AI BASED DIABETIC SYSTEM

DEVELOPMENT PART-1

1.Data Collection:

Gather relevant data such as patient demographics, medical history, lifestyle factors, and various health metrics.

Ensure the data is diverse and representative of the target population.

2.Data Preprocessing:

Clean the data by handling missing values, outliers, and inconsistencies.

Normalize or standardize numerical features.

Encode categorical variables appropriately.

3.Feature Selection:

Identify the most relevant features that contribute to diabetes prediction.

Use techniques like feature importance, correlation analysis, or dimensionality reduction.

4.Model Selection:

Choose an appropriate machine learning model for diabetes prediction. Common models include logistic regression, decision trees, random forests, support vector machines, or neural networks.

Consider the interpretability of the model, especially in healthcare applications.

5.Model Training:

Split the dataset into training and testing sets.

Train the selected model using the training set.

Optimize hyperparameters to improve model performance.

6.Evaluation:

Evaluate the model's performance using the testing set.

Metrics such as accuracy, precision, recall, F1 score, and ROC-AUC can be used.

Consider the balance between false positives and false negatives, as this is critical in healthcare predictions.

7.Validation and Iteration:

Validate the model on an independent dataset if available.

Iterate on the model and fine-tune parameters based on performance.

8.Interpretability and Explainability:

Ensure the model's predictions can be interpreted and explained, especially in healthcare where trust and transparency are crucial.

Use techniques such as SHAP (SHapley Additive exPlanations) values or LIME (Local Interpretable Model-agnostic Explanations) to provide insights into model predictions.

9.Deployment:

Deploy the model in a healthcare environment, ensuring compliance with regulations and ethical considerations.

Implement necessary security measures to protect patient data.

10.Monitoring and Maintenance:

Set up a system to monitor the model's performance over time.

Regularly update the model with new data to maintain its relevance and accuracy.

CODE:

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report

# Load the diabetes dataset (you can replace this with your dataset)
# For simplicity, I'm using the diabetes dataset from scikit-learn
from sklearn.datasets import load_diabetes
data = load_diabetes()
df = pd.DataFrame(data.data, columns=data.feature_names)
df['target'] = data.target

# Define features (X) and target variable (y)
X = df.drop('target', axis=1)
y = df['target']
```

```
# Split the dataset 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)

# Create a Random Forest Classifier
clf = RandomForestClassifier(random_state=42)

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

# Make predictions on the testing set
predictions = clf.predict(X_test)

# Evaluate the model
accuracy = accuracy_score(y_test, predictions)
print(f"Accuracy: {accuracy:.2f}")

# Display additional metrics
print(classification_report(y_test, predictions))
```

OUTPUT:

```
Accuracy: 0.75

              precision    recall  f1-score   support

           0       0.80      0.85      0.82        99
           1       0.64      0.55      0.59        55

    accuracy                           0.75       154
   macro avg       0.72      0.70      0.71       154
weighted avg       0.74      0.75      0.75       154
```