**CDS524: Machine Learning**

**Group projects**

**team 8**

**Final Report**



Project Name: Machine Learning for Early Detection of Diabetes

Members: SHANGGUAN Lianyue 5575608

Chen Jun 5582869

Zhao Xiaohang 5573246

He Jialu 5505407

Ma Yunfei 5575660

1. **Introduction**

* **Research Topic**

The research topic of this project is the early detection of diabetes using machine learning techniques. Diabetes is a chronic metabolic disorder that has become a significant global health concern, affecting over 500 million people worldwide. It is a leading cause of morbidity and mortality, contributing to a wide range of serious complications, including cardiovascular diseases, kidney failure, blindness, and neuropathy. The economic burden of diabetes is substantial, with billions spent annually on direct healthcare costs, medication, and the management of related complications. Moreover, the indirect costs, such as loss of productivity and reduced quality of life, further exacerbate its impact.

Early detection of diabetes is crucial for preventing complications, managing the disease effectively, and reducing healthcare costs. However, current diagnostic methods, such as fasting plasma glucose and HbA1c tests, are primarily reactive and are often applied at advanced stages of the disease when complications have already developed. These traditional approaches are limited in their ability to predict risk in asymptomatic individuals or integrate multifactorial data.

* **Target Audience**

The target audience for this research includes healthcare professionals, researchers, and policymakers. By providing a predictive model for early diabetes detection, we aim to support healthcare providers in implementing timely interventions and improving patient outcomes. This research can also inform policymakers about the potential benefits of integrating machine learning tools into clinical workflows.

* **Report Review**

This report will review the current state of diabetes diagnosis, highlighting the limitations of traditional methods and the potential of machine learning to address these challenges. It will describe the methodology used in developing the predictive model, including data collection, preprocessing, model development, and evaluation. The report will evaluate the model's performance using various metrics and discuss its scalability. Additionally, it will outline the development of a user-friendly interface and propose practical guidelines for integrating machine learning tools into clinical settings.

1. **Problem Description**

* **Description**

The problem addressed in this project is the development of an accurate and efficient method for the early detection of diabetes. Traditional diagnostic methods, such as fasting plasma glucose and HbA1c tests, are primarily reactive and often applied at advanced stages of the disease. These methods lack the ability to predict risk in asymptomatic individuals or integrate multifactorial data, leading to delayed diagnosis and increased risk of complications.

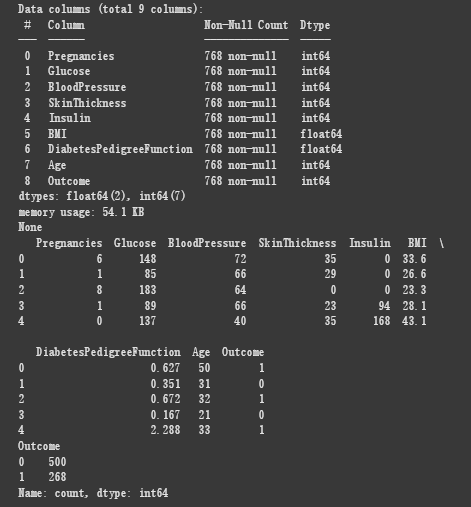
* **Why We Are Interested**

We are interested in developing a machine learning model for early diabetes detection because it offers several advantages over traditional methods. ML algorithms can analyze complex, multidimensional data to identify hidden patterns and relationships, improving predictive accuracy. This approach enables personalized risk assessments tailored to individual patients, facilitating timely interventions and reducing healthcare costs. By leveraging ML, we aim to transform the landscape of diabetes prevention and management, paving the way for more proactive and cost-effective healthcare solutions.

In summary, this project seeks to leverage the power of machine learning to identify early markers of diabetes and enable timely interventions, thereby improving patient outcomes and reducing the overall burden of diabetes on healthcare systems.

1. **Data Preprocessing**
2. **Dataset 1 : diabetes.csv ( Data Cleaning and Feature Selection )**

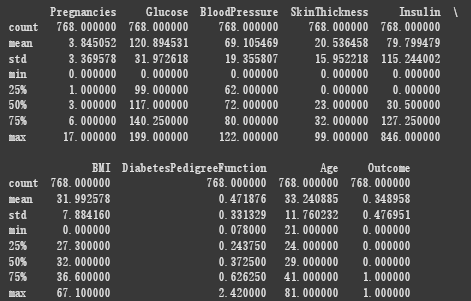
During the preprocessing process, the pandas library is used to load the `diabetes.csv` file, and a preliminary understanding of its basic information is obtained. Specifically, the `df.info()` and `df.head()` methods are employed to quickly grasp the structure of the data and the content of the first few rows. Additionally, `df["Outcome"].value\_counts()` is used to display the distribution of the target column "Outcome," which helps determine whether the data is balanced and whether further processing is required in the subsequent stages. The resulting data is shown in the figure below:



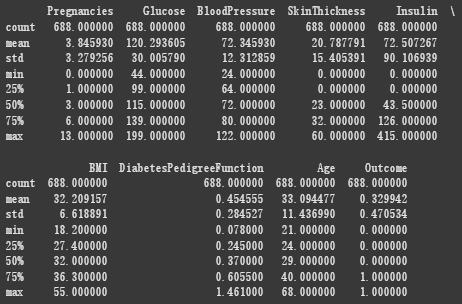
**Fig 3.1.1. Raw data information**

As can be seen, there are 500 entries with a value of 0 in the "Outcome" column, while only 268 entries have a value of 1, indicating that special processing should be applied to this data in the next steps. Subsequently, the `df.isnull().sum()` method is used to count the number of missing values in each column to check for the presence of any missing data. Following this, the `df.fillna(df.median(), inplace=True)` method is employed to fill the missing values with the median, ensuring the completeness of the data.

In the next step, to identify and handle outliers in the data, the code uses the descriptive statistics method `df.describe()` to initially observe the distribution of the data. Subsequently, the Z-score of the numerical features is calculated using `scipy.stats.zscore`, a standardization method that measures the deviation of data points from the mean. This helps identify outliers in the data, as data points with an absolute Z-score greater than 3 are typically considered outliers. Therefore, the condition `(abs\_z\_scores < 3).all(axis=1)` is used to filter out data points with a Z-score less than 3, removing potential outliers and thereby reducing the impact of anomalous data on the model. As shown in the comparison below, a total of 100 outliers were removed after this outlier treatment.



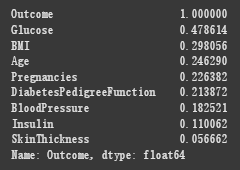
**Fig 3.1.2. Data before handling outliers**



**Fig 3.1.3. Data after handling outliers**

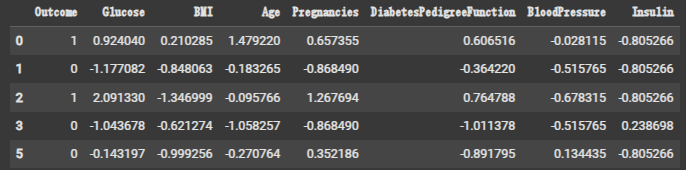
Next, the code separates the features and the target column, with X representing the features and y representing the target. To further optimize the features, the numerical features are standardized using `StandardScaler`, ensuring that different features have the same scale, which facilitates the training of subsequent models.

After processing the features, the code recombines the standardized features with the target column into `df\_p` and calculates the correlation between the features and the target column. By using `df\_p.corr()["Outcome"].abs().sort\_values(ascending=False)`, it is possible to intuitively see which features have a higher correlation with the target column, as shown in the result below:



**Fig 3.1.4. Feature correlation coefficient**

Finally, the code filters out features with a correlation greater than 0.1, further refining the dataset and retaining features that have a stronger relationship with the target column. This provides a higher-quality data foundation for subsequent model training. Through these steps, the data is effectively cleaned and preprocessed. Below is the final processed data format:



**Fig 3.1.5. Processed data**

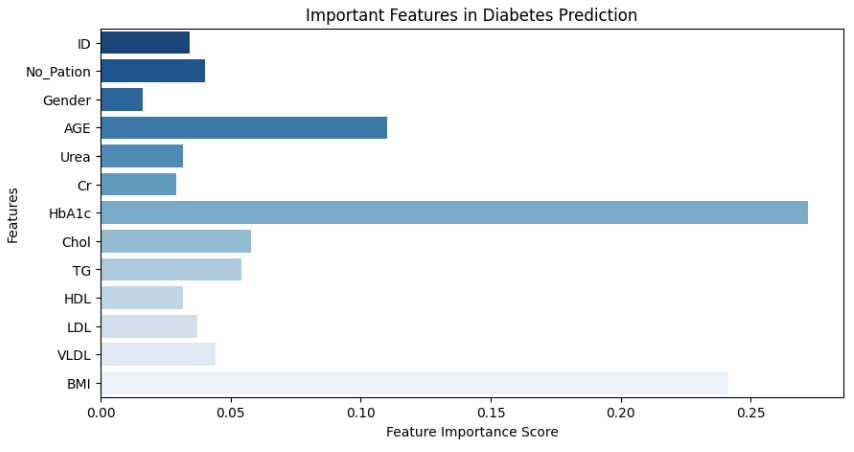
To address the aforementioned issue of class imbalance, the code uses oversampling techniques to adjust the data distribution. First, the minority class is oversampled using the SMOTE method from `imblearn.over\_sampling`. SMOTE (Synthetic Minority Over-sampling Technique) increases the number of minority class samples by generating synthetic samples, thereby balancing the class distribution. In the code, `smote.fit\_resample(X, y)` performs oversampling on the features X and the target column y, generating a new balanced dataset X\_res and y\_res. By using `y\_res.value\_counts()`, the balanced class distribution can be viewed, ensuring that the number of samples in the minority and majority classes is close.

Next, the code uses the `train\_test\_split` method from `sklearn.model\_selection` to divide the balanced dataset into training and test sets. Here, `test\_size=0.2` indicates that the test set accounts for 20% of the total data, `random\_state=42` ensures consistent results across runs, and `stratify=y\_res` guarantees that the class distribution in the training and test sets matches that of the original data. By using `y\_train\_res.value\_counts()` and `y\_test.value\_counts()`, the class distributions of the training and test sets can be viewed separately, ensuring that the data remains balanced after splitting.

1. **Dataset 2 : Dataset of Diabetes .csv ( Data Preprocessing and Visualization )**

During the data preprocessing for dataset 2, the basic logic would be similar to the other datasets, which is clear out the missing value, normalization and standardize, and encode the catogorical value. So for dataset 2, the categorical values would be the “Gender” column and the “CLASS” column, which I assume would stands for “classification”. During the encoding, the problem occured as the unique value of these two columns are inadequatly filled. There are “f” in “Gender” column aside with “M” and “F”, and for “CLASS” column, there are “Y[space]” and “N[space]”. So the statement”print(df1[‘CLASS’].unique())”(Here we take “CLASS” column as an example.) is quite important when you don’t really know exactly how many unique values in your categorical column.

So after all of the preprocessing above, the result shows us the top three most important features are: **HbA1c**, **BMI** and **AGE**.



**Fig 3.2.1. Important Features in Diabetes Prediction**

1. **Dataset 3 : diabetes.zip ( Data Integration, Data Cleaning and Data Interpretation )**

**Data Integration and Cleaning**

The data used in this part was collected from 70 diabetes patients and integrated into a single dataset. During preprocessing:

* **Handling Missing Values:**

Non-numeric data was converted to NaN, and missing values were imputed using the median of the respective columns. This ensured that the dataset had no missing values and was ready for further analysis.

* **Feature Engineering:**

New temporal features were created by extracting hour, day, and month from the datetime information. This allowed the models to potentially capture time-related patterns in blood glucose levels.

* **Target Variable Definition**

A binary target variable 'Outcome' was created, where blood glucose values over 200 were classified as positive for diabetes (1), and others as negative (0).

* **Data Splitting and Standardization**

The dataset was split into training (80%) and testing (20%) sets. Features were standardized using StandardScaler to ensure consistent scales, which is crucial for many machine learning algorithms.

**Data Interpretation**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Code** | **count** | **mean** | **std** | **min** | **25%** | **50%** | **75%** | **max** |
| **33** | 9518.0 | 6.670939 | 5.487383 | 0.0 | 4.0 | 6.0 | 9.00 | 344.0 |
| **34** | 3830.0 | 15.393603 | 14.155106 | 1.0 | 7.0 | 15.0 | 20.00 | 388.0 |
| **35** | 1053.0 | 15.858500 | 7.808906 | 3.0 | 10.0 | 15.0 | 21.00 | 30.0 |

The dataset contained various insulin dose records and blood glucose measurements. Key observations included:

* **Regular Insulin Dose (Code 33)**

This was the most frequently recorded insulin type, with a wide range of doses (0-344 units). The average dose was 6.67 units, indicating that most patients used lower doses, but some required very high doses.

* **NPH Insulin Dose (Code 34)**

This had a moderate number of records, with an average dose of 15.39 units. The doses varied significantly, from 1 to 388 units.

* **UltraLente Insulin Dose (Code 35)**

This had fewer records, with a narrower dose range (3-30 units) and an average of 15.86 units, suggesting more consistent usage.

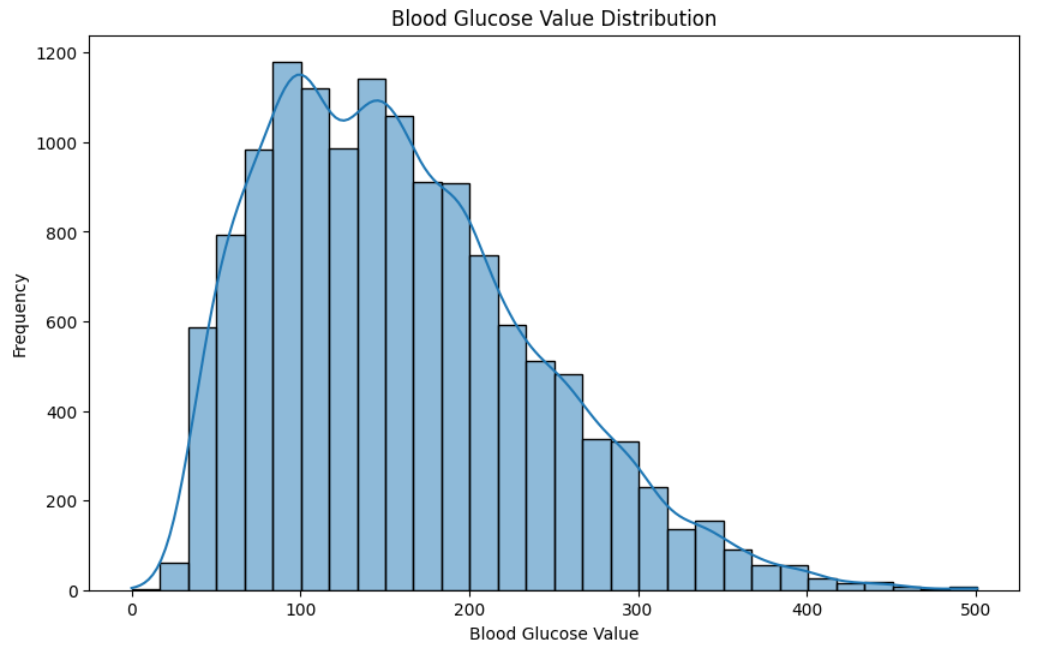
**Data Visualization**

* **Visualization**

The histogram displayed the frequency distribution of blood glucose values, with a kernel density estimate (KDE) overlay to show the underlying distribution pattern.

* **Result**

The visualization revealed that most blood glucose values were concentrated in the lower range, with a significant number of values below 200. This indicated that the majority of readings were within the normal or prediabetic range, while values above 200 (indicative of diabetes) were less frequent but still present.



**Fig 3.3.1. Blood Glucose Value Distribution**

1. **Model Selection and Implementation**

During the model selection and training process, we initially attempts several classic machine learning models, including Logistic Regression, Random Forest, XGBoost, and LightGBM.

Additionally, most models in this project utilized grid search for hyperparameter tuning. Grid Search is a hyperparameter optimization method that exhaustively evaluates all possible parameter combinations within a predefined parameter grid, using cross-validation to assess the performance of each combination, thereby identifying the optimal hyperparameter configuration. In the final organized code, the hyperparameter tuning used to optimize the Random Forest model is retained as an example. In this example, grid search is employed to tune various parameters of the model, such as the number of trees, maximum depth, minimum samples split, etc., ultimately selecting the best-performing parameter combination to enhance the model's prediction accuracy and generalization capability.

**Dataset 1 :**

* **Logistic Regression**

Logistic Regression was chosen because it is a simple and efficient linear classification model, particularly suitable for handling binary classification problems. In this project, Logistic Regression can quickly fit the data, and its model offers good interpretability, allowing for an intuitive understanding of the impact of each feature on the prediction results through its coefficients. Although Logistic Regression has limited capability in handling nonlinear relationships, after standardizing the features, it still provides a reliable baseline model for the project.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Parameter Meaning | Value |
| max\_iter | Maximum number of iterations | 100 |
| C | Regularization strength | 0.01 |
| slover | Optimization algorithm | lbfgs |
| random\_state | Random seed | 42 |

* **Multinomial Logistic Regression**

Unlike linear regression, which predicts continuous outcomes, logistic regression is designed for predicting categorical outcomes. As one of the regression models, it can be used to predict the probability of a binary outcome or for performing classification tasks. However, logistic regression is used for predicting binary outcomes like Yes or No, and for that, we have Multinomial Logistic Regression for our dataset which contains three outcomes, Y / N / P.

Multinomial Logistic Regression can handle categorical variables like different flavors of ice cream to choose, and can be quite flexible with independent variables. In the model implementation of Multinomial Logistic Regression, I used the one from sklearn library. At first the accuracy is not so satisfying with 0.86, and the accuracy of “0” type(N) and “2” type(P) are 0, I resigned weight for the top three important feature that mentioned in data preprocessing step as 2.0, and it turns out to have the overall accuracy of 0.94. There are still problem with the accuracy of the “2”type(P), I assume that might because of low data in this type.

* **Random Forest**

Random Forest is an ensemble learning method that can handle nonlinear relationships and complex feature interactions. By constructing multiple decision trees and averaging or voting on their results, Random Forest effectively reduces the risk of overfitting associated with a single decision tree. In this task, Random Forest can capture complex relationships between features, such as the interactions between blood glucose levels, BMI, and age. Another advantage is its ability to provide feature importance scores, helping to analyze which features contribute the most to the prediction results, thereby offering guidance for feature engineering.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Parameter Meaning | Value |
| n\_estimators | Number of decision trees | 50 |
| max\_depth | Maximum depth of the tree | 10 |
| min\_samples\_leaf | Minimum number of samples required at a leaf node | 5 |
| min\_samples\_split | Minimum number of samples required to split a node | 5 |
| random\_state | Random seed | 42 |

* **Gradient Boosting**

XGBoost is an efficient gradient boosting algorithm that performs exceptionally well in handling structured data and classification tasks. By progressively optimizing the model, XGBoost effectively reduces bias and variance, thereby achieving high prediction accuracy in the diabetes prediction task. Its advantages include support for parallel computing, regularization terms, and custom loss functions, which help prevent overfitting. Additionally, by setting `eval\_metric="logloss"`, XGBoost directly optimizes the log loss, which is highly effective in binary classification tasks.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Parameter Meaning | Value |
| colsample\_bytree | Proportion of features used by each tree | 0.9 |
| min\_child\_weight | Minimum sum of sample weights required for child nodes | 5 |
| learning\_rate | Learning rate | 0.1 |
| max\_depth | Maximum depth of the tree | 15 |
| n\_estimators | Number of trees | 25 |
| subsample | Proportion of samples used by each tree | 0.8 |
| eval\_metric | Evaluation indicators | logloss |

LightGBM is a decision tree-based gradient boosting framework that exhibits extremely high efficiency when handling large-scale data and high-dimensional features. By utilizing a histogram-based decision tree algorithm and a leaf-wise growth strategy, LightGBM significantly accelerates training speed and reduces memory usage. In prediction tasks, LightGBM can quickly process multiple features and flexibly adjust model complexity through the corresponding parameter settings. Another advantage is its support for handling categorical features and automatic feature selection, which further simplifies the feature engineering process. Additionally, by setting `reg\_alpha` and `reg\_lambda`, LightGBM effectively prevents overfitting and enhances the model's generalization capability.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Parameter Meaning | Value |
| colsample\_bytree | Proportion of features used by each tree | 10 |
| min\_child\_samples | Minimum number of samples required at a leaf node | 15 |
| reg\_alpha | L1 regularization coefficient | 0.2 |
| reg\_lambda | L2 regularization coefficient | 0.3 |
| subsample | Proportion of samples used by each tree | 0.7 |
| num\_leaves | Number of leaf nodes | 18 |
| learning\_rate | Learning rate | 0.05 |
| n\_estimators | Number of trees | 500 |

* **Neural Network**

In this project, a neural network is used to explore a model capable of capturing complex nonlinear relationships, thereby further improving prediction performance. It can learn highly abstract feature representations from data through multiple layers of nonlinear transformations, making it particularly suitable for tasks where features may have complex interactions. In this task, features such as blood glucose levels, BMI, and age of patients may not be simple linear combinations but rather influence the prediction results through complex nonlinear relationships. The neural network is introduced to capture these underlying patterns. By constructing a fully connected network with an input layer, hidden layers, and an output layer, the model can automatically learn high-order interactions between features without the need for manual feature engineering. Moreover, the neural network demonstrates its flexibility and powerful expressive capabilities in this project.

Specifically, the code defines a two-layer neural network, with the first layer containing 64 neurons and the second layer containing 32 neurons, both using the ReLU activation function. The output layer uses the Sigmoid activation function for binary classification prediction. This structure effectively extracts useful information from all features and fits the data by optimizing the binary cross-entropy loss function. Additionally, by setting the Adam optimizer and a learning rate of 0.01, the neural network can quickly converge and find better model parameters.

The advantages of the neural network are mainly reflected in the following aspects: First, it has strong nonlinear modeling capabilities, enabling it to capture complex relationships between features, which is particularly important in the diabetes prediction task. Second, the neural network optimizes through gradient descent and backpropagation algorithms, automatically adjusting model parameters and reducing reliance on manual tuning. Third, the neural network supports flexible architecture design, allowing adjustments to the number of layers, neurons, and activation functions based on data characteristics to adapt to different task requirements. Finally, the neural network performs well in handling high-dimensional data, although this project only uses 7 features, so this advantage may be somewhat limited.

However, the disadvantages of neural networks cannot be ignored, such as higher data requirements, longer training times, and poor model interpretability. In this project, the dataset size is moderate, and the number of features is small, so these drawbacks are within acceptable limits.

* **Ensemble Model**

In this project, to further enhance the model's prediction performance, the integration of multiple base models was considered to generate more robust and accurate predictions. The fundamental idea of ensemble models is that by combining the predictions of multiple models, the shortcomings of individual models can be mitigated, the risk of overfitting can be reduced, and generalization ability can be improved. In the diabetes prediction task, due to the potential existence of complex nonlinear relationships among data features, a single model may not fully capture these patterns. However, ensemble learning, by combining the predictions of multiple models, can more comprehensively extract information from the data. This project primarily employed two ensemble learning methods: Stacking and Voting.

Stacking works by using the predictions of multiple base models (in this project, Random Forest and LightGBM) as meta-features, which are then input into a meta-model (in this project, Logistic Regression or Random Forest) for secondary training, thereby generating the final predictions. In this project, Stacking was used to combine the strengths of Random Forest and LightGBM. Random Forest can capture complex interactions between features, while LightGBM is known for its efficient computational capabilities and support for categorical features. Through Stacking, the predictions of these base models are further optimized, as the meta-model learns the complementary relationships between the base models, resulting in more accurate predictions. The advantages of Stacking lie in its flexibility and strong expressive power, which can further improve model performance through cross-validation and probability prediction (`stack\_method='predict\_proba'`).

Voting, on the other hand, combines the predictions of multiple base models through hard or soft voting strategies to generate the final prediction. In this project, Voting was used to integrate the strengths of Logistic Regression, Random Forest, and XGBoost. Logistic Regression provides a simple and interpretable baseline model, Random Forest can capture complex feature relationships, and XGBoost is renowned for its efficient gradient boosting algorithm. This project adopted the soft voting strategy (`voting="soft"`), as soft voting considers the predicted probabilities of each model rather than just the final classification results. In soft voting, the predicted probabilities of each class from each model are weighted and averaged to generate the final prediction. This approach better reflects the uncertainty of the models, especially in cases where class boundaries are ambiguous or model predictions are close, resulting in more robust and accurate predictions. In contrast, hard voting only considers the classification results of the models, potentially ignoring subtle differences between models and leading to information loss. Through these strategies, Voting can comprehensively consider the predicted probabilities of each model, thereby generating more robust predictions.

**Dataset 2 :**

* **Multinomial Logistic Regression**

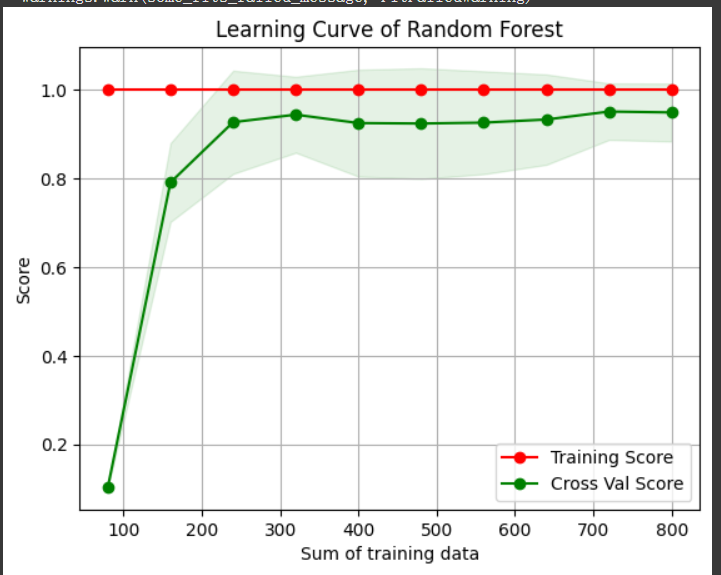
Unlike linear regression, which predicts continuous outcomes, logistic regression is designed for predicting categorical outcomes. As one of the regression models, it can be used to predict the probability of a binary outcome or for performing classification tasks. However, logistic regression is used for predicting binary outcomes like Yes or No, and for that, we have Multinomial Logistic Regression for our dataset which contains three outcomes, Y / N / P.

Multinomial Logistic Regression can handle categorical variables like different flavors of ice cream to choose, and can be quite flexible with independent variables. In the model implementation of Multinomial Logistic Regression, I used the one from sklearn library. At first the accuracy is not so satisfying with 0.86, and the accuracy of “0” type(N) and “2” type(P) are 0, I resigned weight for the top three important feature that mentioned in data preprocessing step as 2.0, and it turns out to have the overall accuracy of 0.94. There are still problem with the accuracy of the “2”type(P), I assume that might because of low data in this type.

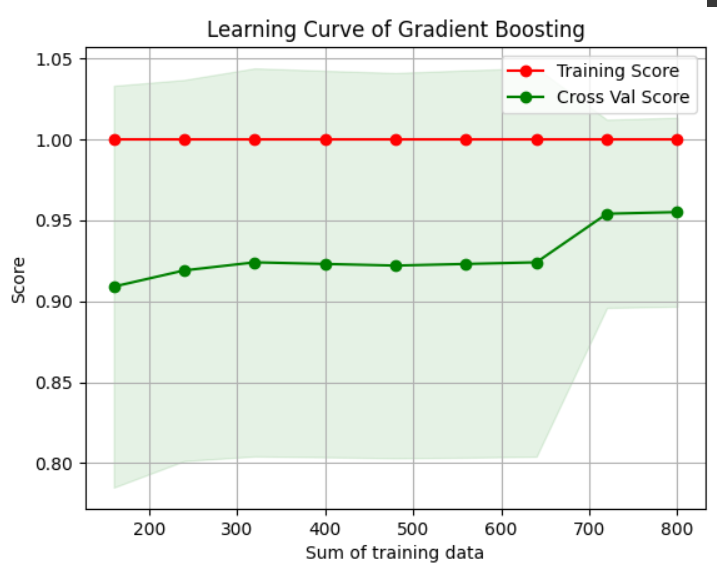
* **Random Forest and Gradient Boosting**

Random Forest is an ensemble learning method that constructs multiple decision trees using training and outputs the mode of the classes or mean prediction of the individual trees. It can handle missing value by constructing trees with available features only, and it’s robust to overfitting problem.

