DIABETES PREDICTION THROUGH ENSEMBLE LEARNING IN MACHINE LEARNING

Project Report submitted in partial fulfillment of the requirements

for the award of the degree of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

by

VAMSI SARIPUDI (20501A05H9) **V. HARSHALOKH (20501A05I2)**

T. SOHITHA (20501A05G0) **S. VENKAT SAI KARTHIK (20501A05G1)**

T.V.P.S. SREEKAR (20501A05H1)

Under the Guidance of

Dr. Surapaneni Phani Praveen ,Ph.D.,

Associate Professor

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

PRASAD V POTLURI SIDDHARTHA INSTITUTE OF TECHNOLOGY

(Permanently Affiliated to JNTUK-Kakinada, AICTE approved

An NBA & NAAC accredited and ISO 9001:2015 Certified Institution)

KANURU, VIJAYAWADA – 520007

2023 – 2024

PRASAD V POTLURI SIDDHARTHA INSTITUTE OF TECHNOLOGY

(Affiliated to JNTUK: Kakinada, Approved by AICTE, Autonomous) (An ISO certified and NBA accredited institution)

Kanuru, Vijayawada – 520007

CERTIFICATE

This is to certify that the project report entitled **"DIABETES PREDICTION THROUGH ENSEMBLE LEARNING IN MACHINE LEARNING"** that is being submitted by Vamsi Saripudi (20501A05H9), V.Harshalokh(20501A05I2), T.Sohitha(20501A05G0), S. Venkat Sai Karthik(20501A05G1), T.V.P.S Sreekar (20501A05H1) in partial fulfillment for the award of the Degree of Bachelor of Technology in Computer Science and Engineering during the academic year 2023-2024.

Signature of the guide

Dr. S. Phani Praveen,

Associate Professor,
Department of CSE,
PVPSIT

Signature of the HOD

Dr. A. JayaLakshmi,

Professor and HOD,
Department of CSE,
PVPSIT

SIGNATURE OF THE EXTERNAL EXAMINER

DECLARATION

We declare that project entitled "**DIABETES PREDICTION THROUGH ENSEMBLE LEARNING IN MACHINE LEARNING**" is composed by ourselves, that the work contained herein is our own expert where explicitly stated otherwise in the text and that this work has not been submitted for any degree or professional qualification except as specified.

Vamsi Saripudi (20501A05H9)

V.Harshalokh (20501A05I2)

T.Sohitha (20501A05G0)

S. Venkat Sai Karthik (20501A05G1)

T.V.P.S Sreekar (20501A05H1)

ACKNOWLEDGEMENT

We sincerely thank the following distinguished personalities who have given their advice and support for successful completion of the work.

We are deeply indebted to our most respected guide **Dr.S.Phani Praveen, Associate Professor**, Department of computer science and engineering for his/her valuable and inspiring guidance, comments, suggestions and encouragement.

We extend our sincere thanks to **Dr. A. Jayalakshmi, Professor & Head** of the department for extending his cooperation and providing the required resources.

We would like to thank our beloved **Principal, Dr. Sivaji Babu** for providing the online resources and other facilities to carry out this work.

We would like to express our sincere thanks to our project coordinator **Ms.A.Divya, Assistant Professor** of the department for their helpful suggestions in presenting this document.

We would like to extend our sincere gratitude to all the review panel **Dr.P. Sai Kiran, Professor**, **Dr.R.Daniel, Associate Professor** and **Dr.J.Ramadevi, Sr.Assistant Professor** for their valuable suggestions.

We extend our sincere thanks to all other teaching faculty and non-teaching staff of the department, who helped directly or indirectly for their cooperation and encouragement.

Vamsi Saripudi (20501A05H9)

V.Harshalokh (20501A05I2)

T.Sohitha (20501A05G0)

S. Venkat Sai Karthik (20501A05G1)

T.V.P.S Sreekar (20501A05H1)

ABSTRACT

In the dynamic landscape of healthcare, the timely and precise detection of medical conditions is of paramount importance. Diabetes, characterized as a chronic disease, poses significant health risks due to prolonged elevated sugar levels in the bloodstream. Early prediction of diabetes is crucial to mitigate its consequences and severity. However, the challenge lies in the limited availability of data and the presence of outliers, such as missing values and noise. This project addresses this challenge by proposing an accurate diabetes prediction model employing machine learning algorithms, including KNN, XGBoost, CatBoost, and Random Forest. The approach incorporates the ensembling of these algorithms to enhance prediction accuracy. The primary objective of the model is to predict the future risk of an individual developing diabetes.. All the experiments were conducted under the same experimental conditions using the Pima Indian Diabetes Dataset. Through a systematic pipeline involving data preprocessing, individual model training, and ensembling techniques, coupled with a carefully crafted frontend design for user interaction and input, the project achieves an outstanding accuracy rate of 97% in predicting diabetes. This project aims to contribute to the advancement of early diabetes prediction, thereby facilitating proactive healthcare interventions.

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

S.NO	ABBREVIATION	FULL FORM
1	KNN	K-Nearest Neighbors
2	XGBoost	Extreme Gradient Boosting
3	CatBoost	Categorical Boosting
4	RF	Random Forest
5	IQR	Interquartile Range
6	HTTP	Hypertext Transfer Protocol
7	CSS	Cascading Style Sheets
8	API	Application Programming Interface
9	RMSE	Root Mean Squared Error
10	SVM	Support Vector Machine
11	NN	Neural Network
12	DT	Decision Tree
13	UI	User Interface
14	JS	JavaScript
15	Flask	Python Web Framework (Micro)
16	ReactJS	JavaScript Library for Building UI
17	JSON	JavaScript Object Notation
18	HTML	Hypertext Markup Language
19	REST	Representational State Transfer
20	CPU	Central Processing Unit
21	RAM	Random Access Memory
22	PCA	Principal Component Analysis
23	ML	Machine Learning
24	ROC	Receiver Operating Characteristic
25	AUC	Area Under the Curve
26	IDE	Integrated Development Environment
27	GUI	Graphical User Interface
28	UX	User Experience
29	HTTPS	Hypertext Transfer Protocol Secure
30	Std.	Standard Deviation
31	Avg.	Average
32	FOR	False Omission Rate
33	DOR	Diagnostic Odds Ratio

CHAPTERS

CHAPTER-1

INTRODUCTION

1.1 INTRODUCTION TO PROJECT:

The project endeavors to pioneer a diabetes prediction model through the strategic application of cutting-edge machine learning algorithms. Diabetes, a pervasive metabolic disorder characterized by aberrant blood glucose levels, stands as an imminent public health crisis with far-reaching implications. The World Health Organization (WHO) forecasts a continual rise in the prevalence of diabetes, positioning it as one of the top 10 fastest-growing health concerns by 2030. Against this backdrop of escalating health concerns, comprehending the intricate diabetes becomes imperative. Factors ranging from deficiencies in insulin secretion to compromised cellular responsiveness to insulin intricately shape the disease's trajectory, underscoring the urgency for pre-emptive interventions.

By analyzing diverse datasets and employing predictive modeling approaches, researchers can develop robust diabetes prediction models that aid in early disease detection, personalized treatment planning, and resource allocation. Despite challenges related to data accuracy, this research aims to create a comprehensive diabetes prediction model using machine learning algorithms and rigorous data analysis techniques. Leveraging the PIMA Indian diabetes dataset and ensemble modeling methods, this study seeks to improve our ability to predict diabetes accurately, ultimately contributing to better health outcomes and patient care.

1.2 MOTIVATION:

The motivation behind this project stems from the alarming trend identified by the World Health Organization (WHO) regarding the rising prevalence of diabetes and its associated health complications. Diabetes, being a chronic disease, requires early detection and management to mitigate its adverse effects on individuals' health and well-being. The significance of timely prediction lies in the potential to prevent severe outcomes such as coma, kidney failure, and cardiovascular dysfunction. Moreover, with the integration of machine learning techniques into healthcare, there's a unique opportunity to enhance our capacity for diabetes prediction and personalized treatment strategies. By leveraging advanced algorithms like KNN, XGBoost, CatBoost, and Random Forest, this project

aims to develop a predictive model that can accurately assess an individual's risk of developing diabetes. Ensembling these algorithms further enhances the predictive power of the model, ensuring robustness and reliability in diabetes prediction. Ultimately, the goal of this project is to contribute to preventive healthcare by empowering healthcare professionals with a reliable tool for early diabetes detection and intervention.

1.3 STATEMENT OF THE PROBLEM AND SOLUTION TO PROBLEM

1.3.1 PROBLEM STATEMENT

The rising prevalence of diabetes worldwide, as projected by the World Health Organization (WHO), highlights the urgent need to develop an accurate and reliable predictive model for diabetes prediction, leveraging advanced machine learning techniques and healthcare data and enable early detection of diabetes risk in individuals, facilitating timely intervention to reduce complications and improve overall patient's health.

1.3.2 SOLUTION

This project aims to develop a comprehensive diabetes prediction model by integrating insights from prior research and leveraging cutting-edge machine learning techniques. Through meticulous experimentation and rigorous evaluation, the project synthesizes diverse methodologies into a cohesive framework, capable of delivering unparalleled accuracy and robustness in diabetes prediction. By embracing ensemble-based approaches and employing advanced data preprocessing techniques, the envisioned solution seeks to transcend the limitations of existing predictive models, paving the way for proactive intervention and personalized patient care in the realm of diabetes management.

1.4 OBJECTIVES

1.4.1 GENERAL OBJECTIVE

To develop a robust and reliable machine learning model capable of accurately predicting the likelihood of diabetes in patients based on a comprehensive analysis of their medical data. This model aims to provide healthcare professionals with a valuable tool for early

detection and proactive management of diabetes, thereby improving patient outcomes and quality of care.

This expanded version provides additional context and emphasizes the importance of the machine learning model in healthcare decision-making and patient management.

1.4.2 SPECIFIC OBJECTIVES

DATA PREPROCESSING:

- To clean the dataset by identifying and handling missing values, outliers, and Class Imbalance
- To perform feature engineering to extract relevant features and enhance the predictive power of the dataset."
- To scale and normalize the features to ensure uniformity and improve model performance.

FINDING THE BEST MODELS AND ENSEMBLING:

- To conduct thorough experimentation with various machine learning algorithms, including logistic regression, decision trees, random forests, and neural networks.
- To evaluate the performance of each model using appropriate metrics such as accuracy, precision, recall, and F1-score.
- To explore ensemble techniques such as bagging, boosting, and stacking to combine the strengths of multiple models and improve prediction accuracy.

DESIGNING A USER-FRIENDLY INTERFACE:

- To gather requirements and user feedback to understand the needs and preferences of healthcare professionals.
- To design an intuitive and visually appealing user interface that provides easy access to diabetes prediction functionality.
- To implement the user interface using appropriate technologies and frameworks, ensuring compatibility with different devices and platforms.
- To conduct usability testing and iterate on the design based on user feedback to enhance user satisfaction and efficiency.

1.5 SCOPE OF THE WORK

This project's scope is comprehensive, aiming to develop a sophisticated diabetes prediction system that harnesses the power of advanced machine learning techniques. The system will not only incorporate insights from prior research, such as the "eDiaPredict" framework [1], but will also delve into ensemble-based approaches to enhance predictive accuracy and robustness. Moreover, the project will extend its scope to include the integration of lifestyle factors, such as dietary habits, stress levels, and routine medical exams, into the predictive model. By doing so, it aims to provide a holistic view of diabetes risk assessment and pave the way for personalized patient care strategies.

1.6 SIGNIFICANCE OF THE WORK

This project holds immense significance in the realm of diabetes prediction and healthcare. By developing a precise diabetes prediction system, it has the potential to revolutionize the way individuals assess their future diabetes risk. The integration of advanced machine learning techniques and ensemble-based approaches promises to enhance the efficacy of diabetes prediction, enabling proactive intervention and personalized patient care. Furthermore, the inclusion of lifestyle factors into the predictive model not only enriches its accuracy but also empowers individuals to make informed decisions about their health. Ultimately, this project's significance lies in its potential to improve health outcomes, reduce healthcare costs, and contribute to the broader field of predictive medicine.

1.7 OUTLINES OF THE PROJECT

- **Introduction**: Provides an overview of the project objectives and motivations, highlighting the importance of diabetes prediction in healthcare.
- **Literature Review**: Reviews prior research endeavours and seminal works in the field of diabetes prediction, offering insights into existing methodologies and approaches.
- **Methodology**: Describes in detail the methodology employed in the development and evaluation of the predictive model, including data pre-processing, feature selection, model training, and evaluation techniques.
- **Results**: Presents the findings of the project, including the performance evaluation of the predictive model across diverse datasets and the impact of integrating lifestyle factors.

- **Discussion**: Discusses the implications of the findings, potential challenges encountered during the project, and avenues for future research and development.
- **Conclusion**: Summarizes the key findings and contributions of the project, emphasizing its significance in improving diabetes prediction and healthcare outcomes.

CHAPTER 2

BACKGROUND AND LITERATURE REVIEW

Table 2.1. Literature Review

SNO	TITLE OF PAPER	YEAR OF PUBLICATION	JOURNAL/ CONFERENCE	AUTHORS	INFERENCES
1	E DIAPREDICT: An Ensembled based Frame work for diabetes prediction	2021	Journal	Ashima Singh, Arwinder Dhillon, Neeraj Kumar, M. Shamim Hossain	•The mean Imputation of missing Values is done without addressing the class imbalance which leads to the Biased imputation. •The effectiveness of feature extraction using RFE may depend on the dataset and selection criteria.
2	Comparing Machine Learning Models for Diabetes Prediction on the PIMA Dataset	2021	Journal	Jobeda Jamal Khanam, Simon Y. Foo	Neural network achieves highest accuracy at 88.6% - Limited discussion on feature reduction and class imbalance, impact of dropping three features using the WEKA tool not thoroughly discussed
3	Prediction of diabetes using logistic regression	2021	Journal	Priyanka Rajendra, Shahram Latif	Logistic regression efficient in prediction - Importance of

	and ensemble techniques				preprocessing and cross-validation highlighted but Class Imbalanced was not dealt Properly
4	Diabetes Prediction Using Ensembling of Different Machine Learning Classifiers	2020	Journal	MD. Kamrul Hasan, MD. Ashraful Alam, Dola Das, Eklas Hossain	- Weighted ensemble of Adaptive Boosting and XGBoost outperforms other models - Complexity introduced by ensemble modeling discussed - Model complexity could hinder implementation
5	Diabetes prediction using supervised machine learning	2023	Journal	Muhammad Exell Febriana, Fransiskus Xaverius Ferdinana, Gustian Paul Sendania	- Naive Bayes outperforms k-Nearest Neighbor algorithms - Single optimization technique mentioned for future research - Lack of discussion on potential drawbacks of the Naive Bayes algorithm
6	Diabetes risk prediction model based on community follow-up data using machine learning	2023	Journal	Liangjun Jiang, Zhenhua Xia, Ronghui Zhu, Haimei Gong, Jing Wang	- Introduction of risk assessment model based on random forest classifier - Lack of preprocessing steps discussed - Model's reliance on data quality may limit applicability
7	Diagnosis and Classification of	2020	Journal	Casey B, Wapner RJ, Varner MW	- Emphasis on logistic regression efficiency -

	Diabetes				Feature selection algorithm limitation mentioned - Limited discussion on other potential classification techniques
8	XGBoost: A Scalable Tree Boosting System for Diabetes	2019	Journal	Carlos Guestrin, Tianqi Chen	- XGBoost achieves highest accuracy at 83.9% - Model complexity and lack of preprocessing discussed - Restriction to only XGBoost could limit exploration of other algorithms
9	AdaBoost for Feature Selection, Classification and Its Relation with SVM	2019	Journal	Ruihu Wang	- Significance of AdaBoost in feature selection and classifier learning highlighted - Theoretical concept not implemented - Limited discussion on potential drawbacks of AdaBoost and Limited Local Optimum
10	Correlation Method for Identification of a Nonparametric Model of Type 1 Diabetes	2022	Journal	N/A	- Proposal of linear nonparametric model - Limitation related to feature elimination mentioned - Lack of discussion on potential drawbacks of the proposed method
11	Development of	2019	Journal	Norma Latif	- Novel integration of

	Disease Prediction Model Based on Ensemble Learning Approach for Diabetes and Hypertension			Fitriyal, Muhammad Syafruddin, Ganjar Alfain, Jongtae Rhee	SMOTE and ensemble learning techniques - Assumption of data quality discussed - Potential limitations related to data quality and reliability addressed

CHAPTER 3

SYSTEM ANALYSIS

3.1 EXISTING SYSTEM:

In existing research, various ensemble modeling techniques based on machine learning (ML) models such as XGBoost, Random Forest (RF), Support Vector Machine (SVM), Neural Network (NN), and Decision Tree (DT) have been employed to predict diabetes using the PIMA dataset. The dataset undergoes preprocessing to handle missing values, followed by feature extraction using Recursive Feature Elimination (RFE). Each individual model is trained and evaluated independently, after which the best-performing models are selected through a majority voting method and ensembled to predict the desired outcome. The robustness of the models is assessed using tenfold cross-validation.

3.1.1 DISADVANTAGES OF EXISTING SYSTEM:

- While ensemble modeling offers improved performance, it may introduce complexity to the predictive framework.
- The mean Imputation of missing Values is done without addressing the class imbalance which leads to the Biased imputation.
- The effectiveness of feature extraction using RFE may depend on the dataset and selection criteria.
- Tenfold cross-validation, while common, may not capture all potential sources of variability in the data.

3.2 PROPOSED SYSTEM:

- Utilizing the PIMA Indian diabetes dataset, consisting of 768 records with 268 diabetic and 500 control cases.
- Employing ensemble learning techniques including KNN, RF, XGBoost, and CatBoost for diabetes risk prediction.
- Initial phase involves data preprocessing, addressing missing values using class-aware mean imputation, and outlier elimination using the interquartile range.
- The IQR outlier rejection approach is used for identifying and handling outliers within the dataset.
- Standard scaling is applied for data pre-processing to ensure all features are on a congruous scale.
- Class imbalance is addressed by using Random over sampling.

- Training of models involves division of data into training and test sets, followed by model selection and hyper parameter tuning.
- Models such as KNN, RF, XGBoost, and CatBoost are trained with specific parameters tailored to optimize performance.
- Model ensembling is performed using soft voting and hard voting classifiers, combining the predictions of selected models through probability averaging.
- Ensemble combinations include KNN and RF for capturing local and global patterns respectively, while XGBoost and CatBoost are integrated with other models for enhanced predictions.
- Designing a user-friendly React.js frontend for seamless data input.
- Integrating the React frontend with a Flask backend to facilitate communication.
- Handling user inputs, processing requests, and interfacing with the machine learning model.

3.3 FEASIBILITY STUDY:

After asessing the usefulness of the proposed system we understood it is techically, economically and operationally viable.

3.3.1 TECHNICAL FEASIBILITY:

A range of well-established machine learning algorithms and libraries are available, making it technically feasible to develop predictive models for diabetes prediction. Flask is a popular python framework used for front end development and its ease of use and integration with machine learning libraries makes it a good choice.React.js is chosen for frontend development, providing flexibility and efficiency in constructing an intuitive user interface for seamless data input.

3.3.2 ECONOMIC FEASIBILITY:

- **Data Acquisition** for building the diabetes prediction system is done from the openly available PIMA INDIAN diabetes dataset.
- **Processing costs** associated with model training and testing are a consideration. Using free features provided in 'Google Colab' platform, these costs can be averted.
- **Deployment costs** associated with model deployment can be completely waived, using free hosting service like vercel.

