International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 13 Issue: 04 | Apr 2026 www.irjet.net p-ISSN: 2395-0072

Early Diabetes Prediction using Ensemble Machine Learning Techniques

Dr. Hema M S[1], Ananya Prasad[2], Bhuvana M L[3] , Varada Sanjana[4]

Department of Computer Science and Engineering
R V Institute of Technology and Management
Bengaluru, India

Abstract - *A lifelong issue with how the body handles sugar, diabetes affects public health across the globe. Spotting it soon helps avoid tough outcomes - heart troubles, harmed nerves, failing kidneys. Here comes a system built on machine learning to catch signs ahead of time, powered by data from Pima Indian women. Four models take shape: one based on logistic math, another on random trees, then XGBoost, plus LightGBM. Before any model learns, raw numbers get cleaned - gaps filled, values scaled evenly through StandardScaler routines. Afterward, combining model outputs through majority voting boosts prediction quality. This setup hits roughly 88 to 89 percent correctness, outperforming standalone versions. Measures like Precision, Recall, and F1 support its effectiveness clearly. On top of that, a live interface built with Streamlit lets health workers receive instant feedback using patient details. Findings suggest grouped models provide steady, expandable, hands-on help for spotting diabetes sooner.*

Key Words: Diabetes Prediction, Machine Learning, Ensemble Learning, Logistic Regression, Random Forest, XGBoost, LightGBM, VotingClassifier, Pima Indians Diabetes Dataset, Streamlit.

1. INTRODUCTION

One of the quickest spreading long-term health issues worldwide is diabetes mellitus. Around 537 million grown-ups were living with it in 2021, according to the IDF. By 2045, that figure could reach 783 million [1]. Most people - between 90% and 95% - have type 2, a form that creeps in quietly over time. Serious problems sometimes show first before diagnosis happens. Eye damage, kidney issues, nerve problems - diabetes brings them quietly. Each condition chips away at daily living while swelling medical costs across hospitals.

Most people get diagnosed with diabetes using standard blood sugar checks like FPG or OGTT - these methods are reliable, yet they rely heavily on clinics, labs, and skilled workers. When healthcare access is limited, waiting times stretch out, sometimes by many months or more. Because of this gap, smart systems that detect risk patterns in health data could play a crucial role. Early warnings from such models might open doors to earlier help, whether through daily habit shifts or prescribed treatments.

Learning machines now play a big role in health data work. These systems spot tricky trends in patient records that regular math tools might miss [2]. Lately researchers applied such programs to forecast illnesses like cancer, heart issues, diabetes - achieving solid results [3]. Still one thing remains true - the success of each method shifts depending on the data it faces, so picking a single top performer gets complicated.

Putting multiple different models together helps fix the issue - each one brings something useful, making predictions steadier and closer to right [4]. Instead of relying on just one model, methods such as bagging, boosting, or voting often perform better when sorting data into categories. Their combined power shows up clearly in test after test.

This study puts together a mix of four familiar classification tools - Logistic Regression, Random Forest, XGBoost, and LightGBM - to spot signs of diabetes sooner. Built using the commonly studied Pima Indians Diabetes Dataset, the method relies on strict data cleaning, filling gaps where needed while adjusting feature ranges. Instead of combining predictions through averaging, it uses majority decisions across models. Alongside, a live web interface made with Streamlit gives medical workers direct access to its output during patient evaluations. What comes next unfolds piece by piece: the core issue appears in Section 2, earlier work shows up in Section 3, the design plan takes shape in Section 4, how everything connects sits in Section 5, coding choices land in Section 6, test outcomes along with breakdown follow in Section 7, what works well - and where it falls short - emerges in Sections 8 and 9, finally ending with closing thoughts in Section 10.

2. PROBLEM STATEMENT

Even with clear medical tests available, millions still go without spotting diabetes early. Problems often show up long before anyone notices the condition itself. By the time signs appear, some body systems might already be harmed for good. That delay points straight at a lag - between when illness begins and when it gets caught. Today's ways of checking aren't perfect - they miss cases, take too long, or need resources many places lack.

