

A PROJECT REPORT ON
AI-BASED DIABETES PREDICTION SYSTEM

Subject in partial fulfillment of the requirements for the degree of

BACHELOR OF ENGINEERING

In

**ELECTRONICS AND COMMUNICATION
ENGINEERING**

Under the guidance of

Mr. BALAJI K



**DEPARTMENT OF ELECTRONICS AND
COMMUNICATION ENGINEERING**

GRT INSTITUTE OF ENGINEERING AND TECHNOLOGY

Approved by AICTE, New Delhi Affiliated to

Anna University, Chennai

GRT Mahalakshmi nagar, Chennai – Tirupathi Highway , Tiruttani-631 209

PROJECT REPORT SUBMITTED BY,

NAME: SUNIL R

NM ID: au110321106053

Mail id: sunilprime03@gmail.com


Year/sem/dept: III/V/ECE

DECLARATION

I SUNIL R hereby declare that the project report entitled diabetes prediction system using artificial intelligence is done by myself under the guidance of Mr BALAJI K is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering degree in Electronics and Communication Engineering.

Date of submission : 01/11/2023

Place: GRT INSTITUTE OF ENGINEERING AND TECHNOLOGY, TIRUTTANI-631209



SIGNATURE OF THE CANDIDATE

PHASE V:-

Abstract

Globally, diabetes affects 537 million people, making it the deadliest and the most common non-communicable disease. Many factors can cause a person to get affected by diabetes, like excessive body weight, abnormal cholesterol level, family history, physical inactivity, bad food habit etc. Increased urination is one of the most common symptoms of this disease. People with diabetes for a long time can get several complications like heart disorder, kidney disease, nerve damage, diabetic retinopathy etc. But its risk can be reduced if it is predicted early. In this paper, an automatic diabetes prediction system has been developed using a private dataset of female patients in Bangladesh and various machine learning techniques. The authors used the Pima Indian diabetes dataset and collected additional samples from 203 individuals from a local textile factory in Bangladesh. Feature selection algorithm mutual information has been applied in this work. A semi-supervised model with extreme gradient boosting has been utilized to predict the insulin features of the private dataset. SMOTE and ADASYN approaches have been employed to manage the class imbalance problem. The authors used machine learning classification methods, that is, decision tree, SVM, Random Forest, Logistic Regression, KNN, and various ensemble techniques, to determine which algorithm produces the best prediction results. After training on and testing all the classification models, the proposed system provided the best result in the XGBoost classifier with the ADASYN approach with 81% accuracy, 0.81 F1 coefficient and AUC of 0.84. Furthermore, the domain adaptation method has been implemented to demonstrate the versatility of the proposed system. The explainable AI approach with LIME and SHAP frameworks is implemented to understand how the model predicts the final results. Finally, a website framework and an Android smartphone application have been developed to input various features and predict diabetes instantaneously.

INTRODUCTION

Diabetes is a chronic disease that directly affects the pancreas, and the body is incapable of producing insulin. Insulin is mainly responsible for maintaining the blood glucose level. Many factors, such as excessive body weight, physical inactivity, high blood pressure, and abnormal cholesterol level, can cause a person get affected by diabetes. It can cause many complications, but an increase in urination is one of the most common ones. It can damage the skin, nerves, and eyes, and if not treated early, diabetes can cause kidney failure and diabetic retinopathy ocular disease. According to IDF (International Diabetes Federation) statistics, 537 million people had diabetes around the world in 2021. In Bangladesh, approximately 7.10 million people had suffered from this disease, according to 2019 statistics.

Early and accurate diagnosis of diabetes mellitus, especially during its initial development, is challenging for medical professionals. Artificial intelligence and machine learning techniques, providing a reference, can help them gain preliminary knowledge about this disease and reduce their workload accordingly. Significant numbers of research have been performed to predict diabetes automatically using machine learning and ensemble techniques. Most of these works employed the open-source Pima Indian dataset. Some of these articles on automatic diabetes prediction employing the Pima Indian dataset are briefly discussed in the following paragraphs. For instance, Kumar used the random forest algorithm to design a system that can predict diabetes quickly and accurately. The dataset used in this work was collected from the UCI learning repository. First, the authors used conventional data preprocessing techniques, including data cleaning, integration, and reduction. The accuracy level was 90% using the random forest algorithm, which is much higher when compared to other algorithms. In a recent paper, Mohan and Jain used the SVM algorithm to analyze and predict diabetes with the help of the Pima Indian Diabetes Dataset. This work used four types of kernels, linear, polynomial, RBF, and sigmoid, to predict diabetes in the machine learning platform. The authors obtained diverse accuracies in

different kernels, ranging between 0.69 and 0.82. The SVM technique with radial basis kernel function obtained the highest accuracy of 0.82. Goyal and his team created a smart home health monitoring scheme to detect diabetes. The authors also employed the Pima Indian dataset for their research. For predicting blood pressure status, they used conditional decision making and for predicting diabetes, they used SVM, KNN, and decision tree. Among these models, SVM worked better as they got 75% accuracy which is better than other classifier algorithms. Hassan attempted to predict diabetes using different ensemble method-based machine learning algorithms and the Pima Indian dataset. The authors considered AUC (area under the ROC curve) as their accuracy measure. Finally, the proposed ensemble classifier accomplished an AUC value of 0.95. Jackins proposed a multi-disease prediction system, including diabetes using machine learning techniques and the Pima Indian dataset. According to the authors, the Naïve Bayes performed better than the random forest technique with accuracy increments of 0.43%. Mounika anticipated diabetes probabilities using machine learning techniques. This work employed the public Pima Indian dataset and multiple machine learning frameworks. Kumari attempted to apply a soft voting classifier-based ensemble approach for diabetes prediction. The proposed soft voting classifier attained the overall highest accuracy and F1 score of 0.791 and 0.716, respectively. Prabhu and Selvabharathi used the open-source Pima Indian diabetes dataset for predicting diabetes using the deep belief network model. The authors constructed the model in three phases, that is, data preprocessing using min–max normalization, constructing the network model, and fine-tuning the test dataset to remove any partiality using NN-FF classification. Finally, the authors have done all the implementation and simulation of the model using MATLAB. The authors reported an F1 score of 0.808, finding the best performance metric compared with the other classification methods.

Some of these works employed custom datasets or a combination of different datasets. In, the authors proposed a type 2 diabetes early prediction system using machine learning approaches. The authors employed a private dataset with more than 253,000 volunteer data from a local hospital in Korea for 6 years. Synthetic oversampling, SMOTE, and undersampling algorithms are applied to deal with the data imbalance problem. Various machine learning approaches are used to anticipate this disease for the following year from the past year's patients' data. Both the random forest and SVM classifiers achieved the highest F1 score of 74%. Pranto utilized Pima Indian and a private dataset from a local hospital in Bangladesh to design an automatic diabetes prediction system. This work trained several machine learning techniques on the Pima Indian dataset. KNN and decision tree models achieved 81.2% and 79.2% accuracies on the private dataset, respectively. Olisah implemented diabetes mellitus forecasting using advanced feature selection and machine learning models. The authors employed two open-source datasets, that is, Pima Indian and LMCH Iraqi databases. A polynomial regression-based preprocessing technique was used for predicting the missing samples. Hyperparameter tuning has been performed for the random forest, decision tree, and deep neural network frameworks. The proposed DNN technique with the optimized hyperparameters accomplished the highest accuracies of 0.972 and 0.973 for the Pima and LMCH datasets, respectively.

The applied machine learning models have been deployed into a website or smartphone application in some of the articles. In one study, the authors designed a website for the automatic prediction of diabetes. This work employed two open-source datasets and various popular machine learning approaches. The decision tree and random forest classifiers obtained the highest performance for this work with an accuracy of 0.968. Ramesh designed a remote and automatic system for diabetes forecasting with the Pima Indian dataset. The authors employed different data preprocessing techniques, that is, feature scaling, feature selection, and SMOTE. SVM with RBF kernel attained a maximum accuracy of 83.2%. The proposed ML framework is employed in an Android application.

We draw the conclusion that researchers have successfully combined multiple machine learning algorithms with diverse data preprocessing approaches for automatic diabetes detection by reviewing the relevant articles. Most

of the works focused on a single accuracy measure, used the open-source Pima Indian dataset, and did not develop the explicability of the prediction of the machine learning frameworks. These reasons have motivated us to evaluate our proposed prediction system based on accuracy, precision, recall, and F1 score, utilize more custom data to merge with the existing dataset, and apply an explainable AI technique.

In this paper, we have employed machine learning and explainable AI techniques to detect diabetes. Along with a private dataset from employees of a local textile industry in Bangladesh, we used the Pima Indian dataset in this paper. As there were many missing values in some attributes, we replaced them with the mean value of each feature. We have used the holdout validation technique to split the data. In this research paper, we have applied various machine learning-based classification algorithms, that is, decision tree, logistic regression, KNN, random forest, SVM, and ensemble techniques. Next, the performance of these classifiers has been evaluated in terms of precision, recall, and F1 measure. Finally, the best classifier has been selected as the final model to deploy into an Android smartphone application.

This paper implements diabetes mellitus prediction through machine learning. The significant contribution of this work is as follows:

A significant contribution of this work is to present a unique dataset of diabetes mellitus containing 203 samples. This private dataset has been obtained from female employees of Rownak Textile Mills Ltd, Dhaka, Bangladesh, referred to as the 'RTML dataset' in this paper. We have collected six features from 203 individuals, that is, pregnancy, glucose, blood pressure, skin thickness, BMI, age, and final outcome of diabetes.

Another contribution of this work is to keep similarities with the feature of the Pima Indian dataset. The missing insulin feature of the RTML dataset was predicted using a semi-supervised technique.

SMOTE and ADASYN techniques are implemented to minimize the class imbalance issue. Hyperparameter tuning has also been performed in this work.

Explainable AI technique with SHAP and LIME libraries is implemented to understand how the model predicts the decision. This approach helps to interpret what features play the most crucial role in terms of prediction.

A website and an Android application have been developed with the finalized best-performed model of this research work to make instantaneous predictions with real-time data.

The novelty of this work is to implement an automatic diabetes prediction website and Android application for a private dataset of female Bangladeshi patients using machine learning and ensemble techniques.

Background:

Diabetes is a chronic medical condition affecting millions of people worldwide. Early detection and timely intervention are crucial in managing and preventing the complications associated with diabetes. AI-based predictive models have the potential to assist healthcare professionals in identifying individuals at risk of developing diabetes or those who may already have undiagnosed diabetes.

Problem Statement:

Develop an AI-Based Diabetes Prediction System that can accurately predict the risk of diabetes in individuals based on their health data and lifestyle factors. The system should aim to achieve the following objectives:

1. **Data Collection:** Create a comprehensive database of health-related data, including but not limited to age, gender, family history of diabetes, body mass index (BMI), blood pressure, blood sugar levels, cholesterol levels, diet patterns, physical activity, and other relevant factors.
2. **Data Preprocessing:** Clean and preprocess the collected data, handling missing values and outliers, and ensuring data quality for model training.
3. **Feature Selection:** Identify the most relevant features for diabetes prediction from the dataset. Consider using techniques such as feature ranking, feature importance, or domain knowledge.
4. **Model Development:** Develop machine learning or deep learning models capable of predicting diabetes risk. Explore various algorithms, such as logistic regression, decision trees, random forests, support vector machines, neural networks, or ensemble methods. The model should take the selected features as input and provide a binary classification output (diabetic or non-diabetic).
5. **Model Evaluation:** Assess the performance of the developed model(s) using appropriate evaluation metrics, such as accuracy, precision, recall, F1-score, and AUC-ROC. Perform cross-validation to ensure the model's generalizability.
6. **User Interface:** Create a user-friendly interface that allows users to input their health and lifestyle data and receive a diabetes risk prediction. The interface should also provide explanations for the prediction and suggest actionable recommendations for individuals at risk.
7. **Privacy and Security:** Implement robust data privacy and security measures to protect the sensitive health data of users.

