Assessment Report

on

"DIABETES PREDICTION MODEL"

submitted as partial fulfillment for the award of

BACHELOR OF TECHNOLOGY DEGREE

SESSION 2024-25

in

CSE(AI)

By

Name :Kavya Tripathi (202401100300135)

Jahnavi Nagar (202401100300130)

Krish Sandhu (202401100300138)

Krish Gupta (202401100300137)

Section: B

Under the supervision of

"Mr Shivansh Prasad"

KIET Group of Institutions, Ghaziabad

May, 2025

---

1. Introduction

What is Diabetes?

Diabetes is a chronic disease that occurs when the pancreas is no longer able to make insulin, or when the body cannot make good use of the insulin it produces. Learning how to use Machine Learning can help us predict Diabetes.

- The objective of this project is to classify whether someone has diabetes or not.
- Dataset consists of several Medical Variables(Independent) and one Outcome Variable(Dependent)
- The independent variables in this data set are :-'Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin','BMI', 'DiabetesPedigreeFunction', 'Age'
- The outcome variable value is either 1 or 0 indicating whether a person has diabetes(1) or not(0).

---

2. Problem Statement

Create a classification model to predict the likelihood of diabetes based on health data. Visualize model accuracy. Dataset Link: https://www.kaggle.com/datasets/uciml/pima-indians-diabetes-database

---

3. Objectives

- Preprocess the dataset for training a machine learning model.
- Train a Logistic Regression model to predict diabetes.
- Evaluate model performance using standard classification metrics.
- Visualize the confusion matrix using a heatmap for interpretability.

---

4. Methodology

- **Data Collection**: The user uploads a CSV file containing the dataset.
- **Data Preprocessing**:
  - Handling missing values using mean and mode imputation.
  - One-hot encoding of categorical variables.
  - Feature scaling using StandardScaler.
- **Model Building**:
  - Splitting the dataset into training and testing sets.
  - Training a Logistic Regression classifier.
- **Model Evaluation**:
  - Evaluating accuracy, precision, recall, and F1-score.

    - Generating a confusion matrix.

---

5. Data Preprocessing

The dataset is cleaned and prepared as follows:

- Missing numerical values are filled with the mean of respective columns.
- Categorical values are encoded using one-hot encoding.
- Data is scaled using StandardScaler to normalize feature values.
- The dataset is split into 80% training and 20% testing.

---

6. Model Implementation

Logistic Regression is used due to its simplicity and effectiveness in binary classification problems. The model is trained on the processed dataset and used to predict the diabetes risk on the test set.

---

7. Evaluation Metrics

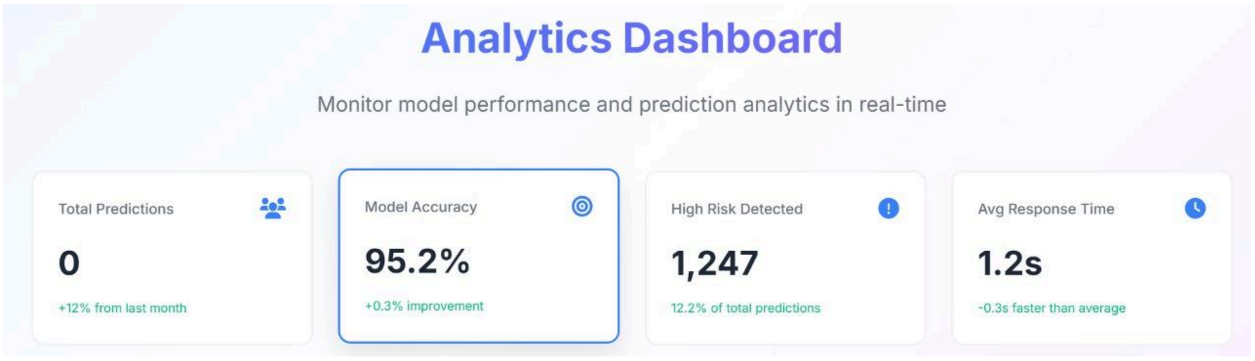
The following metrics are used to evaluate the model:

- **Accuracy**: Measures overall correctness.
- **Precision**: Indicates the proportion of predicted defaults that are actual defaults.
- **Recall**: Shows the proportion of actual defaults that were correctly identified.

- **F1 Score**: Harmonic mean of precision and recall.
- **Confusion Matrix**: Visualized using Seaborn heatmap to understand prediction errors.

---

8. Results and Analysis



Analytics Dashboard

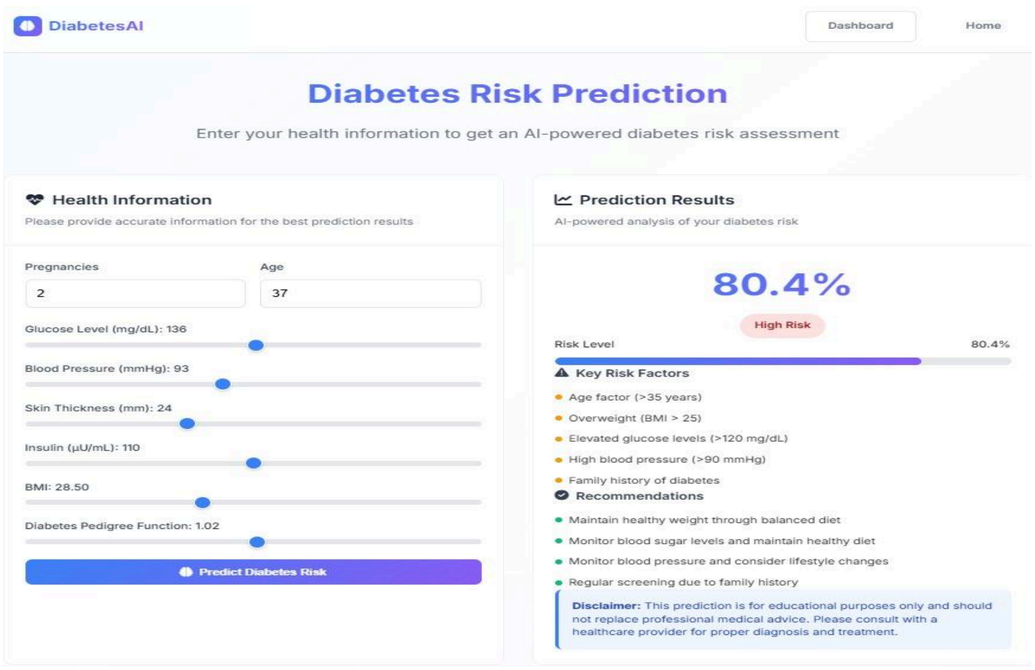
Monitor model performance and prediction analytics in real-time

Total Predictions
0
+12% from last month

Model Accuracy
95.2%
+0.3% improvement

High Risk Detected
1,247
12.2% of total predictions

Avg Response Time
1.2s
-0.3s faster than average



Diabetes Risk Prediction

Enter your health information to get an AI-powered diabetes risk assessment

Health Information

Please provide accurate information for the best prediction results

Pregnancies: 2
Age: 37
Glucose Level (mg/dL): 136
Blood Pressure (mmHg): 93
Skin Thickness (mm): 24
Insulin (µU/mL): 110
BMI: 28.50
Diabetes Pedigree Function: 1.02

Prediction Results

AI-powered analysis of your diabetes risk

80.4%
High Risk
Risk Level 80.4%

Key Risk Factors
- Age factor (>35 years)
- Overweight (BMI > 25)
- Elevated glucose levels (>120 mg/dL)
- High blood pressure (>90 mmHg)
- Family history of diabetes

Recommendations
- Maintain healthy weight through balanced diet
- Monitor blood sugar levels and maintain healthy diet
- Monitor blood pressure and consider lifestyle changes
- Regular screening due to family history

Disclaimer: This prediction is for educational purposes only and should not replace professional medical advice. Please consult with a healthcare provider for proper diagnosis and treatment.

