AI BASED DIABETICES PREDICTION SYSTEM

PHASE 4

Development Part 2

In this part you will continue building your project.

In this phase, we'll continue building the diabetes prediction system by:

Selecting a machine learning algorithm

Training the model

Evaluating its performance

STEP BY STEP PROCEDURE:

Step 1: Data Collection

- Obtain a reliable dataset that contains relevant information for diabetes prediction. This dataset should include features (attributes) and a target variable (indicating diabetes presence or absence).

ATTRIBUTES USED:

Pregnancies,Glucose,BloodPressure,SkinThickness,Insulin,BMI,DiabetesPedigreeFunction,Age,Outcome

Step 2: Data Pre-processing

- Clean and pre-process the dataset to ensure it's suitable for building a predictive model.

- Handle missing values: Impute or remove rows/columns with missing data.

- Handle outliers: Identify and address outliers if present.

- Check for data quality and consistency.

Step 3: Feature Selection

- Determine which features are relevant for predicting diabetes.

- Use feature selection techniques like correlation analysis, feature importance, or domain knowledge to select the most informative features.

- Choose a subset of features that have the most significant impact on the prediction.

Step 4: Data Splitting

- Split the dataset into a training set and a testing set.

- The training set is used to train your predictive model.

- The testing set is used to evaluate the model's performance.

Step 5: Feature Engineering

- Create new features or transform existing ones to make them more informative.

- Normalize or standardize numeric features.

- Encode categorical variables if necessary.

Step 6: Model Selection

- Choose an appropriate machine learning or AI model for your diabetes prediction task.

- Common models include logistic regression, decision trees, random forests, support vector machines, or deep learning models like neural networks.

Step 7: Model Training

process the dataset to ensure it's suitable for building a predictive model.

Handle missing values: Impute or remove rows/columns with missing data.

Handle outliers: Identify and address outliers if present.

Check for data quality and consistency.

DATA LOADING:

We have collected the reliable dataset containing patient health records, with labels indicating diabetics prediction. This dataset contains the sequence of medical reports of patients.

1. Import Libraries:

Import the required python libraries ,especially Pandas for data manipulation and analysis.

Import necessary libraries

import pandas as pd

import numpy as np

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

from sklearn.preprocessing import StandardScaler

from sklearn.impute import SimpleImputer

2.Load the Dataset:

Load the diabetes dataset (replace with your actual dataset path)

data= pd.read\_csv("diabetes.csv")

3.Split Data into Features (X) labels (Y):

X = data.drop('Diabetic', axis=1)

y = data['Diabetic']

4.Split the data into training and testing sets

X = selected\_features # Features

y = data['target'] # Target variable

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

**DATA PREPROCESSING:**

5.Data cleaning:

# Handling Missing Data

imputer = SimpleImputer(strategy='mean')

data = imputer.fit\_transform(data) # Fill missing values with mean values

6.Transformation (if needed)

# You may need to one-hot encode categorical columns or apply other transformations here.

**PYTHON PROGRAM USING THE RANDOM FOREST CLASSIFIER:**

mport necessary libraries

import pandas as pd

import numpy as np

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

from sklearn.preprocessing import StandardScaler

from sklearn.impute import SimpleImputer

# Load the dataset

data = pd.read\_csv('diabetes\_dataset.csv')

# Explore the dataset

print(data.head()) # Display the first few rows of the dataset

print(data.info()) # Get information about the dataset, including missing values

# Handling Missing Data

imputer = SimpleImputer(strategy='mean')

data = imputer.fit\_transform(data) # Fill missing values with mean values

# Data Transformation (if needed)

# You may need to one-hot encode categorical columns or apply other transformations here.

# Feature Selection (you can use your own criteria for feature selection)

selected\_features = data[['feature1', 'feature2', 'feature3']]

# Split the data into training and testing sets

X = selected\_features # Features

y = data['target'] # Target variable

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

# Scaling and Normalization (if needed)

scaler = StandardScaler()

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

X\_test = scaler.transform(X\_test)

MACHINE LEARNING ALGORITHMS:

1. Logistic Regression: This is a simple algorithm often used for binary classification tasks in diabetes prediction.

2. Decision Trees: Decision tree algorithms, such as CART or Random Forest, can be used to create a model that predicts the likelihood of diabetes based on input features.

3. Support Vector Machines (SVM): SVM is useful for both classification and regression tasks, and it can be applied to predict diabetic outcomes.

4. Neural Networks: Deep learning models, like artificial neural networks, can be employed for diabetes prediction, especially for more complex and high-dimensional data.

IOT DEVICES

1. Continuous Glucose Monitoring (CGM) Sensors: IoT-enabled CGM sensors can be worn by diabetic patients to monitor their blood glucose levels in real time. These sensors can send data to a central platform through the internet.

2. IoT Data Collection: Data from various IoT devices, such as CGM sensors, insulin pumps, activity trackers, and dietary intake sensors, can be collected and transmitted to a central server.

3. Data Pre-processing: The collected data can be pre-processed to clean and prepare it for analysis. This may include handling missing data, normalizing values, and aligning time stamps.

EVALUVATING THE PERFORMANCE:

Precision-Recall Curve: Employ a Precision-Recall curve for imbalanced datasets, plotting precision against recall at various thresholds.

Cross-Validation: Implement cross-validation techniques, like k-fold cross-validation, to assess generalization and reduce overfitting risk.

Domain Expert Review: Seek domain experts' input to assess clinical relevance and practicality of the model's predictions.

OUTPUT: