**AI BASED DIABETES PREDICTION SYSTEM**

**Phase-5**

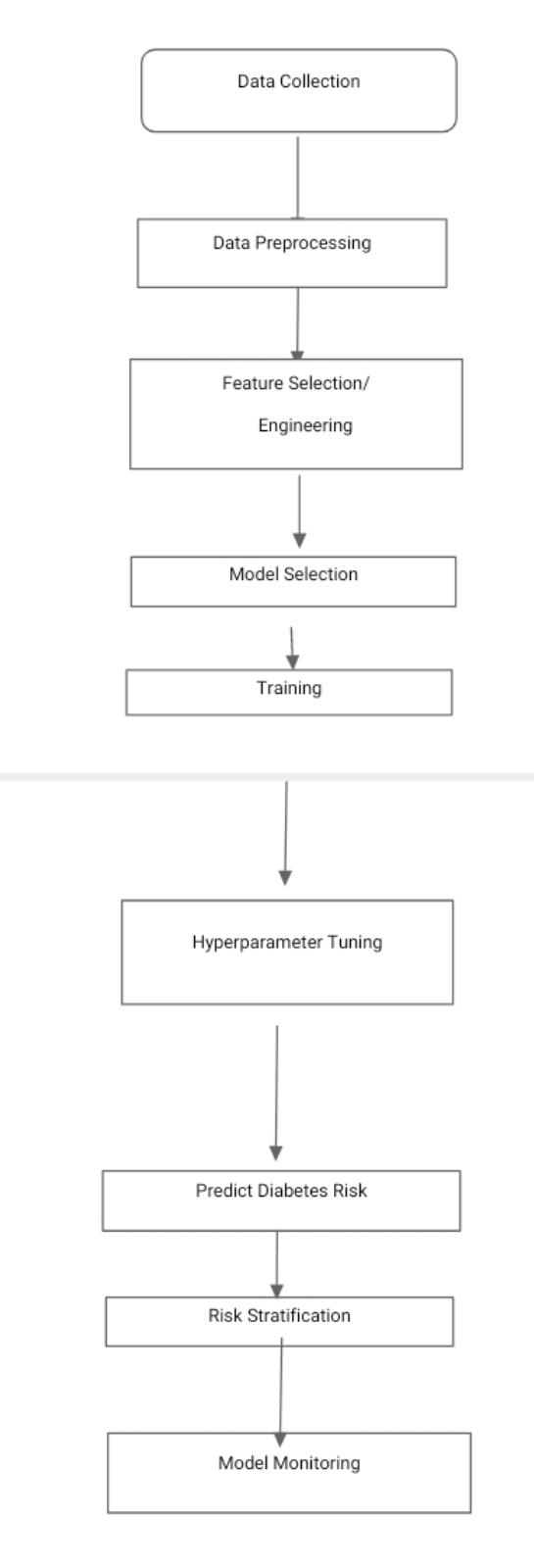
**Problem statement:**

The problem statement is to determine which algorithm is more accurate in predicting diabetes. The methodology involves implementing the SVM and decision tree algorithms on the dataset and evaluating their performance using metrics such as accuracy, precision, and recall.

**Design thinking process**

Design thinking is a human-centered approach to innovation that integrates the needs of people, the possibilities of technology, and the requirements for business success. When applying the design thinking process to the development of an AI-based diabetes prediction system, the following steps can be helpful:

1. Empathize: Understand the needs of potential users, such as patients, healthcare providers, and researchers, by conducting interviews, observations, and surveys to comprehend their experiences and challenges with diabetes.
2. Define: Clearly define the problem and the goals of the AI-based diabetes prediction system, taking into account the specific pain points and requirements identified during the empathize stage.
3. Ideate: Generate a variety of potential solutions and features that could address the identified problem. Encourage a diverse team to brainstorm and come up with innovative ideas for the system, considering both technological capabilities and user needs.
4. Prototype: Develop a preliminary version of the AI-based diabetes prediction system to visualize and demonstrate the proposed solution. Create prototypes that allow for early testing and feedback from users, enabling iterative improvements based on their input.
5. Test: Gather feedback from potential users and stakeholders on the prototype to evaluate its effectiveness in addressing the identified problem. Use this feedback to refine and enhance the system, ensuring that it aligns with user expectations and needs.
6. Implement: Transform the refined prototype into a fully functional AI-based diabetes prediction system, integrating advanced algorithms and data processing techniques to enable accurate and timely predictions for users.
7. Evaluate: Continuously monitor the performance and impact of the AI-based diabetes prediction system, collecting data on its effectiveness and user satisfaction. Use this data to make necessary adjustments and updates to ensure the system remains efficient and user-friendly over time.