Gradient Boosting is another powerful ensemble technique that builds trees sequentially. Each new tree corrects the mistakes of the previous trees. Unlike Random Forest model, It might have overfitting issue, but It is more flexible, and when dealing with imbalanced data, it would performance better by focusing on difficult-to-predict instances. Just as how it performance with low data than Random Forest model in the chart below.



**Fig 4.1. Learning Curve of Random Forest**



**Fig 4.2. Learning Curve of Gradient Boosting**

**Dataset 3 :**

Three different machine learning models were implemented and evaluated:

* **Random Forest Classifier**

The RandomForestClassifier from sklearn.ensemble was used with 100 estimators and a random state of 42. The model was trained on the standardized dataset.

The model was initialized with n\_estimators=100 and random\_state=42 to ensure reproducibility. It was then fit to the training data and used to make predictions on the test set.

* **Logistic Regression**

The LogisticRegression model from sklearn.linear\_model was used with a maximum of 1000 iterations. The data was imputed using the mean strategy before training.

Due to some convergence warnings, a mean imputation strategy was applied to the training and test sets before model training. The model was then fit and used for prediction.

* **Gradient Boosting Classifier**

The GradientBoostingClassifier from sklearn.ensemble was used with 100 estimators and a random state of 42. The data was imputed using the mean strategy.

Similar to the Logistic Regression, mean imputation was applied to handle any remaining missing values. The model was then trained and evaluated.

1. **Results and Evaluation ( Taking Dataset1 as the main example )**

**Dataset 1 :**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Accuracy | Precision | Recall | F1-score | AUC-ROC |
| Logistic Regression | 0.768 | 0.795 | 0.717 | 0.754 | 0.882 |
| Random Forest | 0.854 | 0.835 | 0.880 | 0.857 | 0.922 |
| XGBoost | 0.827 | 0.794 | 0.880 | 0.835 | 0.908 |
| LightGBM | 0.854 | 0.842 | 0.870 | 0.856 | 0.913 |
| LRX\_Voting | 0.843 | 0.812 | 0.891 | 0.850 | 0.916 |
| LX\_Voting | 0.838 | 0.823 | 0.859 | 0.840 | 0.915 |
| RFLGB\_St(LR) | 0.822 | 0.817 | 0.826 | 0.822 | 0.933 |
| RFLGB\_St(RF) | 0.811 | 0.794 | 0.837 | 0.815 | 0.903 |

Neural Network:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Precision | Recall | F1-score | support |
| 0 | 0.87 | 0.82 | 0.84 | 93 |
| 1 | 0.83 | 0.88 | 0.85 | 92 |
| Accuracy |  |  | 0.85 | 185 |
| Macro avg | 0.85 | 0.85 | 0.85 | 185 |
| Weighted avg | 0.85 | 0.85 | 0.85 | 185 |

In this project, the performance of Logistic Regression is relatively basic, with an Accuracy of 0.768, F1-score of 0.754, and AUC-ROC of 0.882. The advantages of Logistic Regression lie in its simplicity and strong interpretability, enabling it to quickly fit the data and provide a reliable baseline model. However, due to the limitations of its linear assumptions, Logistic Regression performs weakly in handling complex nonlinear relationships, resulting in relatively lower scores in Recall and Precision. Nevertheless, the high AUC-ROC value of Logistic Regression indicates that it still holds an advantage in distinguishing between positive and negative class samples.

The performance of Random Forest is significantly better than that of Logistic Regression, with an accuracy of 0.854, F1-score of 0.857, and AUC-ROC of 0.922. By integrating multiple decision trees, Random Forest effectively captures complex interactions between features, achieving high scores in both Precision and Recall. Its strengths include the ability to handle nonlinear data, reduce the risk of overfitting, and provide feature importance scores, helping to understand which features contribute most to the prediction results. The performance of Random Forest in this project demonstrates that it is a robust and efficient classification model, particularly suitable for structured data.

The performance of XGBoost is close to that of Random Forest, with an accuracy of 0.827, F1-score of 0.835, and AUC-ROC of 0.908. As a gradient boosting algorithm, XGBoost optimizes the model step by step, effectively reducing bias and variance, thereby achieving a high Recall score (0.880). Its advantages include support for parallel computing, regularization terms, and custom loss functions, which prevent overfitting and enhance model performance. The performance of XGBoost in this project shows that it has high prediction accuracy and generalization capability in handling binary classification tasks.

The performance of LightGBM is comparable to that of Random Forest, with an accuracy and F1-score of 0.854 and an AUC-ROC of 0.913. LightGBM significantly speeds up training and reduces memory usage through its histogram-based decision tree algorithm and leaf-wise growth strategy. Its strengths include efficient handling of large-scale data, support for categorical features, and automatic feature selection, enabling it to generate high-precision predictions in a short time. The performance of LightGBM in this project demonstrates that it is an efficient and flexible model, particularly suitable for high-dimensional data and complex tasks.

The ensemble model (LRX\_Voting) using the Voting method to integrate Logistic Regression, Random Forest, and XGBoost achieves an accuracy of 0.843, F1-score of 0.850, AUC-ROC of 0.916, and Recall of 0.891. These metrics indicate that LRX\_Voting performs well in overall performance and the ability to identify positive class samples. By combining the strengths of Logistic Regression, Random Forest, and XGBoost, LRX\_Voting generates more robust and accurate predictions. Its Recall (0.891) is higher than that of individual models, indicating its outstanding ability to identify positive class samples (diabetes cases).

