AI Based Diabetes Prediction System

PHASE 3-Submission document

Introduction:

AI in Healthcare is an industry that always makes it necessary to make a precise decision, whether it is a treatment, test, or discharge. Diabetes is common due to modern food intake, and it is necessary to keep track of the body. AI 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.

So the use of the Explainable AI is mandatory in predicting disease that will help gain the confidence of an AI system result. Explainable AI helps to get the fair and correct output without errors.

- **Data:** It explains the data used for the prediction, their correlation, and EDA (Exploratory Data Analysis) to understand the hidden data patterns. It tells how the data is to be used for the AI system.

- **Algorithm:** A complete transparency of the system's algorithm is given with the reason why the system chooses it and how it can be beneficial for the prediction.

- **Output:** Akira AI gives a complete justification for the system's output with the reason. It also provides the factors that contribute to influence the result of the system.

Creating a diabetes prediction system involves several steps, including loading and preprocessing the dataset, feature selection, and model development. Here's a general outline of how you can start building your project.

Given set of data:

Pregnancies	Glucose	Blood Pressure	Skin Thickness	Insulin	BMI	Diabetes PedigreeFunction	Age	Outcome
6	148	72	35	0	33.6	0.627	50	1
1	85	66	29	0	26.6	0.351	31	0
8	183	64	0	0	23.3	0.672	32	1
1	89	66	23	94	28.1	0.167	21	0
0	137	40	35	168	43.1	2.288	33	1
5	116	74	0	0	25.6	0.201	30	0
3	78	50	32	88	31	0.248	26	1
10	115	0	0	0	35.3	0.134	29	0
2	197	70	45	543	30.5	0.158	53	1
…	….	…..	…..	….	….	…..	…….	…

……

7	137	90	41	0	32	0.391	39	0
0	123	72	0	0	36.3	0.258	52	1
1	106	76	0	0	37.5	0.197	26	0
6	190	92	0	0	35.5	0.278	66	1
2	88	58	26	16	28.4	0.766	22	0
9	170	74	31	0	44	0.403	43	1
9	89	62	0	0	22.5	0.142	33	0
10	101	76	48	180	32.9	0.171	63	0
2	122	70	27	0	36.8	0.34	27	0
5	121	72	23	112	26.2	0.245	30	0
1	126	60	0	0	30.1	0.349	47	1
1	93	70	31	0	30.4	0.315	23	0

1. Data Collection:

The first step is to gather a dataset containing relevant information for diabetes prediction. Commonly used datasets for this task include the Diabetes dataset from scikit-learn or the Pima Indians Diabetes Database.

2. Loading the Dataset:

Once you have your dataset, you need to load it into your programming environment. If you're using Python, you can use libraries like Pandas to read data from various file formats (e.g., CSV, Excel).

Program:

```
import pandas as pd


# Load the dataset

data = pd.read_csv('diabetes_dataset.csv')
```

3. Data Exploration:

Explore the dataset to understand its structure and characteristics. This may include checking for missing values, summary statistics, and visualizing the data.

Program:

```
# Check for missing values
missing_values = data.isnull().sum()


# Summary statistics
summary_stats = data.describe()


# Data visualization (e.g., histograms, box plots)
import matplotlib.pyplot as plt
data.hist()
plt.show()
```

4. Preprocessing:

Data preprocessing is crucial for preparing the dataset for machine learning. Common preprocessing steps include:

- Handling missing values (imputation).
- Scaling/normalizing features.
- Encoding categorical variables (if any).
- Splitting the data into training and testing sets.

Program:

```
from sklearn.model_selection import train_test_split
```

```
from sklearn.preprocessing import StandardScaler

# Handle missing values (if any)
data.fillna(data.mean(), inplace=True)

# Split the data into features (X) and target (y)
X = data.drop('diabetes_target_column', axis=1)
y = data['diabetes_target_column']

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

# Standardize features
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

5. Feature Selection:

Selecting relevant features is crucial for building an effective predictive model. You can use feature selection techniques like Recursive Feature Elimination (RFE) or feature importance scores from tree-based models to identify important features.

6. Model Development:

Choose a machine learning model suitable for the task. Common models for binary classification (diabetes prediction) include Logistic Regression, Random Forest, Support Vector Machine (SVM), and Neural Networks. Train and evaluate the model on your training data.

Program:

```
from sklearn.linear_model import LogisticRegression


# Create and train the model

model = LogisticRegression()

model.fit(X_train, y_train)