**INNOVATION**

**Data collection and Integration:**

Gather a comprehensive dataset that includes various health-related features such as age, gender, BMI, blood pressure, family history, glucose levels, cholesterol levels, lifestyle factors (diet, exercise), and medical history.

Integrate data from various sources, including electronic health records, wearable devices, and patient self-respect

**Data Preprocessing and Cleaning:**

Clean and preprocess the data by addressing missing values, outliers, and inconsistencies.

Normalize or scale the features to ensure they have a consistent range.

**Feature Selection and Engineering:**

Identify the most relevant features using techniques like correlation analysis, mutual information, or feature importance scores from machine learning models.

Create new features through engineering, such as calculating the body adiposity index (BAI) or adding interaction terms.

**Model Selection and Development:**

Choose appropriate machine learning algorithms, which may include logistic regression, decision trees, random forests, support vector machines, or deep learning models like neural networks.

Develop an ensemble of models or use a combination of techniques to improve predictive accuracy.

**Training and Validation:**

Split the dataset into training, validation, and testing sets.

Train the selected models on the training data and validate their performance on the validation set.

Use techniques like cross-validation to tune hyperparameters and optimize model performance.

**Model Interpretability**:

Employ interpretability techniques like SHAP (Shapley Additive exPlanations) values or LIME (Local Interpretable Model-agnostic Explanations) to explain the model’s predictions, making it understandable to healthcare professionals and patients.

**Risk Stratification:**

Categorize individuals into risk groups (e.g., low, moderate, high risk) based on model predictions.

Provide personalized risk scores and recommendations for each group

**Deployment and Integretion**:

Integrate the AI system into healthcare facilities, electronic health records (HER) systems, or mobile health apps for widespread use.

Ensure compliance with regulatory requirements and data privacy standards (e.g., HIPAA)

**Monitoring and Updates:**

Continuously monitor the model’s performance in real-world settings.

Update the model periodically to adapt to changing patient demographics, treatment guidelines, and emerging medical knowledge.

**Patient Engagement:**

Develop user-friendly interfaces and patient engagement strategies to encourage individuals to participate in diabetes risk assessment and management.

**Healthcare professional development:**

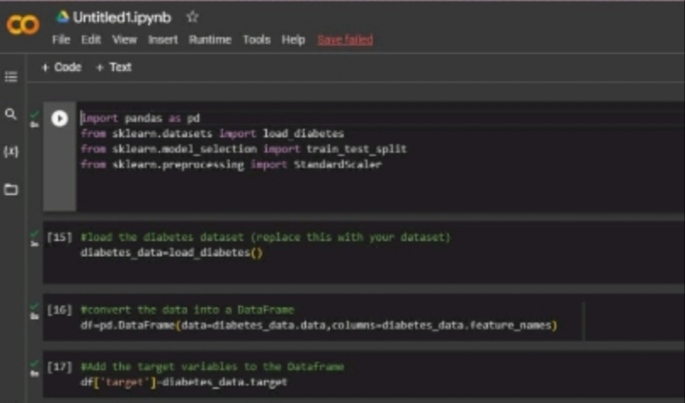
Facilitate collaboration between AI systems and healthcare professionals to provide comprehensive care and treatment recommendations.

**Ethical Consideration:**

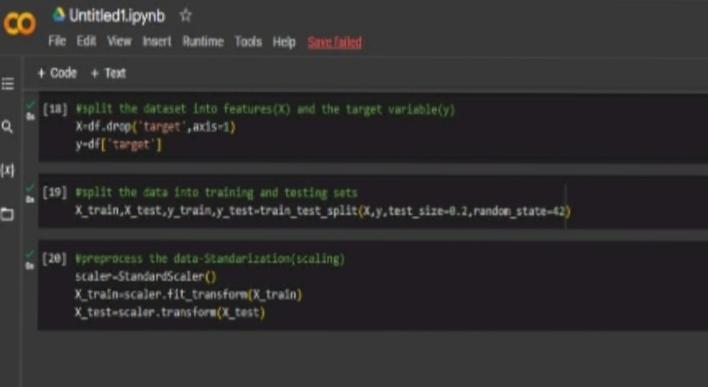
Address ethical concerns related to data privacy, informed consent, bias, and fairness in AI prediction