8. **Scalability and Deployment:** Ensure that the system is scalable to accommodate a growing user base. Deploy the system on a reliable and accessible platform, such as a web application or a mobile app.

9. **Continuous Improvement:** Implement mechanisms for collecting user feedback and improving the model's accuracy over time by incorporating new data and research findings.

10. **Regulatory Compliance:** Ensure that the system complies with relevant healthcare regulations and data protection laws, such as HIPAA (in the United States) or GDPR (in Europe).

Deliverables:

- A functional AI-Based Diabetes Prediction System.
- Documentation detailing the data collection process, preprocessing steps, feature selection, model development, and evaluation metrics.
- User guides and documentation for healthcare professionals and end-users.
- A secure and scalable deployment of the system.
- Compliance with privacy and security standards.
- Ongoing maintenance and improvement plan.

Note: This problem statement outlines the development of an AI-based diabetes prediction system; however, it's essential to work closely with healthcare experts and comply with ethical and legal guidelines throughout the project. Additionally, consider seeking necessary approvals and permissions when dealing with healthcare data.

Introduction to an AI-Based Diabetes Prediction System:

Diabetes is a global health concern affecting millions of people. Early diagnosis, risk assessment, and effective management are critical in combating this chronic disease. Artificial intelligence (AI) offers innovative solutions for improving diabetes care and prediction. An AI-based diabetes prediction system harnesses the power of machine learning and data analysis to assist healthcare providers and patients in understanding and managing diabetes risk.

1. Understanding Diabetes:

- Diabetes is a chronic metabolic disorder characterized by high blood glucose levels.
- It comes in various forms, with type 2 diabetes being the most prevalent, often linked to lifestyle and genetics.
- Early detection and risk assessment are vital for timely intervention and better disease management.

2. Role of AI:

- AI, including machine learning and deep learning techniques, has demonstrated significant potential in healthcare.
- These technologies can analyze extensive datasets, identify patterns, and make predictions, aiding in the early identification of diabetes risk factors.

3. Key Components of an AI-Based Diabetes Prediction System:

- **Data Collection:** Gathering comprehensive datasets including patient demographics, medical history, lifestyle information, and clinical test results.
- **Data Preprocessing:** Cleaning, normalizing, and preparing the data for analysis.
- **Feature Engineering:** Selecting and creating relevant features that influence diabetes risk.
- **Machine Learning Models:** Utilizing various machine learning algorithms to make predictions, classify risk, or diagnose diabetes.
- **Interpretability:** Ensuring the model's predictions are interpretable, offering insights into risk factors.
- **User Interface:** Creating user-friendly interfaces for healthcare providers and patients to input data and access predictions.

4. System Workflow:

- Patients or healthcare providers input relevant data into the system.
- The AI model analyzes the data to assess diabetes risk.
- Predictions and risk scores are provided through the user interface.
- Recommendations for further testing, lifestyle changes, or medical intervention are offered, promoting proactive healthcare.

5. Benefits of an AI-Based Diabetes Prediction System:

- Early Detection: Identifying individuals at risk of diabetes at an early stage, enabling timely intervention.
- Personalized Care: Tailoring recommendations based on individual risk factors.
- Data-Driven Insights: Harnessing the power of data to understand and mitigate diabetes risk.
- Improved Outcomes: Facilitating better disease management and reducing the impact of diabetes-related complications.

6. Regulatory Compliance and Privacy:

- Ensuring that the system complies with healthcare data privacy regulations, such as HIPAA (in the United States).
- Protecting patient data and maintaining data security.

7. Ongoing Development:

- Continuously improving the system through feedback loops, updates, and incorporating the latest medical research.

Developing an AI-based diabetes prediction system is a promising and impactful application of artificial intelligence in healthcare. Such a system can help in early detection, risk assessment, and management of diabetes. Here are the key steps and considerations for creating an AI-based diabetes prediction system:

1. **Data Collection:** Gather a comprehensive dataset that includes relevant features and patient data such as age, gender, family history, lifestyle, medical history, and clinical test results (e.g., glucose levels, HbA1c, BMI, etc.). You can obtain data from healthcare providers, research studies, or public health databases.
2. **Data Preprocessing:** Clean, normalize, and preprocess the data to ensure consistency and accuracy. Handle missing values and outliers appropriately.
3. **Feature Engineering:** Select the most relevant features and create new features if necessary. Feature selection and engineering are crucial for model accuracy.

4. **Model Selection:** Choose an appropriate machine learning or deep learning model for prediction. Common choices include logistic regression, decision trees, random forests, support vector machines, or deep learning models like neural networks.
5. **Training and Validation:** Split the dataset into training and validation sets to train and evaluate the model's performance. Use appropriate evaluation metrics, such as accuracy, precision, recall, F1-score, and area under the ROC curve (AUC), to assess the model's quality.
6. **Hyperparameter Tuning:** Fine-tune model hyperparameters to optimize performance. Techniques like cross-validation and grid search can be helpful.
7. **Model Evaluation:** Assess the model's predictive performance on an independent test dataset to ensure that it generalizes well.
8. **Interpretability:** Consider methods to make the model's predictions interpretable and explainable, which is crucial in a healthcare context.
9. **User Interface:** Develop a user-friendly interface for healthcare providers or end-users to input data and receive predictions. It should be designed with privacy and security in mind, especially when dealing with sensitive health information.
10. **Scalability:** Ensure that the system can handle a growing amount of data and users, as well as adapt to changing medical guidelines.
11. **Regulatory Compliance:** Be aware of the regulatory and legal requirements in your region regarding healthcare data and AI applications. Ensure that the system complies with data privacy laws, such as HIPAA in the United States.

12. **Validation and Clinical Trials:** Before deploying the system for real-world use, it may be necessary to conduct clinical trials and validation studies to assess its accuracy and safety in practice.
13. **Feedback Loop:** Establish a feedback mechanism to continuously improve the system's performance and stay up-to-date with the latest medical research.
14. **Deployment:** Once the system is fully developed and validated, deploy it in healthcare institutions, clinics, or as a mobile or web application for users.
15. **Education and Awareness:** Educate healthcare professionals and patients about the system's capabilities and limitations to ensure responsible and informed use.
16. **Dataset Description:** Provide a comprehensive description of the dataset used, including the number of samples, features, and their meanings. Discuss any challenges or limitations in the dataset.
17. **Model Interpretability:** Explain how the model's predictions are made interpretable for medical professionals and patients. Discuss the use of SHAP values, feature importance scores, or other methods.
18. **Cross-Validation:** Detail the cross-validation techniques used to assess the model's generalization performance. Explain how the data was split into training, validation, and test sets.
19. **Hyperparameter Optimization:** Describe the methods and tools used for hyperparameter tuning. Explain the rationale behind the selected hyperparameters.

- 20.**Model Deployment:** Provide a step-by-step guide on deploying the model, including the choice of deployment platform (e.g., cloud, on-premises). Discuss considerations for continuous model monitoring and updates.
- 21.**Data Security:** Elaborate on data security measures, such as encryption and access control. Explain how sensitive patient data is handled and protected.
- 22.**User Training and Support:** Outline training requirements for healthcare professionals using the system. Describe the support and maintenance plan.
- 23.**Legal and Regulatory Compliance:** Address legal considerations, including healthcare regulations and licensing requirements. Discuss any intellectual property rights associated with the system.
- 24.**Feedback Mechanism:** Explain how feedback from users and the system's performance is collected and utilized for continuous improvement.
- 25.**Cost-Benefit Analysis:** Present a cost-benefit analysis of implementing the AI-based system compared to traditional methods. Include potential cost savings and improvements in patient outcomes.
- 26.**Case Studies or Use Cases:** Provide real-world examples or case studies of the system in action. Share success stories and lessons learned from deployments.
- 27.**Future Development:** Discuss potential future developments and research directions in AI-based diabetes prediction. Consider emerging technologies and trends in healthcare.
- 28.**Collaboration and Partnerships:** Mention collaborations with healthcare providers, research institutions, or technology partners. Highlight the importance of interdisciplinary cooperation.

29. Patient Education and Engagement: Describe initiatives to educate patients about the system's capabilities and encourage their engagement. Include patient testimonials if available.

30. Challenges and Risks: Identify potential challenges, risks, and obstacles the system may face in implementation. Suggest strategies to mitigate these challenges.

Developing an AI-based diabetes prediction system requires collaboration between data scientists, medical professionals, and software engineers, along with a commitment to data privacy, ethics, and healthcare standards.

Creating an AI-based diabetes prediction system involves several steps, including data collection, preprocessing, model development, and deployment. Below, I'll provide a high-level overview of the coding process using Python and common machine learning libraries like scikit-learn.

Here's a simplified example of Python code for a diabetes prediction model using logistic regression:

python

Copy code

```
import pandas as pd

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report
```

```
# Load and preprocess the dataset
```

```
data = pd.read_csv("diabetes_data.csv")
```

```
X = data.drop("diabetes_label", axis=1)
```

```
y = data["diabetes_label"]
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

```
# Standardize features
```

```
scaler = StandardScaler()
```

```
X_train = scaler.fit_transform(X_train)
```

```
X_test = scaler.transform(X_test)
```

```
# Train the model
```

```
model = LogisticRegression()
```

```
model.fit(X_train, y_train)
```

```
# Make predictions
```

```
y_pred = model.predict(X_test)
```

```
# Evaluate the model
```

```
accuracy = accuracy_score(y_test, y_pred)
```

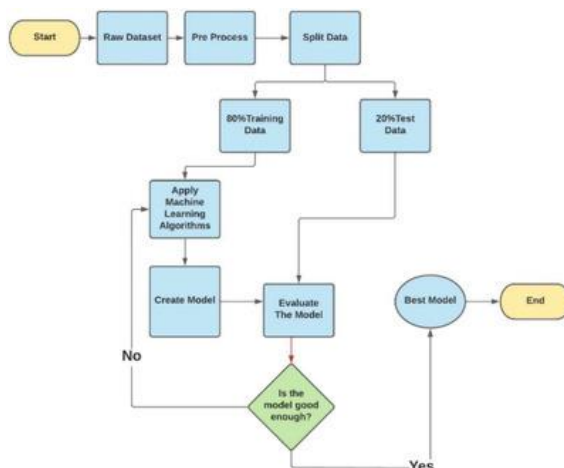
```
print("Accuracy:", accuracy)
```

