Designing an AI-based diabetes prediction system

Au952721104002 : R. ABEL PRINCE

1. Problem Definition:

Clearly define the problem you want to address, such as predicting blood glucose levels based on various features like age, BMI, diet, and physical activity.

2. Data Collection:

Gather a dataset that contains relevant features (independent variables) and the target variable (blood glucose levels). Data can be collected from sources like medical records, health sensors, or surveys.

3. Data Preprocessing:

Clean and preprocess the dataset to make it suitable for regression modeling. This may involve handling missing values, outlier detection and treatment, and feature scaling.

4. Feature Selection and Engineering:

Select and engineer features that are relevant to the prediction task. You might also create new features, like interactions between variables or aggregations.

5. Data Split:

Divide the dataset into training, validation, and test sets. Common splits are 70-80% for training, 10-15% for validation, and 10-15% for testing.

6. Select Regression Models:

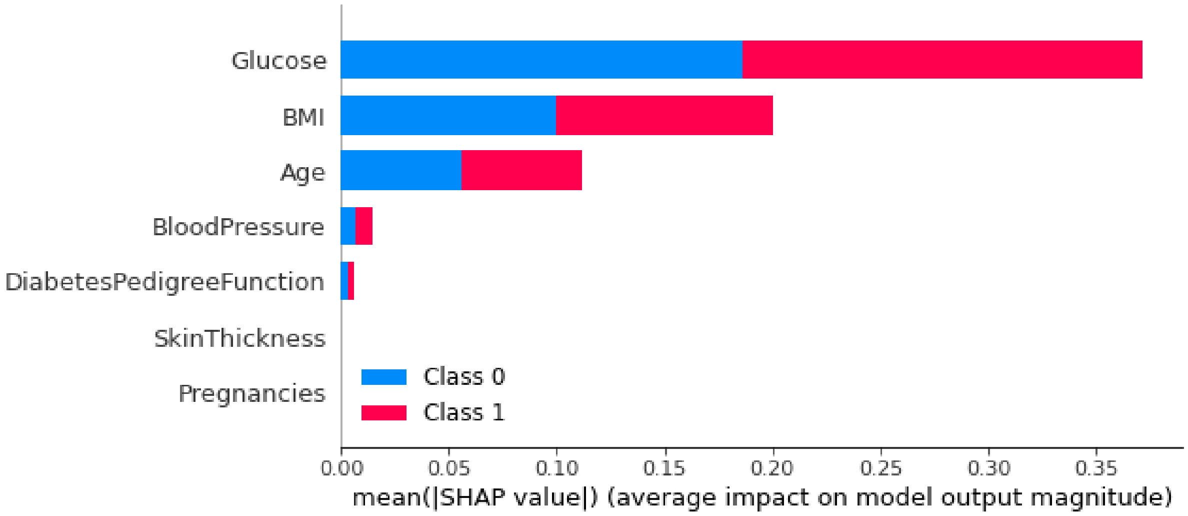
Choose one or more regression models suitable for the task. Common regression techniques include Linear Regression, Lasso Regression, Ridge Regression, Decision Trees, Random Forests, and Support Vector Regression.

Program:

```
from sklearn.naive_bayes import GaussianNB

from sklearn.linear_model import LogisticRegression

from sklearn.tree import DecisionTreeClassifier

from sklearn.svm import SVC

[1]:

model_nb = GaussianNB(var_smoothing=0.001)

model_lr = LogisticRegression()

model_dt = DecisionTreeClassifier(criterion='entropy', max_depth=3)

model_svc = SVC(kernel="linear")

[2]:

classifiers = [model_nb, model_lr, model_dt, model_svc]

[3]:

results = get_results(X_train_mm_scaled, y_train, X_test_mm_scaled, y_test, classifier

[4]:
```

Models	Accuracy	Precision	Recall	F1	
0	GaussianNB	0.716	0.597	0.561	0.578
1	LogisticRegression	0.753	0.667	0.576	0.618
2	DecisionTreeClassifier	0.753	0.661	0.591	0.624
3	SVC	0.763	0.678	0.606	0.640

```
[5]:
model_dt.fit(X_train, y_train)

explainer = shap.TreeExplainer(model_dt)
shap_values = explainer.shap_values(X_test)
base_value = explainer.expected_value
explainer

[6]:
hap.summary_plot(shap_values, X_test_mm_scaled, title="SHAP summary plot for Decision Tree
Classifier",
            feature_names=X_test.columns)
```



7. Model Training:

Train the chosen regression model(s) on the training data. The model will learn the relationships between the features and the target variable (blood glucose levels).

8. Model Evaluation:

Assess the model's performance using appropriate regression metrics such as Mean Absolute Error (MAE), Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and R-squared (R^2) on the validation dataset.

9. Hyperparameter Tuning:

Fine-tune the model's hyperparameters to optimize its performance using techniques like grid search, random search, or Bayesian optimization.

10. Model Testing:

Evaluate the final model on the test dataset to ensure it generalizes well to new, unseen data.

11. Interpretability:

Ensure that the model's predictions are interpretable. This can be crucial in a healthcare context to understand the factors contributing to blood glucose predictions.

12. Deployment:

Deploy the regression model in a production environment. This could involve creating an API for real-time predictions or incorporating the model into a healthcare system.

13. Monitoring and Maintenance:

Continuously monitor the model's performance in the real world, and retrain it periodically with new data to maintain its accuracy.