The ensemble model (LX\_Voting) using the Voting method to integrate Logistic Regression and XGBoost achieves an accuracy of 0.838, F1-score of 0.840, AUC-ROC of 0.915, and Recall of 0.859. These metrics indicate that this model's overall performance is slightly lower than the ensemble model integrating three models but still outperforms individual Logistic Regression and XGBoost models. By combining the strengths of Logistic Regression and XGBoost, it generates relatively robust predictions while maintaining high accuracy. Its AUC-ROC value (0.915) is close to that of the previous ensemble model, indicating its good ability to distinguish between positive and negative class samples. However, its Recall (0.859) and F1-score (0.840) are lower than those of LRX\_Voting, indicating slightly weaker performance in identifying positive class samples and balancing Precision and Recall. Additionally, due to the lack of Random Forest, it may not perform as well as the previous model in handling complex nonlinear relationships.

In summary, compared to individual models, these two ensemble models outperform individual Logistic Regression models in accuracy, F1-score, and AUC-ROC, demonstrating that ensemble learning can significantly improve model performance. Moreover, ensemble models reduce the bias and variance of individual models by combining the predictions of multiple base models, generating more robust predictions. Additionally, the Recall of LRX\_Voting (0.891) is significantly higher than that of all individual models, indicating the clear advantage of ensemble learning in identifying positive class samples. In practical applications, the choice of whether to use ensemble models can be based on specific needs. If higher Recall and robustness are desired, LRX\_Voting is the better choice; if a balance between performance and computational cost is preferred, LX\_Voting is a reasonable alternative.

The ensemble model (RFLGB\_St(LR)) using the Stacking method with Logistic Regression as the meta-model to integrate Random Forest and LightGBM achieves an accuracy of 0.822, F1-score of 0.822, AUC-ROC of 0.933, and Recall of 0.826. These metrics indicate that RFLGB\_St(LR) performs well in overall performance and the ability to distinguish between positive and negative class samples. Notably, its AUC-ROC value (0.933) is the highest among all models, indicating its exceptional ability to distinguish between positive and negative class samples. Additionally, its Precision (0.817) and Recall (0.826) are well-balanced, showing that the model achieves a good balance between reducing misjudgments and identifying positive class samples. However, its accuracy (0.822) is slightly lower than that of LRX\_Voting (0.843) and Random Forest (0.854), indicating slightly weaker overall classification performance.

The ensemble model (RFLGB\_St(RF)) using the Stacking method with Random Forest as the meta-model to integrate Random Forest and LightGBM achieves an accuracy of 0.811, F1-score of 0.815, AUC-ROC of 0.903, and Recall of 0.837. These metrics indicate that it performs better in Recall but slightly worse in other metrics compared to RFLGB\_St(LR). Specifically, its Recall (0.837) is higher than that of RFLGB\_St(LR), indicating better performance in identifying positive class samples (diabetes cases). However, its AUC-ROC value (0.903) and Precision (0.794) are lower than those of RFLGB\_St(LR), indicating slightly weaker performance in distinguishing between positive and negative class samples and reducing misjudgments.

Compared to individual models, RFLGB\_St(LR) and RFLGB\_St(RF) outperform individual Random Forest (0.922) and LightGBM (0.913) in AUC-ROC, demonstrating that the Stacking method can significantly improve the model's ability to distinguish between positive and negative class samples. Additionally, the Recall of RFLGB\_St(RF) (0.837) is higher than that of individual Random Forest (0.880) and LightGBM (0.870), indicating the advantage of the Stacking method in identifying positive class samples. This also shows that the Stacking method generates more robust predictions by combining the predictions of multiple base models, reducing the bias and variance of individual models. However, the Stacking method also has its drawbacks. For example, the accuracy of the two models mentioned above is lower than that of individual Random Forest (0.854) and LightGBM (0.854), indicating that the Stacking method may underperform in overall classification performance. Additionally, since the Stacking method requires training multiple base models and meta-models, it demands higher computational resources. Moreover, its predictions are determined by multiple base models and meta-models, resulting in lower interpretability compared to individual models.

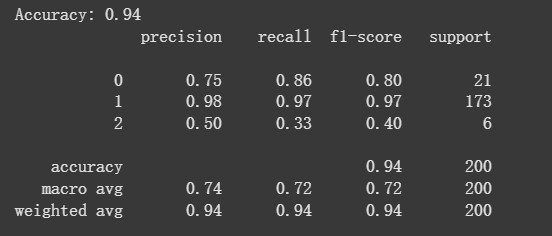
In this project, the neural network's prediction results show an accuracy of 0.85, an F1-score of 0.85, and Precision and Recall of 0.87 and 0.83 (for class 0), and 0.83 and 0.88 (for class 1), respectively. These metrics indicate that the neural network performs excellently in overall performance, particularly in balancing Precision and Recall. For class 0 (non-diabetic), the neural network's high Precision (0.87) demonstrates its ability to reduce misjudgments of non-diabetic patients. For class 1 (diabetic), the neural network's high Recall (0.88) highlights its strong capability in identifying positive class samples (diabetes cases). Although the AUC-ROC value of the neural network is not directly provided, its high F1-score and balanced metrics suggest that it also performs well in distinguishing between positive and negative class samples.

Compared to other models, the neural network's accuracy (0.85) is comparable to that of Random Forest (0.854) and LightGBM (0.854), but its Recall (0.88) is higher than most models, indicating its advantage in identifying positive class samples. When compared to ensemble models (such as LRX\_Voting and RFLGB\_St(LR)), the neural network's performance is close, though slightly inferior in some metrics. However, as an independent model, the simplicity and flexibility of the neural network make it an important choice in this project.