```
print(classification_report(y_test, y_pred))
```

This is a basic example. In a real-world application, you would use a more complex model, perform extensive data preprocessing, and create a user interface for data input and result display. Additionally, you would conduct clinical validation studies to assess the system's accuracy and safety in a healthcare context.

1. PROPOSED SYSTEM

This section describes the working procedures and implementation of various machine learning techniques to design the proposed automatic diabetes prediction system. Figure 1 shows the different stages of this research work. First, the dataset was collected and preprocessed to remove the necessary discrepancies from the dataset,



for example, replacing null instances with mean values, dealing with imbalanced class issues etc. Then the dataset was separated into the training set and test set using the holdout validation technique. Next, different classification algorithms were applied to find the best classification algorithm for this dataset. Finally, the best-performed prediction model is deployed into the proposed website and smartphone application framework.

2.1. Dataset

The Pima Indian dataset is an open-source dataset [6] that is publicly available for machine learning classification, which has been used in this work along with a private dataset. It contains 768 patients' data, and 268 of them have developed diabetes. Figure shows the ratio of people having diabetes in the Pima Indian dataset. Table 1 demonstrates the eight features of the open-source Piman Indian dataset.

An external file that holds a picture, illustration, etc.

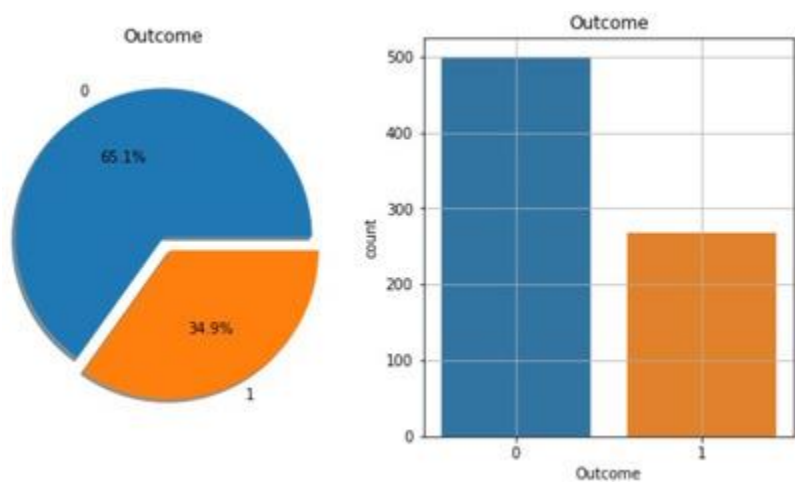


FIGURE
Percentage of people having diabetes in the Pima Indian dataset

TABLE 1

Features of the Pima Indian Dataset

Pregnancies	Skin thickness	Diabetes pedigree function
Glucose	Insulin	Age
Blood pressure	BMI	

RTML private dataset: A significant contribution of this work is to present a private dataset from Rownak Textile Mills Ltd, Dhaka, Bangladesh, referred to as RTML, to the scientific community. Following a brief explanation of the study to the female volunteers, they voluntarily agreed to participate in the study. This dataset comprises six features, that is, pregnancy, glucose, blood pressure, skin thickness, BMI, age, and outcome of diabetes from 203 female individuals aged between 18 and 77. In this work, blood glucose was measured by the GlucoLeader Enhance blood sugar meter. The blood pressure and skin thickness of the participants were

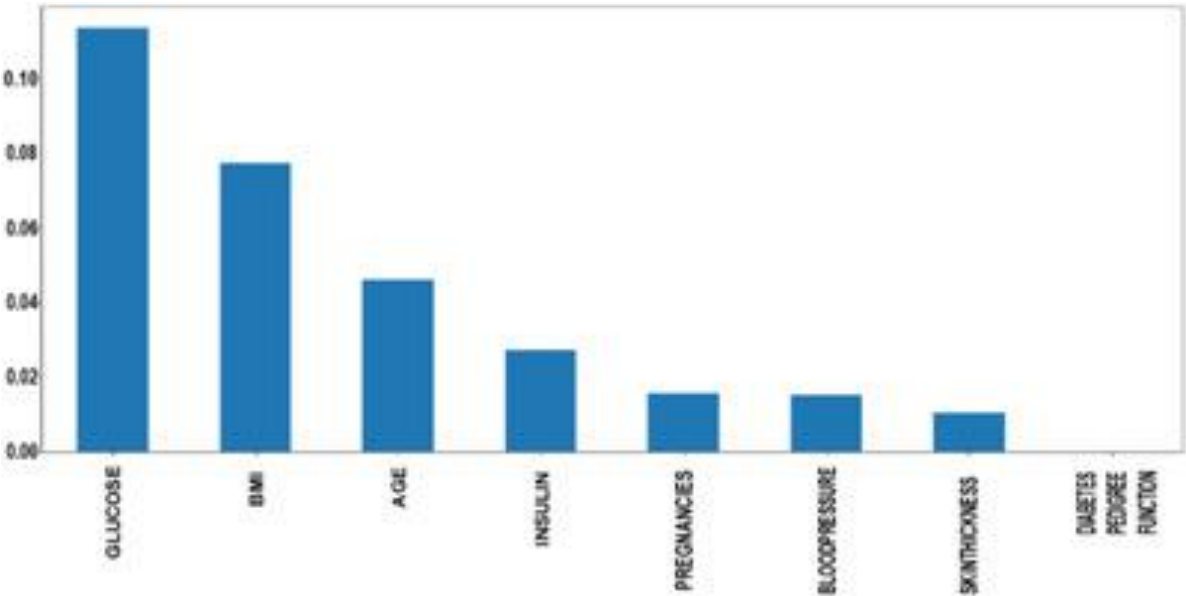
obtained by OMRON HEM-7156T and digital LCD body fat caliper machines, respectively. Table illustrates distinct features of the private RTML dataset with their minimum, maximum, and average values.

2.2. Dataset preprocessing

In the merged dataset, we discovered a few exceptional zero values. For example, skin thickness and Body Mass Index (BMI) cannot be zero. The zero value has been replaced by its corresponding mean value. The training and test dataset has been separated using the holdout validation technique, where 80% is the training data and 20% is the test data.

Mutual Information: Mutual information attempts to measure the interdependence of variables. It produces information gain, and its higher values indicate greater dependency.

Figure shows the mutual information of various features, that is, the importance of each attribute of this dataset. For example, according to this figure, the diabetes pedigree function seems less important according to this mutual information technique.



Semi-supervised learning: A combined dataset has been used in this work by incorporating the open-source Pima Indian and private RTML datasets. According to Table 2, the RTML dataset does not contain the insulin feature, which is predicted using a semi-supervised approach. Before merging the collected dataset with the Pima Indian dataset, a model was created using the extreme gradient boosting technique (XGB regressor). Various regression and ensemble learning techniques have been successfully used in many works to predict missing values [25, 26]. An extensive investigation has been performed while choosing the best-performed regressor technique to predict the insulin feature of the RTML dataset from the Pima Indian dataset. As the actual value of the insulin was not available in the RTML dataset, the Pima Indian dataset was initially used to select the best regression model. First, the Pima Indian dataset was divided into an 8:2 ratio and three supervised regression models, extreme gradient boosting technique (XGB), support vector regression (SVR), and Gaussian process regression (GPR), have been employed to predict the selected outcome, that is, insulin of the validation samples of the Pima Indian dataset.

Next, we computed the root mean square error (RMSE) of various regression frameworks as

$$RMSE = \sqrt{\frac{\sum_{i=1}^N (Predicted_i - Actual_i)^2}{N}}$$

where N denotes the total number of validation samples of the Pima Indian dataset.

According to Table, the XGB technique exhibits the lowest RMSE of insulin on the Pima Indian dataset. Therefore, this model has been used to predict the missing insulin column of the collected RTML dataset from the Pima Indian dataset. The working steps of predicting insulin in the RTML dataset have been illustrated in Figure.

TABLE 3

RMSE of various regression models on the Pima Indian dataset

Regression model	RMSE
XGB	0.36
SVR	0.45
GPR	0.43

Merged dataset: After the semi-supervised approach, we predicted the insulin feature and merged the RTML dataset with the Pima Indian dataset. The merged dataset contained 877 data with all the features, excluding the diabetes pedigree function, as it was the least important feature according to mutual information.

SMOTE and ADASYN for class imbalance: The merged dataset used in this work comprises the imbalance problem with 302 and 669 diabetes and non-diabetes samples, respectively. To take care of this problem, the SMOTE and ADASYN techniques have been applied to the training dataset, leaving the testing data unaffected. Adaptive Synthetic Sampling, known as ADASYN, is a synthetic data generation technique with the characteristics of not duplicating minority samples and generating more data for ‘harder to learn’ examples. As a result, the minority class will be sampled to the same extent as the majority class.

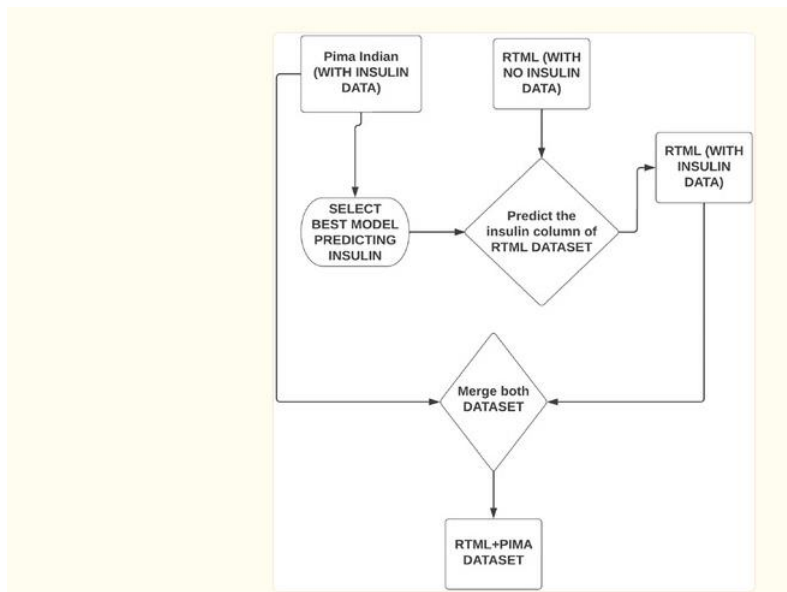


FIGURE 4

Working steps of predicting insulin of the RTML dataset

Min–Max normalization: In this research, we used the min–max normalization technique. The data has been scaled to the same range using the following equation:

$$X_{\text{scaled}} = \frac{X - X_{\min}}{X_{\max} - X_{\min}}$$

where X_{\max} and X_{\min} denote maximum and minimum values in the individual feature column, respectively.

2.3. Machine learning classifiers

In this work, various machine learning and ensemble techniques have been employed to implement the automatic diabetes prediction system, briefly discussed below. GridSearchCV framework has been employed in this research to find the optimal values of different hyperparameters for all the machine learning models to prevent overfitting.

Decision tree: A decision tree represents the learning function provided by a set of rules. The decision tree learning technique performs a method for approximating discrete-valued target functions. Gini or entropy are used to determine information gain, and each node is chosen based on these coefficients, which are expressed as

$$\text{Gini}_i = 1 - \sum_{k=1}^n (p_{i,k})^2$$

(3)

$$\text{Entropy} = - \sum_{i=1}^n p_i \log_2 p_i$$

(4)

In (3) and (4), n represents the number of distinct class values. We observed that max depth = 2, minimum samples leaf = 50, and ‘Gini’ impurity metrics work well in the employed dataset in this work using the GridSearchCV hyperparameter tuning.

KNN classifier: A discrete-valued function can be approximated by K number of nearest classifiers. To categorize, it creates a plane with the available training points and calculates the distance between the query and trained points. It determines the K number of neighbours (depending on the dataset) and classifies them using majority voting. In our research, we used $K = 5$ for the binary classification.

Random forest: Random forest is a machine learning system that averages the predictions of several decision trees. As a result, the random forest can be considered an ensemble learning model. In this research, we have applied random forest with estimators = 400, minimum samples leaf = 5, and ‘Gini’ impurity metrics utilizing hyperparameter tuning.

Support vector machine: SVM performs supervised classification by choosing the best hyperplane. In this study, we experimented with various SVM kernels in the training set. Finally, we discovered the SVM with a linear kernel, parameters $C = 10$ and $\gamma = 1$, produces the best results in this dataset.

Logistic regression: Logistic regression can be used to predict a binary class. To predict the outcome, it fits an ‘S’ shaped function. The hyperparameter optimization technique obtained the maximum number of iterations for the convergence of the logistic regression model to be 150.

AdaBoost: AdaBoost is an ensemble technique. This classifier initially works on the original dataset, then fits repeated copies of the classifier to the same dataset. This framework adjusts the weights of improperly classified instances so that successive classifiers focus more on difficult circumstances. We have applied AdaBoost with estimator = 50 and learning rate = 0.10 in this work.

XGBoost: XGBoost is an ensemble machine learning technique based on decision trees that employ a gradient boosting approach. The parameters used for the proposed XGBoost classifier are as follows: estimators’ maximum depth = 4 and ‘binary logistic’ objective function.

Voting classifier: It is an ensemble technique to improve the classification by voting. This paper implements a voting classifier that selects the majority class predicted by each classifier with a ‘soft’ voting hyperparameter.

Bagging: Bagging classifiers are ensemble classifiers that fit base classifiers to random subsets of the original dataset and then aggregate their individual predictions voting to generate a final classification. In the

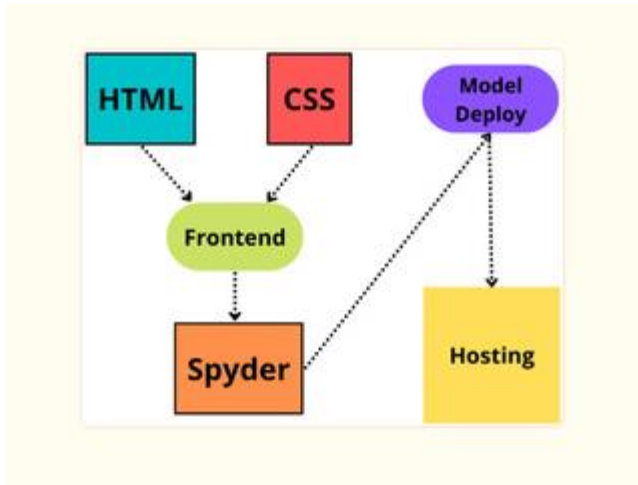
implemented bagging classifier, base estimators = 500, maximum number of samples = 100, and out-of-bag score = 'True' are used as various hyperparameters.

2.4. Deployment of the prediction system

The proposed machine learning-based diabetes prediction system has been deployed into a website and smartphone application framework to work instantaneously on real data.

Web application: We have used HTML and CSS for the frontend part of the proposed website. After that, we finalized the machine learning model XGBoost with ADASYN, as it provided the best performance. The model deployment has been done with Spyder, a Python environment platform that works with Anaconda.

Figure 5 shows the illustration of the website application development process.



This section presents the results and discussion of the proposed automatic diabetes prediction system. First, the performance of various machine learning techniques is discussed. Next, the implemented website framework and Android smartphone application are demonstrated. We used precision, recall, F1 score, AUC, and classification accuracy to evaluate various ML models. Equations of these metrics are expressed as

$$\text{Precision} = \frac{TP}{TP + FP} \quad (5)$$

$$\text{Recall} = \frac{TP}{TP + FN} \quad (6)$$

$$\text{F1 score} = \frac{2 \times \text{Recall} \times \text{Precision}}{\text{Recall} + \text{Precision}} \quad (7)$$

where TP denotes the model is predicting positive, and the result is also positive. FP indicates the positive prediction of the model, but the result is negative. TN expresses the model is predicting negative, and the result is also negative. FN indicates the model predicts negative, but the result is positive. In this work, the holdout validation approach with a stratified 8:2 train-test split has been used for all the machine learning models.

Table 4 compares different performance metrics of various classifiers for the merged dataset with SMOTE synthetic oversampling technique. According to this table, the bagging classifier achieved the best overall performance with 79% accuracy and 0.79 and 0.87 F1 score and AUC, respectively.

TABLE 4					
Performance metrics of various classifiers with SMOTE technique in the merged dataset					
Classifier	Precision	Recall	F1 Score	Accuracy	AUC
Logistic regression	0.78	0.77	0.77	77%	0.88
KNN	0.78	0.76	0.76	76%	0.85
Random forest	0.78	0.78	0.78	78%	0.87
Decision tree	0.75	0.73	0.73	73%	0.75
Bagging	0.80	0.79	0.79	79%	0.87
Adaboost	0.79	0.78	0.78	78%	0.85
XGboost	0.78	0.78	0.78	78%	0.84
Voting	0.79	0.79	0.79	79%	0.86
SVM	0.78	0.75	0.76	75%	0.87

Table 5 shows various performance metrics of all the classifiers using the ADASYN approach in the merged datasets. According to Table 4, the XGBoost framework performed better than other classifiers with 81% accuracy and 0.84 AUC. Conversely, the decision tree approach achieved the lowest accuracy and F1 score.

TABLE 5
Performance metrics of various classifiers using adasyn in the merged dataset

Classifier	Precision	Recall	F1 Score	Accuracy	Auc
Logistic regression	0.76	0.75	0.75	75%	0.84
KNN	0.76	0.73	0.73	73%	0.82
Random forest	0.76	0.76	0.76	76%	0.84
Decision tree	0.81	0.72	0.72	72%	0.78
Bagging	0.80	0.79	0.79	79%	0.84
AdaBoost	0.75	0.76	0.76	76%	0.84
XGBoost	0.81	0.81	0.81	81%	0.84
Voting	0.77	0.77	0.77	77%	0.84
SVM	0.78	0.78	0.77	78%	0.83

Next, the domain adaptation approach has been applied where the machine learning model is trained and evaluated on different samples, that is, source and target datasets, respectively. In this work, initially, the automatic diabetes prediction model is trained on the open-source Pima Indian dataset with a larger size. Finally, the model is evaluated on the private RTML dataset with a much smaller dimension. Table 6 demonstrates the performance metrics for the private dataset. It is interesting to note that the XGBoost with ADASYN framework has been applied in the training dataset in this case.

TABLE 6			
Performance metrics for the private dataset (domain adaptation technique)			
Precision	Recall	F1 score	Accuracy
0.95	0.96	0.95	96%

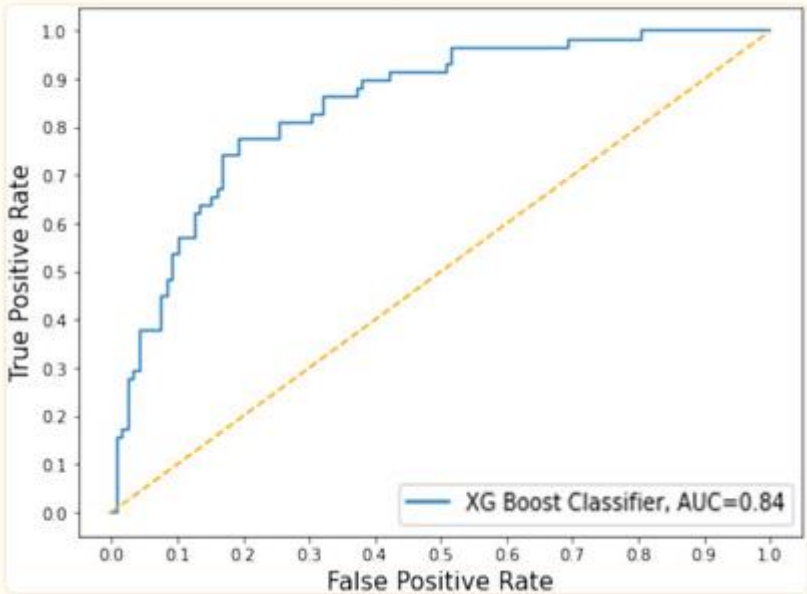


