People's Democratic Republic of Algeria
Ministry of Higher Education and Scientific Research

University of 8 May 1945-Guelma-
Faculty of Mathematics, Computer Science and Science of Matter
Department of Computer Science

Master Thesis

Specialty: Computer Science

Option:

Science and Technology of Information and Communication

Theme

Hybrid System for Diabetes Prediction

Presented by:
Saïd Bouteldja

Supervised by:
Dr Djalila Boughareb

Jury Members:
Dr Hassina Bourssace
Dr Nadia Guerroui

June 2023

Acknowledgments

First and foremost, I would like to express my deepest gratitude to **"Allah"** for granting me strength, guidance, and perseverance throughout my academic journey.

I am incredibly grateful to my supervisor, Dr. Djalila Boughareb, for agreeing to be my supervisor and for her invaluable supervision and guidance throughout this research.

I would like to express my sincere appreciation. to the members of the jury for their time, expertise, and constructive feedback.

To my beloved parents, whose unwavering love, sacrifices, and constant support have been the foundation of my success, I am eternally grateful. Your belief in me and your encouragement have been my driving force, and I owe all my achievements to you. Thank you for always being there for me.

I wish to extend my heartfelt gratitude to my sister, Dr. Meriem, and her husband Aymen, as well as my other family members, Hanane, Mehdi, Fouad, and Mouhammed, for their continuous guidance, support, and encouragement.

I would like to acknowledge and thank all my friends who have accompanied me on this academic journey. In particular, I want to express my appreciation to Heythem, Oussama and Sayed for their camaraderie and support. Your friendship has made this journey enjoyable and memorable.

Lastly, I would like to extend my gratitude to all the individuals who have supported and contributed to my growth as a computer science student, both academically and personally. Your presence, discussions, and collaboration have played a crucial role in shaping my knowledge and understanding.

Dedications

"To my parents, who never understood what computer science was all about, but still supported me all the way. Thank you for nodding and smiling whenever I tried to explain programming to you."

"To Stack Overflow, for providing me with all the answers to my coding problems, and for making me feel like a genius whenever I found a solution."

"To my computer, for never crashing during the final weeks of my thesis. I owe you a lifetime of virus scans and defrags."

Abstarct

Diabetes is a chronic condition that can be caused by the body's inability to produce or use insulin effectively. Over time, this can result in damage to various organs, including the heart, blood vessels, eyes, kidneys, and nerves. The timely detection of diabetes is essential for its prompt treatment, as it can halt the progression of the disease. In this study, we propose an hybrid machine learning approach to predict diabetes using a combination of two powerful algorithms. We used an ensemble learning based on Deep Neural Network and Random Forest classifier, and Support Vector Machine as a meta classifier (SVC). We trained and tested our model on the Pima Indian diabetes dataset , which contains 77568 instances and 8 features, using 5-fold cross validation. Our experimental results show that our proposed approach achieved an accuracy of 95% , outperforming other state of the art machine learning techniques. We propose also an alternative approach that combines random forest and XGBoost by a voting technique getting 92.7% of accuracy. Our findings suggest that the hybrid machine learning approachs we proposed can be used as a reliable tool for early diabetes detection, enabling more timely and effective interventions to improve patient outcomes.

Keywords: Diabetes Prediction, DNN, SVM, XGBoost, Random Forest, Stacking, Machine Learning

Résumé

Le diabète est une maladie chronique qui peut être causée par l'incapacité du corps à produire ou à utiliser efficacement l'insuline. Avec le temps, cela peut entraîner des dommages à divers organes, y compris le cœur, les vaisseaux sanguins, les yeux, les reins et les nerfs. La détection précoce du diabète est essentielle pour un traitement rapide, car elle peut arrêter la progression de la maladie. Dans cette étude, nous proposons une approche hybride d'apprentissage automatique pour prédire le diabète en utilisant une combinaison de deux puissants algorithmes. Nous avons utilisé un apprentissage en ensemble basé sur un réseau de neurones profonds (DNN) et un classificateur de forêt aléatoire (Random Forest), ainsi qu'une machine à vecteurs de support (SVC) en tant que méta-classifieur. Nous avons entraîné et testé notre modèle sur l'ensemble de données du diabète des Indiens Pima, qui contient 77568 exemples et 8 caractéristiques, en utilisant une validation croisée à 5 plis. Nos résultats expérimentaux montrent que notre approche proposée a atteint une précision de 95%, surpassant les autres techniques d'apprentissage automatique de pointe. Nous proposons également une approche alternative qui combine forêt aléatoire et XGBoost par une technique de vote obtenant 92,7% de précision. Nos résultats suggèrent que les approches hybrides d'apprentissage automatique que nous avons proposée peuvent être utilisées comme un outil fiable pour la détection précoce du diabète, permettant des interventions plus rapides et plus efficaces pour améliorer les résultats des patients.

Mots clés: Prédiction du diabète, DNN, SVM, XGBoost , Random Forest, Stacking, Apprentissage automatique.

Contents

List of Tables

List of Figures

General introduction

Diabetes, a chronic metabolic disorder characterized by high blood sugar levels, has emerged as a significant public health concern worldwide. According to the International Diabetes Federation (IDF), an estimated 463 million adults were living with diabetes in 2019, and this number is projected to rise to 700 million by 2045. Diabetes not only poses substantial health risks to individuals but also places a considerable burden on healthcare systems and economies.

The timely and accurate diagnosis of diabetes is crucial for effective management and prevention of complications. Traditional diagnostic methods rely on clinical assessments and laboratory tests, which can be time-consuming, expensive, and dependent on the availability of healthcare professionals. To address these challenges, researchers and practitioners have turned to machine learning techniques for diabetes prediction.

Machine learning, a branch of artificial intelligence, offers the potential to improve diabetes prediction by analyzing large volumes of data and identifying patterns that may not be apparent to human experts.

The problematic of this research lies in the need for more accurate prediction models for diabetes. Although traditional diagnostic methods and clinical expertise play a crucial role in diagnosis, the integration of machine learning algorithms can augment the accuracy and efficiency of the process. By harnessing the power of ensemble learning, which combines multiple models to make collective predictions, we aim to overcome the shortcomings of individual models and create a more robust and reliable diabetes prediction system.

Our objective is to develop a diabetes prediction model using ensemble learning techniques that outperforms the existing state-of-the-art in terms of accuracy, sensitivity, and specificity. This will facilitate the early identification of individuals at risk of developing diabetes, enabling timely interventions and preventive measures. Secondly, we aim to explore and evaluate different ensemble learning algorithms, including **stacking approach** based on Deep Neural Network and Random Forest classifier and Support Vector Machine as a meta classifier (SVC) and **voting approach** that combines random forest and XGBoost. This, in order to determine the most effective approach for diabetes prediction, providing insights into the strengths and weaknesses of each algorithm in this specific context.

The core of this thesis consists of three main chapters. Chapter 1 provides an overview of diabetes mellitus, including its its definition, types and symptoms, prevalence, complications, diagnosis, prevention and how to manage it.
Chapter 2 delves into the foundations of machine learning, specifically focusing on supervised learning algorithms and their applications in healthcare, including diabetes prediction. We review related works in the field, examining the strengths of existing approaches.
Chapter 3 outlines the methodology and implementation details of our proposed ensemble learning-based diabetes prediction system. We describe the dataset used for training and evaluation, discuss the preprocessing steps employed, and present the ensemble learning algorithms utilized. Furthermore, we evaluate and compare the performance of these algorithms, providing insights into their predictive capabilities and their potential for clinical adoption. .

Chapter 1

Diabetes Mellitus

1.1 Introduction

Diabetes mellitus is a significant and chronic condition characterized by inadequate insulin production by the pancreas or the body's inability to effectively utilize the insulin it produces. This metabolic disorder is recognized as a major public health concern and is among the four priority noncommunicable diseases (NCDs) targeted for action by global leaders. Over the past few decades, there has been a consistent rise in the number of diabetes cases and its prevalence, highlighting the urgent need for effective management and prevention strategies [6].

The purpose of this chapter is to provide a brief overview of diabetes. This overview will serve as a foundation for the subsequent chapters of this thesis, which will focus on the application of machine learning techniques for diabetes mellitus prediction. By the end of this chapter, you will have a good understanding of diabetes mellitus and its impact on public health, as well as the importance of early detection and prevention of the condition.

1.2 Diabetes Types

1.2.1 Type 1 Diabetes

Type 1 diabetes, formerly referred to as insulin dependent, is marked by inadequate insulin production within the body. Individuals diagnosed with type 1 diabetes need to administer insulin on a daily basis in order to control their blood glucose levels. Without access to insulin, their survival is not possible. The root cause of type 1 diabetes remains unknown, and currently, there are no preventive measures available. Symptoms of this condition include frequent urination, persistent hunger, changes in vision, fatigue, weight loss and increased thirst [6].

1.2.2 Type 2 Diabetes

Type 2 diabetes, previously known as non-insulin-dependent or adult-onset diabetes, occurs due to the body's inefficient utilization of insulin. It is the most common form of diabetes worldwide. Although symptoms of type 2 diabetes may resemble those of type 1 diabetes, they are typically less pronounced or may not be present at all. As a result, the condition can remain undiagnosed for several years, leading to the development of complications. While type 2 diabetes was once predominantly observed in adults, it has now started to affect children as well [6].
Figure 1.1 explains the difference between type 1 and type 2 diabetes.

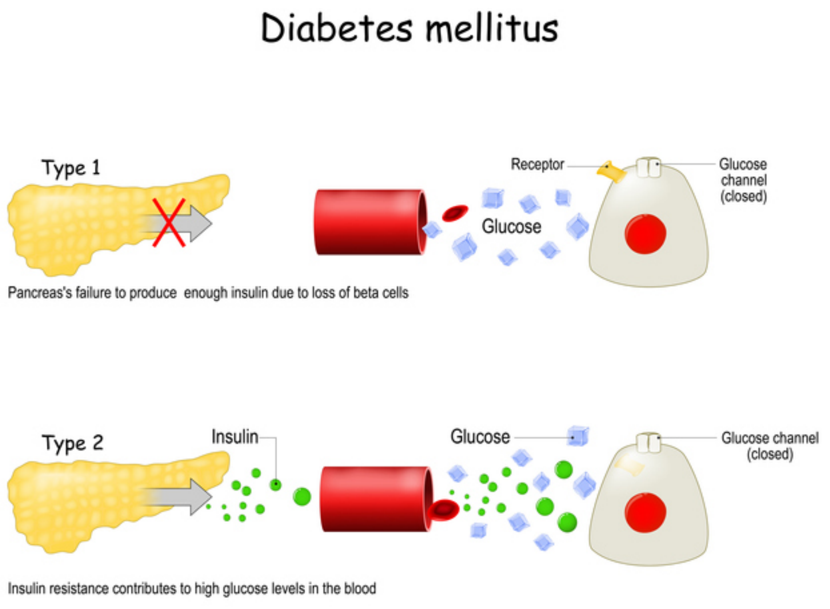

Diabetes mellitus
Type 1
Receptor
Glucose channel (closed)
Glucose
Pancreas's failure to produce enough insulin due to loss of beta cells
Type 2
Insulin
Glucose
Glucose channel (closed)
Insulin resistance contributes to high glucose levels in the blood

Figure 1.1: Diabetes type 1 vs 2 [W1].

1.2.3 Gestational Diabetes

Gestational diabetes is a temporary condition that develops during pregnancy and poses a long-term risk of type 2 diabetes [7]. It is characterized by elevated blood glucose levels that are above normal but below the diagnostic threshold for diabetes [8]. Women diagnosed with gestational diabetes are at an elevated risk of experiencing certain complications during pregnancy and delivery, as well as their infants. The diagnosis of gestational diabetes is typically made through prenatal screening rather than relying on reported symptoms [6].

1.2.4 Other specific types of diabetes

This category includes a variety of rare forms of diabetes that result from specific genetic syndromes, pancreatic diseases, or drug-induced causes.

1.3 Global Burden

In 2014, an estimated 422 million adults globally were living with diabetes, compared to 108 million in 1980. The prevalence of diabetes at a global level, after adjusting for age, has nearly doubled since 1980, escalating from 4.7% to 8.5% among the adult population. This increase can be attributed to a rise in associated risk factors, particularly overweight or obesity. Notably, the prevalence of diabetes has been rising at a faster rate in low- and middle-income countries compared to high-income countries over the past decade. Diabetes accounted for 1.5 million deaths in 2012, while higher-than-optimal blood glucose levels led to an additional 2.2 million deaths, increasing the risk of cardiovascular and other diseases. Alarmingly, 43% of these 3.7 million deaths occurred before the age of 70. Low and middle-income countries bear a higher burden, with a larger percentage of deaths attributable to high blood glucose or diabetes occurring before the age of 70 when compared to high-income countries. It is important to note that global estimates for diabetes prevalence are not distinguished between type 1 diabetes and type 2 diabetes. The majority of individuals affected by diabetes have type 2 diabetes, which was traditionally observed predominantly in adults but is now increasingly diagnosed in children as well [6].

1.4 Complications

All forms of diabetes can result in complications affecting multiple body systems and raising the overall risk of premature mortality. Potential complications include heart attacks, strokes, kidney failure, leg amputations, vision loss, and nerve damage. During pregnancy, inadequate control of diabetes increases the likelihood of fetal death and other associated complications. [6].

1.5 Diagnosis

Currently, there are four recommended diagnostic tests for diabetes mellitus. These include the measurement of fasting plasma glucose, the 2-hour post-load plasma glucose after a 75 g oral glucose tolerance test (OGTT), the HbA1c level, and random blood glucose in the presence of signs and symptoms of diabetes. Diagnosis of diabetes is determined based on specific threshold values for each test. Individuals with fasting plasma glucose levels of 7.0 mmol/L (126 mg/dl), 2-hour post-load plasma glucose levels 11.1 mmol/L (200 mg/dl), HbA1c levels 6.5% (48 mmol/mol), or random blood glucose levels 11.1 mmol/L (200 mg/dl) in the presence of signs and symptoms are considered to have diabetes. In cases where elevated values are detected in asymptomatic individuals, it is recommended to repeat the testing, preferably with the same test, as soon as feasible on a subsequent day to confirm the diagnosis [9] [10] [11].

1.6 Preventing Diabetes

The prevention of type 1 diabetes remains challenging with current knowledge, as there are no known effective approaches for its prevention. However, for type 2 diabetes and the complications associated with all types of diabetes, effective preventive strategies exist. These approaches encompass broad policies and practices implemented at both population and specific setting levels, such as schools, homes, and workplaces, to promote overall good health for everyone, regardless of their diabetes status. These measures include regular exercise, healthy eating habits, smoking avoidance, and the management of blood pressure and lipids.

Adopting a life-course perspective is crucial in preventing type 2 diabetes and many other health conditions. During early life, when eating habits, physical activity patterns, and long-term energy balance regulation are established, there is a critical window for interventions aimed at reducing the risk of obesity and type 2 diabetes later in life. It is important to recognize that achieving this goal requires a comprehensive approach, as no single policy or intervention can ensure success. A whole-of-government and whole-of-society approach is necessary, involving systematic consideration of the health impact of policies in various sectors such as trade, agriculture, finance, transport, education, and urban planning. This recognizes that health outcomes are influenced by policies implemented in these and other areas, and emphasizes the need for collaboration and coordination across sectors to enhance health and prevent diabetes effectively [6].

1.7 Managing Diabetes

Early diagnosis is crucial for promoting positive health outcomes in individuals living with diabetes. Prolonged periods of undiagnosed and untreated diabetes often lead to worsened health conditions. Thus, it is imperative to ensure easy access to essential diagnostic tools like blood glucose testing in primary health-care settings. Furthermore, a well-established system for referral and back-referral is necessary to facilitate periodic specialist assessment or treatment for complications that may arise. Following a diabetes diagnosis, implementing a series of cost-effective interventions can significantly improve outcomes, irrespective of the diabetes type. These interventions encompass blood glucose control achieved through a combination of diet, physical activity, and, when necessary, medication. Controlling blood pressure and lipids is also vital to reducing the risk of cardiovascular complications and other associated health issues. Additionally, regular screening for diabetic retinopathy, nephropathy, and foot problems enables early detection and timely intervention. To enhance diabetes management, the utilization of standards and protocols is recommended. These frameworks contribute to streamlining care processes, ensuring consistency in treatment approaches, and improving the overall quality of care for individuals with diabetes [6].

1.8 Predictive Medicine

A field of medicine known as predictive medicine seeks to locate people who are at high risk of contracting a disease, allowing for early diagnosis and treatment options. To find indicators of a person's propensity for a disease in the future, either single or, more frequently, several studies are utilized.
Predictive medicine is still under development and requires ongoing efforts to improve algorithms and data analysis methods. However, it has the potential to provide more personalized and effective healthcare, by helping doctors make more informed decisions and enabling patients to benefit from more targeted and tailored treatment based on their specific needs.

What is prediction:

Prediction is the act of forecasting or estimating a future event, outcome or result based on past data, information or patterns. It involves using statistical or machine learning models to analyze data and make educated guesses about what will happen in the future. Predictive modeling is often used in fields such as finance, economics, weather forecasting, and healthcare to help decision-makers plan for and manage future risks and opportunities. The accuracy of predictions can vary depending on the quality and quantity of available data, the complexity of the problem, and the sophistication of the prediction model used [12].

In our case, **DIABETES PREDICTION** refers to using machine learning models and other statistical techniques to estimate the risk of developing diabetes in an individual based on their clinical and genetic data. These models analyze large amounts of data from electronic health records, genetic information, and other sources to identify individuals at high risk of developing diabetes mellitus.

1.9 Conclusion

Diabetes mellitus is a chronic disease that has become a major public health concern worldwide. In this chapter, we provided a general introduction to diabetes mellitus, including its definition, types and symptoms, prevalence, complications, diagnosis, pre-

vention and how to manage it.
While current prevention strategies have shown some success in reducing the risk of diabetes, the global burden of diabetes continues to rise, highlighting the need for new and innovative approaches.

In recent years, machine learning has emerged as a promising tool for diabetes prediction and prevention. In the next chapter, we will explore the role of machine learning in diabetes mellitus prediction, and review the literature on various machine learning approaches and their effectiveness.

Chapter 2

Machine Learning and Diabetes Prediction

2.1 Introduction

In recent years, machine learning has emerged as a powerful tool for diabetes prediction, offering higher accuracy and efficiency than traditional statistical methods.
This chapter aims to provide a comprehensive understanding of machine learning and its different types, as well as their applications and evaluation metrics, in the context of diabetes prediction. By the end of this chapter, readers will be able to understand machine learning concepts and techniques, which will enable them to build and evaluate predictive models for diabetes.

2.2 Definition

Machine learning is a technology that involves developing computer algorithms capable of imitating human intelligence. It incorporates concepts from various disciplines, including artificial intelligence, probability, statistics, computer science, and information theory, among others. This technology has been applied across numerous fields, including pattern recognition, computer vision, spacecraft engineering and diseases prediction.
A machine learning algorithm is a computational process that can perform a task without being programmed to produce a specific outcome. The most noteworthy aspect of these algorithms is their ability to learn about the surrounding environment from input data, with or without guidance from a teacher. This allows them to continually improve their performance, making them a powerful tool for processing large datasets and making accurate predictions [1].

2.3 Machine Learning Approaches

Machine learning techniques can be categorized into four different approaches: supervised, unsupervised, semi-supervised, and reinforcement learning based on whether the output values need to be present in the training data or not[1], as shown in figures 2.1 and 2.2.

2.3.1 Supervised Learning

Supervised learning methods necessitate knowing the output variable value for each training sample. Therefore, each sample comprises a set of input and output values. The algorithm then trains a model that utilizes the defined features to predict the output variable value from the input variables [13] .
Supervised learning can be divided into two main categories: classification and regression.

a) Classification

When the output variables take on a discrete set of values, the predictive model is referred to as a "classifier." Automated medical diagnosis is an example of a typical classification problem where a patient's data must be classified as having a certain disease or not [13].

b) Regression

When the output variables are continuous, the predictive model is known as a "regression function." For instance, predicting the temperature at a specific time of year is an example of a regression problem [13].

