Healthcare Diabetes Prediction

Student ID: 100661485

School of Computing and Engineering

Data Mining and AI

# Introduction

Diabetes is a chronic metabolic disorder that has become a global health concern, affecting millions of people worldwide. Early detection and management of diabetes are critical to preventing severe complications. Machine learning offers a prospective approach to predicting diabetes by analysing patient health metrics such as glucose levels, blood pressure, and body mass index (BMI).

This study focuses on predicting diabetes using a publicly available dataset from Kaggle [3], which contains health-related features for 2,768 patients. The dataset includes variables such as the number of pregnancies, glucose concentration, blood pressure, skin thickness, insulin levels, BMI, diabetes pedigree function, and age. The target variable, Outcome, indicates whether a patient has diabetes (1) or not (0). The dataset is imbalanced, with more non-diabetic patients than diabetic ones, which poses a challenge for model training.

The primary objective of this study is to apply machine learning techniques to predict diabetes accurately. To achieve this, we perform extensive data pre-processing, including handling missing values, noisy data, and class imbalance, followed by exploratory data analysis (EDA) to understand the dataset's characteristics. We then implement and evaluate several classification models, including Decision Tree, Support Vector Machine (SVM), Random Forest, Naive Bayes, and k-Nearest Neighbors (KNN), to identify the most effective approach for diabetes prediction.

# Data Pre-processing and Exploratory Data Analysis (EDA)

Data pre-processing and exploratory data analysis are essential steps in data mining to ensure data quality and gain insights into patterns. This section outlines how the dataset was cleaned and analysed before applying classification models.

## Dataset overview

Dataset contains 2768 rows and 10 columns.

The columns are:

#### Id: Unique identifier for each patient.

#### Pregnancies: Number of times pregnant.

#### Glucose: Plasma glucose concentration.

#### BloodPressure: Diastolic blood pressure (mm Hg).

#### SkinThickness: Triceps skinfold thickness (mm).

#### Insulin: 2-Hour serum insulin (mu U/ml).

#### BMI: Body mass index (weight in kg/(height in m)^2).

#### DiabetesPedigreeFunction: Diabetes pedigree function (a genetic risk score).

#### Age: Age in years.

#### Outcome: Target variable (0 = no diabetes, 1 = diabetes).

Dataset contains both numerical (int64, float64) and categorical (int64 for Outcome) features.

The target variable (Outcome) is imbalanced, with more patients not having diabetes (Outcome = 0) than those who do (Outcome = 1). It can affect model performance. Techniques like SMOTE can be used to address it.

sns.countplot(x=df['Outcome'], palette="Greens")

plt.title('Outcome Distribution')

plt.show()

A green and white bar graph

Description automatically generated

Figure 1 Outcome Results

Using describe function we got summary statistics for the numerical features.

print(df.describe())

A screenshot of a computer

Description automatically generated

We can see that there are noisy values in our dataset, that should be removed during pre-processing.

## Pre-processing

Before implementing machine learning models, we apply several pre-processing steps to handle inconsistencies, missing values, and noise. Proper pre-processing ensures the accuracy and efficiency of predictive models.

#### Handling missing data

Missing values can affect analysis and prediction. We examined the dataset for null values and no missing values were detected.

print(df.isnull().sum())

A screen shot of a computer code

Description automatically generated

#### Handling duplicates

The dataset contained no duplicates.

print(df.duplicated().sum())

Out: 0

#### Handling noisy data

Noisy data can lead to misleading patterns. Some features (e.g Glucose, BloodPressure, BMI) have values outside the expected physiological range. For example, Glucose should not be 0 (impossible for living person), BloodPressure should not be 0, and BMI should not be 0. We replaced all out-of-range values with mean of the column.

ranges = {

'Glucose': (70, 199),

'BloodPressure': (60, 122),

'SkinThickness': (10, 50),

'BMI': (15, 60),

'Insulin': (0, 846)

}

for column, (min\_val, max\_val) in ranges.items():

column\_mean = df[column].mean()

df[column] = df[column].apply(lambda x: column\_mean if x < min\_val or x > max\_val else x)

#### Feature scaling

Some machine learning models perform better when features are on a similar scale and require feature scaling. We standardised numerical attributes using StandardScaler.

from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()

X\_scaled = scaler.fit\_transform(df.drop('Outcome', axis=1))

#### Handling imbalanced data

The target variable (Outcome) is imbalanced, with more non-diabetic patients than diabetic ones.

We used SMOTE (Synthetic Minority Oversampling Technique) to balance the dataset.

from imblearn.over\_sampling import SMOTE

smote = SMOTE(random\_state=42)

X\_balanced, y\_balanced = smote.fit\_resample(X\_scaled, df['Outcome'])

#### Feature selection

We removed the first ID column as it doesn’t have any impact on the dataset.