FIGURE 8

ROC curve and AUC value for the XGBoost with ADASYN

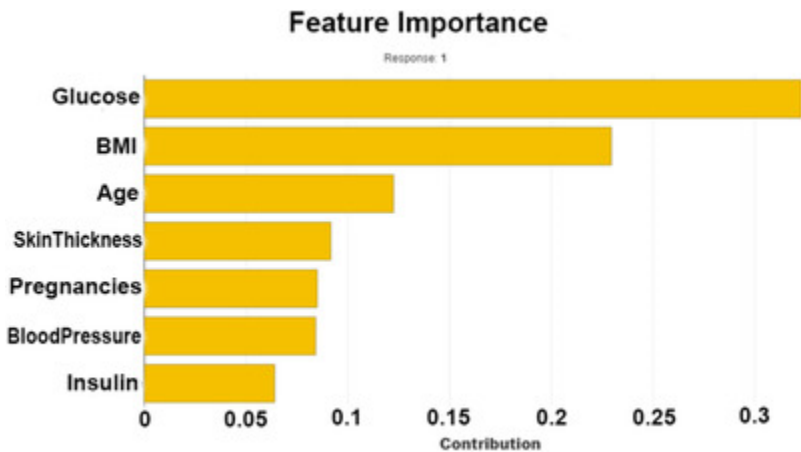


FIGURE 9

Explainable AI interpretation of feature importance of XGBoost with ADASYN

It is worth mentioning that the RTML dataset's insulin feature has been predicted from the Pima Indian dataset by applying the XGB regression technique for all of the results discussed above. However, alternative investigations have been conducted to obtain the

insulin feature of the RTML dataset, that is, the mean and median imputation of various patients’ insulin of the Pima Indian dataset. Tables [7](#) and [8](#) demonstrate various performance metrics of the machine learning models with the ADASYN technique when the RTML dataset's missing insulin features are obtained from the mean and median values of the Pima Indian dataset.

TABLE 7				
Performance metrics of classifiers in the merged dataset (RTML insulin obtained from Pima Indian mean)				
Classifier	Precision	Recall	F1 Score	Accuracy
AdaBoost	0.77	0.77	0.77	77%
Random Forest	0.77	0.76	0.76	76%
XGBoost	0.78	0.78	0.78	78%

TABLE 8				
Performance metrics of classifiers in the merged dataset (RTML insulin obtained from Pima Indian median)				
Classifier	Precision	Recall	F1 Score	Accuracy
AdaBoost	0.78	0.78	0.78	78%
Random Forest	0.76	0.76	0.76	76%
XGBoost	0.77	0.76	0.76	76%

Finally, another scenario has been considered where the insulin feature of the Pima Indian dataset has been removed to maintain consistency with the RTML dataset. Table [9](#) depicts various performance metrics of the merged dataset after removing the insulin feature. According to this table, the performance of all the prediction models degraded.

TABLE 9				
Performance metrics of classifiers in the merged dataset (insulin removed from Pima Indian)				
Classifier	Precision	Recall	F1 Score	Accuracy
AdaBoost	0.73	0.71	0.72	72%
Random Forest	0.72	0.70	0.71	71%

Classifier	Precision	Recall	F1 Score	Accuracy
XGBoost	0.74	0.73	0.73	74%

Table [10](#) illustrates the performance comparison of the proposed automatic diabetes prediction system with similar works to the Pima Indian dataset. According to this table, the proposed XGBoost technique with ADASYN outperformed most of the existing works concerning accuracy and F1 score.

TABLE 10

Comparison of the proposed system with similar diabetes prediction works

Reference	Classifier	F1 score	Accuracy	Other metrics
[3]	Deep belief network model	0.81	N/A	Precision: 0.68
				Recall: 1.0
[5]	SVM with RBF kernel		82%	
[9]	SVM	0.73	75%	Precision: 0.72
				Recall: 0.75
[10]	Ensemble (XGBoost)	0.81	88.8%	Precision: 0.84
				Recall: 0.79
[21]	Soft voting	0.72	79.1%	Precision: 0.73
				Recall: 0.72
This work	XGBoost with ADASYN	0.81	88.5%	Precision: 0.82
				Recall: 0.80

This study aims to predict diabetes mellitus automatically by employing machine learning techniques. Pima Indian dataset and a new RTML dataset comprising physical examination data from the local female patients of Bangladesh have been used. The missing insulin feature values of the RTML dataset have been predicted from the Pima Indian dataset. Our research found that the XGB regression technique accomplished the lowest RMS error in predicting insulin. The mutual information-based feature selection algorithm indicates the glucose level, BMI, age, and insulin to be the most salient features in predicting diabetes. SMOTE and ADASYN synthetic data oversampling and hyperparameters optimization techniques have been applied. The XGBoost technique with ADASYN achieved the best performance. The LIME and SHAP explainable AI

frameworks interpret the prediction provided by the ML approaches. A limitation of this study is the nonavailability of the insulin feature of the used RTML dataset. The prediction of insulin obtained from the XGB regressor and produced from the mean and median values of the Pima India dataset comprises an average deviation for classification accuracy of approximately 1.33% and 2.33%, respectively.

Diabetes can be a reason for reducing life expectancy and quality. Predicting this chronic disorder earlier can reduce the risk and complications of many diseases in the long run. In this paper, an automatic diabetes prediction system using various machine learning approaches has been proposed. The open-source Pima Indian and a private dataset of female Bangladeshi patients have been used in this work. SMOTE and ADASYN preprocessing techniques have been applied to handle the issue of imbalanced class problems. This research paper reported different performance metrics, that is, precision, recall, accuracy, F1 score, and AUC for various machine learning and ensemble techniques. The XGBoost classifier achieved the best performance with 81% accuracy and an F1 score and AUC of 0.81 and 0.84, respectively, with the ADASYN approach. Next, the domain adaptation technique has been applied to demonstrate the versatility of the proposed prediction system. Finally, the best-performed XGBoost framework has been deployed into a website and smartphone application to predict diabetes instantly. There are some future scopes of this work, for example, we recommend getting additional private data with a larger cohort of patients to get better results. Another extension of this work is combining machine learning models with fuzzy logic techniques and applying optimization approaches.

Diabetes is a chronic disease with the potential to cause a worldwide health care crisis. According to International Diabetes Federation 382 million people are living with diabetes across the whole world. By 2035, this will be doubled as 592 million. Diabetes is a disease caused due to the increase level of blood glucose. This high blood glucose produces the symptoms of frequent urination, increased thirst, and increased hunger. Diabetes is a one of the leading cause of blindness, kidney failure, amputations, heart failure and stroke. When we eat, our body turns food into sugars, or glucose. At that point, our pancreas is supposed to release insulin. Insulin serves as a key to open our cells, to allow the glucose to enter and allow us to use the glucose for energy. But with diabetes, this system does not work. Type 1 and type 2 diabetes are the most common forms of the disease, but there are also other kinds, such as gestational diabetes, which occurs during pregnancy, as well as other forms. Machine learning is an emerging scientific field in data science dealing with the ways in which machines learn from experience. The aim of this project is to develop a system which can perform early prediction of diabetes for a patient with a higher accuracy by combining the results of different machine learning techniques. The algorithms like K nearest neighbour, Logistic Regression, Random forest, Support vector machine and Decision tree are used. The accuracy of the model using each of the algorithms is calculated. Then the one with a good accuracy is taken as the model for predicting the diabetes. Keywords : Machine Learning, Diabetes, Decision tree, K nearest neighbour, Logistic Regression, Support vector Machine, Accuracy.

Diabetes is the fast growing disease among the people even among the youngsters. In understanding diabetes and how it develops, we need to understand what happens in the body without diabetes. Sugar (glucose) comes from the foods that we eat, specifically carbohydrate foods. Carbohydrate foods provide our body with its main energy source everybody, even those people with diabetes, needs carbohydrate. Carbohydrate foods include bread, cereal, pasta, rice, fruit, dairy products and vegetables (especially starchy vegetables). When we eat these foods, the body breaks them down into glucose. The glucose moves around the body in the bloodstream. Some of the glucose is taken to our brain to help us think clearly and function. The remainder of the glucose is taken to the cells of our body for energy and also to our liver, where it is stored as energy that is used later by the body. In order for the body to use glucose for energy, insulin is required. Insulin is a hormone that is produced by the beta cells in the pancreas. Insulin works like a key to a door. Insulin attaches itself to doors on the cell, opening the door to allow glucose to move from the blood stream, through the door, and into the cell. If the pancreas is not able to produce enough insulin (insulin deficiency) or if the body cannot use the insulin it produces (insulin resistance), glucose builds up in the bloodstream (hyperglycaemia) and diabetes develops. Diabetes Mellitus means high levels of sugar (glucose) in the blood stream and in the urine.

Types of Diabetes

Type 1 diabetes means that the immune system is compromised and the cells fail to produce insulin in sufficient amounts. There are no eloquent studies that prove the causes of type 1 diabetes and there are currently no known methods of prevention.

Type 2 diabetes means that the cells produce a low quantity of insulin or the body can't use the insulin correctly. This is the most common type of diabetes, thus affecting 90% of persons diagnosed with diabetes. It is caused by both genetic factors and the manner of living.

Gestational diabetes appears in pregnant women who suddenly develop high blood sugar. In two thirds of the cases, it will reappear during subsequent pregnancies. There is a great chance that type 1 or type 2 diabetes will occur after a pregnancy affected by gestational diabetes.

Symptoms of Diabetes

- Frequent Urination

- Increased thirst
- Tired/Sleepiness
- Weight loss
- Blurred vision
- Mood swings
- Confusion and difficulty concentrating
- frequent infections

Causes of Diabetes

Genetic factors are the main cause of diabetes. It is caused by at least two mutant genes in the chromosome 6, the chromosome that affects the response of the body to various antigens. Viral infection may also influence the occurrence of type 1 and type 2 diabetes. Studies have shown that infection with viruses such as rubella, Cocksackievirus, mumps, hepatitis B virus, and cytomegalovirus increase the risk of developing diabetes.

LITERATURE REVIEW

Yasodhaet al.[1] uses the classification on diverse types of datasets that can be accomplished to decide if a person is diabetic or not. The diabetic patient's data set is established by gathering data from hospital warehouse which contains two hundred instances with nine attributes. These instances of this dataset are referring to two groups i.e. blood tests and urine tests. In this study the implementation can be done by using WEKA to classify the data and the data is assessed by means of 10-fold cross validation approach, as it performs very well on small datasets, and the outcomes are compared. The naïve Bayes, J48, REP Tree and Random Tree are used. It was concluded that J48 works best showing an accuracy of 60.2% among others.

Aiswaryaet al. [2] aims to discover solutions to detect the diabetes by investigating and examining the patterns originate in the data via classification analysis by using Decision Tree and Naïve Bayes algorithms. The research hopes to propose a faster and more efficient method of identifying the disease that will help in well-timed cure of the patients. Using PIMA dataset and cross validation approach the study concluded that J48 algorithm gives an accuracy rate of 74.8% while the naïve Bayes gives an accuracy of 79.5% by using 70:30 split.

Gupta et al. [3] aims to find and calculate the accuracy, sensitivity and specificity percentage of numerous classification methods and also tried to compare and analyse the results of several classification methods in WEKA, the study compares the performance of same classifiers when implemented on some other tools which includes Rapidminer and Matlab using the same parameters (i.e. accuracy, sensitivity and specificity). They applied JRIP, Jgrapt and BayesNet algorithms. The result shows that Jgrapt shows highest accuracy i.e 81.3%, sensitivity is 59.7% and specificity is 81.4%. It was also concluded that WEKA works best than Matlab and Rapidminner.

Lee et al. [4] focus on applying a decision tree algorithm named as CART on the diabetes dataset

after applying the resample filter over the data. The author emphasis on the class imbalance problem and the need to handle this problem before applying any algorithm to achieve better accuracy rates. The class imbalance is a mostly occur in a dataset having dichotomous values, which means that the class variable have two possible outcomes and can be handled easily if observed earlier in data preprocessing stage and will help in boosting the accuracy of the predictive model.

METHODOLOGY

In this section we shall learn about the various classifiers used in machine learning to predict diabetes. We shall also explain our proposed methodology to improve the accuracy. Five different methods were used in this paper. The different methods used are defined below. The output is the accuracy metrics of the machine learning models. Then, the model can be used in prediction.

Dataset Description

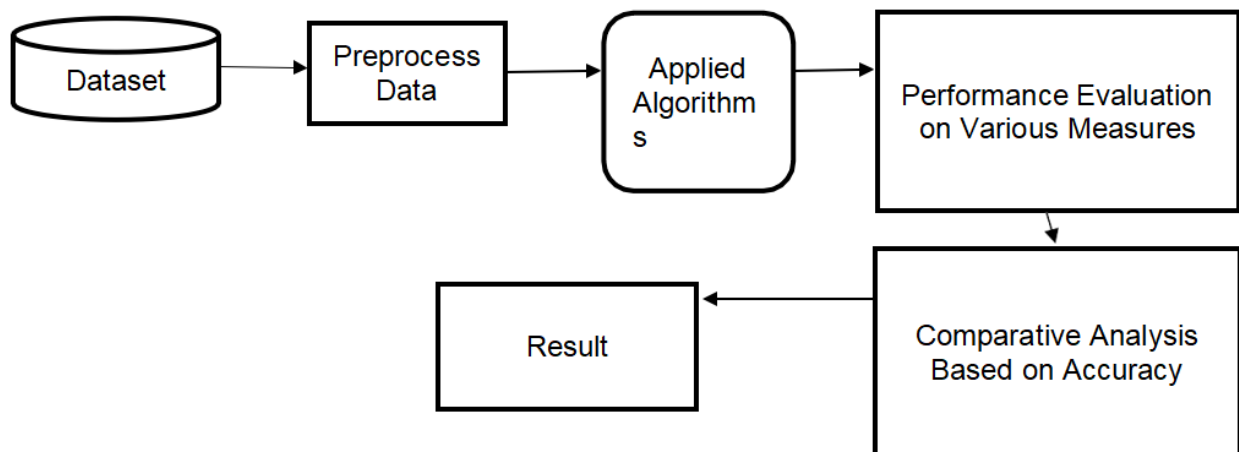
The diabetes data set was originated from <https://www.kaggle.com/johndasilva/diabetes>. Diabetes dataset containing 2000 cases. The objective is to predict based on the measures to predict if the patient is diabetic or not.

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	2	138	62	35	0	33.6	0.127	47	1
1	0	84	82	31	125	38.2	0.233	23	0
2	0	145	0	0	0	44.2	0.630	31	1
3	0	135	68	42	250	42.3	0.365	24	1
4	1	139	62	41	480	40.7	0.536	21	0

- ➔ The diabetes data set consists of 2000 data points, with 9 features each.
- ➔ “Outcome” is the feature we are going to predict, 0 means No diabetes, 1 means diabetes.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2000 entries, 0 to 1999
Data columns (total 9 columns):
#   Column                                Non-Null Count  Dtype
---  ---                                ---
0   Pregnancies                          2000 non-null   int64
1   Glucose                             2000 non-null   int64
2   BloodPressure                       2000 non-null   int64
3   SkinThickness                      2000 non-null   int64
4   Insulin                            2000 non-null   int64
5   BMI                                2000 non-null   float64
6   DiabetesPedigreeFunction            2000 non-null   float64
7   Age                                2000 non-null   int64
8   Outcome                             2000 non-null   int64
dtypes: float64(2), int64(7)
memory usage: 140.8 KB
```