# Evaluate the model

accuracy = model.score(X_test, y_test)
```

7. Model Evaluation and Optimization:

Assess the model's performance using appropriate evaluation metrics (e.g., accuracy, precision, recall, F1-score, ROC AUC). If necessary, fine-tune hyperparameters and explore different models to improve performance.

8. Deployment (Optional):

If you plan to deploy the diabetes prediction system, you'll need to create a user-friendly interface for input and output, possibly using a web application framework or other deployment methods.

This is just a high-level overview of the initial steps to build a diabetes prediction system. The specific steps and techniques may vary depending on your dataset and goals. The process can be iterative, involving further feature engineering, model tuning, and testing.

Loading the diabetes prediction system:

To load a diabetes prediction system, you'll need a trained machine learning model and any required preprocessing steps or data transformations that were part of the original system. Below are step-by-step instructions for loading and using a diabetes prediction system.

1. Import Necessary Libraries:

Program:

```
import joblib
# For loading the pre-trained model
```

2. Load the Pre-Trained Model:

Assuming you have a saved model file (e.g., a joblib file), load the pre-trained model.

3. Prepare New Data:

If you want to make predictions for new data, you'll need to prepare the input data in the same way it was prepared during the training phase. This typically includes handling missing values, scaling or normalizing features, and encoding categorical variables.

If your original data preprocessing steps were more complex, make sure to follow those steps for the new data.

4. Preprocess the New Data (if necessary):

If your pre-trained model expects the input data to be preprocessed, perform the same preprocessing steps on the new data:

5. Make Predictions

Use the loaded model to make predictions on the new data:The `predictions` variable will contain the predicted class (0 or 1 for binary classification) for each data point in the new data.

6. Interpret the Predictions:

Depending on your use case, you may want to interpret the predictions, for example, by providing a probability score or a more human-readable result to the user.

Remember to replace 'diabetes_prediction_model.joblib' with the actual file path of your pre-trained model, and adapt the data preprocessing steps as necessary to match the preprocessing applied during the training phase.

Program:

```
import joblib  # For loading the pre-trained model
import pandas as pd  # For data preprocessing if necessary
from sklearn.preprocessing import StandardScaler
 # For scaling features if needed
model = joblib.load('diabetes_prediction_model.joblib')
# Replace with the actual file path of your model
# Example: Load new data into a DataFrame
new_data = pd.DataFrame({'feature1': [value1], 'feature2': [value2], ...})
# Example: If you need to scale features using the same scaler used during training
scaler = StandardScaler()
  # Use the same scaler you used during training
new_data_scaled = scaler.transform(new_data)
predictions = model.predict(new_data_scaled)
  # Replace 'new_data_scaled' with the preprocessed new data if needed
```

Preprocessing the diabetes prediction system:

Data preprocessing is a crucial step in building a diabetes prediction system. This step ensures that the data is cleaned, transformed, and ready for use in training a machine learning model. Below are step-by-step instructions for preprocessing the data for a diabetes prediction system.

1. Import Necessary Libraries:

Program:

```
import pandas as pd
 # For data manipulation
from sklearn.model_selection import train_test_split  # For splitting the data
from sklearn.preprocessing import StandardScaler  # For feature scaling
```

2. Load the Dataset:

Load your diabetes dataset into a pandas DataFrame

3. Explore the Dataset:

Explore the dataset to understand its structure and identify any issues.

4. Handle Missing Values:

If there are missing values in your dataset, you should decide how to handle them. One common approach is to impute missing values with the mean or median of the respective feature.

5. Split the Data:

Split the data into features (X) and the target variable (y). Also, split it into training and testing sets.

6. Standardize Features:

Standardize or normalize the features to ensure they have a mean of 0 and a standard deviation of 1. This is important for some machine learning algorithms.

7. Additional Preprocessing (if necessary):

Depending on your dataset and the specific requirements of your model, you may need to perform additional preprocessing steps. This could include encoding categorical variables, feature engineering, or addressing outliers.

8. Data Ready for Model Training:

At this point, your data is preprocessed and ready for training a diabetes prediction model using machine learning algorithms. You can proceed with model development as described in the previous response.

These preprocessing steps are just a general guideline. Depending on your specific dataset and the requirements of your project, you may need to adapt and expand these steps accordingly.

Program:

```
import pandas as pd
# For data manipulation
from sklearn.model_selection import train_test_split
 # For splitting the data
from sklearn.preprocessing import StandardScaler
 # For feature scaling
# Replace 'diabetes_dataset.csv' with the actual path to your dataset
data = pd.read_csv('diabetes_dataset.csv')
# Check for missing values
missing_values = data.isnull().sum()

# Summary statistics
summary_stats = data.describe()

# Data visualization (e.g., histograms, box plots)
import matplotlib.pyplot as plt
data.hist()
plt.show()
data.fillna(data.mean(), inplace=True)
X = data.drop('diabetes_target_column', axis=1)  # Replace with the actual target column name
y = data['diabetes_target_column']
```

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

scaler = StandardScaler()

X_train = scaler.fit_transform(X_train)

X_test = scaler.transform(X_test)
```

Conclusion:

A diabetes prediction system can provide valuable insights and aid in early diagnosis and prevention of diabetes, contributing to better healthcare outcomes. It's important to continually refine and update the system as more data becomes available and as machine learning techniques advance.