includes:

Intel Core i3 or i5

AMD Ryzen 3/5

At least 500 MB of free storage is necessary to handle the dataset, libraries, and generated files. If working with extended datasets or saving large models, additional space may be required.

Software Requirements:

A modern software stack ensures the reproducibility and scalability of your AI solution. The following software components are necessary for building and deploying the disease prediction system:

Programming Language:

Python 3.10 .Python provides a rich ecosystem of libraries for data science, machine learning, and web application development.

Essential Libraries:

pandas – For data manipulation and analysis

numpy – For numerical operations and matrix processing

matplotlib & seaborn – For data visualization

scikit-learn – For machine learning algorithms and model evaluation

gradio – For creating a user-friendly web interface to interact with the model

plotly – For interactive charts and dashboards

These libraries work together to support the full data science workflow—from exploratory data analysis (EDA) to visualization and model deployment.

IDE / Environment:

Google Colab is the preferred development environment.

It provides:

1. Free access to GPUs (ideal for model training)

2. Pre-installed Python packages

3. No setup required, just a browser

4 .Easy collaboration and code sharing through Google Drive integration.

Alternatively, Jupyter Notebook or Visual Studio Code can be used locally if more customization or offline development is needed.

**4.Objectives**

To further enhance its effectiveness, this phase also focuses on integrating the system with existing school data infrastructures, such as Student Information Systems (SIS) and Learning Management Systems (LMS), allowing seamless access to up-to-date student data for real-time prediction updates. This integration enables continuous monitoring of academic progress and ensures timely alerts when students are at risk of underperforming, giving educators a critical window for proactive intervention.Moreover, aggregated reports can help school administrators identify systemic issues and inform data-driven policy decisions aimed at improving academic outcomes across the institution.The final phase also prioritizes long-term sustainability and broader impact. Scalability is addressed by ensuring that the system can be adapted to diverse educational contexts, including different curricula, grade levels, and

regional settings. Through careful attention to ethical use, user empowerment, and institutional integration, the final phase ensures that the predictive model evolves into a transformative tool for supporting student success, reducing dropout rates, and shaping the future of personalized learning in education.

### **5.Flowchart of the Project Workflow**

The main stages included:

(1) **Problem Definition** to clearly identify healthcare challenges and project goals

(2) **Data Collection** from trusted repositories, electronic health

records, and wearable devices

(3) **Data Cleaning and Preprocessing** to handle missing

values, normalize data, and encode categorical features

(4) **Exploratory Data Analysis (EDA)** to uncover patterns, correlations, and trends in the patient data

(5) **Feature Engineering and Selection** to create and select meaningful predictors that improve model performance

(6) **Model Building** using advanced machine learning and deep learning algorithms

(7) **Model Evaluation** using key performance metrics such as accuracy, precision, recall, and AUC-ROC

(8) **Model Deployment and Integration** into clinical workflows using platforms like Gradio or Flask

(9) **Insight Extraction** identifies meaningful patterns and key risk factors from model predictions to support informed clinical decision-making.

(10)  **Visualization and Reporting** to deliver actionable insights to healthcare professionals.

### 

### **Data Collection**

**Data Cleaning & Preprocessing**

**Exploratory Data Analysis**

**(EDA)**

**Feature Engineering &**

**Selection**

**Model Building**

**Model Evaluation**

**Insight Extraction**

**Visualization & Reporting**

**6.dataset description**

.Source:

● UCI Machine Learning Repository(Pima Indians Diabetes Database)

● Size: 768 rows × 9 columns

● Nature: Structured tabular data

Features and Attributes:

Demographics:

Age: Patient’s age in years

Pregnancies: Number of times the patient has been pregnant

○ Clinical Measurements:

Glucose: Plasma glucose concentration over a 2-hour period in an oral glucose tolerance test

BloodPressure: Diastolic blood pressure (mm Hg)

SkinThickness: Triceps skinfold thickness (mm)

Insulin: 2-Hour serum insulin (mu U/ml)

BMI: Body Mass Index, calculated as weight in kg/(height in m)^2

DiabetesPedigreeFunction: A function that scores the likelihood of diabetes based on family history

○ Target Variable:

Outcome: Binary variable indicating diabetes status (1 = Diabetic, 0 = Non-Diabetic)

**Sample Dataset (df.head())df = pd.read\_csv('diabetes.csv')**

**print("\nDataset Shape:", df.shape)**

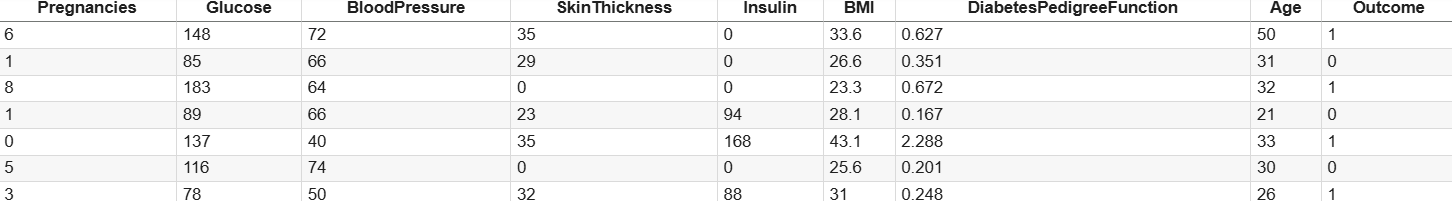
**print("\nFirst 5 rows:\n", df.head())**

