**Data Mining and Foundation of AI**

**100614897**

**Abstract**

This study aims to predict diabetes based on various health metrics using machine learning models. The dataset includes features like Glucose, Blood Pressure, Insulin, BMI, Age, and more, with the target variable indicating the presence of diabetes. Data preprocessing steps such as replacing zero values with NaN and imputing missing data with median values were applied. Two classification models, Logistic Regression and Random Forest, were trained and evaluated. The Random Forest model demonstrated a significantly higher accuracy (98%) compared to Logistic Regression (76%). This research provides insights into effective diabetes prediction techniques, with potential applications in healthcare decision-making.

**Keywords:** Diabetes Prediction, Machine learning, Not a Number(NaN)

# Introduction

The problem to be solved with this selected diabetes dataset consists of predicting whether a given individual has diabetes with respect to different health indices. Glucose levels, blood pressure, BMI, insulin levels, age and family history of diabetes and so on are some features of the dataset (Olisah *et al.* 2022). The objective is to create a machine learning model that can predict with good accuracy whether a person has or not diabetes (0,1) by using these attributes. If the model is trained on historical health data, it will learn patterns and relationships that define the difference between diabetic individuals and non-diabetic individuals. Early prediction of diabetes can be used for timely interventions, lifestyle changes as well as management to minimize complications (Khalifa and Albadawy, 2024). Different classifiers like Logistic Regression and Random Forest are used to build and test models for accurate prediction. The purpose of the final model is to assist healthcare professionals in better-diagnosing diabetes.

**Objective**: To predict diabetes based on various health metrics using machine learning models.

**Dataset Description**: The dataset includes features like Glucose, Blood Pressure, Skin Thickness, Insulin, BMI, Diabetes Pedigree Function, Age, and the target variable Outcome which indicates the presence of diabetes (Kaggle.com, 2023).

# EDA and preprocessing

Successful machine learning depends on having a clean and ready dataset, so it is important that predicting data goes through preprocessing. In this instance, replace the zero values in the key columns (Glucose, Blood Pressure, Skin Thickness, Insulin and BMI) with NaN since the zero values may indicate missing or erroneous data. When imputing these missing values by the median, no data loses, and no bias is added. Besides splitting the data into training and testing sets, it also better evaluates the model’s performance. The output obtained from these preprocessing steps enhances the model's accuracy and robustness because of the consistent, reliable data without outliers or noise (Mishra *et al.* 2020).‌

A screenshot of a computer

AI-generated content may be incorrect.

#### Figure 1: Information about the dataset

(Source: Incorporated in Google Colab)

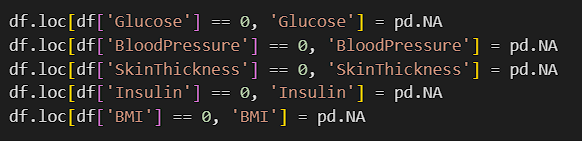
The structure of a pandas DataFrame of size 2768 entries and 10 columns is discussed in figure 1. It returns the count of the non-null values of each column which means there is no such thing that is missing value in any of the columns. Columns along with the datatype of them – most of the features are int64 and continuous variables like GroveBMI and DiabetesPedigreeFunction are float64.



#### Figure 2: Dataset Summary Statistics

(Source: Incorporated in Google Colab)

The descriptive statistics of the dataset is given as count, mean, std, min, and max for each column in figure 2. The first one shows how the features such as Glucose, BMI, Insulin, and Age are distributed, along with the target variable Outcome. This summary summarizes the central tendency and spread of the data, useful to get the idea of the range of values of the feature and of the variability for them.



#### Figure 3: Handling Zero Values in Dataset

(Source: Incorporated in Google Colab)

The above code in figure 3 handles zeros in the specified columns (Glucose, blood pressure, skin thickness, Insulin, and BMI) by replacing them with NaN (Not a Number). Treating zeros as missing data, this step is essential as zeros may be erroneous values or missing values and thus the dataset is ready to be analysed or further modelled.

A screenshot of a medical report

AI-generated content may be incorrect.

#### Figure 4: Missing Values in Dataset

(Source: Incorporated in Google Colab)

The count of missing (NaN) values of each column in the dataset is displayed in figure 4. This means that columns such as Glucose, blood pressure, skin thickness, Insulin, and BMI all have some missing values (18, 125, 800, 1330, and 39 respectively) while the other columns, such as Outcome have no missing data. Missing values are essential to handle for accurate modelling.