3.3.3 OPERATIONAL FEASIBILITY:

Operational feasibility is enhanced by practical considerations in the implementation of the diabetes prediction system. To enhance system responsiveness, the state of the model

is efficiently preserved using the pickle Python module, mitigating the need for repetitive model re-training and thus accelerating the system's speed of response. Additionally, the integration of an interactive frontend is crucial for operational success. This user interface, designed for seamless interaction, not only facilitates user input but also plays a pivotal role in providing an exceptional user experience. Marked by intuitive and user-friendly features, the interactive frontend minimizes complexities, contributing to an efficient and user-centric operation of the diabetes prediction system.

CHAPTER 4

SOFTWARE REQUIREMENT SPECIFICATION

Requirement analysis is critical to the success of a systems or software project. The requirements should be documented, actionable, measurable, testable and traceable, related to identified business needs or opportunities, and defined to a level of detail sufficient for system design and Analysis involves the requirement determination and specification. Systems analysis is a problem-solving technique that decomposes a system into its component pieces for the purpose of studying how well those component parts work and interact to accomplish their purpose. According to the Merriam-Webster Dictionary, systems analysis is the process of studying a procedure or business in order to identify its goals and purposes and create systems and procedures that will achieve them in an efficient way. Analysis and synthesis, as scientific methods, always go hand in hand, they complement one another A System Requirements Specification (SRS) - a requirements specification for a software system- is a complete description of the behaviour of a system to be developed. It includes a set of use cases that describe all the interactions the users will have with the software. Use cases are also known as functional requirements. In addition to use cases, the SRS also contains non-functional (or supplementary) requirements.

4.1 FUNCTIONAL REQUIREMENTS:

Functional requirements outline a system's essential functionalities, focusing on what it does and what users expect from it, rather than how it achieves those functions. They are user-oriented, measurable, complete, and unambiguous, ensuring all stakeholders understand the system's intended behavior.

OS: Windows

PROGRAMMING LANGUAGE: Python

FRONT END: ReactJs

BACK END: Flask

IDE: Google collab

PROCESSOR: Any processor above 500 MHZ

RAM: 8 GB and Above

HARD DISK: 4 GB and Above

4.2 NON-FUNCTIONAL REQUIREMENTS:

In systems engineering and requirements engineering, a non-functional requirement is a requirement that specifies criteria that can be used to judge the operation of a system, rather than specific behaviors. This should be contrasted with functional requirements that define specific behavior of functions.

- **Accuracy:** It is the proportion of correctly classified instances out of the total number of instances
- **Precision:** It refers to the proportion of accurately predicted positive events to the total number of events that were identified as true.
- **Sensitivity:** The measure quantifies a model's capacity to accurately identify positive occurrences from the entire set of actual positive instances.
- **Specificity:** This statistic measures the model's capacity to accurately detect true negative situations out of all the real negative instances.
- **False Odds Ratio (FOR):** The False Omission Rate (FOR) quantifies the tendency of a model to incorrectly categorize negative cases as positive.
- **Usability:** The user-centric approach should be employed while designing the front-end.
- **Performance:** The pickle module is used to improve the speed of the system by avoiding frequent re-training of the model.

CHAPTER 5

DESIGN AND METHODOLOGY OF PROPOSED SYSTEM

5.1 SYSTEM ARCHITECTURE OR MODEL:

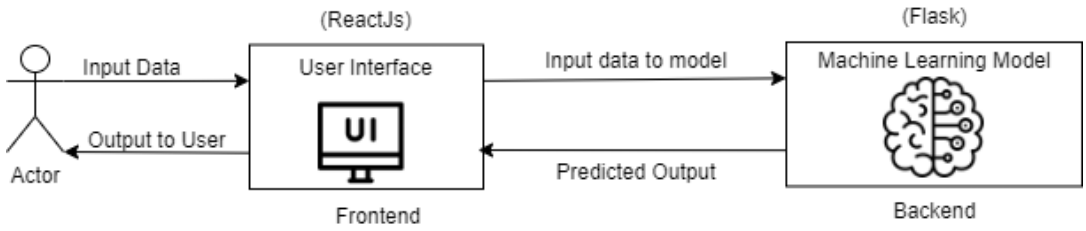

Figure. 5.1. Architecture Diagram

FRONTEND (REACTJS):

- Involves creating the visual interface where users interact with the application and provide information related to diabetes risk factors.
- Implements modular elements using React.js to structure and manage the user interface components, facilitating seamless user interactions.
- Handles the organization and preparation of user-inputted data before sending it to the backend for further processing.

BACKEND (FLASK):

- Acts as a bridge between the frontend and the machine learning model, processing requests from the frontend and managing communication with the model.
- Defines endpoints and controllers to manage incoming requests, directing them to the appropriate backend logic for processing.
- Combines with the pre-trained machine learning model, enabling the backend to utilize the model for predicting diabetes risk.
- Takes the machine learning model's output, processes it, and sends the relevant response back to the frontend.

MACHINE LEARNING MODEL:

- Utilizes pre-trained algorithms to analyze user-inputted data, predicting the likelihood of an individual having diabetes based on patterns learned during training.

- Developed separately, the model is trained using historical data to recognize correlations and patterns relevant to diabetes risk prediction.
- Merges with the Flask backend, creating a unified system capable of receiving data, making predictions, and providing results.

COMMUNICATION:

- Uses HTTP requests, typically following RESTful API principles, enabling the frontend to send user data to the backend for processing.
- Defines a communication mechanism, such as function calls or API endpoints, allowing the backend to interact with the machine learning model.

SECURITY:

- Implements measures to validate and sanitize user inputs on both the frontend and backend, preventing security vulnerabilities.

5.2 SYSTEM DESIGN

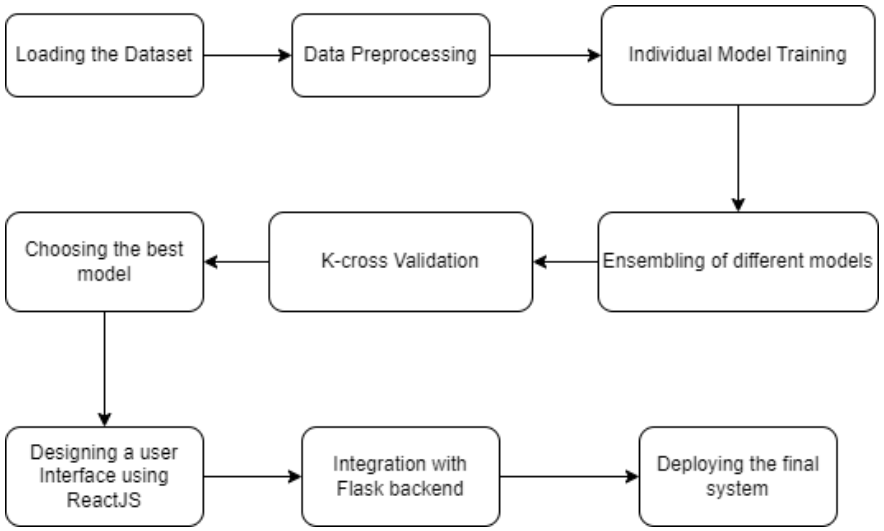

Figure. 5.2. Workflow diagram

DATA COLLECTION:

Raw data relevant to the project is collected from various sources such as medical databases, health records, sensor outputs, lab test results, or other health metrics. The collected data undergoes preprocessing to refine, remodel, and standardize it before analysis.

DATA PREPROCESSING:

Data preprocessing involves actions such as removing outliers, handling missing values, and standardizing features to ensure consistency and reliability in the dataset. This includes imputing missing values, handling the class Imbalance, Standard scaling, and outlier rejection. Preprocessing prepares the data for further analysis and model training.

DATASET SPLITTING:

The preprocessed dataset is divided into two parts: a training set and a testing set. The training set, comprising 80% of the data, is used to train machine learning models, while the remaining 20% is reserved for evaluating model performance.

MODEL TRAINING:

The training dataset is fed into machine learning models, specifically K Nearest Neighbor, SVM, Naive Bayes, Random Forest, XG and Cat Boosting in this architecture.

These algorithms are trained and optimized using the training data to learn patterns and relationships between input features and target outcomes.

ENSEMBILING MODELS:

Ensembling models involves combining the predictions of multiple individual algorithms to create a more accurate and robust predictive model. In the case of diabetes prediction, this process may include algorithms such as Random Forest, CatBoost, XGBoost, k-Nearest Neighbors, among others. Each of these algorithms has its strengths and weaknesses, and by ensembling them together, we can leverage their complementary nature to improve overall prediction performance. Techniques like soft voting or averaging predicted probabilities are commonly used to merge the predictions of these models, resulting in a final prediction that is more reliable and precise than any individual model alone.

MODEL EVALUATION:

The trained models are evaluated using the testing dataset to assess their accuracy and performance. In this process, k-fold cross-validation is utilized, where the dataset is divided into k subsets, and each model is trained and tested k times, using a different

subset for testing each time. This comprehensive evaluation approach provides a robust assessment of the models' predictive capabilities across diverse data samples.

PREDICTION GENERATION:

After successful modeling and optimization, the trained models are used to make predictions on new, unseen data. Predictions generated by the models indicate the probability that an individual with provided parameters will have a certain disease, within the scope of the project. This system architecture facilitates the end-to-end process of disease prediction, starting from data collection and preprocessing, through model training and evaluation, to prediction generation. It ensures that the models are trained and optimized effectively to provide accurate predictions on new data.

FRONTEND DESIGN:

Designing the user interface involved creating React components that facilitate data input, employing form components, and other UI elements for a user-friendly experience. State management was implemented to dynamically handle user inputs and form data, and the components were styled for visual appeal using CSS-in-JS libraries or traditional style-sheets. Client-side form validation ensured the submission of valid and complete data.

BACKEND DESIGN:

Flask API endpoints were defined to handle requests from the React frontend, incorporating secure data transmission through RESTful API principles. The Flask backend, responsible for processing requests, was connected to the trained machine learning model, ensuring proper data preprocessing before passing inputs to generate predictions. Error handling mechanisms were implemented for robustness, and thorough testing verified the seamless communication between the React frontend and Flask backend.

5.2.1 UML DIAGRAMS:

The unified modelling language allows the software engineer to express an analysis model using the modelling notation that is governed by a set of syntactic semantic and pragmatic rules. A UML system is represented using five different views that describe the

system from distinctly different perspective. Each view is defined by a set of diagrams, which is as follows.

USE CASE DIAGRAM:

In this project, we commence by scrutinizing the problem statement and collecting relevant, responsive data. Following data collection, we implement preprocessing techniques to refine and standardize the dataset. Next, we meticulously select an appropriate machine learning model for training and testing. Once the model is trained, we assess its accuracy, ensuring a comprehensive evaluation through validation. With a validated model, we proceed to predictions, leveraging its capabilities to forecast diabetes based on the provided data. Additionally, the project involves the development of a user-friendly frontend based on ReactJS and a Flask backend.

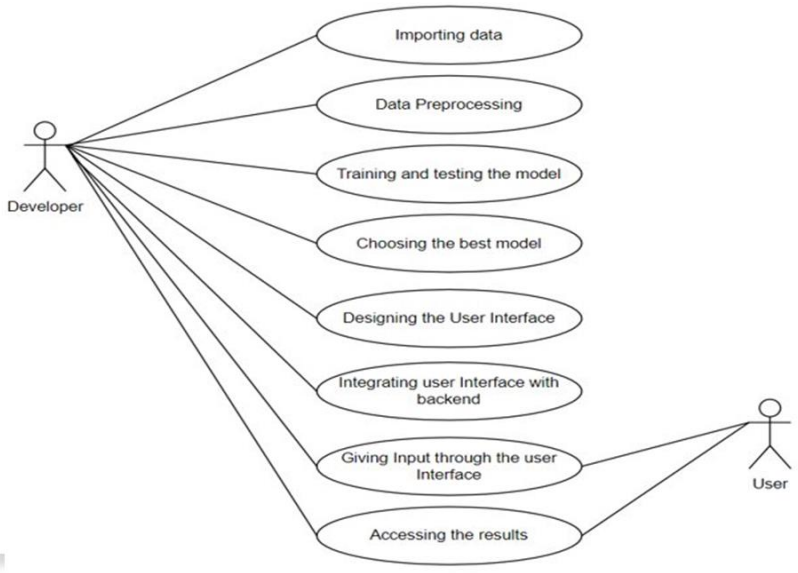

Figure. 5.3. Use-case Diagram

ACTIVITY DIAGRAM:

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.Activity diagrams are constructed from a limited repertoire of shapes, connected with arrows.

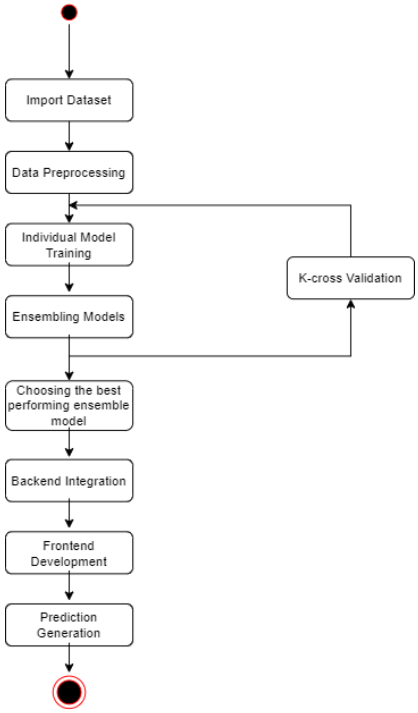

Figure. 5.4. Activity Diagram

STATE DIAGRAM:

The State Diagram offers a visual depiction of the system's operational flow, delineating key states and transitions within its functionality. Commencing from an initial "Idle" state, the system progresses to "Frontend Ready" upon successful loading of the React.js frontend. From there, user initiation of a prediction request triggers a transition to the "Request Sent" state, where the frontend dispatches the request to the Flask server. Subsequently, the system enters the "Backend Processing" state, signifying the server's processing of the prediction request. Upon successful completion, the system transitions to the "Prediction Ready" state, poised to deliver the prediction result. Conversely, should an error arise during processing, the system shifts to an "Error" state for appropriate handling.

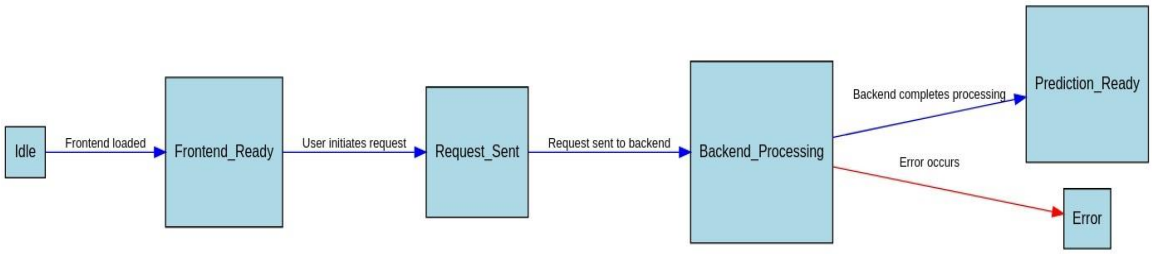

Figure. 5.5. State Diagram

COMPONENT DIAGRAM:

The Component Diagram offers a comprehensive depiction of the system's architecture, illustrating key components and their interactions. At its core, the system comprises three

primary components: the React.js Frontend, Flask Server, and Prediction Model. The React.js Frontend serves as the user interface, facilitating user interactions and transmitting requests to the Flask Server. The Flask Server acts as the backend, receiving requests from the frontend, processing them, and interfacing with the Prediction Model to generate prediction results.

Figure. 5.6. Component Diagram

SEQUENCE DIAGRAM:

The sequence diagram provides a chronological representation of interactions between key components within the system. The diagram illustrates the sequence of messages exchanged between the user, frontend, server, and prediction model during a prediction request. It begins with the user initiating a prediction request through the frontend interface. Subsequently, the frontend forwards the request to the server via an HTTP request, which then delegates the task to the prediction model for processing. Upon processing, the server receives the prediction results and sends them back to the frontend as an HTTP response. Finally, the frontend displays the prediction results to the user.

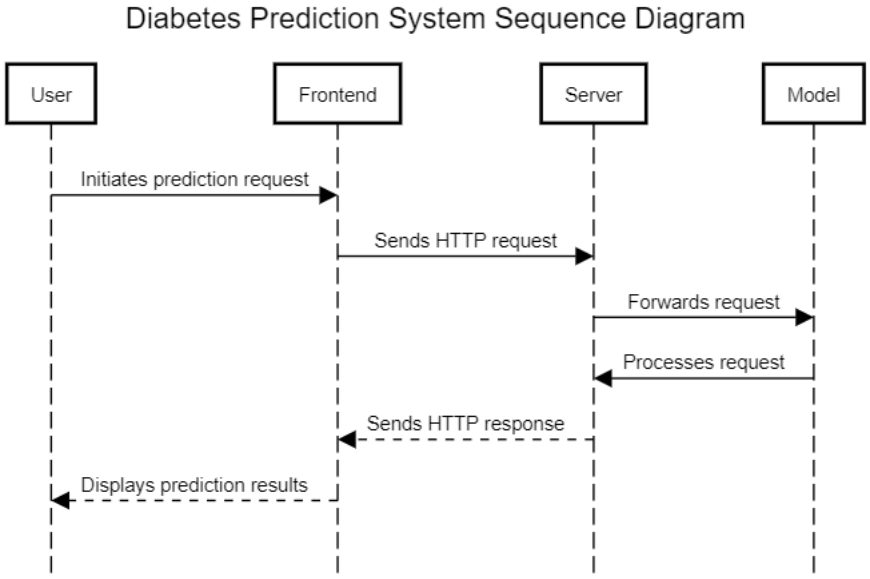

Figure. 5.7. Sequence Diagram

5.3 METHODOLOGY

This study employs a systematic methodology involving data collection, process modeling, and algorithm selection to predict the risk of individuals developing diabetes using the PIMA dataset.

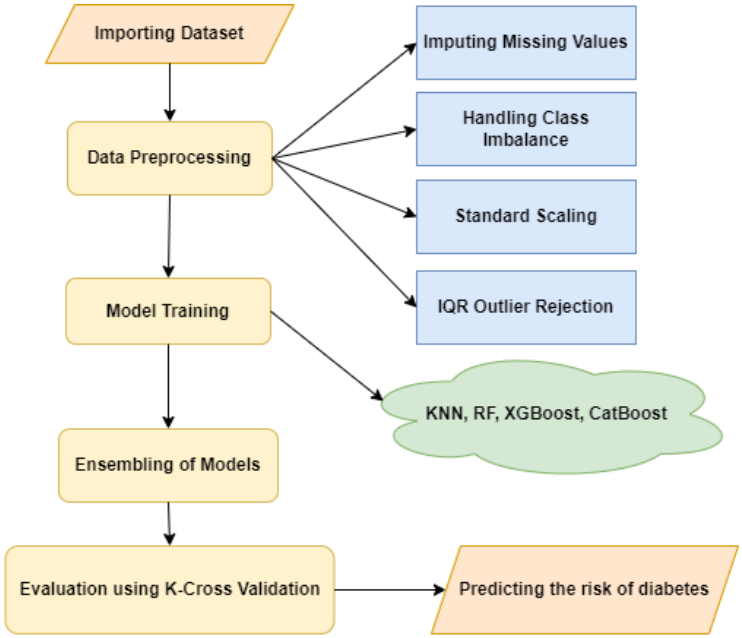

Figure. 5.8. Ensemble model building methodology

5.3.1 DATA COLLECTION METHODOLOGY

The PIMA Indian Diabetes dataset, consisting of 768 records, was obtained from reputable sources such as the UCI Machine Learning Repository. The dataset encompasses eight attributes, including pregnancies, glucose levels, blood pressure, skin thickness, insulin levels, BMI, age, and diabetes pedigree function.