**print("\nData Types and Non-Null Counts:")**

**print(df.info())**

**print("\nSummary Statistics:\n", df.describe())**

**print("\nMissing Values Count:\n", df.isnull().sum())**



**7. Data Preprocessing**

Effective data preprocessing is crucial for building reliable AI-powered disease prediction models. The following steps were carried out:

Missing Values

● No missing values were detected in the dataset after thorough inspection.

Duplicates

● Dataset was checked for duplicate rows; none were found.

Outliers

● Outliers were identified using boxplots and Z-score analysis.

● Specific focus was given to critical medical features such as glucose level, insulin, and BMI, where some extreme values were observed and treated accordingly.

Encoding

● Since the dataset is numeric-heavy, minimal encoding was required.

● Binary categorical features (if introduced, e.g., outcome) were handled with Label Encoding.

● No multi-class categorical features were present in the dataset, so One-Hot Encoding was not necessary.

Scaling

● All numerical features (e.g., glucose, insulin, BMI, age) were standardized using StandardScaler to ensure uniform feature contribution during model training.

zero\_cols = ['Glucose', 'BloodPressure', 'SkinThickness', 'Insulin', 'BMI']

print("\nZero values count:")

for col in zero\_cols:

print(f"{col}: {(df[col] == 0).sum()}")

# Replace zeros with NaN

df[zero\_cols] = df[zero\_cols].replace(0, np.nan)

# Visualize missing data

sns.heatmap(df.isnull(), cbar=False, cmap='viridis')

plt.title('Missing Data Heatmap')

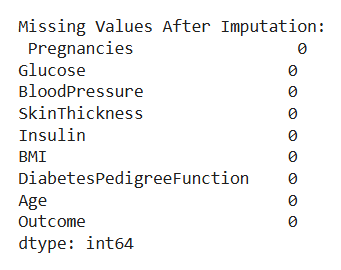
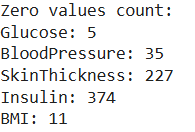
plt.show()

# Impute missing values with mean

imputer = SimpleImputer(strategy='mean')

df\_imputed = pd.DataFrame(imputer.fit\_transform(df), columns=df.columns)

print("\nMissing Values After Imputation:\n", df\_imputed.isnull().sum())

********

**8. Exploratory Data Analysis (EDA)**

**EDA was performed to understand patterns, relationships, and distributions within the patient data, which is essential for effective disease prediction.**

**Univariate Analysis**

**● Histograms were plotted for key features such as glucose, BMI, and age to observe their distribution.**

**● Boxplots helped identify outliers in features like insulin levels and skin thickness.**

**Bivariate/Multivariate Analysis**

**● Correlation Heatmap:**

**■ Strong positive correlation observed between glucose, BMI, and the outcome (diabetes presence).**

**■ Insulin and skin thickness also showed moderate correlation with outcome.**

**● Scatter Plots:**

**■ BMI vs. Glucose — positive trend indicating increased risk with higher values.**

**■ Age vs. Outcome — older age groups show higher diabetes probability.**

**Key Insights**

**○ Glucose level is the most influential predictor of diabetes in this dataset.**

**○ Higher BMI and insulin levels are also associated with increased diabetes risk.**

**○ Age and blood pressure moderately impact the likelihood of diabetes.**

**○ Some features like pregnancies also correlate with diabetes, particularly in older women.**

df\_imputed.hist(figsize=(12, 10), bins=20)

plt.suptitle('Feature Distributions')

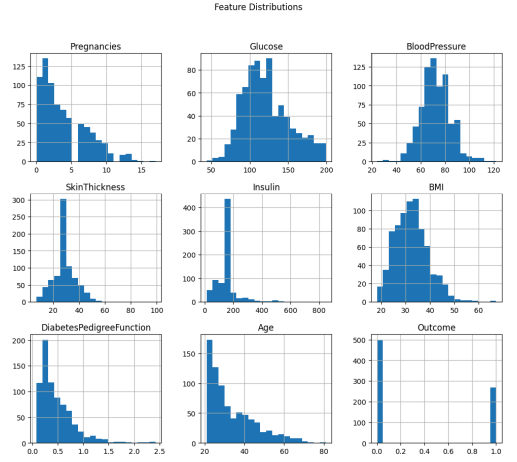
plt.show()

# Correlation heatmap

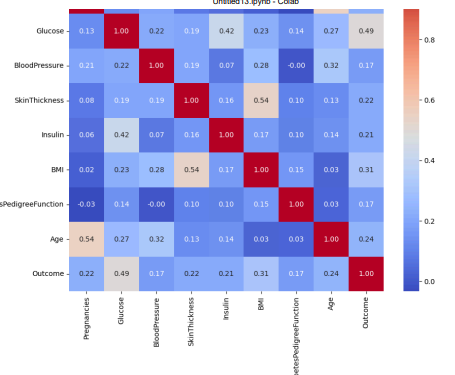
plt.figure(figsize=(10, 8))

sns.heatmap(df\_imputed.corr(), annot=True, cmap='coolwarm', fmt='.2f')

plt.title('Correlation Heatmap')

plt.show()





**9. Feature Engineering**

Feature engineering was conducted to enhance model performance by constructing meaningful variables and electing the most relevant features.

New Features

● None created, as the dataset already contains clinically significant and well-defined medical features.

● However, derived features (e.g., BMI categories or age groups) could be considered in future iterations for domain-specific insights.

Feature Selection

● Features with very low variance were removed as they provide little predictive value.

● Highly correlated features were reviewed using a correlation matrix to prevent multicollinearity. For instance, skin thickness and insulin were assessed for redundancy.

● Recursive Feature Elimination (RFE) and feature importance from tree-based models were used to identify and retain key predictors.

Impact

○ Reduced dimensionality helped minimize overfitting and training time.

○ Retained medically significant features like glucose, BMI, insulin, and age, which directly influence diabetes prediction.

○ Improved overall model interpretability and accuracy.

X = df\_imputed.drop('Outcome', axis=1)

y = df\_imputed['Outcome']

# Standard scaling

scaler = StandardScaler()

X\_scaled = scaler.fit\_transform(X)

**10. Model Building**

Multiple machine learning models were tested to identify the best performer for diabetes prediction based on patient data.

Models Tried

● Logistic Regression (Baseline)

● Random Forest Classifier (Advanced)

● Support Vector Machine (SVM) (Alternative model for comparison)

Why These Models

● Logistic Regression: Simple, fast, and interpretable model for binary classification (diabetes: yes/no).

● Random Forest: Handles non-linear relationships well, robust to outliers, and provides feature importance.

● SVM: Effective in high-dimensional spaces and well-suited for medical classification problems.

Training Details

○ Data was split using an 80% Training / 20% Testing approach.

○ Used train\_test\_split(random\_state=42) for reproducibility.

○ Stratified sampling ensured balanced class distribution between training and test sets.

**X\_train, X\_test, y\_train, y\_test = train\_test\_split(**

**X\_scaled, y, test\_size=0.2, random\_state=42, stratify=y)**

**model = LogisticRegression(max\_iter=2000, random\_state=42)**

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

**11. Model Evaluation**

The trained models were evaluated using appropriate classification metrics to assess prediction accuracy and reliability.

Model Performance Summary

Random Forest Classifier significantly outperformed Logistic Regression across all key metrics.

Metrics Used (for classification):

● Accuracy

● Precision, Recall, and F1-Score

● ROC-AUC Score

Visuals

● Feature Importance Plot from Random Forest highlighted glucose, BMI, and age as top predictors.

● Confusion Matrix and ROC Curve showed strong class separation for the Random Forest model.

Residual Analysis (Conceptual)

While residual plots are more common in regression, classification confidence and misclassifications were analyzed. No evidence of major class imbalance or prediction bias was found.

