

Diabetes Risk Prediction Using Classical Machine Learning Models and Flask-Based Web Deployment

1st Karthick Aerupula

*Dept. of Computer Science and Engineering(Data Science)
Geethanjali College of Engineering
and Technology
Hyderabad, India*

2nd Dacheppally Vybhavi

*Dept. of Computer Science and Engineering(Data Science)
Geethanjali College of Engineering
and Technology
Hyderabad, India*

3rd Donga Ravi Teja

*Dept. of Computer Science and Engineering (Data Science)
Geethanjali College of Engineering
and Technology
Hyderabad, India*

4th Gopalapuram Rajasri

*Asst. Professor
Dept. of Computer Science and Engineering (Data Science)
Geethanjali College of Engineering
and Technology
Hyderabad, India*

Abstract—Diabetes mellitus is a chronic metabolic disorder and a major public health concern worldwide. Early prediction of diabetes risk from routinely collected clinical attributes can support timely intervention and reduce long-term complications. This paper presents DiabetesML, a comparative study of classical machine learning models for diabetes risk prediction using the Pima Indians Diabetes Dataset. The system evaluates Logistic Regression, Decision Tree, and Random Forest classifiers on eight clinical features including pregnancies, glucose level, blood pressure, skin thickness, insulin, body mass index (BMI), diabetes pedigree function, and age. Logistic Regression achieves the highest test accuracy of 78.57%, while Random Forest and Decision Tree achieve 75.97% and 69.48%, respectively, on a stratified 80–20 train–test split. In addition to accuracy, models are assessed using precision, recall, F1-score, ROC-AUC, and confusion matrices to provide a comprehensive performance analysis. The best-performing model is deployed via a Flask-based web application with an HTML frontend, enabling end-users to submit clinical parameters and receive real-time diabetes risk predictions. The proposed system demonstrates the practicality of integrating classical machine learning with lightweight web technologies for accessible clinical decision support.

Index Terms—Diabetes prediction, Machine Learning, Logistic Regression, Random Forest, Decision Tree, Flask, Web Application

I. INTRODUCTION

Diabetes mellitus is one of the fastest-growing non-communicable diseases globally, affecting over 537 million adults as of 2021, a figure projected to rise significantly by 2045 [1]. The disease is associated with severe complications including cardiovascular disease, nephropathy, retinopathy, and neuropathy [2]. A significant proportion of individuals with diabetes remain undiagnosed, which delays treatment and increases the risk of complications. Early identification of individuals at high risk of developing diabetes is therefore a critical public health priority.

Traditional diagnostic methods rely on laboratory tests such as fasting plasma glucose and glycated hemoglobin (HbA1c). While these tests are accurate, they require clinical visits,

specialized equipment, and may not be feasible for frequent screening in resource-constrained settings [3]. In contrast, machine learning (ML) techniques can leverage routinely collected clinical variables to build predictive models that estimate a patient's risk of diabetes [4].

In this work, we develop DiabetesML, a machine learning-based system for diabetes risk prediction using the Pima Indians Diabetes Dataset [5]. The system compares three widely used classifiers—Logistic Regression, Decision Tree, and Random Forest—and deploys the best-performing model using a Flask-based web application. This approach demonstrates how classical ML methods, combined with simple web technologies, can provide an accessible and interpretable clinical decision support tool [6].

The key contributions of this work are as follows:

- A systematic comparison of three classical ML classifiers on the Pima Indians Diabetes Dataset using multiple evaluation metrics.
- Deployment of the best-performing model as a lightweight Flask web application with an HTML frontend for real-time risk prediction.
- Demonstration of a practical, low-infrastructure pipeline suitable for educational and community health settings.

The remainder of this paper is organized as follows. Section II reviews related work. Section III describes the dataset. Section IV presents the methodology. Section V reports results and discussion. Section VI concludes the paper.

