Machine Learning for Early Detection of Diabetes: A Predictive and Preventative Approach

DENG Siwei , WEI Xinquan , CHENG Yingtong ,
LI Linzu , DONG Xinyu , GUAN Yuqi

March 24, 2025

1 Introduction

Diabetes is a chronic metabolic disease with a rising global prevalence, causing significant health and economic burdens to individuals and society. Early detection and intervention are crucial for preventing or delaying complications associated with diabetes, such as heart disease, stroke, kidney failure, and blindness. However, traditional diabetes screening methods often rely on invasive testing or diagnoses after the onset of symptoms, which may delay treatment. In recent years, machine learning technologies have shown great potential in healthcare, particularly in disease prediction and risk stratification. Machine learning algorithms can learn complex patterns and relationships from vast amounts of data, identifying subtle risk factors that traditional methods may overlook.

This project aims to develop an efficient diabetes prediction model using machine learning based on publicly available datasets, and to apply it in practical healthcare settings through a user-friendly interface, thereby promoting early prevention and management of diabetes.

2 Project Objectives

This project aims to achieve the following objectives through machine learning techniques:

- **Identify Key Risk Factors for Early Diabetes.** By conducting exploratory data analysis and feature selection, we aim to identify variables closely associated with diabetes risk, providing guidance for subsequent model construction.

- **Build a Reliable Prediction Model.** Utilizing machine learning algorithms, we will develop a model capable of predicting an individual's risk of developing diabetes based on publicly available datasets.
- **Evaluate Model Performance.** We will assess the model's prediction effectiveness based on metrics such as accuracy, sensitivity, specificity, F1 score, and AUC-ROC, and explore its applicability in different scenarios.
- **Develop a User-Friendly Interface.** We aim to design an intuitive and easy-to-use interface that allows healthcare professionals to quickly utilize the model in clinical environments.

3 Dataset Selection

To ensure the model's universality and scientific rigor, we selected the Mendeley Diabetes Dataset from the following three datasets for the following reasons:

- **Pima Indians Diabetes Database.** *Sample Limitations*: This dataset only includes diabetes data from Pima Indian women, lacking gender and racial diversity, which may lead to limitations and biases in research outcomes. *Privacy Issues*: As the data involves specific populations, there may be privacy and ethical considerations that restrict the dataset's usage.
- **UCI Diabetes Dataset.** *Complexity*: This dataset has a complex format, containing time-sensitive details, making it less convenient to process and increasing the difficulty of data analysis. *Low User Engagement*: The data collection requires a high level of expertise, resulting in poorer user input convenience, which may lead to insufficient sample sizes and affect the representativeness and reliability of research.
- **Mendeley Diabetes Dataset.** *Privacy Protection*: This dataset does not involve sensitive information about specific individuals, ensuring the ethical compliance of the research and avoiding potential privacy breaches during data input. *Professionalism and Credibility*: As an academic platform, Mendeley provides datasets that are typically peer-reviewed and possess high credibility and scientific rigor, making them suitable for rigorous research and analysis.

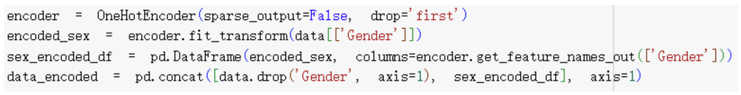
4 Data Preprocessing

Data preprocessing is a crucial step for the success of machine learning models, aimed at improving data quality and the model's predictive capability. In this study, we carried out the following significant preprocessing steps:

4.1 Code Implementation

- **Data Loading and Missing Value Handling.** First, we loaded the diabetes dataset (Dataset of Diabetes.csv) and checked for missing values. For numerical data, we filled missing values with the median; for categorical variables, we used the mode. This method effectively avoids data loss due to missing values.

- **Categorical Variable Encoding.** Categorical variables in the data (e.g., Gender) need to be one-hot encoded to convert them into numerical data for subsequent machine learning processing. We used OneHotEncoder to encode the Gender column and generated a new column, Gender_M.



```
encoder = OneHotEncoder(sparse_output=False, drop='first')
encoded_sex = encoder.fit_transform(data[['Gender']])
sex_encoded_df = pd.DataFrame(encoded_sex, columns=encoder.get_feature_names_out(['Gender']))
data_encoded = pd.concat([data.drop('Gender', axis=1), sex_encoded_df], axis=1)
```

Figure 1: Categorical Variable Encoding

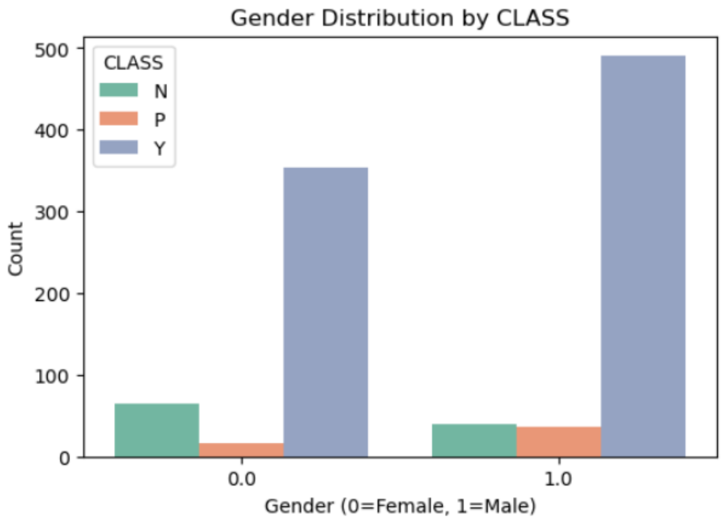
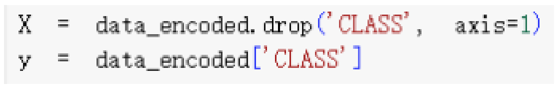


Figure 2: Gender Distribution by CLASS

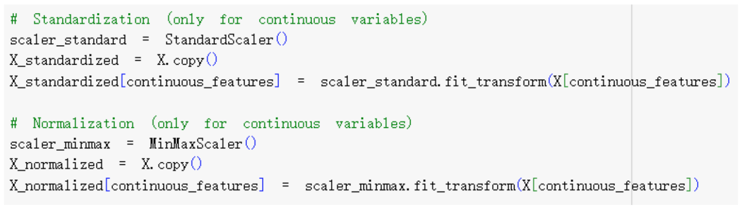
- **Separation of Features and Target Variable.** During preprocessing, we clarified the feature matrix X and the target variable Y. The feature matrix contains all input features, while the target variable is the result we wish to predict, namely the classification of diabetes.



```python
X = data_encoded.drop('CLASS', axis=1)
y = data_encoded['CLASS']
```