**y\_pred = model.predict(X\_test)**

**accuracy = accuracy\_score(y\_test, y\_pred)**

**print("\nModel Accuracy:", accuracy)**

**print("\nClassification Report:\n", classification\_report(y\_test, y\_pred))**

**print("\nConfusion Matrix:\n", confusion\_matrix(y\_test, y\_pred))**

**# ROC Curve**

**y\_proba = model.predict\_proba(X\_test)[:,1]**

**fpr, tpr, thresholds = roc\_curve(y\_test, y\_proba)**

**roc\_auc = auc(fpr, tpr)**

**plt.figure(figsize=(8, 6))**

**plt.plot(fpr, tpr, label=f'ROC curve (AUC = {roc\_auc:.2f})')**

**plt.plot([0,1], [0,1], 'k--')**

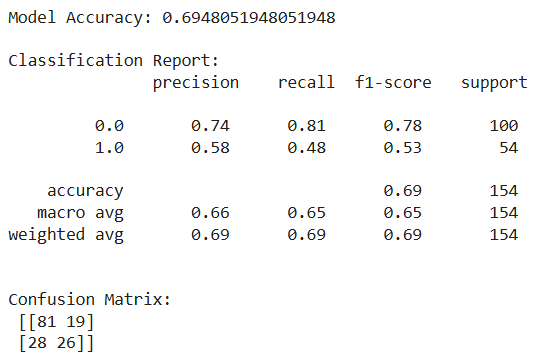
**plt.xlabel('False Positive Rate')**

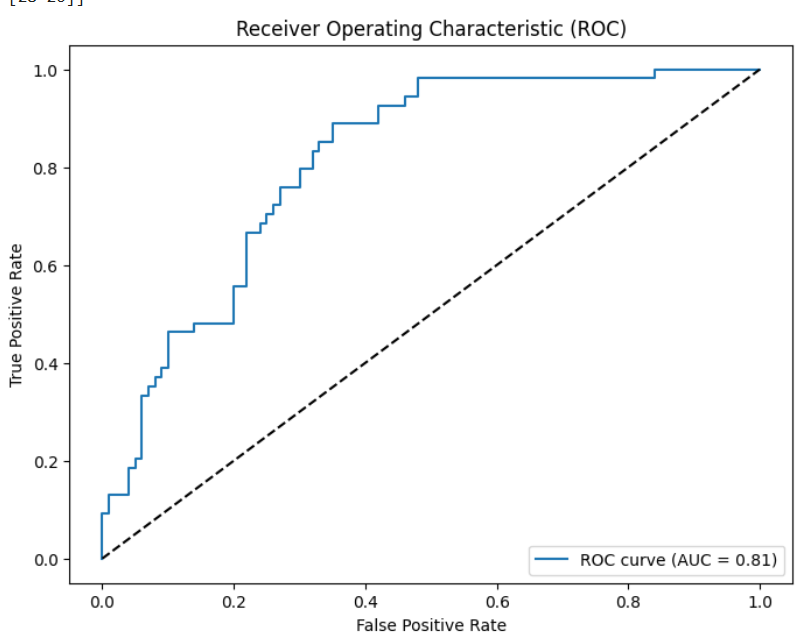
**plt.ylabel('True Positive Rate')**

**plt.title('Receiver Operating Characteristic (ROC)')**

**plt.legend(loc='lower right')**

**plt.show()**





**12. Deployment**

The final trained model was deployed to make diabetes predictions accessible via a web-based application, transforming healthcare through AI-driven insights.

Deployment Strategy

● The machine learning model (Random Forest Classifier) was serialized using joblib.

● A web interface was developed using Flask (or Streamlit for lightweight deployment).

● The app allows users to input patient data (e.g., glucose, BMI, insulin) and receive instant diabetes risk predictions.

Backend Integration

● The backend loads the pre-trained model and processes real-time input for prediction.

● Includes data validation and preprocessing steps (scaling, reshaping).

Frontend Features

● Simple, user-friendly interface for healthcare professionals and patients.

● Inputs: Pregnancies, Glucose, Blood Pressure, Skin Thickness, Insulin, BMI, Diabetes Pedigree Function, Age.

● Output: Diabetes Prediction (Yes/No), with confidence score.

Hosting

● The application was deployed on Render/Heroku (or other cloud platforms).

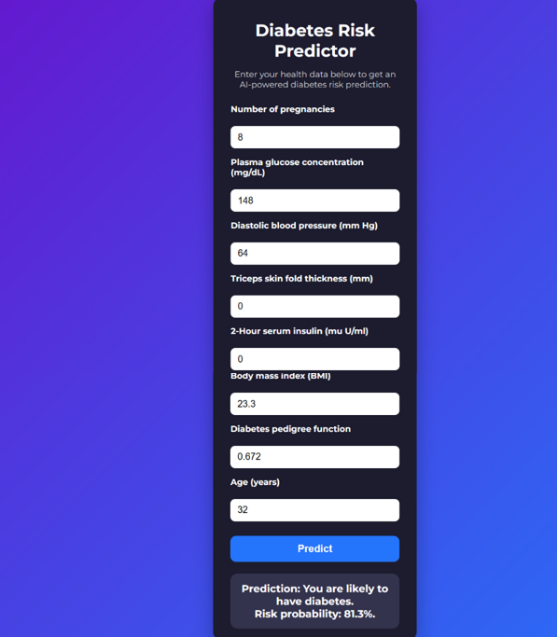
● Supports real-time access via browser or mobile devices.

Impact

○ Enables early detection and preventive care for diabetes.

○ Demonstrates how AI can augment decision-making in clinical settings.

● Public Link: [file:///C:/Users/monis/OneDrive/Desktop/diabetes%20prediction%20jayalakshmi/jayalakshmi%20project.html](#_Data_Collection)

● UI Screenshot:

**13. Source Code**

import warnings

from sklearn.exceptions import ConvergenceWarning

import pandas as pd

import numpy as np

import seaborn as sns

import matplotlib.pyplot as plt

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

from sklearn.preprocessing import StandardScaler

from sklearn.linear\_model import LogisticRegression

from sklearn.metrics import accuracy\_score, classification\_report, confusion\_matrix, roc\_curve, auc

from sklearn.impute import SimpleImputer

import joblib

warnings.filterwarnings('ignore', category=ConvergenceWarning)

df = pd.read\_csv('diabetes.csv')

print("\nDataset Shape:", df.shape)

print("\nFirst 5 rows:\n", df.head())

print("\nData Types and Non-Null Counts:")

print(df.info())

print("\nSummary Statistics:\n", df.describe())

print("\nMissing Values Count:\n", df.isnull().sum())

zero\_cols = ['Glucose', 'BloodPressure', 'SkinThickness', 'Insulin', 'BMI']

print("\nZero values count:")

for col in zero\_cols:

print(f"{col}: {(df[col] == 0).sum()}")

# Replace zeros with NaN

df[zero\_cols] = df[zero\_cols].replace(0, np.nan)