II. RELATED WORK

The Pima Indians Diabetes Dataset [5] has been extensively used as a benchmark in diabetes prediction research. Early studies employed traditional statistical methods such as logistic regression to analyze the association between clinical features and diabetes onset [7]. Subsequent work introduced machine learning algorithms including Support Vector Machines (SVM), k-Nearest Neighbors (k-NN), and Artificial

Neural Networks (ANN), reporting varying levels of predictive performance [4].

Ensemble methods, particularly Random Forest and Gradient Boosting, have consistently demonstrated strong performance on structured medical datasets due to their ability to handle feature interactions and noisy data [8]. Several studies report that Random Forest offers a good trade-off between accuracy, robustness, and interpretability through feature importance scores [9]. Decision Trees are also popular for their intuitive, rule-based structure, though they are prone to overfitting without ensemble averaging [10].

Recent research has focused not only on offline evaluation of models but also on deploying predictive systems as web or mobile applications for real-time usage in clinical or community settings [6]. However, many such systems are complex to deploy or require substantial infrastructure. In contrast, this work emphasizes a lightweight deployment stack based on Flask and HTML, making it suitable for educational and small-scale clinical environments while retaining solid predictive performance.

III. DATASET DESCRIPTION

A. Pima Indians Diabetes Dataset

The experiments in this study use the Pima Indians Diabetes Dataset [5], which contains 768 records of female patients of Pima Indian heritage aged 21 years or older. Each record consists of eight input features and one binary target label indicating the presence (1) or absence (0) of diabetes.

The dataset used in the implementation is stored in `diabetes.csv` with the following columns:

- **Pregnancies:** Number of times pregnant.
- **Glucose:** Plasma glucose concentration.
- **BloodPressure:** Diastolic blood pressure (mm Hg).
- **SkinThickness:** Triceps skin fold thickness (mm).
- **Insulin:** Two-hour serum insulin (mu U/ml).
- **BMI:** Body Mass Index (weight in kg/(height in m)²).
- **DiabetesPedigreeFunction:** A function representing diabetes heredity.
- **Age:** Age in years.
- **Outcome:** Binary label (0 = No Diabetes, 1 = Diabetes).

The dataset comprises 768 instances and 9 columns (8 features + 1 target). The class distribution is imbalanced, with 500 non-diabetic cases and 268 diabetic cases, corresponding to approximately 65% negative and 35% positive samples.

IV. METHODOLOGY

A. Data Preprocessing

The dataset is loaded using the `pandas` library in Python [11]. The `Outcome` column is treated as the target variable, while the remaining eight columns constitute the input feature matrix. Since the dataset is numeric and does not contain categorical string variables, no explicit encoding is required.

The data is split into training and testing sets using an 80–20 ratio with stratified sampling to preserve the original

TABLE I
SUMMARY OF DATASET FEATURES

Feature	Type	Description
Pregnancies	Integer	Number of pregnancies
Glucose	Continuous	Plasma glucose concentration
BloodPressure	Continuous	Diastolic blood pressure (mm Hg)
SkinThickness	Integer	Triceps skin fold thickness (mm)
Insulin	Integer	Two-hour serum insulin (mu U/ml)
BMI	Continuous	Body mass index
DiabetesPedigreeFunction	Continuous	Diabetes pedigree function
Age	Integer	Age in years
Outcome	Binary	Diabetes status (0/1)

class distribution across splits [12]. A fixed random seed (`random_state = 1`) ensures reproducibility of the experimental results. Basic checks are performed to verify dataset dimensions and class balance.

B. Model Design

Three classifiers are implemented using `scikit-learn` [12]:

- **Logistic Regression:** A linear model suitable for binary classification [7]. The implementation uses `LogisticRegression` with a maximum of 1000 iterations to ensure convergence on this dataset.
- **Decision Tree:** A tree-based model implemented using `DecisionTreeClassifier` with a fixed random state [10]. Decision Trees partition the feature space into axis-aligned regions and are easy to interpret, but they can overfit if not regularized.
- **Random Forest:** An ensemble of Decision Trees implemented using `RandomForestClassifier` with 300 estimators and `random_state = 1` [8]. Random Forest reduces variance by averaging predictions of multiple trees trained on bootstrapped subsets of the data with feature subsampling.