2.3.2 Unsupervised Learning

Unsupervised learning techniques require only the input feature values in the training data and the learning algorithm discovers hidden structure in the training data based on them. Clustering techniques that try to partition the data into coherent groups fall into this category [13].

2.3.3 Semi-supervised Learning

Semi-supervised learning is a machine learning approach that involves training with a small set of labeled data along with a larger set of unlabeled data. This type of learning falls between unsupervised learning, which has no labeled data, and supervised learning, which solely relies on labeled data [W2].

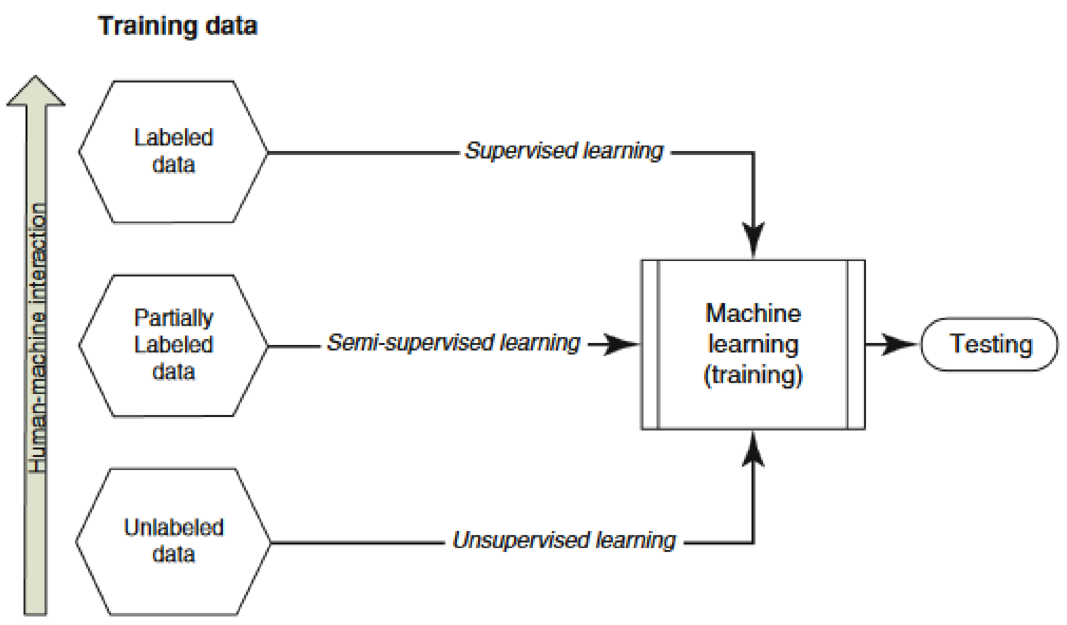


Figure 2.1: Categories of ML algorithms [1].

2.3.4 Reinforcement Learning

Reinforcement learning is a type of dynamic programming used in artificial intelligence to train algorithms through a system of rewards and punishments. In this learning approach, an agent interacts with its environment and receives rewards for performing tasks correctly or penalties for incorrect performance. Through this process of trial and error, the agent learns to maximize its rewards and minimize its penalties without any human intervention. For example, in games, the agent can learn from every move made, whether it was correct or not [14].

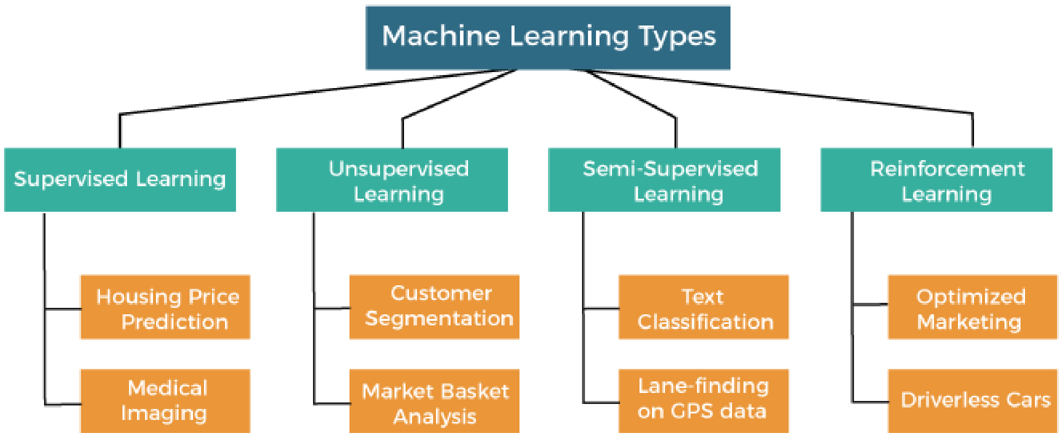


Figure 2.2: The four different Machine Learning types [W3].

2.4 Machine Learning Algorithms

2.4.1 K-Means

The K-Means clustering algorithm is a widely used and straightforward analytical technique that involves selecting a training set and a predetermined number of clusters (k) to identify. The algorithm then groups items in the training set into clusters based on their similarity, which is often determined by measuring their distance from each other using metrics like Euclidean distance[15].
Figure2.3 explains the clustering using K-means technique:

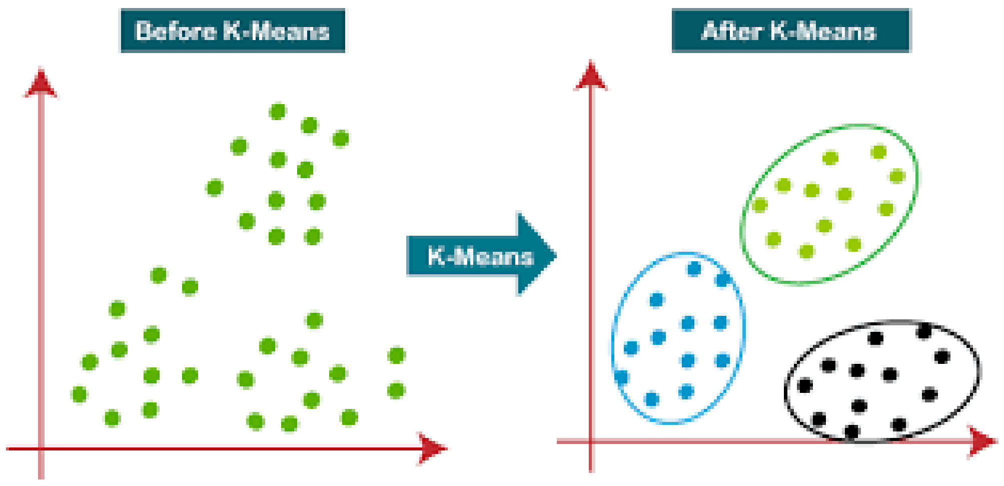


Figure 2.3: K-Means clustering algorithm [W4].

2.4.2 Support Vector Machine

Support Vector Machines (SVM) are popular machine learning technique used for classification and other learning tasks. SVM is a discriminative classifier that is characterized by an optimal hyperplane. The hyperplane is used to classify new examples, and the data points that support the hyperplane are called support vectors. In a two-dimensional region, the hyperplane is a line that separates the data into two segments. The selection of the optimal hyperplane is not a trivial task, as it should be noise-insensitive and accurately generalize the data set. SVM tries to find an optimized hyperplane that provides a significant minimum distance to the trained data set. In mathematical notation, for a two-dimensional space, a line can distinguish linearly separable data, and its equation is y = ax + b.
Renaming x with x1 and y with x2, the equation becomes ax1 - x2 + b = 0.
By defining X = (x1, x2) and w = (a, -1), we get wx + b = 0, which is known as the equation of the hyperplane. The figure below shows an example of multiple line data

classification using two distinct datasets, squares and dots [16].
Figure2.4 shows the data classification using multiple lines vs optimal hyperplane.

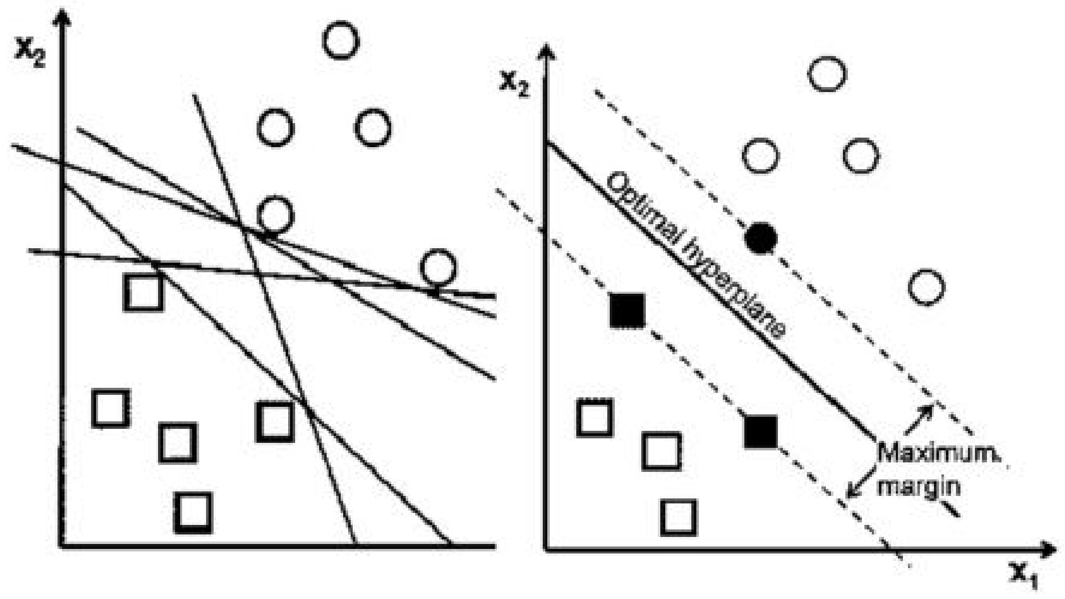


Figure 2.4: Data classification using multiple lines vs optimal hyperplane [W5].

2.4.3 Linear Regression

Linear regression is a statistical method that estimates the value of a dependent variable based on one or more independent variables. This technique evaluates the relationship between two variables and predicts the value of the dependent variable using the independent variable(s) [17].

2.4.4 Logistic Regression

Logistic regression is a statistical technique that employs a logistic function to model a binary dependent variable. It has extensive applications in biomedical research to model disease risk and other binary outcomes, especially when the dependent variable is dichotomous or the relationship between the dependent variable and independent variables is nonlinear. Logistic regression estimates the probability of an event occurring based on a set of predictor variables and offers insights into the impact of each predictor variable on the likelihood of the event taking place [18].
The equation for logistic regression with a single predictor variable is:

$$logit(p) = 0 + 1x \tag{2.1}$$

Where p is the probability of the response variable being equal to 1, x is the value of the predictor variable, 0 is the intercept, and 1 is the coefficient for the predictor variable.The logistic function is:

$$p = 1/(1 + exp(-logit(p))) \tag{2.2}$$

Where exp() is the exponential function.

2.4.5 Decision Trees

Decision trees are machine learning models that provide a high level of interpretability by allowing data to be stratified or segmented. These models enable the continuous splitting of data based on specific parameters until a final decision is reached. it is used in both classification and regression problems [W6].

a) How does a decision tree decide on the first variable to split on?

Entropy

Entropy is a measure of the purity of a split, indicating how the data is partitioned. This metric ranges between 0 and 1, where 0 represents a pure split and 1 represents an impure one. The selection of the first node in a decision tree is always based on the feature with the lowest entropy [W6].

Information gain

Information gain is a metric used in constructing decision trees that quantifies the reduction in entropy when making a split. Decision trees can be built in several ways, starting with the selection of a feature to split on. By using information gain, we can identify the optimal tree that minimizes entropy. The best possible tree is the one that achieves the highest information gain [W6].

b) Components of a decision tree

Root node: At the top of a decision tree is the root node, which is the initial feature used to divide the dataset. The root node represents the best feature that allows for the optimal split of data [W6].

Internal nodes: After the root node, these are the nodes that divide the data [W6].

Leaf nodes: Nodes located at the bottom of the decision tree are considered terminal nodes, beyond which additional splits cannot be made [W6].

Branches: The connections between nodes in a decision tree are known as branches, which depict the results of a test [W6]

The structure of the decision tree is demonstrated in figure2.5:

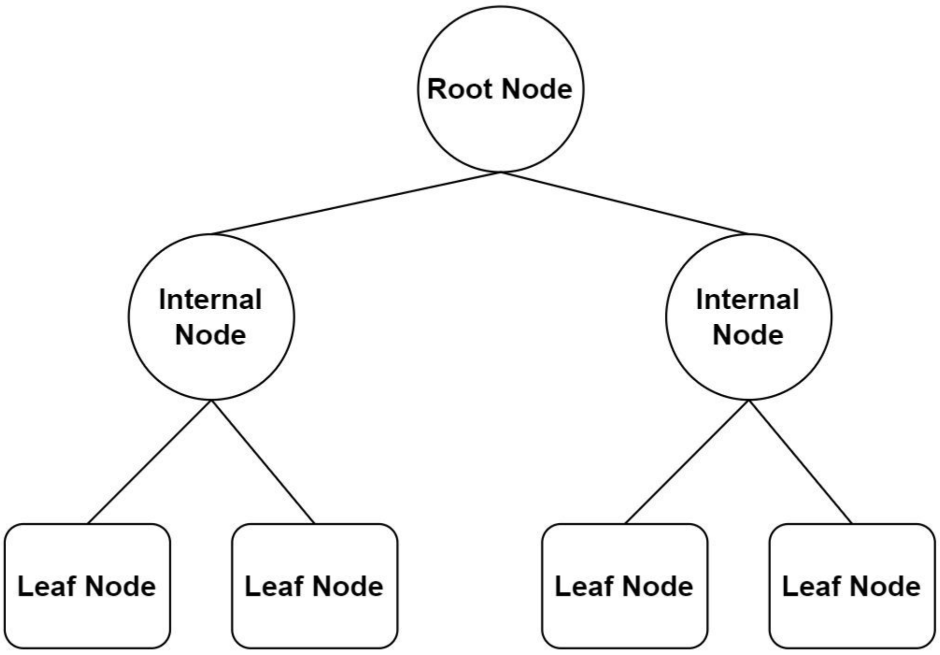


Figure 2.5: Components of a decision tree [W6].

c) Advantages and disadvantages of this algorithm [W6]

+The structure of decision trees is straightforward and easy to comprehend, making them user-friendly for interpretation.
+Decision trees are versatile and can be applied to both classification and regression problems.
+Decision trees are capable of partitioning data that is not linearly separable, broadening their applicability to a wide range of scenarios.
-Decision trees are susceptible to overfitting, which occurs when a model is overly complex and fits to noise in the training data, resulting in poor generalization to new data.
-Decision trees are sensitive to changes in the training dataset, even small ones, which can lead to significant alterations in the decision logic.

d) Example

Figure 2.6 shows an example of how to predict whether an individual is diabetic or not, using a model that includes three attributes: minimum systolic blood pressure, age, and glucose. The model has two classes, diabetic and non-diabetic. The attributes function as internal nodes that partition the decision tree into branches. At the end of each branch, there is a leaf node (the decision point) where no further divisions occur, and we can make a prediction about whether the person is diabetic or not.

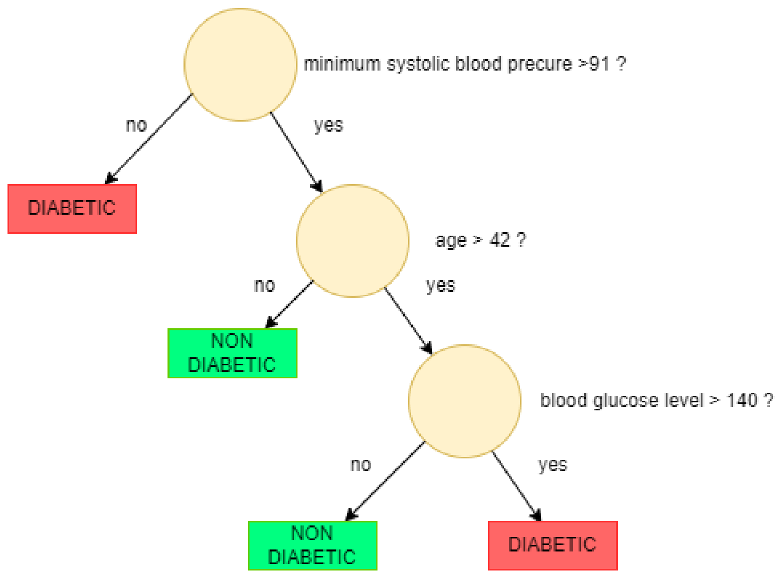


Figure 2.6: Decision tree example.

2.4.6 Random Forest

The decision tree algorithm has a significant flaw, which is its tendency to overfit. Overfitting leads to complex models with high variance that have good accuracy during training but poor generalization to other datasets. To address this issue, the random forest algorithm uses bagging, an extension of a technique that reduces model variance by averaging a set of observations (we will explain it later). The random forest combines the predictions from multiple decision trees to produce a single output [W6].

Comparison between Decision Trees and Random Forest

Random forests typically perform better than decision trees due to the following reasons: Random forests address the issue of overfitting by aggregating the predictions of multiple decision trees to arrive at a final prediction. Unlike decision trees, where small changes in data can result in significant changes in the model's prediction, random forests avoid this problem by sampling the data several times to generate a prediction. However, constructing multiple decision trees in a random forest model takes more time, making it slower than decision trees. Although the accuracy of a random forest model can be improved by adding more trees, it also increases computation time. Decision trees are simpler and easier to interpret than random forests. The decision tree algorithm is straightforward and easy to visualize, making it easier to understand the reasoning behind the algorithm's outcome. In contrast, random forests are more complex and combine the output of multiple decision trees, making them harder to deconstruct [W6].

2.5 Deep learning

Deep learning is a subfield of machine learning that focuses on algorithms utilizing artificial neural networks with a high number of layers and nodes, resembling the structure and functioning of the human brain. These algorithms process information through layers, with each layer receiving and passing on information to the next. In contrast to traditional machine learning models, deep learning models automatically extract features without human intervention. Additionally, deep learning models continue to improve performance with larger amounts of data, whereas traditional machine learning models reach a saturation point. Therefore, deep learning is a specialized form of machine learning and a component of artificial intelligence[14].

2.5.1 Artificial Neural Network:

The neural network, or artificial neural network (ANN), is a machine learning method that emulates the functioning of the human brain [2]. As diabetes prediction research faces a deluge of data, advanced analysis methods are required to identify hidden causal relationships between various properties and the likelihood of developing diabetes. The ANN is a versatile tool that can be used for diabetes prediction modeling. Unlike traditional regression approaches, the ANN can model complex nonlinear relationships and has excellent fault tolerance [2]. Additionally, the ANN's fast and highly scalable parallel processing capabilities make it an efficient option for diabetes prediction research.

a) Biological Neural Network

The brain is composed of specialized cells known as neurons, which when connected together form a complex neural network. In the human brain, there are approximately 1011 neurons, each with around 10,000 connections to other neurons[19]. Artificial neural networks (ANNs) seek to replicate this natural neural network by connecting artificial neurons in a similar fashion. A biological neuron is comprised of a cell body, axon, and dendrite. The dendrite receives signals from other neurons and transmits them to the cell

body, while the axon carries the signal from the neuron to other neurons or muscle cells. The connection between dendrites of two neurons, or between a neuron and muscle cells, is referred to as a synapse[19] [20], as shown in figure 2.7.
The dendrites of a neuron receive signals from other neurons, and if the strength of the incoming signal surpasses a specific threshold, the neuron transmits its own signal to the next neuron via the axon using synapses. The signal sent through synapses triggers neighboring neurons to fire, and this cycle repeats[21]. A vast number of neurons work together simultaneously in the brain, allowing it to store and process large amounts of information[19].

