**Diabetes Prediction Web App using Machine Learning**

**Aim:**

To predict the likelihood of diabetes by analyzing various health-related features such as glucose level, BMI, insulin, and more using classification techniques like logistic regression.

**Programs:**

**Model.py**

import pandas as pd import numpy as np import joblib

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, precision\_score, recall\_score

# Load dataset

df = pd.read\_csv("D:/ml\_assignment/diabetes\_dataset.csv")

# Drop unnecessary columns

columns\_to\_remove = ["LDL", "Hip Circumference", "WHR", "Diet Type"] existing\_columns = [col for col in columns\_to\_remove if col in df.columns] X = df.drop(columns=["Outcome"] + existing\_columns)

y = df["Outcome"]

# Train-test split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Scaling

scaler = StandardScaler()

X\_train\_scaled = scaler.fit\_transform(X\_train) X\_test\_scaled = scaler.transform(X\_test)

# Model

model = LogisticRegression() model.fit(X\_train\_scaled, y\_train)

# Metrics

y\_pred = model.predict(X\_test\_scaled) print("Accuracy:", accuracy\_score(y\_test, y\_pred))

print("Precision:", precision\_score(y\_test, y\_pred, zero\_division=0)) print("Recall:", recall\_score(y\_test, y\_pred, zero\_division=0))

# Save model and scaler joblib.dump(model, "logistic\_model.pkl") joblib.dump(scaler, "scaler.pkl")

joblib.dump(X.columns.tolist(), "features.pkl")

**app.py**

import streamlit as st import joblib

import numpy as np

# Load saved components

model = joblib.load("logistic\_model.pkl") scaler = joblib.load("scaler.pkl")

features = joblib.load("features.pkl")

# Streamlit config

st.set\_page\_config(page\_title="Diabetes Prediction App", page\_icon="◻", layout="centered") st.markdown("""

<style>

.stApp { background-color: #f0f2f6; }

.title {

text-align: center; font-size: 36px; font-weight: bold; color: #4CAF50;

}

.stButton>button {

background-color: #4CAF50; color: white;

font-size: 20px; border-radius: 8px; padding: 10px;

}

</style>

""", unsafe\_allow\_html=True)

# Title and instructions

st.markdown("<p class='title'>Diabetes Prediction App ◻</p>", unsafe\_allow\_html=True) st.write("Fill in the details below to predict your diabetes risk.")

# Input form

st.subheader("Enter Your Health Information") input\_data = []

col1, col2 = st.columns(2)

for i, col in enumerate(features): with col1 if i % 2 == 0 else col2:

value = st.number\_input(col, value=0.0) input\_data.append(value)

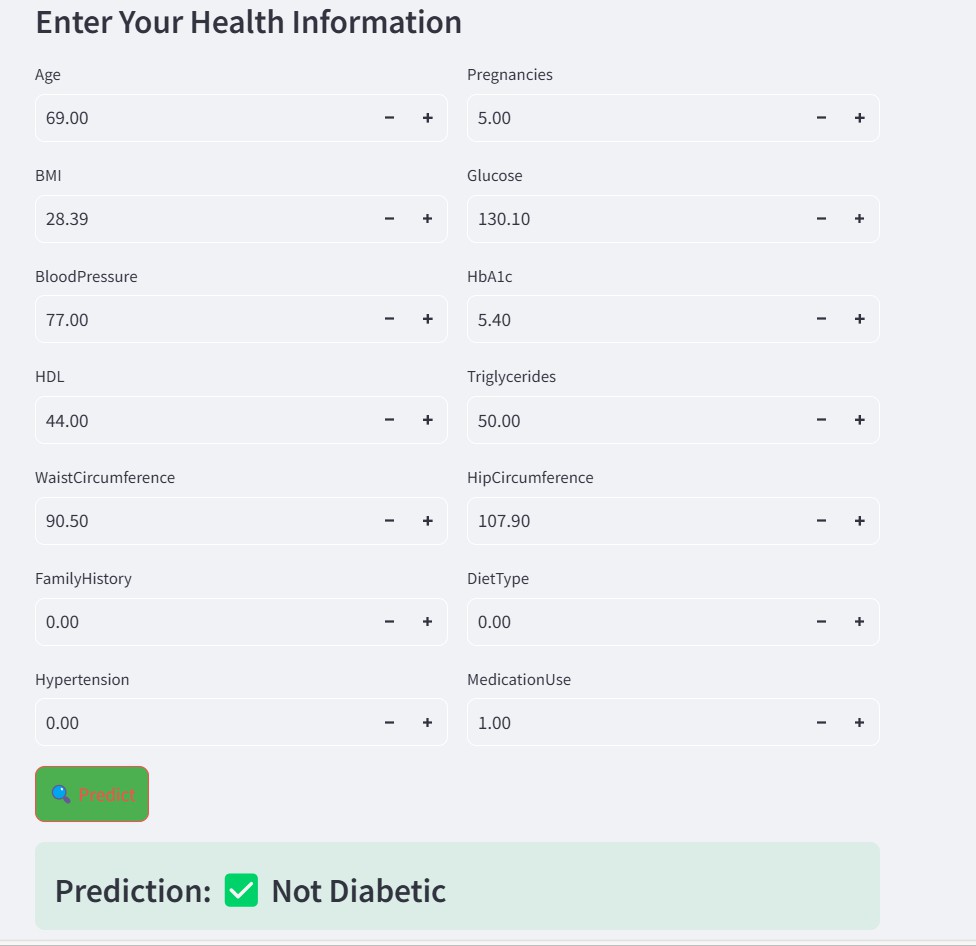
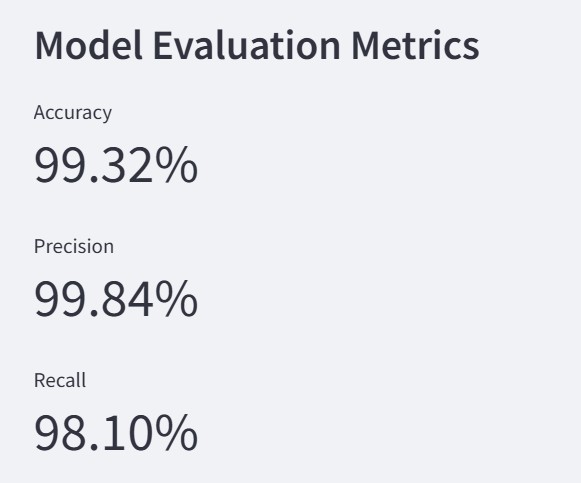
# Predict button

if st.button("Q Predict"):

input\_scaled = scaler.transform([input\_data])

prediction = "🛑 Diabetic" if model.predict(input\_scaled)[0] == 1 else "✅ Not Diabetic" st.success(f"### Prediction: {prediction}")

**output:**



**Link:**



https:/[/www.kaggle.com/datasets/hasibur013/diabetes-dataset](http://www.kaggle.com/datasets/hasibur013/diabetes-dataset)