AI-Based Diabetes Prediction System Implementation Procedure

Introduction

The goal of this document is to outline the steps for transforming the design of the AI-based diabetes prediction system, as previously defined, into an innovative solution.

Building a complete AI-based diabetes prediction system involves multiple steps, including obtaining the dataset, preprocessing the data, and building a predictive model. Below, I'll provide a step-by-step guide with code examples for each stage of the project.

Prerequisite step :

As you very well know, there are a quite a few programming languages (like Python, R) and tools that can be used to machine learning projects like this. But the most popular one is python and I going to use Python to do this project.

Initially, I will setup the project environment, here are the steps I will follow :

- Download **Miniconda** (an environment management system for installing and maintaining software packages), we use this to ease the process of installing necessary libraries for our project.
- Then I will create a repository and create a conda environment along with the necessary dependencies – **Pandas, NumPy, Matplotlib and Scikit.**

```
conda create --prefix ./env pandas numpy matplotlib scikit-learn
```

- Activate the environment and install Jupyter notebook – an IDE to run our project.

```
conda activate c:\Users\tvaru\Desktop\sample_project_1\env
```

- In **Jupyter notebook**, I will create a new python file and began the coding process.

```
conda install jupyter
```

Step 1: Obtaining the Dataset

The first step in any machine learning project is to obtain a dataset to work with. There are many online resources available to find the datasets suitable for our project. One such popular platform that hosts datasets is **Kaggle**. Since the goal of our project is to predict Diabetes, I search for diabetes patient's dataset, which is readily available as were already many such similar projects.

1. Go to the Kaggle dataset page you mentioned: Diabetes Data Set.
2. Click the "Download" button to get the dataset files.
3. Unzip the downloaded files to a directory on your local machine into your project repository.

This dataset has the following attributes : Pregnancies, Glucose, Blood Pressure, Skin Thickness, Insulin, BMI, Diabetes Pedigree Function and Age, It also contains the Outcome label which tells whether the patient has diabetes or not. Since there is a class label present by default, this dataset can be studied using Classification based algorithms.

Step 2: Data Preprocessing

Now that we have the dataset, we need to prepare it for analysis and modeling. This involves cleaning and organizing the data. Now, since I took this dataset from Kaggle, it is already fully pre-processed (i.e.) there are no missing values, all features have numerical values.

```
# Import necessary libraries

import pandas as pd

import numpy as np


# Load the dataset

data = pd.read_csv('diabetes_data.csv')


# Check the first few rows of the dataset

print(data.head())


# Check for missing values

print(data.isnull().sum())


# Split the data into features (X) and target variable (y)

X = data.drop('diabetes', axis=1)
```

```
y = data['diabetes']
```

In this step, we load the dataset, inspect the first few rows, and check for missing values. Since the data set contains the target class label, I can go ahead with classification or regression based approaches to build the model.

Step 3: Data Exploration (Optional)

Data exploration helps us understand the dataset better. We can create visualizations to gain insights into the data. We can visualize the data using the graphing library – **matplotlib**. I use the pairplot to understand the relationship between variables.

```
import matplotlib.pyplot as plt

import seaborn as sns


# Pairplot to visualize relationships between variables

sns.pairplot(data, hue='diabetes', diag_kind='kde')

plt.show()
```

Step 4: Data Splitting

The necessary step is to split the dataset into training and testing sets to train and evaluate our model. I use **Scikit's train_test_split** method to achieve this, the code for which as follows.

```
from sklearn.model_selection import train_test_split


# Split the data into training and testing sets (80% train, 20% test)

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)
```

Step 5: Model Building (Logistic Regression)

Now, we'll build a logistic regression model to predict diabetes. As previously mentioned in the report, I am going to use Logistic Regression to build the model since it is best suitable for datasets with class label, and also the class label has binary values, which in our case is true, since we have 0 or 1 for the Outcome.

The **Logistic Regression** method is found Scikit's linear_model sub module. After building and fitting the model to the training dataset, we predict the result of the testing dataset. And I evaluate the prediction using measures like accuracy, confusion matrix and classification report which standard metrics of classification based models.

```
from sklearn.preprocessing import StandardScaler

from sklearn.linear_model import LogisticRegression

from sklearn.metrics import accuracy_score, classification_report,
confusion_matrix


# Data preprocessing: Standardize features

scaler = StandardScaler()

X_train = scaler.fit_transform(X_train)

X_test = scaler.transform(X_test)


# Create and train the logistic regression model

model = LogisticRegression()

model.fit(X_train, y_train)


# Make predictions on the test set

y_pred = model.predict(X_test)


# Evaluate the model

accuracy = accuracy_score(y_test, y_pred)

confusion = confusion_matrix(y_test, y_pred)

report = classification_report(y_test, y_pred)


print(f'Accuracy: {accuracy}')
print(f'Confusion Matrix:\n{confusion}')
print(f'Classification Report:\n{report}')
```

Step 6: Model Deployment (Optional)

To deploy the model, you would need to integrate it into a healthcare system, application, or web platform, which involves additional coding and considerations beyond the scope of this explanation.