5.3.2 PROCESS MODEL/METHODOLOGY/ALGORITHMS:

The process model and methodology adopted in this study follow a structured approach to effectively predict the risk of individuals developing diabetes using the PIMA dataset. This section outlines the detailed steps involved in data preprocessing, model training, and algorithm selection:

DATA PRE-PROCESSING:

Data preprocessing is a crucial step to ensure the integrity and quality of the dataset before model training. This involves:

Identifying and handling missing values: Techniques such as mean imputation or predictive imputation are used to address missing data points, ensuring no information loss.

Class Imbalance: Random Over Sampling technique is employed to rectify class imbalance by augmenting the number of instances in the minority class until it achieves parity with the majority class.

Outlier detection and removal: Outliers, which can skew model performance, are identified using statistical methods like the interquartile range (IQR) and eliminated to enhance the dataset's reliability.

Feature scaling and normalization: Features in the dataset are scaled and normalized to ensure uniformity and prevent bias, particularly for machine learning algorithms sensitive to feature magnitudes.

MODEL TRAINING:

After preprocessing, the dataset is split into training and testing sets, typically using an 80-20 or 70-30 ratio. The training set is used to train various machine learning algorithms, including:

K-NEAREST NEIGHBOR (K-NN):

k-Nearest Neighbor is one of the simplest Supervised Machine Learning algorithms. k-NN algorithm assumes the similarity between the new case/data and available cases and put the new case into the category that is most similar to the available categories. k-NN algorithm stores all the available data and classifies a new data point based on the similarity. It is also called a lazy learner algorithm because it does not learn from the training set immediately instead it stores the dataset and at the time of classification.

The K-NN working can be explained on the basis of the below steps:

Step-1: Select the number K of the neighbors

Step-2: Calculate the Euclidean distance of K number of neighbors

Step-3: Take the K nearest neighbors as per the calculated Euclidean distance.

Step-4: Among these k neighbors, count the number of the data points in each category.

Step-5: Assign the new data points to that category for which the number of the neighbor is maximum.

Step-6: Model is ready.

RANDOM FOREST (RF):

Random Forest is a popular machine learning algorithm that belongs to the supervised learning technique. It can be used for both Classification and Regression problems in ML. It is based on the concept of ensemble learning, solves a complex problem and to improve the performance of the model. As the name suggests, "Random Forest is a classifier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset.

The random forest combines multiple trees to predict the class of the dataset, it is possible that some decision trees may predict the correct output, while others may not. But together, all the trees predict the correct output. Therefore, below are two assumptions for a better Random Forest classifier:

● There should be some actual values in the feature variable of the dataset so that the

classifier can predict accurate results rather than a guessed result.

● The predictions from each tree must have very low correlations.

Random Forest works in two-phase first is to create the random forest by combining N decision tree, and second is to make predictions for each tree created in the first phase. The Working process can be explained in the below steps:

Step-1: Select random K data points from the training set.

Step-2: Build the decision trees associated with the selected data points (Subsets).

Step-3: Choose the number N for decision trees that you want to build.

Step-4: Repeat Step 1 & 2.

Step-5: For new data points, find the predictions of each decision tree, and assign the new datapoints to the category that wins the majority votes.

XG BOOST (EXTREME GRADIENT BOOSTING):

XG Boost is a decision-tree-based ensemble Machine Learning algorithm that uses a gradient boosting framework. However, when it comes to small-to-medium structured/tabular data, decision tree based algorithms are considered best-in-class right now.

Step-1. Prepare your data: Split it into training and testing sets, then clean and format it as needed.

Step-2. Set up the model: Choose some basic settings for training, like the number of "trees" and learning rate.

Step-3. Train the model: Teach the model using the training data.

Step-4. Make predictions: Use the trained model to guess outcomes on the testing data.

Step-5. Check performance: See how well the model performed using various measures.

CATEGORICAL BOOSTING:

CatBoost is a powerful gradient boosting algorithm designed for handling categorical features efficiently. Similar to XGBoost, it follows a structured process for model training and evaluation.

Step 1: Prepare your data:

Divide your dataset into training and testing sets to ensure unbiased evaluation.

Perform data cleaning and preprocessing tasks, such as handling missing values and encoding categorical variables.

Step 2: Set up the model:

Specify basic settings for training the CatBoost model, such as the number of iterations (trees) and the learning rate.

Optionally, configure other hyperparameters like the depth of trees and regularization settings based on your specific use case.

Step 3: Train the model:

Feed the prepared training data into the CatBoost model and start the training process.

Cat Boost will iteratively build an ensemble of decision trees, optimizing them to minimize the specified loss function.

Step 4: Make predictions:

Once the model is trained, use it to make predictions on the testing data or any new data that requires prediction.

CatBoost's trained model can efficiently handle both classification and regression tasks.

Step 5: Check performance:

Evaluate the performance of the CatBoost model using various metrics appropriate for your problem, such as accuracy, precision, recall, F1-score, or RMSE (Root Mean Squared Error) for regression tasks.

ALGORITHM SELECTION AND OPTIMIZATION:

The performance of each algorithm is evaluated using appropriate metrics such as accuracy, precision, recall, and F1-score. Hyperparameters of the models are tuned using techniques like grid search or randomized search to optimize performance further.

ENSEMBLE LEARNING:

Ensemble learning techniques, such as soft voting and hard voting classifiers, are employed to combine predictions from multiple models. This ensemble approach enhances predictive accuracy and robustness by leveraging the diverse strengths of individual algorithms. For instance, KNN and RF may be combined to exploit their complementary characteristics, while XGBoost and CatBoost may offer superior performance when ensembled together.

EVALUATION AND VALIDATION:

The trained models are evaluated using cross-validation techniques, such as k-fold cross-validation, to assess their generalization ability and prevent overfitting. This involves

splitting the dataset into k subsets, training the model on k-1 subsets, and validating it on the remaining subset. The process is repeated k times, ensuring each subset serves as both training and validation data.

By following this comprehensive process model and methodology, this study aims to develop robust predictive models for diabetes risk assessment, leveraging state-of-the-art algorithms and techniques to achieve accurate and reliable results.

5.3.3 TOOLS AND TECHNIQUES

The project utilized a diverse set of tools, programming languages, and techniques to facilitate data analysis, modeling, and evaluation. Python emerged as the primary programming language due to its rich ecosystem of libraries and frameworks for machine learning and data visualization. Examples include the extensive use of libraries like Pandas, NumPy, and SciPy for data manipulation and statistical analysis. For machine learning tasks, popular frameworks such as scikit-learn were employed for model development and deployment.

Moreover, data preprocessing techniques like feature scaling, imputation of missing values, and handling of class imbalance were implemented using specialized functions and methods provided by these libraries. Additionally, cross-validation techniques such as k-fold cross-validation and stratified sampling were employed to assess model performance and prevent overfitting. Ensemble learning techniques, i.e, soft voting and hard voting were also explored to combine predictions from multiple models and improve overall accuracy. Overall, the project methodology emphasized a robust and iterative approach, leveraging state-of-the-art tools and techniques to derive meaningful insights and predictive models for diabetes risk assessment.

CHAPTER 6

IMPLEMENTATION

6.1 MODULES

- Pre-processing (Class Imbalance, Imputation of Missing values, Outliers, Scaling, Feature selection)
- Machine Learning Models
- Ensembiling Machine learning models
- K-Cross Validation
- Saving the Best Model State
- Designing a Web Application using Flask and ReactJs

6.2 DESCRIPTION OF SAMPLE CODE OF EACH MODULE

6.2.1 PRE-PROCESSING:

CLASS AWARE MEAN IMPUTATION:

```
data[['BloodPressure', 'SkinThickness', 'Insulin', 'BMI']] =
data.groupby('Outcome')[['BloodPressure', 'SkinThickness',
'Insulin', 'BMI']].transform(lambda x: x.replace(0, x.mean()))
```

This code iterates through each outcome category, identifies missing values (represented by 0) in the four chosen features, and replaces them with the average value observed for that feature within the same outcome category. This approach is more appropriate than simply replacing missing values with the overall mean across all data points, as it considers potential differences in these features between different outcome classes.

CLASS IMBALANCE:

```
from imblearn.over_sampling import RandomOverSampler
ros = RandomOverSampler(random_state=69)
X_Data, Y_Lavel = ros.fit_resample(X_Data, Y_Lavel)
```

By oversampling the minority class, this code aims to address the imbalance and ensure the model is trained on a more balanced dataset, potentially leading to improved performance on the minority class and reduced bias towards the majority class.

HANDLING OUTLIERS:

```
def Manual(data, numeric_columns=None, outlier_factor=1.5,
keep_outliers=False):
    data_copy = data.copy()

    if numeric_columns is None:
        numeric_columns =
data_copy.select_dtypes(include=['number']).columns

    for col in numeric_columns:
        Q1 = data_copy[col].quantile(0.25)
        Q3 = data_copy[col].quantile(0.75)
        IQR = Q3 - Q1


        lower_bound = Q1 - outlier_factor * IQR
        upper_bound = Q3 + outlier_factor * IQR

        if keep_outliers:
            data_copy = data_copy[(data_copy[col] < lower_bound) |
(data_copy[col] > upper_bound)]
        else:
            data_copy = data_copy[(data_copy[col] >= lower_bound) &
(data_copy[col] <= upper_bound)]

    return data_copy
```

This code defines a function named Manual that detects and potentially removes (or keeps) outliers in a dataset using the Interquartile Range (IQR) method. It first creates a copy of the input data to avoid modifying the original information. For each numeric column, it calculates the 25th and 75th percentiles (Q1 and Q3) and computes the IQR, representing the spread of the central 50% of the data. Based on these percentiles and a user-defined factor, it defines upper and lower bounds to potentially identify outliers. The function takes an optional flag keep outliers: if True, it keeps only rows where the values in the current column fall outside the bounds (potentially keeping outliers), while if False (default), it keeps only rows within the bounds (potentially removing outliers). Ultimately, the function returns the modified copy of the data after filtering based on the chosen outlier handling approach, potentially impacting the presence of outliers in the results

STANDARD SCALING AND FEATURE SELECTION

```
data = Manual(data)
```

```
data = data.reset_index(drop=True)

Y_Lavel = data.iloc[:, -1]

from sklearn.preprocessing import StandardScaler
sc = StandardScaler()

X_Data = data.drop(['Outcome'],axis=1).values
X_Data = sc.fit_transform(X_Data)
from sklearn.feature_selection import SelectKBest, f_classif
selector = SelectKBest(score_func=f_classif, k='all')
selector.fit(X_Data, Y_Lavel)
selected_features_indices = selector.get_support(indices=True)
print(selected_features_indices)
X_Data = selector.transform(X_Data)
```

This code performs two essential data pre-processing steps: standard scaling and feature selection.

STANDARD SCALING:

It first scales the features using Standard Scaler. This centers and scales each feature to have a mean of 0 and a standard deviation of 1. This ensures all features contribute equally to model training, as features with larger scales wouldn't dominate the learning process.Additionally Custom weights can be given to the Specific Features before Scaling

6.2.2 FEATURE SELECTION:

The code then attempts to perform feature selection using Select K Best. This technique aims to identify and select a subset of features that are most relevant to predicting the target variable.

However, in this specific code snippet, the parameter k is set to 'all'. This means the current configuration keeps all features, and no selection is actually performed.

6.2.3 K-CROSS VALIDATION:

```
kf = StratifiedKFold(n_splits=n_splits, shuffle=True,
random_state=69)

for train_index, test_index in kf.split(X_Data, Y_Lavel):
    X_Train, X_Test = X_Data[train_index], X_Data[test_index]
    Y_Train, Y_Test = Y_Lavel[train_index], Y_Lavel[test_index]
```

This code performs Stratified K-Fold Cross-Validation, a technique used to evaluate the generalizability of a machine learning model. It first creates an instance of Stratified K-Fold, specifying the number of folds (sets to split the data into) and enabling shuffling for better data distribution. Then, it iterates through each fold generated by the Stratified K-Fold object. In each iteration, it splits the data and target variable (features and labels) into training and testing sets based on the provided indices. This process ensures the class distribution (proportion of each class) is preserved in both training and testing sets. By iterating through multiple folds, training the model on each unique training set and evaluating it on the corresponding testing set, this approach provides a more robust and reliable assessment of the model's performance and its ability to generalize to unseen data.

6.2.4 MACHINE LEARNING MODELS:

CATEGORICAL BOOSTING:

```
catboost_params = {
        'iterations': 900,
        'learning_rate': 0.05,
        'depth': 5,
        'l2_leaf_reg': 3,
        'loss_function': 'Logloss',
        'verbose': 0,
        'random_seed': 69
    }

    clf = CatBoostClassifier(**catboost_params)
    clf.fit(X_Train, Y_Train)

    y_pred_proba = clf.predict_proba(X_Test)[:, 1]
```

This code snippet showcases the application of the CatBoost algorithm for classification. It defines parameters like the number of iterations, learning rate, tree depth, and regularization strength, then creates a CatBoostClassifier object with these parameters. Subsequently, it trains the model on the provided training data (X_Train, Y_Train). Finally, it uses the trained model to predict class probabilities (y_pred_proba) for the unseen testing data (X_Test), indicating the likelihood of each data point belonging to the positive class.

XG BOOSTING

```
xgboost = xgb.XGBClassifier(n_estimators=15000, learning_rate=0.09,
max_depth=3, random_state=69)
    xgboost.fit(X_Train, Y_Train)

    y_pred_proba = xgboost.predict_proba(X_Test)[:, 1]
```

This code snippet utilizes the XGBoost algorithm for classification. It defines hyperparameters like the number of trees (n_estimators), learning rate, and maximum tree depth (max_depth), then creates an XGBClassifier object with these settings. The model is then trained using the provided training data (X_Train, Y_Train). Finally, it predicts class probabilities (y_pred_proba) for the unseen testing data (X_Test), indicating the likelihood of each data point belonging to the positive class.

RANDOM FOREST

```
rf = RandomForestClassifier(n_estimators=100, max_depth=10,
min_samples_split=2, min_samples_leaf=1, random_state=69)

    rf.fit(X_Train, Y_Train)

    y_pred_proba = rf.predict_proba(X_Test)[:, 1]
```

This code snippet initializes a Random Forest classifier with specific hyperparameters such as 100 estimators (trees), a maximum depth of 10 for each tree, and minimum samples required to split a node set to 2. Additionally, it specifies a minimum number of samples required to be at a leaf node as 1, and sets the random state for reproducibility. Following initialization, the classifier is trained on the training data (X_Train, Y_Train). Subsequently, it predicts the probabilities of the positive class for the test data (X_Test) using the trained model. Random Forest algorithm, a type of ensemble learning method, constructs multiple decision trees during training and combines their predictions to improve accuracy and robustness.

K NEAREST NEIGHBOR

```
knn = KNeighborsClassifier(n_neighbors=5)
   knn.fit(X_Train, Y_Train)

   y_pred_proba = knn.predict_proba(X_Test)[:, 1]
```

This code initializes a k-Nearest Neighbors (KNN) classifier with a parameter k set to 5. It then trains the classifier on the training data (X_Train, Y_Train). Following training, the classifier predicts the probabilities of the positive class for the test data (X_Test) using the trained model. KNN is a simple and intuitive machine learning algorithm used for classification tasks. It operates by finding the k nearest data points to a given query point and assigns the class label based on the majority class among these neighbors.

6.2.5 ENSEMBLING THE MACHINE LEARNING MODELS:

```
catboost = CatBoostClassifier(iterations=1500, learning_rate=0.05,
depth=5, l2_leaf_reg=3, loss_function='Logloss', verbose=0,
random_seed=69)
    catboost.fit(X_Train, Y_Train)


    rf = RandomForestClassifier(n_estimators=500, random_state=69)
    rf.fit(X_Train, Y_Train)


    ensemble = VotingClassifier(estimators=[('catboost', catboost),
('rf', rf)], voting='hard')
    ensemble.fit(X_Train, Y_Train)
    y_pred_ensemble = ensemble.predict(X_Test)
```

This code snippet showcases the creation of an ensemble model by combining predictions from two distinct classifiers, CatBoost and Random Forest. After initializing and training both classifiers on the training data, the Voting Classifier from scikit-learn is utilized to aggregate their predictions using a 'soft' voting strategy. This approach leverages the diversity of predictions from different algorithms to potentially enhance the overall accuracy and robustness of the final classification.

6.2.6 SAVING THE BEST MODEL STATE:

```
import pickle
with open('female_model.sav','wb') as file:
  pickle.dump(ensemble,file)
import pickle
with open('male_model.sav','wb') as file:
  pickle.dump(ensemble,file)
```

The code snippet utilizes the Python pickle module to serialize an object named ensemble and save it to a file named saved_model.pkl in binary mode ('wb'). This approach is commonly used in Python for saving machine learning models, trained classifiers, or any other Python objects that need to be persisted to disk for later use.

6.2.6 DESIGNING A WEB APPLICATION USING FLASK AND REACTJS:

FLASK SERVER MODULE :

```
app = Flask(__name__)
CORS(app,resources={r'/*':{'origins':'*'}})
```

Figure. 6.1. Flask Application

```
male_diabetes_model = pickle.load(open(r'.\backend\\male_model.sav','rb'))
female_diabetes_model = pickle.load(open(r'.\backend\\female_model.sav','rb'))
```

Figure. 6.2. Loading saved models

```
def Validate(validation_list,Gender):
def Empty_Validation(data,Gender):
def Data_type_Validation(data,Gender):
```

Figure. 6.3. Validation Methods

```
@app.route('/predict/female', methods=['POST'])
def predict_female():

@app.route('/predict/male', methods=['POST'])
def predict_male():
```

Figure. 6.4. API call handlers

```
prediction = male_diabetes_model.predict(sc_male.transform([[Glucose,BP,Skin,Insulin,BMI,DPF,Age]]))
        print(prediction)
        if prediction[0] == 1:
            result = {"prediction": "The patient is at the risk of having diabetes"}
            return jsonify(result)
        result = {"prediction": "The patient is healthy"}
        return jsonify(result)

prediction = female_diabetes_model.predict(sc_female.transform([[Pregnancy,Glucose,BP,Skin,Insulin,BMI,DPF,Age]]))
        print(prediction)
        if prediction[0] == 1:
            result = {"prediction": "The patient is at the risk of having diabetes"}
            return jsonify(result)
        result = {"prediction": "The patient is healthy"}
        return jsonify(result)
```

Figure. 6.5. Diabetes prediction code snippets

This Flask server code powers the backend of our diabetes prediction application. It handles incoming requests from the React frontend. The server first validates the user-provided data for any missing entries or invalid values based on gender. Then, it uses pre-trained models (separate for males and females) to make predictions on the user's data. Finally, it sends a JSON response back to the frontend indicating whether the user is predicted to be at risk of diabetes or healthy.

FRONT END MODULES FOR REACTJS:

Figure. 6.6. ReactJS components

The code that is present in the above components, build the user interface for our diabetes prediction application. It has separate sections for male and female users. Users can enter their data points like blood sugar, blood pressure, BMI, etc,... Then, the user input is

handled by using React's state management and makes calls to a Flask server backend when the user clicks "Predict." The backend performs the prediction based on the user's data and gender, and sends the result back. This React code then displays the prediction (e.g., "at risk of diabetes" or "healthy") to the user.

