

***University of Ibadan Journal of
Science and Logics in ICT
Research (UIJSLICTR)***
ISSN: 2714-3627

A Journal of the Faculty of Computing, University of Ibadan, Ibadan, Nigeria

Volume 17 No. 1, June 2026

***journals.ui.edu.ng/uijslictr
<http://uijslictr.org.ng/>
uijslictr@gmail.com***



Development of Machine Learning Based System for Prediction of Diabetes

¹✉ Agbolade Sunday. J.

Redeemer's University, Ede, Osun-state.
agbolades@run.edu.ng,

Abstract

The early prediction of diabetes mellitus is of great importance in controlling the progression of the disease and in managing its complications. Since traditional diagnostic methods are unable to predict the disease in its early stages, the present study suggests a diabetes prediction system based on machine learning in order to improve the rate at which diabetes can be predicted. In order to achieve this aim, the PIMA Indians Diabetes Dataset was used to create the prediction model. The data obtained from the dataset were preprocessed and put in order for analysis. Recursive feature elimination was carried out in order to select the most relevant features for the prediction model. A hybrid model based on the Voting Classifier was designed by using Random Forest, AdaBoost, and XGBoost classifiers. The performance of the proposed model was assessed by means of the 5-fold cross-validation technique. Statistical parameters such as accuracy, precision, recall, and AUC-ROC curves were calculated in order to determine the performance metrics of the model. The model suggested had an accuracy of 88 per cent and a recall of 87 per cent. Moreover, an optimal threshold value of 0.52 was found for the model in order to decrease prediction errors, such as false positives and false negatives. Lastly, a user-friendly web application was created using the Streamlit library in order to predict whether or not a person has diabetes. The present study has developed an efficient machine learning-based model which is able to accurately predict the onset of diabetes. The model proposed can be altered and put into use in a clinical setting to help healthcare professionals in the effective management of the disease.

Keywords: *Diabetes Prediction, Machine Learning, Ensemble Learning, Voting Classifier, Random Forest, AdaBoost, XGBoost, Recursive Feature Elimination, Streamlit, Early Disease Detection.*

1. Introduction

Diabetes mellitus (DM) is a chronic condition which impairs the body's ability to turn food into usable energy, especially glucose, as pointed out by Olorunfemi [1]. It is mainly marked by high blood sugar levels, a state called hyperglycemia, which result from either a lack of insulin production or from the body's inability to make effective use of insulin. The World Health Organization states that approximately 425 million people worldwide had diabetes in 2017 and that this figure is expected to increase to 629 million by 2045, the vast majority of cases being found in adults aged between 20 and 79.

Diabetes may be divided into three primary types. Type 1, which is also called insulin-dependent diabetes mellitus (IDDM), generally

appears in people under the age of 30 and happens when the immune system incorrectly destroys the cells in the pancreas that produce insulin. Type 2, or non-insulin-dependent diabetes mellitus (NIDDM), usually impacts adults and is due to a decreased sensitivity of the body to insulin, causing poor regulation of glucose. Gestational diabetes occurs during pregnancy in women who have never had the condition before, and if it is not properly managed it can present risks to both the mother and the baby. When diabetes is not well controlled, it can lead to serious complications such as damage to the eyes (retinopathy), the kidneys (nephropathy), the nerves (neuropathy), heart disease, and in some cases amputations of a limb. Because of this, screening for early detection is essential if effective treatment and control of the disease are to be achieved.

In recent years, machine learning (ML) has significantly advanced healthcare by supporting automated diagnosis and early prediction of

Agbolade Sunday. J. (2026). Development of Machine Learning Based System for Prediction of Diabetes. *University of Ibadan Journal of Science and Logics in ICT Research (UIJSLICTR)*, Vol. 17 No. 1, pp. 184 – 201

diseases like diabetes. One particularly promising development is the use of ensemble models, which combine multiple algorithms to improve predictive accuracy. Models such as Random Forest, AdaBoost, and XGBoost have demonstrated strong performance in classification tasks compared to other methods [11]. These ensemble techniques fall into two broad categories: sequential methods like AdaBoost, where each model learns from the errors of its predecessor to boost overall performance, and parallel methods like Random Forest, which build multiple independent models simultaneously to reduce variance and enhance reliability.

2. Related Works

Recent research has made significant progress in using machine learning tools to predict diabetes. Lukmanto *et al.* [2] explored an approach combining feature selection with a fuzzy support vector machine (FSVM) to enable early detection of diabetes mellitus. The researchers used the PIMA Indian Diabetes dataset and applied the F-Score method to identify the most informative features prior to classification. These selected features were then used to train an FSVM model, which employed fuzzy inference rules to distinguish between diabetic and non-diabetic patients. The method achieved an accuracy of 89.02%, demonstrating its effectiveness in classification based on a well-defined rule system. However, a limitation remains: the study did not compare FSVM performance with ensemble methods such as boosting or bagging, leaving room for further comparison.

In another development, Natarajan *et al.* [3] proposed AdaptDiab, an adaptive ensemble feature selection technique designed to improve diabetes prediction across various models. This method integrates both filter and wrapper strategies—such as ANOVA F-score, Fisher score, variance threshold, and point-biserial correlation—to determine the most effective set of features. When evaluated with classifiers including Multilayer Perceptron, Random Forest, Gradient Boosting, Logistic Regression, and Gaussian Naive Bayes, AdaptDiab consistently outperformed conventional feature selection approaches. Stability tests using the Jaccard index showed perfect consistency across iterations. Applied to the PIMA Indian Diabetes dataset, it enhanced model generalization and

reduced overfitting, highlighting its potential in supporting clinical decisions.

Meanwhile, Talari *et al.* [4] developed a hybrid framework for early identification of Type 2 diabetes and assessment of its severity. The approach incorporated K-fold cross-validation and addressed class imbalance using the Engineered Minority Oversampling Technique (referred to as "Destroyed"). The study evaluated ensemble strategies like voting, bagging, and boosting, using the UCI's PIMA Indian Diabetes (PID) dataset. A two-phase model combined SMOTE (Synthetic Minority Over-sampling Technique) with Sequential Minimal Optimization (SMO), resulting in a highly efficient system. This combination achieved 99.07% accuracy with a processing time of just 0.1 milliseconds, proving particularly effective for identifying insulin resistance and early-stage diabetes in real-world medical environments.

Rustam *et al.* [5] came up with a fresh ensemble feature engineering approach to sharpen diabetes mellitus detection. They tackled issues like small datasets and low accuracy by merging multiple datasets into a bigger one for testing. They paired Convolutional Neural Network (CNN) and Long Short-Term Memory (LSTM) models for feature extraction, then compared features from an Extra Tree Classifier, CNN, and LSTM. The CNN-LSTM combo with a Random Forest classifier nailed a 0.99 accuracy, outpacing existing methods and proving its robustness with k-fold cross-validation.

Oliullah *et al.* [6] explored predicting diabetes, especially early stages in females, with a stacked ensemble approach. They used top-notch algorithms like Random Forest with GridSearchCV, XGBoost, NGBoost, Bagging, LightGBM, and AdaBoost as base models. After careful data preprocessing, they hit a 92.91% accuracy, holding its own against other methods. They also used the Shapley additive explanation (SHAP) to break down the decision-making, giving researchers and doctors handy insights for diabetes prediction.

Nagpal *et al.* [7] unveiled a smart ensemble machine learning framework to catch Diabetes Mellitus early, a chronic condition that can wreak havoc with kidney, heart, and nerve damage if missed. They used Principal Component Analysis (PCA) to trim dimensions

and built a bagging decision tree model. Tested on a 520-instance dataset with features like Polyuria and sudden weight loss, it scored 93.26% accuracy, 96.9% precision, 94.1% recall, and 95.4% F-Score, outshining other methods for chronic disease prediction.

Ali *et. al.* [10] looked at how feature transformation and machine learning models affect early diabetes detection. They tried no transformation, normalization, and min-max scaling on twelve models, including traditional and ensemble types, using a 768-sample dataset with 8 features. Light Gradient Boosting Machine (LGBM) led with 82.91% accuracy using min-max scaling, proving ensemble models often edge out traditional ones with the right prep.

Kaliappan *et. al.* [9] investigated multiple machine learning models and feature selection strategies for diabetes prediction across different datasets. By combining filter and wrapper methods with explainable AI approaches such as LIME and SHAP, alongside models like Random Forest and XGBoost, they assessed the significance of key features—identifying age, polyuria, and family history as particularly influential. Integrating explainability with traditional selection techniques helped achieve precision and recall scores of 0.9, enhancing confidence in AI applications within medical decision-making.

Olorunfemi *et. al.* [1] proposed an enhanced ensemble approach to improve diabetes diagnosis, combining sequential and parallel learning methods with feature selection to address overfitting. Using the PIMA India Diabetes Dataset—split into 70% training and 30% testing—they applied preprocessing steps including forward and backward feature selection. The first phase combined Random Forest with J48 and CART, while the second introduced XGBoost and AdaBoostM1 within a voting framework, resulting in perfect performance across all metrics: accuracy, F1 score, MCC, precision, recall, AUC-ROC, and AUC-PR all reached 100%.

Kishor and Chakraborty [8] developed a predictive healthcare model aimed at early and reliable diabetes detection, addressing challenges like redundant features and class imbalance using the Fast Correlation-Based

Filter (FCBF) and SMOTE. Several classifiers - Logistic Regression, KNN, Naive Bayes, Random Forest, and SVM—were evaluated, with Random Forest outperforming others at 97.81% accuracy, 99.32% sensitivity, 98.86% specificity, and 99.35% AUC.

3 Research methodology

The methodology outlines the strategy, data collection methods, tools, and evaluation measures employed to tackle the research problem and meet the study's objectives. It details the experimental design, data sources, initial data processing steps, and performance assessment criteria for the proposed models. This approach ensures the reliability and validity of the results while maintaining consistent procedures throughout. The chosen strategy was designed to support the development of machine learning models for diabetes prediction and to aid in improving early diagnostic tools within healthcare.

3.1 System Architecture and Flowchart

Figure 3 illustrates the architecture of a well-designed machine learning pipeline aimed at early and accurate detection of diabetes. The process began with gathering a suitable dataset containing essential patient information, including demographic details, physical measurements, and medical history. After collection, the data went through preprocessing steps such as cleaning, formatting, encoding categorical variables, and normalization to ensure consistency and quality. Next, an appropriate feature selection technique was applied to eliminate irrelevant or redundant data, thereby reducing noise and improving signal clarity. This resulted in a refined dataset ready for modeling.

The dataset was automatically filtered to retain only the most significant features, which were then used to train an ensemble machine learning model. This ensemble combined several individual models—such as Random Forest, AdaBoost, and XGBoost—each bringing distinct strengths to the system. By leveraging the combined predictive power of these algorithms, the integrated model achieved enhanced accuracy in identifying diabetes cases. Alongside pipeline development, the model's performance was rigorously assessed using standard evaluation metrics like accuracy, precision, recall, F1-score, and AUC. These

measures provided a clear, quantitative assessment of the model's effectiveness in correctly predicting diabetes. Ultimately, the resulting model demonstrated strong performance, enabled by strategic feature selection and the use of ensemble learning techniques, both of which played a critical role in achieving reliable detection outcomes.

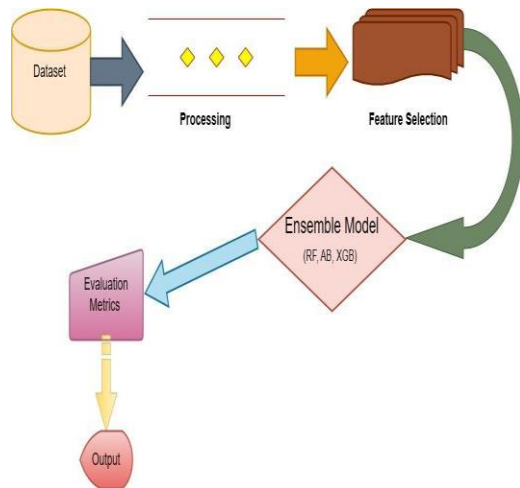


Figure 3.1: Diabetes Mellitus prediction Architecture

3.2 Data Collection and Description

This research uses the Pima Indian Diabetes dataset obtained from the UCI Machine Learning Repository, a widely recognized source of data for academic studies. The dataset contains 768 records, each corresponding to a female patient of Pima Indian heritage. It features eight numerical input variables and one binary target variable that specifies the presence (1) or absence (0) of diabetes. The Pima Indian population, residing in the Gila River Indian Community in southern Arizona, has been the focus of long-term diabetes research since 1965. This group is of particular interest due to its notably high incidence of Type 2 diabetes, with close to 50% of individuals under age 35 diagnosed—among the highest rates recorded worldwide.

The dataset was compiled in partnership with the National Institute of Diabetes and Digestive and Kidney Diseases (NIDDK), a U.S. agency dedicated to supporting research on diabetes and related conditions. It includes eight continuous predictor variables and one categorical outcome variable indicating diabetes diagnosis, as outlined in Table 1.

Table 1: Pima Indian Diabetic Patient Data

S/No	Attribute	Description	Type
1	Pregnant	Number of times the patient has been pregnant	Real
2	Plasma	Plasma glucose concentration from an oral glucose tolerance test	Real
3	Diastolic	Diastolic blood pressure (mmHg)	Real
4	Triceps	Triceps skinfold thickness (mm)	Real
5	Serum Insulin	2-hour serum insulin ($\mu\text{U}/\text{ml}$)	Real
6	BMI	Body Mass Index (kg/m^2)	Real
7	PedigreeFun	Diabetes pedigree function (genetic influence)	Real
8	Age (years)	Patient's age in years	Real
9	Diabetes	Diagnosis outcome (0 = non-diabetic, 1 = diabetic)	Discrete

3.3 Data Processing

Processing data is a crucial step in building any machine learning model, as real-world datasets typically need to be cleaned, organized, and transformed before they can be used for training and evaluation. This process helps ensure data quality and can greatly enhance model performance. Here, the key steps involved in preprocessing data for the Pima Indian Diabetes dataset, covering data collection, cleaning, transformation, train-test splitting, feature scaling, and the selection of relevant features are outlined.

3.3.1 Dataset Acquisition

The dataset was obtained from UCI Machine Learning Repository. The data format allows using the database in ML programs written in Python programming language since it is stored in CSV – Comma-Separated Values format.

3.3.2 Importing Required Libraries

To carry out preprocessing and model development, the following Python libraries were imported:

- **NumPy**: For mathematical operations and handling multidimensional arrays.
- **Pandas**: For data manipulation and analysis.
- **Matplotlib**: For visualizing data distributions and trends.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

3.3.3 Loading the Dataset

The dataset was loaded into a Pandas DataFrame using the `read_csv()` function:

```
data = pd.read_csv('Dataset.csv')
```

3.3.4 Feature and Target Separation

The dataset comprises eight independent variables (e.g., Glucose, Blood Pressure, Age) and one dependent variable (Diabetes status). The independent features were extracted as follows:

```
X = data.iloc[:, :-1].values # Independent features
y = data.iloc[:, -1].values # Dependent variable
```

3.3.5 Handling Missing Values

Missing values, which could lead to biased results or model errors, were handled using mean imputation. This method replaces missing entries with the mean of the respective feature column.

```
from sklearn.impute import SimpleImputer
imputer=SimpleImputer(missing_values=np.nan,
strategy='mean')
X[:, 1:3] = imputer.fit_transform(X[:, 1:3])
```

This technique is especially effective for numerical variables such as glucose levels, BMI, and age.

3.3.6 Splitting the Dataset

The dataset was divided into training and testing sets to evaluate model performance. 80% of the data was used for training, while 20% was reserved for testing.

3.3.7 Feature Scaling

Feature scaling was applied to standardize the range of independent variables, ensuring each feature had equal influence during model training. This step is crucial, as many machine learning algorithms are sensitive to variations in feature scales. Without normalization, features with larger magnitudes could disproportionately affect the learning process, potentially skewing results and compromising model accuracy.

3.3.8 Feature Selection

To improve model performance and reduce the likelihood of overfitting, feature selection was applied. This study utilized Recursive Feature Elimination (RFE) due to its proven ability to pinpoint the most valuable variables. By applying RFE, the analysis kept only the strongest predictors of diabetes mellitus, eliminating those that were redundant or contributed little information. As a result, the model became both more accurate and less complex, enhancing its efficiency and interpretability.

3.4 Description of Machine Learning Algorithms

To enhance the accuracy and reliability of diabetes mellitus prediction, this study utilized an ensemble learning strategy. Rather than relying on a single model, ensemble techniques merge several learning algorithms to create a more powerful predictive system. By leveraging

the advantages of diverse classifiers, this approach helps minimize errors, improve performance on new data, and reduce the risk of overfitting.

The study applied three well-established ensemble methods: Random Forest, AdaBoost, and Extreme Gradient Boosting (XGBoost). These were chosen due to their strong track record in classification problems and their capacity to manage intricate patterns commonly found in medical data.

3.4.1 Random Forest

Random Forest is an ensemble method that builds on the decision tree model by creating a group of trees instead of using just one. It trains each tree on a different subset of the training data, selected randomly with replacement—a process known as bagging, or Bootstrap Aggregating. To ensure variety among the trees, the algorithm also incorporates randomness when choosing features during tree construction.

In classification problems, predictions are made by combining the results from all trees through majority voting, where each tree casts a vote for a specific class label. By leveraging the collective decisions of multiple diverse trees, Random Forest enhances prediction accuracy, minimizes overfitting, and offers more consistent performance than a single decision tree. These advantages make it well-suited for applications such as medical diagnosis and predicting diseases.

3.4.2 AdaBoost (Adaptive Boosting)

AdaBoost is an ensemble technique that works sequentially by combining simple models, typically decision stumps—trees with just a single split. It places greater emphasis on instances that were incorrectly classified in earlier steps. With each iteration, the algorithm updates the data weights to prioritize those harder-to-classify examples.

Computation of AdaboostM1

Initialization: -

1. Using instance space training data $S = \{(x_1, y_1), \dots, (x_m, y_m)\}$ where $x_i \in \mathcal{X}$ and $y_i \in \mathcal{Y} = \{-1, +1\}$.
2. Initialize the distribution $D_1(i) = \frac{1}{m}$.

Algorithm:

For $t = 1, \dots, T$: do

Train a weak learner $h_t: \mathcal{X} \rightarrow \mathbb{R}$ using distribution D_t .

Determine – weight α_t of h_t .

Update the distribution over the training set:

$$D_{t+1}(i) = \frac{D_t(i)e^{-\alpha_t y_i h_t(x_i)}}{Z_t}$$

Where Z_t is a normalization factor chosen so that D_{t+1} will be a distribution.

End for

Final score:

$$f(x) = \sum_{t=1}^T \alpha_t h_t(x) \text{ and } H(x) = \text{sign}(f(x))$$

Advantages:

- i. Improves accuracy by focusing on hard-to-classify instances.
- ii. Less prone to overfitting than individual trees.
- iii. Performs well with minimal tuning.

Limitations:

- i. Sensitive to noisy data and outliers.
- ii. Slower to train compared to some simpler algorithms.

3.4.3 XGBoost (Extreme Gradient Boosting)

XGBoost is a sophisticated gradient boosting implementation that has been tuned for speed and efficiency. It constructs decision trees one after the other, fixing the mistakes of the earlier trees. XGBoost offers parallel processing and uses regularisation to avoid overfitting.

Computation of XG Boost (Deok, 2019)

Data: Dataset and hyperparameters

Initialize $f_0(x)$;

For $k = 1, 2, \dots, M$ do

$$\text{Calculate } g_k = \frac{\partial L(y, f)}{\partial f};$$

$$\text{Calculate } h_k = \frac{\partial^2 L(y, f)}{\partial f^2};$$

Determine the structure by choosing splits with maximized gain

$$A = \frac{1}{2} \left[\frac{G_L^2}{H_L} + \frac{G_R^2}{H_R} + \frac{G^2}{H} \right];$$

Determine the leaf weights $w^* = -\frac{G}{H}$;

Determine the base learner $\hat{b}(x) = \sum_{j=1}^T \omega_j$;

Add trees $f_k(x) = f_{k-1}(x) + \hat{b}(x)$;

End

Result: $f(x) = \sum_{k=0}^M f_k(x)$

Advantages:

- i. Highly efficient and scalable.
- ii. Offers regularization (L1 and L2) to reduce overfitting.
- iii. Often achieves state-of-the-art performance on structured/tabular data.

Limitations:

- i. Complex and requires careful parameter tuning.
- ii. Longer training time for very large datasets.

The evaluation of the model was done using a confusion matrix. It sums up the number of correct and incorrect predictions. It is a 2 X 2-dimensional matrix because we are dealing with binary classification. Table 3.2 shows the representation of the confusion matrix that was used for evaluation. The two classes are 0 and 1,

		Predicted Class		
		Yes	NO	Total
Actual Class	Yes	TP	FN	N
	No	FP	TN	
	Total	P	N	P+N

implying negative (no diabetes) and positive (diabetes) results, respectively. The diagonal values represent accurate predictions, while the non-diagonal values indicate inaccurate predictions.

Terminologies of confusion matrix as follows:

1. True Positives [TP]: These are the positive cases that the classifier properly classified.
2. True Negatives [TN]: These are the negative cases that the classifier properly classified.
3. False Positives [FP]: These are the negative cases that were wrongly classified as positive.
4. False Negatives [FN]: These are the positive cases that were wrongly classified as unfavorable.

The following measures were used to evaluate the model's performance:

- i. Accuracy: The rate of accuracy was calculated using the following formula, which is based on the confusion matrix:

$$Accuracy = \frac{TP+FP}{TP+TN+FP+FN} \quad (25)$$

- ii. Precision: Positive predictive values are what this is known as. This formula is used to calculate it:

$$Precision = \frac{TP}{TP+FP} \quad (26)$$

- iii. Recall: This is also known as sensitivity, and this formula is used to determine it:

$$Recall = \frac{TP}{TP+FN} \quad (27)$$

- iv. Matthews correlation coefficient (MCC) or phi coefficient

$$MCC = \frac{TP \times FP - FN \times TN}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}} \quad (28)$$

4.Result and Discussion

The first step in the implementation procedure was to load the libraries, as shown in Figure 1.

Figure 2 shows a breakdown of all the attributes under their corresponding categories. The dataset was loaded into a pandas data frame called diabetes, after which the head () function was used to show the first four records from the dataset. The shape of the data consists of 768 instances and 9 features.

Seven columns in the dataset are of type int64, and two are of type float64, as shown in Figure 3. The data includes several medical predictor variables—such as number of pregnancies, BMI, insulin levels, age, glucose concentration, skin thickness, and diabetes pedigree function - along with one target variable, Outcome.

```

import pandas as pd
import numpy as np
import time
import warnings
from contextlib import contextmanager

# --- Visualization Libraries ---
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline

import plotly.offline as py
import plotly.graph_objs as go
from plotly.offline import download_plotlyjs, init_notebook_mode, plot, iplot
import plotly.figure_factory as ff
import plotly.tools as tls
import squarify # Make sure to install this via `pip install squarify`

# --- Scikit-Learn for Data Processing, Modeling, and Evaluation ---
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.model_selection import (
    GridSearchCV, cross_val_score, train_test_split,
    RandomizedSearchCV, KFold, cross_val_predict
)
from sklearn.metrics import (
    precision_score, recall_score, confusion_matrix,
    roc_curve, precision_recall_curve, accuracy_score,
    roc_auc_score, auc
)
from sklearn.ensemble import VotingClassifier
from sklearn.neighbors import KNeighborsClassifier

# --- LightGBM ---
import lightgbm as lgbm # Ensure `lightgbm` is installed via `pip install lightgbm`

# --- Yellowbrick Visualizations ---
from yellowbrick.classifier import DiscriminationThreshold # Install via `pip install yellowbrick`

# --- Statistical Functions ---
import scipy.stats as ss
from numpy import interp # Correct replacement for deprecated `scipy.interp`
from scipy.stats import randint as sp_randint, uniform as sp_uniform

```

Figure 1: Loading the libraries

```

data = pd.read_csv('./diabetes.csv')

display(data.info(), data.head())

```

```

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 columns):
 #   Column                Non-Null Count  Dtype
---  -
 0   Pregnancies           768 non-null   int64
 1   Glucose               768 non-null   int64
 2   BloodPressure         768 non-null   int64
 3   SkinThickness         768 non-null   int64
 4   Insulin              768 non-null   int64
 5   BMI                  768 non-null   float64
 6   DiabetesPedigreeFunction 768 non-null   float64
 7   Age                  768 non-null   int64
 8   Outcome              768 non-null   int64
dtypes: float64(2), int64(7)
memory usage: 54.1 KB

None

```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	86	29	0	26.6	0.351	31	0
2	8	183	84	0	0	23.3	0.672	32	1
3	1	89	86	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1

Figure 2: Output showing the shape and the first four instances in the dataset.

Figure 4 reveals an imbalance in the distribution of the outcome classes: there are more instances of patients without diabetes (coded as 0) than with diabetes (coded as 1). Specifically, there are 268 non-diabetic cases compared to 500 diabetic cases.

4.2 Data Preprocessing

This section presents the tools and results obtained in the data preprocessing.

4.2.1 Missing Values

The data.head() output showed that certain features contain zero values, which are

unrealistic and likely indicate missing data. These zeros are replaced with NaN below. Missing values are then filled using the mean, median, or mode derived from other observations. The mode is most suitable for categorical variables, whereas the mean and median are preferred for numerical data. For imputation to be effective, the sample should be statistically homogeneous. Mean, median, and mode imputation can be applied in two ways: generally across the entire dataset or more specifically within defined subsets.

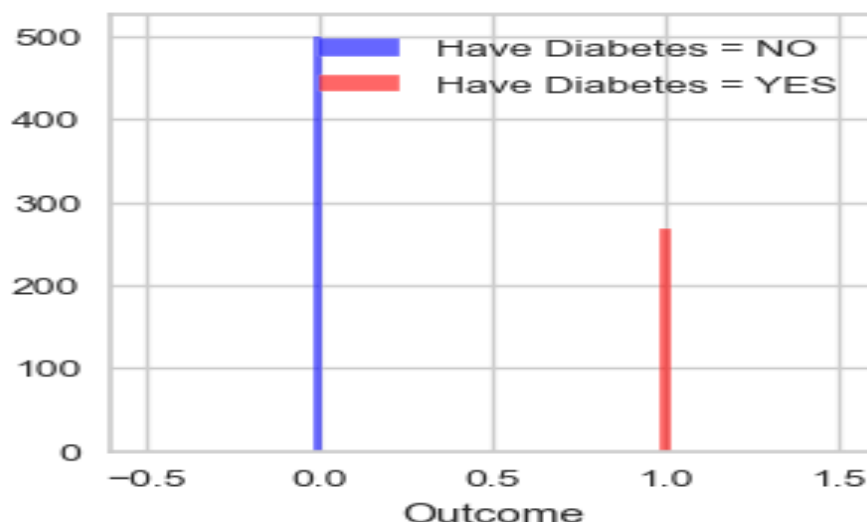


Figure 3: Data distribution

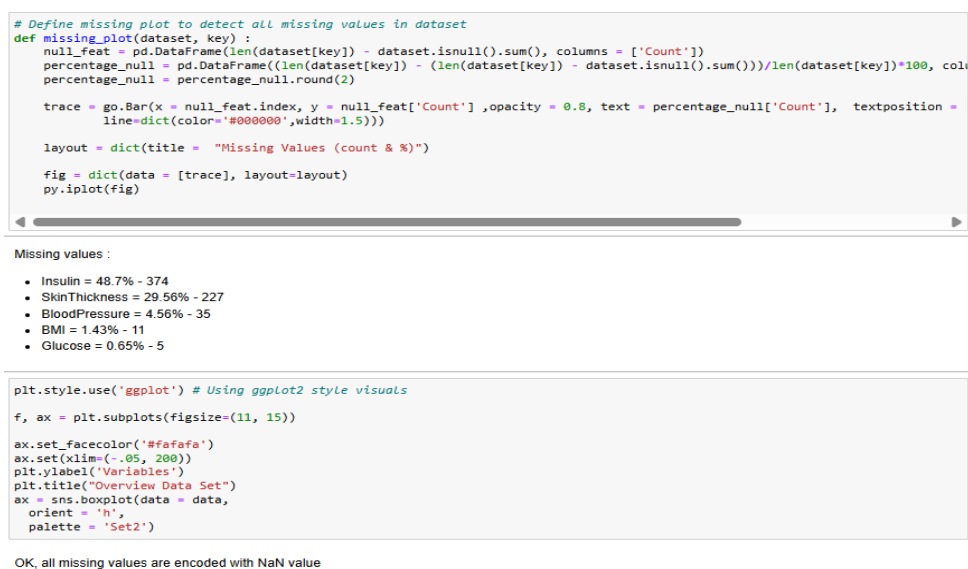


Figure 4: Detecting Missing Values

i. Generalized imputation

The missing value is substituted by the mean/median of all the non-missing values of that variable.

ii. Specialized imputation

The missing values are filled in using the average of the available data points for that particular variable, calculated from the observed cases. Initially, NaN entries were replaced with zeros to simplify counting, after which these zeros were substituted with suitable

estimated values. To determine appropriate replacements, the relationships between variables were analyzed.

In Figure 5, stronger correlations are represented by lighter shades. The table and heatmap indicate that age, BMI, number of pregnancies, and glucose levels have notable associations with the outcome. Also evident are pairwise relationships between features, such as between insulin and skin thickness, or age and number of pregnancies.



Figure 5: Correlation Heatmap of features

4.2.2 Feature Scaling

StandardScaler standardizes features by removing the mean and scaling to unit variance. This is achieved by computing the mean and standard deviation for each feature independently across the training samples. These calculated values are then stored and later applied when transforming new data using the transform method. A number of machine learning algorithms work best when input data is

standardized. Without this step, performance can suffer, especially if the features do not approximately follow a normal distribution with zero mean and unit variance. LabelEncoder converts class labels into integers ranging from 0 to n_classes minus one.

As illustrated in Figure 6, this encoding step ensures the data is in the appropriate format for input into our algorithm.

```
target_col = ["Outcome"]
cat_cols = data.nunique()[data.nunique() < 12].keys().tolist()
cat_cols = [x for x in cat_cols]
#numerical columns
num_cols = [x for x in data.columns if x not in cat_cols + target_col]
#Binary columns with 2 values
bin_cols = data.nunique()[data.nunique() == 2].keys().tolist()
#Columns more than 2 values
multi_cols = [i for i in cat_cols if i not in bin_cols]

#Label encoding Binary columns
le = LabelEncoder()
for i in bin_cols:
    data[i] = le.fit_transform(data[i])

#Duplicating columns for multi value columns
data = pd.get_dummies(data = data, columns = multi_cols)

#Scaling Numerical columns
std = StandardScaler()
scaled = std.fit_transform(data[num_cols])
scaled = pd.DataFrame(scaled, columns=num_cols)

#dropping original values merging scaled values for numerical columns
df_data_og = data.copy()
data = data.drop(columns = num_cols,axis = 1)
data = data.merge(scaled,left_index=True,right_index=True,how = "left")
```

Figure 6: Feature Scaling

We define X and y:

```
X = data.drop('Outcome', axis=1)
y = data['Outcome']
```

Figure 7: Creating Training Data

4.3 Model Training and Performance

The algorithm will be trained and evaluated using k-fold cross-validation. In this approach, the original dataset is randomly split into k equally sized subsets. For each iteration, k-1 of these subsets are used to train the model, while the remaining subset serves as the validation set. The process is repeated k times, with each of the k subsets used exactly once for validation. The final performance estimate is obtained by averaging the results from all k iterations. This method offers a key benefit over repeated random subsampling: every data point is included in both training and validation phases, and each is used for validation only once, leading to a more reliable and balanced assessment.

4.4 Model Performance Evaluation

To assess the effectiveness of the proposed ensemble model for early detection of diabetes, a VotingClassifier was trained and evaluated using various performance metrics. One of the key tools used in evaluating the model's behavior across decision thresholds is the Threshold Plot, which provides a holistic view of trade-offs between different evaluation metrics as the discrimination threshold changes.

4.4.1. Model Performance Evaluation and Comparative Analysis for Diabetes Mellitus Prediction

As illustrated in Figure 7, the ensemble model—combining Random Forest (RF), AdaBoost, and XGBoost classifiers—was assessed using 5-fold cross-validation. The confusion matrix shows strong performance in distinguishing both diabetic and non-diabetic patients, evident from the high values along the diagonal representing true positives and true negatives. This suggests the model effectively classifies cases in both categories.

Performance metrics indicate an accuracy of roughly 88%, precision near 86%, recall around 87%, and an F1-score of approximately 85%. These results reflect consistent and dependable performance, with particularly strong recall,

meaning the model is effective at identifying actual diabetes cases—an important factor in clinical screening where missing a case could delay treatment and worsen outcomes.

The Receiver Operating Characteristic (ROC) curve for the ensemble yields an AUC of 0.938, highlighting its strong ability to differentiate between classes. Additionally, the precision-recall curve confirms the model's robustness, showing it maintains a favorable balance between precision and recall across various decision thresholds. This characteristic is especially valuable in medical datasets, such as those involving diabetes, which often exhibit class imbalance.

Figure 8 displays the results obtained with the XGBoost classifier, which showed the best performance among all individual models tested. The confusion matrix reveals a high count of both true positives and true negatives, indicating that the model is effective at distinguishing between patients with and without diabetes. It reached an accuracy of about 89%, along with precision, recall, and F1-scores of 87%, 86%, and 86% respectively—marking the highest values across the evaluated models. The ROC curve for this classifier yielded an AUC of 0.94, the top result among all models, which underscores its strong discriminative ability. Furthermore, the precision-recall curve shows consistent performance, reinforcing its suitability and reliability for use in clinical prediction settings.

Figure 9 presents the performance evaluation of the AdaBoost classifier. While AdaBoost demonstrated strong results, its performance metrics were somewhat lower compared to XGBoost and the ensemble model. The classifier reached an accuracy of roughly 87%, with precision and recall values in the range of 84–85%, and an F1-score of 84%. Its AUC stood at 0.929—indicating good discriminative ability, though slightly weaker in distinguishing between classes. The precision-recall curve suggests the model maintains a decent balance between precision and recall, though performance varies moderately across different decision thresholds.

Random Forest (RF), although not depicted in the figures, was also evaluated using the same 5-fold cross-validation method. It achieved an

accuracy of about 87%, precision of 85%, recall of 84%, and an F1-score near 84%, along with an AUC of approximately 0.93. These results confirm Random Forest as a reliable baseline

with solid overall performance, yet it still falls short of the effectiveness shown by XGBoost and the ensemble model.

```
def model_performance(model, subtitle):
    #Fold
    cv = KFold(n_splits=5, shuffle=False, random_state=42)
    y_real = []
    y_proba = []
    tprs = []
    auc = []
    mean_fpr = np.linspace(0, 1, 100)
    i = 1

    for train, test in cv.split(X, y):
        model.fit(X.iloc[train], y.iloc[train])
        pred_proba = model.predict_proba(X.iloc[test])
        precision, recall = precision_recall_curve(y.iloc[test], pred_proba[:, 1])
        y_real.append(y.iloc[test])
        y_proba.append(pred_proba[:, 1])
        fpr, tpr, _ = roc_curve(y[test], pred_proba[:, 1])
        tprs.append(interp(mean_fpr, fpr, tpr))
        roc_auc = auc(fpr, tpr)
        auc.append(roc_auc)

    # Confusion matrix
    y_pred = cross_val_predict(model, X, y, cv=cv)
    conf_matrix = confusion_matrix(y, y_pred)
    trace1 = go.Heatmap(x=conf_matrix, y=conf_matrix, z=[0, 1], name="Confusion Matrix", colorscale="Viridis", showscale=False)

    #Show metrics
    tp = conf_matrix[1, 1]
    fn = conf_matrix[0, 1]
    tn = conf_matrix[0, 0]
    fp = conf_matrix[1, 0]
    Accuracy = ((tp+tn)/(tp+tn+fp+fn))
    Precision = (tp/(tp+fp))
    Recall = (tp/(tp+fn))
    F1_score = 2*((tp/(tp+fp))*(tp/(tp+fn)))/((tp/(tp+fp))+(tp/(tp+fn)))

    show_metrics = pd.DataFrame(data=[Accuracy, Precision, Recall, F1_score])
    show_metrics = show_metrics.T

    colors = ['gold', 'lightgreen', 'lightcoral', 'lightskyblue']
    trace2 = go.Bar(x=[Accuracy, Precision, Recall, F1_score], y=[Accuracy, Precision, Recall, F1_score], text=np.round(show_metrics[0].values, 4),
        textposition="auto", textfont=dict(color="black"),
        orientation="h", opacity=1, marker=dict(
            line=dict(color="black", width=1.5)))

    #ROC curve
    mean_tpr = np.mean(tprs, axis=0)
    mean_auc = auc(mean_fpr, mean_tpr)
    trace3 = go.Scatter(x=mean_fpr, y=mean_tpr,
        name="ROC curve", line=dict(color="black", width=2), fill="tozeroy")
    trace4 = go.Scatter(x=[0, 1], y=[0, 1],
        name="Diagonal", line=dict(color="black", width=1.5,
        dash=[8, 4]))

    #Precision - recall curve
    y_real = y
    y_proba = np.concatenate((y_proba, y_proba))
    precision, recall = precision_recall_curve(y_real, y_proba)
    traces = go.Scatter(x=recall, y=precision,
        name="Precision - Recall curve", line=dict(color="black", width=2), fill="tozeroy")

    mean_auc = round(mean_auc, 2)
    #Subplots
    fig = plt.figure(figsize=(12, 10), print_grid=False,
        subplot_titles=(
            "Confusion Matrix",
            "Metrics",
            "ROC curve",
            "Precision - Recall curve",
        ))

    #Trace and layout
```

Figure 8 Model Training and Performance Function

Fold	Accuracy	Precision	Recall	F1 Score	ROC AUC
1	0.87	0.815	0.815	0.815	0.927
2	0.844	0.8	0.741	0.769	0.933
3	0.87	0.84	0.778	0.808	0.935
4	0.908	0.882	0.849	0.865	0.949
5	0.902	0.88	0.83	0.854	0.95
mean	0.879	0.843	0.803	0.822	0.939
std	0.024	0.033	0.039	0.035	0.009

Figure 8a: Random Forest

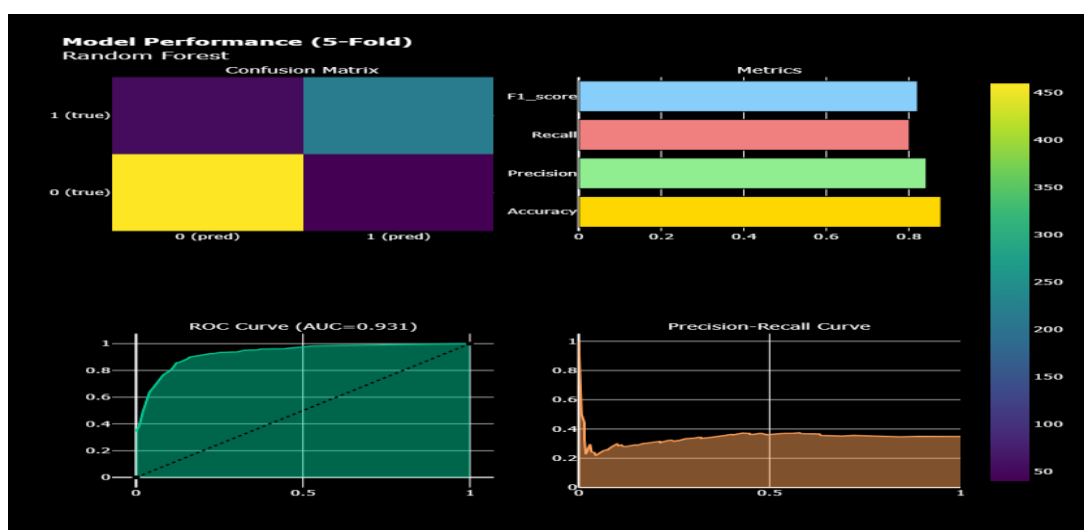


Figure 8b: Random Forest

Fold	Accuracy	Precision	Recall	F1 Score	ROC AUC
1	0.864	0.824	0.778	0.8	0.945
2	0.831	0.792	0.704	0.745	0.902
3	0.864	0.811	0.796	0.804	0.941
4	0.85	0.768	0.811	0.789	0.918
5	0.882	0.807	0.868	0.836	0.952
mean	0.858	0.8	0.791	0.795	0.932
std	0.017	0.019	0.053	0.029	0.019

Figure 9a: Adaboost

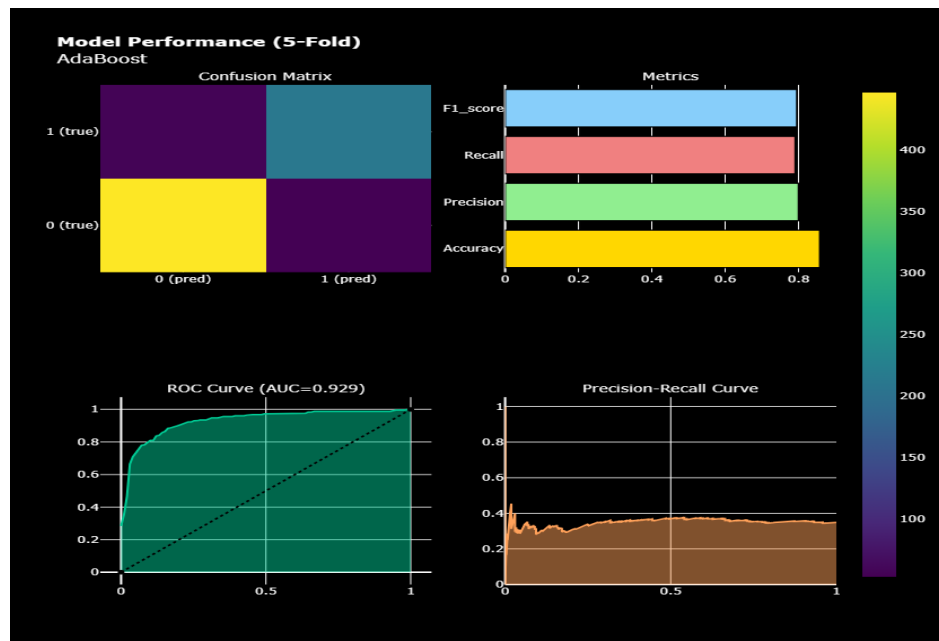


Figure 9b: Adaboost

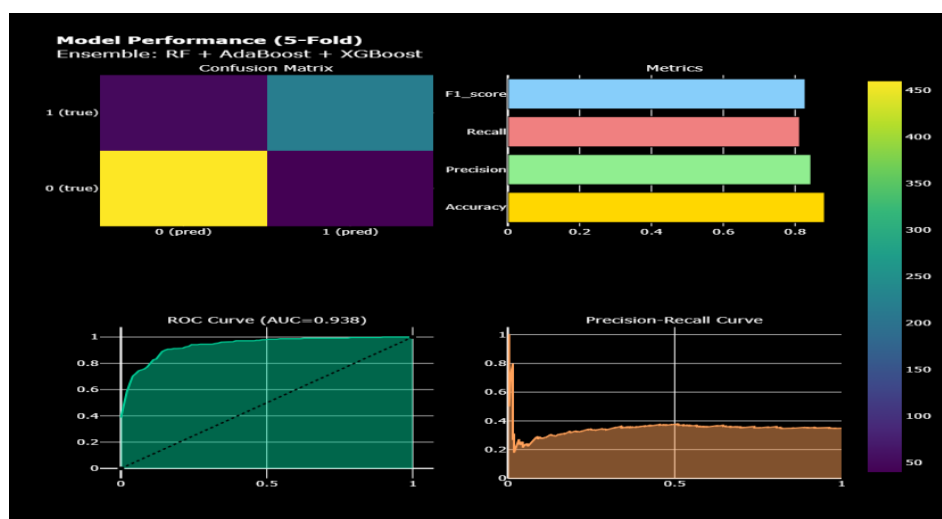


Figure 10a: Ensemble

Cross Validation Scores

Ensemble: RF + AdaBoost + XGBoost

Fold	Accuracy	Precision	Recall	F1 Score	ROC AUC
1	0.864	0.824	0.778	0.8	0.938
2	0.857	0.786	0.815	0.8	0.929
3	0.89	0.863	0.815	0.838	0.943
4	0.902	0.865	0.849	0.857	0.961
5	0.902	0.896	0.811	0.851	0.957
mean	0.883	0.847	0.814	0.829	0.946
std	0.019	0.038	0.023	0.025	0.012

Figure 10b: Ensemble

4.4.2 Threshold Analysis of VotingClassifier

The chart above shows the model's performance across various discrimination thresholds from 0 to 1, measured by precision, recall, F1 score, and queue rate.

- Precision (Red Line): At lower thresholds, precision is modest but improves as the threshold increases. This reflects a reduction in false positives when the model adopts a stricter criterion for assigning positive classifications, thus improving accuracy among predicted positives.
- Recall (Blue Line): Recall is high when the threshold is low, meaning the model captures more actual positive cases. However, it declines as the threshold rises, since a stricter threshold causes some true positives to be missed.
- F1 Score (Purple Line): As a harmonic mean of precision and recall, the F1 score reaches its peak near a threshold of 0.52, indicated by a vertical dashed line. This point represents the best trade-off between correctly identifying positive cases and minimizing false alarms, making it the recommended operating threshold.
- Queue Rate (Black Line): This measures the fraction of samples labeled as positive. It steadily drops as the threshold increases, since fewer instances surpass the higher confidence level needed for a positive classification.

4.4.3 Interpretation

The **optimal threshold ($t_x = 0.52$)** provides a balanced trade-off where the model achieves reasonable precision and recall. This is particularly important in the context of diabetes detection, where missing positive cases (false negatives) can have serious consequences, but too many false positives can lead to unnecessary follow-up tests and anxiety.

The threshold plot confirms that the VotingClassifier performs reliably across a spectrum of thresholds, and that a threshold around 0.52 is ideal for achieving a balanced performance. This threshold will be adopted in deployment to ensure that both early detection (recall) and accuracy of detection (precision) are optimized.

4.5 Interface for Diabetes Mellitus Prediction Using Ensemble Model

To improve access to and ease of use of the predictive model, a web-based interface was created using Streamlit—an open-source Python framework designed for developing interactive applications in machine learning and data science. This interface enables patients and healthcare providers to enter key clinical information and immediately obtain a prediction regarding diabetes status, generated by the trained ensemble model combining Random Forest, AdaBoost, and XGBoost. As illustrated in Figure 10, the application's landing page features a clear and user-friendly layout. It greets users and outlines the tool's objective: supporting early diabetes detection through machine learning. The page also includes a concise explanation of how the model was developed, basic usage instructions, and a prominent button that directs users to the prediction form. Figure 11 shows the prediction interface, where users are guided to input critical clinical variables such as:

- Glucose level
- Blood pressure
- Body mass index (BMI)
- Age
- Insulin levels
- Skin thickness
- Number of pregnancies
- Diabetes pedigree function

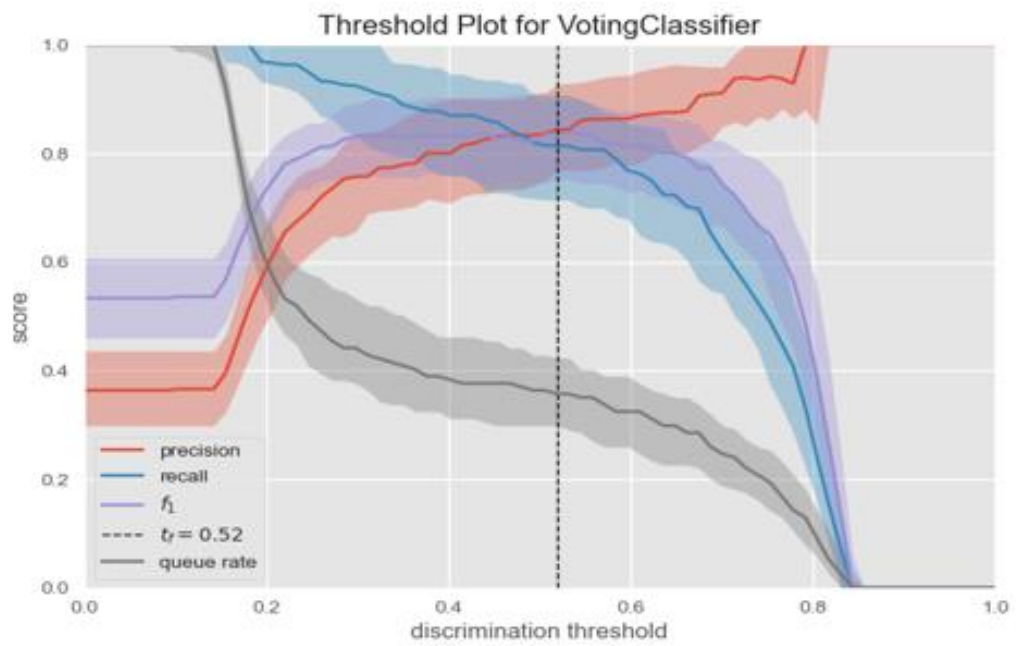


Figure 11: Threshold plot for voting classifier

The figure shows the landing page of a "Diabetes Mellitus Prediction System". The page has a dark background with white text. At the top, there is a logo and the title "Diabetes Mellitus Prediction System". Below the title, there is a paragraph: "Enter the following health details to assess the likelihood of having diabetes. The prediction is based on a machine learning model trained on the Pima Indians dataset." Below this paragraph, there is a form with several input fields, each with a label, a value, and a range. The fields are: "Pregnancies" (value: 1, range: - to +), "Glucose Level (mg/dL)" (value: 100, range: 0 to 200), "Blood Pressure (mm Hg)" (value: 70, range: 0 to 140), "Skin Thickness (mm)" (value: 20, range: 0 to 100), "Insulin Level" (value: 85, range: 0 to 900), "BMI" (value: 25.00, range: 0.00 to 70.00), "Diabetes Pedigree Function" (value: 0.50, range: 0.00 to 2.50), and "Age" (value: 30, range: 10 to 100). At the bottom left, there is a "Predict" button with a blue icon.