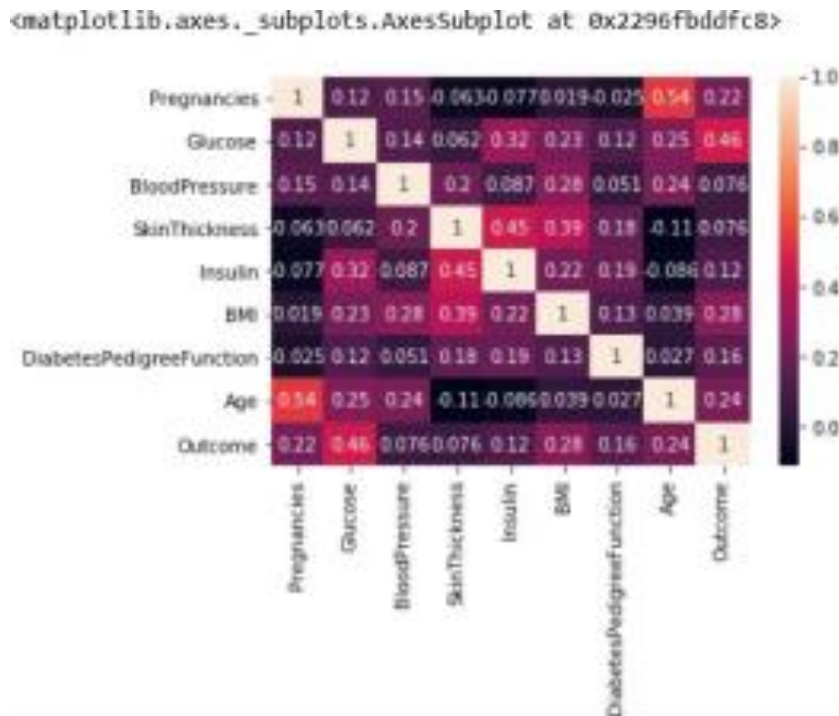
There is no null values in dataset.



