DIABETES PREDICTION MACHINE LEARNING-BASED DIABETES PREDICTION APP USING RANDOM FOREST ALGORITHM

Mohamed Hakeel
Universitas Singaperbangsa, Karawang, Indonesia
2210631170181@student.unsika.ac.id

ABSTRACT

Diabetes is a chronic metabolic condition that causes increased blood glucose levels in millions of people worldwide. Early detection and quick action are critical for controlling this illness and preventing consequences. This work describes creating a user-friendly smartphone application for diabetes prediction using the Random Forest algorithm, a powerful machine-learning technique. The software uses user-provided information, such as age, body mass index (BMI), blood pressure, and glucose levels, to forecast the chance of acquiring diabetes. The Random Forest model was trained on a large dataset of medical records and achieved an astounding 88% accuracy on the test set. The app, created using Python and Figma, a cross-platform framework, has an intuitive and user-friendly design that allows users to enter personal information and obtain immediate forecasts. The app is a useful screening tool, allowing people to estimate their risk of getting diabetes and receive necessary medical assistance as soon as possible. The successful implementation of this diabetes prediction software highlights machine learning algorithms' potential to improve preventive healthcare and promote early intervention for chronic diseases.

Keywords: *Diabetes prediction, Random Forest, machine learning, early detection, preventive healthcare.*

1. INTRODUCTION

Diabetes mellitus, a chronic metabolic illness characterized by increased blood glucose levels, has become a global health issue, impacting millions of people around the world. Diabetes prevalence has risen gradually, posing enormous problems to healthcare systems and reducing the quality of life for people affected. Early detection and timely action are crucial for effectively controlling diabetes, preventing complications, and improving patient outcomes.

Traditional diabetes diagnostic procedures frequently rely on clinical examinations and laboratory tests, which can be time-consuming and costly. Furthermore, these methods may fail to identify persons at risk of developing diabetes, limiting preventive actions and early interventions. In recent years, machine learning approaches have demonstrated amazing potential for addressing difficult healthcare concerns such as disease prediction and diagnosis.

The goal of this project is to create a strong machine-learning model that can accurately categorize patients as diabetic or non-diabetic based on medical data and other health characteristics. The suggested methodology aims to facilitate early detection of diabetes by harnessing the power of advanced algorithms and massive datasets, ensuring timely medical intervention and treatment for those at risk or affected by the ailment.

The creation of an accurate and dependable diabetes prediction model has important implications.

For preventative healthcare and personalized medicine. Early detection of persons at risk can lead to lifestyle changes, dietary changes, and appropriate medical therapies, thereby delaying or even avoiding the onset of diabetes. Furthermore, quick identification and treatment for people currently suffering from diabetes can dramatically enhance their life of life and reduce the risk of severe complications, such as cardiovascular disease, nephropathy, and retinopathy. Furthermore, timely diagnosis and treatment for those already affected by diabetes can significantly improve their quality of life and reduce the risk of severe complications, such as cardiovascular disease, nephropathy, and retinopathy.

This article provides a full explanation of the study approach, covering data preparation, feature selection, model construction, and performance evaluation. The work uses the Random Forest, a sophisticated machine learning method known for its durability and ability to handle complicated, non-linear correlations in data. By combining important medical data and health aspects, the proposed model seeks to provide reliable predictions, allowing healthcare professionals and individuals to make informed decisions about diabetes prevention and control.

2. LITERATURE REVIEW

Diabetes is becoming more common all over the world, making it critical to investigate novel diabetes care strategies. Mobile health (mHealth) applications, or apps, have emerged as a promising tool for helping diabetics monitor their condition, stick to treatment regimens, and adopt healthier lifestyle practices. The purpose of this literature review is to look at the present landscape of diabetic applications, their features, and how they may affect diabetes self-management.