**Loading the Dataset:**

The first step in building the diabetes prediction system is to load the dataset. Typically, the dataset will be in a file format such as CSV. In Jupyter Notebook, you can use pandas library to load the dataset into a Dataframe

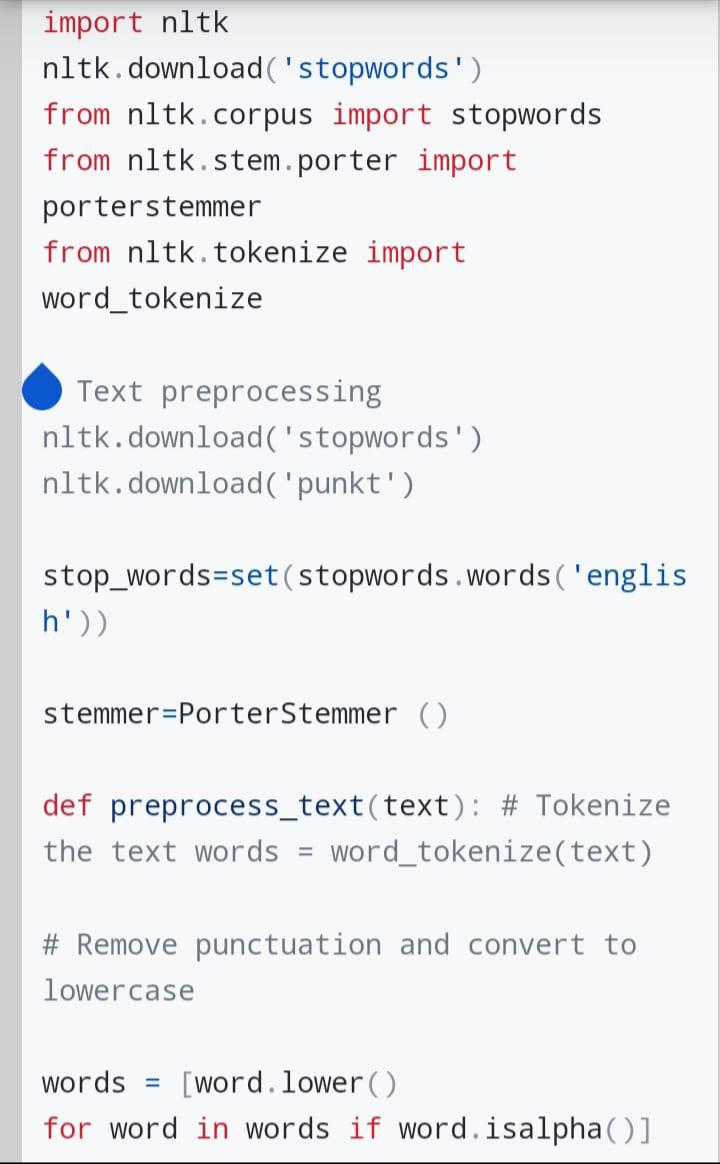


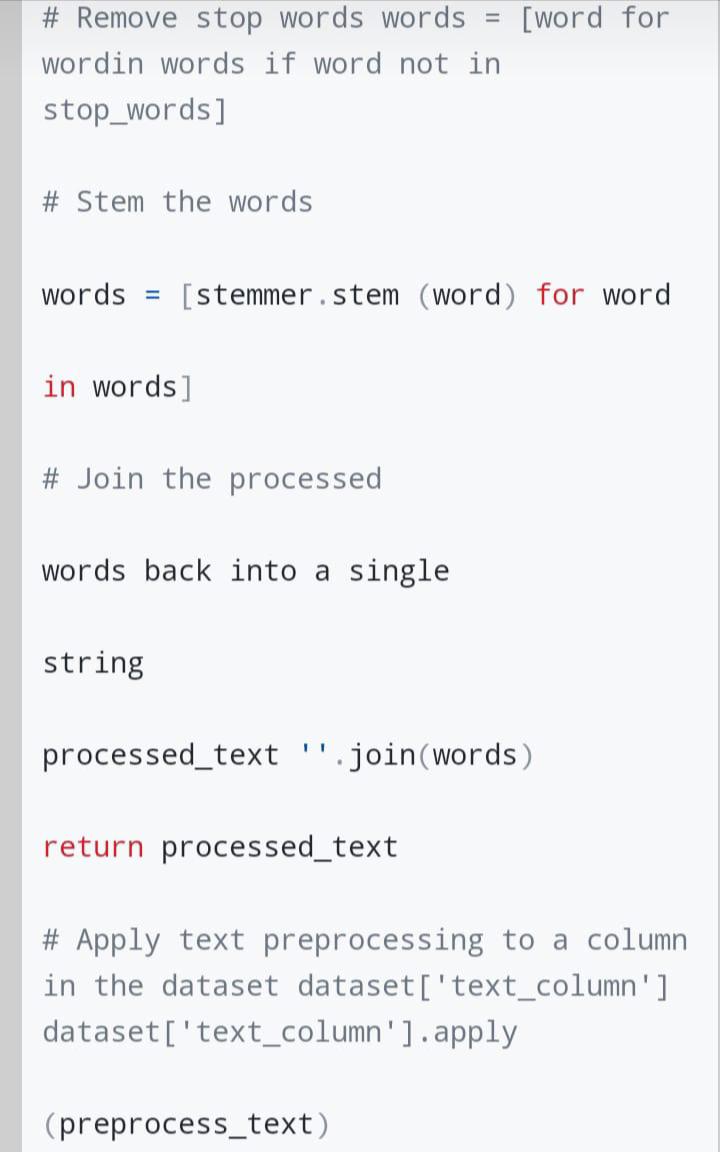
**Text Preprocessing:**

Data preprocessing for diabetes prediction system includes cleaning and transforming the raw dataset, handling missing values and outliers, scaling and normalizing the features, and splitting the data into training and testing sets.

**Using Natural Language ToolKit Library for Text Preprocessing**

Next, you need to perform text preprocessing on the dataset to clean and transform the textual data. This involves tasks such as removing punctuation, converting text to lowercase, removing stop words, and stemming/lemmatizing the words





**Selecting a machine learning algorithm:**

There are several algorithms you can consider for this task, such as Logistic Regression, Decision Trees, Random Forest, Support Vector Machines (SVM), or Neural Networks.

To determine the best algorithm, you need to consider factors like the size and quality of your dataset, the interpretability of the model, the computational resources available, and the specific requirements of your project.

We use the random forest algorithm to implement AI BASED DIABETES PREDICTION SYSTEM

**Random Forest:**

Building the model using RandomForest

From sklearn.ensemble import RandomForestClassifier

Rfc = RandomForestClassifier(n\_estimators=200)

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

Now after building the model let’s check the accuracy of the model on the training dataset.

Rfc\_train = rfc.predict(X\_train)