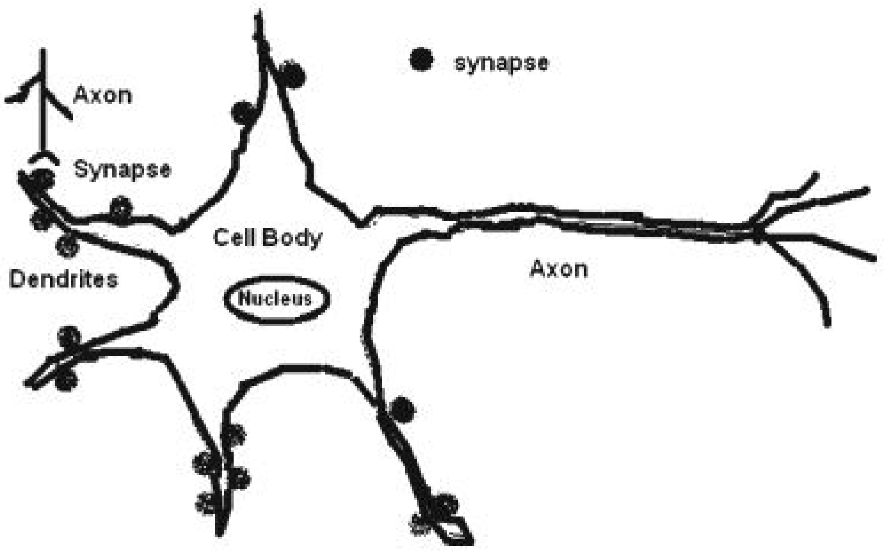


Figure 2.7: Biological neuron [2].

b) Artificial Neuron

Artificial neural networks are made up of individual processing units known as neurons, which attempt to replicate the behavior and structure of natural neurons. Similar to natural neurons, an artificial neuron has inputs, which are dendrites, and a single output, which is a synapse via an axon. The activation of the neuron is determined by a specific function[19]. (Figure 2.8).
The inputs to a neuron are represented as x1...xn, and a bias value is typically added alongside these inputs. The bias value is usually initialized to 1. Additionally, the neuron is connected to other neurons or inputs through weights, which represent the strength of the signal. Each weight is multiplied by its corresponding input to determine the overall strength of the signal received by the neuron. A neuron may receive inputs from multiple

sources, but it only produces a single output[19].

Several activation functions are used in artificial neural networks, with the sigmoid function being among the most frequently utilized. The sigmoid function can be defined as follows:

$$\sigma(x) = \frac{1}{1+e^{-sum}} \tag{2.3}$$

where sum is the sum of $x_i * w_i$.

A variety of activation functions can be employed, including the step function, linear function, ramp function, and hyperbolic tangent function. The Hyperbolic tangent (tanh) function has a similar shape to the sigmoid function, but its output values range from -1 to +1, as opposed to the sigmoid function which ranges from 0 to 1[19].

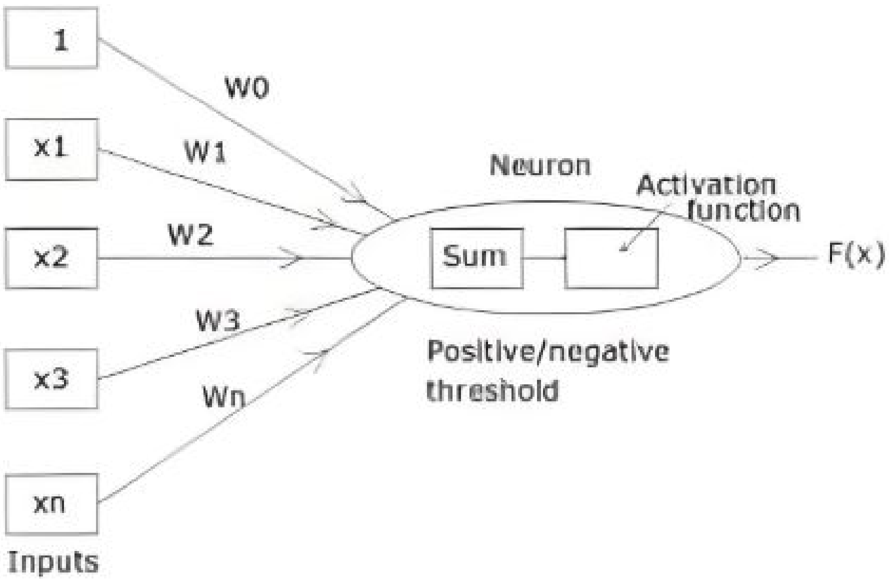


Figure 2.8: Model of an artificial neuron [3].

By multiplying the inputs with the weights between layers, a sum is obtained. This sum is weighted and passed through an activation function, which in this case is a sigmoid function. The sigmoid function is a smooth approximation of a step function and is both continuous and differentiable, as referenced in [21]. The neural network is formed by connecting multiple individual neurons together[19].

ANN architecture:
According to [19], the neural network typically consists of three layers, as shown in figure

2.9
a. Input layer, which receives the input values.
b. Hidden layer(s), which is a set of neurons located between the input and output layers. There can be a single or multiple hidden layers.
c. Output layer, which usually contains one neuron and produces output ranging between 0 and 1. However, there can be multiple outputs present as well[22].

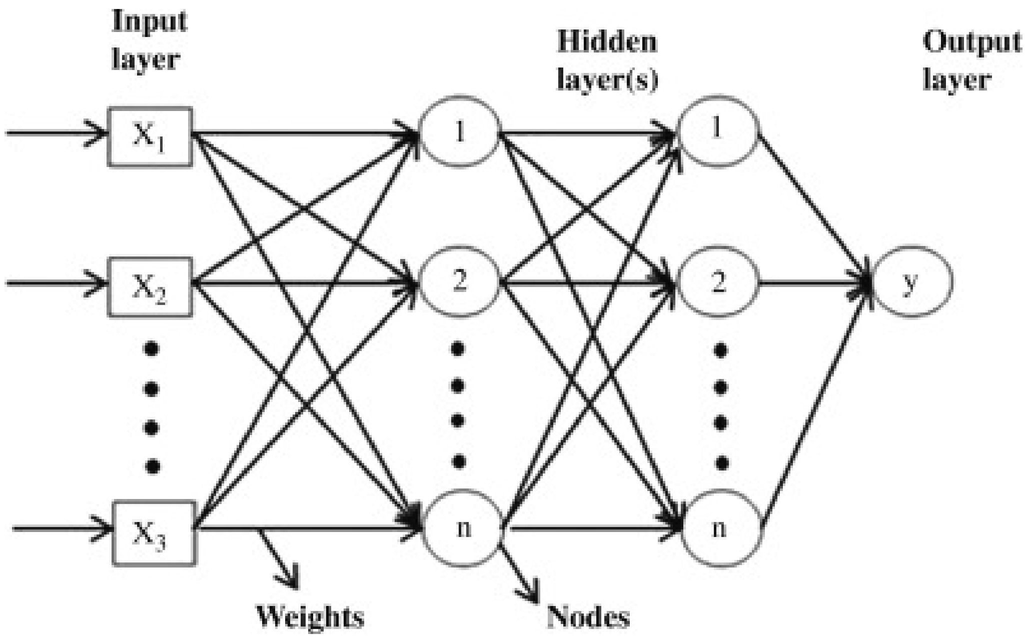


Figure 2.9: ANN architecture [4].

The processing power of an artificial neural network is stored in the inter unit connection strengths known as weights, as stated in [23]. The strength of the input signal depends on the weight value, which can be positive, negative, or zero. A negative weight value implies a reduction or inhibition of the signal, while a zero weight value indicates no connection between the two neurons. The weights need to be adjusted to produce the desired output, which can be accomplished using algorithms designed to adjust the weights of the artificial neural network. This process of modifying weights is referred to as learning or training, as mentioned in [21].

c) ANN Training

Artificial neural networks (ANNs) are categorized based on supervised and unsupervised learning methods. The simplest form of ANN architecture is the Perceptron, which contains one neuron with two inputs and one output. The Perceptron utilizes a step or ramp function as its activation function and is used for the classification of data into two separate classes. For more complex applications, multilayer Perceptrons (MLPs) are used, which consist of one input layer, one output layer, and one or more hidden layers[19].

The most commonly used method for training an ANN is the backpropagation algorithm, which is a supervised learning method that utilizes feed-forward architecture. It is frequently used for classification and prediction, and the weights between neurons are adjusted by propagating the difference between the targeted and obtained output back to the layers. In the backpropagation algorithm, the output of the hidden layers is propagated to the output layer to calculate the output. This output is then compared to the desired output, and the error is propagated back from the output layer to the hidden layer and input layer, adjusting the weights between the neurons. This process is called an epoch, and the network undergoes many epochs until the error is within a certain tolerance. Once the network is trained, the weights between the neurons in all layers are set. These trained weights are then used to calculate the response of the network to unknown data[19].

2.5.2 Deep Learning Techniques

a) Fully Connected Deep Neural Networks

Fully Connected Neural Networks, commonly referred to as multilayer perceptrons, are a class of neural networks characterized by the presence of connections between every neuron in one layer and every neuron in the subsequent layer. These networks demonstrate notable efficacy in handling tabular datasets formatted in CSV, effectively addressing classification and regression tasks involving real-valued input variables. Moreover, they exhibit considerable versatility, allowing their application to a diverse range of problem domains, akin to other artificial neural network architectures [24].

b) Convolutional Neural Network

Convolutional Neural Networks (CNNs) are a type of deep learning algorithm that has gained significant popularity for their excellent performance in image and video analysis tasks. They are designed to extract spatial hierarchies of features from input images by utilizing convolutional layers that apply filters to input data at different resolutions and positions. The learned features are then merged and passed through fully connected layers to produce output predictions. CNNs have demonstrated superior performance on numerous image and video analysis tasks, such as object detection, image segmentation, and action recognition[25].

c) Recurrent Neural Network

Recurrent neural networks (RNNs) are a specialized type of neural network intended to process sequential data. Unlike feedforward neural networks that analyze the complete input all at once, RNNs examine input sequences one by one while keeping an internal state that retains information about prior elements. This internal state allows RNNs to adeptly model the temporal connections in the input data. RNNs have found wide-ranging applications in various fields, including speech recognition, language modeling, and machine translation, among others, and have demonstrated impressive results[26].

d) Autoencoders

Autoencoders are a type of neural network that operates without supervision and is designed to encode input data into a reduced-dimensional representation, then decode it back to its original form, all while minimizing the reconstruction error. Autoencoders are typically composed of an encoder that takes in the input data and maps it into a compressed representation called a latent code, and a decoder that generates the reconstructed output from the latent code. They have found widespread use in applications like data compression, feature extraction, and dimensionality reduction[27].

2.6 Ensemble Learning

Ensemble learning is a machine learning approach that involves training multiple models or applying multiple learners to datasets in order to solve the same problem. Each model generates its own prediction, and these predictions are then combined into a composite prediction. The ensemble consists of a set of models, each trained through a learning process on the same problem, and their outputs are integrated in some manner to obtain the final prediction [5].
Ensemble learning can be applied to various types of machine learning algorithms, including both supervised and unsupervised learning, as well as regression and classification tasks. Some popular ensemble methods include:

2.6.1 Bootstrap Aggregating (Bagging)

Bagging, or Bootstrap Aggregating, is an ensemble learning technique where each model in the ensemble casts an equal-weighted vote. To promote model variance, bagging trains each model using a randomly sampled subset of the training set. For example, the random forest algorithm combines bagging with random decision trees to achieve high classification accuracy [5].
Bagging involves creating multiple training sets of size n (instead of just one training set) and building a classifier for each training set. The predictions of these classifiers are then combined through voting or averaging, with each model receiving equal weight. This "idealized" version of bagging helps to reduce variance error by leveraging different learners on different subsets of the training data [5].
In generalized bagging, different learners can be used on different populations, further reducing variance error and improving the ensemble's performance [5].

Figure 2.10 shows the different steps of the Bagging technique.

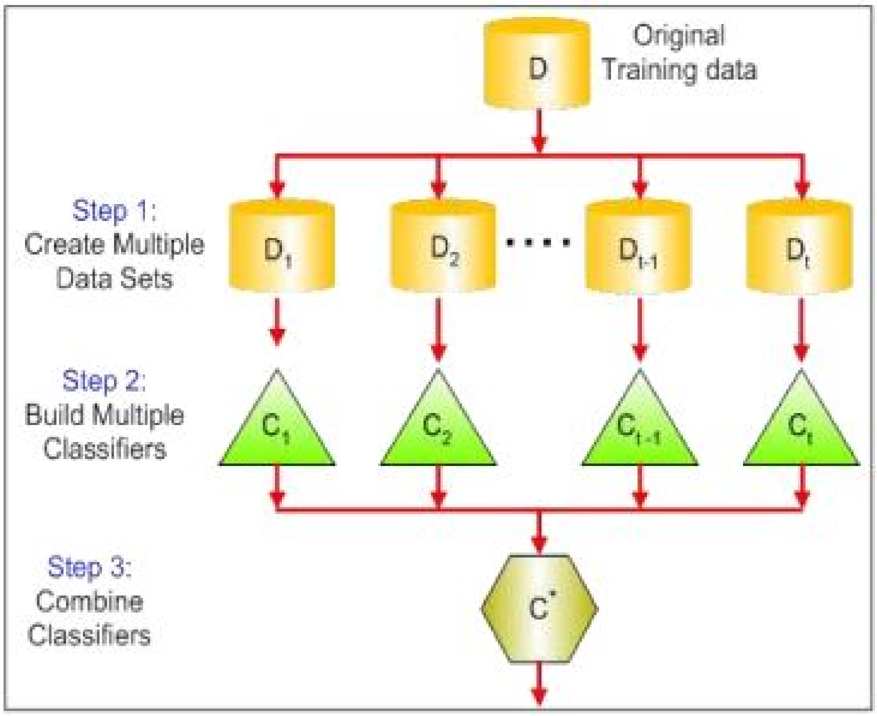

D
Original Training data
Step 1: Create Multiple Data Sets
D_1
D_2
D_{t-1}
D_t
Step 2: Build Multiple Classifiers
C_1
C_2
C_{t-1}
C_t
Step 3: Combine Classifiers
C^*

Figure 2.10: Bagging technique [5].

2.6.2 Stacking

Stacking is an ensemble learning technique that involves combining multiple base classification models using a meta-classifier. This approach aims to create a generalized machine learning model by training different base learning models (L1, ..., LN) on the same dataset S, which consists of examples (si = (xi, yi)), where xi is a feature vector and yi is its corresponding target class. To generate a training set for the meta-level classifier, a leave-one-out or cross-validation procedure is applied, where the meta-classifier is trained on the predictions of the base models for the withheld examples. This way, stacking utilizes the predictions of multiple base models to build a more powerful and accurate ensemble model [28].

2.6.3 Voting

Voting is a straightforward and effective ensemble algorithm that can be used for both classification and regression problems. It involves creating multiple sub-models, each of which makes predictions. These predictions are then combined in some manner, such as by taking the mean (soft voting) or mode (hard voting) of the predictions, allowing each sub-model to contribute to the final outcome through voting[5].
Figure 2.11 shows the two types of voting technique.

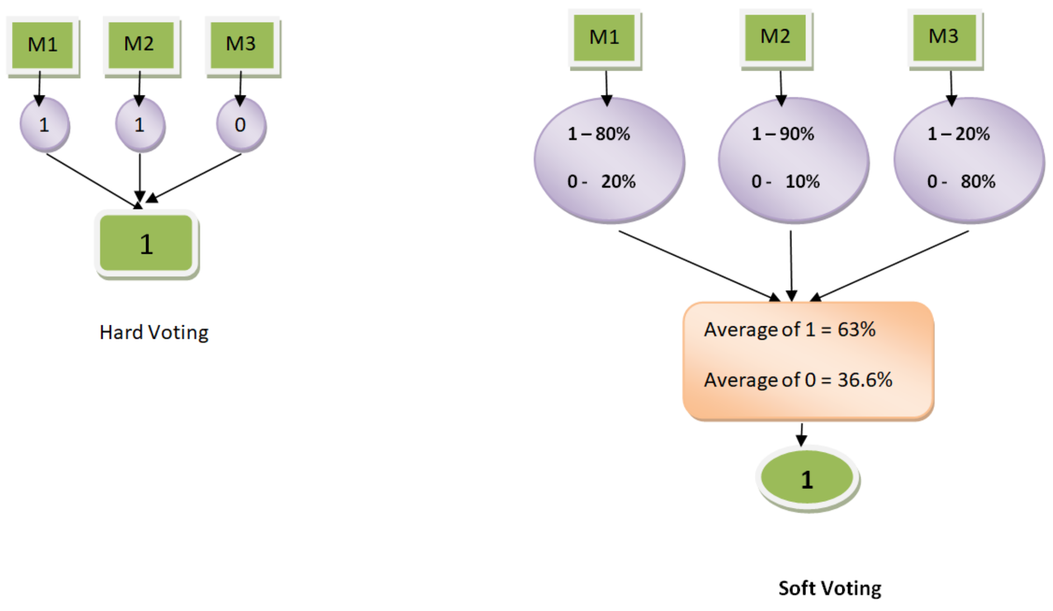


Figure 2.11: Voting types [W7].

2.6.4 Boosting

Boosting is a powerful machine learning technique that integrates predictions from multiple weak learners, such as decision trees or linear models, to create a robust ensemble model with enhanced predictive performance. The boosting algorithm sequentially fits the weak learners to the training data, with each subsequent learner focusing on samples that were misclassified by the previous learners. The predictions of the weak learners are then combined, often using weighted averaging, to generate the final prediction. Boosting is an iterative process that adjusts the weights of the training samples to emphasize misclassified samples, thus assigning more importance to challenging samples that are difficult to classify accurately. This technique has been proven effective in various machine learning tasks, including classification, regression, and feature selection, and is widely adopted in practical applications due to its ability to significantly improve predictive accuracy of models [29].

Examples of boosting techniques used in machine learning include:

- AdaBoost (Adaptive Boosting).
- Gradient Boosting.
- XGBoost (Extreme Gradient Boosting).

- LightGBM (Light Gradient Boosting Machine).
- CatBoost (Categorical Boosting).

2.7 Cross Validation

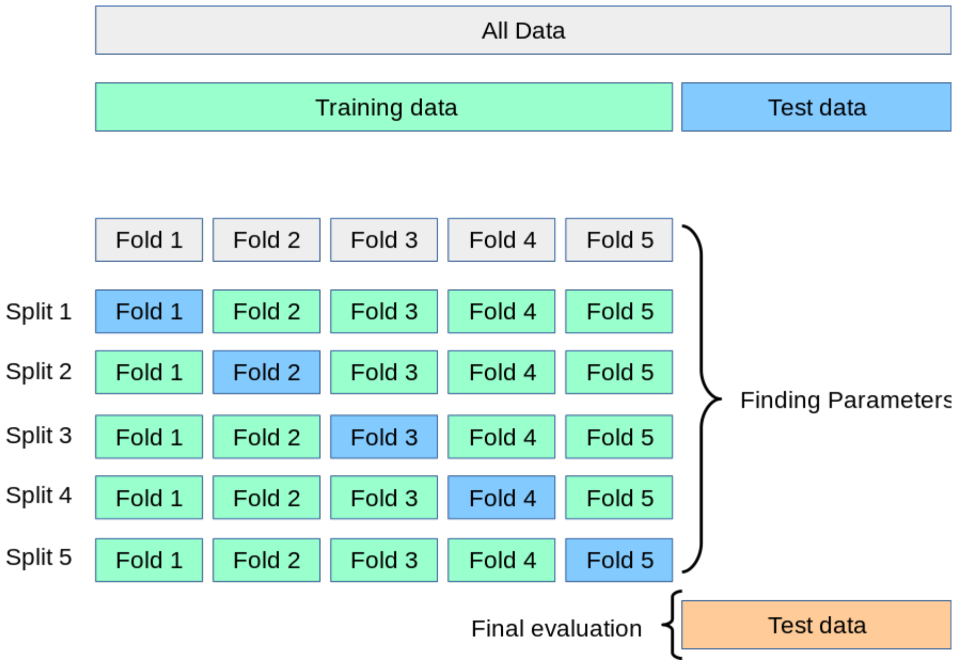


Figure 2.12: Cross validation technique [W8].