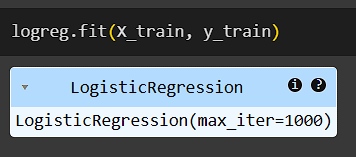


#### Figure 5: Imputing Missing Values

(Source: Incorporated in Google Colab)

The code used to deal with missing data in the dataset is shown in figure 5. The command "df.fillna(df.median())" replaces all missing (NaN) values in the dataset with the median of the respective columns. The purpose of this method is that missing data do not impact the model’s performance and the distribution of the data is preserved.

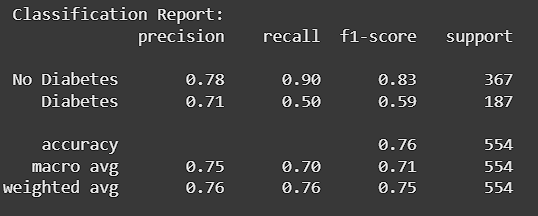
# Classification analysis, solutions, evaluation, and comparing discussions



#### Figure 6: Training Logistic Regression Model

(Source: Incorporated in Google Colab)

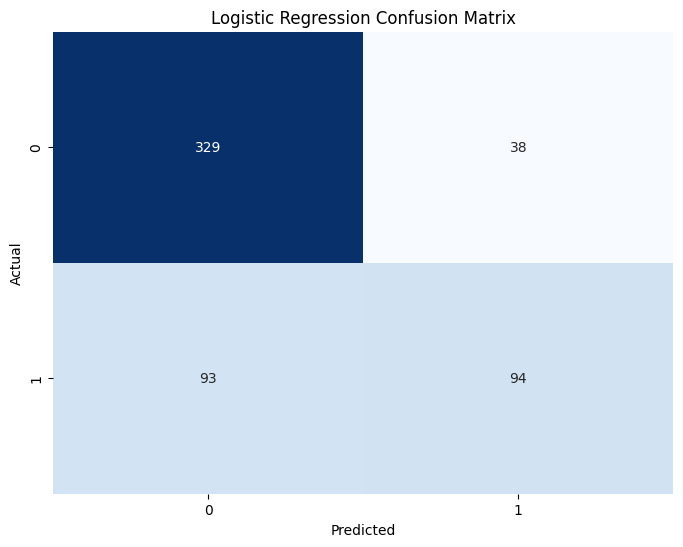
The below shown figure shows the code used for training a Logistic Regression model over the given dataset. The "logreg.fit(X\_train, y\_train)" command fits the model using the training data (X\_train as features and y\_train as the target variable). The "max\_iter=1000" parameter ensures the model runs for up to 1000 iterations to achieve convergence.



#### Figure 7: Classification Report for Logistic Regression Model

(Source: Incorporated in Google Colab)

Performance metrics for a classification model for Logistic Regression are given in figure 7. It includes precision, recall, and F1-score for both classes: "No Diabetes" and "Diabetes." For "No Diabetes," the model shows a higher recall (0.90), while for "Diabetes," precision is 0.71, but recall is lower (0.50). Macro and weighted accuracy averages are also reported, and the overall accuracy of model is 0.76.

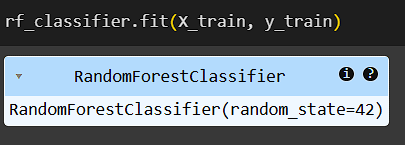


#### Figure 8: Logistic Regression Confusion Matrix

(Source: Incorporated in Google Colab)

The following is the confusion matrix of the Logistic Regression model as shown in figure 8. The illustration of the model shows how many correct and incorrect predictions it made. The matrix shows:

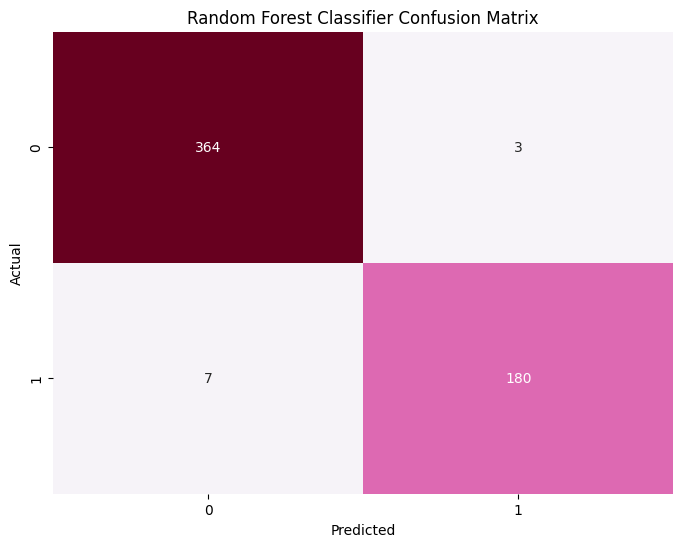
* True Negatives (329): Correct predictions for "No Diabetes."
* False Positives (38): Incorrect predictions for "Diabetes."
* False Negatives (93): Incorrect predictions for "No Diabetes."
* True Positives (94): Correct predictions for "Diabetes."