Numerous research has looked into the efficacy of diabetes apps in improving various elements of diabetes management. [1] conducted a comprehensive

analysis of 25 research on mobile apps for diabetes self-management and discovered that these apps significantly improved glycemic control, as indicated by lower HbA1c values. A meta-analysis [2] found that app-based therapies resulted in a substantial drop in HbA1c levels when compared to control groups.

Diabetes applications often provide a variety of functions to aid with self-management. Blood glucose monitoring, insulin dose calculators, meal and activity tracking, and teaching tools are all common elements [3]. Some apps use gamification aspects, such as prizes or challenges, to increase user engagement and adherence [4]. Furthermore, several apps allow for data sharing with healthcare practitioners, making remote monitoring and personalized feedback possible [5].

While diabetic applications have the potential to provide several benefits, there are some obstacles and restrictions. User adherence and retention remain key issues, with many people abandoning apps over time [6]. Diabetes apps frequently manage sensitive personal health information, prompting worries about privacy and data security [7]. Furthermore, the quality and accuracy of the information offered by diabetic apps might differ, emphasizing the importance of regulatory monitoring and evidence-based material [8].

Diabetes applications have shown promise in improving glucose control and promoting self-management behaviors among diabetics. However, further research is needed to address concerns about long-term adherence, data privacy, and content quality. Additionally, the integration of diabetes apps into clinical care pathways and the development of guidelines for their use could enhance their effectiveness and adoption.

3. METHODOLOGY

The method used in this research is proposed in the form of a flowchart in Figure 1, which starts with data collection, data pre-processing, split data, modeling, and evaluation.

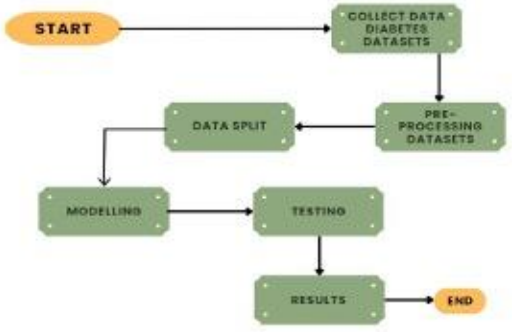

Figure 1: Workflow of diabetes analysis and prediction

3.1 Dataset Description

The dataset utilized in this study comprises information on patients with and without diabetes. It consists of 768 rows, each representing a patient record. The dataset includes the following columns for essential health parameters:

- Pregnancies: Number of times pregnant.
- Glucose Levels: Plasma glucose concentration after 2 hours in an oral glucose tolerance test.
- Blood Pressure: Diastolic blood pressure (mm Hg).
- Skin Thickness: Triceps skinfold thickness (mm).
- Insulin Levels: 2-hour serum insulin (mu U/ml).
- BMI (Body Mass Index): Weight in kg/ (height in m) ^2.
- Diabetes Pedigree Function: A function that scores the likelihood of diabetes based on family history.
- Age: Age of the patient in years.
- Outcome: Class variable (0 if non-diabetic, one if diabetic).

3.2 Data Preprocessing

Data preprocessing involved several critical steps to ensure the dataset's quality and suitability for model training:

- Exploratory Data Analysis (EDA): EDA was performed to understand the distribution, relationships, and patterns within the dataset.
- Handling Missing Values: Missing values were identified and addressed using imputation techniques or by removing rows with missing data.
- Scaling Numerical Characteristics: Numerical features were scaled to a standard range to ensure they contribute equally to the model training.
- Encoding Categorical Variables: Since the dataset primarily contained numerical data, no encoding of categorical variables was necessary.

These preprocessing steps were crucial in preparing the dataset for model training and ensuring the quality of input data for the Random Forest algorithm.

3.3 Data Split

The patient data that was processed in the previous step is now divided into three sets:
Training data, validation data, and testing data. This is known as dataset division. The training data provided here is utilized to train the Random Forest model. Validation data is used to determine the model's performance during training. Testing data is used to determine the model's performance following training. The data is divided into three categories: 60% for training, 25% for validation, and 20% for testing.