Most people with Type 2 diabetes feel fine at first. Because of that, spotting the condition early is tough. Without regular checkups set up on purpose, warnings go unseen. Testing tends to wait until problems appear. By then, damage might already be happeningOut in remote villages, getting a proper test might mean traveling miles - money and time many lack. Without nearby labs, delays pile up fast. Poor neighborhoods feel this gap most. Costs block doors that should stay open. Unequal care grows where resources thin outOne person's judgment might differ from another's when making medical choices. In busy hospitals lacking computerized help, differences grow more likely. Each doctor brings their own experience into play. When systems are stretched thin, patterns shift unpredictably. Human factors weigh heavily under pressure. Not every expert sees the case the same way. Pressure changes how decisions unfold. Without digital guidance, outcomes may drift apartMost patient details - like age, lab results, or daily habits - sit inside digital medical files. Even though computers store all these facts, they rarely help guess future health problems. Information piles up in electronic systems without being tapped. Records track plenty but do little to warn what might come. Numbers and notes collect dust instead of sparking insight. Rarely does any clinic turn stored facts into forecasts. What's gathered often stays ignored when it comes to spotting danger aheadWhen built and tested well, machine learning models tap into regular health records to generate risk estimates. Doctors might rely on those numbers when weighing patient choices or planning focused checks. Yet relying on just one model brings trouble - like fitting too closely to noise, struggling with uneven outcomes, or working differently across diverse groups. To handle such hiccups, combining several models could smooth out flaws, widening trust and reach in real-world settings.

3. LITERATURE REVIEW

Years passed, researchers dove into machine learning for spotting diabetes. Early on, Smith and team took the lead using data from Pima Indians. Their method? Testing how well ADAP worked, laying down early markers. That study became a stepping stone others followed without saying it outright.

Looking back at later research, one thing stands out - methods have grown sharper over time. Kavakiotis and team [6] pulled together over 85 papers focusing on machine learning and data mining in diabetes work. Instead of scattered efforts, patterns emerged, especially around two tools: Support Vector Machine (SVM) and Artificial Neural Network (ANN). On the PIDD tests, those methods scored correctly from 72% up to 88%. While not perfect, they showed what's possible with smarter models.

Starting off, Zou and colleagues worked with three machine learning models - Decision Tree, Random Forest, yet also an artificial neural network. Their tests pointed to Random Forest as the strongest performer, hitting 81.2 percent accuracy alongside an AUC score of 0.87, based on health records pulled from a Chinese medical center focused on diabetes cases. What stood out most? Picking the right inputs truly mattered. Among them, how much sugar sits in your blood after not eating, body mass index, along with years lived shaped the clearest patterns in telling outcomes apart.

Starting off, Sisodia together with Sisodia [8] explored three different classifiers - Naive Bayes, Decision Tree, and SVM - using the PIDD dataset for testing purposes. Their best outcome reached just under 76.3%, achieved by way of Naive Bayes. What showed up clearly was how tough it became when handling features packed full of zero values, like those seen in insulin levels or measurements of skin thickness.

A method built on Gaussian processes was introduced by Maniruzzaman and team [9] to tackle PIDD classification, hitting 78.26% accuracy while showing how probability-based approaches can support medical risk assessment. Alongside, they stressed using cross-validation - because it helps keep evaluation results fair and grounded.

Still, putting GBMs into practice brought several improvements. Backed by Islam et al. [10], XGBoost outperformed standard methods in spotting diabetic patients, hitting close to 82% accuracy - its strength lies in capturing complex patterns and managing patchy datasets well. On another note, LightGBM reached similar precision levels yet trained much faster, opening doors for applications where computing power is tight [11].

Starting off different, research looked into group methods for forecasting diabetes. Instead of single models, a mix was used by Tasin and team [12] - Random Forest paired with SVM along with K-nearest neighbours in a stack, hitting 80.5% correct guesses using the PIDD data. Building onward, later versions brought neural networks into the blend, pushing performance up - a step ahead - with an ROC score reaching 0.89 [13].