Figure 3: X and Y

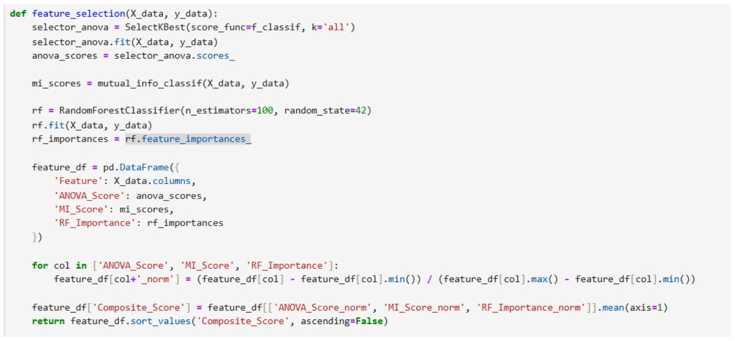
- **Standardization and Normalization of Continuous Variables.** To ensure the effectiveness of model training, we standardized and normalized continuous variables. Standardization transformed the data into a distribution with a mean of 0 and a standard deviation of 1 using StandardScaler, while normalization scaled the data between 0 and 1 using MinMaxScaler. This process helps eliminate the impact of different feature scales on model training.



```python
# Standardization (only for continuous variables)
scaler_standard = StandardScaler()
X_standardized = X.copy()
X_standardized[continuous_features] = scaler_standard.fit_transform(X[continuous_features])

# Normalization (only for continuous variables)
scaler_minmax = MinMaxScaler()
X_normalized = X.copy()
X_normalized[continuous_features] = scaler_minmax.fit_transform(X[continuous_features])
```

Figure 4: Standardization and Normalization

- **Feature Selection.** In the feature selection phase, we defined a function called feature_selection to evaluate feature importance. This function calculates feature scores using ANOVA, mutual information, and random forest algorithms and returns a comprehensive score.



```python
def feature_selection(X_data, y_data):
    selector_anova = SelectKBest(score_func=f_classif, k='all')
    selector_anova.fit(X_data, y_data)
    anova_scores = selector_anova.scores_

    mi_scores = mutual_info_classif(X_data, y_data)

    rf = RandomForestClassifier(n_estimators=100, random_state=42)
    rf.fit(X_data, y_data)
    rf_importances = rf.feature_importances_

    feature_df = pd.DataFrame({
        'Feature': X_data.columns,
        'ANOVA_Score': anova_scores,
        'MI_Score': mi_scores,
        'RF_Importance': rf_importances
    })

    for col in ['ANOVA_Score', 'MI_Score', 'RF_Importance']:
        feature_df[col+'_norm'] = (feature_df[col] - feature_df[col].min()) / (feature_df[col].max() - feature_df[col].min())

    feature_df['Composite_Score'] = feature_df[['ANOVA_Score_norm', 'MI_Score_norm', 'RF_Importance_norm']].mean(axis=1)
    return feature_df.sort_values('Composite_Score', ascending=False)
```

Figure 5: feature selection

4.2 Results Presentation

Ultimately, we identified five features most closely related to the CLASS variable, as illustrated below:

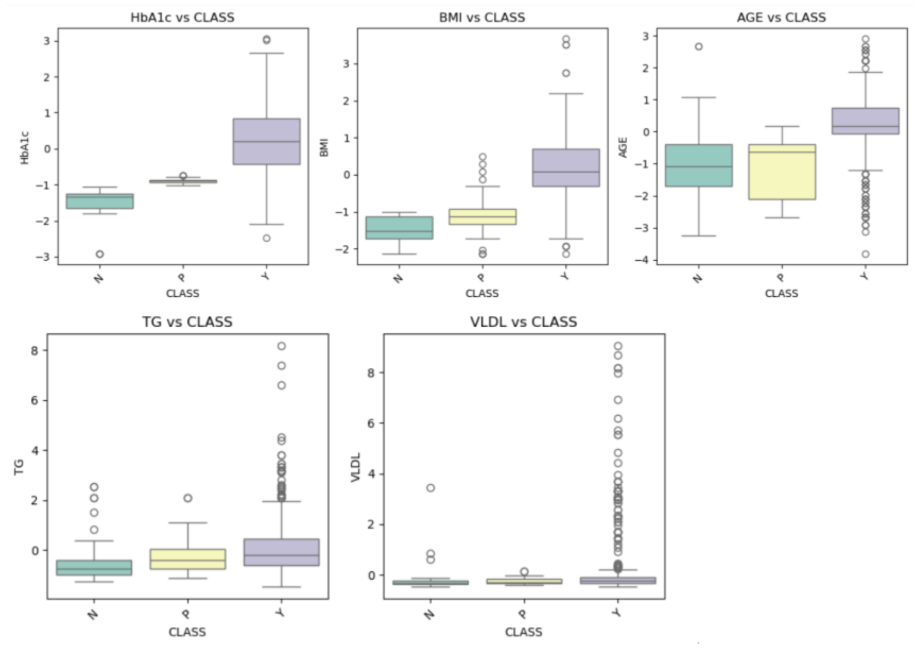


Figure 6: five features

4.3 Issues Encountered and Solutions

During the data preprocessing process, we noticed five categories appearing in the generated classification distribution chart (as shown below), two of which were "Y" and "N," which were clearly incorrect. Upon careful inspection of the data, we found that these two categories were actually caused by spaces in the data records.

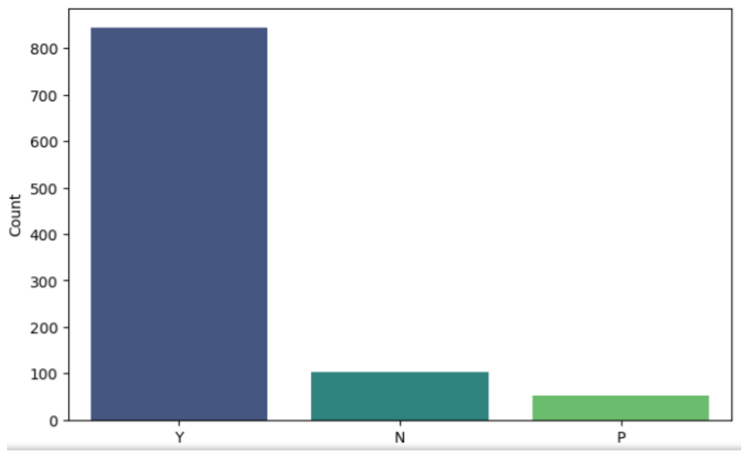


Figure 7: issues

5 Machine Learning Model Development

5.1 Neural Networks (NN)

Neural Networks (NN) are computational models that simulate biological neural systems, consisting of multiple layers of neurons (nodes). Each layer maps input features to a new feature space through linear transformations and nonlinear activation functions. By stacking layers, neural networks can gradually learn complex feature interactions, demonstrating outstanding performance across various tasks.