From sklearn import metrics

Print(“Accuracy\_Score =”, format(metrics.accuracy\_score(y\_train, rfc\_train)))

Output:

Accuracy = 1.0

So here we can see that on the training dataset our model is overfitted.

Getting the accuracy score for Random Forest

From sklearn import metrics

Predictions = rfc.predict(X\_test)

Print(“Accuracy\_Score =”, format(metrics.accuracy\_score(y\_test, predictions)))

Output:

Accuracy\_Score = 0.7677165354330708

**Training the model:**

Split your dataset: Divide your dataset into two parts: a training set and a testing set. The training set will be used to train the model, while the testing set will be used to evaluate its performance.

Preprocess the data: Clean and preprocess your data by handling missing values, normalizing or standardizing numerical features, and encoding categorical variables if necessary.

Train the model: Use the training set to train your model using the selected machine learning algorithm. Fit the algorithm to the training data, allowing it to learn the patterns and relationships in the data.

**Splitting the dataset**

X = diabetes\_df.drop(‘Outcome’, axis=1)

Y = diabetes\_df[‘Outcome’]

Now we will split the data into training and testing data using the train\_test\_split function

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

X\_train, X\_test, y\_train, y\_test

**Evaluating it’s performance:**

**Evaluate the model:**

Once the model is trained, use the testing set to evaluate its performance. Predict the target variable for the testing data and compare the predictions with the actual values. You can use evaluation metrics such as accuracy, precision, recall, F1-score, or area under the ROC curve (AUC-ROC) to assess how well the model is performing.