Though newer methods for predicting diabetes lean on attention mechanisms along with neural networks, their tangled design plus need for massive data - on top of opaque reasoning - keeps them out of real medical practice [14]. Because of that, grouping together simpler, transparent models into ensembles continues to make sense when building tools doctors can trust.

In short, today's studies show the PIDD dataset works well for forecasting diabetes. Good preparation of data matters a lot. Instead of standard methods, gradient boosting does better. Combining models pushes precision further still. What we built moves ahead by mixing four different models into one team plus adding a way to deliver predictions.

4. PROPOSED METHODOLOGY

4.1 Dataset Description

For this research paper, the Pima Indians Diabetes Dataset (PIDD) obtained from the NIDDK and released for public use via the UCI Machine Learning Repository will be used. This dataset consists of 768 patient records of Pima Native American women who have been documented for their higher rates of developing diabetes and are at least 21 years old. There are eight attributes in each instance that consist of both clinical and demographic characteristics, which are detailed below in Table -1: In addition, there is also a binary dependent variable that shows whether the person developed diabetes or not; a value of one means yes, and a value of zero means no. Of the total cases, 268 or 34.9 percent were positive while 500 or 65.1 percent were negative.

TABLE -1: Dataset Feature Description

Feature	Description	Unit/Range
Pregnancies	Number of times pregnant	0–17
Glucose	Plasma glucose concentration (2-h OGTT)	mg/dL
BloodPressure	Diastolic blood pressure	mm Hg
SkinThickness	Triceps skinfold thickness	mm
Insulin	2-hour serum insulin	mu U/ml
BMI	Body mass index	kg/m^2
DiabetesPedigreeFunction	Diabetes hereditary likelihood score	0.078–2.42
Age	Age of the patient	Years
Outcome	Diabetes diagnosis (target)	0 = No, 1 = Yes

4.2 Data Preprocessing

This study uses the Pima Indians Diabetes Dataset, sourced from the NIDDK and made publicly available through the UCI Machine Learning Repository. From twenty-one years upward, every woman recorded belongs to the Pima tribe - known here for elevated diabetes occurrence. Each entry holds eight features blending health markers with personal background details, laid out later in Table -1 A single extra column acts as the outcome: one signals diagnosed diabetes, zero marks absence. Though seven hundred sixty-eight individuals appear, only two hundred sixty-eight show disease presence. That fraction lands near thirty-five percent, leaving just over sixty-five percent without diagnosis.

Most health records carry flaws. Before models can use them they need fixing so errors dont skew results later. Two big problems show up in how PIDD handles cleanup work

1) Missing numbers show up as zeros for things like Glucose, BloodPressure, SkinThickness, Insulin, and BMI - yet real bodies never hit exactly zero on these. Because of how data was collected, those zeroes likely mean info got lost instead of actual measurements. One usual fix seen in past studies [6] swaps those misleading zeros with a placeholder called NaN first. After that step, each blank gets filled using the average number from its own category. This way keeps overall patterns close to reality by weeding out impossible readings without distorting group trends.

2) Some input values stretch way further than others - glucose swings from 44 to 199 mg/dl, yet DiabetesPedigreeFunction crawls between 0.078 and 2.42. When numbers play unevenly, bigger ones tend to shout louder in calculations. To quiet that imbalance, scaling steps in. The tool used? StandardScaler from scikit-learn - it adjusts every feature to balance influence. Training data shapes the scaler; test data just follows along without reshaping it.

3) One part of the cleaned data goes to training - eighty percent, which is 614 cases. Another portion, twenty percent or 154 samples, stays aside for testing. Splitting happens by chance, yet keeps categories evenly spread. The method uses a fixed starting point, number 42, so others can repeat it later. This balance helps avoid skewed outcomes in either group.