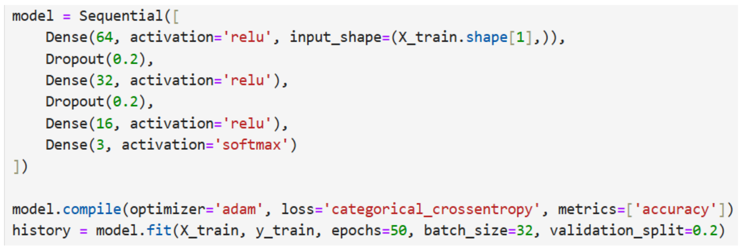
In this section, we constructed a simple fully connected neural network using Keras for handling classification tasks, divided into data preprocessing, model construction, and training:

5.1.1 Data Preprocessing

- **Feature Selection**: This study removed features unrelated to the task (such as ID and No_Pation) to reduce model complexity and improve training efficiency.
- **Label Encoding**: Class labels (CLASS) were mapped to numerical values for model processing, specifically mapped as N:0, Y:1, P:2.
- **Feature Separation**: Features x and labels y were separated to prepare for subsequent processing.

- **One-Hot Encoding of Labels**: The to_categorical function was used to convert labels into one-hot encoding format to meet the needs of multi-class tasks.
- **Dataset Splitting**: Finally, the dataset was split into training and testing sets to ensure the effectiveness of model training and evaluation.

5.1.2 Model Construction



```
model = Sequential([
    Dense(64, activation='relu', input_shape=(X_train.shape[1],)),
    Dropout(0.2),
    Dense(32, activation='relu'),
    Dropout(0.2),
    Dense(16, activation='relu'),
    Dense(3, activation='softmax')
])

model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
history = model.fit(X_train, y_train, epochs=50, batch_size=32, validation_split=0.2)
```

Figure 8: model construction

We constructed a fully connected neural network with the following specific structure:

- **Input Layer**: Input dimension is $X\_train.shape[1]$, i.e., the number of features.
- **Hidden Layers**:
  - First Layer: 64 neurons, activation function is ReLU (Rectified Linear Unit).
  - Second Layer: 32 neurons, activation function is ReLU.
  - Third Layer: 16 neurons, activation function is ReLU.
- **Output Layer**: 3 neurons (corresponding to 3 categories), activation function is Softmax.

5.1.3 Principle Analysis

Neural networks can automatically learn complex interactions between features through multiple layers of nonlinear transformations (such as ReLU activation functions). Each layer of neurons combines features from the previous layer to generate higher-level feature representations. Moreover, the introduction of dropout layers effectively enhances the model's generalization capability, thereby preventing overfitting.

5.1.4 Loss Function and Optimizer

During model training, we employed categorical cross-entropy loss as the loss function and selected the Adam optimizer for parameter optimization. Categorical cross-entropy loss is used to evaluate the difference between the model's predicted probability distribution and the true label distribution, especially suitable for multi-class tasks. True labels are generally represented in one-hot encoding format, while model outputs are generated through the Softmax function to produce the corresponding probability distribution.

5.2 XGBoost

5.2.1 Overview of the XGBoost Model

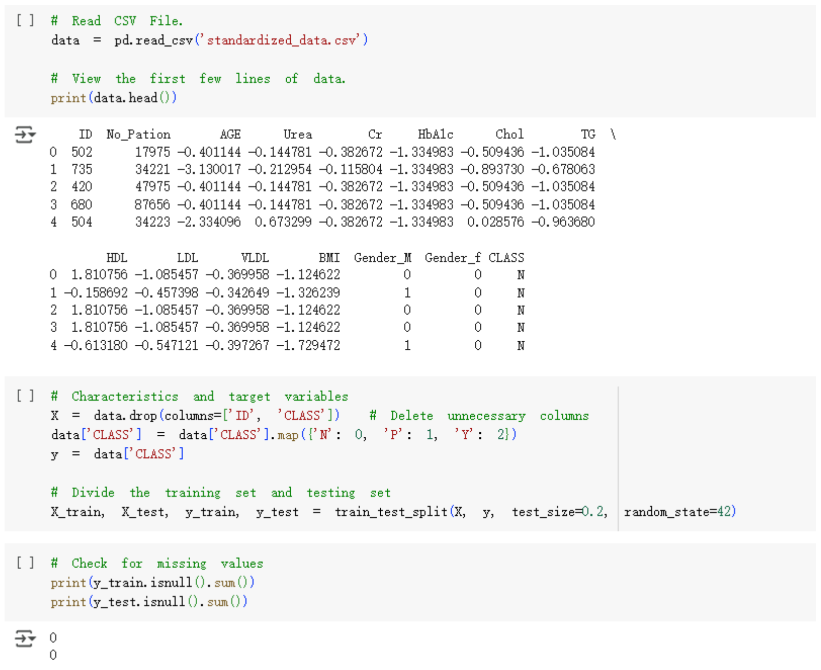
XGBoost (eXtreme Gradient Boosting) is an advanced version of Gradient Boosted Decision Trees (GBDT), which predicts residuals by gradually training subsequent trees, thereby continuously improving the model's prediction performance. XGBoost stands out in the machine learning field with its unique features, particularly excelling in handling complex data and large-scale datasets. It effectively balances model complexity and prediction performance by introducing regularization terms and optimizing objective functions.

5.2.2 Core Principles of XGBoost

The basic principles of XGBoost can be summarized as follows:

- **Tree Boosting**: XGBoost uses tree boosting as its fundamental model, making predictions by progressively adding the results of multiple decision trees. Each tree learns how to correct the predictive errors of the previous trees, thereby gradually constructing a powerful predictive model.
- **Regularized Objective Function**: XGBoost defines an objective function that includes regularization terms aimed at controlling model complexity to prevent overfitting. The regularization term penalizes the number of leaf nodes to regulate model complexity.
- **Gradient Boosting**: XGBoost optimizes the objective function using gradient boosting. At each iteration, it adds a new tree to minimize the objective function. XGBoost employs Newton's method to solve for the extreme values of the loss function, using second-order Taylor expansion to accelerate convergence.