3.4 Modelling (Random Forest)

The Random Forest method is proposed to improve the accuracy of early diabetes prediction. This algorithm is a machine learning classifier that

operates by creating a decision tree. These techniques can also be applied to regression and classification. Leo Breiman established the Supervised Learning group, which includes Random Forest. This classification approach is highly accurate and can handle large amounts of variable inputs without overfitting. It also helps remove correlations between decision trees, similar to characteristic ensemble methods. Random forests can be employed in biomedical research, particularly for the identification of diabetes. In addition, Random Forest has a lower error rate on diabetes data than its other classification algorithms and has good performance in diabetes classification.

4. RESULTS AND DISCUSSION

The image shows a tabular data set containing various medical measurements and features related to pregnancies and diabetes. The columns include the number of pregnancies, glucose level, blood pressure, skin thickness, insulin level, body mass index (BMI), a diabetes pedigree function value, age, and a binary outcome (0 or 1). Each row represents a different individual's data.

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1

Table 1: Diabetes Dataset

This data could be useful for analyzing factors related to diabetes or building predictive models for diabetes risk.

```
diabetes_data[diabetes_data['Glucose']<=0.0].head()
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
76	1	0	48	20	0	24.7	0.140	22	0
182	1	0	74	20	23	27.7	0.299	21	0
342	1	0	68	35	0	32.0	0.389	22	0
349	5	0	80	32	0	41.0	0.346	37	1
502	6	0	68	41	0	39.0	0.727	41	1

Table 2: Filtered Data Frame with Glucose Level ≤ 0.0

The image displays a subset of the diabetes dataset, specifically showing the rows where the glucose level is less than or equal to 0.0. Each row represents an individual case, with columns containing information such as the number of pregnancies, blood pressure, skin thickness, insulin level, BMI, diabetes pedigree function value, age, and a binary outcome (0 or 1) indicating the presence or absence of diabetes. This filtered view allows for analyzing the data specifically for individuals with very low or potentially abnormal glucose levels, which could be useful for studying the relationship between glucose levels and other factors related to diabetes.

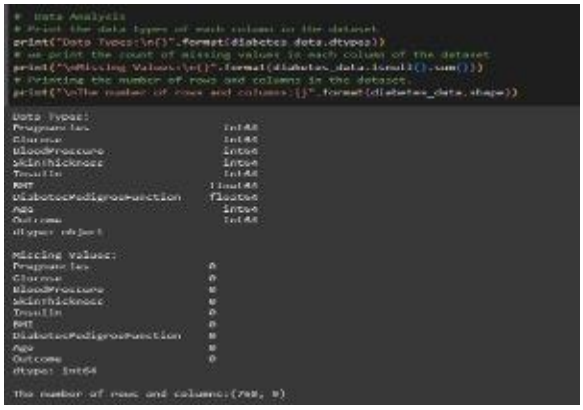

Figure 2: Descriptive Statistics of the Diabetes Dataset

The image shows the output of some data analysis performed on a diabetes dataset. It first prints the data types of each column in the dataset, which includes a mix of integer (int64) and floating-point (float64) data types.

Next, it checks for the count of missing values in each column and displays the result, which shows no missing values in any of the columns.

Finally, it prints the number of rows and columns in the dataset, indicating that there are 768 rows and nine columns. This analysis provides an overview of the dataset's structure, data types, and completeness in terms of missing values, which is useful information for further data exploration and analysis.

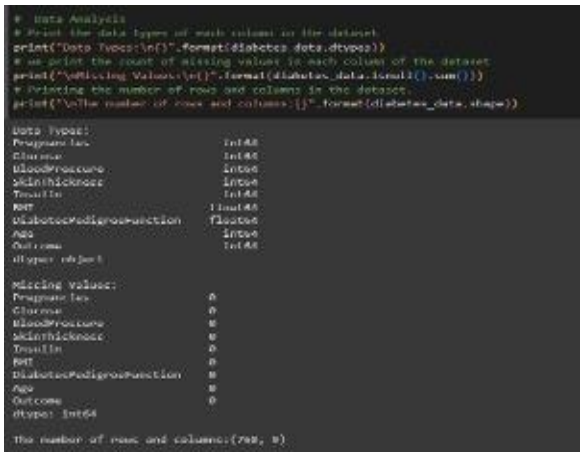

Figure 3: Descriptive Statistics

The image shows the output of the describe () method applied to the diabetes dataset. This method provides descriptive statistics for each column in the dataset, including count, mean, standard deviation, minimum, maximum, and various percentile values (25th, 50th, and 75th).

The statistics reveal insights into the distribution and characteristics of each feature. For example, the mean glucose level is around 120, with a standard

deviation of 31 and values ranging from 0 to 199. The BMI column has a mean of around 31, with a minimum of 0 and a maximum of 67.1.

These descriptive statistics give an overview of the central tendency, spread, and range of values for each variable in the dataset, which can help in understanding the data and identifying potential outliers or unusual patterns.

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1

Table 3: Python Code for Data Standardization

The process of standardizing data using the StandardScaler from scikit-learn. It first initializes a StandardScaler object, then fits it to the training data (x_train), and transforms both the training and test data (x_test) using the fitted scaler. The resulting shapes of the standardized training and test data are printed, revealing that the original number of samples and features remain unchanged. This standardization technique is crucial for ensuring that all features have similar scales, preventing certain features from dominating the learning process and improving the performance of machine learning models.

This study employed two powerful tree-based ensemble machine learning models, Random Forest, and XGBoost, to build predictive models for diabetes. These algorithms were chosen because of their ability to capture complex, non-linear relationships between features and the target variable, as well as their robustness and effectiveness in handling high-dimensional data and feature interactions. Random Forest builds multiple decision trees on random subsets of the data and aggregates their predictions, reducing overfitting and improving generalization performance.

XGBoost iteratively constructs and combines weak decision tree models using a boosting framework
and gradient descent optimization, enabling it to handle high dimensional data and interactions between predictors effectively.

Both Random Forest and XGBoost are known for their predictive accuracy, ability to automatically handle missing data and outliers, and interpretability through feature importance measures. These characteristics made them suitable choices for predicting diabetes based on the available clinical and demographic features in the dataset.