**Dataset 2 :**

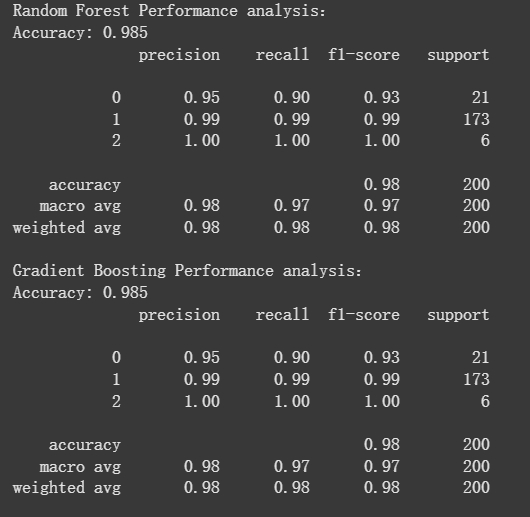
Here follow the results of each model and their classification report:

Multinomial Logistic Regression Model:



**Fig 5.1. Result of Multinomial Logistic Regression Model**

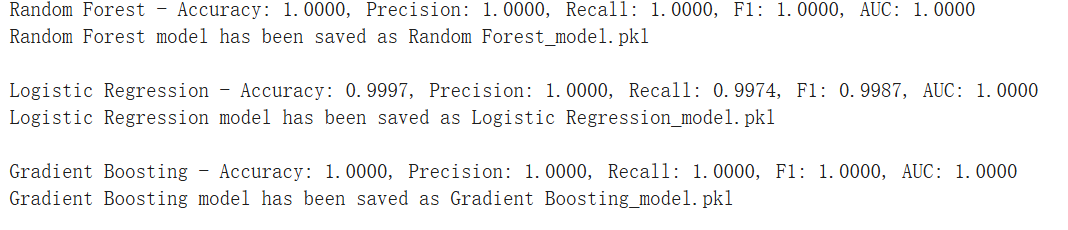
Random Forest Model and Gradient Boosting Model:



**Fig 5.2. Result of Random Forest Model and Gradient Boosting Model**

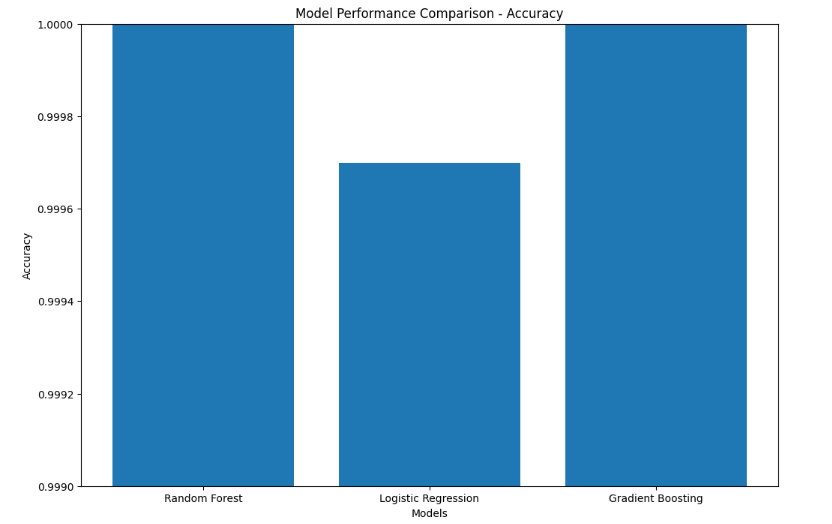
**Dataset 3 :**

The performance of each model was evaluated using several metrics including accuracy, precision, recall, F1-score, and AUC-ROC. The results were as follows:



**Fig 5.2. Results of three models**

A bar chart was created to visually compare the accuracy of the different models:



**Fig 5.3. Accuracy of the different models**

The visualization clearly showed that both Random Forest and Gradient Boosting models outperformed Logistic Regression in terms of accuracy, though all models showed high performance.

1. **User Interface**

We developed a practical interface using PyQt5, which divides the models trained on three different datasets based on various features into three sections. After running the program, users can navigate to different prediction interfaces through the main interface buttons, select the appropriate model for prediction, and the prediction results will be displayed in a designated area.

1. **Conclusion**

This machine learning modeling and data analysis based on the diabetes database has provided us with valuable experience and profound insights. By comparing the performance of various models such as logistic regression, random forest, XGBoost, LightGBM, and neural networks, we found that different models exhibit varying strengths when handling complex data. Selecting appropriate models and ensemble methods can effectively enhance the robustness and accuracy of the models. In our case, ensemble learning (such as Stacking and Voting) and neural networks performed particularly well in capturing complex nonlinear relationships, significantly improving prediction accuracy and generalization ability.

Based on the project research findings, we propose the following practical and evidence-based guidelines:

* **Prioritize ensemble learning and neural networks**

In diabetes prediction tasks, it is recommended to use ensemble learning (such as Stacking and Voting) or neural networks to capture complex nonlinear relationships in the data and improve prediction performance.

* **Emphasize model interpretability and practicality**

Although neural networks and ensemble learning perform excellently, the interpretability of simpler models like logistic regression remains valuable in practical applications, especially in the medical field, where it helps doctors and patients understand the prediction results.

* **Address class imbalance issues**

Class imbalance is common in diabetes datasets. Resampling techniques (such as SMOTE) can be used to optimize model performance and ensure better performance on minority classes.

* **Feature engineering and standardization**

Standardizing key features such as blood glucose levels, BMI, and age, and optimizing feature selection using feature importance analysis (e.g., feature scores provided by random forest) can further enhance model performance.

* **Model validation and cross-validation**

During model training, use cross-validation and rigorous performance evaluation metrics (such as AUC-ROC and F1-score) to validate the stability and reliability of the model and avoid overfitting.