Each model is trained on the training set and evaluated on the held-out test set. Probability scores (`predict_proba`) are used to compute ROC-AUC for each classifier [13].

C. Evaluation Metrics

To comprehensively assess model performance, the following metrics are computed on the test set [14]:

- **Accuracy:** Proportion of correctly classified instances.
- **Precision:** Proportion of true positives among predicted positives.
- **Recall (Sensitivity):** Proportion of true positives among actual positives.
- **F1-score:** Harmonic mean of precision and recall.
- **ROC-AUC:** Area under the Receiver Operating Characteristic curve [13].
- **Confusion Matrix:** Counts of true negatives (TN), false positives (FP), false negatives (FN), and true positives (TP).

These metrics allow a balanced comparison of models, especially under class imbalance where accuracy alone can be misleading [14].

D. System Architecture and Deployment

The trained Logistic Regression model is serialized using `pickle` and stored as `model.pkl`. A lightweight Flask web application [15] acts as the backend interface to this model. The application defines two primary routes:

- The root route (`/`) renders the home page (`index.html`).
- The `/predict` route accepts form submissions via POST, extracts feature values, and passes them to the loaded model to generate predictions.

The HTML frontend (`index.html`) presents a simple form with numeric input fields for all eight clinical parameters: Pregnancies, Glucose, Blood Pressure, Skin Thickness, Insulin, BMI, Diabetes Pedigree Function, and Age. Upon submission, the form sends the data to the Flask backend, which returns a prediction message such as “High Risk of Diabetes” or “Low Risk of Diabetes”. The result is dynamically rendered on the same page, providing an interactive user experience.

V. RESULTS AND DISCUSSION

A. Overall Performance

All three models are trained on the same training data and evaluated on the same test data. Table II summarizes the performance metrics.

TABLE II
PERFORMANCE OF ML MODELS ON TEST SET

Model	Acc.	Prec.	Rec.	F1	AUC
Logistic Reg.	78.57%	78.38%	53.70%	63.74%	0.815
Random Forest	75.97%	70.73%	53.70%	61.05%	0.815
Decision Tree	69.48%	57.14%	51.85%	54.37%	0.654

Logistic Regression achieves the highest accuracy (78.57%) and F1-score, indicating that a simple linear decision boundary performs competitively on this dataset [7]. Random Forest slightly lags behind in accuracy but offers comparable ROC-AUC (0.815), suggesting strong ranking capability even when the hard classification threshold is fixed [8]. The Decision Tree classifier shows the lowest accuracy and ROC-AUC, which aligns with its tendency to overfit without ensemble averaging [10].

B. Confusion Matrix Analysis

The confusion matrices for the three models provide deeper insight into their error patterns. Table III reports the confusion matrix entries (TN, FP, FN, TP) for each classifier on the test set.

TABLE III
CONFUSION MATRICES OF MODELS

Model	TN	FP	FN	TP
Logistic Reg.	92	8	25	29
Random Forest	88	12	25	29
Decision Tree	79	21	26	28

Logistic Regression yields the highest true negative count while maintaining the same number of true positives as Random Forest. This balance between correctly identifying non-diabetic and diabetic cases explains its superior overall accuracy. Random Forest produces slightly more false positives, leading to marginally lower accuracy but still competitive performance. The Decision Tree incurs more false positives and slightly more false negatives, which degrades both its accuracy and F1-score [14].

From a practical standpoint, the choice between Logistic Regression and Random Forest may depend on interpretability and deployment preferences. Logistic Regression offers more straightforward interpretation of feature coefficients, whereas Random Forest can provide feature importance rankings and may handle non-linear relationships better [8].

C. System Usability

The Flask-based deployment and HTML frontend make the system accessible to non-technical users [15]. Health practitioners or patients can input clinical parameters via a simple web form and receive an immediate risk assessment without needing to interact with Python scripts or command-line tools.