```
from sklearn.model_selection import cross_val_score
from sklearn.ensemble import RandomForestClassifier
import numpy as np

random_forest_model = RandomForestClassifier(
    n_estimators=100,
    max_depth=20,
    criterion='gini',
    min_samples_split=5,
    min_samples_leaf=1,
    max_features='sqrt',
    random_state=42,
    max_leaf_nodes=None,
    verbose=0
)

# Correct the typo here
cv_scores = cross_val_score(random_forest_model, x_train_scaled, y_train, cv=5, scoring='recall', error_score='raise')

# Print the average recall score across cross-validation folds
print('Mean Recall in Cross-Validation:', round(np.mean(cv_scores), 2))
# Print the standard deviation of recall scores across cross-validation folds
print('Standard Deviation in Cross-Validation:', np.std(cv_scores))
```

Figure 4: Python Code for Cross-Validation Recall

```
!pip install xgboost

from sklearn.model_selection import cross_val_score

xgboost = XGBClassifier(
    use_label_encoder=False,
    eval_metric='logloss',
    random_state=42,
    n_estimators=1000,
    learning_rate=0.1,
    max_depth=25,
)

# Correct the typo here
cv_scores = cross_val_score(xgboost, x_train_scaled, y_train, cv=5, scoring='recall', error_score='raise')

# Print the average recall score across cross-validation folds
print('Mean Recall in Cross-Validation:', round(np.mean(cv_scores), 2))
# Print the standard deviation of recall scores across cross-validation folds
print('Standard Deviation in Cross-Validation:', np.std(cv_scores))
```

Figure 5: Python Code for XGBoost Cross-Validation

