AI-Based Diabetes Prediction System

Phase 4

In this phase, we'll continue building our project AI-based diabetes prediction system by:

Selecting a machine learning algorithm
Training the model
Evaluating its performance

Selecting a machine learning algorithm:

In general, for classification problems, several machine learning algorithms have shown promising results. Few commonly used algorithms were:
Logistic Regression, Random Forest, Support Vector Machines (SVM), Gradient Boosting and Neural Networks

Here, we have chosen the **Random Forest algorithm**.

Random Forest is an ensemble method that combines multiple decision trees to make accurate predictions. It can handle complex and high-dimensional datasets, making it suitable for diabetes prediction with multiple input features.

[Preprocessing steps, such as handling missing values and feature scaling, should be performed before training the Random Forest model.]

Training the model:

Training the model is a crucial step in machine learning where the model learns patterns and relationships from labeled data to make predictions or classifications.

The training process involves feeding the model with input features and their corresponding target labels, which indicate the desired output or prediction.

```
import pandas as pd
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

# Load the dataset into a DataFrame
data = pd.read_csv('diabetes.csv')

# Split the dataset into features and labels
X = diabetes_dataframe.drop('Outcome', axis=1)
y = diabetes_dataframe['Outcome']
```

```
[43] # 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)

     # Initialize the Random Forest classifier
     model = RandomForestClassifier()

     # Train the model
     model.fit(X_train, y_train)
```

```
▾ RandomForestClassifier
RandomForestClassifier()
```

```
[44] # Make predictions on the testing data
     y_pred = model.predict(X_test)

     # Calculate the accuracy of the model
     accuracy = accuracy_score(y_test, y_pred)
     print("Accuracy:", accuracy)
```

```
Accuracy: 0.7532467532467533
```

Evaluating the performance:

Evaluating the performance of a machine learning model is essential to assess its accuracy, reliability, and effectiveness in making predictions or classifications.

Performance evaluation involves measuring the model's performance metrics using test data that was not used during training.

Common performance metrics for classification tasks include accuracy, precision, recall, F1 score

```python
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score

# Make predictions on the testing data
y_pred = model.predict(X_test)

# Calculate the accuracy of the model
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)

# Calculate the precision of the model
precision = precision_score(y_test, y_pred)
print("Precision:", precision)

# Calculate the recall of the model
recall = recall_score(y_test, y_pred)
print("Recall:", recall)

# Calculate the F1 score of the model
f1 = f1_score(y_test, y_pred)
print("F1 score:", f1)
```

```
Accuracy: 0.7532467532467533
Precision: 0.6440677966101694
Recall: 0.6909090909090909
F1 score: 0.6666666666666665
```

TEAM MEMBERS

JEFFY J
JENIFER P
PRIYA MAHALAKSHMI P
MALLIKA S
J P College of Engineering