Proposed Model Diagram

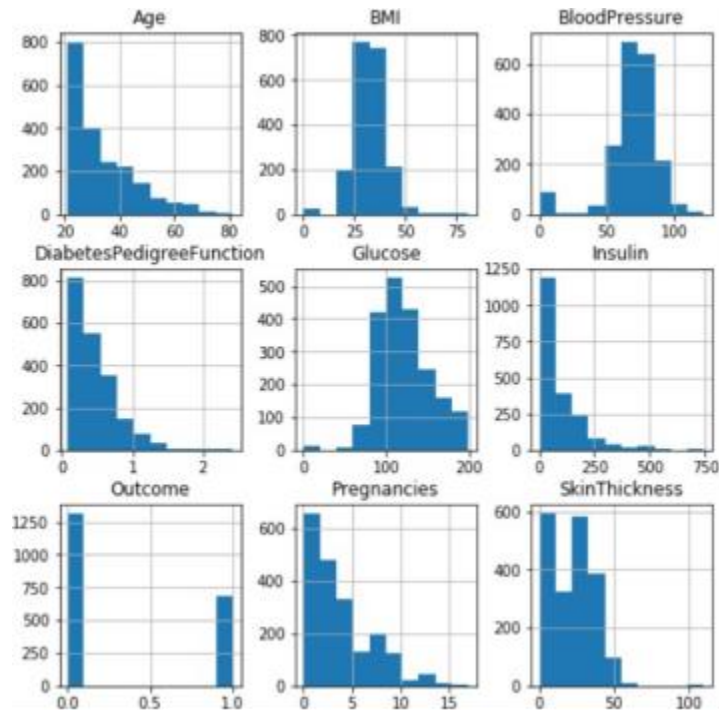
RESULT & DISCUSSION

Correlation Matrix:

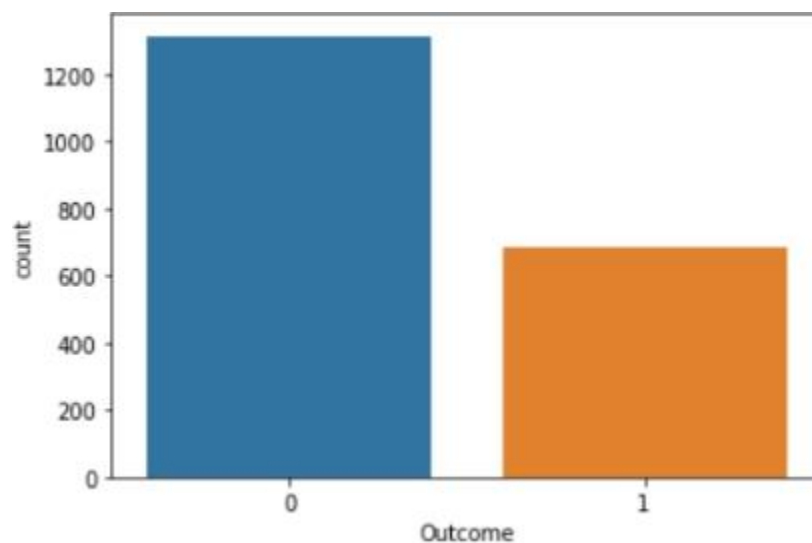


It is easy to see that there is no single feature that has a very high correlation with our outcome value. Some of the features have a negative correlation with the outcome value and some have positive.

Histogram:



Let's take a look at the plots. It shows how each feature and label is distributed along different ranges, which further confirms the need for scaling. Next, wherever you see discrete bars, it basically means that each of these is actually a categorical variable. We will need to handle these categorical variables before applying Machine Learning. Our outcome labels have two classes, 0 for no disease and 1 for disease.



Bar Plot For Outcome Class

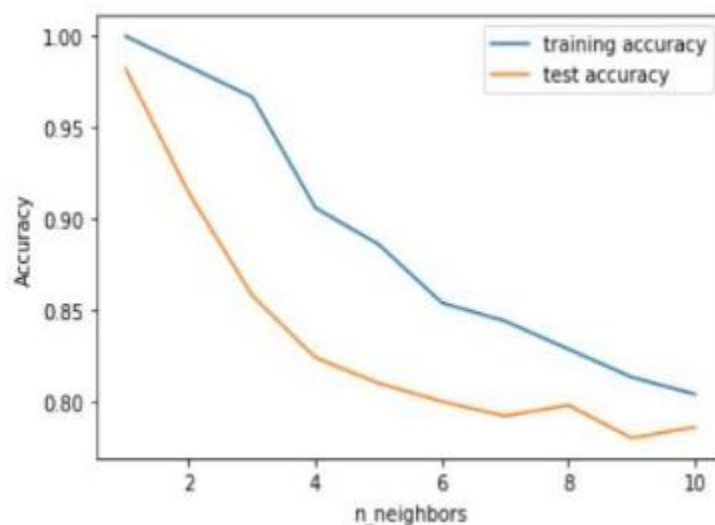
The above graph shows that the data is biased towards datapoints having outcome value

as 0 where it means that diabetes was not present actually. The number of non-diabetics is almost twice the number of diabetic patients.

k-Nearest Neighbors:

The k-NN algorithm is arguably the simplest machine learning algorithm. Building the model consists only of storing the training data set. To make a prediction for a new data point, the algorithm finds the closest data points in the training data set, its “nearest neighbors.”

First, let’s investigate whether we can confirm the connection between model complexity and accuracy:



The above plot shows the training and test set accuracy on the y-axis against the setting of n_neighbors on the x-axis. Considering if we choose one single nearest neighbor, the prediction on the training set is perfect. But when more neighbors are considered, the training accuracy drops, indicating that using the single nearest neighbor leads to a model that is too complex. The best performance is somewhere around 9 neighbors.

Training Accuracy	0.8 1
Testing Accuracy	0.7 8

Table-1

Logistic regression:

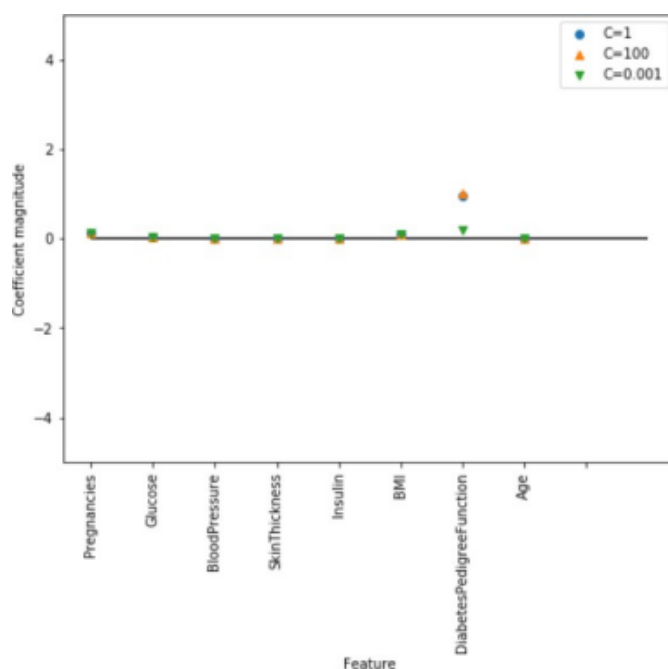
Logistic Regression is one of the most common classification algorithms.

	Training Accuracy	Testing Accuracy
C=1	0.779	0.788
C=0.01	0.784	0.780
C=100	0.778	0.792

Table-2

- ➔ In first row, the default value of $C=1$ provides with 77% accuracy on the training and 78% accuracy on the test set.
- ➔ In second row, using $C=0.01$ results are 78% accuracy on both the training and the test sets.
- ➔ Using $C=100$ results in a little bit lower accuracy on the training set and a little bit higher accuracy on the test set, confirming that less regularization and a more complex model may not generalize better than the default setting.

Therefore, we should choose the default value $C=1$.



Decision Tree:

This classifier creates a decision tree based on which, it assigns the class values to each data point. Here, we can vary the maximum number of features to be considered while creating the model.

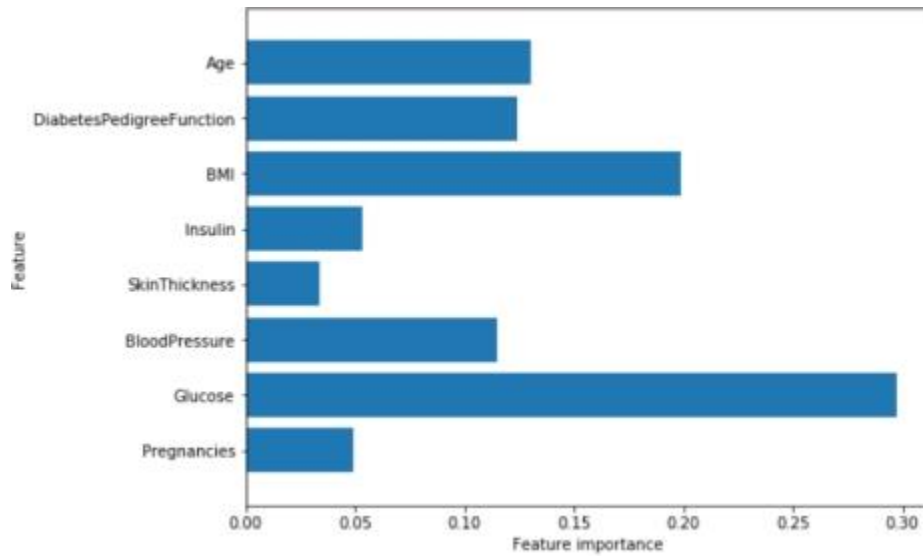
Training Accuracy	1.0 0
Testing Accuracy	0.9 9

Table-3

The accuracy on the training set is 100% and the test set accuracy is also good.