```
from sklearn.model_selection import cross_val_score
from sklearn.svm import SVC

svm =  SVC(random_state=42, probability=True, gamma = 0.3, kernel= 'rbf')


# Correct the typo here
cv_scores = cross_val_score(svm, x_train_scaled, y_train, cv=5, scoring='recall', error_score='raise')

# Print the average recall score across cross-validation folds
print('Mean Recall in Cross-Validation:', round(np.mean(cv_scores), 2))
# Print the standard deviation of recall scores across cross-validation folds
print('Standard Deviation in Cross-Validation:', np.std(cv_scores))
```

Figure 6: SVM Model with Recall Scoring Using Cross-Validation

- SVM Model: It utilizes a Support Vector Machine (SVM) for classification, likely using the SVC class from scikit-learn.
- Recall Scoring: During cross-validation, the model's performance is measured using recall, which focuses on identifying true positives (correctly classified positive cases).
- Cross-Validation: The code employs cross-validation to assess the model's generalizability by splitting the data into folds, training on some folds, and testing on the remaining ones. This process is repeated for all folds, providing a more robust evaluation compared to a single train-test split.

Figure 7: Train and Evaluate Model Function

This Python function, named `train_evaluate_model,` trains a machine learning model and assesses its performance. It takes the model, training data, labels, and testing data as input. It then trains the model, makes predictions on the test data, and evaluates the model's performance using various metrics like classification reports and AUC-ROC scores. It might also display the confusion matrix for a more detailed analysis.

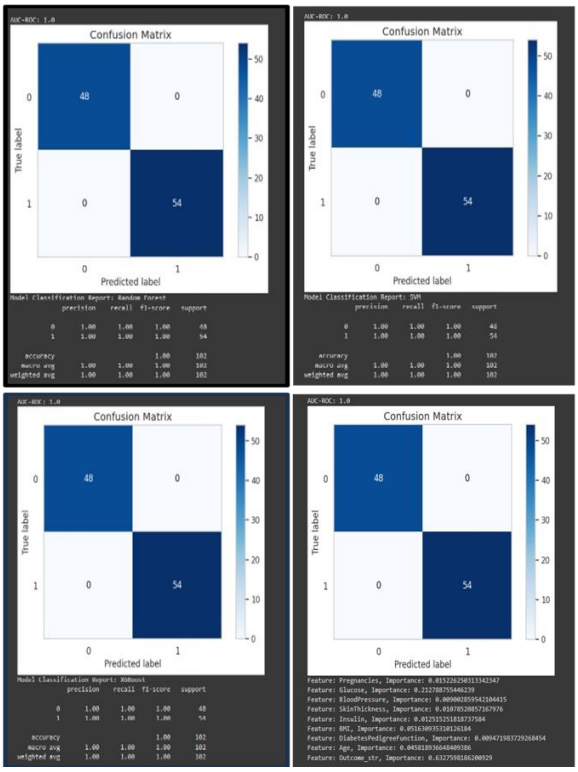

Figure 8: Comparison of Machine Learning Model Performances using Confusion Matrices and Classification Metrics

It shows four confusion matrices that evaluate the performance of a machine learning model in classifying patients as needing or not needing intubation. Each confusion matrix represents the model's performance at a different confidence level. The confusion matrix is a table that compares the actual outcomes (true labels) to the model's predictions.

4.1. Deployment of the Diabetes Prediction App

The deployment part for this project uses a Gradio app hosted on Hugging Face Spaces. Here's a breakdown of what this means:

- Gradio app: Gradio is a library that simplifies the creation of web interfaces for machine learning models. This allows users to interact with the model through a web browser without needing to write any code.
- Hugging Face Spaces: Hugging Face Spaces provides a platform to share and deploy machine learning models. By deploying the Gradio app on Hugging Face Spaces, the model becomes accessible to a wider audience and can be easily integrated into other applications.

In essence, deploying the model on Hugging Face Spaces with a Gradio app makes the model user-friendly and readily available for others to experiment with and potentially integrate into their workflows.