5.2.3 Code Demonstration



```
# Read CSV File.
data = pd.read_csv('standardized_data.csv')

# View the first few lines of data.
print(data.head())
```

```
    ID  No_Pation       AGE      Urea        Cr     HbA1c      Chol        TG  \
0  502      17975 -0.401144 -0.144781 -0.382672 -1.334983 -0.509436 -1.035084
1  735      34221 -3.130017 -0.212954 -0.115804 -1.334983 -0.893730 -0.678063
2  420      47975 -0.401144 -0.144781 -0.382672 -1.334983 -0.509436 -1.035084
3  680      87656 -0.401144 -0.144781 -0.382672 -1.334983 -0.509436 -1.035084
4  504      34223 -2.334096  0.673299 -0.382672 -1.334983  0.028576 -0.963680

        HDL       LDL      VLDL       BMI  Gender_M  Gender_f CLASS
0  1.810756 -1.085457 -0.369958 -1.124622         0         0     N
1 -0.158692 -0.457398 -0.342649 -1.326239         1         0     N
2  1.810756 -1.085457 -0.369958 -1.124622         0         0     N
3  1.810756 -1.085457 -0.369958 -1.124622         0         0     N
4 -0.613180 -0.547121 -0.397267 -1.729472         1         0     N
```

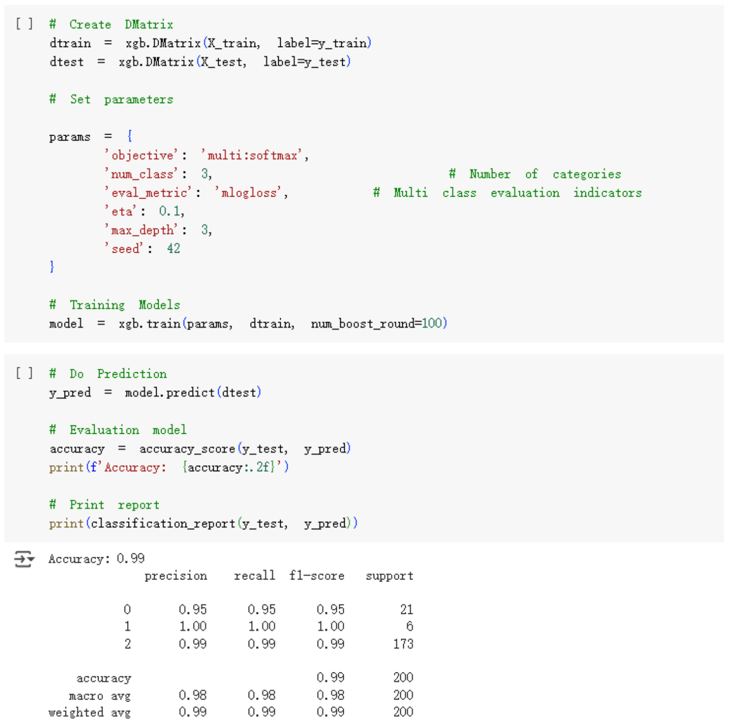
```
# Characteristics and target variables
X = data.drop(columns=['ID', 'CLASS'])   # Delete unnecessary columns
data['CLASS'] = data['CLASS'].map({'N': 0, 'P': 1, 'Y': 2})
y = data['CLASS']

# Divide the training set and testing set
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

```
# Check for missing values
print(y_train.isnull().sum())
print(y_test.isnull().sum())
```

```
0
0
```

Figure 9: Code 1

We first defined the features and target variables. We removed the CLASS column from the data, leaving the remaining columns to form X. Since the values in CLASS ("N", "P", "Y") are of string type, XGBoost cannot directly process them; therefore, we replaced these variables with 0, 1, and 2, respectively. Finally, we split the dataset into training and testing sets at an 80% to 20% ratio. Although we had previously preprocessed the data, we still needed to check for missing values.



```
[ ] # Create DMatrix
    dtrain = xgb.DMatrix(X_train, label=y_train)
    dtest = xgb.DMatrix(X_test, label=y_test)

    # Set parameters

    params = {
            'objective': 'multi:softmax',
            'num_class': 3,                                  # Number of categories
            'eval_metric': 'mlogloss',            # Multi class evaluation indicators
            'eta': 0.1,
            'max_depth': 3,
            'seed': 42
    }

    # Training Models
    model = xgb.train(params, dtrain, num_boost_round=100)
```

```
[ ] # Do Prediction
    y_pred = model.predict(dtest)

    # Evaluation model
    accuracy = accuracy_score(y_test, y_pred)
    print(f'Accuracy: {accuracy:.2f}')

    # Print report
    print(classification_report(y_test, y_pred))
```

```
Accuracy: 0.99
              precision    recall  f1-score   support

           0       0.95      0.95      0.95        21
           1       1.00      1.00      1.00         6
           2       0.99      0.99      0.99       173