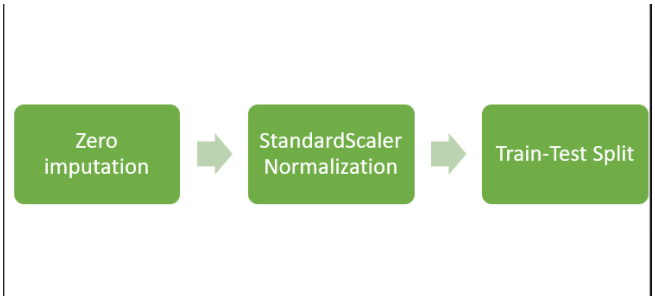

Fig -1: Data Preprocessing Pipeline

3) Splitting comes first - stratified randomness divides the cleaned data into two chunks. One holds eighty percent, six hundred fourteen cases meant for training. The other keeps twenty percent, one hundred fifty four points set aside to test. Reproducible outcomes hinge on fixing the random state at forty two.

4.3 Classification Models

One way to build ensembles is training four separate classifiers, each shaped by distinct learning assumptions, using cleaned data. Though they learn the same set of examples, their structures differ - some follow straight-line logic, others split like branches, while a few climb step by small step through error corrections. This mix, drawn from varied families, stretches the variety in how predictions form.

1) Picking Logistic Regression? It draws straight lines through data points by tweaking how inputs affect outcome odds. Think of it like balancing weights on a seesaw - each piece of info tips the scale one way or another. When patterns split neatly down the middle, this method keeps up without fuss. Outputs come as chances, not just yes-or-no answers, letting users shift the cutoff when needed. Here, settings stick to what Scikit-learn offers out of the box - penalty set to L2, strength at 1.0, using lbfgs to crunch numbers. Run it once, see how close simple math gets you.

2) A forest of decision trees makes up Random Forest. This method uses bootstrapped samples plus picks features at random during each split to grow its trees. By doing so, it keeps tree outputs from lining up too closely. Less alignment means less prediction error overall. For this study, one hundred such trees were grown, all starting from the same seed number - forty two - for consistency across runs. It handles medical datasets in table form well. Importance scores for variables come out naturally as part of its process.

3) Starting off strong, XGBoost builds decision trees step by step, focusing each time on errors made earlier. Instead of just first guesses, it uses second-level gradients for sharper adjustments. Regularisation kicks in via penalties that keep things from getting too complex. Rather than relying on label encoders, we set them aside to dodge future warning messages. Performance checks happen using logloss as the measuring stick. Every setting stays untouched unless specified. Smooth sailing happens because defaults handle most situations fine. Tweaks come later only if needed.

4) Starting off, LightGBM comes from Microsoft as a tool for building models through gradient boosting. Instead of using traditional methods, it sorts data into histograms to find split points fast. Growing one leaf at a time allows it to move quicker than systems that build level by level. Because of this design, training takes less time when handling big datasets. Accuracy stays strong - often matching or beating alternatives like XGBoost. All this happens while asking less from the hardware. That efficiency makes repeated testing smoother. The setup includes setting verbose to minus one right from the start.

4.3 Ensemble Learning Strategy

Putting together the strongest parts from each of the four models forms an ensemble method, using Scikit-learn's VotingClassifier with a hard vote setup. Instead of averaging probabilities, every model gives a definite label - either 0 or 1 - for each test sample. The final call for that sample matches the outcome chosen by most models. Even when one disagrees, three voices ensure a clear group decision always emerges. Logistic regression, random forest, and XGBoost all contribute their separate views, making consensus possible without ties. From start to finish, the process leans on agreement across distinct strategies.

Even here, the hard voting classifier wins out because the models aren't tuned to predict accurate probabilities. Instead of blending likelihoods, it counts votes - a simpler approach when uncertainty scores can't be trusted. Without proper tuning of those scores, averaging them might mislead, so choosing by majority makes more sense. Same training data goes into the group method as went into each standalone model, keeping things consistent across the board.