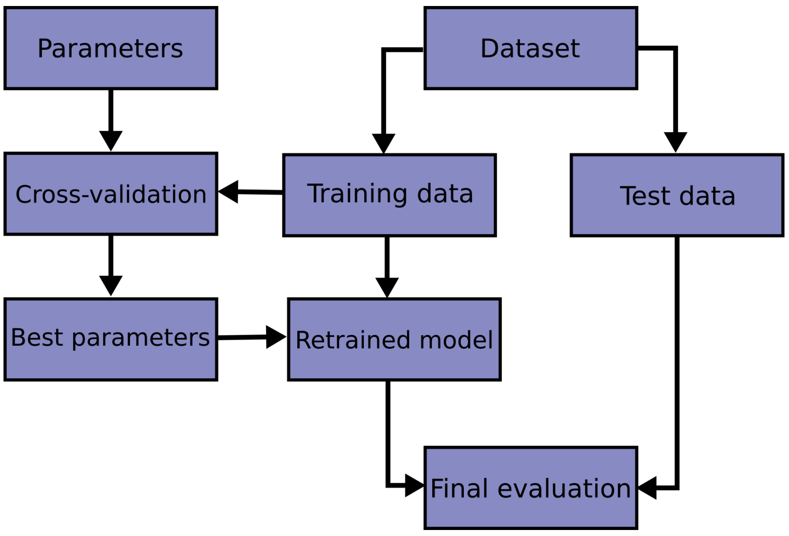


Figure 2.13: Cross validation workflow in model training [W8].

The performance of classification algorithms is commonly assessed using k-fold cross-validation. This evaluation technique involves dividing the dataset into k separate and non-overlapping folds, ensuring that each fold contains a roughly equal number of instances. Each fold takes turns serving as the testing set, while the model is trained on

the remaining k-1 folds. This process is repeated k times, with each fold serving as the testing set exactly once. By using cross-validation, we mitigate the impact of the random partitioning of the data. So, we can avoid overfitting [30] [31].
Figures 2.12 and 2.13 explain the process of cross-validation technique.

2.8 Evaluation metrics

Being able to evaluate a model's performance is essential both for assessing risks and for comparing different algorithms or models. Performance metrics have been mainly designed to address the question of how reliable a model is at predicting future events. Furthermore, various parameters need to be computed, including:
True Positive (TP): The cases predicted 1 and the actual output was also 1.
True Negative (TN): The cases predicted 0 and the actual output was 0.
False Positive (FP): The cases predicted 1 and the actual output was 0.
False Negative (FN): The cases predicted 0 and the actual output was 1.

2.8.1 Confusion Matrix

The confusion matrix is a widely used evaluation measure in classification tasks, applicable to both binary and multiclass classification problems. Table 2.1 illustrates the confusion matrix.

		Predicted classes	
		class = Negative	class = Positive
Actual classes	class = Negative	TN	FP
	class = Positive	FN	TP

Table 2.1: Confusion matrix.

2.8.2 Accuracy

Accuracy can be calculated as follows:

$$\textbf{Accuracy} = \frac{\textbf{TP+TN}}{\textbf{TP+TN+FP+FN}} \tag{2.4}$$

2.8.3 True Positive Rate/ Recall/ Sensitivity

Recall can be defined as the proportion of true positives with respect to all the positives that exist in the ground truth.

$$\textbf{Recall} = \frac{\textbf{TP}}{\textbf{TP+FN}} \tag{2.5}$$

2.8.4 Precision

Which is the percentage of true positive predictions out of all positive predictions.

$$\textbf{Precision} = \frac{\textbf{TP}}{\textbf{TP+FP}} \tag{2.6}$$

2.8.5 F-Measure

The F-measure is defined as a harmonic mean of precision and recall.

$$\textbf{F1-Mesure} = \mathbf{2} \times \frac{\textbf{precision * recall}}{\textbf{precision+recall}} \tag{2.7}$$

2.8.6 Precision-Recall curve

A Precision-Recall curve is a graphical representation of the performance of a binary classifier model, which plots the precision (proportion of true positive predictions among positive predictions) against the recall (proportion of true positive cases among all actual positive cases) at different classification thresholds. It is useful in situations where the class distribution is imbalanced and the accuracy of the classifier is not a good measure of performance, particularly when false negatives are more important to avoid than false positives. An ideal classifier would have a PR curve that passes through the point (1,1) with an area under the curve of 1, while a random classifier would have a diagonal line with an AUC of 0.5.

2.9 Related Works

Machine learning and deep learning methods have been extensively used in the prediction of diabetes. Many studies have focused on developing accurate and reliable predictive models for diabetes using various machine learning and deep learning algorithms. We will show you some studies in this field:

In [32], authors focused on predicting diabetes using a stacking classifier approach. They employed the Pima dataset, which is commonly used for diabetes prediction. The stacking classifier methodology involved combining multiple base classifiers and utilizing a meta classifier for the final prediction. In this study, the authors employed six base classifiers. The meta classifier utilized in the stacking approach was logistic regression. The stacking classifier model was trained and evaluated using the Pima dataset. The results showed that the stacking classifier achieved an accuracy of 82.68 % in predicting diabetes.

Also, an approach that combines the output probabilities of different machine learning algorithms, and other algorithms like Xgboost, Bagging, K-Nearest Neighbors, Support Vector Machines, Random Forest, and Decision Trees has been proposed in [33]. The study evaluates the proposed approachs using the Pima Indian diabetes dataset and demonstrates that the ensemble approach with the soft voting classifier outperforms individual machine learning algorithms, achieving an accuracy of 79.08%.

Moreover, authors in [34] developed a predictive model for the early diagnosis of diabetes mellitus using PIMA dataset with The XGBoost algorithm and Data feature stitching . they achieved an accuracy of 80.2% . The study analyzed the feature importance of the dataset and identified the top five features that were most relevant for predicting diabetes.

In the study of Mercaldo et al.[35]. They utilized six distinct classifier algorithms, including the Multilayer Perceptron, JRip, Hoeffding Tree, J48, BayesNet, and Random Forest algorithms. The authors focused their research on the PIMA Indian Dataset, using the Best Initial and Greedy Stepwise algorithms to assess the differential characteristics that help to define concept classification. Specifically, they examined four characteristics: age, BMI, plasma glucose concentration, and diabetes pedigree function. The authors used ten-fold cross-validation on the dataset and evaluated the classifiers based on recall, accuracy, and F-measure. Their findings indicated an accuracy value of 75.7%, an F-measure value of 75.9%, and a recall value of 76.2%. Furthermore, the Hoeffding Tree algorithm performed the best compared to the other algorithms.

Authors in [36] employed the Decision Tree, SVM, and Naive Bayes classifiers algorithms to predict diabetes. Their primary goal was to determine the classifier with the highest accuracy. The PIMA Indian Dataset was utilized in their research. The authors conducted 10-fold cross-validation partitioning and evaluated the classifiers based on recall, accuracy, precision, and F-measure. According to the authors, "the Naive Bayes classifier achieved the highest accuracy, with a measurement of 76.30%".

In addition, Sivaranjani et al used machine learning algorithms to predict diabetes in a sample of patients, and compared the performance of these al- gorithms with and without feature selection and dimensionality reduction techniques. Feature selection involves selecting a subset of the most relevant features from the data- set, while dimensionality reduction techniques aim to reduce the number of features in the dataset. The authors used a number of different machine learning algorithms, including decision trees, k-nearest neighbors (k-NN), and support vector machines (SVMs), and compared their performance in terms of accuracy and other metrics. They found that all three algorithms performed better when used with feature selection and dimensionality reduction techniques, with the highest accuracy achieved by the Random Forest algorithm (83%) when combined with the principal component analysis (PCA) dimensionality reduction technique [37].

Authors in [38] introduced a method called Hierarchical Multi-level classifiers bagging with Multi-objective upgraded Voting (HMBag Moov) for classifying diabetes. They compared this method with several other strategies, including Naïve Bayes (NB), Support Vector Machine (SVM), Logistic Regression (LR), Quadratic discriminant analysis (QDA), K-Nearest Neighbors (k-NN), Random Forest (RF), and Artificial Neural Network (ANN). However, the authors did not use hyper-tuning and cross-validation techniques and only considered a limited number of machine learning algorithms for ensembling. The HM-Bag Moov Voting Classifier achieved an accuracy of 77.21%.

Authors in [39] aims to develop a diabetes prediction model and evaluates the performance of eleven machine learning algorithms on the Pima Indians Diabetes Database. After cross-validation and hyper-tuning, K-neighbors, SVC, and MLP are identified as the top three classifiers among eleven classifiers. An Ensemble Voting Classifier combining these three classifiers achieves an accuracy of 86%, outperforming the other algorithms.

Febrian et al in their study utilized supervised machine learning to predict diabetes using PIMA dataset. Two k-Nearest Neighbor algorithms and the Naive Bayes algorithm were compared, and the study concluded that the Naive Bayes algorithm outperformed KNN. The Confusion Matrix was used to evaluate the algorithms, and the Naive Bayes algorithm had an average accuracy of 76.07%, precision of 73.37%, and recall of 71.37%, while KNN had an average accuracy of 73.33%, precision of 70.25%, and recall of 69.37% [40] .

Yahyaoui et al compared machine learning and deep learning-based algorithms to predict diabetes using PIMA dataset The results indicated that RF was the most effective algorithm for classifying diabetes in all rounds of experiments, with an overall prediction accuracy of 83.67%. SVM had a prediction accuracy of 65.38%, while the DL method produced a prediction accuracy of 76.81% [41].

El_Jerjawi et Abu-Naser proposed an Artificial Neural Network (ANN) based diabetes prediction model with an average error function of 0.01% and an accuracy rate of 87.3%. This model could be very useful for healthcare officials and practitioners, given the severe complications of the disease [42].

A new model for classifying type 2 diabetes data was proposed by authors in [43]. Which utilized a deep neural network (DNN) constructed by cascading stacked autoencoders with a softmax classifier. This model achieved a classification accuracy of 86.26%.

In their proposal, authors in [44] suggested incorporating a hybrid evolutionary method with a convolutional neural network (CNN), and customized the number of layers and filters to suit the specific application and user requirements.

More recently, Rastogi and Bansal (2023) [45] proposed a diabetes prediction model and they found that the accuracy of the logistic regression method was higher than three other data mining techniques which are: RF, SVM, and NB. While Hou et al. 2023 [46] found that Random Forest outperformed Logistic Regression with an AUR of 0.815.

The primary challenge prevalent in previous machine learning methodologies pertains to the selection of a suitable classifier or the optimal combination of algorithms to attain superior outcomes. Consequently, our proposed approach relies upon meticulous data preprocessing and the precise integration of algorithms to surpass alternative techniques.

2.10 Conclusion

In this chapter, we provided an overview of machine learning and deep learning techniques for diabetes prediction. We showed various algorithms and related works in the field, as well as evaluation metrics for assessing model performance. We found that machine learning and deep learning methods have the potential to improve diabetes management, but further research is needed to address the challenges in developing accurate and reliable predictive models. This chapter provides a foundation for the methodology and implementation details of our proposed diabetes prediction system, which we will present in the next chapters.

Chapter 3

Methodology and Implementaion

3.1 Introduction

The rise of diabetes mellitus has prompted the development of various prediction models to aid in early detection and prevention. Machine learning techniques, in particular, have been leveraged to identify individuals at risk of developing diabetes based on a multitude of factors. Our approach to improving the accuracy of these models is through ensemble learning , where multiple models are combined to provide a more robust prediction. In this chapter, we will present the methodology and implementation of our proposed model.

3.2 Environment

3.2.1 Hardware

- **CPU :** Intel(R) Core(TM) i3-3110M CPU @ 2.40GHz
- **RAM :** 8 GB
- **Drive :** 1 TB HDD

3.2.2 Software

Jupyter

Jupyter is a freely available web-based tool that enables users to generate and disseminate documents containing executable code, mathematical expressions, graphics, and written explanations. It accommodates a wide range of programming languages and creates an interactive computing workspace for researchers, data scientists, and programmers [W9].

Python

Python is a dynamically-typed, interpreted high-level programming language that focuses on simplicity, readability, and maintainability. It features a modular architecture, which enables developers to organize their code into reusable and extensible components. Python comes with a comprehensive standard library and a vast collection of third-party packages, making it a versatile language that can be used for various purposes such as scientific computing, data analysis, machine learning, and web development. It is also known for its ease of use, which makes it a popular choice among beginners and experts alike [47].

Used Libraries

NumPy

NumPy is a Python library that supports numerical computing through an extensive collection of mathematical functions, and offers a flexible and efficient support for large, multi-dimensional arrays and matrices. It is a critical component of scientific computing with Python, and its applications are widespread in various fields such as engineering, physics, and data science.
NumPy's array objects are more efficient and powerful than Python's built-in data structures for numerical calculations. It allows for vectorized operations on arrays, which can significantly reduce computation time. Additionally, NumPy includes functionalities such as linear algebra, Fourier analysis, random number generation, and provides tools to connect with other programming languages and libraries [48].

Pandas

Pandas is a Python library that is open-source and designed to provide data structures for handling large and complicated data sets in an efficient manner. Additionally, it includes an array of tools for tasks such as data cleaning, transformation, and exploration. It is commonly used for managing tabular data, which is frequently encountered in scientific applications such as social science and bioinformatics [49].

Sklearn

Scikit-learn, or sklearn, is a Python library that offers a wide range of machine learning algorithms for both supervised and unsupervised learning tasks. It is built on top of well-established libraries such as NumPy, SciPy, and Matplotlib, and provides a user-friendly interface that facilitates data manipulation. Scikit-learn includes algorithms for classification, regression, clustering, and dimensionality reduction, as well as tools for data preprocessing, model selection, and evaluation.
Due to its ease of use, flexibility, and scalability, Scikit-learn has gained popularity in both

academic research and industry. Its comprehensive documentation and active community of users and developers also make it a valuable resource for anyone interested in machine learning [50].

Matplotlib

Matplotlib is a Python library used for creating 2D plots, including static, animated, and interactive visualizations. It is a popular tool in scientific computing for data exploration and visualization. With a comprehensive set of 2D plotting functions and a high degree of customization, Matplotlib enables users to create sophisticated and complex plots with ease. It is also compatible with various other Python libraries and frameworks, which enhances its versatility for visualizing data in different contexts [51].

Seaborn

Seaborn is a Python data visualization library that builds upon matplotlib and is closely integrated with pandas data structures. Its primary focus is on visualization, making it a valuable tool for exploring and understanding data [52].

Tkinter

The Tkinter module ("Tk interface") is the standard Python interface to the Tk GUI toolkit from Scriptics (formerly developed by Sun Labs). Both Tk and Tkinter are available on most Unix platforms, as well as on Windows and Macintosh systems. Starting with the 8.0 release, Tk offers a native look and feel on all platforms[53].

3.3 Data Collection

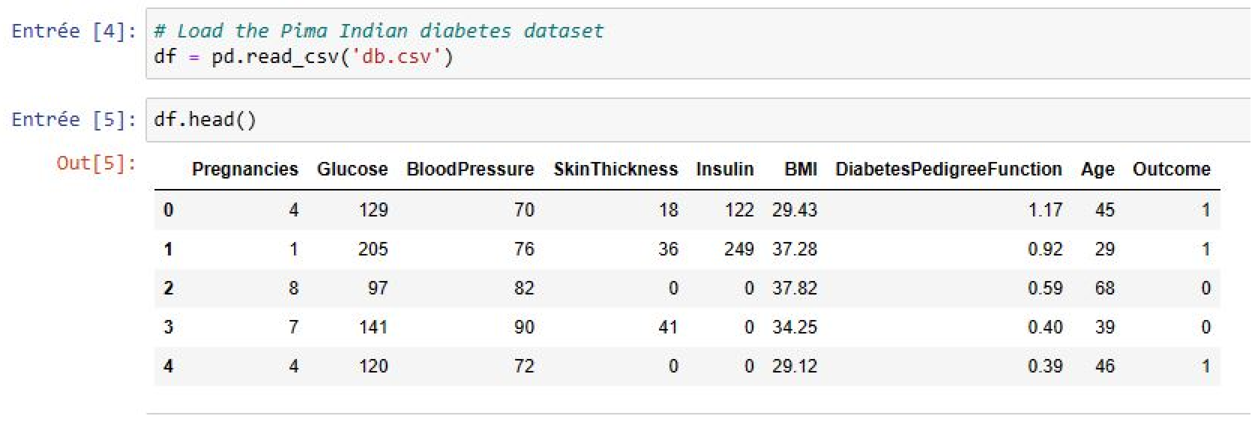
The dataset used in this study was the Pima Indian Diabetes, which is a well-known dataset for predicting the onset of diabetes. The original dataset contains 768 instances, each with eight attributes:

- Pregnancies: Number of times pregnant.
- Glucose: Plasma glucose concentration a 2 hours in an oral glucose tolerance test.
- BloodPressure: Diastolic blood pressure (mm Hg).
- SkinThickness: Triceps skin fold thickness (mm).
- Insulin: 2-Hour serum insulin (mu U/ml).
- BMI: Body mass index.
- DiabetesPedigreeFunction: Diabetes pedigree function.
- Age : Age(years).

And one binary output variable (dependent variable):
Outcome: Class variable (0 or 1).

To improve the performance of the model, we used an augmented dataset with 77,568 instances that was sourced from a reliable external dataset repository, "Kaggle" [W10]. The augmented dataset contains the same attributes as the original dataset.
We acknowledge that the use of an augmented dataset may have limitations, such as introducing bias and noise into the model. However, we believe that the use of a larger dataset is necessary to improve the accuracy and robustness of our diabetes prediction model.
Figure 3.1 represents how we can load the dataset from a CSV file named 'db.csv' into a Pandas DataFrame called 'df' and then display the first five rows of the DataFrame:
The code in figure 3.2 separates the original DataFrame 'df' into two parts:
X: which contains the features, and y: which contains the corresponding labels (0/1).

CSV files: CSV (Comma-Separated Values) files are used to store tabular data in plain text format. Each line in the file represents a row of data, and the values within each row are separated by a delimiter, usually a comma. CSV files are widely adopted for saving and sharing data due to their simplicity and compatibility with various software tools and computer languages.

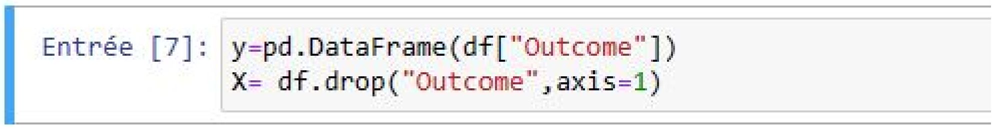


```
Entrée [4]: # Load the Pima Indian diabetes dataset
            df = pd.read_csv('db.csv')

Entrée [5]: df.head()

Out[5]:
   Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin    BMI  DiabetesPedigreeFunction  Age  Outcome
0            4      129             70             18      122  29.43                      1.17   45        1
1            1      205             76             36      249  37.28                      0.92   29        1
2            8       97             82              0        0  37.82                      0.59   68        0
3            7      141             90             41        0  34.25                      0.40   39        0
4            4      120             72              0        0  29.12                      0.39   46        1
```

Figure 3.1: Loading the dataset.



```
Entrée [7]: y=pd.DataFrame(df["Outcome"])
            X= df.drop("Outcome",axis=1)
```

Figure 3.2: Dataset separation.