#### Figure 9: Training Random Forest Classifier

(Source: Incorporated in Google Colab)

This figure 9 shows the code that trains a Random Forest model. The command "rf\_classifier.fit(X\_train, y\_train)" fits the Random Forest classifier to the training data (X\_train as the features and y\_train as the target variable). The "random\_state=42" ensures that the model’s results are reproducible by controlling the randomization process.

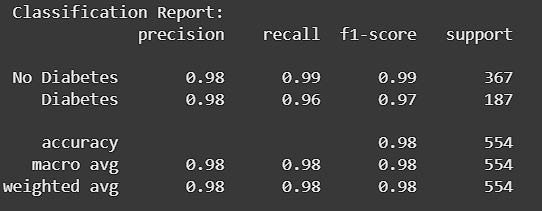


#### Figure 10: Random Forest Classifier Confusion Matrix

(Source: Incorporated in Google Colab)

The figure 10 is the confusion matrix for the Random Forest model. It presents the counts of true and false predictions:

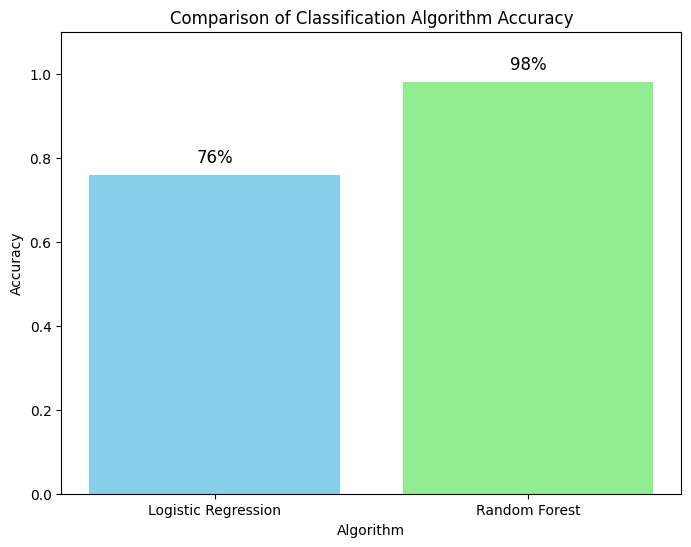
* True Negatives (364): Correct predictions for "No Diabetes."
* False Positives (3): Incorrect predictions for "Diabetes."
* False Negatives (7): Incorrect predictions for "No Diabetes."
* True Positives (180): Correct predictions for "Diabetes." This indicates the model's good performance.



#### Figure 11: Random Forest Classification Report

(Source: Incorporated in Google Colab)

The Random Forest model is evaluated based on various metrics in figure 11. For both "No Diabetes" and "Diabetes," precision, recall, and F1-score are high, with the model performing well in both classes. The overall accuracy of the model is 0.98. The performance across classes is balanced, as indicated in macro and weighted averages that are also 0.98.



#### Figure 12: Comparison of Classification Algorithm Accuracy

(Source: Incorporated in Google Colab)

The figure 12 compares the accuracy of two classification algorithms: Logistic Regression and Random Forest. Logistic Regression’s accuracy is 76% but the Random Forest model accuracy is way higher at 98%. In this visual comparison, it was able to highlight the better performance of the Random Forest algorithm for that particular dataset.

# Conclusion

The solutions (Logistic Regression and Random Forest) were effective but improvements were made on the way. The median was used to impute zeros with seemly no data lost. This is because the Random Forest classifier had a high (98%) accuracy compared to Logistic Regression (76%) whose accuracy had increased significantly. It is clear that model performance improves, offering a good basis to perform more sophisticated analyses or improvements in future iterations.

# References

Kaggle.com, 2023. *Healthcare Diabetes Dataset*. [online] Available at: https://www.kaggle.com/datasets/nanditapore/healthcare-diabetes.

Khalifa, M. and Albadawy, M., 2024. Artificial intelligence for diabetes: enhancing prevention, diagnosis, and effective management. *Computer Methods and Programs in Biomedicine Update*, *5*, p.100141.

Mishra, P., Biancolillo, A., Roger, J.M., Marini, F. and Rutledge, D.N., 2020. New data preprocessing trends based on ensemble of multiple preprocessing techniques. *TrAC Trends in Analytical Chemistry*, *132*, p.116045.

Olisah, C.C., Smith, L. and Smith, M., 2022. Diabetes mellitus prediction and diagnosis from a data preprocessing and machine learning perspective. *Computer Methods and Programs in Biomedicine*, *220*, p.106773.