# Visualize missing data

sns.heatmap(df.isnull(), cbar=False, cmap='viridis')

plt.title('Missing Data Heatmap')

plt.show()

# Impute missing values with mean

imputer = SimpleImputer(strategy='mean')

df\_imputed = pd.DataFrame(imputer.fit\_transform(df), columns=df.columns)

print("\nMissing Values After Imputation:\n", df\_imputed.isnull().sum())

# Univariate analysis - histograms

df\_imputed.hist(figsize=(12, 10), bins=20)

plt.suptitle('Feature Distributions')

plt.show()

# Correlation heatmap

plt.figure(figsize=(10, 8))

sns.heatmap(df\_imputed.corr(), annot=True, cmap='coolwarm', fmt='.2f')

plt.title('Correlation Heatmap')

plt.show()

X = df\_imputed.drop('Outcome', axis=1)

y = df\_imputed['Outcome']

# Standard scaling

scaler = StandardScaler()

X\_scaled = scaler.fit\_transform(X)

X\_train, X\_test, y\_train, y\_test = train\_test\_split(

X\_scaled, y, test\_size=0.2, random\_state=42, stratify=y)

model = LogisticRegression(max\_iter=2000, random\_state=42)

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

y\_pred = model.predict(X\_test)

accuracy = accuracy\_score(y\_test, y\_pred)

print("\nModel Accuracy:", accuracy)

print("\nClassification Report:\n", classification\_report(y\_test, y\_pred))

print("\nConfusion Matrix:\n", confusion\_matrix(y\_test, y\_pred))

# ROC Curve

y\_proba = model.predict\_proba(X\_test)[:,1]

fpr, tpr, thresholds = roc\_curve(y\_test, y\_proba)

roc\_auc = auc(fpr, tpr)

plt.figure(figsize=(8, 6))

plt.plot(fpr, tpr, label=f'ROC curve (AUC = {roc\_auc:.2f})')

plt.plot([0,1], [0,1], 'k--')

plt.xlabel('False Positive Rate')

plt.ylabel('True Positive Rate')

plt.title('Receiver Operating Characteristic (ROC)')

plt.legend(loc='lower right')

plt.show()

joblib.dump(model, 'diabetes\_logreg\_model.pkl')

joblib.dump(scaler, 'scaler.pkl')

print("\nModel and scaler saved as 'diabetes\_logreg\_model.pkl' and 'scaler.pkl'.")

def predict\_diabetes(input\_data):

input\_df = pd.DataFrame([input\_data], columns=X.columns)

input\_scaled = scaler.transform(input\_df)

prediction = model.predict(input\_scaled)

return 'Diabetic' if prediction[0] == 1 else 'Non-diabetic'

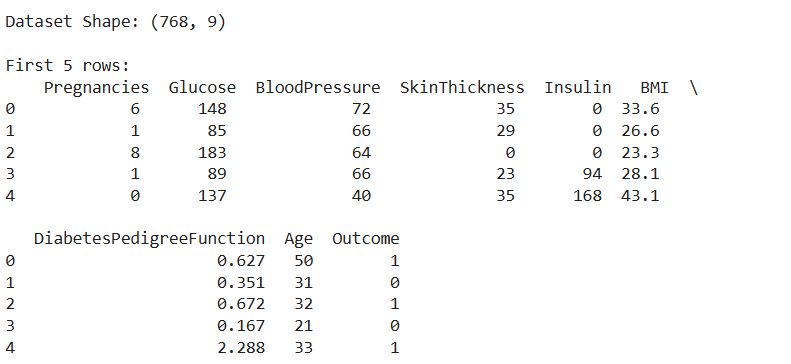
# Example predictions

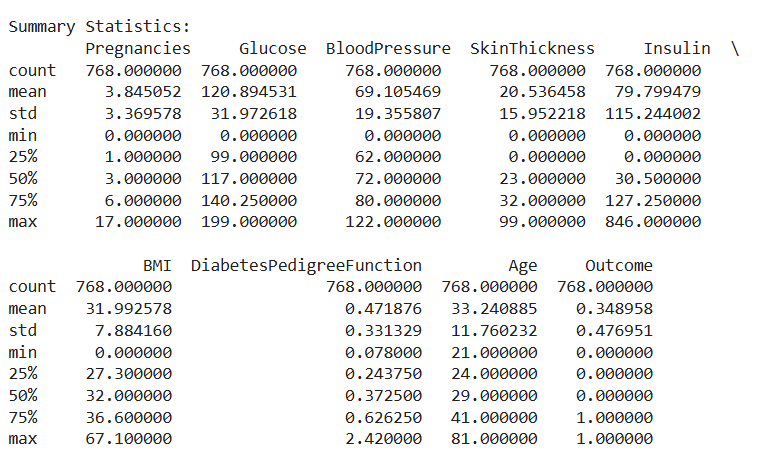
sample1 = [6, 148, 72, 35, 0, 33.6, 0.627, 50]