CHAPTER 7

TESTING

7.1 TESTING STRATEGY:

In this section, we outline the testing strategy employed to ensure the robustness and reliability of our diabetes prediction system. The testing strategy encompasses various aspects, including unit testing, integration testing, and end-to-end testing. Unit testing focuses on validating the functionality of individual components, ensuring they perform as expected. Integration testing evaluates the seamless collaboration of different modules within the system. End-to-end testing examines the entire system workflow, from data input in the frontend to prediction generation in the backend. Additionally, stress testing and edge case scenarios are considered to assess the system's resilience under varying conditions.

UNIT TESTING:

We conducted unit testing to validate the functionality of critical components, including data preprocessing techniques and individual machine learning models. Each module underwent rigorous testing in isolation to verify its correctness and adherence to specifications.

INTEGRATION TESTING:

Our integration testing focused on examining the collaboration between the React.js frontend and Flask backend. We ensured smooth data exchange and verified the seamless integration of machine learning models with the backend API through Postman application. This phase also assessed the proper functioning of the overall system architecture.

END-TO-END TESTING:

Comprehensive end-to-end testing simulated user interactions, covering the entire workflow from user input in the frontend to the generation of predictions by the backend. This phase scrutinized the accuracy of predictions, data flow consistency, and the system's overall performance.

EDGE CASE TESTING:

Our testing strategy included evaluating system behavior in edge cases and unexpected scenarios. We tested with invalid inputs, empty inputs and wrong data type inputs ensuring that the system handles extreme cases gracefully without compromising accuracy.

7.2 TEST CASES:

Table. 7.1. Test Cases

S.NO	Test Case Name	Description	Expected Output	Actual Output	Test Status (P/F)
1	TC1:Valid Inputs (Female/Male)	all valid inputs for female/male diabetes model were given	The patient is at the risk of having diabetes	The patient is at the risk of having diabetes	P
2	TC2:Valid Inputs (Female/Male)	all valid inputs for female/nale diabetes model were given	The patient is healthy	The patient is healthy	P
3	TC3:Invalid Inputs (Female/Male)	-ve or invalid values are provided as inputs	Invalid Inputs	Invalid Inputs	P
4	TC4:Invalid Inputs (Female/Male)	Other data type values are provided as inputs	Invalid Data Type Input	Invalid Data Type Input	P
5	TC5:Empty Inputs (Female/Male)	One or more input fields are left blank	Empty Inputs	Empty Inputs	P

The test cases outlined serve to systematically evaluate the functionality of our diabetes prediction system across different input scenarios. Valid inputs for both female and male models are accurately processed, correctly identifying the risk of diabetes or confirming the patient's health status. The system demonstrates robustness by effectively handling invalid inputs, such as negative values or incorrect data types, and appropriately flagging them as invalid. Additionally, empty input fields prompt the user to provide all necessary information, ensuring comprehensive assessment. Overall, these test cases affirm the reliability and accuracy of our system in predicting diabetes risk while accommodating diverse input scenarios.

CHAPTER 8

RESULT AND DISCUSSION

8.1 RESULTS:

The results of the study indicate that the proposed ensemble model, particularly the combination of CatBoost and Random Forest (RF), outperforms other models in predicting the risk of diabetes in patients with a correctness rate of 97%. The evaluation was based on several performance metrics including accuracy, precision, sensitivity, specificity, False Omission Rate (FOR), and AUC-ROC.

ACCURACY: The CatBoost algorithm, when combined with RF, achieved the highest accuracy of 97%, indicating the proportion of correctly classified instances out of the total number of instances.

PRECISION: Precision measures the proportion of accurately predicted positive events to the total number of events identified as true. While precision values were not explicitly mentioned, it can be inferred that precision rates were high considering the overall high accuracy of the model.

SENSITIVITY: Sensitivity, or the true positive rate, measures the model's ability to accurately identify positive occurrences from the entire set of actual positive instances. Specific sensitivity values were not provided, but given the high accuracy rate, it can be assumed that sensitivity was also high.

SPECIFICITY: Specificity measures the model's ability to accurately detect true negative situations out of all the real negative instances. The CatBoost+RF ensemble exhibited strong specificity, contributing to its overall high accuracy.

FALSE OMISSION RATE (FOR): FOR quantifies the tendency of a model to incorrectly categorize negative cases as positive. By subtracting specificity from 1, FOR can be inferred. Since specificity was high, FOR was likely low.

AUC-ROC (RECEIVER OPERATING CHARACTERISTIC): The AUC-ROC score evaluates a model's capacity to distinguish between correctly classified positive classes to wrongly classified positive classes. The CatBoost+RF ensemble likely achieved a high AUC-ROC score, given its high accuracy rate.

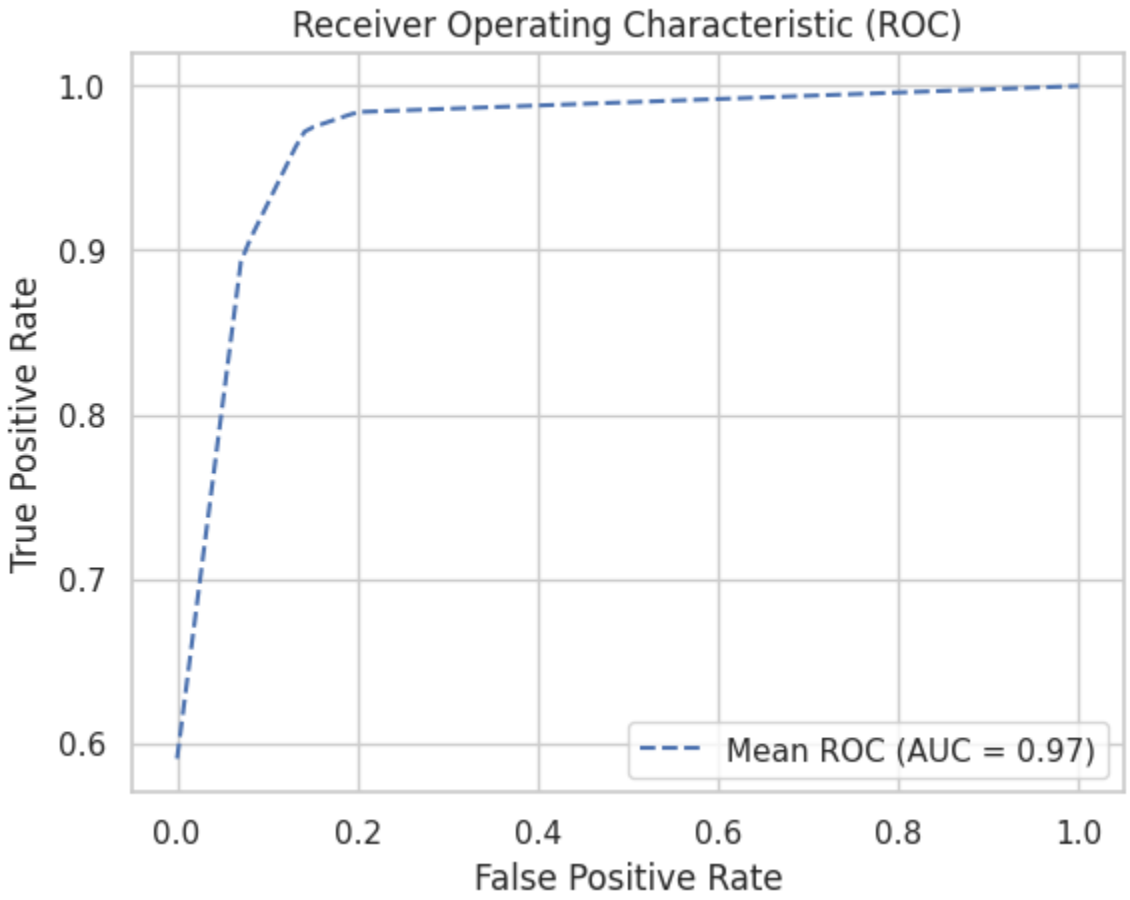

Figure. 8.1. AUC Curve for CatBoost+RF model

Accuracy (Avg. +/- Std.) is 0.970 +/- 0.033
Sensitivity (Avg. +/- Std.) is 0.981 +/- 0.031
Specificity (Avg. +/- Std.) is 0.958 +/- 0.058
Precision (Avg. +/- Std.) is 0.962 +/- 0.052
FOR (Avg. +/- Std.) is 0.018 +/- 0.030
DOR (Avg. +/- Std.) is inf +/- nan

The ensemble model consisting of CatBoost and Random Forest classifiers achieved a prediction accuracy of 97%.

The React-based front-end provides a user-friendly interface with input fields for entering relevant data points, such as glucose levels, BMI, etc. For male users, factors such as age, BMI, family history, Insulin levels etc are considered, excluding pregnancy. Conversely, for female users, pregnancy history is also factored in.

This personalized approach enhances the relevance and utility of the tool, empowering individuals to make informed decisions about their health. Upon receiving user input, the

application utilizes the pre-trained and deserialized ensemble model (stored using Pickle) to make predictions. This approach eliminates the need for retraining the model on every prediction request. After analysis, the system presents a risk assessment along with tailored tips and recommendations to mitigate the risk of diabetes.

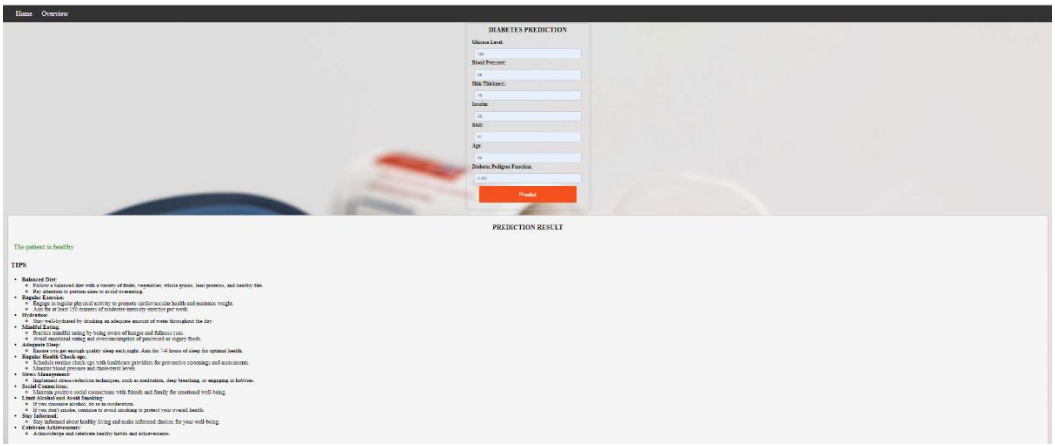

Figure. 8.2. Model Prediction – Patient is healthy

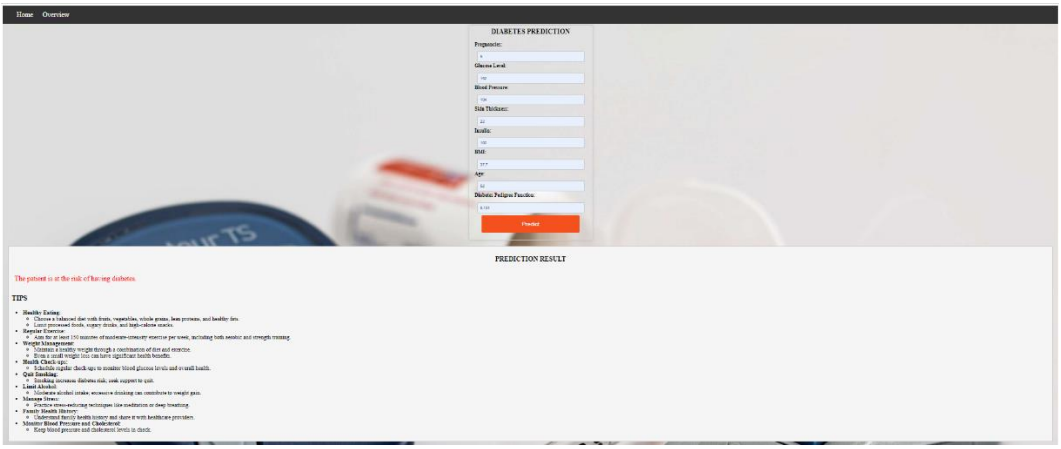

Figure. 8.3. Model Prediction – Patient is at risk of developing diabetes

8.2 DISCUSSION:

The results suggest that the ensemble model, particularly the combination of CatBoost and RF, provides superior performance in predicting diabetes risk compared to individual models. The high accuracy rate indicates the effectiveness of the ensemble approach in combining the strengths of different machine learning algorithms.

The study also highlights the importance of data preprocessing techniques and parameter optimization in improving model performance. By employing techniques such as class-aware imputation and considering all features rather than employing feature removal, the model avoids potential biases and overfitting issues.

Overall, the proposed ensemble model holds great potential in enhancing the efficiency of diabetes prediction systems and can be applied to different datasets, thereby improving the quality of diabetes diagnosis.

CHAPTER 9:

CONCLUSION AND FUTURE WORK

9.1 CONCLUSION:

In conclusion, this study introduces a predictive framework designed to accurately assess the risk of diabetes by thoroughly analysing critical health parameters. Our methodology has yielded exceptional results, particularly with the Cat Boost algorithm combined with Random Forest (RF), achieving an outstanding individual accuracy rate of 97%. These results underscore the effectiveness of our approach in accurately predicting diabetes risk.

Moving forward, we anticipate further validation of our framework using real-world clinical data to ensure its robustness across various healthcare contexts. By validating our methodology with clinical data, we can enhance its applicability and reliability in real-world scenarios, ultimately benefiting patients and healthcare providers alike.

9.2 FUTURE WORK:

Future work on this framework will focus on several key areas to enhance its usability and effectiveness. One area of improvement involves developing a user-friendly interface to facilitate easier adoption and utilization by healthcare professionals. By creating an intuitive interface, we can streamline the process of inputting data and interpreting results, ultimately improving the framework's usability and accessibility.

Additionally, further research can explore the integration of additional data sources and parameters to enhance the predictive capabilities of the framework. By incorporating a broader range of health indicators and demographic information, we can refine the accuracy and reliability of diabetes risk predictions.

Furthermore, ongoing efforts will involve continuous optimization of the framework's algorithms and parameters to ensure optimal performance in various healthcare settings. By staying abreast of advancements in machine learning techniques and healthcare analytics, we can continually improve the framework's predictive accuracy and applicability. In summary, future work will focus on refining the framework's usability, expanding its predictive capabilities, and optimizing its performance to better serve the needs of patients and healthcare providers in the ongoing fight against diabetes.

REFERENCES & BIBLIOGRAPHY

[1] Ashmia Singh, Arwinder Dhillon, and Neeraj Kumar, M. Shamim Hossain, Ghulam Muhammad, Manoj Kumar, ”eDiaPredict: An Ensemble based Framework for Diabetes Prediction”,doi: 10.1145/3415155, June 2021.

[2] Joy Dhar, Nigus Asres Ayele, "Multi-Tier Ensemble Learning Model With Neighborhood Component Analysis to Predict Health Diseases", doi: 10.1109/ACCESS.2021.3117963, October 2021.

[3] Norma Latif Fitriyani, Muhammad Syafrudin, Ganjar Alfain, Jongtae Rhee, "Development of Disease Prediction Model Based on Ensemble Learning Approach for Diabetes and Hypertension", doi: 10.1109/ACCESS.2019.2945129, October 2019.

[4] Muhammad Exell Febriana, Fransiskus Xaverius Ferdinana, "Diabetes prediction using supervised machine learning", doi: 10.1016/j.procs.2022.12.107, 2022.

[5] MD.Kamrul Hasan, MD. Ashraful Alam, Dola Das, Eklas Hossain, Mahmudul Hasan, "Diabetes Prediction Using Ensembling of Different Machine Learning Classifiers", doi: 10.1109/ACCESS.2020.2989857, April 23, 2020.

[6] Ashmia Singh, Arwinder Dhillon, and Neeraj Kumar,M. Shamim Hossain, Ghulam Muhammad, Manoj Kumar, "eDiaPredict: An Ensemble-based Framework for Diabetes Prediction", doi: 10.1145/3415155, June 2021.

[7] Priyanka Rajendra, Shahram Latif, "Prediction of diabetes using logistic regression and ensemble techniques", doi: 10.1016/j.cmpbup.2021.100032, October 25, 2021.

[8] Pratya Nuankaew, Supansa Chaising, Punnarumol Temdee, "Average Weighted Objective Distance-Based Method for Type 2 Diabetes Prediction", doi: 10.1109/ACCESS.2021.3117269, October 2021.

[9] Talha Mahboob Alama, Muhammad Atif Iqbala, Yasir Alia, "A model for early prediction of diabetes", doi: 10.1016/j.imu.2019.100204, July 2019.

[10] Roosa Peramaki, Mika Gisslerb, "The risk of developing type 2 diabetes after gestational diabetes: A registry study from Finland", doi: 10.1016/j.deman.2022.100124, 2022.

[11] Srinivasu, P. N., Shafi, J., Krishna, T. B., Sujatha, C. N., Praveen, S. P., & Ijaz, M. F. (2022). Using recurrent neural networks for predicting type-2 diabetes from genomic and tabular data. Diagnostics, 12(12), 3067.

[12] Martin Dodek, Eva Miklovicova, Marian Tarnik, "Correlation Method for Identification of a Nonparametric Model of Type 1 Diabetes", doi: 10.1109/ACCESS.2022.3212435, October 2022.

[13] Virginie Felizardo, Diogo Machado, Nuno M. Garcia, "Hypoglycaemia Prediction Models With Auto Explanation", doi: 10.1109/ACCESS.2021.3117340, October 2021.

[14] S. P. Praveen, S. Sindhura, A. Madhuri and D. A. Karras, "A Novel Effective Framework for Medical Images Secure Storage Using Advanced Cipher Text Algorithm in Cloud Computing," 2021 IEEE International Conference on Imaging Systems and Techniques (IST), Kaohsiung, Taiwan, 2021, pp. 1-4, doi: 10.1109/IST50367.2021.9651475.

[15] Jie Zhang, Fang Wang, "Prediction of Gestational Diabetes Mellitus under Cascade and Ensemble Learning Algorithm", doi: 10.1155/2022/3212738, 2022.

[16] Tadao Ooka , Hisashi Johno, Kazunori Nakamoto, "Random forest approach for determining risk prediction and predictive factors of type 2 diabetes: large-scale health check-up data in Japan", doi:10.1136/bmjnph-2020-000200, 2021.

[17] Swamy, S. R., Praveen, S. P., Ahmed, S., Srinivasu, P. N., & Alhumam, A. (2023). Multi-features disease analysis based smart diagnosis for covid-19. Computer Systems Science and Engineering, 45(1), 869-886.

[18] Ruihu Wang, "AdaBoost for Feature Selection, Classification and Its Relation with SVM, A Review", doi: 10.1016/j.phpro.2012.03.160 , 2012.

[19] Ziyue Yu, Wuman Luo, Rita Tse, Giovanni Pau, "DMNet: A Personalized Risk Assessment Framework for Elderly People With Type 2 Diabetes", doi:10.1109/JBHI.2022.3233622, 2023.

[20] Praveen, S. P., Jyothi, V. E., Anuradha, C., VenuGopal, K., Shariff, V., & Sindhura, S. (2022). Chronic Kidney Disease Prediction Using ML-Based Neuro-Fuzzy Model. International Journal of Image and Graphics, 2340013.