    accuracy                           0.99       200
   macro avg       0.98      0.98      0.98       200
weighted avg       0.99      0.99      0.99       200
```

Figure 10: Code 2

Since we need to predict three categories (0, 1, 2), we specified this in the parameter settings. We set the number of training rounds to 100. Based on the evaluation results from the testing set, we found that the prediction performance of XGBoost was excellent, achieving good results in precision, recall, and F1-score, with an accuracy of 0.99. Notably, it achieved 100% accuracy for Type I diabetes. This outcome is partly due to our thorough preprocessing of the data and possibly because the XGBoost model itself is powerful enough to capture complex feature relationships.

Of course, we must be cautious about the potential risk of overfitting. Although XGBoost has already incorporated regularization to avoid overfitting, and the training results have been validated on the testing set, it does not preclude further checks to confirm the model's robustness.

5.3 Ensemble Learning

In classification problems, Average Ensemble is a method to improve classification performance by combining the predictions of multiple classifiers.

5.3.1 Specific Steps and Principles

First, researchers need to select multiple base learners, which can be of the same type (e.g., multiple decision trees) or different types (e.g., decision trees, support vector machines, logistic regression, etc.). During the training phase, the same training dataset or different subsets of data can be used to enhance model diversity.

Each classifier outputs a probability value for each class, indicating the likelihood that the sample belongs to each class. Assume that there are N classifiers and the predicted probability of sample x is $P_n(y_k \mid x)$(where $n = 1, 2, \ldots, N$ is the classifier index and k is the class index), then the average probability for each class can be calculated as follows:

$$P(y_k \mid x) = \frac{1}{N} \sum_{n=1}^{N} P_n(y_k \mid x)$$

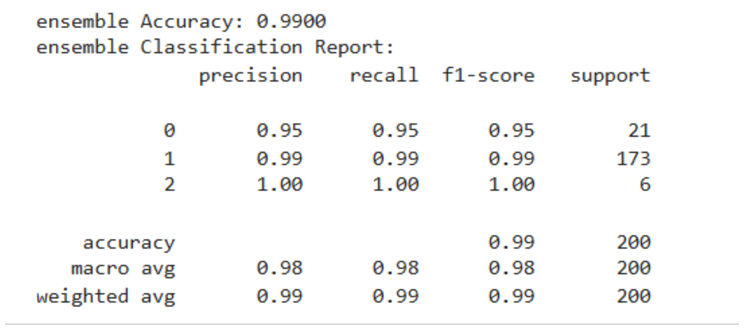
Finally, the model will select the class with the highest probability as the predicted result:

$$\hat{y} = \arg\max_{y_k} P(y_k \mid x)$$

The advantage of the average ensemble method is that it can enhance model stability and robustness; by averaging multiple models, it effectively reduces the bias and variance of individual models, thus improving overall prediction performance. Additionally, it has stronger resistance to outliers and noise. However, when dealing with class imbalance issues, it may be necessary to weight the predicted probabilities to prevent the model from biasing toward the majority class. Therefore, selecting different models to increase diversity is crucial for improving ensemble effectiveness.

5.3.2 Results and Analysis

After data preprocessing, we first used ensemble learning methods (such as random forests) for feature selection, yielding the following results:



```
ensemble Accuracy: 0.9900
ensemble Classification Report:
              precision    recall  f1-score   support

           0       0.95      0.95      0.95        21
           1       0.99      0.99      0.99       173
           2       1.00      1.00      1.00         6

    accuracy                           0.99       200
   macro avg       0.98      0.98      0.98       200
weighted avg       0.99      0.99      0.99       200
```

Figure 11: result

From the results, we can see that the overall accuracy of the model reached 99%. The various metrics in the classification report indicate that the model's performance across different categories is relatively balanced. Specific analyses are as follows:

- **Category 0**: Precision and recall are both high, indicating that the model performs well in identifying this category, but there is still room for improvement.
- **Categories 1 and 2**: Both performed exceptionally well, achieving perfect precision and recall (both 1.00). This indicates that the model can accurately identify these two categories without misclassification.
- **Macro Average and Weighted Average**: The results of both macro and weighted averages indicate stability in the model's overall performance, particularly the weighted average metric (0.99), which further emphasizes the model's comprehensive performance across categories.

In conclusion, the average ensemble method performed excellently in this classification task, demonstrating high accuracy and stability. Future work can further explore addressing class imbalance issues to enhance recognition capabilities for minority classes.

6 Model Evaluation

We conducted a detailed evaluation of three models (Neural Network, XGBoost, and Ensemble Model), analyzing aspects such as accuracy, classification reports, confusion matrices, and ROC curves with AUC values. The following are the evaluation results and comparisons for each model:

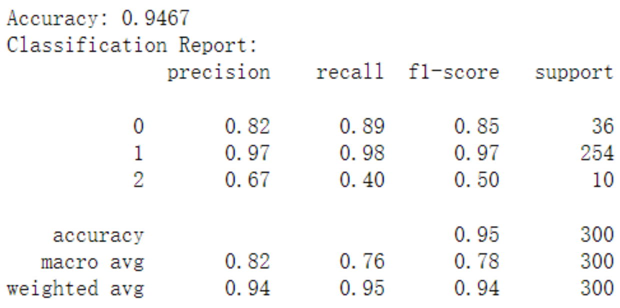
6.1 Neural Network (NN)

6.1.1 Accuracy

The overall accuracy of the neural network model on the testing set was 94.67%, indicating that the model can correctly classify samples in most cases, demonstrating good overall performance. However, compared to the other two models, the accuracy of the neural network was lower.

6.1.2 Classification Report

- **Class 0**: Both precision and recall are relatively high, indicating that the model can identify samples from this class well.
- **Class 1**: The performance is exceptional, with precision, recall, and F1 score all close to 1, indicating that the model classifies this class very accurately.
- **Class 2**: The performance is weaker, with a recall of only 0.40 and an F1 score of 0.50, indicating that the model's ability to identify this class is lacking. This may be due to the limited number of samples in this class (support count of 10), which prevented the model from adequately learning its features.
- **Macro Avg**: The macro average is 0.78, reflecting the model's average performance across all categories. The poor performance of Class 2 lowers the overall average.
- **Weighted Avg**: The weighted average is 0.94, benefiting from the higher weight of Class 1's larger sample size.



```
Accuracy: 0.9467
Classification Report:
              precision    recall  f1-score   support

           0       0.82      0.89      0.85        36
           1       0.97      0.98      0.97       254
           2       0.67      0.40      0.50        10