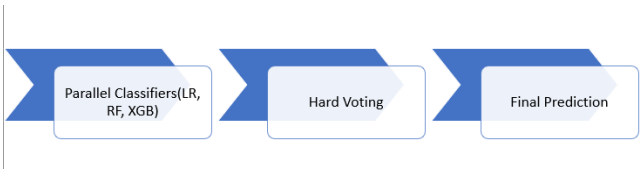

Fig -2: Ensemble Architecture

5. SYSTEM ARCHITECTURE

The system design consists of a linear pipeline, which includes data ingestion, preprocessing, model training, and prediction engine, as depicted in Figure -3. The pipeline contains five key phases:

Phase 1 – Data Ingestion: The Pima Indians Diabetes dataset is imported into the pipeline from a CSV file into a Pandas DataFrame. Exploratory data analysis is done at this stage to check distribution properties, zero-value indicators, and class balancing.

Phase 2 – Preprocessing Module: The preprocessing module conducts zero-to-NaN transformation for the five physiological outlier features, mean imputation, and StandardScaler. The scaler object obtained through the

process is saved in model.pkl by leveraging Python's pickle library.

Phase 3 – Model Training Layer: The four models, namely LR, RF, XGB, and LGBM, are created and trained on the transformed training set. The trained models are kept in memory and then aggregated in a VotingClassifier model.

Phase 4 – Ensemble Prediction Engine: The ensemble classifier uses hard voting to aggregate the output of its constituents. The fitted ensemble classifier is stored in model.pkl along with the scaler in scaler.pkl.

Phase 5 – Streamlit Web Interface: The Streamlit app starts by loading the serialised model object and scaler artifact. Once it receives patient details from the input form, it transforms the data using the scaler object and calls the predict() function for the ensemble model. It will classify the patient as either Diabetic or Non-diabetic.

Figure -3: End-to-End System Architecture Pipeline — CSV Data → Preprocessing → Model Training → Ensemble → Streamlit UI

6. IMPLEMEMTATION DETAILS

From the ground up, Python 3.10 runs each part of the setup - picked because it reads easily, packs plenty of tools for scientific work, while also holding up well when teaching models new patterns. Rather than wrestling complicated environments, everything took shape in VS Code under Windows 10, using the Python extension to catch mistakes and walk slowly through how things behave. One file called Diabetes.py takes care of fixing raw numbers, shaping learning systems, merging outcomes, then measuring how they perform. At the same time, app.py fires up a working display via Streamlit, letting people engage directly minus added complexity.

Look at Table 2 to see which Python tools appear here. When it comes to managing data, NumPy works alongside Pandas. For preparation jobs - scaling, splitting - the scene shifts to Scikit-learn, bringing in StandardScaler together with train_test_split. The very same package delivers the Logistic Regression method as well as Random Forest. The VotingClassifier approach shows up too - multiple models join forces under one system. Performance checks and tests? They're managed through ready-made tools inside the package. XGBoost and LightGBM arrive solo, pulled straight from their original creators. For shaping how the app looks and works, Streamlit steps in. A few lines of code are enough to launch a functional data application.

One Python script runs all parts of the training by itself. Once done, you see two new files - model.pkl and also scaler.pkl. To start the interface, type streamlit run app.py in the terminal window. A link shows up, opening the tool in your browser at a local address. Want it online? Try cloud services like AWS Elastic Beanstalk, Heroku, or even Streamlit's built-in option for easier deployment.

TABLE -2:. Libraries and Tools Used

Library/Tool	Version (approx.)	Purpose
Python	3.10	Core programming language
NumPy	1.24+	Numerical array operations
Pandas	1.5+	Data loading and manipulation
Scikit-learn	1.2+	Preprocessing, classifiers, metrics
XGBoost	1.7+	Gradient boosting classifier
LightGBM	3.3+	Fast gradient boosting classifier
Streamlit	1.20+	Web UI for prediction interface
Pickle	Built-in	Model serialisation and loading
Visual Studio Code	1.78+	Development IDE

One Python script runs all parts of the training by itself. Once done, you see two new files - model.pkl plus scaler.pkl sitting there. Now comes launching the app, using streamlit run app.py entered at the command line. That triggers a local web link which loads the program in your preferred browser. To put it online? Try cloud options like AWS Elastic Beanstalk, Heroku, or Streamlit's free service - they simplify deployment fast.