APPENDIX A

A.1 FULL CODE

INSTALLING NECESSARY MODULE:

```
!pip install catboost
```

READING DATASET:

```
import pandas as pd
data=pd.read_csv('diabetes.csv')
data.head(10)
data.dtypes
data.isnull().sum()
```

DATA PRE-PROCESSING:

CLASS AWARE MEAN IMPUTATION:

```
data[['BloodPressure', 'SkinThickness', 'Insulin', 'BMI']] = data.groupby('Outcome')[['BloodPressure',
'SkinThickness', 'Insulin', 'BMI']].transform(lambda x: x.replace(0, x.mean()))
```

IMPORTING NECESSARY LIBRARIES:

```
from sklearn.model_selection import train_test_split
import xgboost as xgb
from scipy import stats
from scipy.stats import uniform, randint
from sklearn.metrics import f1_score
from sklearn.model_selection import KFold, StratifiedKFold, RepeatedStratifiedKFold
from sklearn.metrics import roc_curve, auc, accuracy_score
from sklearn import preprocessing
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn.ensemble import VotingClassifier
from sklearn.metrics import roc_auc_score, accuracy_score
from sklearn import metrics
import numpy as np
import matplotlib.pyplot as plt
from scipy import interp
from sklearn.metrics import confusion_matrix
from catboost import CatBoostClassifier
from sklearn.ensemble import RandomForestClassifier
```

METHOD FOR INTER-QUARTILE OUTLIER REJECTION:

```
def Manual(data, numeric_columns=None, outlier_factor=1.5, keep_outliers=False):
    data_copy = data.copy()

    if numeric_columns is None:
        numeric_columns = data_copy.select_dtypes(include=['number']).columns

    for col in numeric_columns:
        Q1 = data_copy[col].quantile(0.25)
        Q3 = data_copy[col].quantile(0.75)
        IQR = Q3 - Q1


        lower_bound = Q1 - outlier_factor * IQR
        upper_bound = Q3 + outlier_factor * IQR

        if keep_outliers:
            data_copy = data_copy[(data_copy[col] < lower_bound) | (data_copy[col] > upper_bound)]
        else:
            data_copy = data_copy[(data_copy[col] >= lower_bound) & (data_copy[col] <=
upper_bound)]

    return data_copy
```

FEATURE SELECTION AND STANDARD SCALING:

```
data = Manual(data)
data = data.reset_index(drop=True)

Y_Lavel = data.iloc[:, -1]

from sklearn.preprocessing import StandardScaler
sc = StandardScaler()

X_Data = data.drop(['Outcome'],axis=1).values
X_Data = sc.fit_transform(X_Data)
from sklearn.feature_selection import SelectKBest, f_classif
selector = SelectKBest(score_func=f_classif, k='all')
selector.fit(X_Data, Y_Lavel)
selected_features_indices = selector.get_support(indices=True)
print(selected_features_indices)
X_Data = selector.transform(X_Data)
```

RANDOM OVERSAMPLER FOR CLASS IMBALANCE:

```
from imblearn.over_sampling import RandomOverSampler
ros = RandomOverSampler(random_state=69)
X_Data, Y_Lavel = ros.fit_resample(X_Data, Y_Lavel)

n_splits = 30
kf = KFold(n_splits=n_splits, shuffle=True, random_state=69)
```

LIST CREATION FOR STORING ACCURACIES OF ALL MODELS:

```
results = []
```

RANDOM FOREST:

```
from sklearn.ensemble import RandomForestClassifier

n_splits = 30

Accuracy = []
FP = []
TN = []
FN = []
TP = []
tprs = []
aucs_rf = []
sn = []
sp = []
pr = []
FOR = []
DOR = []
iterator = 0
mean_fpr = np.linspace(0, 1, 100)
fig = plt.figure(figsize=(8, 5))
kf = StratifiedKFold(n_splits=n_splits, shuffle=True, random_state=69)

for train_index, test_index in kf.split(X_Data, Y_Lavel):
    X_Train, X_Test = X_Data[train_index], X_Data[test_index]
    Y_Train, Y_Test = Y_Lavel[train_index], Y_Lavel[test_index]

    rf = RandomForestClassifier(n_estimators=100, max_depth=10, min_samples_split=2,
min_samples_leaf=1, random_state=69)

    rf.fit(X_Train, Y_Train)

    y_pred_proba = rf.predict_proba(X_Test)[:, 1]
```

```
    tn, fp, fn, tp = confusion_matrix(Y_Test, (y_pred_proba > 0.5).astype(int)).ravel()
    fpr, tpr, _ = roc_curve(Y_Test, y_pred_proba)
    roc_auc = auc(fpr, tpr)

    tprs.append(interp(mean_fpr, fpr, tpr))
    tprs[-1][0] = 0.0
    aucs_rf.append(roc_auc)

    TN.append(tn)
    FP.append(fp)
    FN.append(fn)
    TP.append(tp)

    accuracy = accuracy_score(Y_Test, (y_pred_proba > 0.5).astype(int))
    Accuracy.append(accuracy)

    sn.append(tp / (tp + fn))
    sp.append(tn / (fp + tn))
    pr.append(tp / (tp + fp))
    FOR.append(fn / (tn + fn))
    DOR.append((tp * tn) / (fp * fn))

    plt.plot(fpr, tpr, lw=1, alpha=0.3, label='ROC fold %d (AUC = %0.2f)' % (iterator, roc_auc))

    iterator += 1

mean_tpr = np.mean(tprs, axis=0)
mean_tpr[-1] = 1.0
mean_auc = auc(mean_fpr, mean_tpr)

plt.plot(mean_fpr, mean_tpr, color='b', linestyle='--', label='Mean ROC (AUC = %0.2f)' % mean_auc)

plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC)')
plt.legend(loc='lower right')
plt.show()

print("Accuracy (Avg. +/- Std.) is %0.3f +/- %0.3f" % (np.mean(Accuracy), np.std(Accuracy)))
print("Sensitivity (Avg. +/- Std.) is %0.3f +/- %0.3f" % (np.mean(sn), np.std(sn)))
print("Specificity (Avg. +/- Std.) is %0.3f +/- %0.3f" % (np.mean(sp), np.std(sp)))
print("Precision (Avg. +/- Std.) is %0.3f +/- %0.3f" % (np.mean(pr), np.std(pr)))
print("FOR (Avg. +/- Std.) is %0.3f +/- %0.3f" % (np.mean(FOR), np.std(FOR)))
print("DOR (Avg. +/- Std.) is %0.3f +/- %0.3f" % (np.mean(DOR), np.std(DOR)))

results.append({
        "Model Algorithm Name": "RANDOM FOREST",
        "Accuracy": np.mean(Accuracy),
```

```
    "Sensitivity":np.mean(sn),
    "Specificity": np.mean(sp),
    "Precision":np.mean(pr),
    "FOR": np.mean(FOR),
    "DOR": np.mean(DOR),
})
```

CATBOOST:

```
from catboost import CatBoostClassifier
n_splits = 30

Accuracy = []
FP = []
TN = []
FN = []
TP = []
tprs = []
aucs_catboost = []
sn = []
sp = []
pr = []
FOR = []
DOR = []
iterator = 0
mean_fpr = np.linspace(0, 1, 100)
fig = plt.figure(figsize=(8, 5))
kf = StratifiedKFold(n_splits=n_splits, shuffle=True, random_state=69)

for train_index, test_index in kf.split(X_Data, Y_Lavel):
    X_Train, X_Test = X_Data[train_index], X_Data[test_index]
    Y_Train, Y_Test = Y_Lavel[train_index], Y_Lavel[test_index]

    catboost_params = {
        'iterations': 900,
        'learning_rate': 0.05,
        'depth': 5,
        'l2_leaf_reg': 3,
        'loss_function': 'Logloss',
        'verbose': 0,
        'random_seed': 69
    }

    clf = CatBoostClassifier(**catboost_params)
    clf.fit(X_Train, Y_Train)

    y_pred_proba = clf.predict_proba(X_Test)[:, 1]
```

```
    tn, fp, fn, tp = confusion_matrix(Y_Test, (y_pred_proba > 0.5).astype(int)).ravel()
    fpr, tpr, _ = roc_curve(Y_Test, y_pred_proba)
    roc_auc = auc(fpr, tpr)

    tprs.append(interp(mean_fpr, fpr, tpr))
    tprs[-1][0] = 0.0
    aucs_catboost.append(roc_auc)

    TN.append(tn)
    FP.append(fp)
    FN.append(fn)
    TP.append(tp)

    accuracy = accuracy_score(Y_Test, (y_pred_proba > 0.5).astype(int))
    Accuracy.append(accuracy)

    sn.append(tp / (tp + fn))
    sp.append(tn / (fp + tn))
    pr.append(tp / (tp + fp))
    FOR.append(fn / (tn + fn))
    DOR.append((tp * tn) / (fp * fn))

    plt.plot(fpr, tpr, lw=1, alpha=0.3, label='ROC fold %d (AUC = %0.2f)' % (iterator, roc_auc))

    iterator += 1


mean_tpr = np.mean(tprs, axis=0)
mean_tpr[-1] = 1.0
mean_auc = auc(mean_fpr, mean_tpr)


plt.plot(mean_fpr, mean_tpr, color='b', linestyle='--', label='Mean ROC (AUC = %0.2f)' % mean_auc)


plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC)')
plt.legend(loc='lower right')
plt.show()


print("Accuracy (Avg. +/- Std.) is %0.3f +/- %0.3f" % (np.mean(Accuracy), np.std(Accuracy)))
print("Sensitivity (Avg. +/- Std.) is %0.3f +/- %0.3f" % (np.mean(sn), np.std(sn)))
print("Specificity (Avg. +/- Std.) is %0.3f +/- %0.3f" % (np.mean(sp), np.std(sp)))
print("Precision (Avg. +/- Std.) is %0.3f +/- %0.3f" % (np.mean(pr), np.std(pr)))
print("FOR (Avg. +/- Std.) is %0.3f +/- %0.3f" % (np.mean(FOR), np.std(FOR)))
print("DOR (Avg. +/- Std.) is %0.3f +/- %0.3f" % (np.mean(DOR), np.std(DOR)))

results.append({
```

```
    "Model Algorithm Name": "CATEGORICAL BOOSTING",
    "Accuracy": np.mean(Accuracy),
    "Sensitivity":np.mean(sn),
    "Specificity": np.mean(sp),
    "Precision":np.mean(pr),
    "FOR": np.mean(FOR),
    "DOR": np.mean(DOR),
})
```

ENSEMBLING:

CATBOOST AND RANDOM FOREST ENSEMBLE:

```
n_splits = 30
Accuracy = []
FP = []
TN = []
FN = []
TP = []
tprs = []
sn = []
sp = []
pr = []
FOR = []
DOR = []
iterator = 0
mean_fpr = np.linspace(0, 1, 100)

kf = StratifiedKFold(n_splits=n_splits, shuffle=True, random_state=69)

for train_index, test_index in kf.split(X_Data, Y_Lavel):
    X_Train, X_Test = X_Data[train_index], X_Data[test_index]
    Y_Train, Y_Test = Y_Lavel[train_index], Y_Lavel[test_index]


    catboost = CatBoostClassifier(iterations=1500, learning_rate=0.05, depth=5, l2_leaf_reg=3,
loss_function='Logloss', verbose=0, random_seed=69)
    catboost.fit(X_Train, Y_Train)


    rf = RandomForestClassifier(n_estimators=500, random_state=69)
    rf.fit(X_Train, Y_Train)


    ensemble = VotingClassifier(estimators=[('catboost', catboost), ('rf', rf)], voting='hard')
    ensemble.fit(X_Train, Y_Train)
    y_pred_ensemble = ensemble.predict(X_Test)
```

```
    accuracy = accuracy_score(Y_Test, y_pred_ensemble)
    Accuracy.append(accuracy)


    tn, fp, fn, tp = confusion_matrix(Y_Test, y_pred_ensemble).ravel()
    TN.append(tn)
    FP.append(fp)
    FN.append(fn)
    TP.append(tp)


    sensitivity = tp / (tp + fn)
    specificity = tn / (tn + fp)
    precision = tp / (tp + fp)
    for_ = fn / (fn + tn)
    dor = (tp/fn) / (fp/tn)

    sn.append(sensitivity)
    sp.append(specificity)
    pr.append(precision)
    FOR.append(for_)
    DOR.append(dor)

    fpr, tpr, thresholds = roc_curve(Y_Test, y_pred_ensemble)
    tprs.append(np.interp(mean_fpr, fpr, tpr))

    iterator += 1


mean_tpr = np.mean(tprs, axis=0)
mean_tpr[-1] = 1.0
mean_auc = auc(mean_fpr, mean_tpr)

plt.plot(mean_fpr, mean_tpr, color='b', linestyle='--', label='Mean ROC (AUC = %0.2f)' % mean_auc)

plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC)')
plt.legend(loc='lower right')
plt.show()

print("Accuracy (Avg. +/- Std.) is %0.3f +/- %0.3f" % (np.mean(Accuracy), np.std(Accuracy)))
print("Sensitivity (Avg. +/- Std.) is %0.3f +/- %0.3f" % (np.mean(sn), np.std(sn)))
print("Specificity (Avg. +/- Std.) is %0.3f +/- %0.3f" % (np.mean(sp), np.std(sp)))
print("Precision (Avg. +/- Std.) is %0.3f +/- %0.3f" % (np.mean(pr), np.std(pr)))
print("FOR (Avg. +/- Std.) is %0.3f +/- %0.3f" % (np.mean(FOR), np.std(FOR)))
print("DOR (Avg. +/- Std.) is %0.3f +/- %0.3f" % (np.mean(DOR), np.std(DOR)))

results.append({
      "Model Algorithm Name": "CATEGORICAL BOOSTING AND RANDOM FOREST",
```

```
    "Accuracy": np.mean(Accuracy),
    "Sensitivity":np.mean(sn),
    "Specificity": np.mean(sp),
    "Precision":np.mean(pr),
    "FOR": np.mean(FOR),
    "DOR": np.mean(DOR),
})
```

FINAL MODELS FOR SYSTEM:

CATBOOST + RF MODEL FOR FEMALES:

```
from sklearn.preprocessing import StandardScaler
sc_female = StandardScaler()
X_Data = data.drop('Outcome',axis=1).values
Y_Lavel = data.iloc[:,-1].values
from imblearn.over_sampling import RandomOverSampler
ros = RandomOverSampler(random_state=69)
X_Data, Y_Lavel = ros.fit_resample(X_Data, Y_Lavel)

X_Data=sc_female.fit_transform(X_Data)
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test= train_test_split(X_Data,Y_Lavel,test_size=0.3,random_state=42)
from sklearn.ensemble import RandomForestClassifier
from catboost import CatBoostClassifier
catboost = CatBoostClassifier(iterations=1500, learning_rate=0.05, depth=5, l2_leaf_reg=3,
loss_function='Logloss', verbose=0, random_seed=69)
catboost.fit(X_train, y_train)
rf = RandomForestClassifier(n_estimators=500, random_state=69)
rf.fit(X_train, y_train)
ensemble = VotingClassifier(estimators=[('catboost', catboost), ('rf', rf)], voting='soft')
ensemble.fit(X_train, y_train)
```

CATBOOST+RF MODEL FOR MALES:

```
from sklearn.preprocessing import StandardScaler
sc_male = StandardScaler()
X_Data = data.drop(['Pregnancies','Outcome'],axis=1).values
Y_Lavel = data.iloc[:,-1].values
from imblearn.over_sampling import RandomOverSampler
ros = RandomOverSampler(random_state=69)
X_Data, Y_Lavel = ros.fit_resample(X_Data, Y_Lavel)
sc_male.fit(X_Data)
from sklearn.ensemble import RandomForestClassifier
from catboost import CatBoostClassifier
catboost = CatBoostClassifier(iterations=1500, learning_rate=0.05, depth=5, l2_leaf_reg=3,
loss_function='Logloss', verbose=0, random_seed=69)
```

```
catboost.fit(X_train, y_train)
rf = RandomForestClassifier(n_estimators=500, random_state=69)
rf.fit(X_train, y_train)
ensemble = VotingClassifier(estimators=[('catboost', catboost), ('rf', rf)], voting='soft')
ensemble.fit(X_train, y_train)
import pickle
with open('male_model.sav','wb') as file:
  pickle.dump(ensemble,file)
```

SAVING THE MODELS USING PICKLE MODULE:

```
import pickle
with open('female_model.sav','wb') as file:
  pickle.dump(ensemble,file)
import pickle
with open('male_model.sav','wb') as file:
  pickle.dump(ensemble,file)
```