3.4 Exploratory Data Analysis

Exploratory Data Analysis (EDA) is an approach used in statistics and data analysis to summarize and visualize datasets, identify patterns and relationships among variables, and detect potential outliers or anomalies.
Tukey emphasizes that EDA is not a substitute for formal statistical inference, but rather a complementary approach that can provide valuable insights into the data and guide the selection of appropriate statistical models and methods. EDA techniques include plotting histograms, scatter plots, box plots, and other visualizations, as well as calculating summary statistics such as mean, median, standard deviation, and correlation coefficients [54].

3.4.1 Outcome Distribution

Figure 3.3 represents the pie chart of our dataset, we can see that the dataset is imbalanced. There are 65.1% negative cases (labeled as 0) and 34.9% positive cases (labeled as 1) in the dataset. This means that if we were to build a machine learning model to predict diabetes, it would be biased towards predicting negative cases since they are over-represented in the dataset.

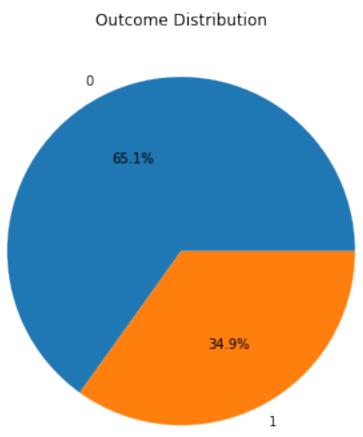


Figure 3.3: Outcome distribution.

3.4.2 Features Correlation

As demonstrated in figure 3.4, Age and Pregnancies, SkinThickness and BMI, Insuline and Glucose are highly correlated.

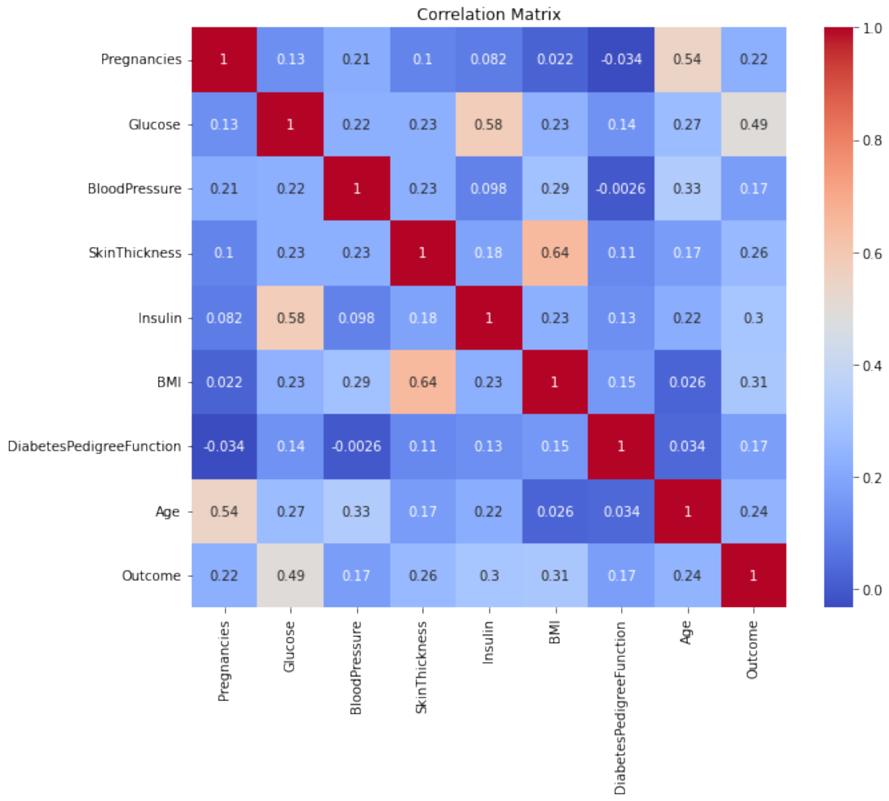


Figure 3.4: Features correlation.

3.4.3 Checking Missing Values

In our dataset, missing values are represented by zeros for some features such as glucose, blood pressure, and BMI. Therefore, in this case, we need to replace those zeros

with NaN values before detecting and plotting missing values.
After doing this step, we get this bar plot (figure 3.5):

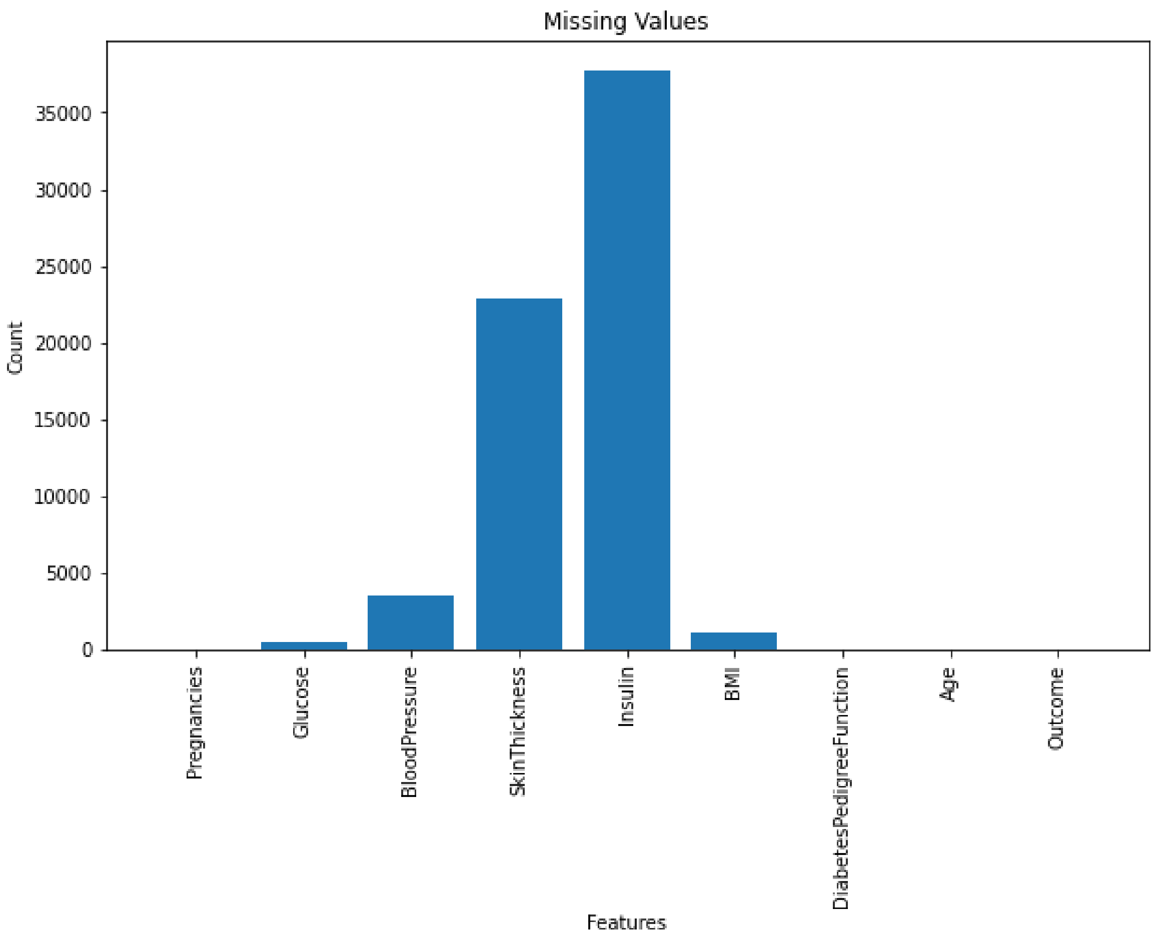


Figure 3.5: Missing values bar plot.

We see that there are 5 features that contain missing values:
Glucose, BloodPressure, SkinThickness, Insulin, and BMI.

It is important to handle missing values appropriately for the mentioned features by imputing them., and we will see how in the next section.

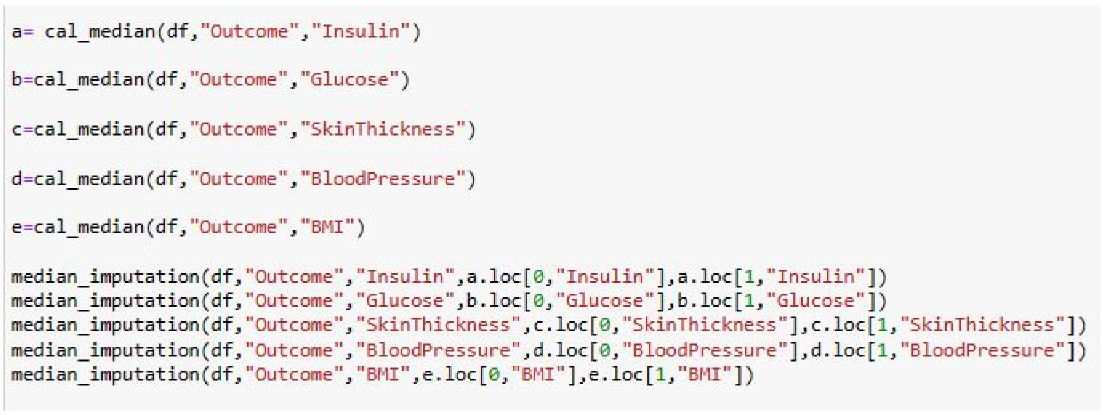
3.5 Data Preprocessing

3.5.1 Data Imputaion

Imputation is the process of replacing missing values in a data set with substituted values. This is a common approach to handling missing values in a data set when there is insufficient information to infer the actual missing values or when excluding incomplete data points would result in significant data loss.There are several methods of imputation that can be used, including: Mean, Mode imputation.

In our case, we chose median imputation by target, which is a technique that calculates the median of the values according to the corresponding outcome (diabetic or not).
For the implementation, we implemented two functions:

cal_median() and median_imputation():
cal_median(): the purpose of this function is to calculate the median of a specific variable (var) based on different groups defined by a target variable (target).
median_imputation() : it iterates over each row in the data frame. It checks specific conditions based on the target variable and the variable of interest (if the variable is equal to 0). If the conditions are met, it assigns specific imputed values to the variable of interest in the data frame. This process performs median imputation on the variable, filling in missing or designated values based on the target variable's values and specific conditions.
Here's an example of how to call the functions (figure 3.6):



```
a= cal_median(df,"Outcome","Insulin")

b=cal_median(df,"Outcome","Glucose")

c=cal_median(df,"Outcome","SkinThickness")

d=cal_median(df,"Outcome","BloodPressure")

e=cal_median(df,"Outcome","BMI")

median_imputation(df,"Outcome","Insulin",a.loc[0,"Insulin"],a.loc[1,"Insulin"])
median_imputation(df,"Outcome","Glucose",b.loc[0,"Glucose"],b.loc[1,"Glucose"])
median_imputation(df,"Outcome","SkinThickness",c.loc[0,"SkinThickness"],c.loc[1,"SkinThickness"])
median_imputation(df,"Outcome","BloodPressure",d.loc[0,"BloodPressure"],d.loc[1,"BloodPressure"])
median_imputation(df,"Outcome","BMI",e.loc[0,"BMI"],e.loc[1,"BMI"])
```

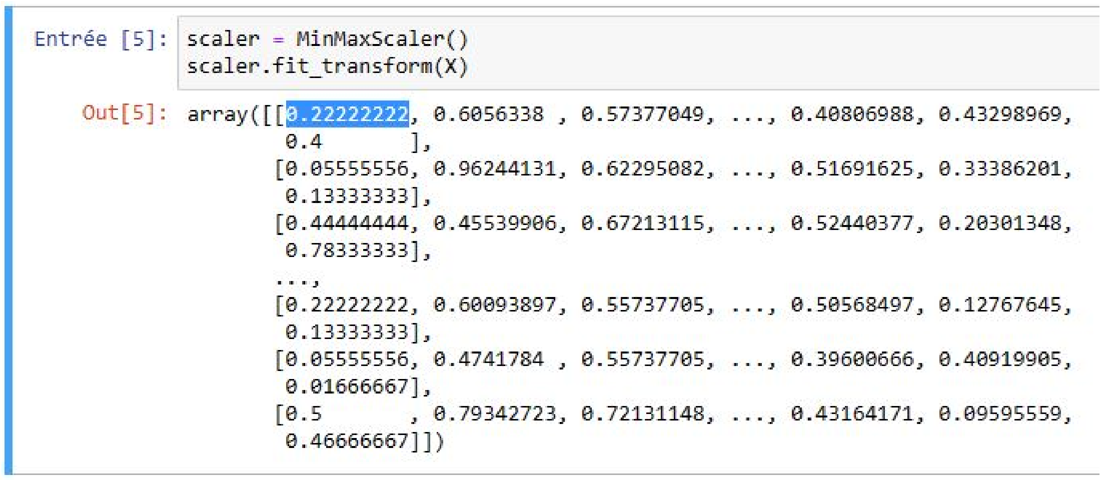
Figure 3.6: Calling functions.

3.5.2 Data Normalization

Normalization is a data transformation method that adjusts the values of numerical data to a specific range, typically between 0 and 1 or -1 and 1. This technique is particularly useful for mining algorithms such as classification, clustering, and artificial neural networks.
In the case of artificial neural networks, normalization can be used to scale the data attributes, which can speed up the learning process, especially for back-propagation neural network algorithms. Popular normalization techniques include min-max normalization, z-score normalization, decimal scaling, and other forms that adjust data values to a desired range [55].
We used mix-max normalization, we can express it by this equation:

$$x_{norm} = (x - min)/(max - min) \tag{3.1}$$

The code in figure 3.7 represents data normalization:



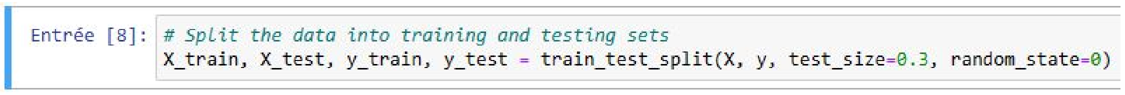
```
Entrée [5]: scaler = MinMaxScaler()
            scaler.fit_transform(X)

    Out[5]: array([[0.22222222, 0.6056338 , 0.57377049, ..., 0.40806988, 0.43298969,
                     0.4       ],
                    [0.05555556, 0.96244131, 0.62295082, ..., 0.51691625, 0.33386201,
                     0.13333333],
                    [0.44444444, 0.45539906, 0.67213115, ..., 0.52440377, 0.20301348,
                     0.78333333],
                    ...,
                    [0.22222222, 0.60093897, 0.55737705, ..., 0.50568497, 0.12767645,
                     0.13333333],
                    [0.05555556, 0.4741784 , 0.55737705, ..., 0.39600666, 0.40919905,
                     0.01666667],
                    [0.5       , 0.79342723, 0.72131148, ..., 0.43164171, 0.09595559,
                     0.46666667]])
```

Figure 3.7: Data normalization.

3.5.3 Data Splitting

We split the data set into training and testing sets using a 70%-30% ratio, as shown in figure 3.8. This ensures that we have a portion of the data reserved for evaluation purposes.



```
Entrée [8]: # Split the data into training and testing sets
            X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=0)
```

Figure 3.8: Data splitting.

3.6 Main Approach: "Stacking"

In this section, we present our main model for diabetes prediction, which is a stacking approach. The model combines two base classifiers: random forests and deep neural networks. The predictions from these base classifiers are then combined using a stacking ensemble approach with a final estimator of a support vector machine (figure 3.9).

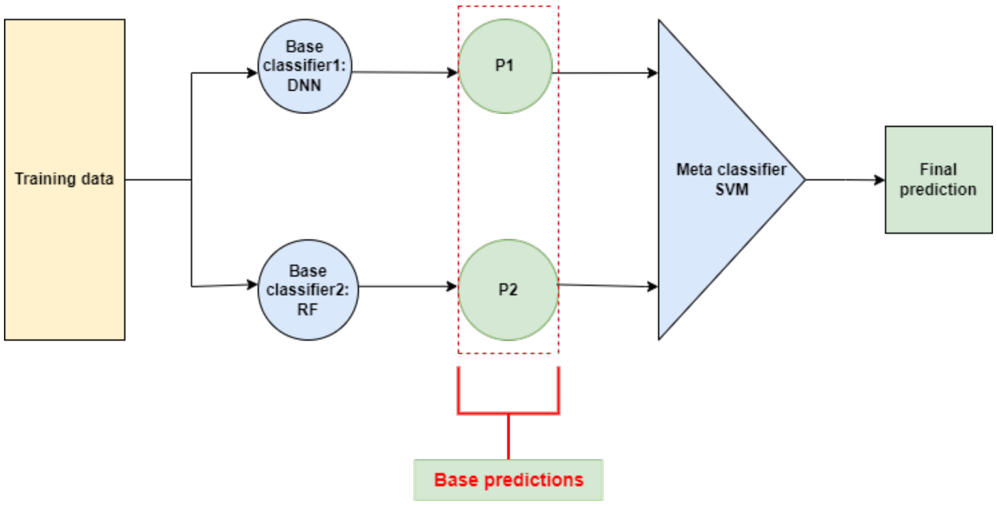
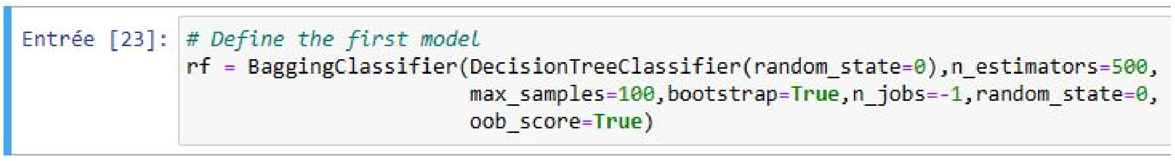


Figure 3.9: Stacking model architecture.

3.6.1 Base Classifier 1: Random Forest Classifier

The RF classifier is implemented using the `BaggingClassifier` class from the Sklearn ensemble module (figure 3.10). We used a decision tree classifier with a random state of 0. For the bagging, we set the number of estimators to 500, the maximum samples per estimator to 100, and enabled bootstrap sampling with replacement. We also set the n_jobs parameter to -1 to use all available CPUs and the random_state parameter to 0 for reproducibility. We also enable out-of-bag (OOB) scoring to estimate the generalization performance of the RF model.



```
Entrée [23]: # Define the first model
             rf = BaggingClassifier(DecisionTreeClassifier(random_state=0),n_estimators=500,
                                    max_samples=100,bootstrap=True,n_jobs=-1,random_state=0,
                                    oob_score=True)
```

Figure 3.10: RF implementation.

3.6.2 Base Classifier 2: Deep Neural Network

The DNN is implemented using the `MLPClassifier` from Sklearn (figure 3.12), which contains three layers:
Two hidden layers with 12 and 8 units, respectively, and a single output layer with a sigmoid activation function to predict the probability of having diabetes.
We used the ReLU activation function for the hidden layers and the binary cross-entropy loss function for training the model.
We compiled the model using the Adam optimizer with a learning rate of 0.001. We set the number of epochs to 100.

The sigmoid activation function can be defined as follows:

$$\sigma(x) = \frac{1}{1 + e^{-sum}} \tag{3.2}$$

- $\sigma(x)$ denotes the sigmoid function applied to the input x.
- e represents the mathematical constant Euler's number, approximately equal to 2.71828.
- $-sum$ signifies the negation of the value of "sum".
- $1 + e^{-sum}$ calculates the exponential of $-sum$ and adds 1 to it.
- $\frac{1}{1+e^{-sum}}$ computes the reciprocal of the quantity $1 + e^{-sum}$, resulting in a value between 0 and 1.

The sigmoid function maps any real-valued input x to a value between 0 and 1, this transforming the input into a probability like value.

ReLU (Rectified Linear Unit) activation function is defined by this equation:

$$f(x) = \max(0, x) \tag{3.3}$$

- $f(x)$ denotes the ReLU function applied to the input x.
- $\max(a, b)$ denotes the maximum value between a and b.
- 0 represents the lower threshold value.