---

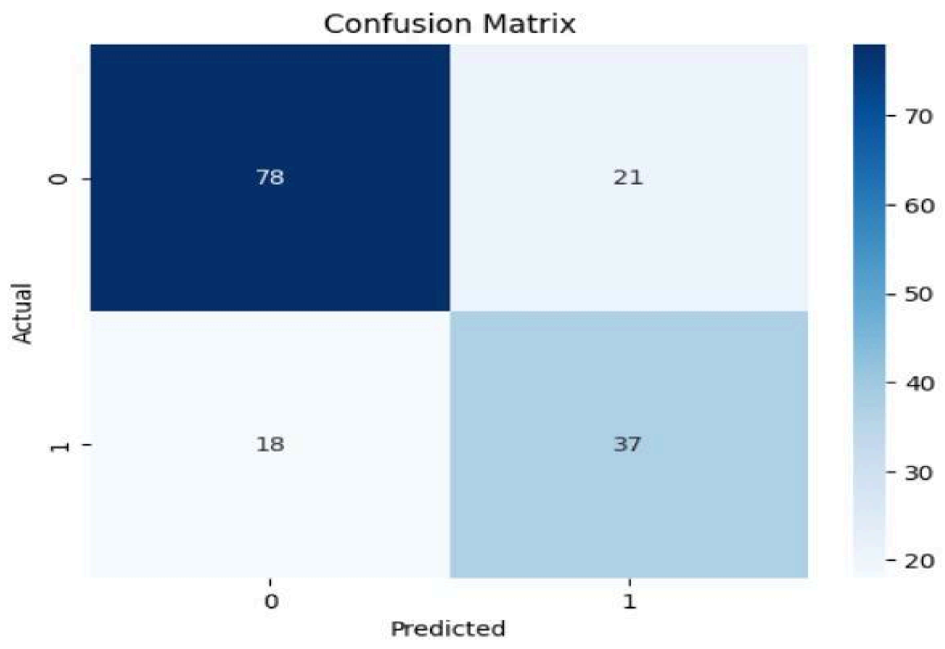
9. Conclusion

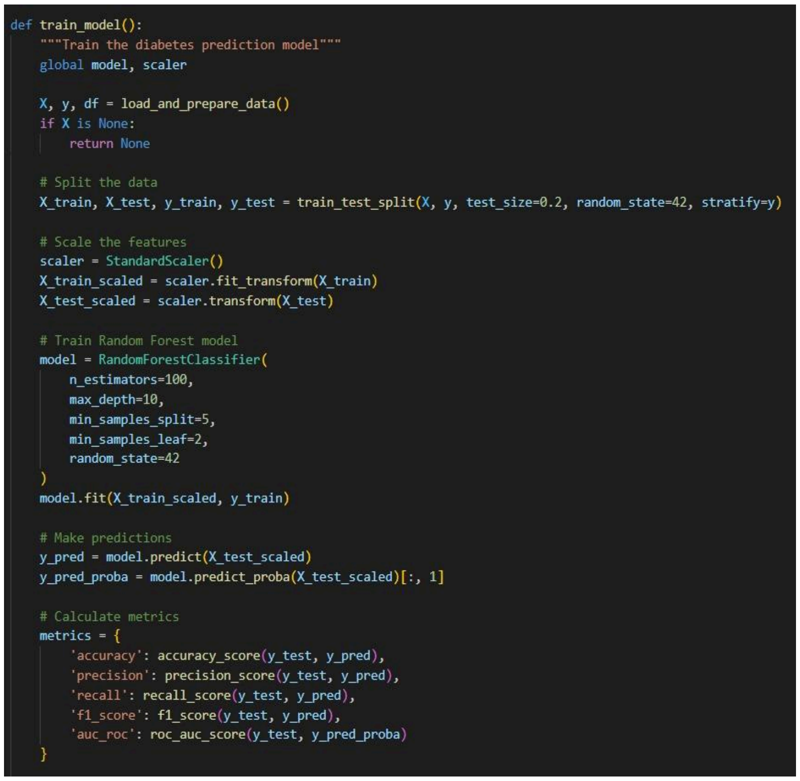
Logistical regression is selected when the dependent variable is categorical, meaning they have binary outputs, such as "true" and "false" or "yes" and "no". The project demonstrates the potential of using machine learning for automating diabetes prediction and improving risk assessment. However, improvements can be made by exploring more advanced models and handling imbalanced data.