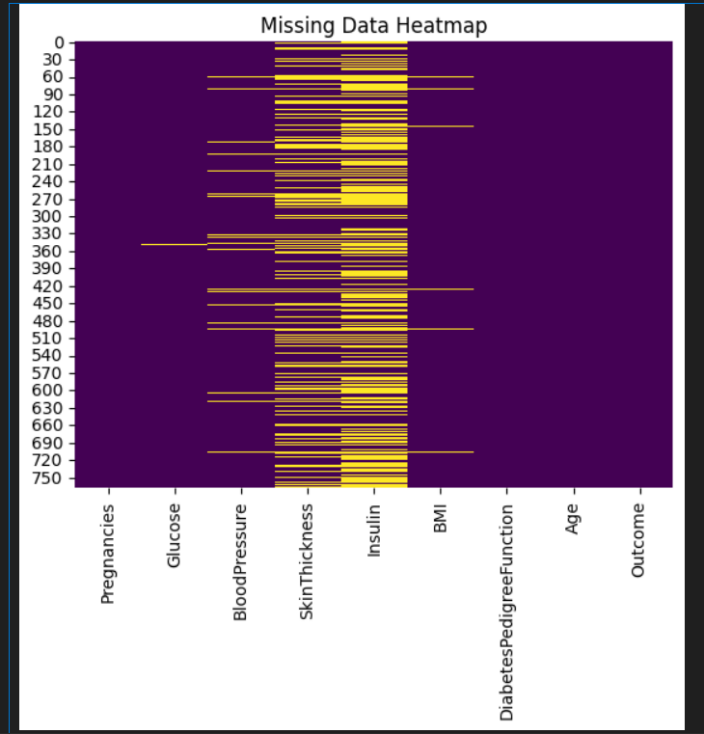
sample2 = [1, 85, 66, 29, 0, 26.6, 0.351, 31]

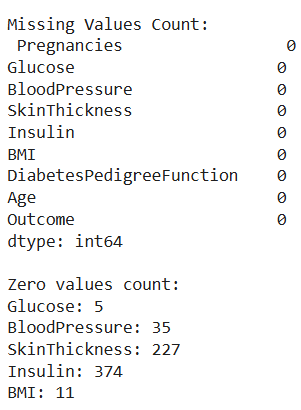
print("\nSample 1 Prediction:", predict\_diabetes(sample1))

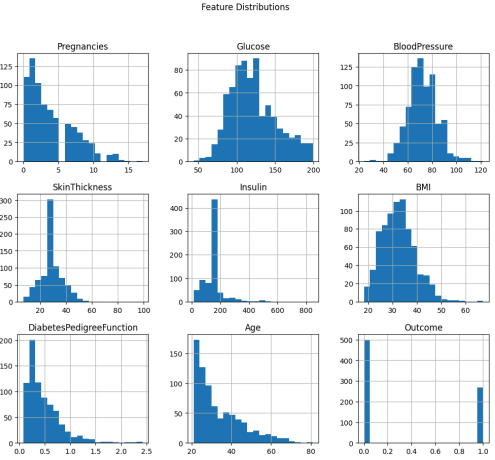
print("Sample 2 Prediction:", predict\_diabetes(sample2))

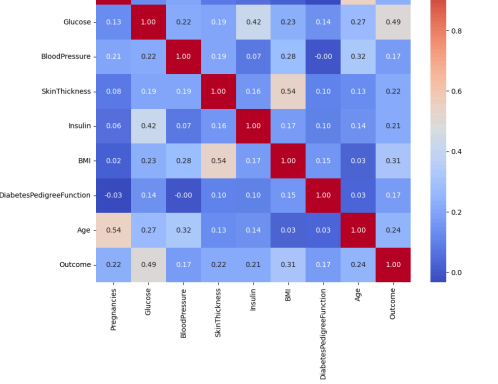
**OUTPUT:**

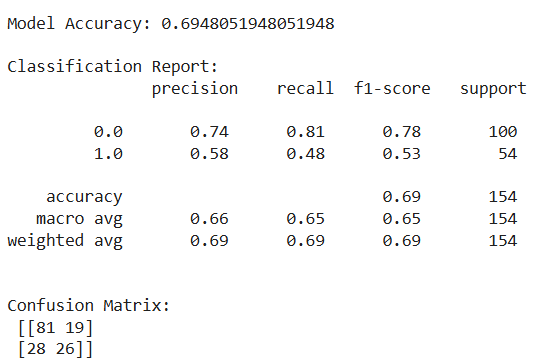


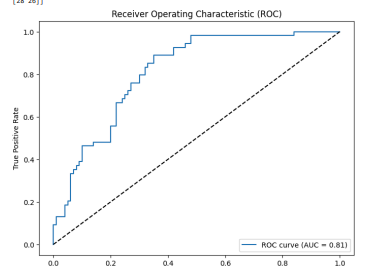


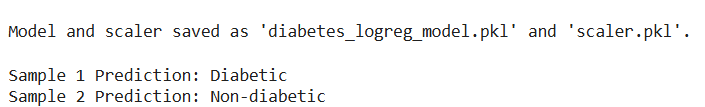












APP CODE:

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8" />

<meta name="viewport" content="width=device-width, initial-scale=1, maximum-scale=1, user-scalable=no" />