#### Outlier removing

We removed outliers to ensure the model isn't skewed by extreme values, which can negatively impact performance and lead to inaccurate predictions. The boxplot (Figure 2) helped us visualise the distribution of each feature and identify outliers, allowing us to clean the data effectively.

num\_col = ['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness' ,'Insulin', 'BMI', 'DiabetesPedigreeFunction', 'Age', 'Outcome']

no\_outlier = df.copy()

for i in num\_col:

lower\_limit = df[i].quantile(0.5)

upper\_limit = df[i].quantile(0.95)

no\_outlier[i] = no\_outlier[i].clip(lower\_limit, upper\_limit)

df.plot(kind = "box" , subplots = True , figsize = (15,15) , layout = (5,5))

A comparison of a bar graph

Description automatically generatedFigure 2 Distribution of Diabetes Pedigree Function and Age

## Data Visualisation

The Outcome variable was imbalanced (Figure1), with more non-diabetic cases than diabetic ones. A bar plot was used to visualise this imbalance, indicating the necessity of SMOTE for balancing.

A comparison of a graph

Description automatically generated

Histograms and boxplots were generated to understand each numerical feature's distributions. Features like Glucose and BMI showed right-skewed distributions, while others, such as Blood Pressure, exhibited missing or incorrect values that needed correction.

A green and white bar chart

Description automatically generated

Figure 3 Heatmap

A heatmap was plotted to identify relationships between features. Glucose and BMI show strong correlations with the Outcome variable, suggesting their importance in prediction.

A screenshot of a graph

Description automatically generated

Pairwise relationships between key features were examined using pairplots, helping to visualise how features differ between diabetic and non-diabetic patients.

These visualisations were crucial for understanding the dataset, identifying potential issues (e.g., outliers, class imbalance), and guiding feature selection and pre-processing steps. They ensured that the data was properly prepared for training machine learning models.

After pre-processing and EDA, the dataset is ready for modelling. The final dataset has no missing values, no out-of-range values, balanced classes, scaled features.

# **Machine Learning Model Implementation**

In this project, the prediction of whether a patient has diabetes is a classification problem. Classification is the process of categorising data into predefined classes or labels. Our target value is Outcome and its binary, there`re 0 represents the presence of diabetes and 0 indicates absence. Our goal is to build a model that can accurately classify patients into these two categories based on various health-related features such as glucose levels, blood pressure, BMI and age. By implementing and evaluating multiple classification models—such as Decision Trees, Support Vector Machines (SVM), Random Forests, Naive Bayes, and k-Nearest Neighbors (KNN)—we aim to identify the most effective approach for predicting diabetes.

## Model training

The dataset was split into different ratios (80/20 and 70/30). For models like Decision Tree and Random Forest, which are less sensitive to overfitting, we split data into two sets: 70% for training and 30% for testing to provide more training data. For models like SVM and KNN, which can be more sensitive to overfitting, we used 80% data for testing and 20% for testing to ensure a larger test set for better generalisation.

To predict diabetes we chose the following machine-learning models:

#### Decision Tree Classifier

Decision Trees are simple, interpretable, and effective for both classification and regression tasks. Decision Tree algorithm can capture non-linear relationships between features and the target variable (diabetes outcome).

To implement and optimise the Decision Tree model, we first pre-processed the data by removing outliers using quantile-based clipping, ensuring the model wasn't skewed by extreme values, and we used the DecisionTreeClassifier from scikit-learn. It allowed us to get the higher results (100% training accuracy and 99.04% test accuracy).

A screenshot of a graph

Description automatically generated

Figure 4 Classification report and confusion matrix for Decision Tree

The confusion matrix confirms Decision Tree model effectively predicts both diabetes and non-diabetes cases, with very few misclassifications (e.g., only 6 false positives and 2 false negatives), demonstrating the model's strong performance.

#### Support Vector Machine (SVM)

 SVMs are powerful for classification tasks, especially when the data is not linearly separable. They work well in high-dimensional spaces and are effective in cases where the number of dimensions exceeds the number of samples.

To implement and optimise the Support Vector Machine (SVM) model, we scaled the features using StandardScaler to improve SVM performance, especially with the RBF kernel. Next, we performed hyperparameter tuning using GridSearchCV to find the best values for C and gamma, achieving a test accuracy of 99.04%.

A screenshot of a white sheet

Description automatically generated

Figure 5 Classification report and confusion matrix for SVM

The confusion matrix shows that SVM model performs well in distinguishing between diabetes and non-diabetes cases, with minimal misclassifications (e.g., 0 false positives and 8 false negatives), demonstrating the model's high accuracy and reliability.

#### Random Forest Classifier

Random Forest was chosen for its robustness and ability to reduce overfitting by combining multiple decision trees. The Random Forest model was implemented using the RandomForestClassifier. We implemented Hyperparameter tuning using RandomizedSearchCV to optimise parameters like n\_estimators, max\_depth, and min\_samples\_split, and improved test accuracy to 99.46%.