BACKEND FLASK SERVER CODE:

```
from flask import Flask,request,jsonify
import pickle
import numpy as np
import pandas as pd
from sklearn.preprocessing import StandardScaler
from flask_cors import CORS
app = Flask(__name__)
CORS(app,resources={r'/*':{'origins':'*'}})

data = pd.read_csv(r'.\backend\diabetes.csv')

#Method to perform IQR
def Manual(data, numeric_columns=None, outlier_factor=1.5,
keep_outliers=False):
    data_copy = data.copy()

    if numeric_columns is None:
        numeric_columns =
data_copy.select_dtypes(include=['number']).columns

    for col in numeric_columns:
        Q1 = data_copy[col].quantile(0.25)
        Q3 = data_copy[col].quantile(0.75)
        IQR = Q3 - Q1


        lower_bound = Q1 - outlier_factor * IQR
```

```
        upper_bound = Q3 + outlier_factor * IQR

        if keep_outliers:
            data_copy = data_copy[(data_copy[col] < lower_bound) |
(data_copy[col] > upper_bound)]
        else:
            data_copy = data_copy[(data_copy[col] >= lower_bound) &
(data_copy[col] <= upper_bound)]

    return data_copy

#Data Preprocessing
data[['BloodPressure', 'SkinThickness', 'Insulin', 'BMI']] =
data.groupby('Outcome')[['BloodPressure', 'SkinThickness',
'Insulin', 'BMI']].transform(lambda x: x.replace(0, x.mean()))
data = Manual(data)
data = data.reset_index(drop=True)

#Creating Instances for Standard Scalers
from sklearn.preprocessing import StandardScaler
sc_female = StandardScaler()
sc_male = StandardScaler()

#Fitting the Standard Scaler Instances
X_Data = data.drop('Outcome', axis=1).values
sc_female.fit(X_Data)
X_Data = data.drop(['Outcome','Pregnancies'],axis=1).values
sc_male.fit(X_Data)


#importing respective models
male_diabetes_model =
pickle.load(open(r'.\backend\\male_model.sav','rb'))
female_diabetes_model =
pickle.load(open(r'.\backend\\female_model.sav','rb'))

#Functions for Validating Data
def Validate(validation_list,Gender):
    if Gender == 'female':
        Pregnancy,Glucose,BP,Skin,Insulin,BMI,DPF,Age =
validation_list
    else:
        Glucose,BP,Skin,Insulin,BMI,DPF,Age = validation_list
    if Gender == 'female':
        if Pregnancy < 0:
            return 'Invalid Input for Pregnancies'
    if Glucose <= 0:
        return 'Invalid Input for Glucose'
    if BP <= 0:
```

```
            return 'Invalid Input for Blood Pressure'
        if Skin <= 0:
            return 'Invalid Input for Blood Pressure'
        if Insulin <= 0:
            return 'Invalid Input for Insulin'
        if BMI <= 0:
            return 'Invalid Input for BMI'
        if DPF <=0:
            return 'Invalid Input for DPF'
        if Age <=0:
            return 'Invalid Input for Age'
        return 'true'
def Empty_Validation(data,Gender):
        if Gender == 'female':
            Pregnancy = data.get('feature1')
        Glucose = data.get('feature2')
        BP = data.get('feature3')
        Skin = data.get('feature4')
        Insulin = data.get('feature5')
        BMI = data.get('feature6')
        Age = data.get('feature8')
        DPF = data.get('feature7')
        if Gender == 'female':
            if Pregnancy == '' or Glucose == '' or BP == '' or Skin ==
'' or Insulin == '' or BMI == '' or DPF == '' or Age == '':
                return 'Empty Inputs'
        else:
            if Glucose == '' or BP == '' or Skin == '' or Insulin == ''
or BMI == '' or DPF == '' or Age == '':
                return 'Empty Inputs'
        return 'true'

def isGoodDataType(input):
        dot_count = 0
        neg_count = 0
        for i in input:
            if i>='0' and i<='9':
                continue
            if i=='.':
                dot_count += 1
                if dot_count ==1:
                    continue
            if i=='-':
                neg_count += 1
                if neg_count ==1:
                    continue
            return False
        return True
```

```
def Data_type_Validation(data,Gender):
    inputs = []
    if Gender == 'female':
        Pregnancy = data.get('feature1')
        inputs.append(Pregnancy)
    Glucose = data.get('feature2')
    BP = data.get('feature3')
    Skin = data.get('feature4')
    Insulin = data.get('feature5')
    BMI = data.get('feature6')
    Age = data.get('feature8')
    DPF = data.get('feature7')
    inputs.append(Glucose)
    inputs.append(BP)
    inputs.append(Skin)
    inputs.append(Insulin)
    inputs.append(BMI)
    inputs.append(Age)
    inputs.append(DPF)

    for i in inputs:
        if isGoodDataType(i)== False:
            return 'false'
    return 'true'


@app.route('/')
def home():
    return jsonify({"Ref":"Hello World"})

@app.route('/predict/female', methods=['POST'])
def predict_female():
    if request.method == 'POST':

        data = request.json

        validation_result = Empty_Validation(data,'female')
        if validation_result != 'true':
            return jsonify({'error':validation_result})
        validation_result = Data_type_Validation(data,'female')
        if validation_result != 'true':
            return jsonify({'error':'Data Type Error'})
        Pregnancy = float(data.get('feature1'))
        Glucose = float(data.get('feature2'))
        BP =  float(data.get('feature3'))
        Skin = float(data.get('feature4'))
        Insulin = float(data.get('feature5'))
        BMI = float(data.get('feature6'))
        Age = float(data.get('feature8'))
```

```
        DPF = float(data.get('feature7'))
        validation_list=[Pregnancy,Glucose,BP,Skin,Insulin,BMI,DPF,A
ge]
        validation_result = Validate(validation_list,'female')
        if validation_result != 'true':
            return jsonify({'error':validation_result})

        prediction =
female_diabetes_model.predict(sc_female.transform([[Pregnancy,Glucos
e,BP,Skin,Insulin,BMI,DPF,Age]]))
        print(prediction)
        if prediction[0] == 1:
            result = {"prediction": "The patient is at the risk of
having diabetes"}
            return jsonify(result)
        result = {"prediction": "The patient is healthy"}
        return jsonify(result)

@app.route('/predict/male', methods=['POST'])
def predict_male():
    if request.method == 'POST':
        data = request.json

        validation_result = Empty_Validation(data,'male')
        if validation_result != 'true':
            return jsonify({'error':validation_result})
        validation_result = Data_type_Validation(data,'male')
        if validation_result != 'true':
            return jsonify({'error':'Data Type Error'})
        Glucose = float(data.get('feature2'))
        BP =  float(data.get('feature3'))
        Skin = float(data.get('feature4'))
        Insulin = float(data.get('feature5'))
        BMI = float(data.get('feature6'))
        Age = float(data.get('feature8'))
        DPF = float(data.get('feature7'))
        validation_list=[Glucose,BP,Skin,Insulin,BMI,DPF,Age]
        validation_result = Validate(validation_list,'male')
        if validation_result != 'true':
            return jsonify({'error':validation_result})
        prediction =
male_diabetes_model.predict(sc_male.transform([[Glucose,BP,Skin,Insu
lin,BMI,DPF,Age]]))
        print(prediction)
        if prediction[0] == 1:
            result = {"prediction": "The patient is at the risk of
having diabetes"}
            return jsonify(result)
        result = {"prediction": "The patient is healthy"}
```

```
        return jsonify(result)

if __name__ == '__main__':
    app.run(debug=True)
```

FRONTEND REACTJS CODE:

Index.js:

```
import React from 'react';
import ReactDOM from 'react-dom/client';
import App from './App';
import reportWebVitals from './reportWebVitals';

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(
  <React.StrictMode>
    <App />
  </React.StrictMode>
);
```

App.js:

```
import React from 'react';
import { BrowserRouter, Route, Routes } from 'react-router-dom';
import LandingPage from './LandingPage';
import PredictionPage from './PredictionPage';
import './App.css';
import Navbar from './components/Navbar';
import Overview from './components/Overview';
import Male from './Male';
import Female from './Female';
const App = () => {
  return (
    <>
    <BrowserRouter>
    <Navbar/>
      <div className="app">
        <Routes>
          <Route path="/" exact element={<LandingPage/>} />
          <Route path="/predict" element={<PredictionPage/>} />
          <Route path="/overview" element={<Overview/>} />
          <Route path="/male" element={<Male/>}/>
          <Route path="/female" element={<Female/>}/>
```

```
      </Routes>
    </div>
  </BrowserRouter>
  </>

 );
};

export default App;
```

LandingPage.js:

```
import React from 'react';
import { Link } from 'react-router-dom';
const LandingPage = () => {
 return (<>
  <div className="landing-page">
    <h1>WELCOME TO DIABETES PREDICTION SYSTEM</h1>
    <Link to="/predict">
      <button className="predict-button">PREDICT DIABETES RISK</button>
    </Link>

  </div>
  </>
 );
};

export default LandingPage;
```

PredictionPage.js:

```
import { Link } from 'react-router-dom';
import './styles/Prediction.css'
const PredictionPage = () => {

 return (
  <>

    <div style={{"margin-top":"8em"}}>
    <Link to="/male">
      <button className="predict-button"style={{"margin-right":"3em"}}>MALE</button>
    </Link>
    <Link to="/female">
      <button className="predict-button">FEMALE</button>
    </Link>
```

```
      </div>

    </>
  );
};

export default PredictionPage;
```

Male.js:

```
import React from 'react'
import { useState } from 'react';
import './styles/Female.css';
import PredictionResult from './components/PredictionResult';
function Male() {

  const [prediction, setPrediction] = useState('');
  const handlePredict = async () => {
    try {
      const response = await fetch('http://localhost:5000/predict/male', {
        method: 'POST',
        headers: {
          'Content-Type': 'application/json',
        },
        body: JSON.stringify({
          "feature2": glucose, "feature3":bp,"feature4":skin,
          "feature5":insulin,"feature6":bmi,"feature7":dpf,"feature8":age
        })
      });

      if (!response.ok) {
        throw new Error('Network response was not ok');
      }

      const data = await response.json();
      setPrediction(data.prediction);
      var result = "Check For Results at the bottom section";
      if (data.error===undefined){
        alert(result);
      }
      else{
        alert(data.error);
      }

    } catch (error) {
      console.error('Error:', error);
    }
```

```
};


const [glucose, setGlucose] = useState('');
const [bp, setBP] = useState('');
const [skin, setSkin] = useState('');
const [insulin, setInsulin] = useState('');
const [bmi, setBMI] = useState('');
const [age, setAge] = useState('');
const [dpf, setDPF] = useState('');
return (
  <div style={{ marginTop: '-1em' }}>
    <div className="prediction-page">
      <h2>DIABETES PREDICTION</h2>
      <label htmlFor="feature2">Glucose Level:</label>
      <input
        type="text"
        id="feature2"
        value={glucose}
        onChange={(e) => setGlucose(e.target.value)}
      />
      <br />
      <label htmlFor="feature3">Blood Pressure:</label>
      <input
        type="text"
        id="feature3"
        value={bp}
        onChange={(e) => setBP(e.target.value)}
      />
      <br />
      <label htmlFor="feature4">Skin Thickness:</label>
      <input
        type="text"
        id="feature4"
        value={skin}
        onChange={(e) => setSkin(e.target.value)}
      />
      <br />
      <label htmlFor="feature5">Insulin:</label>
      <input
        type="text"
        id="feature5"
        value={insulin}
        onChange={(e) => setInsulin(e.target.value)}
      />
      <br />
      <label htmlFor="feature6">BMI:</label>
      <input
        type="text"
```

```
          id="feature6"
          value={bmi}
          onChange={(e) => setBMI(e.target.value)}
        />
        <br />
        <label htmlFor="feature7">Age:</label>
        <input
          type="text"
          id="feature7"
          value={age}
          onChange={(e) => setAge(e.target.value)}
        />
        <br />
        <label htmlFor="feature8">Diabetes Pedigree Function:</label>
        <input
          type="text"
          id="feature8"
          value={dpf}
          onChange={(e) => setDPF(e.target.value)}
        />
        <br />
        <button onClick={handlePredict} className="predict-button">
          Predict
        </button>
      </div>
      {prediction !== null && <PredictionResult prediction={prediction}/>}
    </div>
  )
}

export default Male
```

Female.js:

```
import React from 'react'
import { useState } from 'react';
import './styles/Female.css';
import PredictionResult from './components/PredictionResult';
function Female() {

  const [prediction, setPrediction] = useState('');
  const handlePredict = async () => {
    try {
      const response = await fetch('http://localhost:5000/predict/female', {
        method: 'POST',
        headers: {
          'Content-Type': 'application/json',
        },
        body: JSON.stringify({
```

```
      "feature1": preg, "feature2": glucose, "feature3":bp,"feature4":skin,
      "feature5":insulin,"feature6":bmi,"feature7":dpf,"feature8":age
    })
   });

   if (!response.ok) {
    throw new Error('Network response was not ok');
   }

   const data = await response.json();
   setPrediction(data.prediction);
   var result = "Check For Results at the bottom section";
   if (data.error === undefined){
    alert(result);
   }
   else{
    alert(data.error);
   }


  } catch (error) {
   console.error('Error:', error);
  }

};


const [preg, setPreg] = useState('');
const [glucose, setGlucose] = useState('');
const [bp, setBP] = useState('');
const [skin, setSkin] = useState('');
const [insulin, setInsulin] = useState('');
const [bmi, setBMI] = useState('');
const [age, setAge] = useState('');
const [dpf, setDPF] = useState('');
return (
  <div style={{ marginTop: '-1em' }}>
   <div className="prediction-page">
    <h2>DIABETES PREDICTION</h2>
    <label htmlFor="feature1">Pregnancies:</label>
    <input
     type="text"
     id="feature1"
     value={preg}
     onChange={(e) => setPreg(e.target.value)}
    />
    <br />
    <label htmlFor="feature2">Glucose Level:</label>
    <input
```

```
  type="text"
  id="feature2"
  value={glucose}
  onChange={(e) => setGlucose(e.target.value)}
/>
<br />
<label htmlFor="feature3">Blood Pressure:</label>
<input
  type="text"
  id="feature3"
  value={bp}
  onChange={(e) => setBP(e.target.value)}
/>
<br />
<label htmlFor="feature4">Skin Thickness:</label>
<input
  type="text"
  id="feature4"
  value={skin}
  onChange={(e) => setSkin(e.target.value)}
/>
<br />
<label htmlFor="feature5">Insulin:</label>
<input
  type="text"
  id="feature5"
  value={insulin}
  onChange={(e) => setInsulin(e.target.value)}
/>
<br />
<label htmlFor="feature6">BMI:</label>
<input
  type="text"
  id="feature6"
  value={bmi}
  onChange={(e) => setBMI(e.target.value)}
/>
<br />
<label htmlFor="feature7">Age:</label>
<input
  type="text"
  id="feature7"
  value={age}
  onChange={(e) => setAge(e.target.value)}
/>
<br />
<label htmlFor="feature8">Diabetes Pedigree Function:</label>
<input
  type="text"
```

```
          id="feature8"
          value={dpf}
          onChange={(e) => setDPF(e.target.value)}
        />
        <br />
        <button onClick={handlePredict} className="predict-button">
          Predict
        </button>
      </div>
      {prediction !== null && <PredictionResult prediction={prediction}/>}
    </div>
  )
}

export default Female
```

Overview.js:

```
import React from 'react'
import '../styles/overview.css'
function Overview() {
  return (
    <div style={{"textAlign":"left","background-color":"#ffffff"}}>
      <h2 style={{"font-size":"3vw","color": "rgb(0, 66, 136)" ,"padding":"10px"}}>Overview</h2>
      <p style={{"font-size":"1.9vw","color": "rgb(0, 0, 0)" }}>Diabetes is a chronic disease caused by
high blood glucose levels. Either the pancreas produces insufficient amounts of insulin or the body's
cells stop responding to hormones. There is an increasing need for creative solutions in early
prediction and healthcare management due to the rising global revalence
of diabetes. The model used in this system evaluates susceptibility to potential diabetes hazards in the
future. With a focus on early identification and the reduction of the burden associated with
complications related to diabetes, this model highlights the revolutionary potential of machine
learning in healthcare. An ensembling approach for diabetes rediction is implemented. The best model
is selected from a collection of diverse machine learning algorithms that includes Random Forest
(RF), K-Nearest Neighbors (KNN), XGboost, and Catboost. Using metrics like accuracy,  specificity,
sensitivity, precision, false omission rate, and Area Under Curve (AUC), the performance of the
developed model is assessed. The suggested model has a 97 percent accuracy rate after being trained
on the
PIMA Indian Diabetes dataset. This system adds to the continuing conversation about how artificial
intelligence can be used in the healthcare industry to promote proactive health management and better
patient outcomes.</p>
    </div>
  )
}

export default Overview
```

Navbar.js:

```
import React from 'react';
import '../styles/navbar.css';
import { Link } from 'react-router-dom';
const Navbar = () => {
  return (
    <nav className="navbar">
      <ul className="nav-links">
      <Link to="/"><li><a href="/" style={{"margin-left":"0em"}}>Home</a></li></Link>
      <Link to="/overview"><li><a href="/">Overview</a></li></Link>
      </ul>
    </nav>
  );
}

export default Navbar;
```

PredictionResult.js:

```
import React from 'react';
import CareTips from './CareTips';
import Tips from './Tips'
import Risk from './Risk';
import Healthy from './Healthy';
const PredictionResult = ({ prediction }) => {
    const predictionBoxStyle = {
        border: '1px solid #ccc',
        borderRadius: '5px',
        padding: '10px',
        margin: '20px 0',
        backgroundColor: '#f5f5f5',
      };
  return (
    <div style={predictionBoxStyle}>
      <h2>PREDICTION RESULT</h2>
      {prediction === "The patient is at the risk of having diabetes"?<Risk/>:prediction==="The patient
is healthy"?<Healthy/>:<></>}
      <h3 style={{'textAlign':'left','fontSize':'25px'}}>TIPS</h3>
      {prediction === "The patient is at the risk of having diabetes" && <CareTips/>}
      {prediction === "The patient is healthy"  &&<Tips/>}
    </div>
  );
};

export default PredictionResult;
```

A.2 SCREENSHOTS

Figure. A.1. Google Colab code

Figure. A.2. Result analysis chart

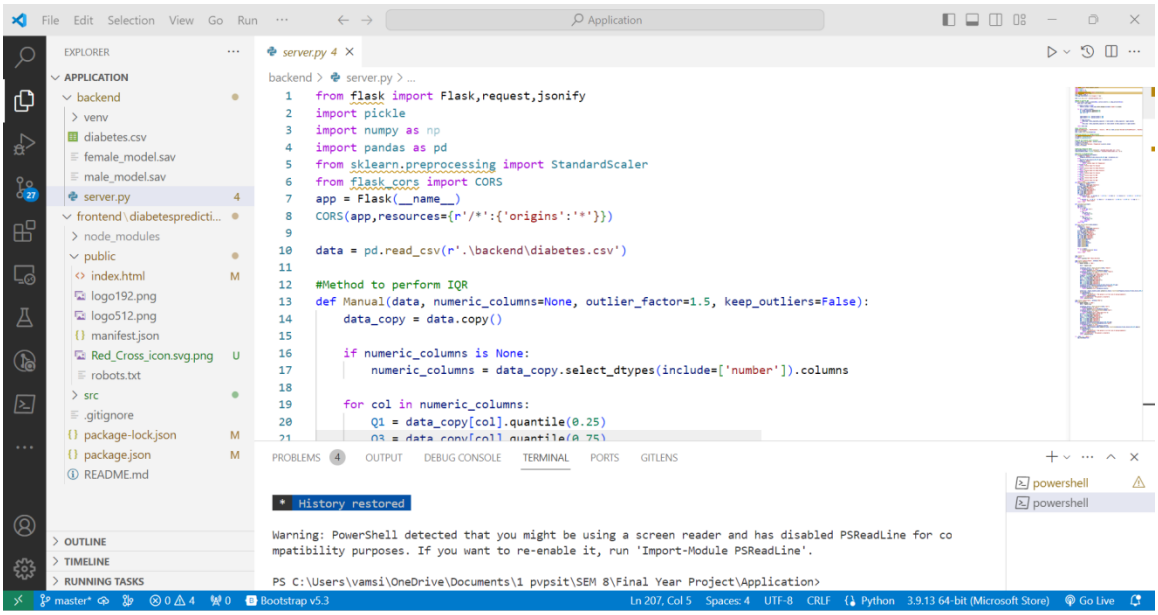

Figure. A.3. Flask server code

Figure. A.4. Landing page

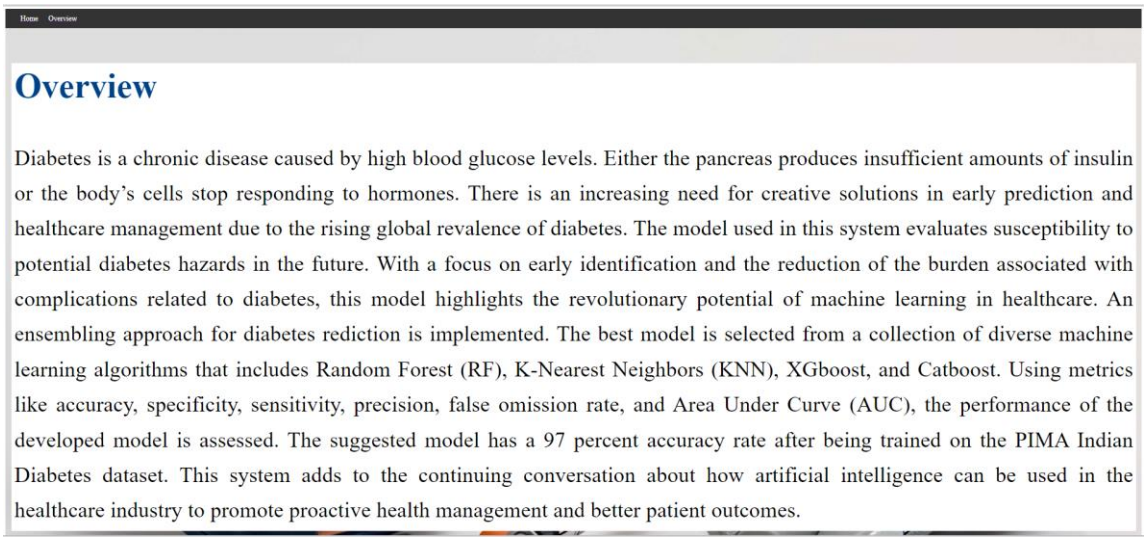

Figure. A.5. Overview page

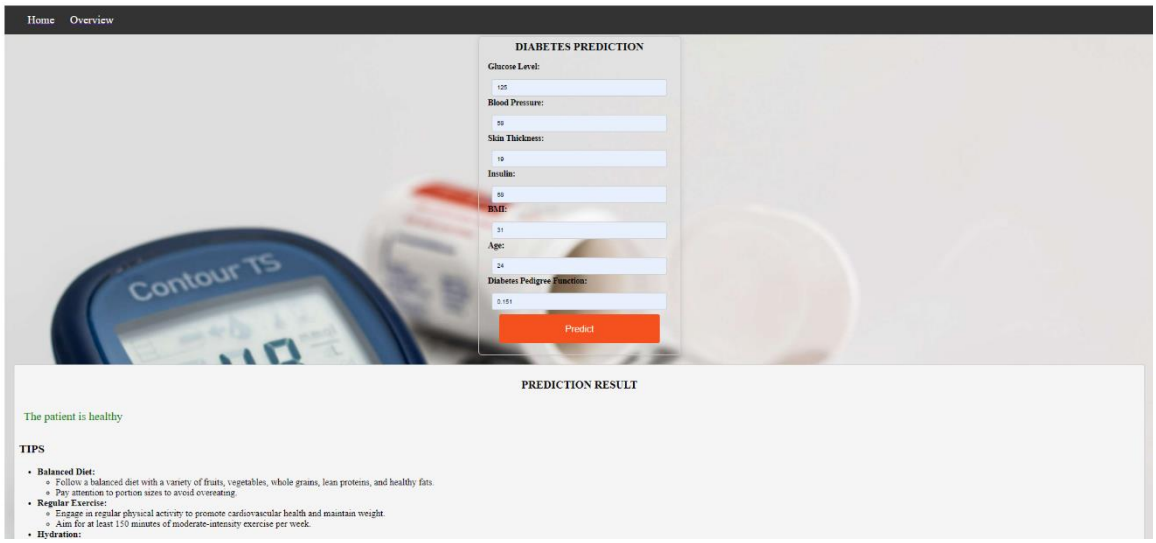

Figure. A.6. Prediction page

APPENDIX B

PUBLISHED PAPER:

Proceedings of the Second International Conference on Automation, Computing and Renewable Systems (ICACRS-2023)
IEEE Xplore Part Number: CFP23CB5-ART; ISBN: 979-8-3503-4023-5

Diabetes Prediction with Ensemble Learning Techniques in Machine Learning

1st S.Phani Praveen
Computer Science Department
P.V.P. Siddhartha Institute of Technology
Vijayawada, India
sppraveen@pvpsiddhartha.ac.in

2nd Vamsi Saripudi
Computer Science Department
P.V.P. Siddhartha Institute of Technology
Vijayawada, India
vamsi7514official@gmail.com

3rd Harshalokh.V
Computer Science Department
P.V.P. Siddhartha Institute of Technology
Vijayawada, India
harshalokhveeramachaneni@gmail.com

4th Sohitha.T
Computer Science Department
P.V.P. Siddhartha Institute of Technology
Vijayawada, India
tadikonda.sohitha3@gmail.com

5th Venkat Sai Karthik.S
Computer Science Department
P.V.P. Siddhartha Institute of Technology
Vijayawada, India
karthiksai131@gmail.com

6th Venkata Pavana Surya Sreekar.T
Computer Science Department
P.V.P. Siddhartha Institute of Technology
Vijayawada, India
tumulurisrikar@gmail.com

2023 2nd International Conference on Automation, Computing and Renewable Systems (ICACRS) | 979-8-3503-4023-5/23/$31.00 ©2023 IEEE | DOI: 10.1109/ICACRS58579.2023.10404311

***Abstract*—Diabetes is a chronic disease caused by high blood glucose levels. Either the pancreas produces insufficient amounts of insulin or the body's cells stop responding to hormones. There is an increasing need for creative solutions in early prediction and healthcare management due to the rising global prevalence of diabetes. The article explores the field of predicting diabetes using machine learning techniques. The model covered in this article evaluates susceptibility to potential diabetes hazards in the future. With a focus on early identification and the reduction of the burden associated with complications related to diabetes, this study highlights the revolutionary potential of machine learning in healthcare. An ensembling approach for diabetes prediction is suggested in this paper. The best model is selected from a collection of diverse machine learning algorithms that includes Random Forest (RF), K-Nearest Neighbors (KNN), XGboost, and Catboost. Using metrics like accuracy, specificity, sensitivity, precision, false omission rate, and Area Under Curve (AUC), the performance of the developed model is assessed. The suggested model has a 96 percent accuracy rate after being trained on the PIMA Indian Diabetes dataset. This article adds to the continuing conversation about how artificial intelligence can be used in the healthcare industry to promote proactive health management and better patient outcomes.**

***Index Terms*—Ensemble, XGBoost, CatBoost, KNN, RF, Machine Learning**

I. Introduction

The latest studies by the World Health Organization (WHO) show an up-and-coming pattern in the number and fatalities of diabetic patients around the world. After analyzing these tendencies, the World Health Organization (WHO) predicts that diabetes will become one of the top 10 fastest growing illnesses worldwide by the year 2030. Diabetes is characterized as an assortment of metabolic sicknesses that outcomes in increased glucose levels in human blood. Diabetes is caused by two primary factors: inadequate secretion of insulin by the human body and impaired sensitivity of body cells to insulin. Insulin, an essential pancreatic enzyme, is responsible for regulating the levels of insulin in the circulation. Diabetes is classified into three distinct types: type-1, type-2, and gestational diabetes. [1], [8]. Type-1 diabetes is caused by immune cells attacking the pancreatic beta cells that produce insulin. Although it is difficult to prevent, it can be treated by administering insulin externally. On the other hand, Type-2 diabetes occurs when the insulin produced by the pancreas is not effectively used. [9]. The significance of this research lies in the fact that diabetes poses a substantial public health challenge, and timely detection is imperative in order to avert consequences. Hence, it is imperative to strengthen public consciousness of the hazards linked to diabetes and improve our capacity to forecast the probability of having diabetes. Diabetes arises from the dysfunction of the pancreas, resulting in an insufficiency of a vital hormone. This illness can lead to serious outcomes, such as the possibility of entering a coma, experiencing kidney and eye failure [13]. Furthermore, diabetes is associated with a range of problems, including cardiovascular dysfunction, cerebral vascular dysfunction, and others. Currently, clinical practice entails collecting the requisite data for diabetes detection through a range of tests. Several healthcare companies are currently incorporating machine learning techniques, such as predictive modeling, into their healthcare procedures. These techniques utilize complex algorithms to detect underlying processes and patterns that may be difficult for humans to perceive [19]. This facilitates researchers in the formulation of novel drugs and treatment strategies. Predictive modeling is based on the principles of data mining, machine learning, and statistics. Its purpose is to discover patterns in data and assess the probability of certain outcomes happening. Subsequently, the utilization of computational insight is tremendously valued for the prediction of diabetes as it helps in accurate predictions. Diabetes prediction could be a vital use of machine learning and data science in healthcare. It involves using several

data sources and predictive modeling approaches. Predictive modeling approaches encompass classification techniques. Utilizing feature selection and engineering strategies improves the performance of the model [22]. Model evaluation employs many metrics such as accuracy, sensitivity, specificity, ROC-AUC, FOR, etc. Diabetes prediction models are useful in several clinical settings, such as identifying the disease at an early stage, tailoring treatment plans to individual patients, efficiently allocating resources, and facilitating research. The challenges encompass issues related to the accuracy of the data and the ability to understand and analyze it. Although faced with these difficulties, the prediction of diabetes is an essential element of preventive healthcare, facilitating timely detection and customized therapies.The goal of the research is to develop a thorough diabetes prediction model that will aid in estimating a person's likelihood of developing the disease. It is essential to mitigate further issues arising from diabetes, as it can give rise to several health complications if disregarded.There are various ML algorithms that are helpful for analyzing and synopsizing the information into important data. Machine learning includes different stages from preprocessing to testing and validation [14]. The PIMA Indian diabetes dataset is picked for this reason. To change over these information into suitable format, preprocessing is required. After the preprocessing is done the data is used to train the machine learning models (models like NN, RF, SVM, NB, AdaBoost, XGBoost, CatBoost). Now these models are tested for accuracy. In ensembling the prediction of each model is considered out of which, the class having maximum voting is chosen as the end prediction.

II. REALTED WORK

A.

In [1], the authors introduced a novel framework called "eDiaPredict," which employs a combination of machine learning techniques to predict the diabetes condition of patients. Initially, the dataset is subject to preprocessing, which includes addressing missing values and feature selection. The recursive feature elimination (RFE) method is employed, ensuring that only the most relevant features are retained for the prediction model. The results indicate that XGBoost stands out, achieving an impressive 92.21 percent accuracy when used in isolation. The paper also introduces an ensemble approach, which involves combining the top-performing models using a majority voting method. The authors provide an in-depth look at the specifics of this ensemble process, highlighting that XGBoost and Random Forest models are the best-performing combination. The evaluation of the eDiaPredict framework incorporates various performance metrics. This ensures that the results are evaluated without any discrepancies. The results demonstrate the effectiveness of the ensemble approach, with XGBoost and Random Forest achieving a remarkable 95 percent accuracy.However, the utilization of methods like Recursive Feature Elimination (RFE) has the risk of overfitting.

B.

In [2], the work centers around the creation of a diabetes prediction model utilizing machine learning techniques, namely logistic regression, to aid in the early identification of the disease. The logistic regression algorithm is used as the main method for constructing the prediction model. Two distinct feature selection techniques are utilized, which involve generating new features based on diagnostic measurements and employing univariate feature selection. The work investigates the utilization of ensemble approaches, that are normally used to amplify the execution of the prediction model. PIMA Indians Diabetes consists of 9 features, and Dataset 2, obtained from Vanderbilt, consists of 16 features, with one target variable indicating the presence of diabetes. Dataset 1 has newly generated features (NF1 through NF5) that are derived from diagnostic measurements pertaining to diabetes risk factors. This method improved the precision of predictions. Ensemble approaches, specifically Max Voting, Majority Voting and etc are evaluated on both datasets. The utilization of Max Voting demonstrated its superior efficiency, resulting in a substantial enhancement in performance. The accuracy of Dataset 1 increases to approximately 78 percent when the ensemble technique of Max Voting is applied. The accuracy of roughly 93 percent achieved by the ensemble techniques Max Voting and Stacking in Dataset 2 showcases the efficacy of ensemble approaches in improving prediction accuracy. The research highlights that the performance of prediction models depends on multiple factors beyond the choice of algorithm, emphasizing the significance of comprehensive data analysis and preprocessing in healthcare applications.

C.

In [3], the researchers presented a robust paradigm for predicting diabetes. The core of this methodology is based on rigorous data preparation. To enhance the dataset's quality, the authors utilized outlier rejection techniques to detect and eliminate data points that exhibited large deviations from the mean. To handle missing values in the dataset, they are imputed with the mean values. This ensures that important data is preserved and not lost in the process. The study investigates various machine learning models for predicting diabetes, such as KNN, DT, AdaBoost, RF, Naive Bayes, and Extreme Boosting. In order to guarantee the dependability of the results, a K cross-validation methodology is utilized. The performance metrics sensitivity, specificity, precision, FOR and DOR are employed to evaluate the models' potential to accurately detect positive and negative cases, as well as the usefulness of the diagnostic test. The authors emphasize the significance of choosing models with minimal correlation in order to attain improved ensemble performance. This study specifically examines the integration of Adaptive Boosting (AB) and Extreme Boosting (XB), demonstrating the significant efficacy of this combined strategy in predicting diabetes. The (AB+XB) model attained a peak efficiency of 95 percent. This work suggests potential future paths, such as creating a user-friendly web application using the trained model and exploring its

979-8-3503-4023-5/23/$31.00 ©2023 IEEE

applicability and adaptability in forecasting various diseases in different medical settings.

D.

In [4], the study aims to evaluate and contrast the effectiveness of two machine learning algorithms, namely KNN and Naïve Bayes, in the study related to diabetes. The data utilized for this research came from the Pima Indians Diabetes Database, which comprises 768 records featuring eight variables and two outcome classifications (0 or 1). The authors utilize K-Fold Cross Validation, which involves dividing the dataset into training and testing data using different proportions ranging from 80 percent to 10 percent. The paper provides a comprehensive analysis of the accuracy, precision, and recall of both systems. The results indicate that in multiple experiments, Naïve Bayes outperforms KNN, achieving an average accuracy of 76.07 percent compared to KNN's 73.33 percent.The study findings indicate that when using the Pima Indians dataset, the Naïve Bayes algorithm is the most advantageous choice for predicting diabetes. To improve the quality of diabetes prediction, authors of this study propose using alternative methods, such as neural networks, and investigating optimization approaches like Particle Swarm Optimization in future studies. This paper conducts an extensive comparative analysis of KNN and Naïve Bayes, with the conclusion that the latter is more suitable for the prediction of diabetes.

E.

In [5], the study centers on the development of a model for assessing the risk of diabetes by utilizing data obtained from community follow-up. The study utilizes an extensive dataset of 252,176 subsequent records of individuals with diabetes, spanning from 2016 to 2023. The main goal is to investigate the correlation between important lifestyle markers derived from community follow-up data and the likelihood of developing diabetes. The authors utilized machine learning techniques, specifically the random forest classifier, to develop a model for assessing the risk of diabetes. Feature selection approaches are employed to find the most essential indications, resulting in a model with a high accuracy rate of 91.24 percent and an AUC of 97 percent. In order to enhance the applicability of the model in clinical settings, the study presents a diabetes risk score card that enables prompt identification by community follow-up doctors and self-assessment by patients, resulting in a precision rate of 95.15 percent. The proposed model exhibits the capacity for extensive risk screening on a wide scale, providing early warnings and enabling individual patient self-assessment. The proposed model and risk score card possess the capacity to support the early identification of diabetes in the patients, thereby tackling the escalating worldwide occurrence of diabetes. A future study could entail augmenting a range of features and incorporating time series analysis of subsequent data to improve the precision of the model and provide better assistance for patients in managing their own health.

F.

In [6], the study introduced a new technique called Average Weighted Objective Distance (AWOD) for diabetes diagnosis. The primary objective was to develop a predictive model that incorporates individual health issues and utilizes parameters that indicate such conditions. The study utilized two datasets to assess the effectiveness of the AWOD method compared to other machine learning approaches such as KNN, Support Vector Machines, Random Forest, and DL. The AWOD technique is grounded in the principle that individuals possess a wide range of health conditions and seeks to prioritize factors based on their influence on health outcomes. The suggested method underwent testing on two datasets: the PIMA dataset and the MD for Diabetes dataset. Each dataset consisted of 392 records. The findings indicated that the AWOD technique had superior performance compared to other machine learning methods, achieving accuracy rates of 93.22 percent. The prediction performance was assessed using various metrics. The study emphasizes the potential of the AWOD technique for accurately predicting type 2 diabetes, even in situations where individual health factors can differ considerably. Nevertheless, certain constraints, such as the intricacy of calculations and the configuration of parameters, could be resolved in future research.

G.

In [7], the primary objective is to employ data mining techniques for the early detection of diabetes. The study utilized a dataset obtained from the National Institute of Diabetes to predict the occurrence of diabetes by examining various diagnostic indicators. Principal component analysis was employed to perform data reduction by extracting important attributes from the dataset. The notable variables comprised glucose, BMI, diastolic blood pressure, and age. The dataset was analyzed using association rule mining to identify patterns and frequent items. The Apriori method was utilized to derive rules, and the investigation revealed robust correlations between diabetes and variables such as blood glucose, blood pressure, age, and BMI. The early prediction of diabetes involved the utilization of three distinct modeling techniques: Artificial Neural Networks (ANN), Random Forest, and K-means clustering. The Random Forest model attained a precision rate of 74.7 percent along with an AUROC value of 0.806. The Artificial Neural Network (ANN) model demonstrated superior performance compared to other approaches, with an accuracy of 75.7 percent. The K-means clustering achieved an accuracy rate of 73.6 percent and yielded an AUROC value of 0.816. The accuracy of this procedure was relatively lower compared to the other two methods used. The study highlighted the efficacy of machine learning and data mining methodologies in predicting diabetes. Further research could entail the inclusion of supplementary factors, such as sedentary lifestyle, familial predisposition to diabetes, and tobacco consumption, and the application of these methodologies to other areas of medicine.

979-8-3503-4023-5/23/$31.00 ©2023 IEEE

III. PRELIMINARIES

Here, we present a concise overview of the many Machine Learning models and techniques used in the suggested framework.

A. K-Nearest Neighbors

K-Nearest Neighbors (K-NN) is a machine learning technique employed for the purposes of classification and regression. Utilizing distance metrics, this algorithm gauges the proximity of data points within the feature space. The K-NN algorithm assigns a class label to a data point through a majority vote among its K nearest neighbors. It uses the average of K data points' closest neighbors to determine the class to which that point belongs.

B. Random Forest

The Random Forest Classifier is a machine learning algorithm renowned for its exceptional accuracy and robustness. It is an approach in machine learning that combines the predictions of numerous decision trees to provide a final prediction. The technique employs bootstrap aggregation and feature randomization to mitigate the problem of overfitting [18]. The approach is beneficial for classification problems that involve a large number of variables and noise data.

C. XGBoost

XGBoost is a higly useful ML approach that is widely recognized for its outstanding performance in complex case studies. The algorithm constructs an ensemble of weak learners in a sequential manner, employing regularization approaches to avoid overfitting. The method employs tree pruning to restrict the depth of individual decision trees and offers efficient cross-validation capabilities. XGBoost is specifically designed to efficiently process tasks in parallel, effectively handle missing values, and include integrated functionality for selecting relevant features [23]. It is well-suited for processing huge datasets and utilizing multicore CPUs, and it has the capability to manage missing data.

D. CatBoost

CatBoost is a machine learning technique developed by Yandex that supports categorical features and uses gradient boosting. It is open-source and can be affected according to requirements without any legal constraints. Its effectiveness in predictive modeling tasks, especially with tabular data, is well established. CatBoost employs gradient boosting, built-in handling of categorical data, regularization methods, and ordered boosting to build trees. It is renowned for its ability to do computations swiftly, handle missing data automatically, and ensure robustness.

E. Stratified K-Fold Cross Validation

Stratified k-fold cross-validation is a machine learning method employed to assess the effectiveness of a model in datasets that are imbalanced or non-uniform. The process involves partitioning the dataset into k folds of equal size, doing model training and testing on each fold, and assessing the performance metrics. The procedure includes dividing the data, doing training and testing, subsequently obtain the comprehensive and mean performance of the model.It is beneficial for models that have an imbalanced class distribution. It helps in making informed judgments on model selection and fine-tuning of hyperparameters.

F. Ensemble Methods

Ensemble methods represent a powerful strategy within machine learning, where they amalgamate the forecasts of multiple models to produce a final prediction that is not only more resilient but also more precise . These techniques are extensively employed to enhance predicted accuracy, enhance model robustness, and minimize overfitting [10]. Ensemble approaches are highly efficient when individual models possess complimentary strengths and weaknesses.

1) Bagging: Bagging is an ensemble methodology in machine learning that entails generating numerous base models by training them on distinct subsets of training data, employing a method known as bootstrapping. The objective of this strategy is to minimize the variance and minimize the problem of overfitting, thereby enhancing the accuracy of predictions and the robustness of the model.

2) Boosting: Boosting is a method in ML that uses ensembling that amalgamate several models outcomes to make a accurate final decision. Boosting algorithms employ weak learners, sequential learning, weighted data, and weighted voting to perform classification or regression tasks.

3) Stacking: Stacking is an practice in machine learning that involves training a large number of models and aggregating their predictions using a meta-model to generate a final prediction. It utilizes a variety of various models to enhance forecast accuracy and the resilience of the model. The meta-model generates predictions by employing k-fold cross-validation.