    accuracy                           0.95       300
   macro avg       0.82      0.76      0.78       300
weighted avg       0.94      0.95      0.94       300
```

Figure 12: Classification Report of NN

6.1.3 Confusion Matrix

- **Class 0**: Most samples were correctly classified, but a small number were misclassified as other categories.
- **Class 1**: Almost all samples were correctly classified, with only a few misclassifications.
- **Class 2**: A considerable number of samples were misclassified as other categories, especially Class 1. The high misclassification rate may be due to the limited number of samples, which hindered the model's ability to learn their features.

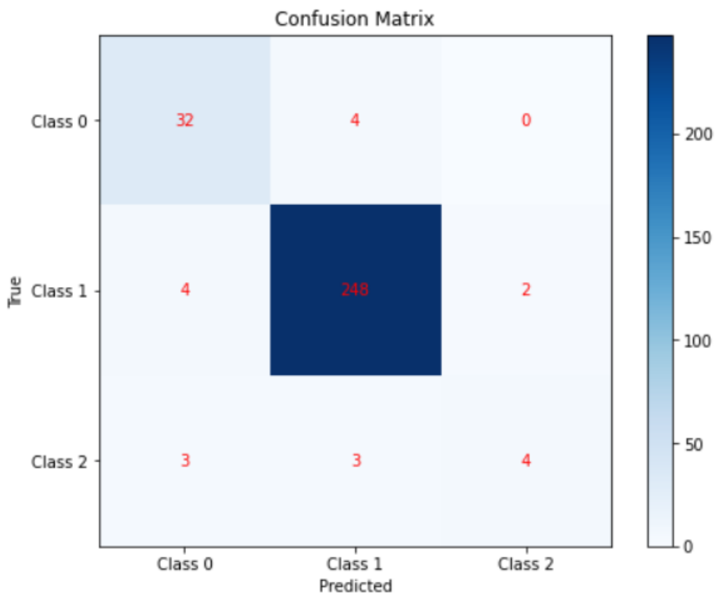


Figure 13: Confusion Matrix of NN

6.1.4 ROC Curve and AUC Value

The AUC values for all categories are close to 1, indicating that the model performs very well in classifying different categories. Although Class 2 has a lower recall, its AUC value remains high, demonstrating that the model performs well in distinguishing Class 2 from other categories.

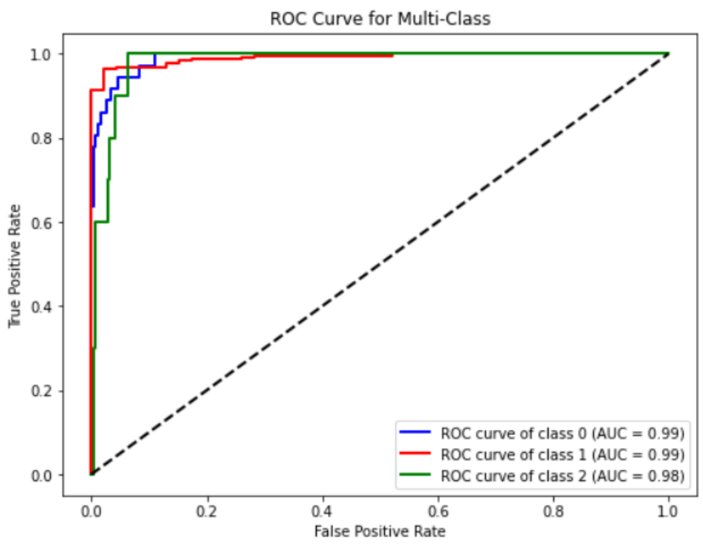


Figure 14: ROC Curve of NN

6.2 XGBoost Model

6.2.1 Accuracy

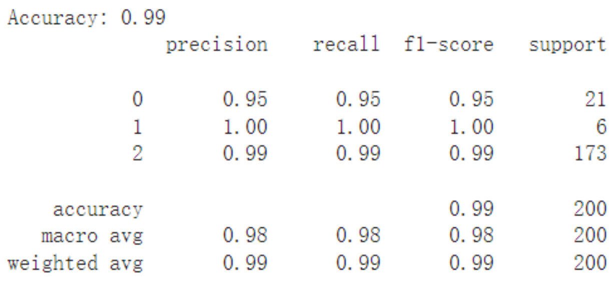
The overall accuracy of the XGBoost model on the testing set was 99%, indicating that the model performs exceptionally well, correctly classifying the vast majority of samples. The high accuracy signifies that the model performs well in the overall classification task.

6.2.2 Classification Report

- **Class 0**: Precision, recall, and F1 score are all 0.95, indicating that the model performs well in classifying this category, albeit with some misclassifications.
- **Class 1**: Precision, recall, and F1 score are all 1.00, indicating perfect classification for this category, with all samples correctly identified.
- **Class 2**: Precision, recall, and F1 score are all 0.99, indicating that the model also performs very well in classifying this category, with almost no misclassification.
- **Macro Avg**: The macro average is 0.98, reflecting the model's average performance across all categories. Class 0's slightly lower performance

than other categories results in a slightly lower macro average.

- **Weighted Avg**: The weighted average is 0.99, benefiting from the higher sample size (support count of 173) for Class 2.



```
Accuracy: 0.99
              precision    recall  f1-score   support

           0       0.95      0.95      0.95        21
           1       1.00      1.00      1.00         6
           2       0.99      0.99      0.99       173

    accuracy                           0.99       200
   macro avg       0.98      0.98      0.98       200
weighted avg       0.99      0.99      0.99       200
```

Figure 15: Classification Report of XGBoost Model

6.2.3 Confusion Matrix

- **Class 0**: 20 samples were correctly classified, but 1 was misclassified as Class 2.
- **Class 1**: All samples were correctly classified, with no misclassifications.
- **Class 2**: 172 samples were correctly classified, but a few were misclassified as Class 0.

Overall, the classification ability of the XGBoost model is very strong, especially with nearly perfect classification for Class 1 and Class 2, while misclassifications for Class 0 were minimal.

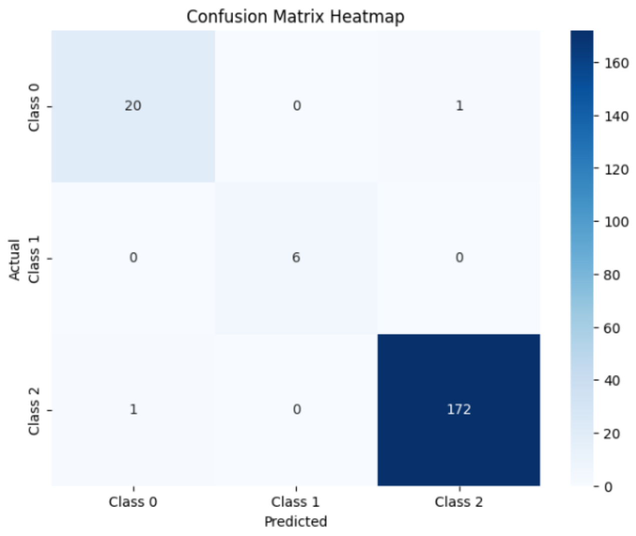
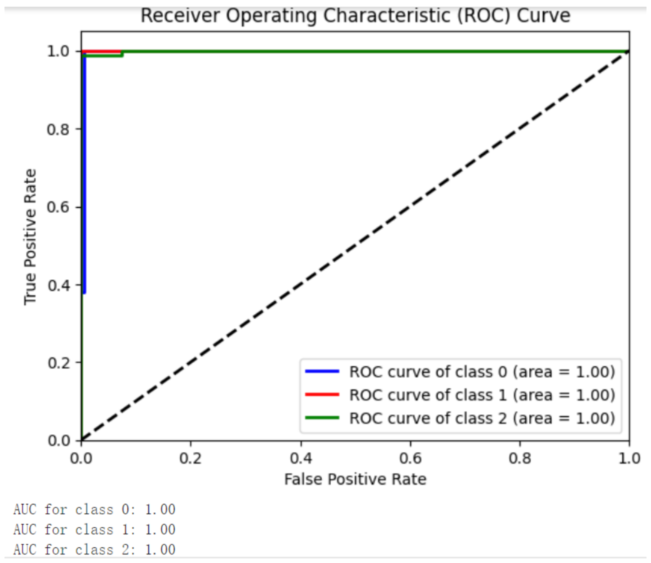


Figure 16: Confusion Matrix of XGBoost Model

6.2.4 ROC Curve and AUC Value

The AUC values for all categories are 1.00, indicating that the model performs exceptionally well in classifying different categories. The ROC curve is close to the top left corner, suggesting that the model has a high true positive rate and a low false positive rate in distinguishing between different categories.