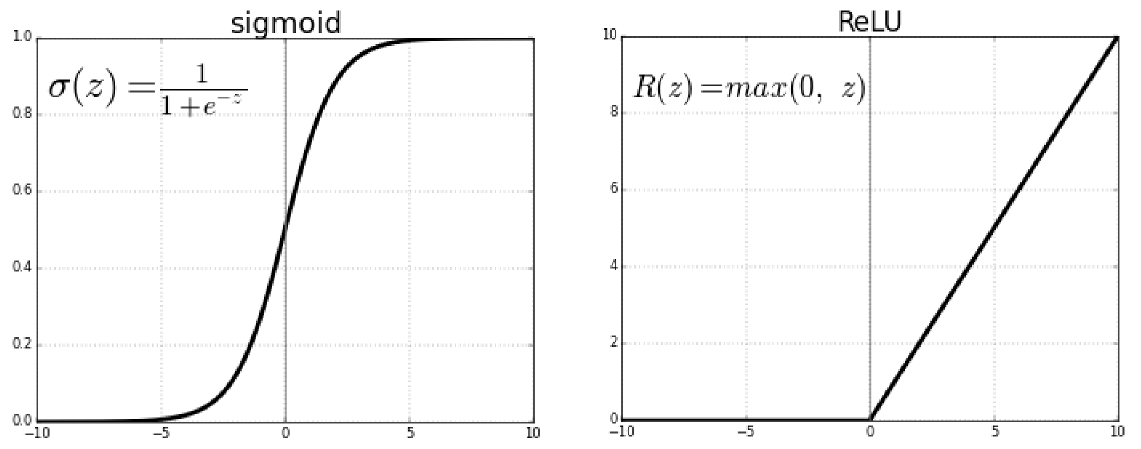


Figure 3.11: Sigmoid vs ReLU [W11].

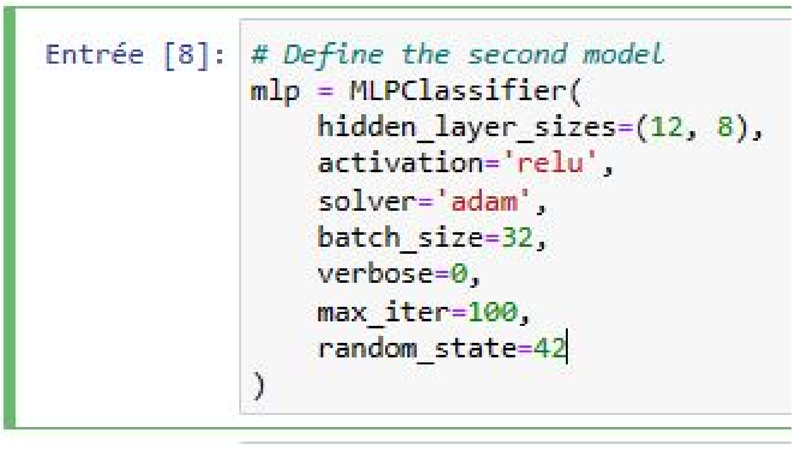
Loss function: The cross-entropy function quantifies the disparity between the predicted values and the actual values for each class. It calculates the individual errors for each class and then computes the average of these errors to determine the overall loss [56]. The binary cross-entropy is represented by the following equation:

$$L(y, \hat{y}) = -\frac{1}{N}\sum_{i=1}^{N}(y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i)) \tag{3.4}$$

- $L(y, \hat{y})$ denotes the binary cross-entropy loss between the true labels (y) and the predicted probabilities $(\hat{y})$.
- N represents the total number of samples or instances.
- $\sum_{i=1}^{N}$ signifies the summation over all samples, from $i = 1$ to $i = N$.
- y_i represents the true label (either 0 or 1) for the ith sample.
- $\hat{y}_i$ represents the predicted probability (between 0 and 1) for the ith sample.
- $\log(\hat{y}_i)$ computes the natural logarithm of the predicted probability $(\hat{y}_i)$.
- $(1 - y_i)$ evaluates to 1 if the true label y_i is 0, and 0 if y_i is 1.
- $\log(1 - \hat{y}_i)$ computes the natural logarithm of $1 - \hat{y}_i$.

The equation calculates the binary cross-entropy loss by summing the individual loss contributions for each sample and then normalizing it by dividing by the total number of samples (N). The loss penalizes larger differences between the true labels and predicted probabilities, aiming to minimize the discrepancy between them during model training.

The code below shows the implementation of our DNN model:



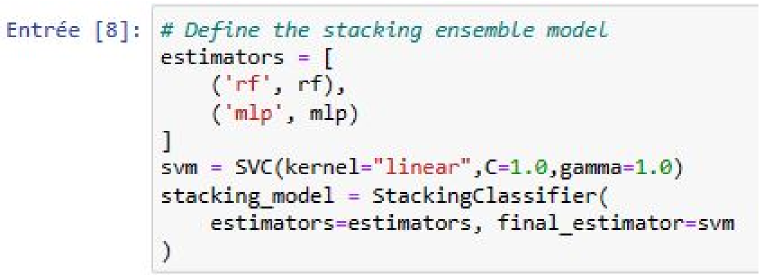
```
Entrée [8]: # Define the second model
            mlp = MLPClassifier(
                hidden_layer_sizes=(12, 8),
                activation='relu',
                solver='adam',
                batch_size=32,
                verbose=0,
                max_iter=100,
                random_state=42
            )
```

Figure 3.12: DNN implementation.

3.6.3 The Stacking Model With Meta Classifier: "SVC"

To combine the RF and DNN models, we used a stacking ensemble learning method, where the outputs of the RF and DNN models are used as input to a support vector classifier, which learns to combine the predictions of the two base models. (The code is shown in figure 3.13).

- The `estimators` list defines the individual classifiers to be stacked in the ensemble, there are two classifiers:
  - `'rf'`: random forest classifier.
  - `'mlp'`: Deep neural network.
- The `StackingClassifier` is then instantiated with the following parameters:
  - `estimators`: The list of individual classifiers specified in the stimators variable.
  - `final_estimator`: The meta-classifier that combines the predictions of the individual classifiers, which is the support vector classifier.



```
Entrée [8]: # Define the stacking ensemble model
estimators = [
    ('rf', rf),
    ('mlp', mlp)
]
svm = SVC(kernel="linear",C=1.0,gamma=1.0)
stacking_model = StackingClassifier(
    estimators=estimators, final_estimator=svm
)
```

Figure 3.13: Stacking model.

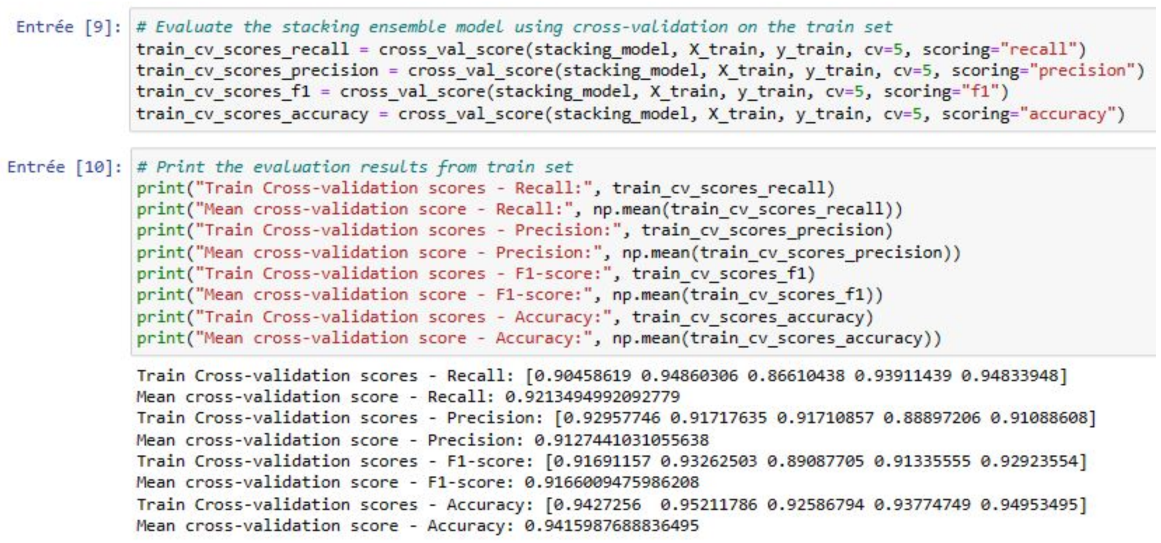
The `StackingClassifier` combines the predictions of the individual classifiers using the specified final_estimator. During training, the base classifiers (`RF` and `DNN`) are fitted on the training data, and their predictions are then used as inputs for the final_estimator. The final_estimator is trained to make the final predictions based on the combined outputs of the base classifiers.

3.7 Train and Evaluation

3.7.1 Applying CV on train set

To evaluate the performance of the stacking ensemble model, 5-fold cross-validation was applied on the training set. In this study, the stacking ensemble model (`stacking_model`) was evaluated using four key evaluation metrics: recall, precision, F1-score, and accuracy. The `cross_val_score` function from the scikit-learn library was utilized to calculate the cross-validated scores for each metric.

For each metric, the model was trained and evaluated on each fold independently. The resulting individual cross-validated scores were stored in the variables `train_cv_scores_recall`, `train_cv_scores_precision`, `train_cv_scores_f1`, and `train_cv_scores_accuracy`. These scores represent the performance of the stacking ensemble model on each fold of the training set.
To provide an overall measure of the model's performance, the mean cross-validated scores were also calculated. The mean scores were obtained by averaging the individual scores across all folds. The mean cross-validated scores provide a summarized measure of the stacking ensemble model's performance across the entire training set.
The evaluation results, including both the individual cross-validated scores and the mean cross-validated scores, were printed to assess the stacking ensemble model's performance.
By employing 5-fold cross-validation and reporting the evaluation results, a rigorous evaluation of the stacking ensemble model's performance is achieved. This evaluation provides valuable insights into the model's effectiveness in classifying instances and allows for meaningful comparisons with other models of diabetes prediction
Figure 3.14 represents how we get the results:



```
Entrée [9]: # Evaluate the stacking ensemble model using cross-validation on the train set
train_cv_scores_recall = cross_val_score(stacking_model, X_train, y_train, cv=5, scoring="recall")
train_cv_scores_precision = cross_val_score(stacking_model, X_train, y_train, cv=5, scoring="precision")
train_cv_scores_f1 = cross_val_score(stacking_model, X_train, y_train, cv=5, scoring="f1")
train_cv_scores_accuracy = cross_val_score(stacking_model, X_train, y_train, cv=5, scoring="accuracy")
```

```
Entrée [10]: # Print the evaluation results from train set
print("Train Cross-validation scores - Recall:", train_cv_scores_recall)
print("Mean cross-validation score - Recall:", np.mean(train_cv_scores_recall))
print("Train Cross-validation scores - Precision:", train_cv_scores_precision)
print("Mean cross-validation score - Precision:", np.mean(train_cv_scores_precision))
print("Train Cross-validation scores - F1-score:", train_cv_scores_f1)
print("Mean cross-validation score - F1-score:", np.mean(train_cv_scores_f1))
print("Train Cross-validation scores - Accuracy:", train_cv_scores_accuracy)
print("Mean cross-validation score - Accuracy:", np.mean(train_cv_scores_accuracy))
```

```
Train Cross-validation scores - Recall: [0.90458619 0.94860306 0.86610438 0.93911439 0.94833948]
Mean cross-validation score - Recall: 0.9213494992092779
Train Cross-validation scores - Precision: [0.92957746 0.91717635 0.91710857 0.88897206 0.91088608]
Mean cross-validation score - Precision: 0.9127441031055638
Train Cross-validation scores - F1-score: [0.91691157 0.93262503 0.89087705 0.91335555 0.92923554]
Mean cross-validation score - F1-score: 0.9166009475986208
Train Cross-validation scores - Accuracy: [0.9427256  0.95211786 0.92586794 0.93774749 0.94953495]
Mean cross-validation score - Accuracy: 0.9415987688836495
```

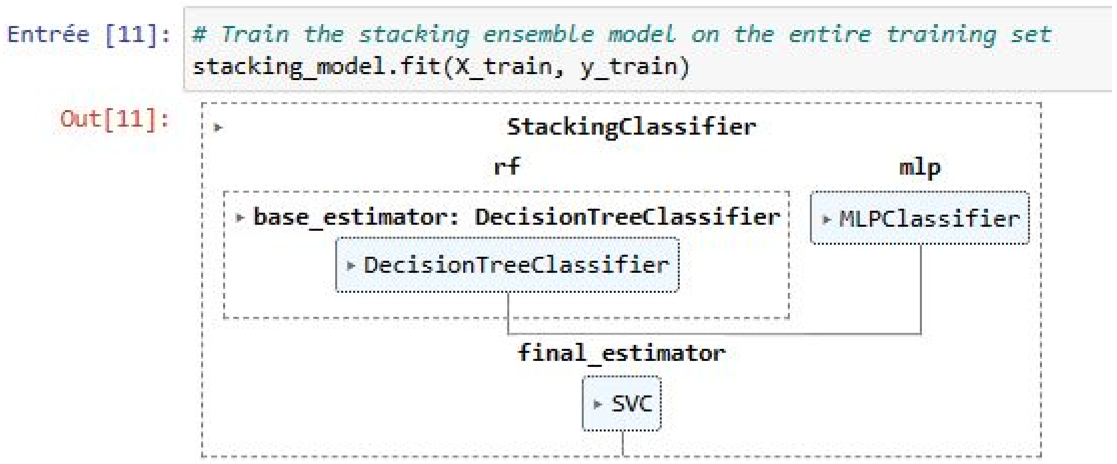
Figure 3.14: Train set results.

3.7.2 Model Fitting

After applying cross-validation to the train set, we should train the stacking ensemble model on the entire training set.
the code `stacking_model.fit(X_train, y_train)` is used. This code fits the stacking model (`stacking_model`) to the training data, where `X_train` represents the input features and `y_train` represents the corresponding target labels.

By calling the `fit()` method on the `stacking_model`, the individual base classifiers (such as random forest and neural network) are trained on the training data. Their predictions are then combined by the `final_estimator` (support vector classifier) to make the final predictions of the stacking ensemble model.
After training, the stacking ensemble model is ready for the final evaluation or making predictions on new, unseen data.

The code in figure 3.15 shows how to fit the stacking model on the entire train set:



```
Entrée [11]: # Train the stacking ensemble model on the entire training set
             stacking_model.fit(X_train, y_train)
Out[11]:
```

Figure 3.15: Model fitting.

3.7.3 Evaluation on test set

The performance of the stacking ensemble model was further assessed on the test set to evaluate its generalization ability and robustness.

Confusion Matrix

After predicting the `x_test`, we will obtain the predicted class labels (`y_pred`). Subsequently, the confusion matrix was computed using the predicted labels (figure 3.16). The confusion matrix provides a tabular representation that summarizes the performance of a classification model by showing the counts of true positive, true negative, false positive, and false negative predictions. It serves as a valuable tool for analyzing the model's performance and assessing the accuracy of its predictions.

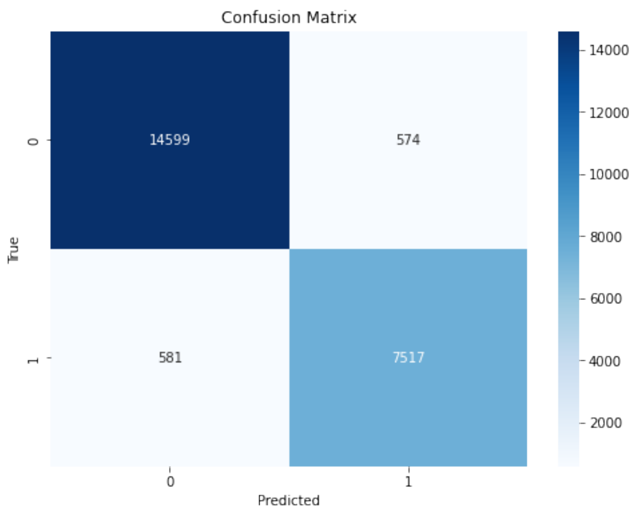


Figure 3.16: Confusion matrix.

As mentioned in the previous chapter, the performance evaluation of a classification model can be conducted using various metrics, which offer insights into different facets of the model's predictive capabilities.
One widely employed metric is **"accuracy"**, which quantifies the ratio of correctly classified instances to the total number of instances. It can be computed by utilizing the `accuracy_score` function. Higher accuracy values are indicative of superior overall performance.
"Precision" constitutes another significant metric, determining the ratio of true positive predictions to all positive predictions. It specifically focuses on the accuracy of positive predictions and can be derived through employment of the `precision_score` function.
"Recall", also known as sensitivity or true positive rate, ascertains the ratio of true positive predictions to all actual positive instances. It accentuates the model's capacity to identify positive instances and can be calculated utilizing the `recall_score` function.
"F1 score" represents a metric that consolidates precision and recall into a singular value. This score furnishes a balanced measurement of the model's performance and can be computed using the `f1_score` function. The F1 score takes into account both false positives and false negatives and is particularly valuable in scenarios characterized by imbalanced class distributions. Table 3.1 represents the obtained results:

Accuracy	Precision	Recall	F1-score
95%	92.9%	92.8%	92.8%

Table 3.1: Obtained results on test set.

The obtained results can be expressed using the bar plot in figure 3.17:

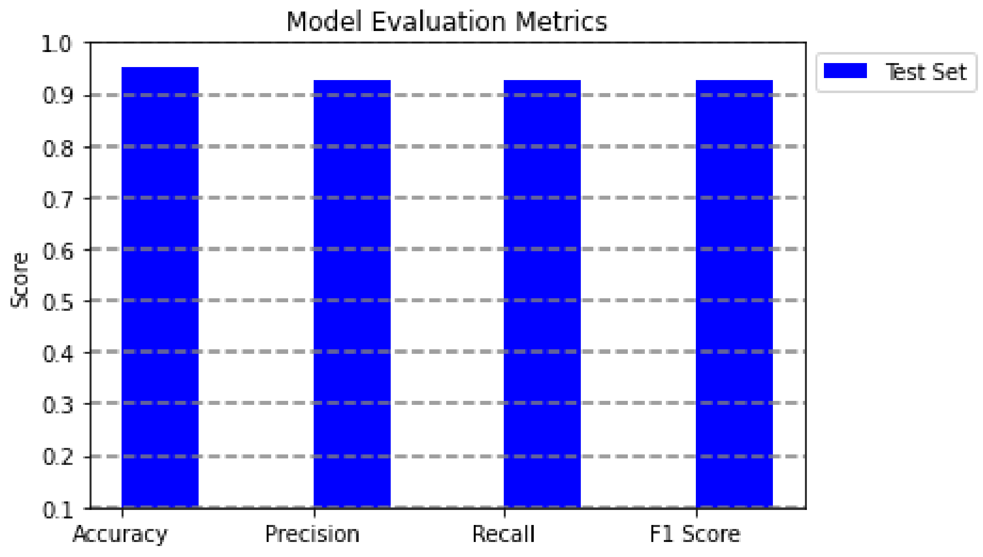
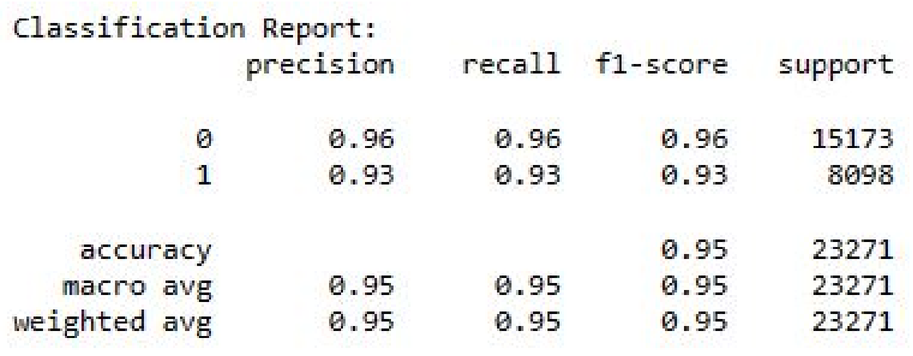


Figure 3.17: Bar plot.