Feature Importance in Decision Trees

Feature importance rates how important each feature is for the decision a tree makes. It is a number between 0 and 1 for each feature, where 0 means “not used at all” and 1 means “perfectly predicts the target”.



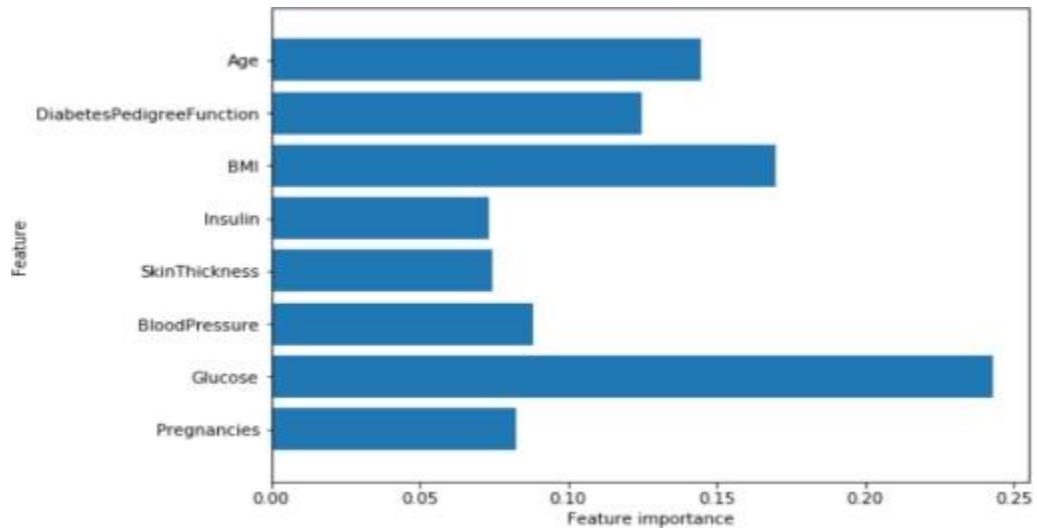
Feature “Glucose” is by far the most important feature.

Random Forest:

This classifier takes the concept of decision trees to the next level. It creates a forest of trees where each tree is formed by a random selection of features from the total features.

Training Accuracy	1.00
Testing Accuracy	0.97 4

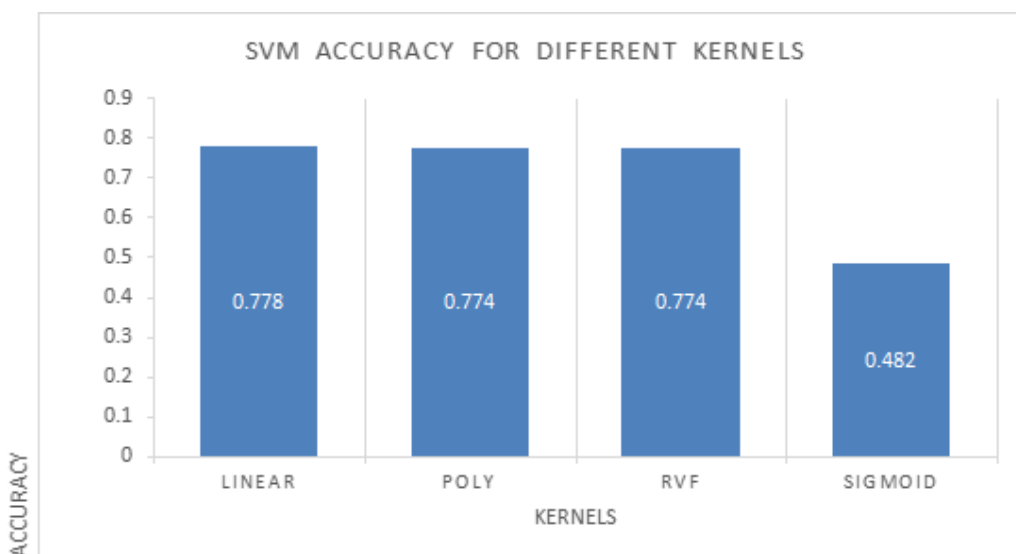
Feature importance in Random Forest:



Similarly to the single decision tree, the random forest also gives a lot of importance to the “Glucose” feature, but it also chooses “BMI” to be the 2nd most informative feature overall.

Support Vector Machine:

This classifier aims at forming a hyper plane that can separate the classes as much as possible by adjusting the distance between the data points and the hyper plane. There are several kernels based on which the hyper plane is decided. I tried four kernels namely, linear, poly, rbf, and sigmoid.



As can be seen from the plot above, the linear kernel performed the best for this dataset and achieved a score of 77%.

Accuracy Comparison:

Algorithms	Training Accuracy	Testing Accuracy
k-Nearest Neighbors	81%	78%
Logistic Regression	78%	78%
Decision Tree	98%	99%
Random Forest	94%	97%
SVM	76%	77%

Table-5

Table-5 shows the accuracy values for all five machine learning algorithms.

Table-5 shows that Decision Tree algorithm gives the best accuracy with 98% training accuracy and 99% testing accuracy.

SOURCE CODE

Here's a simplified example of Python code for a diabetes prediction model using logistic regression:

python

Copy code

```
import pandas as pd

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report

# Load and preprocess the dataset
data = pd.read_csv("diabetes_data.csv")
X = data.drop("diabetes_label", axis=1)
y = data["diabetes_label"]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Standardize features
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# Train the model
model = LogisticRegression()
model.fit(X_train, y_train)

# Make predictions
y_pred = model.predict(X_test)

# Evaluate the model
```

```

accuracy = accuracy_score(y_test, y_pred)

print("Accuracy:", accuracy)

print(classification_report(y_test, y_pred))

#Installation of required libraries

import numpy as np
import pandas as pd
import statsmodels.api as sm
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.preprocessing import scale, StandardScaler
from sklearn.model_selection import train_test_split, GridSearchCV, cross_val_score
from sklearn.metrics import confusion_matrix, accuracy_score, mean_squared_error, r2_score, roc_auc_score,
roc_curve, classification_report
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.neural_network import MLPClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import GradientBoostingClassifier
from lightgbm import LGBMClassifier
from sklearn.model_selection import KFold
import warnings
warnings.simplefilter(action = "ignore")
#Reading the dataset
df = pd.read_csv("../input/pima-indians-diabetes-database/diabetes.csv")

```

In [3]:

```

# The first 5 observation units of the data set were accessed.
df.head()

# The size of the data set was examined. It consists of 768 observation units and 9 variables.
df.shape

#Feature information
df.info()

<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

In [6]:

Descriptive statistics of the data set accessed.

```
df.describe([0.10,0.25,0.50,0.75,0.90,0.95,0.99]).T
```

The distribution of the Outcome variable was examined.

```
df["Outcome"].value_counts()*100/len(df)
```

The classes of the outcome variable were examined.

```
df.Outcome.value_counts()
```

The histogram of the Age variable was reached.

```
df["Age"].hist(edgecolor = "black"
```

```
print("Max Age: " + str(df["Age"].max()) + " Min Age: " + str(df["Age"].min()))
```

Histogram and density graphs of all variables were accessed.

```
fig, ax = plt.subplots(4,2, figsize=(16,16))
```

```
sns.distplot(df.Age, bins = 20, ax=ax[0,0])
```

```
sns.distplot(df.Pregnancies, bins = 20, ax=ax[0,1])
```

```
sns.distplot(df.Glucose, bins = 20, ax=ax[1,0])
```

```
sns.distplot(df.BloodPressure, bins = 20, ax=ax[1,1])
```

```
sns.distplot(df.SkinThickness, bins = 20, ax=ax[2,0])
```

```
sns.distplot(df.Insulin, bins = 20, ax=ax[2,1])
```

```
sns.distplot(df.DiabetesPedigreeFunction, bins = 20, ax=ax[3,0])
```

```
sns.distplot(df.BMI, bins = 20, ax=ax[3,1])
```

```
df.groupby("Outcome").agg({"Insulin": "mean"})
```

```
df.groupby("Outcome").agg({"Insulin": "mean"})
```

```
df.groupby("Outcome").agg({"Insulin": "mean"})
```

```
df.groupby("Outcome").agg({"Glucose": "max"})
```

The distribution of the outcome variable in the data was examined and visualized.

```
f,ax=plt.subplots(1,2,figsize=(18,8))
```

```
df['Outcome'].value_counts().plot.pie(explode=[0,0.1],autopct='%1.1f%%',ax=ax[0],shadow=True)
```

```
ax[0].set_title('target')
```

```
ax[0].set_ylabel('')
```

```
sns.countplot('Outcome',data=df,ax=ax[1])
```

```
ax[1].set_title('Outcome')
```

```
plt.show()
```

Access to the correlation of the data set was provided. What kind of relationship is examined between the variables.

If the correlation value is > 0, there is a positive correlation. While the value of one variable increases, the value of the other variable also increases.

Correlation = 0 means no correlation.

If the correlation is < 0, there is a negative correlation. While one variable increases, the other variable decreases.

When the correlations are examined, there are 2 variables that act as a positive correlation to the Salary dependent variable.

These variables are Glucose. As these increase, Outcome variable increases.

```
df.corr()
```

Correlation matrix graph of the data set

```
f, ax = plt.subplots(figsize= [20,15])
sns.heatmap(df.corr(), annot=True, fmt=".2f", ax=ax, cmap = "magma" )
ax.set_title("Correlation Matrix", fontsize=20)
plt.show()
df[['Glucose','BloodPressure','SkinThickness','Insulin','BMI']] =
df[['Glucose','BloodPressure','SkinThickness','Insulin','BMI']].replace(0,np.NaN)

df.head()

# Now, we can look at where are missing values
df.isnull().sum()
# Have been visualized using the missingno library for the visualization of missing observations.
# Plotting
import missingno as msno
msno.bar(df);
```

The missing values will be filled with the median values of each variable.