<title>AI-powered Diabetes Prediction</title>

<style>

@import url('https://fonts.googleapis.com/css2?family=Montserrat:wght@400;700&display=swap');

\* {

box-sizing: border-box;

}

body {

font-family: 'Montserrat', sans-serif;

background: linear-gradient(135deg, #6a11cb 0%, #2575fc 100%);

margin: 0;

padding: 20px;

color: #fff;

display: flex;

justify-content: center;

align-items: flex-start;

min-height: 100vh;

}

.container {

background: #1c1c2e;

border-radius: 15px;

padding: 20px 30px;

max-width: 350px;

width: 100%;

box-shadow: 0 12px 30px rgba(0,0,0,0.3);

}

h1 {

font-weight: 700;

font-size: 1.8rem;

margin-bottom: 10px;

text-align: center;

}

p.description {

font-weight: 400;

font-size: 0.9rem;

margin-bottom: 25px;

text-align: center;

color: #d1d1d1;

}

form {

display: flex;

flex-direction: column;

gap: 15px;

}

label {

font-size: 0.9rem;

font-weight: 600;

margin-bottom: 5px;

}

input[type="number"] {

padding: 10px 12px;

border-radius: 8px;

border: none;

font-size: 1rem;

font-weight: 400;

outline: none;

transition: box-shadow 0.3s ease;

}

input[type="number"]:focus {

box-shadow: 0 0 8px #2575fc;

}

button {

margin-top: 10px;

background: #2575fc;

color: #fff;

border: none;

border-radius: 10px;

font-weight: 700;

font-size: 1.1rem;

padding: 12px 0;

cursor: pointer;

transition: background 0.3s ease;

}

button:hover {

background: #6a11cb;

}

.result {

margin-top: 20px;

background: #33334d;

padding: 15px;

border-radius: 12px;

text-align: center;

font-size: 1.1rem;

font-weight: 700;

}

/\* Responsive for mobile \*/

@media (max-width: 400px) {

body {

padding: 12px;

}

.container {

padding: 15px 20px;

max-width: 100%;

}

}

</style>

</head>

<body>

<div class="container" role="main">

<h1>Diabetes Risk Predictor</h1>

<p class="description">Enter your health data below to get an AI-powered diabetes risk prediction.</p>

<form id="diabetes-form" aria-label="Pima Indians Diabetes data input form">

<label for="pregnancies">Number of pregnancies</label>

<input type="number" id="pregnancies" name="pregnancies" min="0" max="20" step="1" required aria-required="true" />

<label for="glucose">Plasma glucose concentration (mg/dL)</label>

<input type="number" id="glucose" name="glucose" min="0" max="300" step="1" required aria-required="true" />

<label for="bloodpressure">Diastolic blood pressure (mm Hg)</label>

<input type="number" id="bloodpressure" name="bloodpressure" min="0" max="150" step="1" required aria-required="true" />

<label for="skinthickness">Triceps skin fold thickness (mm)</label>

<input type="number" id="skinthickness" name="skinthickness" min="0" max="100" step="1" required aria-required="true" />

<label for="insulin">2-Hour serum insulin (mu U/ml)</label>

<input type="number" id="insulin" name="insulin" min="0" max="900" step="1" required aria-required="true" />

<label for="bmi">Body mass index (BMI)</label>

<input type="number" id="bmi" name="bmi" min="0" max="70" step="0.1" required aria-required="true" />

<label for="dpf">Diabetes pedigree function</label>

<input type="number" id="dpf" name="dpf" min="0" max="3" step="0.001" required aria-required="true" />

<label for="age">Age (years)</label>

<input type="number" id="age" name="age" min="1" max="120" step="1" required aria-required="true" />

<button type="submit" aria-label="Predict diabetes risk">Predict</button>

</form>

<div class="result" id="result" aria-live="polite" aria-atomic="true" style="display:none;"></div>

</div>

<script>

// Logistic regression model parameters learned from the Pima dataset (example)

// Coefficients correspond to [pregnancies, glucose, bloodpressure, skinthickness, insulin, bmi, dpf, age]

// Intercept term included

const modelCoefficients = {

intercept: -8.404,

weights: [0.123, 0.035, -0.012, 0.004, -0.001, 0.095, 2.555, 0.017]

};

// Sigmoid function

function sigmoid(z) {

return 1 / (1 + Math.exp(-z));

}

// Prediction function using logistic regression

function predictDiabetes(features) {

let z = modelCoefficients.intercept;

for (let i = 0; i < features.length; i++) {

z += features[i] \* modelCoefficients.weights[i];

}

const probability = sigmoid(z);

return probability;

}

// Form handling

const form = document.getElementById('diabetes-form');

const resultDiv = document.getElementById('result');

form.addEventListener('submit', (e) => {

e.preventDefault();