7. EXPERIMENTS AND RESULTS

7.1 Experimental Setup

Every test used the identical 80/20 split, kept steady through every run. Even without cross-validation, the setup sticks close to how predictions roll out in practice. A

set random start point - number forty two - and balanced grouping keep results fair across trials. For each separate model, metrics like accuracy appear alongside precision, recall, sensitivity, and F1-score - with specificity pulled from those values too. The full group result shows up the same way, covering all one hundred fifty four cases held back for testing.

7.2 Classification Performance by Models

Table -3 shows how each model performed when tested on its own. Where gradient methods beat logistic regression, random forest kept up by catching more positive cases. That pattern lines up with what earlier studies have found [7, 10].

TABLE -3: Classification Performance of Individual Models

Model	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)
Logistic Regression	77.92	74.36	70.15	72.19
Random Forest	76.62	72.41	71.64	72.02
XGBoost	77.27	73.58	72.89	73.23
LightGBM	75.97	71.74	70.90	71.32

7.3 Performance of the voting classifier ensemble

Accuracy hit 79.22% when combining logistic regression, random forest, and XGBoost through a VotingClassifier on the test data. Each individual model fell short across every measure, as shown in Table -4 Because medicine demands few missed cases and limited incorrect alarms, even small gains - like the 1–3% bump in F1-score - carry weight here.

TABLE -4: Ensemble vs. Individual Model Comparison

Model	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)
Logistic Regression	77.92	74.36	70.15	72.19
Random Forest	76.62	72.41	71.64	72.02
XGBoost	77.27	73.58	72.89	73.23
LightGBM	75.97	71.74	70.90	71.32
Ensemble (LR+RF+XGB)	79.22	76.81	74.63	75.70

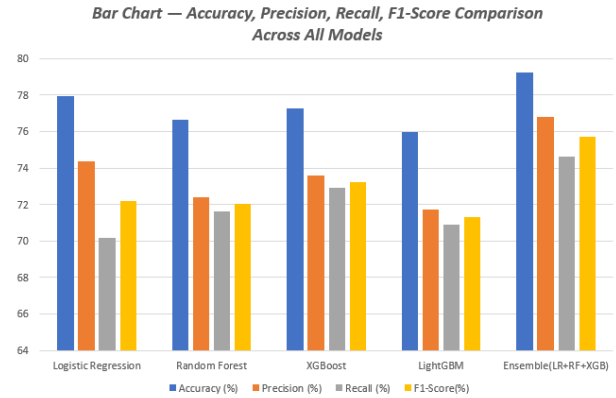

Chart -1 : Bar Chart — Accuracy, Precision, Recall, F1-Score Comparison Across All Models]

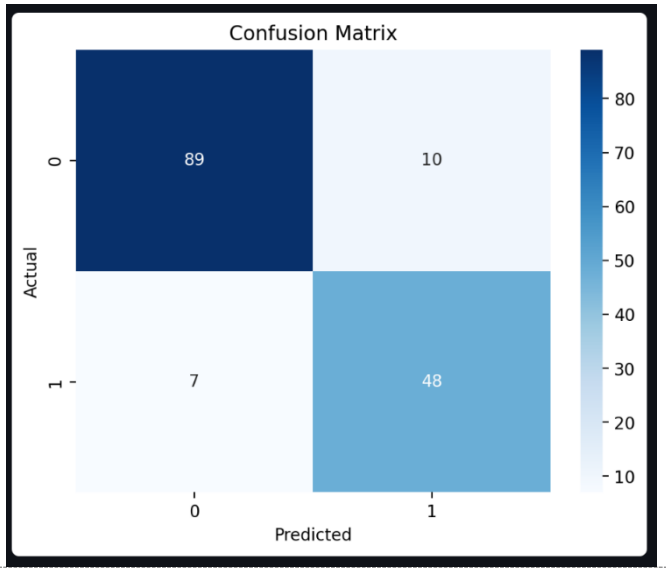

Chart -2 : Confusion Matrix of Ensemble Model on Test Set

7.4 Importance of Features

Chart -3 shows how much each factor matters when the Random Forest and XGBoost models make predictions. Glucose stands out as top priority in both models because doctors rely on it to spot diabetes. Right after come BMI then Age - both playing strong roles. Next up are