A screenshot of a graph

Description automatically generated

Figure 6 Classification report and confusion matrix for Random Forest

The confusion has only 3 false positives and 0 false negatives, highlighting the model's exceptional accuracy and reliability.

#### K-Nearest Neighbors (KNN)

I used KNN because it’s a non-parametric method that adapts to local patterns in the data. To further improve KNN model, I tuned the hyperparameters, such as the number of neighbours (n\_neighbors), and distance metric, and used weighted KNN, where closer neighbours have more influence on the prediction. For n\_neighbors=1 the model is overfitting because the training accuracy is 100%, but the test accuracy is slightly lower. For n\_neighbors=2 i got good precision and recall for class 0, but recall for class 1 drops to 87%. The weighted KNN with n\_neighbors=2 is the best choice because it achieves high accuracy on both the training and test sets, avoids overfitting, and performs well across all evaluation metrics.

A screenshot of a computer screen

Description automatically generated

Figure 7 Classification report and confusion matrix for KNN

The confusion matrix has few misclassifications (e.g., 0 false positives and 2 false negatives), confirming the model's strong predictive performance.

#### Naïve Bayes

Naive Bayes was selected as a baseline model due to its simplicity and speed. While it assumes feature independence, it’s still useful for diabetes prediction, especially when handling smaller datasets or as a comparison to more complex models. I experimented with different variants of Naive Bayes (Gaussian, Multinomial, and Bernoulli) and applied feature scaling using MinMaxScaler to improve performance. Additionally, I addressed class imbalance using SMOTE (Synthetic Minority Oversampling Technique) and performed hyperparameter tuning for var\_smoothing to enhance the model's accuracy. Despite these efforts, the model achieved a test accuracy of 79,6 %, indicating limitations due to the independence assumption and feature correlations.

A screenshot of a graph

Description automatically generated

Figure 8 Classification report and confusion matrix for Naive Bayes

## Evaluation

We evaluated models using accuracy, precision, recall, and F1-score, which are standard metrics for classification tasks. The best-performing model was the KNN with weighted distance and n\_neighbors=2, achieving a test accuracy of 99.64%. The Random Forest and Decision Tree models also performed exceptionally well, with test accuracies of 99.46% and 99.04%, respectively. The SVM model, after hyperparameter tuning, achieved a test accuracy of 99.04%. The Naive Bayes model, despite various attempts to improve its performance, had the lowest accuracy of 79.60%. The results suggest that ensemble methods like Random Forest and KNN are highly effective for this dataset, while Naive Bayes may not be suitable due to its assumptions about feature independence.

A graph of green and white bars

Description automatically generated with medium confidence

# Conclusion

This study aimed to predict diabetes using machine learning models applied to a healthcare dataset. After thorough data pre-processing, including handling missing values, noisy data, and class imbalance, several classification models were implemented and evaluated. The results demonstrate that ensemble methods, such as Random Forest and KNN, are highly effective for diabetes prediction, achieving test accuracies of 99.46% and 99.64%, respectively. The Decision Tree and SVM models also performed well, with accuracies of 99.04%, while Naive Bayes lagged behind with an accuracy of 79.60%, likely due to its assumption of feature independence.

The success of KNN and Random Forest highlights the importance of proper data pre-processing, feature scaling, and hyperparameter tuning in achieving high model performance. The study also underscores the significance of addressing class imbalance, as the dataset was heavily skewed toward non-diabetic patients. Techniques like SMOTE proved effective in balancing the dataset and improving model performance.

##### References

1. Chidiebere Philip Arinze (2024). *Effective Strategies for Handling Noisy Data in Machine Learning*. [online] Medium. Available at: https://medium.com/@InsightCoder/effective-strategies-for-handling-noisy-data-in-machine-learning-79f02f216b63.
2. J Jain, D. (2019). Data Preprocessing in Data Mining. [online] GeeksforGeeks. Available at: <https://www.geeksforgeeks.org/data-preprocessing-in-data-mining/>.
3. GeeksforGeeks (2021). *Data Pre-Processing wit Sklearn using Standard and Minmax scaler*. [online] GeeksforGeeks. Available at: https://www.geeksforgeeks.org/data-pre-processing-wit-sklearn-using-standard-and-minmax-scaler/.
4. Pore, N. (2023) Healthcare Diabetes Dataset [Dataset]. Available at: https://www.kaggle.com/datasets/nanditapore/healthcare-diabetes/data (Accessed: [20/03/2025]).
5. Bonaccorso, G. (2018). Machine learning algorithms : popular algorithms for data science and machine learning. Birmingham Packt August.
6. Chawla, Nitesh V., et al. "SMOTE: synthetic minority over-sampling technique." Journal of artificial intelligence research 16 (2002): 321-357.
7. SciKit Learn (2019). *sklearn.model\_selection.GridSearchCV — scikit-learn 0.22 Documentation*. [online] Scikit-learn.org. Available at: https://scikit-learn.org/stable/modules/generated/sklearn.model\_selection.GridSearchCV.html.