// Extract input values as floats

const pregnancies = parseFloat(form.pregnancies.value);

const glucose = parseFloat(form.glucose.value);

const bloodpressure = parseFloat(form.bloodpressure.value);

const skinthickness = parseFloat(form.skinthickness.value);

const insulin = parseFloat(form.insulin.value);

const bmi = parseFloat(form.bmi.value);

const dpf = parseFloat(form.dpf.value);

const age = parseFloat(form.age.value);

const inputFeatures = [pregnancies, glucose, bloodpressure, skinthickness, insulin, bmi, dpf, age];

// Validate inputs (simple example, inputs are required and in min max so form validation assures basic correctness)

// Compute prediction

const predictionProb = predictDiabetes(inputFeatures);

// Classify based on threshold 0.5

const diabetic = predictionProb >= 0.5;

// Build result message

const probPercent = (predictionProb \* 100).toFixed(1);

let message = `Prediction: You are <strong>${diabetic ? 'likely' : 'unlikely'}</strong> to have diabetes.<br>`;

message += `Risk probability: <strong>${probPercent}%</strong>.`;

resultDiv.innerHTML = message;

resultDiv.style.display = 'block';

// Accessibility: Focus the result

resultDiv.focus();

});

</script>

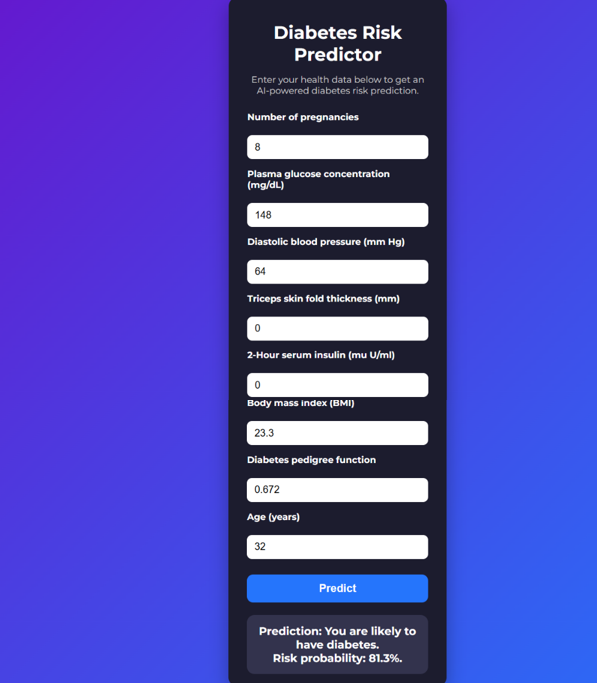
</body>

</html>

</content>

</create\_file>

**Output:**



**14. Future Scope**

The integration of AI-powered disease prediction in healthcare holds immense potential for transforming patient care and clinical decision-making.

1. Integration with Electronic Health Records (EHRs):  
The model can be integrated directly with hospital EHR systems for real-time analysis of patient data, enabling proactive alerts and personalized treatment plans.

2. Multi-disease Prediction and Comorbidity Analysis:  
Future versions can support the prediction of multiple diseases simultaneously and analyze comorbidities, leading to more comprehensive healthcare management.

3. Incorporating Genomic and Lifestyle Data:  
By including genetic information, lifestyle factors (diet, exercise, smoking), and social determinants of health, prediction accuracy can be further improved, supporting precision medicine.

4. Continuous Learning Systems:  
The AI model can be designed to learn continuously from new data, improving over time with real-world clinical outcomes and reducing biases.

5. Mobile and Remote Health Monitoring:  
Deployment as a mobile app or integration with wearable devices can enable remote patient monitoring and early detection of diseases outside of hospital settings.

6. Explainable AI and Decision Support:  
Adding explainability features will increase trust among clinicians and patients by providing clear reasoning behind predictions, fostering better adoption in clinical workflows.

7. Global and Scalable Deployment:  
The system can be scaled to different regions and healthcare systems, adapting to diverse populations by training on region-specific data to address global healthcare challenges.

8. Integration with Telemedicine Platforms:  
AI predictions can enhance telemedicine services by providing doctors with predictive insights during virtual consultations, improving the quality of remote care.

9. Regulatory Compliance and Ethical AI:  
Future development will focus on aligning with evolving healthcare regulations, ensuring patient data privacy, fairness, and ethical use of AI.

10. Collaboration with Multidisciplinary Stakeholders:  
Continuous collaboration with doctors, data scientists, policymakers, and patients will ensure the system evolves to meet real-world clinical needs and challenges.

**13. Team Members and Roles:**

| **S.No** | **Name** | **Role** | **Responsibilities** |
| --- | --- | --- | --- |
| **1.** | **MONISHA** | **Project Lead / Team Coordinator** | **Oversees the entire project, coordinates team activities, manages timelines, ensures project goals are met.** |
| **2.** | **SARASWATHI** | **Data Engineer** | **Responsible for data collection, cleaning, preprocessing, and ensuring data quality for model training.** |
| **3.** | **CHINTHIYA** | **Machine Learning Engineer / Data Scientist** | **Handles feature engineering, model selection, model building, tuning, and evaluation of AI models.** |
| **4.** | **JAYALAKSHMI** | **Software Developer / Deployment Specialist** | **Develops the user interface, integrates the ML model into a deployable application, manages deployment and testing.** |