**Fine-tune the model:**

If the model’s performance is not satisfactory, you may need to fine-tune it by adjusting hyperparameters or trying different variations of the algorithm. This can be done through techniques like grid search or random search.

**Validate the model:**

After fine-tuning, it is essential to validate the model’s performance on an independent dataset, known as the validation set. This helps ensure that the model hasn’t overfit to the training data.

**Deploy the model:**

Once you are satisfied with the model’s performance, you can deploy it into your AI-based diabetes prediction system to make predictions on new, unseen data.

Evaluation of a diabetes prediction system based on artificial intelligence (AI) can be done through various approach :

**Accuracy**: The most straightforward evaluation measure is to assess the accuracy of the system in predicting the occurrence or presence of diabetes. The prediction results can be compared against the ground truth data to calculate the accuracy of the system.

**Sensitivity and Specificity:** Diabetes prediction systems need to be able to correctly identify individuals with diabetes (high sensitivity) while also accurately classifying those without diabetes (high specificity). Sensitivity measures the proportion of true positives identified by the system, while specificity assesses the proportion of true negatives identified.

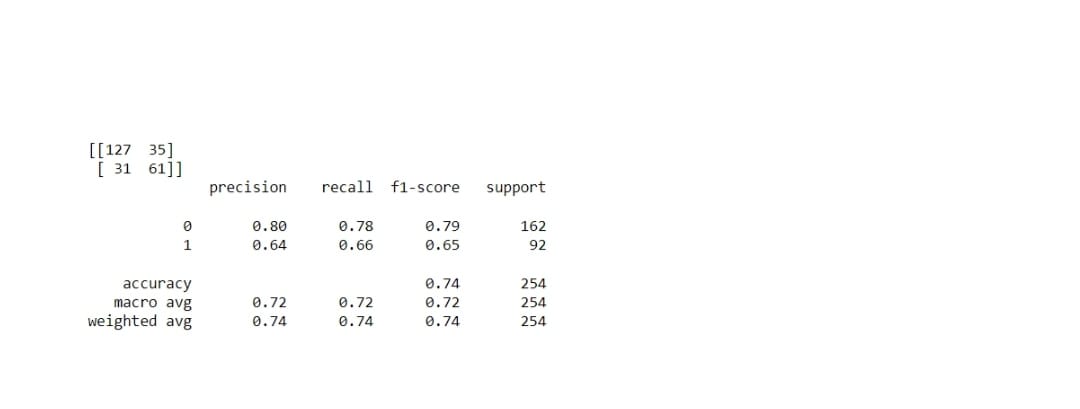
**Receiver Operating Characteristic (ROC) curve analysis:** This evaluation approach helps assess the trade-off between sensitivity and specificity and determines the system’s ability to discriminate between individuals with and without diabetes. The ROC curve plots sensitivity against 1-specificity, and the area under the curve (AUC) can be calculated to quantify the system’s performance.

**Precision and Recall**: Precision measures the proportion of true positive predictions out of the total predicted positives, whereas recall measures the proportion of true positives identified out of all actual positives. Precision and recall are helpful for evaluating the system’s ability to predict diabetes accurately without missing important cases.

**Cross-validation:** To ensure the system’s robustness and generalizability, cross-validation can be performed. This involves dividing the dataset into multiple subsets and training/evaluating the system on different combinations of these subsets. The average performance across all iterations can then be calculated.

**External Testing:** The system’s performance can be evaluated on an entirely new dataset obtained from a different population or healthcare setting. This approach helps validate the system’s performance beyond the original training dataset and indicates its potential real-world applicability.

**Clinical Validation**: Lastly, the diabetes prediction system should undergo clinical validation, where healthcare professionals assess the system’s predictions in a real-world clinical setting. This can involve comparing the system’s predictions to the gold standard diagnostic tests or obtaining feedback from healthcare providers on the system’s usefulness in clinical decision-making.