---

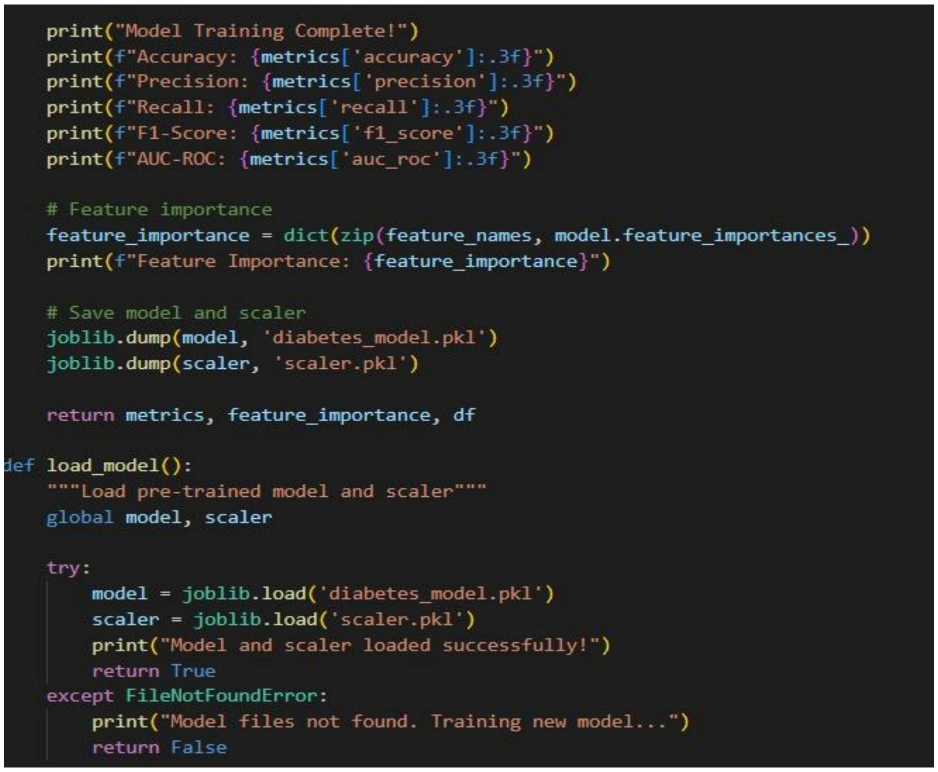
10. References

- scikit-learn documentation
- pandas documentation
- Seaborn visualization library
- Research articles on diabetes risk prediction