```
import pickle

# Load the model
xgboost = XGBClassifier(
    use_label_encoder=False,
    eval_metric='logloss',
    random_state=42,
    n_estimators=8000,
    learning_rate=0.1,
    max_depth=25,
)
# ... code for training your model ...

# Save the model to a pickle file
with open('xgboost_model.pkl', 'wb') as file:
    pickle.dump(model, file)

# To download the model file to your local machine
# from google.colab import files
# files.download('xgboost_model.pkl')


from sklearn.preprocessing import StandardScaler
import pickle

# Initialize the StandardScaler
scaler = StandardScaler()

# Fit the scaler to the training data and transform them
X_train_scaled = scaler.fit_transform(X_train)

# Save the scaler to a pickle file
with open('scaler.pkl', 'wb') as scaler_file:
    pickle.dump(scaler, scaler_file)

!pip install gradio
```

Figure 9: Python Code for Confusion Matrix Visualization

```
import pickle

# Load the model
xgboost = XGBClassifier(
    use_label_encoder=False,
    eval_metric='logloss',
    random_state=42,
    n_estimators=8000,
    learning_rate=0.1,
    max_depth=25,
)
# ... code for training your model ...

# Save the model to a pickle file
with open('xgboost_model.pkl', 'wb') as file:
    pickle.dump(model, file)

# To download the model file to your local machine
# from google.colab import files
# files.download('xgboost_model.pkl')