Using streamlit Library:

To create an AI-based diabetes prediction system using the Streamlit library, you can design a web interface that allows users to input their diabetes-related features and receive predictions. Here’s an example code that demonstrates how you can accomplish this using Streamlit:

```python

Import streamlit as st

Import pandas as pd

From sklearn.ensemble import RandomForestClassifier

From sklearn.metrics import accuracy\_score

# Load the pre-trained model

Model = RandomForestClassifier()

Model.load\_model(‘diabetes\_model.pkl’)

# Function to preprocess input data

Def preprocess\_data(input\_data):

# Preprocess the input data (e.g., scaling, encoding)

# …

# Return the preprocessed data

Return input\_data

# Function to make predictions

Def make\_prediction(input\_data):

# Preprocess the input data

Preprocessed\_data = preprocess\_data(input\_data)

# Generate predictions

Predictions = model.predict(preprocessed\_data)

# Return the predictions

Return predictions

# Create the web interface

Def main():

# Set the title and description of the web app

St.title(‘Diabetes Prediction System’)

St.write(‘Enter the diabetes-related features below to get predictions.’)

# Create the input form

Form = st.form(key=’diabetes\_prediction\_form’)

# Add input fields

Age = form.number\_input(‘Age’, min\_value=0, max\_value=120, step=1)

Blood\_pressure = form.number\_input(‘Blood Pressure’, min\_value=0, max\_value=200, step=1)

Insulin\_level = form.number\_input(‘Insulin Level’, min\_value=0, max\_value=1000, step=1)

# Add more input fields for other features

# Add a submit button

Submit\_button = form.form\_submit\_button(label=’Predict’)

# Perform prediction when the submit button is clicked

If submit\_button:

# Store the input data into a pandas DataFrame

Input\_data = pd.DataFrame([{

‘Age’: age,

‘BloodPressure’: blood\_pressure,

‘Insulin’: insulin\_level,

# Add more features based on your dataset

}])

# Make predictions

Predictions = make\_prediction(input\_data)

# Display the predictions

If predictions[0] == 0:

St.write(‘The predicted outcome is: No diabetes’)

Else:

St.write(‘The predicted outcome is: Diabetes’)

# Run the web interface

If \_\_name\_\_ == ‘\_\_main\_\_’:

Main()

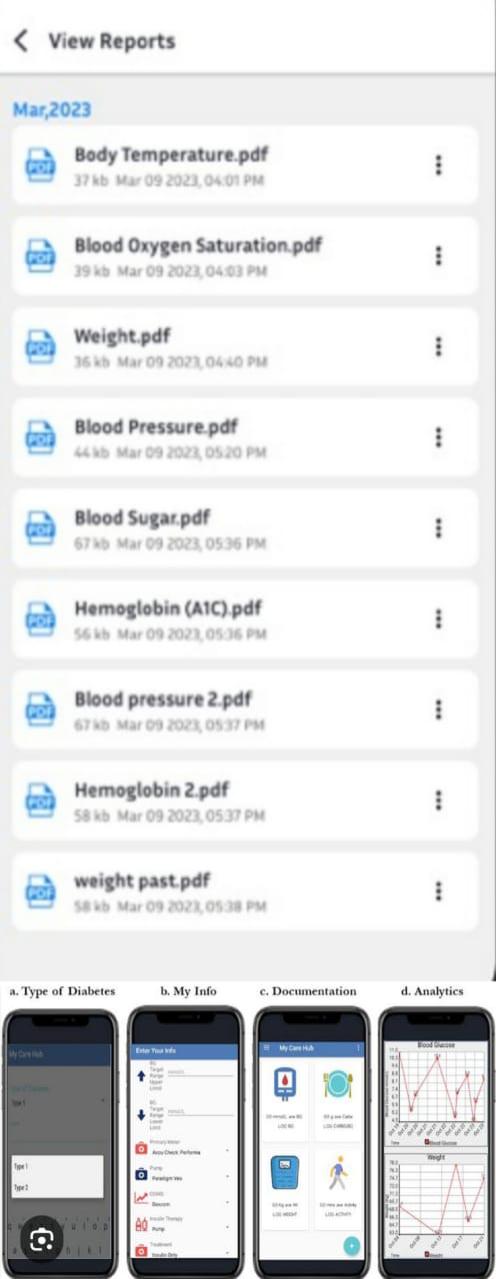
```