Classification Report



```
Classification Report:
              precision    recall  f1-score   support

           0       0.96      0.96      0.96     15173
           1       0.93      0.93      0.93      8098

    accuracy                           0.95     23271
   macro avg       0.95      0.95      0.95     23271
weighted avg       0.95      0.95      0.95     23271
```

Figure 3.18: Classification report.

Precision Recall Curve

Based on the precision-recall curve shown in figure 3.19, it can be observed that the curve closely aligns with the optimal point, indicating an exceptional performance of our model.

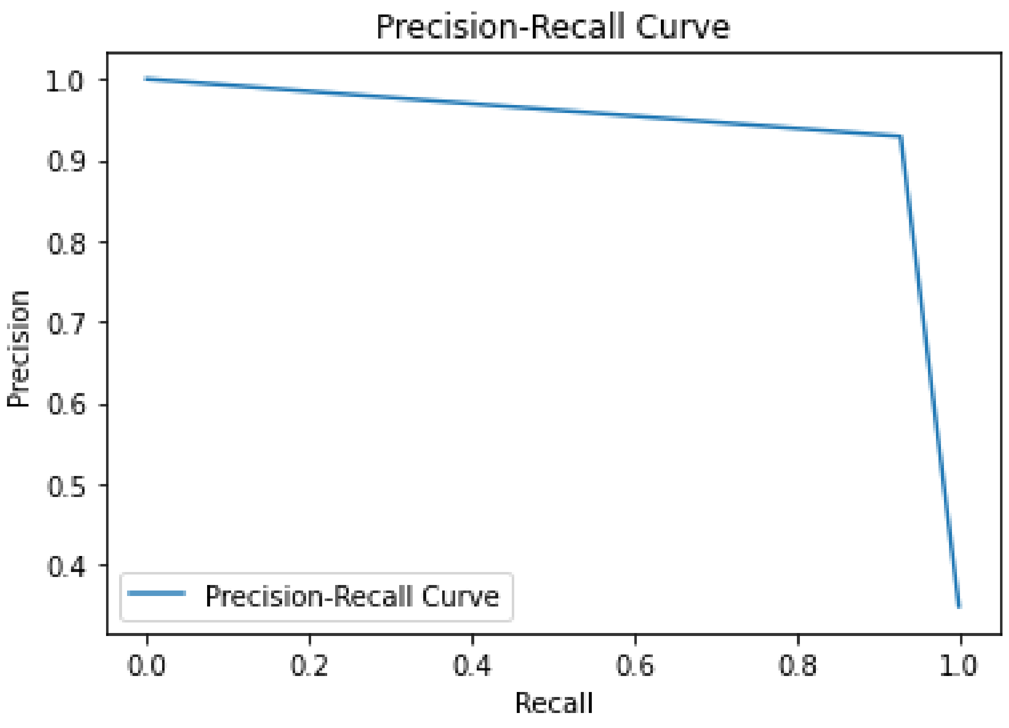


Figure 3.19: Precision Recall curve.

3.7.4 Evaluating other Meta-Clasifiers

During our extensive research, we employed various techniques and methods to obtain the best possible results, ultimately leading us to develop the stacking model as our final approach. Initially, we experimented with different meta classifiers, including KNN and logistic regression, to enhance the performance of our stacking model (table 3.2). Furthermore, we conducted analyses to evaluate the effects of data imputation on the results (table 3.3).

The selection of the meta classifier was deliberated between SVM and logistic regression, given the specific characteristics of the Pima dataset, including class imbalance. Recognizing that accuracy alone may not sufficiently capture the performance of the model, we considered additional important metrics, such as precision, recall (table 3.2). After careful evaluation, we concluded that SVM demonstrated superior performance in terms of these metrics, leading us to choose it as the preferred meta-classifier.

Meta-classifier	**Accuracy (%)**	**Precision (%)**	**Recall (%)**	**F1 Score (%)**
SVM	95.0	92.9	92.8	92.8
Logistic Regression	95.0	93.0	92.5	92.8
KNN (k=10)	94.7	92.3	92.7	92.5

Table 3.2: Comparison of meta classifiers.

Based on the presented results in table 3.3, we can see that the approach with imputation generally performed better than the approach without imputation across all metrics. Imputation, involving filling in missing values, seems to have positively impacted the performance of our proposed model.

Our approach	**Accuracy (%)**	**Precision (%)**	**Recall (%)**	**F1 Score (%)**
With Imputation	95.0	92.9	92.8	92.8
Without Imputation	86.5	81.9	78.6	80.0

Table 3.3: Results of our approach with and without imputation.

3.8 Alternative Approach: "Voting"

As an additional approach, we have devised an alternative approach within the domain of ensemble learning, akin to stacking. This technique utilizes a voting mechanism reliant on the fusion of two algorithms, namely random forest and XGBoost. The accompanying flowchart in figure 3.20 serves to visually represent our secondary approach.

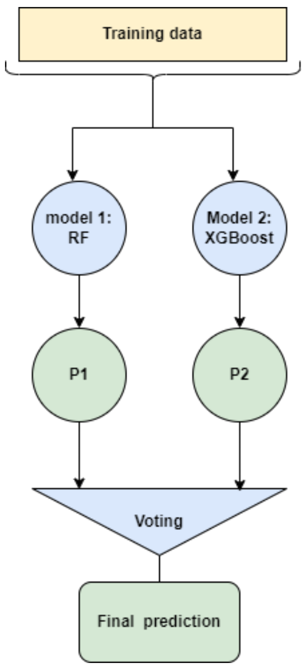


Figure 3.20: Voting model architecture.

The first constituent of this combination is the random forest model, which is identical to the one employed in our primary approach "stacking". The second component is XG-Boost, an ensemble learning algorithm belonging to the boosting family.

Here's a simplified explanation of how XGBoost works [W12]:
Base Learners: Decision trees, specifically a particular variety of decision tree known as a "CART" (Classification and Regression Tree), serve as the foundational learners in XGBoost. Because these decision trees are shallow, or have a modest depth, overfitting is less likely to occur.
Objective Function: An objective function that needs to be optimized during training is defined by XGBoost. The objective function is made up of two parts: a regularization term that regulates the model's complexity and a loss function that calculates the discrepancy between predicted and actual values.
for binary classification, the objective function used is the logistic loss function, also known as the log loss or cross-entropy loss.
The objective function for binary classification in XGBoost can be written as follows:

$$Objective = \sum [\log(1 + \exp(-y_i \cdot F(x_i))) + \lambda \cdot \Omega(F)] \tag{3.5}$$

- y_i is the true label of the i-th instance (either 0 or 1).
- $F(x_i)$ represents the predicted probability for the i-th instance.
- $\Omega(F)$ is a regularization term that controls the complexity of the model, often incorporating L1 or L2 regularization.
- λ is the regularization parameter that balances the impact of the regularization term.

Gradient Calculation: The algorithm for XGBoost calculates the gradients of the loss function with respect to the predicted values in each iteration. The degree to which each instance was "missed" by the present ensemble of models is indicated by the gradient information.
Tree Construction: In order to identify the patterns in the gradient information, XGBoost constructs decision trees. In greedy tree construction, splits that minimize the objective function are chosen. To quickly identify the ideal splits, XGBoost employs a method known as "approximate algorithmic optimization."
Weighted Updates: During training, XGBoost gives the instances weights. Based on the mistakes produced by the ensemble of models up to that time, the weights are assigned. Poorly forecasted instances are given more weight, which increases their influence on succeeding iterations.
Ensemble Building: Each tree is built, then it is included in the ensemble. By adding up all of the individual forecasts, weighted by importance, the ensemble of trees produces predictions collectively.
Regularization and Control Parameters: Regularization techniques are used by XGBoost to limit model complexity and avoid overfitting. To achieve the required level of

regularization, regularization parameters like gamma, L1 and L2 regularization can be changed.

Prediction: Following training, XGBoost can be used to make predictions on fresh, unexplored data. The final forecast is generated by adding up all of the ensemble's guesses, weighted by importance.

We implemented the model using the parameters outlined in table 3.4.

Parameter	**Value**
learning_rate	0.01
n_estimators	500
max_depth	5
min_child_weight	2
gamma	0.4
subsample	0.8
colsample_bytree	0.8
objective	binary:logistic

Table 3.4: XGBoost Parameters.

After obtaining the two fundamental predictions generated by the Random Forest and XGBoost algorithms, we merge them through a voting classifier employing a hard voting strategy.

Given N individual models, denoted as $f_1, f_2, \ldots, f_N$, the ultimate prediction for a given input x can be mathematically expressed using equation 3.6, where $argmax$ selects the class label with the highest accumulated votes:

$$y_{pred} = \arg\max \left(\sum_{i=1}^{N} f_i(x) \right) \tag{3.6}$$

3.8.1 Evaluation Results

After training the model on the training set, we evaluated its performance on the test set, obtaining the following results (table 3.5):

Approach	**Accuracy (%)**	**Precision (%)**	**Recall (%)**	**F1 Score (%)**
Voting (RF+XGBoost)	92.7	97.1	81.7	88.7

Table 3.5: Voting technique results.

3.8.2 Comparaison between Stacking and Voting

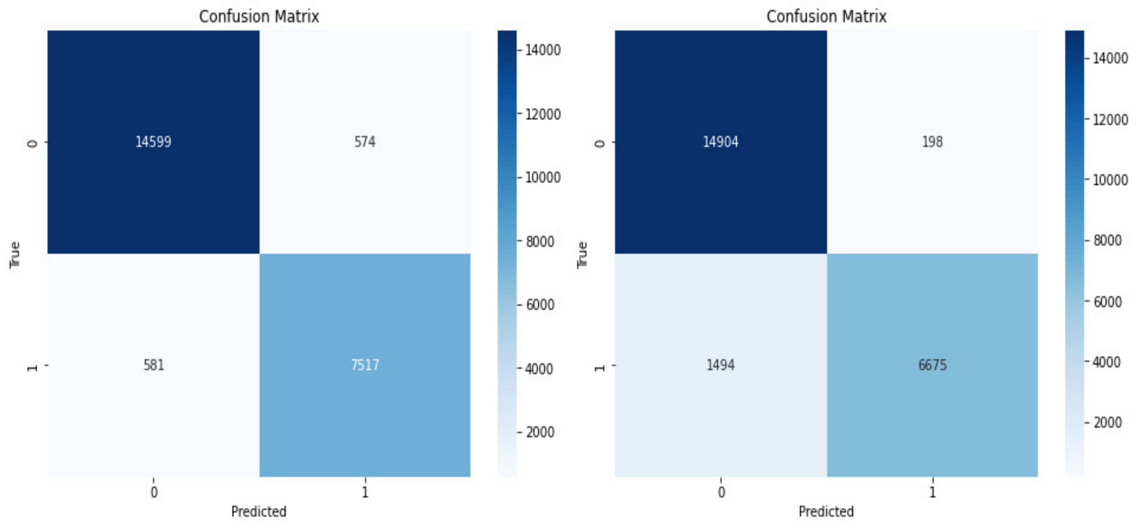


Figure 3.21: Stacking vs Voting Confusion matrix.

The primary rationale for selecting stacking over voting as our primary approach stems from the significance of minimizing false negatives. Given the context of our diabetes prediction problem, misclassifying an individual as non-diabetic when they are, in fact, diabetic can have severe consequences. Failure to identify diabetes in a person may lead to a lack of preventive measures, thereby exacerbating the progression and complications of the illness. Consequently, our objective is to minimize the occurrence of false negatives, as a lower incidence of such errors reflects the superior performance of our model.

3.9 Discussion

Our proposed approaches employ ensemble learning techniques, specifically stacking and voting, are novel techniques to address the task of diabetes prediction is a novel ensemble learning technique.
Ensemble learning offers several advantages in the context of our research, providing improved prediction accuracy and mitigating overfitting issues.

Firstly, stacking combines the strengths of different classifiers, namely Deep Neural Networks (DNNs), Random Forests (RF), and Support Vector Machines (SVM). By leveraging the complementary characteristics of these classifiers, we can achieve higher accuracy compared to using them individually. Stacking creates a more robust and generalized model that captures diverse patterns, thus addressing the challenge of overfitting in complex datasets.

Moreover, our approach benefits from the enhanced feature representation capabilities of DNNs, allowing for the capture of complex relationships and patterns in the data. The inclusion of RF further enhances prediction performance by handling categorical variables and capturing non-linear relationships. SVM, with its unique decision boundaries, contributes to the overall predictive power of the ensemble.

In addition to stacking, we also explored the voting approach using Random Forest and XGBoost classifiers. Voting combines the predictions of these models through a voting scheme, aggregating their outputs to make the final prediction. This approach takes advantage of the strengths of both Random Forest and XGBoost, which excel in handling different types of data and capturing diverse patterns.

The Random Forest classifier, with its interpretability and feature importance measures, aids in identifying relevant factors that contribute to predicting diabetes, providing valuable insights for medical practitioners and researchers. XGBoost, on the other hand, offers superior gradient boosting capabilities and efficient handling of large datasets, further improving the ensemble's predictive performance.

Furthermore, ensemble models, whether through stacking or voting, exhibit flexibility and adaptability. They allow for easy incorporation of new classifiers or modifications, enabling the model to adapt to evolving data and incorporate the latest advancements in classifier techniques.

However, it is important to note that stacking can be computationally complex, requiring significant computational resources and time, especially when dealing with large datasets. This limitation should be taken into consideration when applying ensemble learning techniques.

Despite the limitations, our novel approaches holds promise for improving prediction performance and interpretability in the domain of diabetes prediction.

Table 3.6 shows a comparision of accuracy achieved by other state-of-the-art works:

Technique	**Author and year**	**Accuracy**
Stacking (DNN + RF)	**1st contribution**	**95%**
Voting (RF +XGBoost)	**2nd contribution**	**92.7%**
Stacking technique	[Khilwani et al., 2021] [32]	82.68%
Soft voting classifier	[Kumari et al., 2021] [33]	79.08%
XGBoost	[Kumari et al., 2021] [33]	75.75%
Bagging	[Kumari et al., 2021] [33]	74.89%
Random Forest	[Kumari et al., 2021] [33]	77.48%
Support Vector	[Kumari et al., 2021] [33]	74.02%
XGBoost+ Data feature stitching	[Li et al., 2020] [34]	80.2%
The Hoeffding Tree algorithm	Mercaldo et al., 2017] [35]	75.7%
Naive Bayes	[Larabi-Marie-Sainte et al., 2019] [36]	76.30%
FS and DR with Random forest	[Sivaranjani et al., 2021] [37]	83%
HM-Bag Moov Voting Classifier	[Bashir et al., 2016] [38]	77.21%
Voting Classifier	[Mahabub, 2019] [39]	86%
KNN	[Febrian et al., 2023] [40]	73.33%
Naive Bayes	[Febrian et al., 2023] [40]	76.07%
DL	[Yahyaoui et al., 2019] [41]	76.81%
ANN	[El_Jerjawi et Abu-Naser, 2018] [42]	87.3%
Autoencoders + Softmax classifier	[Kannadasan et al., 2019] [43]	86%

Table 3.6: Results comparison table for machine and deep learning approaches.

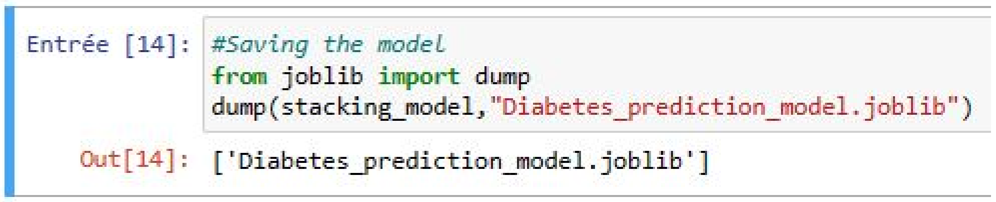
By comparing our results with these related works, it is evident that our ensemble learning approaches achieved higher accuracy (92.7 for Voting approach and 95 for Stacking approach) compared to individual algorithms and other ensemble methods.

In conclusion, our ensemble learning approach demonstrates its superiority in diabetes prediction compared to many other state of the art techniques.

3.10 Model Deployment

3.10.1 Model Saving

We saved our model to a file named "diabetes_prediction_model.joblib" This allows for easy retrieval and reuse of the model in future tasks or deployments. The "joblib" format is commonly used for saving Scikit-Learn models.
The code is demonstrated in figure 3.22.



```
Entrée [14]: #Saving the model
             from joblib import dump
             dump(stacking_model,"Diabetes_prediction_model.joblib")

    Out[14]: ['Diabetes_prediction_model.joblib']
```

Figure 3.22: Model saving.

3.10.2 User Interface

To facilitate user interaction and enable the input of new data for predicting the presence of diabetes, we implemented a user interface using the tkinter library. This graphical user interface (GUI) provides a platform for users to enter relevant information, such as medical attributes and health indicators, which is then utilized by the underlying model to make predictions.
The user interface consists of input labels and entry boxes for each feature, such as "pregnancies," "glucose," "blood pressure," and so on. Users can input their values in the corresponding entry boxes. The interface also includes a "Predict" button, which triggers the prediction process based on the user's input as shown in figure 3.23.

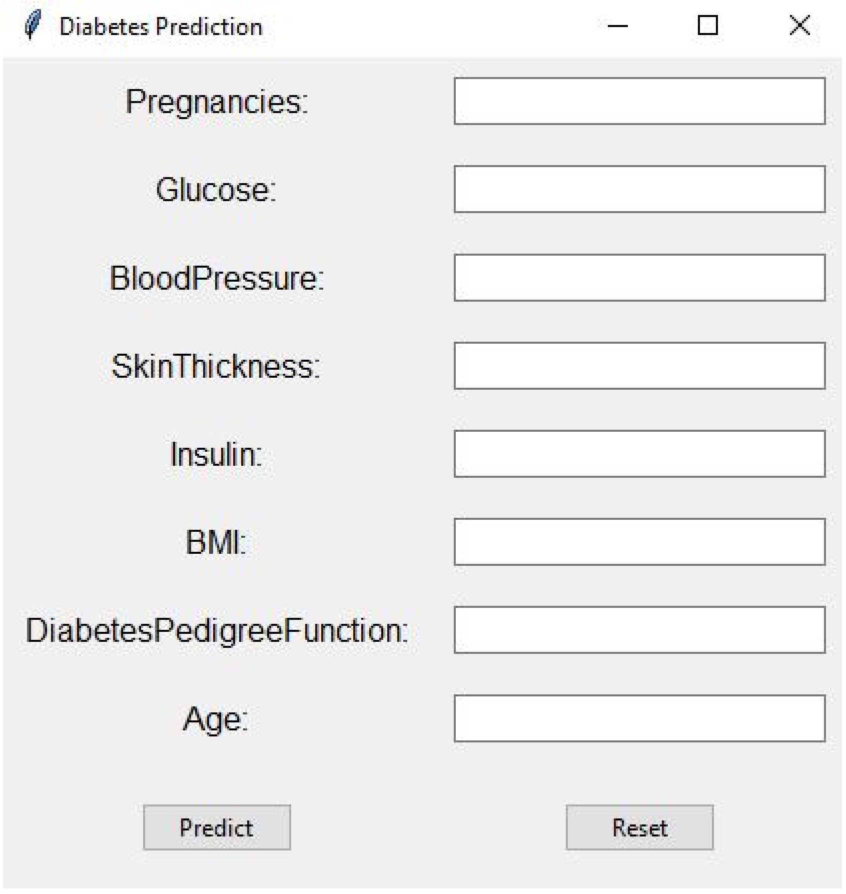
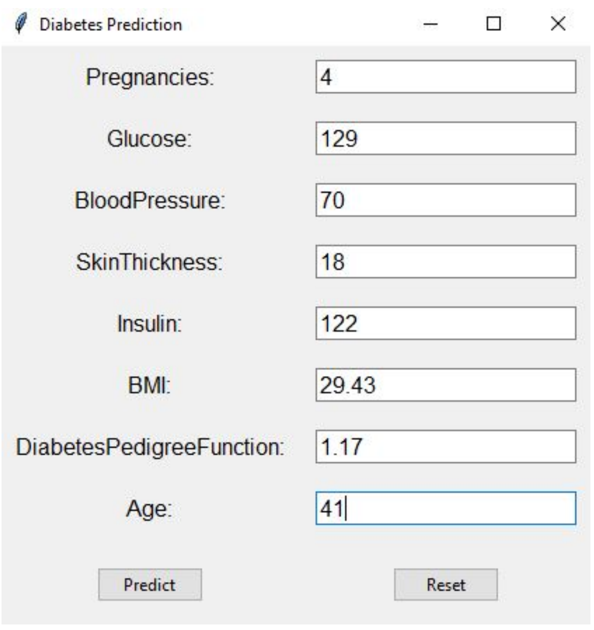


Figure 3.23: User interface.