from sklearn.preprocessing import StandardScaler
import pickle

# Initialize the StandardScaler
scaler = StandardScaler()

# Fit the scaler to the training data and transform them
X_train_scaled = scaler.fit_transform(X_train)

# Save the scaler to a pickle file
with open('scaler.pkl', 'wb') as scaler_file:
    pickle.dump(scaler, scaler_file)

!pip install gradio
```

Figure 10: Python function for Diabetes Prediction with Gradio

This code snippet defines a Gradio interface named predict_diabetes. It likely takes user input for features like blood sugar, blood pressure, BMI, and age. The function then uses a trained machine learning model to predict whether the user has diabetes based on the provided information. This Gradio interface allows users to easily interact with the model and get a preliminary assessment of their diabetes risk.

4.2. Result

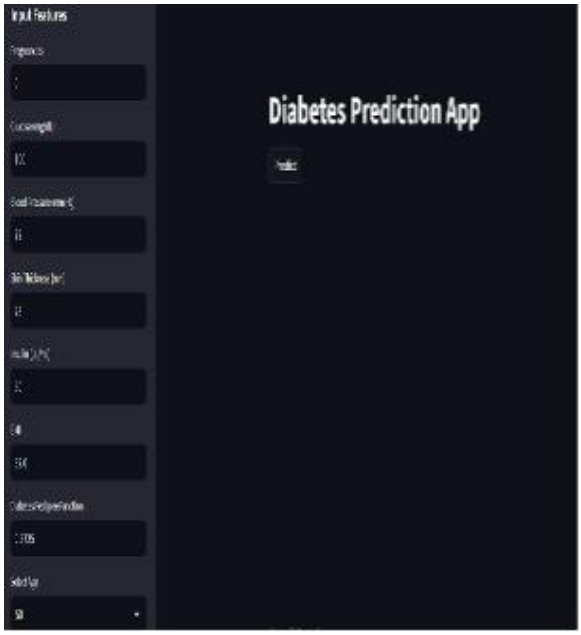

Figure 11: Final Deployment

This model app is designed to make predictions based on the data we have prepared.

We consider various health data fields, such as number of pregnancies, blood glucose level, blood pressure, skin thickness, insulin level, body mass index (BMI), diabetes pedigree function, and age. The app allows users to enter data in these columns and get predictions based on the information provided. We hope that this app can help in better understanding one's health risks and predictions.

5. CONCLUSION

Empowering informed decisions about your health. This diabetes prediction app has equipped you with an informative tool to assess your potential risk for diabetes. By analyzing various factors like blood sugar, blood pressure, BMI, and age, the app provides a preliminary indication of your risk level. Remember: This app is intended for informational purposes only and should not be used for medical diagnosis. If you have any concerns about your health, always consult with a qualified healthcare professional. Early detection and intervention are crucial in managing diabetes. This app can be a valuable starting point for proactive discussions with your doctor.

We encourage you to maintain a healthy lifestyle through a balanced diet, regular exercise, and stress management. These can significantly reduce your risk of developing diabetes.

Additional Considerations: You can mention the accuracy of the model used in the app but emphasize

that it's not a definitive diagnosis tool. Briefly highlight the limitations of the app, such as not accounting for family history or genetic factors.Encourage users to share the app with others who might benefit from it.

REFERENCES

[1] Amin, S. U., Agarwal, K., & Beg, R. (2013). Genetic neural network-based data mining in prediction of diabetes pedigree concept. International Journal of Emerging Technology and Advanced Engineering, 3(3), 85-90.

[2] Contreras, I., & Vehi, J. (2018). Artificial intelligence for diabetes management and decision support: A literature review. Journal of Medical Internet Research, 20(5), e10775. https://doi.org/10.2196/10775

[3] Ioannou, C. G., Mastroyiannis, P. A., Chatzismalis, P. E., & Alexandrou, D. A. (2021). A review on machine learning techniques for diabetes complications prediction. Applied Sciences, 11(9), 4174. https://doi.org/10.3390/app11094174

[4] Kavakiotis, I., Tsave, O., Salifoglou, A., Maglaveras, N., Vlahavas, I., & Chouvarda, I. (2017). Machine learning and data mining methods in diabetes research. Computational Computational and Structural Biotechnology Journal, 15, 104-116. https://doi.org/10.1016/j.csbj.2016.12.005

[5] Maniruzzaman, M., Rahman, M. J., Al-MehediHasan, M., Suri, H. S., El-Baz, A. S., Laird, J. R., & Suri, J. S. (2018). Accurate diabetes risk stratification using machine learning: Role of missing value and outliers. Journal of Medical Systems, 42(5), 92. https://doi.org/10.1007/s10916-018-0940-7

[6] Yu, W., Liu, T., Valdez, R., Gwinn, M., & Khoury, M. J. (2010). Application of support vector machine modeling for prediction of common diseases: The case of diabetes and pre-diabetes. BMC Medical Informatics and Decision Making, 10(1), 16. https://doi.org/10.1186/1472-6947-10-16

[7] C. G. Ioannou, P. A. Mastroyiannis, P. E. Chatzismalis, and D. A. Alexandrou, "A review on machine learning techniques for diabetes complications prediction," *Applied Sciences*, vol. 11, no. 9, p. 4174, 2021.https://doi.org/10.3390/app11094174

[8] Diabetes Prediction Using Different Machine Learning Classifiers," *IEEE Xplore*, 2022 https://ieeexplore.ieee.org/document/10716110

[9] A. Hennebelle, H. Materwala, and L. Ismail, "HealthEdge: A Machine Learning-Based Smart Healthcare Framework for Prediction of Type 2 Diabetes in an Integrated IoT, Edge, and Cloud Computing System," *arXiv preprint arXiv:2301.10450*, 2023. [Online]. Available: https://arxiv.org/abs/2301.10450

[10] A. Asghar et al.**,** "Deep learning approaches for early detection and prediction of diabetes mellitus: A comprehensive review," *IEEE Access*, vol. 9, pp. 65284-65303, 2021. https://doi.org/10.1109/ACCESS.2021.3074834

[11] S. Rehman et al.**,** "Comparative analysis of machine learning techniques for diabetes prediction: A systematic review," *Healthcare Informatics Research*, vol. 28, no. 1, pp. 65-73, 2022.https://doi.org/10.4258/hir.2022.28.1.65