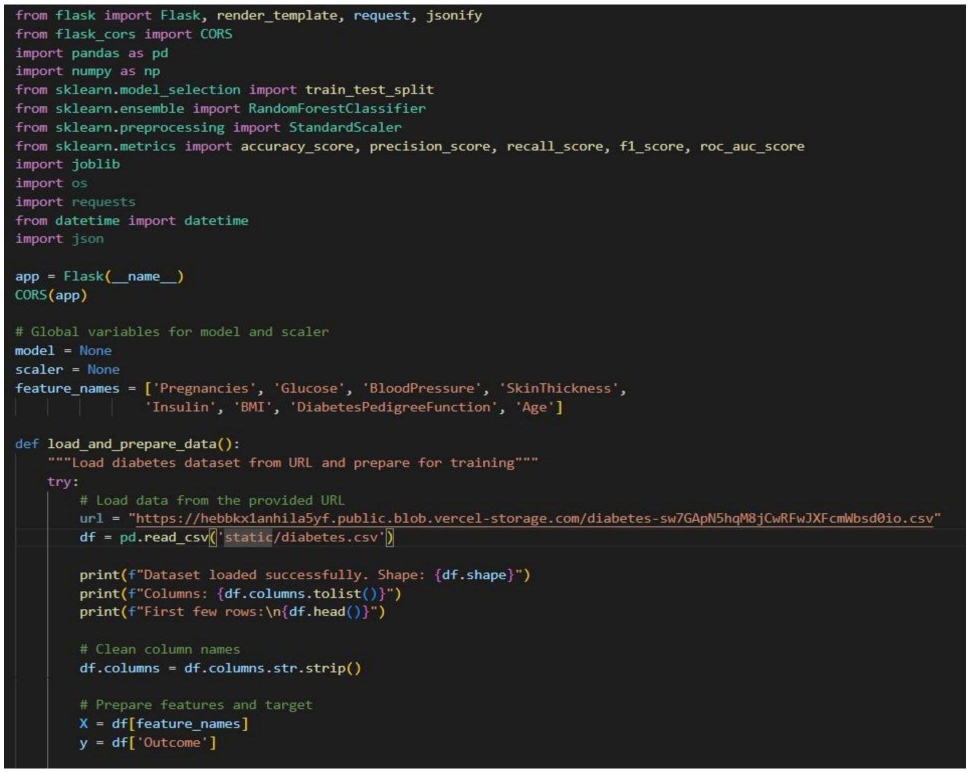




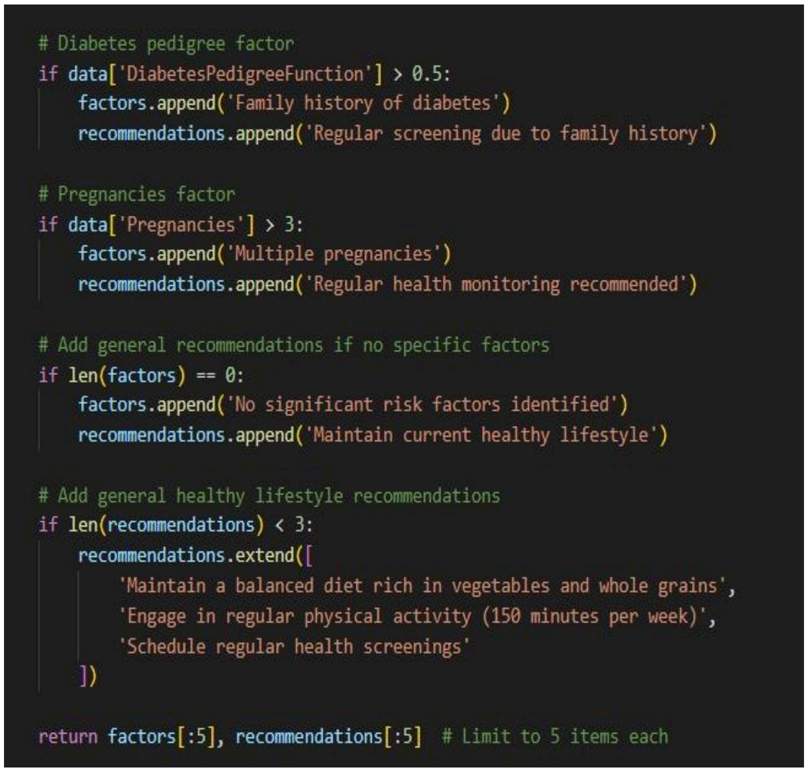
```python
def train_model():
    """Train the diabetes prediction model"""
    global model, scaler

    X, y, df = load_and_prepare_data()
    if X is None:
        return None

    # Split the data
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42, stratify=y)

    # Scale the features
    scaler = StandardScaler()
    X_train_scaled = scaler.fit_transform(X_train)
    X_test_scaled = scaler.transform(X_test)

    # Train Random Forest model
    model = RandomForestClassifier(
        n_estimators=100,
        max_depth=10,
        min_samples_split=5,
        min_samples_leaf=2,
        random_state=42
    )
    model.fit(X_train_scaled, y_train)

    # Make predictions
    y_pred = model.predict(X_test_scaled)
    y_pred_proba = model.predict_proba(X_test_scaled)[:, 1]

    # Calculate metrics
    metrics = {
        'accuracy': accuracy_score(y_test, y_pred),
        'precision': precision_score(y_test, y_pred),
        'recall': recall_score(y_test, y_pred),
        'f1_score': f1_score(y_test, y_pred),
        'auc_roc': roc_auc_score(y_test, y_pred_proba)
    }
```