Here is an example of predicting diabetes for a diabetic individual (figure 3.24):

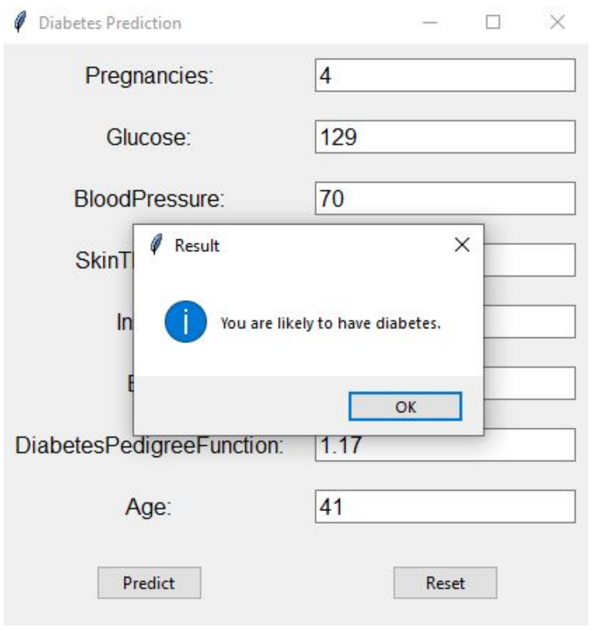


Diabetes Prediction

Pregnancies:	4
Glucose:	129
BloodPressure:	70
SkinThickness:	18
Insulin:	122
BMI:	29.43
DiabetesPedigreeFunction:	1.17
Age:	41

Figure 3.24: Example of diabetic person.

Result:



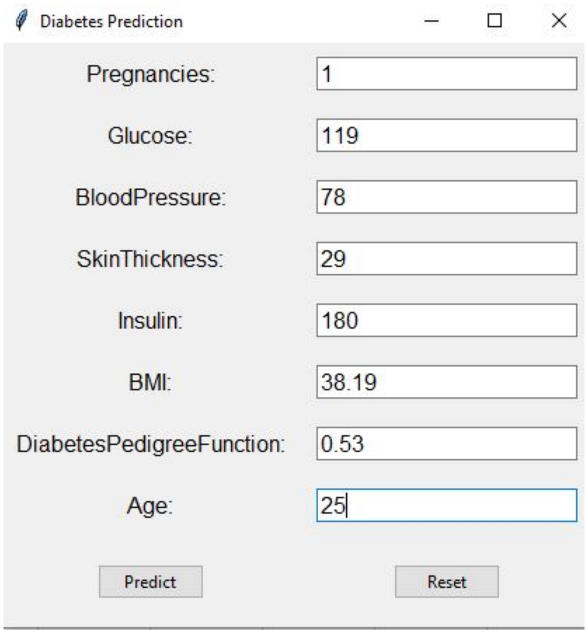
Diabetes Prediction

Result

You are likely to have diabetes.

Figure 3.25: Result of the prediction.

Another example of predicting diabetes for a non-diabetic individual (figure 3.26):

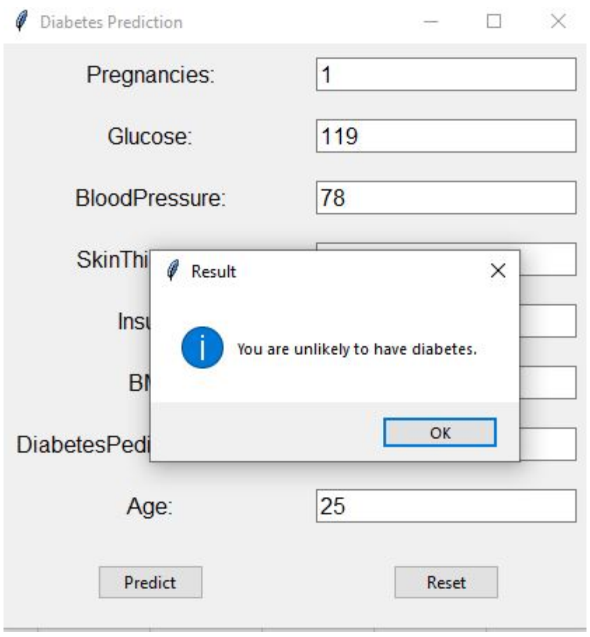


Diabetes Prediction

Pregnancies:	1
Glucose:	119
BloodPressure:	78
SkinThickness:	29
Insulin:	180
BMI:	38.19
DiabetesPedigreeFunction:	0.53
Age:	25

Figure 3.26: Example of non-diabetic person.

Result:



Diabetes Prediction

Pregnancies:	1
Glucose:	119
BloodPressure:	78
Age:	25

Result

You are unlikely to have diabetes.

Figure 3.27: Result of the prediction.

3.11 Conclusion

In conclusion, this chapter has provided an overview of the methodology and implementation for predicting diabetes using ensemble learning. We have outlined the steps followed to collect and preprocess the dataset, ensuring its suitability for training and evaluation. Additionally, we have presented the details of our proposed ensemble learning models and the rationale behind our approachs.
Also, we have shared the code used to build and train the stacking model, as well as the libraries and software employed in our implementation. By combining the methodology and implementation sections into a single chapter, we aimed to provide a comprehensive understanding of our contribution.

General conclusion

In this thesis, we employed ensemble learning techniques to enhance the accuracy of diabetes prediction. Specifically, we applied the stacking approach, combining a Deep Neural Network (DNN) with Random Forest (RF) as the base classifiers and SVM as the meta-classifier, as well as the voting technique using XGBoost and Random Forest.

The stacking approach achieved an accuracy of 95%, precision of 92.9%, recall of 92.8%, and an F1 score of 92.8%. Similarly, the voting approach, combining XGBoost and Random Forest, yielded an accuracy of 92.7%, precision of 97.1%, recall of 81.7%, and an F1 score of 88.7%.

While these results demonstrate the effectiveness of the ensemble learning models in improving the accuracy of diabetes prediction, there are still exciting avenues for further research and improvement.

One perspective to consider is exploring different combinations of algorithms within ensemble learning frameworks. By testing various combinations of base classifiers and meta-classifiers, such as Decision Trees, Gradient Boosting Machines, or Neural Networks, we can potentially discover more powerful models that offer higher accuracy and improved predictive performance for diabetes prediction. Expanding the scope of the research to include diverse datasets is also crucial. Working with datasets from different sources and populations allows us to evaluate the generalizability and robustness of the ensemble learning models. It enables us to assess how well the models perform across various demographic, ethnic, and geographical groups, ensuring their effectiveness in real-world scenarios. By conducting extensive experiments with different algorithm combinations and diverse datasets, we can gain a deeper understanding of the strengths and limitations of ensemble learning techniques in diabetes prediction. This knowledge will empower us to refine and optimize the models, pushing the boundaries of accuracy and contributing to the development of more advanced and reliable prediction systems in healthcare.

In conclusion, while the stacking and voting ensemble learning models have shown promising results, there is still room for further research to improve the accuracy of diabetes prediction. Exploring different algorithm combinations and working with diverse datasets will open up new possibilities and help us develop more accurate and reliable prediction models in the field of healthcare.

Bibliography

[1] Issam El Naqa and Martin J Murphy. *What is machine learning?* Springer, 2015.

[2] Jinming Zou, Yi Han, and Sung-Sau So. Overview of artificial neural networks. *Artificial neural networks: methods and applications*, pages 14–22, 2009.

[3] Kuldeep Shiruru. An introduction to artificial neural network. *International Journal of Advance Research and Innovative Ideas in Education*, 1:27–30, 09 2016.

[4] NJ Sairamya, L Susmitha, S Thomas George, and MSP Subathra. Hybrid approach for classification of electroencephalographic signals using time–frequency images with wavelets and texture features. In *Intelligent Data Analysis for Biomedical Applications*, pages 253–273. Elsevier, 2019.

[5] Ledisi G Kabari and Ugochukwu C Onwuka. Comparison of bagging and voting ensemble machine learning algorithm as a classifier. *International Journals of Advanced Research in Computer Science and Software Engineering*, 9(3):19–23, 2019.

[6] World Health Organization. *Global report on diabetes.* World Health Organization, 2016.

[7] Leanne Bellamy, Juan-Pablo Casas, Aroon D Hingorani, and David Williams. Type 2 diabetes mellitus after gestational diabetes: a systematic review and meta-analysis. *The lancet*, 373(9677):1773–1779, 2009.

[8] World Health Organization et al. Diagnostic criteria and classification of hyperglycaemia first detected in pregnancy. Technical report, World Health Organization, 2013.

[9] Gudisa Bereda. Brief overview of diabete mellitus. *Diabetes Manag S*, 1:21–27, 2021.

[10] World Health Organization et al. Definition and diagnosis of diabetes mellitus and intermediate hyperglycaemia: report of a who/idf consultation. 2006.

[11] World Health Organization et al. Report of a world health organization consultation. use of glycated haemoglobin (hba1c) in the diagnosis of diabetes mellitus. *Diabetes Res Clin Pract*, 93(2), 2011.

[12] Trevor Hastie, Robert Tibshirani, Jerome H Friedman, and Jerome H Friedman. *The elements of statistical learning: data mining, inference, and prediction*, volume 2. Springer, 2009.

[13] Yalin Baştanlar and Mustafa Özuysal. Introduction to machine learning. *miRNomics: MicroRNA biology and computational analysis*, pages 105–128, 2014.

[14] Ivan Vasilev, Daniel Slater, Gianmario Spacagna, Peter Roelants, and Valentino Zocca. *Python Deep Learning: Exploring deep learning techniques and neural network architectures with Pytorch, Keras, and TensorFlow*. Packt Publishing Ltd, 2019.

[15] Yingqiu Liu, Wei Li, and Yunchun Li. Network traffic classification using k-means clustering. In *Second international multi-symposiums on computer and computational sciences (IMSCCS 2007)*, pages 360–365. IEEE, 2007.

[16] Gopi Battineni, Nalini Chintalapudi, and Francesco Amenta. Machine learning in medicine: Performance calculation of dementia prediction by support vector machines (svm). *Informatics in Medicine Unlocked*, 16:100200, 2019.

[17] Khushbu Kumari, Suniti Yadav, et al. Linear regression analysis study. *Journal of the practice of Cardiovascular Sciences*, 4(1):33, 2018.

[18] David W Hosmer Jr, Stanley Lemeshow, and Rodney X Sturdivant. *Applied logistic regression*, volume 398. John Wiley & Sons, 2013.

[19] Harsh Kukreja, N Bharath, CS Siddesh, and S Kuldeep. An introduction to artificial neural network. *Int J Adv Res Innov Ideas Educ*, 1:27–30, 2016.

[20] Emil M Petriu. Professor, university of ottawa,". *Neural Networks Basics*, 2017.

[21] Carlos Gershenson. Artificial neural networks for beginners. *arXiv preprint cs/0308031*, 2003.

[22] Meera Shukla and M Abdelrahman. Artificial neural networks based steady state security analysis of power systems. In *Thirty-Sixth Southeastern Symposium on System Theory, 2004. Proceedings of the*, pages 266–269. IEEE, 2004.

[23] GS Anitha and S Kuldeep. Neural network approach for processing substation alarms. *International Journal of Power Electronics Controllers and Converters*, 1(1):21–28, 2015.

[24] Tara N. Sainath, Oriol Vinyals, Andrew Senior, and Haşim Sak. Convolutional, long short-term memory, fully connected deep neural networks. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4580–4584, 2015.

[25] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

[26] Zachary C Lipton, John Berkowitz, and Charles Elkan. A critical review of recurrent neural networks for sequence learning. *arXiv preprint arXiv:1506.00019*, 2015.

[27] Md Zahangir Alom, Tarek M Taha, Chris Yakopcic, Stefan Westberg, Paheding Sidike, Mst Shamima Nasrin, Mahmudul Hasan, Brian C Van Essen, Abdul AS Awwal, and Vijayan K Asari. A state-of-the-art survey on deep learning theory and architectures. *electronics*, 8(3):292, 2019.

[28] Tagel Aboneh, Abebe Rorissa, and Ramasamy Srinivasagan. Stacking-based ensemble learning method for multi-spectral image classification. *Technologies*, 10(1):17, 2022.

[29] Jerome H Friedman. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232, 2001.

[30] Leo Breiman. Heuristics of instability and stabilization in model selection. *The annals of statistics*, 24(6):2350–2383, 1996.

[31] Tzu-Tsung Wong and Po-Yang Yeh. Reliable accuracy estimates from k-fold cross validation. *IEEE Transactions on Knowledge and Data Engineering*, 32(8):1586–1594, 2019.

[32] Vinay O Khilwani, Vasu Gondaliya, Shreya Patel, Jay Hemnani, Bhuvan Gandhi, and Santosh Kumar Bharti. Diabetes prediction, using stacking classifier. In *2021 International Conference on Artificial Intelligence and Machine Vision (AIMV)*, pages 1–6. IEEE, 2021.

[33] Saloni Kumari, Deepika Kumar, and Mamta Mittal. An ensemble approach for classification and prediction of diabetes mellitus using soft voting classifier. *International Journal of Cognitive Computing in Engineering*, 2:40–46, 2021.

[34] Mingqi Li, Xiaoyang Fu, and Dongdong Li. Diabetes prediction based on xgboost algorithm. In *IOP conference series: materials science and engineering*, volume 768, page 072093. IOP Publishing, 2020.

[35] Francesco Mercaldo, Vittoria Nardone, and Antonella Santone. Diabetes mellitus affected patients classification and diagnosis through machine learning techniques. *Procedia computer science*, 112:2519–2528, 2017.

[36] Souad Larabi-Marie-Sainte, Linah Aburahmah, Rana Almohaini, and Tanzila Saba. Current techniques for diabetes prediction: review and case study. *Applied Sciences*, 9(21):4604, 2019.

[37] S Sivaranjani, S Ananya, J Aravinth, and R Karthika. Diabetes prediction using machine learning algorithms with feature selection and dimensionality reduction. In *2021 7th International Conference on Advanced Computing and Communication Systems (ICACCS)*, volume 1, pages 141–146. IEEE, 2021.

[38] Saba Bashir, Usman Qamar, and Farhan Hassan Khan. Intellihealth: a medical decision support application using a novel weighted multi-layer classifier ensemble framework. *Journal of biomedical informatics*, 59:185–200, 2016.

[39] Atik Mahabub. A robust voting approach for diabetes prediction using traditional machine learning techniques. *SN Applied Sciences*, 1(12):1667, 2019.

[40] Muhammad Exell Febrian, Fransiskus Xaverius Ferdinan, Gustian Paul Sendani, Kristien Margi Suryanigrum, and Rezki Yunanda. Diabetes prediction using supervised machine learning. *Procedia Computer Science*, 216:21–30, 2023.

[41] Amani Yahyaoui, Akhtar Jamil, Jawad Rasheed, and Mirsat Yesiltepe. A decision support system for diabetes prediction using machine learning and deep learning techniques. In *2019 1st International informatics and software engineering conference (UBMYK)*, pages 1–4. IEEE, 2019.

[42] Nesreen Samer El_Jerjawi and Samy S Abu-Naser. Diabetes prediction using artificial neural network. 2018.

[43] K Kannadasan, Damodar Reddy Edla, and Venkatanareshbabu Kuppili. Type 2 diabetes data classification using stacked autoencoders in deep neural networks. *Clinical Epidemiology and Global Health*, 7(4):530–535, 2019.

[44] Soniya, Sandeep Paul, and Lotika Singh. Application and need-based architecture design of deep neural networks. *International Journal of Pattern Recognition and Artificial Intelligence*, 34(13):2052014, 2020.

[45] Rashi Rastogi and Mamta Bansal. Diabetes prediction model using data mining techniques. *Measurement: Sensors*, 25:100605, 2023.

[46] Siyuan Hou, Xiaomin Li, Fanyue Meng, Shaokun Liu, and Zhenlin Wang. A machine learning–based prediction of diabetes insipidus in patients undergoing endoscopic transsphenoidal surgery for pituitary adenoma. *World Neurosurgery*, 2023.

[47] G Van Rossum and FL Drake. Python 3 reference manual (scotts valley, ca: Createspace;).[google scholar]. 2009.

[48] Stefan Van Der Walt, S Chris Colbert, and Gael Varoquaux. The numpy array: a structure for efficient numerical computation. *Computing in science & engineering*, 13(2):22–30, 2011.

[49] Wes McKinney et al. Data structures for statistical computing in python. In *Proceedings of the 9th Python in Science Conference*, volume 445, pages 51–56. Austin, TX, 2010.

[50] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, 12:2825–2830, 2011.

[51] John D Hunter. Matplotlib: A 2d graphics environment. *Computing in science & engineering*, 9(03):90–95, 2007.

[52] Kiranbala Nongthombam and Deepika Sharma. " data analysis using python. *International Journal of Engineering Research & Technology (IJERT)*, 2021.

[53] Fredrik Lundh. An introduction to tkinter. *URL: www. pythonware. com/library/tkinter/introduction/index. htm*, 1999.

[54] John W Tukey et al. *Exploratory data analysis*, volume 2. Reading, MA, 1977.

[55] Suad A Alasadi and Wesam S Bhaya. Review of data preprocessing techniques in data mining. *Journal of Engineering and Applied Sciences*, 12(16):4102–4107, 2017.

[56] Prashant Brahmbhatt and Siddhi Nath Rajan. Skin lesion segmentation using segnet with binary crossentropy. In *Proceedings of the International Conference on Artificial Intelligence and Speech Technology (AIST2019), Delhi, India*, pages 14–15, 2019.

Webography

[W1] , https://www.painscale.com/article/type-1-vs-type-2-diabetes,
Last access to the site: 08/05/2023

[W2] , https://en.wikipedia.org/wiki/Semi-Supervised_Learning,
Last access to the site: 18/03/2023

[W3] , https://www.javatpoint.com/types-of-machine-learning,
Last access to the site: 08/05/2023

[W4] , https://www.javatpoint.com/k-means-clustering-algorithm-in-machine-learning,
Last access to the site: 08/05/2023

[W5] , https://shorturl.at/duBF0,
Last access to the site: 08/05/2023

[W6] , https://www.kdnuggets.com/2022/08/decision-trees-random-forests-explained.html,
Last access to the site: 08/03/2023

[W7] , https://vitalflux.com/hard-vs-soft-voting-classifier-python-example/,
Last access to the site: 08/05/2023

[W8] , https://scikit-learn.org/stable/modules/cross_validation.html,
Last access to the site: 06/06/2023

[W9] , https://jupyter.org/about,
Last access to the site: 14/05/2023

[W10] , https://www.kaggle.com/datasets/pradeepgurav/multiples,
Last access to the site: 09/05/2023

[W11] , https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91
Last access to the site: 29/05/2023

[W12] , https://mdnice.com/writing/4cf0d60eeba64eb1b8fa39cac3694290,
Last access to the site: 18/06/2023