```
AUC for class 0: 1.00
AUC for class 1: 1.00
AUC for class 2: 1.00
```

Figure 17: ROC Curve of XGBoost Model

6.3 Ensemble Model

6.3.1 Accuracy

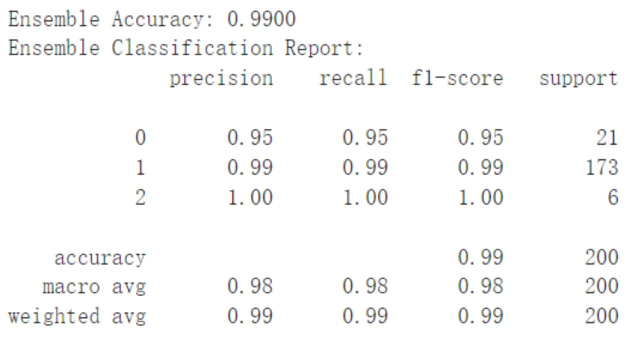
The overall accuracy of the ensemble model was 99.00%, indicating that the model performs exceptionally well on the testing set, correctly classifying the vast majority of samples. High accuracy signifies that the model performs well in the overall classification task.

6.3.2 Classification Report

- **Class 0**: Precision, recall, and F1 score are all 0.95, indicating that the model performs well in classifying this category, with some misclassifications.
- **Class 1**: Precision, recall, and F1 score are all 0.99, indicating that the model performs very well in classifying this category, with nearly all samples correctly identified.
- **Class 2**: Precision, recall, and F1 score are all 1.00, indicating perfect classification for this category, with all samples correctly identified.
- **Macro Avg**: The macro average is 0.98, reflecting the model's average performance across all categories. Class 0's slightly lower performance

results in a slightly lower macro average.

- **Weighted Avg**: The weighted average is 0.99, benefiting from the higher sample size (support count of 173) for Class 1.



```
Ensemble Accuracy: 0.9900
Ensemble Classification Report:
              precision    recall  f1-score   support

           0       0.95      0.95      0.95        21
           1       0.99      0.99      0.99       173
           2       1.00      1.00      1.00         6

    accuracy                           0.99       200
   macro avg       0.98      0.98      0.98       200
weighted avg       0.99      0.99      0.99       200
```

Figure 18: Classification Report of Ensemble Model

6.3.3 Confusion Matrix

- **Class 0**: Most samples were correctly classified, but a small number were misclassified as other categories.
- **Class 1**: The vast majority of samples were correctly classified, with almost no misclassifications.
- **Class 2**: All samples were correctly classified, with no misclassifications.

Overall, the classification ability of the ensemble model is very strong, particularly with nearly perfect classification of Class 1 and Class 2, while misclassifications of Class 0 were minimal.

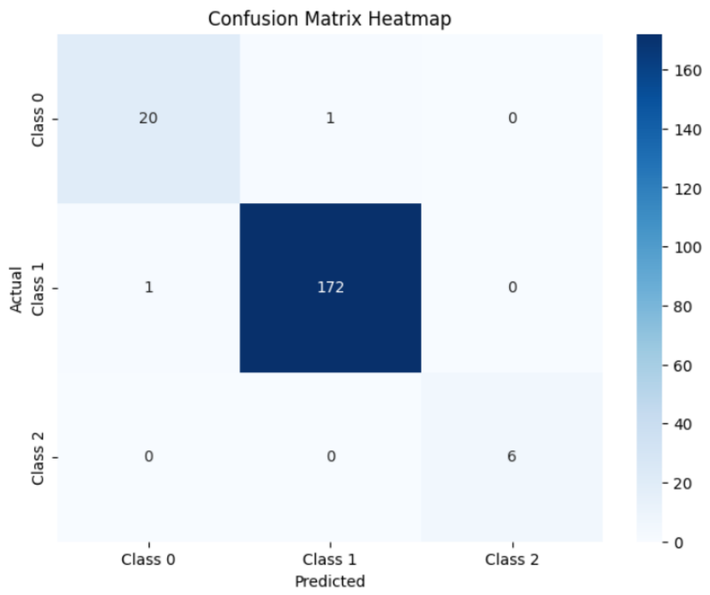
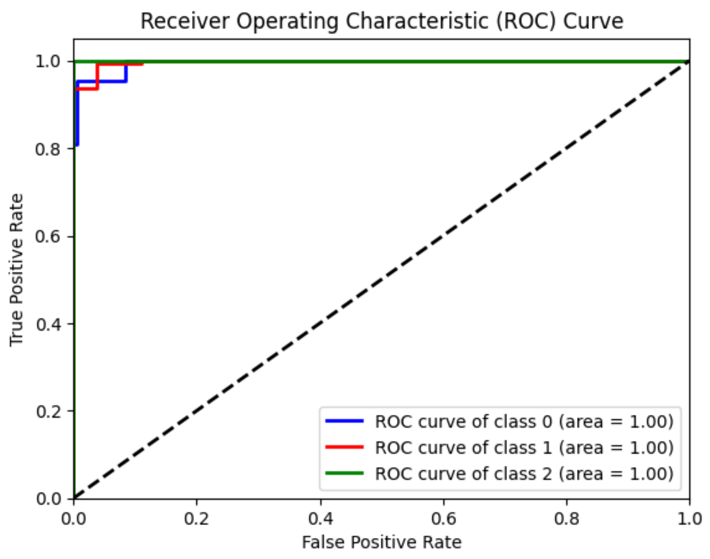


Figure 19: Confusion Matrix of Ensemble Model

6.3.4 ROC Curve and AUC Value

The AUC values for all categories are 1.00, indicating that the model performs exceptionally well in classifying different categories. The ROC curve is close to the top left corner, demonstrating that the model exhibits a high true positive rate and a low false positive rate when distinguishing between these categories. Although Class 0 has a small number of misclassifications, its AUC value remains at 1.00, indicating excellent performance in distinguishing these categories.