The architecture is lightweight enough to run on a local machine or a low-cost cloud server, making it suitable for educational demonstrations, small clinics, or community health screenings [6]. Additional features, such as input validation, logging, and multi-language support, can be integrated without major architectural changes.

VI. CONCLUSION

This paper presented DiabetesML, a classical machine learning-based system for diabetes risk prediction using the Pima Indians Diabetes Dataset [5] and a Flask-powered web interface [15]. Three models—Logistic Regression, Random Forest, and Decision Tree—were implemented and evaluated using multiple performance metrics. Logistic Regression achieved the best overall test performance, with an accuracy of 78.57% and competitive ROC-AUC, while Random Forest provided similar discriminative power with slightly lower accuracy.

The integration of a trained model into a Flask web application with an HTML frontend demonstrates how predictive analytics can be brought closer to end-users through simple yet effective deployment strategies. Future work may include hyperparameter tuning, cross-validation, feature engineering (e.g., handling outliers and missing values), comparison with more advanced ensemble methods such as Gradient Boosting [9], and deployment on cloud platforms or mobile applications. Additionally, extending the system to support explanations using SHAP values [16] could improve transparency and trust for clinical stakeholders.

REFERENCES

- [1] International Diabetes Federation, “IDF Diabetes Atlas, 10th edition,” 2021. [Online]. Available: <https://www.diabetesatlas.org>

- [2] World Health Organization, "Diabetes," WHO Fact Sheets, 2023. [Online]. Available: <https://www.who.int/news-room/fact-sheets/detail/diabetes>
- [3] N. H. Cho, J. E. Shaw, S. Karuranga, Y. Huang, J. D. da Rocha Fernandes, A. W. Ohlrogge, and B. Malanda, "IDF Diabetes Atlas: Global estimates of diabetes prevalence for 2017 and projections for 2045," *Diabetes Research and Clinical Practice*, vol. 138, pp. 271–281, 2018.
- [4] I. Kavakiotis, O. Tsave, A. Salifoglou, N. Maglaveras, I. Vlahavas, and I. Chouvarda, "Machine learning and data mining methods in diabetes research," *Computational and Structural Biotechnology Journal*, vol. 15, pp. 104–116, 2017.
- [5] J. W. Smith, J. E. Everhart, W. C. Dickson, W. C. Knowler, and R. S. Johannes, "Using the ADAP learning algorithm to forecast the onset of diabetes mellitus," *Proceedings of the Annual Symposium on Computer Application in Medical Care*, pp. 261–265, 1988.
- [6] Z. Obermeyer and E. J. Emanuel, "Predicting the future — big data, machine learning, and clinical medicine," *New England Journal of Medicine*, vol. 375, no. 13, pp. 1216–1219, 2016.
- [7] D. W. Hosmer, S. Lemeshow, and R. X. Sturdivant, *Applied Logistic Regression*, 3rd ed. Hoboken, NJ: John Wiley & Sons, 2013.
- [8] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.
- [9] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016, pp. 785–794.
- [10] J. R. Quinlan, "Induction of decision trees," *Machine Learning*, vol. 1, no. 1, pp. 81–106, 1986.
- [11] W. McKinney, "Data structures for statistical computing in Python," in *Proceedings of the 9th Python in Science Conference*, vol. 445, 2010, pp. 51–56.
- [12] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg *et al.*, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.
- [13] T. Fawcett, "An introduction to ROC analysis," *Pattern Recognition Letters*, vol. 27, no. 8, pp. 861–874, 2006.
- [14] D. M. W. Powers, "Evaluation: From precision, recall and F-measure to ROC, informedness, markedness and correlation," *Journal of Machine Learning Technologies*, vol. 2, no. 1, pp. 37–63, 2011.
- [15] M. Grinberg, *Flask Web Development: Developing Web Applications with Python*, 2nd ed. Sebastopol, CA: O'Reilly Media, 2018.
- [16] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," in *Advances in Neural Information Processing Systems*, vol. 30, 2017, pp. 4765–4774.