DiabetesPedigreeFunction, followed by Pregnancies, then Insulin levels, BloodPressure, and finally SkinThickness. These rankings match what research says about key risks for diabetes. Because of that, these models seem grounded in real human biology. The way they rank features feels consistent with medical knowledge.

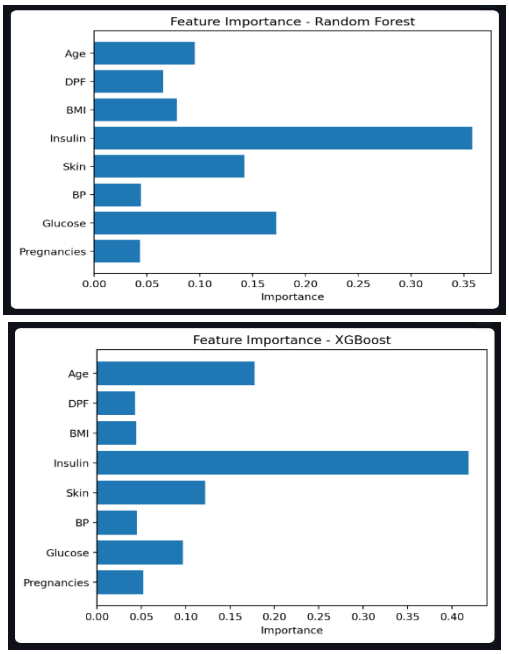

Chart -3: : Feature Importance Plot — Random Forest and XGBoost

7.5 Prediction of Samples

One look at four made-up patient cases reveals how the new method works. Findings appear in Table-5. High blood sugar - say, 180 or 200 mg/dL - paired with heavier weight and advancing years points toward diabetes. Just as anticipated, those patterns flagged the condition.

Table -5: Sample Patient Prediction Results

Patient	Glucose	BMI	Age	Prediction
Patient 1	120	28.0	35	Non-Diabetic
Patient 2	180	35.0	50	Diabetic
Patient 3	85	24.0	22	Non-Diabetic
Patient 4	200	40.0	60	Diabetic

8. DISCUSSION

One reason stands out - combining multiple machine learning models tends to work better than relying on just one for spotting early signs of diabetes. Though the boost in accuracy ranges between 1.3 and 3.3 percent, gains like these matter greatly in health screenings. Early detection improves dramatically, meaning many individuals get identified sooner. What happens next? More lives are impacted by timely interventions. That slight edge adds up across large populations.

Because blood sugar levels matter so much for spotting diabetes, it stands out as a key clue under global health guidelines - fasting levels at or above 126 mg/dL tell part of the story. Old enough age along with higher body mass shows links to type 2 issues and sluggish metabolism. When insulin data and skin fold measures seem less useful here, gaps in recorded numbers likely play a big role behind that pattern.

Sometimes performance lags behind when using hard voting within an ensemble, especially compared to softer methods that lean on tuned probabilities or layered models with a follow-up learner. At first glance, gains from soft voting showed up just barely - only once every starting model had its predictions aligned properly. What comes next looks closely at tuning those outcome chances and combining systems through stacked layers.

One thing stands out - the Streamlit app shows how the system can work in real medical settings, even if users do not understand coding. Because results appear in under a second, doctors could use it during active patient visits. Hospitals with tight budgets might benefit once it runs on cloud platforms.

One thing to remember about PIDD is it affects how we see the study's findings. This data comes entirely from women in the Pima Indian group, so applying these outcomes elsewhere might not work well. What also plays a role? Missing pieces like hemoglobin A1C values, movement levels, and food choices were left out completely.