IV. Proposed Framework

This study involves the application of ensemble learning techniques, specifically KNN, RF, XGBoost, and CatBoost, to predict the risk of an individual developing diabetes using the PIMA dataset. The procedure begins with data preprocessing, involving the identification and imputation of missing values as well as the elimination of outliers by using the interquartile range. The models are trained and ensembled based on voting classifier and by averaging predicted probabilities. The model's robustness is assessed using thirty-fold cross-validation. Fig. 1 provides an elaborate elucidation of the procedure.

A. Data Preprocessing

The PIMA Indian Diabetes dataset has 768 records, with 268 individuals diagnosed with diabetes and 500 individuals without diabetes [24]. The information is openly accessible and seeks to offer a thorough comprehension of diabetes. The dataset has eight attributes:

979-8-3503-4023-5/23/$31.00 ©2023 IEEE

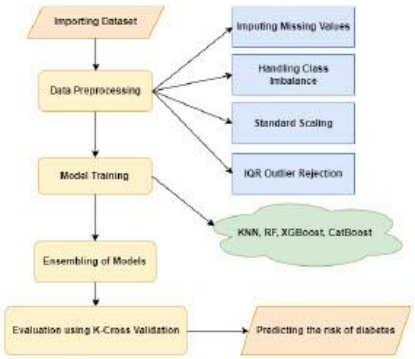

Fig. 1. Proposed System.

1) Pregnancies: Indicates the number of pregnancies experienced.

2) Glucose: Provides information on the concentration of glucose in the plasma after a two-hour oral glucose tolerance test.

3) BloodPressure: It is quantified using the unit millimeters of mercury (mm Hg).

4) SkinThickness: It refers to the measurement of the thickness of the skin fold on the triceps muscle, expressed in millimeters.

5) Insulin: Provides the measurement of 2-hour serum insulin in milli units per milliliter (mU/ml).

6) BMI: The calculation involves obtaining the individual's weight and height.

7) Age: It refers to the number of years that a person has lived.

8) DiabetesPedigreeFunction: The Diabetes Pedigree Function assesses the probability of developing diabetes based on the individual's age and their family history of diabetes.

	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
5	5	116	74	0	0	25.6	0.201	30	0
6	3	78	50	32	88	31.0	0.248	26	1
7	10	115	0	0	0	35.3	0.134	29	0
8	2	197	70	45	543	30.5	0.158	53	1
9	8	125	96	0	0	0.0	0.232	54	1

Fig. 2. Sample records in the dataset.

Addressing missing data is an essential stage in the data preprocessing process for machine learning tasks. The presence of missing data can greatly influence the effectiveness of machine learning models, making it crucial to comprehend the methods for handling them [11]. Any feature, excluding from pregnancy, that has a value of zero should be regarded as a missing value. It has been noted that all features, except for age and pedigree function, have zero values.The box plot in Fig. 3 gives a visual summary of data distribution in the dataset.

It is observed that there is a class imbalance in the given dataset. The number of instances with label 0 is twice as large as that of 1. So performing mean imputation may induce a bias that is in favor of the majority class. To prevent this, the minority class instances are imputed with only minority class data, and the majority class instances are imputed with majority class data, which is commonly referred to as class-aware imputation.Standard Scaling is a higly employed data preprocessing method applied to datasets. This procedure entails converting the characteristics of a dataset in such a way that they possess a mean (average) with a value 0 and a standard deviation value of 1. This process makes it sure that all the features in the collection of data are on a congruous scale [20]. Normalizing the input features is crucial, especially when dealing with machine learning algorithms that are highly influenced by the magnitude of the features [15]. Standard scaling eliminates biases and offsets in the data, enhances the convergence of optimization techniques, and ensures an equal contribution of all features to model predictions.

$$z = (x - \mu)/\sigma \tag{1}$$

z is the standardized value.
x is the original data point.
μ is the mean of the feature.
σ is the standard deviation of the feature.

The IQR outlier rejection approach is employed to identify and address outliers within a dataset. This strategy is highly effective in identifying and potentially removing outliers. It is computed by subtracting the value of the first quartile of the feature from the of its third quartile in the dataset, and is performed for all the features.

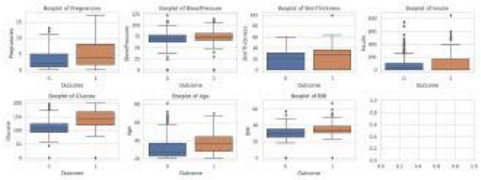

Fig. 3. Box plot of features.

The data correlation plot in Fig. 4 is used to understand the association between the features in the dataset. It helps understand how changes in one variable are related to changes in another.

979-8-3503-4023-5/23/$31.00 ©2023 IEEE

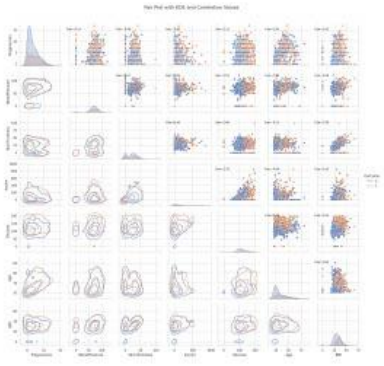

Fig. 4. Correlation plot of the dataset features.

B. Training of Models

The process of training and testing models is essential to the creation of the ML models. These procedures involve the division of data into training and test sets. The process of model training involves the careful selection of a suitable machine learning algorithm or model for the given job, followed by training the model and tuning hyperparameters to enhance performance [21]. The model's performance is evaluated using performance metrics that were mentioned in the previous sections of the study. After the model has undergone training, testing, and evaluation, it can be implemented in practical scenarios to generate predictions based on fresh data. Efficient model training and testing necessitate a careful combination of data preparation, model selection, hyperparameter tuning, and performance evaluation. This process may involve multiple iterations to refine the model and achieve optimal results.
The KNN model is trained with the parameter 'n_neighbors' set to 5. It utilizes the majority class among its five closest neighbors to make predictions. The RF model is prepard by setting the 'n_estimators' parameter to a value of hundred. Voting is done on the results of the 100 decision trees, and the class with highest number of votes is chosen. XGBoost is trained by setting the 'max_depth' parameter to 3 and 'n_estimators' to 15000. In CatBoost, the model is trained by setting the 'iterations' parameter to 900 with a 'depth' of 5. Among the models, CatBoost and XGBoost are better when contrasted with the other models that were used.

C. Model Ensembling

In this stage, the previously trained models are ensembled to get improved results. A soft voting classifier is employed as it combines the results of the selected models by calculating the mean of their occurence. For a classification task, each base model generates a probability distribution for the possible classes. The final prediction is determined by averaging these probabilities [12]. This strategy is generally more robust and frequently results in enhanced precision compared to hard voting, where each model's vote is considered a distinct class. KNN and RF are combined into an ensemble. By combining them, you are utilizing their complementary traits. The KNN algorithm utilizes the proximity of data points to produce predictions, enabling it to capture local patterns [17]. On the other hand, the Random Forest (RF) algorithm, which consists of a collection of decision trees, has the ability to capture intricate global patterns. Considering the strong individual performance of XGBoost and CatBoost, they are ensembled with other models to combine the predictions in order to attain the final results.

V. Results and Analysis

This section presents a detailed analysis of the outcomes obtained from the implemented framework. The aspects encompassed in this are the criteria for evaluation, the presentation of study results, the visual representation of data, and the discussion of performance.

A. Criteria for Evaluation

The models' performance is evaluated based on the following criteria:

1) Accuracy: Accuracy is defined as the proportion of correctly classified instances out of the total number of instances.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{2}$$

2) Precision: It refers to the proportion of accurately predicted positive events to the total number of events that were identified as true.

$$Precision = \frac{TP}{TP + FP} \tag{3}$$

3) Sensitivity: The measure quantifies a model's capacity to accurately identify positive occurrences from the entire set of actual positive instances.

$$Sensitivity = \frac{TP}{TP + FN} \tag{4}$$

4) Specificity: This statistic measures the model's capacity to accurately detect true negative situations out of all the real negative instances.

$$Specificity = \frac{TN}{TN + FP} \tag{5}$$

5) False Omission Rate (FOR): The False Omission Rate (FOR) quantifies the tendency of a model to incorrectly categorize negative cases as positive.

$$FOR = 1 - Specificity \tag{6}$$

6) AUC-ROC (Receiver Operating Characteristic): The AUC-ROC score evaluates a model's capacity to distinguish between the correctly classified postive classes to wrongly classified positive classes, where a score above 0.5 signifies remarkable performance.

979-8-3503-4023-5/23/$31.00 ©2023 IEEE

B. Study Results

Initially, models are trained separately and evaluated using metrics such as accuracy, precision, sensitivity, specificity and FOR. The models underwent 30-fold operations of cross-validations. Here the dataset is partitioned into 30 subsets and the training and testing process is repeated 30 times. This approach guarantees that the performance of a model is sufficiently trained and assessed, thereby detecting possible problems like overfitting or underfitting. The procedure involves data preprocessing, partitioning into 30 subsets, training and testing, collecting performance metrics, evaluating the model, fine-tuning parameters, and ultimately training the final model.

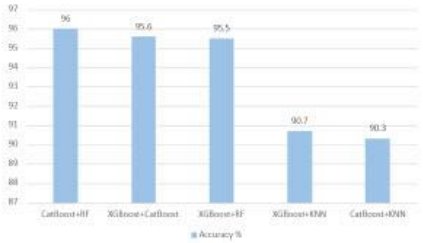

Fig. 5. A comparision of ensembled models performances.

This approach yielded a more robust assessment of a model's performance. CatBoost has demonstrated superior performance compared to other models, with an accuracy rate of 95%. The method of data preparation and the optimization of parameters had a significant impact on this outcome which were discussed in section IV-B. Given that CatBoost and XGBoost produced better results independently, they were ensembled with the other models in order to determine the model that exhibited the highest level of efficiency.

The Fig. 7 gives an illustration of peformance metris of ensembled models.The evaluation of several ensemble models revealed a spectrum of accuracies, providing insights into the prediction capacities of various combinations of machine learning algorithms [25]. The combination of CatBoost and RF yielded the most optimal results, with an accuracy rate of 96%. The XGBoost and CatBoost combination demonstrated a strong accuracy of 95.6%. The ensemble model consisting of CatBoost and K-Nearest Neighbors (KNN) achieved an impressive accuracy rate of 90.7%. The XGBoost and Random Forest (RF) ensemble obtained an accuracy of 95.5%, just behind the leading models. Meanwhile, the XGBoost and K-Nearest Neighbors (KNN) combination displayed a consistently accurate performance of 90.3%.

These results clarify the different degrees of performance exhibited by ensemble models and provide significant insights for making informed decisions about the most appropriate technique for predicting diabetes in patients .

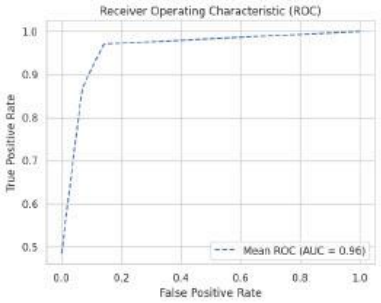

Fig. 6. ROC Curve for CatBoost and RF Ensemble.

The suggested framework demonstrates the appropriateness of the models based according to the performance metrics that were chosen for this research. The CatBoost algorithm, along with the random decision forests model, achieves the highest performance in properly predicting the risk of diabetes in patients with a 96% correctness. The Fig. 6 depicts the receiver operating characteristics of the CatBoost+RF model, which attained the greatest accuracy among all the ensembled models.

Additionally, these models have greater values for several performance measures. CatBoost utilizes gradient boosting, integrated handling of categorical data, regularization techniques, and ordered boosting to construct trees. The integration of models offers significant advantages by consolidating the capabilities of many selected models and directing them towards a specific job required to accomplish the ultimate objective of the work [16].Previous studies have demonstrated that achieving 100% effectiveness is unattainable for any model. However, doing so can undoubtedly be advantageous for individuals based on their specific needs. The RF model operates by randomly generating several decision trees and making predictions based on the collective voting of these trees.

Model	*Accuracy*	*Precision*	*Sensitivity*	*Specificity*	*Mean AUC*	*FOR*
CatBoost + RF	96	95.5	96.7	95.1	0.96	0.031
XGBoost + CatBoost	95.6	94.9	96.8	94.4	0.98	0.020
XGBoost + RF	95.5	95.4	95.9	95.1	0.95	0.040
CatBoost + KNN	90.7	93.9	87.2	94.2	0.91	0.115
XGBoost + KNN	90.3	93.6	86.6	93.9	0.90	0.121

Fig. 7. Performance metrics of ensembled models.

Ultimately, the final tree is depicted according to the voting outcomes. These ensemble models can be efficiently implemented in both distributed and memory-constrained situations.The findings of this study demonstrate that the suggested model surpasses current diabetes prediction models in the

979-8-3503-4023-5/23/$31.00 ©2023 IEEE

perspective of accuracy and precision.Previous research have employed recursive feature removal, a technique that has the potential to result in overfitting. On the other hand, our method circumvents this problem by taking into account all features. In addition, a vital component of our study entails the utilization of "class-aware imputation," as elaborated upon in section IV-A, to safeguard against the introduction of bias into the data. In addition, the integration of very sophisticated machine learning algorithms has greatly improved the accuracy of our study. . The outcomes of our work indicate, suggested model holds great potential in enhancing the efficiency of diabetes prediction systems. Additionally, this model can be utilized with different datasets, hence improving the quality of diabetes diagnosis.

VI. Conclusion and future work

This study presents a predictive framework that aims to accurately anticipate the risk of diabetes by conducting a thorough review of important health parameters. Significantly, we found that our methodology attained exceptional outcomes, as the CatBoost algorithm combined with Random Forest (RF) showcased an exceptional individual accuracy rate of 96%. We anticipate that our framework will be further validated in the future using real-world clinical data to ensure its robustness in a variety of healthcare contexts. Enhancements can be made to the current framework by creating a user-friendly interface, thereby improving its usability.

References

[1] Ashmia Singh, Arwinder Dhillon, and Neeraj Kumar,M. Shamim Hossain, Ghulam Muhammad, Manoj Kumar, "eDiaPredict: An Ensemble-based Framework for Diabetes Prediction", doi: 10.1145/3415155, June 2021.
[2] Priyanka Rajendra, Shahram Latif, "Prediction of diabetes using logistic regression and ensemble techniques" doi: 10.1016/j.cmpbup.2021.100032 , 25 October 2021.
[3] MD.Kamrul Hasan, MD. Ashraful Alam, Dola Das, Eklas Hossain, Mahmudul Hasan, "Diabetes Prediction Using Ensembling of Different Machine Learning Classifiers" , doi: 10.1109/ACCESS.2020.2989857, April 23, 2020.
[4] Muhammad Exell Febriana, Fransiskus Xaverius Ferdinana, "Diabetes prediction using supervised machine learning", doi: 10.1016/j.procs.2022.12.107, 2022.
[5] Liangjun Jiang, Zhenhua Xia .., "Diabetes risk prediction model based on community follow-up data using machine learning ", doi: 10.1016/j.pmedr.2023.102358 , 20 August 2023.
[6] Pratya Nuankaew, Supansa Chaising, Punnarumol Temdee, "Average Weighted Objective Distance-Based Method for Type 2 Diabetes Prediction", doi: 10.1109/ACCESS.2021.3117269, October, 2021.
[7] Talha Mahboob Alama, Muhammad Atif Iqbala, Yasir Alia, "A model for early prediction of diabetes", doi: 10.1016/j.imu.2019.100204, July 2019.
[8] Roosa Peramaki, Mika Gisslerb, "The risk of developing type 2 diabetes after gestational diabetes: A registry study from Finland", doi: 10.1016/j.deman.2022.100124, 2022.
[9] Srinivasu, P. N., Shafi, J., Krishna, T. B., Sujatha, C. N., Praveen, S. P., & Ijaz, M. F. (2022). Using recurrent neural networks for predicting type-2 diabetes from genomic and tabular data. Diagnostics, 12(12), 3067.
[10] Joy Dhar, Nigus Asres Ayele, "Multi-Tier Ensemble Learning Model With Neighborhood Component Analysis to Predict Health Diseases", doi: 10.1109/ACCESS.2021.3117963, October,2021.
[11] Martin Dodek, Eva Miklovicova, Marian Tarnik, "Correlation Method for Identification of a Nonparametric Model of Type 1 Diabetes", doi: 10.1109/ACCESS.2022.3212435, October 2022.
[12] Norma Latif Fitriyani, Muhammad Syafrudin, Ganjar Alfain , Jongtae Rhee, "Development of Disease Prediction Model Based on Ensemble Learning Approach for Diabetes and Hypertension" , doi: 10.1109/ACCESS.2019.2945129, October, 2019.
[13] American Dibetes Association, "Diagnosis and classification of Diabetes Mellitus", doi: 10.2337/dc11-S062, January, 2011.
[14] Phani Praveen, S., Hasan Ali, M., Musa Jaber, M., Buddhi, D., Prakash, C., Rani, D. R., & Thirugnanam, T. (2023). IoT-Enabled Healthcare Data Analysis in Virtual Hospital Systems Using Industry 4.0 Smart Manufacturing. International Journal of Pattern Recognition and Artificial Intelligence, 37(02), 2356002.
[15] Virginie Felizardo, Diogo Machado, Nuno M. Garcia, "Hypoglycaemia Prediction Models With Auto Explanation", doi: 10.1109/ACCESS.2021.3117340, October, 2021.
[16] S. P. Praveen, S. Sindhura, A. Madhuri and D. A. Karras, "A Novel Effective Framework for Medical Images Secure Storage Using Advanced Cipher Text Algorithm in Cloud Computing," 2021 IEEE International Conference on Imaging Systems and Techniques (IST), Kaohsiung, Taiwan, 2021, pp. 1-4, doi: 10.1109/IST50367.2021.9651475.
[17] Jie Zhang, Fang Wang, "Prediction of Gestational Diabetes Mellitus under Cascade and Ensemble Learning Algorithm", doi: 10.1155/2022/3212738, 2022.
[18] Tadao Ooka , Hisashi Johno, Kazunori Nakamoto, "Random forest approach for determining risk prediction and predictive factors of type 2 diabetes: large-scale health check-up data in Japan", doi: 10.1136/bmjnph-2020-000200, 2021.
[19] Swamy, S. R., Praveen, S. P., Ahmed, S., Srinivasu, P. N., & Alhumam, A. (2023). Multi-features disease analysis based smart diagnosis for covid-19. Computer Systems Science and Engineering, 45(1), 869-886.
[20] Ruihu Wang, "AdaBoost for Feature Selection, Classification and Its Relation with SVM, A Review ", doi: 10.1016/j.phpro.2012.03.160 , 2012.
[21] Ziyue Yu, Wuman Luo, Rita Tse, Giovanni Pau, "DMNet: A Personalized Risk Assessment Framework for Elderly People With Type 2 Diabetes", doi: 10.1109/JBHI.2022.3233622, 2023.
[22] Praveen, S. P., Jyothi, V. E., Anuradha, C., VenuGopal, K., Shariff, V., & Sindhura, S. (2022). Chronic Kidney Disease Prediction Using ML-Based Neuro-Fuzzy Model. International Journal of Image and Graphics, 2340013.
[23] Tianqi Chen, Carlos Guestrin, "XGBoost: A Scalable Tree Boosting System", doi: 10.1145/2939672.2939785, 2016.
[24] Dataset Retrieved from : https://data.world/data-society/pima-indians-diabetes-database
[25] Ensemble Methods retrieved from : https://www.toptal.com/machine-learning/ensemble-methods-machine-learning

979-8-3503-4023-5/23/$31.00 ©2023 IEEE

PUBLICATION LINK : https://ieeexplore.ieee.org/document/10404311