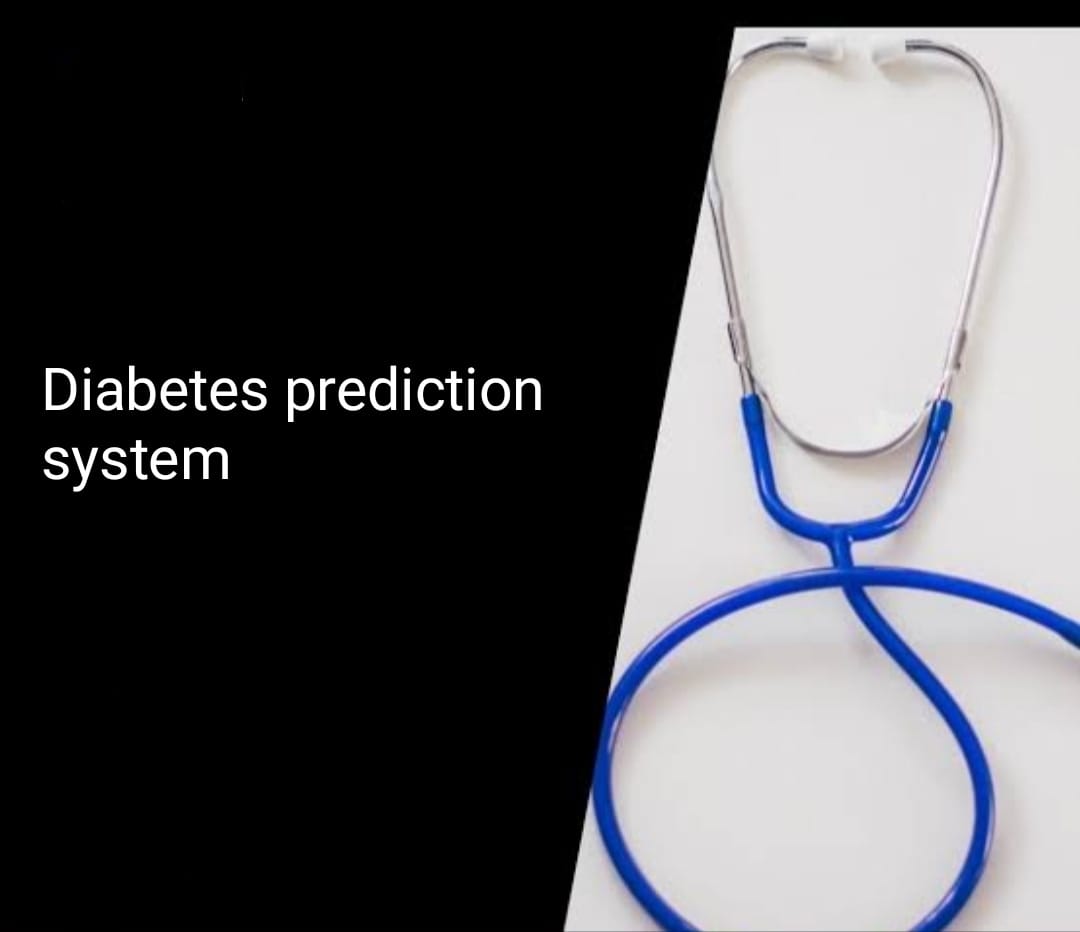
In this example code, we assume that you have a pre-trained Random Forest Classifier model stored as ‘diabetes\_model.pkl’. You can load the model using the appropriate function, such as `model.load\_model()`.

The code defines three functions: `preprocess\_data()` to preprocess the input data, `make\_prediction()` to generate predictions using the preprocessed data, and `main()` to create the web interface using Streamlit.

The web Interface has a title, description, and input form created using `st.form()`. It includes input fields for diabetes-related features like age, blood pressure, and insulin level. You can add more input fields based on the features in your dataset.

When the user clicks the ‘Predict’ button, it triggers the prediction process. The input data is stored in a pandas DataFrame, and then the `make\_prediction()` function generates predictions for the input data using the pre-trained model. Finally, the predicted outcome is displayed on the web interface using `st.write()`.





Conclusion:

Al in Diabetes helps to predict or Detect Diabetes. Any neglect in health can have a high cost for the patients and the medical practitioner. It becomes challenging for the patient to trust that this decision is taken by the machine that does not explain how it reaches a particular conclusion.