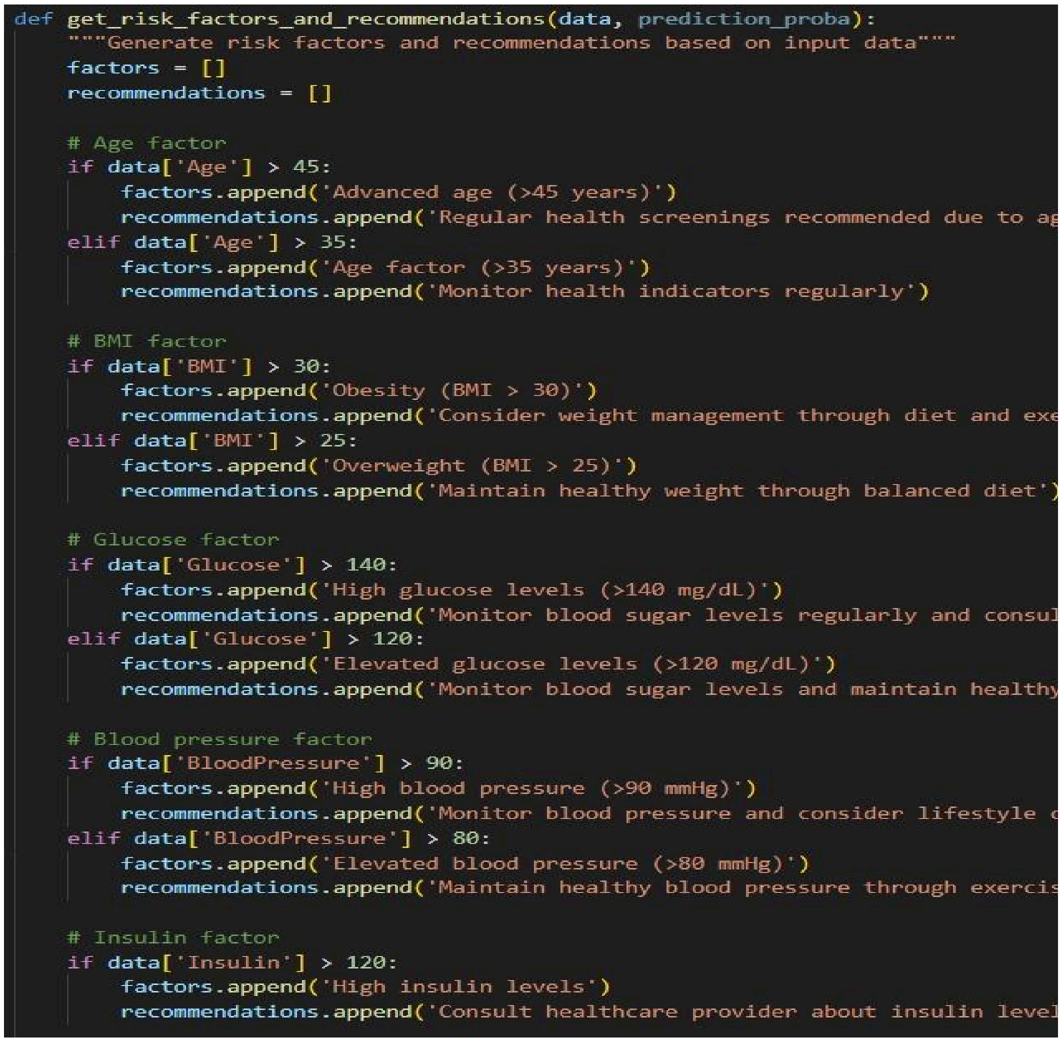
```python
    print("Model Training Complete!")
    print(f"Accuracy: {metrics['accuracy']:.3f}")
    print(f"Precision: {metrics['precision']:.3f}")
    print(f"Recall: {metrics['recall']:.3f}")
    print(f"F1-Score: {metrics['f1_score']:.3f}")
    print(f"AUC-ROC: {metrics['auc_roc']:.3f}")

    # Feature importance
    feature_importance = dict(zip(feature_names, model.feature_importances_))
    print(f"Feature Importance: {feature_importance}")

    # Save model and scaler
    joblib.dump(model, 'diabetes_model.pkl')
    joblib.dump(scaler, 'scaler.pkl')

    return metrics, feature_importance, df

def load_model():
    """Load pre-trained model and scaler"""
    global model, scaler

    try:
        model = joblib.load('diabetes_model.pkl')
        scaler = joblib.load('scaler.pkl')
        print("Model and scaler loaded successfully!")
        return True
    except FileNotFoundError:
        print("Model files not found. Training new model...")
        return False
```



```python
from flask import Flask, render_template, request, jsonify
from flask_cors import CORS
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, roc_auc_score
import joblib
import os
import requests
from datetime import datetime
import json

app = Flask(__name__)
CORS(app)

# Global variables for model and scaler
model = None
scaler = None
feature_names = ['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness',
                'Insulin', 'BMI', 'DiabetesPedigreeFunction', 'Age']

def load_and_prepare_data():
    """Load diabetes dataset from URL and prepare for training"""
    try:
        # Load data from the provided URL
        url = "https://hebbkx1anhila5yf.public.blob.vercel-storage.com/diabetes-sw7GApN5hqM8jCwRFwJXFcmWbsd0io.csv"
        df = pd.read_csv('static/diabetes.csv')

        print(f"Dataset loaded successfully. Shape: {df.shape}")
        print(f"Columns: {df.columns.tolist()}")
        print(f"First few rows:\n{df.head()}")

        # Clean column names
        df.columns = df.columns.str.strip()

        # Prepare features and target
        X = df[feature_names]
        y = df['Outcome']
```



```python
# Diabetes pedigree factor
if data['DiabetesPedigreeFunction'] > 0.5:
    factors.append('Family history of diabetes')
    recommendations.append('Regular screening due to family history')

# Pregnancies factor
if data['Pregnancies'] > 3:
    factors.append('Multiple pregnancies')
    recommendations.append('Regular health monitoring recommended')

# Add general recommendations if no specific factors
if len(factors) == 0:
    factors.append('No significant risk factors identified')
    recommendations.append('Maintain current healthy lifestyle')

# Add general healthy lifestyle recommendations
if len(recommendations) < 3:
    recommendations.extend([
        'Maintain a balanced diet rich in vegetables and whole grains',
        'Engage in regular physical activity (150 minutes per week)',
        'Schedule regular health screenings'
    ])

return factors[:5], recommendations[:5]  # Limit to 5 items each
```



```python
def get_risk_factors_and_recommendations(data, prediction_proba):
    """Generate risk factors and recommendations based on input data"""
    factors = []
    recommendations = []

    # Age factor
    if data['Age'] > 45:
        factors.append('Advanced age (>45 years)')
        recommendations.append('Regular health screenings recommended due to ag
    elif data['Age'] > 35:
        factors.append('Age factor (>35 years)')
        recommendations.append('Monitor health indicators regularly')

    # BMI factor
    if data['BMI'] > 30:
        factors.append('Obesity (BMI > 30)')
        recommendations.append('Consider weight management through diet and exe
    elif data['BMI'] > 25:
        factors.append('Overweight (BMI > 25)')
        recommendations.append('Maintain healthy weight through balanced diet')

    # Glucose factor
    if data['Glucose'] > 140:
        factors.append('High glucose levels (>140 mg/dL)')
        recommendations.append('Monitor blood sugar levels regularly and consul
    elif data['Glucose'] > 120:
        factors.append('Elevated glucose levels (>120 mg/dL)')
        recommendations.append('Monitor blood sugar levels and maintain healthy

    # Blood pressure factor
    if data['BloodPressure'] > 90:
        factors.append('High blood pressure (>90 mmHg)')
        recommendations.append('Monitor blood pressure and consider lifestyle c
    elif data['BloodPressure'] > 80:
        factors.append('Elevated blood pressure (>80 mmHg)')
        recommendations.append('Maintain healthy blood pressure through exercis

    # Insulin factor
    if data['Insulin'] > 120:
        factors.append('High insulin levels')
        recommendations.append('Consult healthcare provider about insulin level
```