```
AUC for class 0: 1.00
AUC for class 1: 1.00
AUC for class 2: 1.00
```

Figure 20: ROC Curve of Ensemble Model

6.4 Comparison of the Three Models

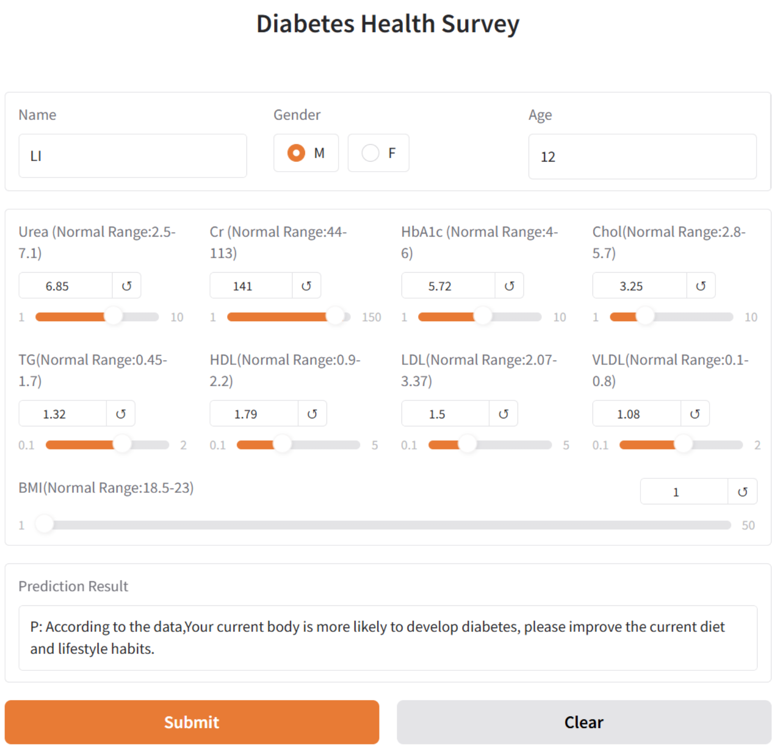
- **Accuracy**: The ensemble model and XGBoost model both achieved an accuracy of 99%, demonstrating exceptional performance. In contrast, the neural network model had an accuracy of 94.67%, which is significantly lower than the former two models.
- **Classification Report**: The ensemble model and XGBoost model both exhibited excellent classification performance across all categories, particularly with near-perfect classification of Class 1 and Class 2. The neural network model performed poorly on Class 2, with a recall of only 0.40 and an F1 score of 0.50.
- **Confusion Matrix**: The confusion matrices for the ensemble model and XGBoost model show nearly no misclassifications. The neural network model displayed numerous misclassifications within Class 2.
- **ROC Curve and AUC Value**: The AUC values for both the ensemble model and XGBoost model were 1.00, indicating the best performance. Although the neural network model also achieved a relatively high AUC value, it was slightly lower than that of the other two models.

In summary, the ensemble model and XGBoost model performed exceptionally well, with both achieving an accuracy of 99%, alongside excellent classification reports and AUC values. In contrast, the neural network model's performance was notably inferior, especially regarding Class 2's classification. Therefore, the ensemble model and XGBoost model are more suitable for practical applications in this task.

7 Development of a User-Friendly Interface

In this project, we designed an intuitive and easy-to-use user interface that enables users to quickly conduct diabetes risk assessments. The main functionalities and features of the interface are as follows:

- **Data Input Area**: Users can input various health indicators, including gender, age, urea, HbA1c, cholesterol, etc. Next to each indicator, the normal range is displayed to assist users in understanding and entering data accurately. (For example, the normal range for HbA1c is between 4% and 6%, but in the dataset, it corresponds to 4.0 to 6.0, thus users should input according to this feature.)
- **Risk Assessment Results**: After submitting data, the system will conduct risk assessments using ensemble learning models (such as neural networks and XGBoost) and clearly display the prediction results in the results area. We categorize the results into three types: Normal (N), Potential Risk (P), and High Risk (Y), providing users with intuitive feedback.
- **Model Support**: The interface utilizes ensemble learning models to ensure high accuracy and stability through effective combinations of different algorithms. After multiple rounds of testing, we confirmed that as long as the input data falls within the normal range, the model's risk assessment consistently indicates "Normal," demonstrating robust predictive capabilities.
- **User-Friendliness**: The entire interface is designed to be clean and simple, allowing users to easily input data and view results, thereby reducing the learning curve and increasing efficiency.


Diabetes Health Survey

Name: LI
Gender: [✓ M] F
Age: 12

Urea (Normal Range:2.5-7.1): 6.85 (1–10)
Cr (Normal Range:44-113): 141 (1–150)
HbA1c (Normal Range:4-6): 5.72 (1–10)
Chol(Normal Range:2.8-5.7): 3.25 (1–10)
TG(Normal Range:0.45-1.7): 1.32 (0.1–2)
HDL(Normal Range:0.9-2.2): 1.79 (0.1–5)
LDL(Normal Range:2.07-3.37): 1.5 (0.1–5)
VLDL(Normal Range:0.1-0.8): 1.08 (0.1–2)
BMI(Normal Range:18.5-23): 1 (1–50)

Prediction Result

P: According to the data,Your current body is more likely to develop diabetes, please improve the current diet and lifestyle habits.

Figure 21: UI

8 Conclusion

This project is dedicated to leveraging machine learning technology to enhance the early detection and intervention capabilities for diabetes. In the early stages of the project, we conducted in-depth exploratory data analysis to identify key factors closely associated with diabetes risk and selected the most suitable samples from three publicly available datasets.

During the data preprocessing phase, we employed various methods to fill in missing values, perform feature selection, and standardize the data, ensuring high data quality and providing a solid foundation for subsequent model construction.

In the model development process, we compared multiple algorithms, including neural networks, XGBoost, and ensemble learning, ultimately finding that both the XGBoost and ensemble models exhibited superior accuracy and stability, especially in handling complex feature relationships. This not only validated the effectiveness of machine learning in managing complex feature relationships but also provided reliable technical support for practical applications.

To facilitate clinical use, we also developed a user-friendly interface that allows healthcare professionals to quickly apply this model.

In conclusion, this project not only showcases the potential of machine learning in public health but also provides us with valuable experience in data analysis and model construction. In the future, we will continue to explore new technologies and strive to enhance model performance to better serve the early identification and intervention of diabetes.

9 Others

GitHub link:GitHub Repository