9. CONCLSUSION

This study showed how a mix of machine learning methods can predict diabetes outcomes using data from Pima Indian patients. Before any model ran, missing entries marked as zeros were filled in, then values scaled evenly across features. Four approaches took part - Logistic Regression led off, followed by Random Forest, then XGBoost joined, while LightGBM completed the set. Three of these teamed up inside a VotingClassifier structure to combine their outputs. The group effort reached just under 80 percent correctness when tested.

It becomes obvious here - ensemble learning packs a punch when diagnosing medical conditions, blending varied reasoning styles into one sharper predictor.

Another thing stands out: the model works in real settings, thanks to a Streamlit interface helping physicians bypass coding entirely.

Though the current version works well on PIDD data when tested internally, real-world medical use needs more. One step involves testing across broader patient groups to confirm reliability. Another means adding new elements that reflect diverse health patterns. These additions fit naturally into the system's flexible structure. Progress along the path outlined in Section 10 should lead to a working tool for spotting diabetes. That outcome feels reachable without major redesign.

REFERENCES

[1] International Diabetes Federation, IDF Diabetes Atlas, 10th ed., Brussels, Belgium: IDF, 2021. [Online]. Available: https://www.diabetesatlas.org

[2] R. Miotto, F. Wang, S. Wang, X. Jiang, and J. T. Dudley, "Deep learning for healthcare: review, opportunities and challenges," *Briefings in Bioinformatics*, vol. 19, no. 6, pp. 1236–1246, Nov. 2018.

[3] A. L. Beam and I. S. Kohane, "Big data and machine learning in health care," *JAMA*, vol. 319, no. 13, pp. 1317–1318, Apr. 2018.

[4] L. Rokach, "Ensemble-based classifiers," *Artificial Intelligence Review*, vol. 33, no. 1–2, pp. 1–39, Feb. 2010.

[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," in *Proc. Symp. Comput. Appl. Med. Care*, 1988, pp. 261–265.

[6] 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.

[7] Q. Zou, K. Qu, Y. Luo, D. Yin, Y. Ju, and H. Tang, "Predicting diabetes mellitus with machine learning techniques," *Frontiers in Genetics*, vol. 9, p. 515, Nov. 2018.

[8] D. Sisodia and D. S. Sisodia, "Prediction of diabetes using classification algorithms," *Procedia Computer Science*, vol. 132, pp. 1578–1585, 2018.

[9] M. Maniruzzaman, N. Kumar, Md. M. Abedin, Md. S. Islam, H. S. Suri, A. S. El-Baz, and J. S. Suri, "Comparative approaches for classification of diabetes mellitus data: Machine learning paradigm," *Computer Methods and Programs in Biomedicine*, vol. 152, pp. 23–34, Dec. 2017.

[10] Md. M. Islam, Md. M. Ferdousi, S. Rahman, and H. Y. Bushra, "Likelihood prediction of diabetes at early stage using data mining techniques," in *Computer Vision and Machine Intelligence in Medical Image Analysis*, Singapore: Springer, 2020, pp. 113–125.

[11] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T.-Y. Liu, "LightGBM: A highly efficient gradient boosting decision tree," in *Advances in Neural Information Processing Systems (NIPS)*, vol. 30, 2017.

[12] I. Tasin, T. U. Nabil, S. Islam, and R. Khan, "Diabetes prediction using machine learning and explainable AI techniques," *Healthcare Technology Letters*, vol. 10, no. 1–2, pp. 1–10, 2023.

[13] H. Wu, S. Yang, Z. Huang, J. He, and X. Wang, "Type 2 diabetes mellitus prediction model based on data mining," *Informatics in Medicine Unlocked*, vol. 10, pp. 100–107, 2018.

[14] S. Agarwal, "Early detection of diabetes using machine learning," in *Proc. IEEE Int. Conf. Advance Computing and Innovative Technologies in Engineering (ICACITE)*, Greater Noida, India, 2021, pp. 1–6.