Figure 12: Landing page

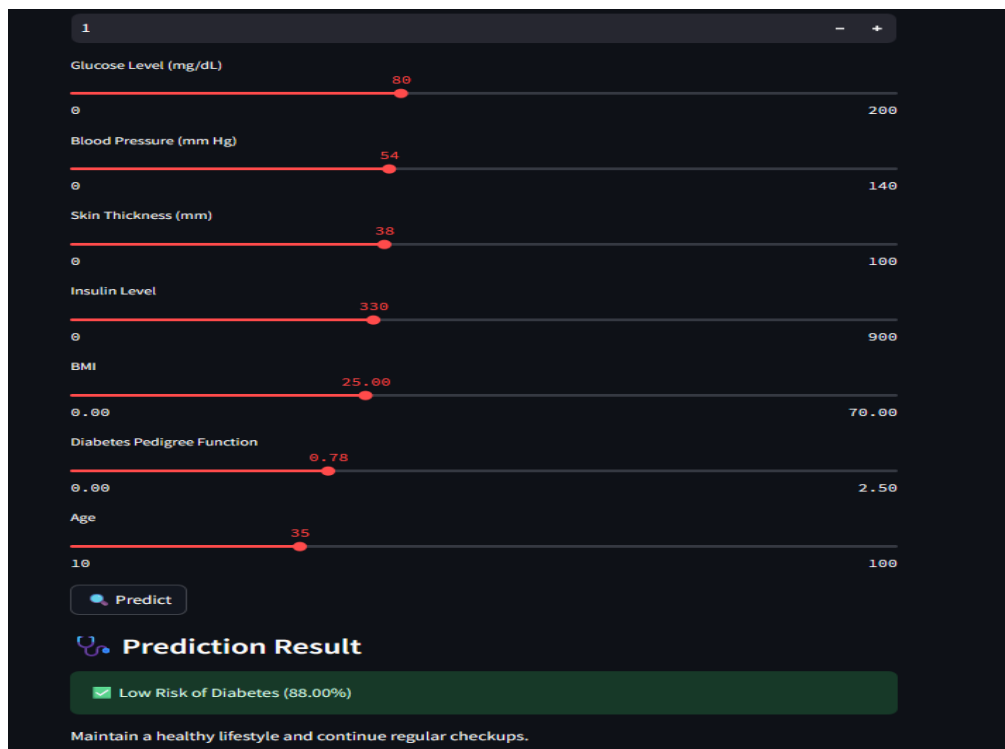


Figure 13: Result Page

After entering the required input values, the user can initiate the prediction by clicking the “Predict” button. This action activates the backend ensemble model, which analyzes the data and generates a classification indicating whether the individual is likely to be “Diabetic” or “Non-Diabetic.” As illustrated in Figure 12 once the model completes its analysis, the result is presented clearly on the interface. If diabetes is predicted, the system displays a warning along with advice to seek guidance from a healthcare provider. If the result suggests a non-diabetic status, the user receives a reassuring message, accompanied by a reminder to maintain regular medical checkups for ongoing health assessment.

The app includes a sidebar providing supplementary details on the dataset, the machine learning methods employed, and access to additional resources or documentation. This promotes transparency and helps educate users—key considerations in healthcare contexts. By embedding an ensemble-based predictive model within a Streamlit interface, the tool enables real-time diabetes risk evaluation. It effectively connects sophisticated algorithms with end-users, empowering individuals, especially those in areas with limited healthcare resources, to interact meaningfully with their health data. Merging predictive capabilities with an intuitive user interface, the system supports early detection efforts and enhances public awareness of diabetes mellitus.



Figure 14: Result page

5. Conclusion

This research highlights the development of a machine learning model aimed at the early identification of diabetes mellitus. The strategy combines feature selection with ensemble learning methods to enhance diagnostic precision. As diabetes ranks among the most widespread chronic illnesses worldwide, prompt diagnosis and effective management are essential.

The model was assessed using the Pima Indians Diabetes Dataset. The process comprised a series of data mining phases: preprocessing steps such as addressing missing data and normalizing features, followed by the identification of the most pertinent features through recursive feature elimination. Three ensemble techniques—Random Forest, AdaBoost, and XGBoost—were employed, and their predictions were aggregated using a VotingClassifier that integrated both parallel and sequential strategies.

The evaluation of the model's performance was conducted through precision-recall analysis, accuracy measurements, F1-score calculations, and threshold assessments. An optimal classification threshold of 0.52 was determined for the VotingClassifier, providing the best compromise between precision and recall. The

results suggest that the proposed model is effective in detecting cases of diabetes. Given that early detection can avert serious health issues, such as life-threatening complications, this method presents a valuable tool for enhancing the promptness and reliability of diabetes diagnosis.

References

- [1] Olorunfemi, B. O., Ogunde, A. O., Almogren, A., Adeniyi, A. E., Ajagbe, S. A., Bharany, S., ... & Hamam, H. (2025). Efficient diagnosis of diabetes mellitus using an improved ensemble method. *Scientific Reports*, 15, Article 3235. [https://doi.org/10.1038/s41598-025-10335-9]
- [2] Lukmanto, R. B., Suhajito, Nugroho, A., & Akbar, H. (2019). Early detection of diabetes mellitus using feature selection and fuzzy support vector machine. *Procedia Computer Science*, 157, 46–54. [https://doi.org/10.1016/j.procs.2019.08.140]
- [3] Natarajan, K., Baskaran, D., & Kamalanathan, S. (2025). An adaptive ensemble feature selection technique for model-agnostic diabetes prediction. *Scientific Reports*, 15(1), Article 6907. [https://doi.org/10.1038/s41598-025-13944-w]

- [4] Talari, P., Bharathiraja, N., Kaur, G., Alshahrani, H., Al Reshan, M. S., Sulaiman, A., & Shaikh, A. (2024). Hybrid feature selection and classification technique for early prediction and severity of diabetes type 2. *PLOS ONE*, 20(5), e0292100. [https://doi.org/10.1371/journal.pone.0292100]
- [5] Rustam, F., Al-Shamayleh, A. S., Shafique, R., Aparicio Obregon, S., Calderon Iglesias, R., Miramontes Gonzalez, J. P., & Ashraf, I. (2024). Enhanced detection of diabetes mellitus using novel ensemble feature engineering approach and machine learning model. *Scientific Reports*, 14, 23274. [https://doi.org/10.1038/s41598-024-23274-1]
- [6] Oliullah, K., Rasel, M. H., Islam, M. M., Islam, M. R., Wadud, M. A. H., & Whaiduzzaman, M. (2023). A stacked ensemble machine learning approach for the prediction of diabetes. *Journal of Diabetes & Metabolic Disorders*, 23, 603–617. [https://doi.org/10.1007/s40200-023-00896-x]
- [7] Nagpal, A., Sabharwal, M., & Tripathi, R. (2023). A novel ensemble machine learning framework for early-stage diabetes mellitus prediction. *Multidisciplinary Science Journal*, 6(3), e2024031. [https://doi.org/10.31893/multiscience.2024031]
- [8] Kishor, A., & Chakraborty, C. (2024). Early and accurate prediction of diabetics based on FCBF feature selection and SMOTE. *International Journal of System Assurance Engineering and Management*, 15, 4649–4657. [https://doi.org/10.1007/s13198-021-01374-6]
- [9] Kaliappan, J., Kumar, I. J. S., Sundaravelan, S., Anesh, T., Rithik, R. R., Singh, Y., ... & Srinivasan, K. (2024). Analyzing classification and feature selection strategies for diabetes prediction across diverse diabetes datasets. *Frontiers in Artificial Intelligence*, 7, 1421751. [https://doi.org/10.3389/frai.2024.1421751]
- [10] Ali Linkon, A., Noman, I. R., Islam, M. R., Bortty, J. C., Bishnu, K. K., & Islam, A. (2024). Evaluation of feature transformation and machine learning models on early detection of diabetes mellitus. *IEEE Access*, 12, 1–13. https://doi.org/10.1109/ACCESS.2024.1234567
- [11] Temidayo, K. M., Kuboye, B. M., & AYOKUNLE, A. (2024). Early Identification of Diabetes Mellitus using Parallel and Sequential Ensemble Methods. *International Journal of Computer Applications*, 186(56), 8-15.