```
def median_target(var):
    temp = df[df[var].notnull()]
    temp = temp[[var, 'Outcome']].groupby(['Outcome'])[var].median().reset_index()
    return temp
```

In [28]:

```
# The values to be given for incomplete observations are given the median value of people who are not sick and
the median values of people who are sick.
columns = df.columns
columns = columns.drop("Outcome")
for i in columns:
    median_target(i)
    df.loc[(df['Outcome'] == 0 ) & (df[i].isnull()), i] = median_target(i)[i][0]
    df.loc[(df['Outcome'] == 1 ) & (df[i].isnull()), i] = median_target(i)[i][1]
```

In [29]:

```
df.head()

# Missing values were filled.
df.isnull().sum()
# In the data set, there were asked whether there were any outlier observations compared to the 25% and 75%
quarters.
# It was found to be an outlier observation.
for feature in df:

    Q1 = df[feature].quantile(0.25)
    Q3 = df[feature].quantile(0.75)
    IQR = Q3-Q1
    lower = Q1- 1.5*IQR
    upper = Q3 + 1.5*IQR

    if df[(df[feature] > upper)].any(axis=None):
        print(feature,"yes")
    else:
        print(feature, "no")

# The process of visualizing the Insulin variable with boxplot method was done. We find the outlier observations
on the chart.
```

```

import seaborn as sns
sns.boxplot(x = df["Insulin"]);
#We conduct a stand alone observation review for the Insulin variable
#We suppress contradictory values
Q1 = df.Insulin.quantile(0.25)
Q3 = df.Insulin.quantile(0.75)
IQR = Q3-Q1
lower = Q1 - 1.5*IQR
upper = Q3 + 1.5*IQR
df.loc[df["Insulin"] > upper,"Insulin"] = upper

```

In [34]:

```

import seaborn as sns
sns.boxplot(x = df["Insulin"]);

# We determine outliers between all variables with the LOF method
from sklearn.neighbors import LocalOutlierFactor
lof =LocalOutlierFactor(n_neighbors= 10)
lof.fit_predict(df)
df_scores = lof.negative_outlier_factor_
np.sort(df_scores)[0:30]
#We choose the threshold value according to lof scores
threshold = np.sort(df_scores)[7]
threshold
#We delete those that are higher than the threshold
outlier = df_scores > threshold
df = df[outlier]

```

In [39]:

```

# The size of the data set was examined.
df.shape

# According to BMI, some ranges were determined and categorical variables were assigned.
NewBMI = pd.Series(["Underweight", "Normal", "Overweight", "Obesity 1", "Obesity 2", "Obesity 3"], dtype =
"category")
df["NewBMI"] = NewBMI
df.loc[df["BMI"] < 18.5, "NewBMI"] = NewBMI[0]
df.loc[(df["BMI"] > 18.5) & (df["BMI"] <= 24.9), "NewBMI"] = NewBMI[1]
df.loc[(df["BMI"] > 24.9) & (df["BMI"] <= 29.9), "NewBMI"] = NewBMI[2]
df.loc[(df["BMI"] > 29.9) & (df["BMI"] <= 34.9), "NewBMI"] = NewBMI[3]
df.loc[(df["BMI"] > 34.9) & (df["BMI"] <= 39.9), "NewBMI"] = NewBMI[4]
df.loc[df["BMI"] > 39.9 , "NewBMI"] = NewBMI[5]

```

In [41]:

```

df.head()

# A categorical variable creation process is performed according to the insulin value.
def set_insulin(row):
    if row["Insulin"] >= 16 and row["Insulin"] <= 166:
        return "Normal"
    else:
        return "Abnormal"

```

In [43]:

```

# The operation performed was added to the dataframe.

```

```
df = df.assign(NewInsulinScore=df.apply(set_insulin, axis=1))
```

```
df.head()
```

Some intervals were determined according to the glucose variable and these were assigned categorical variables.

```
NewGlucose = pd.Series(["Low", "Normal", "Overweight", "Secret", "High"], dtype = "category")
```

```
df["NewGlucose"] = NewGlucose
```

```
df.loc[df["Glucose"] <= 70, "NewGlucose"] = NewGlucose[0]
```

```
df.loc[(df["Glucose"] > 70) & (df["Glucose"] <= 99), "NewGlucose"] = NewGlucose[1]
```

```
df.loc[(df["Glucose"] > 99) & (df["Glucose"] <= 126), "NewGlucose"] = NewGlucose[2]
```

```
df.loc[df["Glucose"] > 126, "NewGlucose"] = NewGlucose[3]
```

In [45]:

```
df.head()
```

Here, by making One Hot Encoding transformation, categorical variables were converted into numerical values. It is also protected from the Dummy variable trap.

```
df = pd.get_dummies(df, columns=["NewBMI", "NewInsulinScore", "NewGlucose"], drop_first = True)
```

In [47]:

```
df.head()
```

```
categorical_df = df[['NewBMI_Obesity 1', 'NewBMI_Obesity 2', 'NewBMI_Obesity 3',  
'NewBMI_Overweight', 'NewBMI_Underweight',  
                    'NewInsulinScore_Normal', 'NewGlucose_Low', 'NewGlucose_Normal', 'NewGlucose_Overweight',  
'NewGlucose_Secret']]
```

In [49]:

```
categorical_df.head()
```

```
y = df["Outcome"]
```

```
X = df.drop(["Outcome", 'NewBMI_Obesity 1', 'NewBMI_Obesity 2', 'NewBMI_Obesity 3',
```

```
'NewBMI_Overweight', 'NewBMI_Underweight',
```

```
'NewInsulinScore_Normal', 'NewGlucose_Low', 'NewGlucose_Normal', 'NewGlucose_Overweight',
```

```
'NewGlucose_Secret'], axis = 1)
```

```
cols = X.columns
```

```
index = X.index
```

In [51]:

```
X.head()
```

The variables in the data set are an effective factor in increasing the performance of the models by standardization.

There are multiple standardization methods. These are methods such as "Normalize", "MinMax", "Robust" and "Scale".

```
from sklearn.preprocessing import RobustScaler
```

```
transformer = RobustScaler().fit(X)
```

```
X = transformer.transform(X)
```

```
X = pd.DataFrame(X, columns = cols, index = index)
```

In [53]:

```
X.head()
```

```
X = pd.concat([X, categorical_df], axis = 1)
```

In [55]:

```
X.head()
```

```
y.head()
```

```
rf_params = {"n_estimators": [100,200,500,1000],  
            "max_features": [3,5,7],  
            "min_samples_split": [2,5,10,30],  
            "max_depth": [3,5,8,None]}
```

In [60]:

```
rf_model = RandomForestClassifier(random_state = 12345)
```

In [61]:

```
gs_cv = GridSearchCV(rf_model,  
                    rf_params,  
                    cv = 10,  
                    n_jobs = -1,  
                    verbose = 2).fit(X, y)
```

```
gs_cv.best_params_  
rf_tuned = RandomForestClassifier(**gs_cv.best_params_)
```

In [64]:

```
rf_tuned = rf_tuned.fit(X,y)
```

In [65]:

```
cross_val_score(rf_tuned, X, y, cv = 10).mean()  
feature_imp = pd.Series(rf_tuned.feature_importances_,  
                        index=X.columns).sort_values(ascending=False)
```

```
sns.barplot(x=feature_imp, y=feature_imp.index)  
plt.xlabel('Significance Score Of Variables')  
plt.ylabel('Variables')  
plt.title("Variable Severity Levels")  
plt.show()
```

```
models = []
```

```
models.append(('RF', RandomForestClassifier(random_state = 12345, max_depth = 8, max_features = 7,  
min_samples_split = 2, n_estimators = 500)))  
models.append(('XGB', GradientBoostingClassifier(random_state = 12345, learning_rate = 0.1, max_depth = 5,  
min_samples_split = 0.1, n_estimators = 100, subsample = 1.0)))  
models.append(('LightGBM', LGBMClassifier(random_state = 12345, learning_rate = 0.01, max_depth = 3,  
n_estimators = 1000)))
```

```
# evaluate each model in turn
```

```
results = []  
names = []
```

In [82]:

```
for name, model in models:
```

```
    kfold = KFold(n_splits = 10, random_state = 12345)  
    cv_results = cross_val_score(model, X, y, cv = 10, scoring= "accuracy")  
    results.append(cv_results)  
    names.append(name)
```

```
msg = "%s: %f (%f)" % (name, cv_results.mean(), cv_results.std())
print(msg)
```

```
# boxplot algorithm comparison
```

```
fig = plt.figure(figsize=(15,10))
fig.suptitle('Algorithm Comparison')
ax = fig.add_subplot(111)
plt.boxplot(results)
ax.set_xticklabels(names)
plt.show()
```

```
lgbm = LGBMClassifier(random_state = 12345)
```

In [68]:

```
lgbm_params = {"learning_rate": [0.01, 0.03, 0.05, 0.1, 0.5],
               "n_estimators": [500, 1000, 1500],
               "max_depth": [3,5,8]}
```

In [69]:

```
gs_cv = GridSearchCV(lgbm,
                      lgbm_params,
                      cv = 10,
                      n_jobs = -1,
                      verbose = 2).fit(X, y)
```

```
gs_cv.best_params_
lgbm_tuned = LGBMClassifier(**gs_cv.best_params_).fit(X,y)
```

In [72]:

```
cross_val_score(lgbm_tuned, X, y, cv = 10).mean()
feature_imp = pd.Series(lgbm_tuned.feature_importances_,
                        index=X.columns).sort_values(ascending=False)
```

```
sns.barplot(x=feature_imp, y=feature_imp.index)
plt.xlabel('Significance Score Of Variables')
plt.ylabel('Variables')
plt.title("Variable Severity Levels")
plt.show()
```

```
xgb_tuned = GradientBoostingClassifier(**xgb_cv_model.best_params_).fit(X,y)
```

In [79]:

```
cross_val_score(xgb_tuned, X, y, cv = 10).mean()
feature_imp = pd.Series(xgb_tuned.feature_importances_,
                        index=X.columns).sort_values(ascending=False)
```

```
sns.barplot(x=feature_imp, y=feature_imp.index)
plt.xlabel('Significance Score Of Variables')
plt.ylabel('Variables')
plt.title("Variable Severity Levels")
plt.show()
```

```
models = []
```

```
models.append(('RF', RandomForestClassifier(random_state = 12345, max_depth = 8, max_features = 7,
min_samples_split = 2, n_estimators = 500)))
models.append(('XGB', GradientBoostingClassifier(random_state = 12345, learning_rate = 0.1, max_depth = 5,
min_samples_split = 0.1, n_estimators = 100, subsample = 1.0)))
models.append(("LightGBM", LGBMClassifier(random_state = 12345, learning_rate = 0.01, max_depth = 3,
n_estimators = 1000)))
```

```
# evaluate each model in turn
```

```
results = []
```

```
names = []
```

In [82]:

```
for name, model in models:
```

```
    kfold = KFold(n_splits = 10, random_state = 12345)
```

```
    cv_results = cross_val_score(model, X, y, cv = 10, scoring= "accuracy")
```

```
    results.append(cv_results)
```

```
    names.append(name)
```

```
    msg = "%s: %f (%f)" % (name, cv_results.mean(), cv_results.std())
```

```
    print(msg)
```

```
# boxplot algorithm comparison
```

```
fig = plt.figure(figsize=(15,10))
```

```
fig.suptitle('Algorithm Comparison')
```

```
ax = fig.add_subplot(111)
```

```
plt.boxplot(results)
```

```
ax.set_xticklabels(names)
```

```
plt.show()
```

CONCLUSION AND FUTURE WORK

One of the important real-world medical problems is the detection of diabetes at its early stage. In this study, systematic efforts are made in designing a system which results in the prediction of diabetes. During this work, five artificial intelligence classification algorithms are studied and evaluated on various measures. Experiments are performed on John Diabetes Database. Experimental results determine the adequacy of the designed system with an achieved accuracy of 99% using Decision Tree algorithm.

In future, the designed system with the used machine learning classification algorithms can be used to predict or diagnose other diseases. The work can be extended and improved for the automation of diabetes analysis including some other artificial intelligence algorithms.

THANK YOU