Yugoslav Journal of Operations Research
35 (2025), Number 4, 919-946
DOI: https://doi.org/10.2298/YJOR240315054A

Research Article

DIAGNOSIS OF DIABETES USING BAYESIAN AND BOOSTING CLASSIFIER

Zahra AHMADIAN
Faculty of Statistics, Mathematics, and Computer Science, Allameh Tabataba'i University, Tehran, Iran
zahmdyan0@gmail.com, ORCID: 0009-0006-6572-8723

Farzad ESKANDARI
Faculty of Statistics, Mathematics, and Computer Science, Allameh Tabataba'i University, Tehran, Iran
askandari@atu.ac.ir, ORCID: 0000-0002-3760-5746

Shokouh SHAHBEYK*
Faculty of Statistics, Mathematics, and Computer Science, Allameh Tabataba'i University, Tehran, Iran
sh_shahbeyk@atu.ac.ir, ORCID: 0000-0001-7502-5921

Received: March 2024 / Accepted: October 2024

Abstract: In recent years, the diagnosis of diseases using artificial intelligence and machine learning algorithms has gained significant importance. Using data from relevant medical studies enables the extraction of valuable insights that can reduce the occurrence of numerous fatalities. One of the rapidly growing chronic diseases is diabetes, which has shown a growing prevalence due to urbanization and reduced physical activity. Hence, early detection of diabetes in individuals has immense significance. This paper utilizes a dataset comprising information from individuals who underwent diabetes diagnostic tests and employs classification techniques to determine whether their test results were positive or negative for diabetes. The novelty of this work lies in the comparative analysis of Bayesian classifiers and boosting methods, which have not been extensively explored in the literature. The utilized classification methods include Bayesian classifiers such as Bayesian Support Vector Machine, Bayesian k-nearest neighbor, Bayesian decision tree and boosting methods like Catboost, Adaboost, and XGboost. Performance evaluation metrics, including accuracy, precision, recall, F1-score, and ROC curve analysis, are employed to compare the efficacy of these methods in analyzing the data. The findings of this study will contribute to the advancement of accurate and efficient diabetes diagnosis

* Corresponding author

using machine learning techniques, potentially helping in early intervention and management of the disease.

Keywords: Bayesian support vector machine, Bayesian k-nearest neighbor, Bayesian decision tree, boosting classification, diagnosis of diabetes.

MSC: 62H30, 62F15, 68T05, 62P10.

1. INTRODUCTION

The main goal of data science is to develop theoretical foundations and concepts for extracting valuable information and useful insights from given data and finally apply them to solve real-world problems. One of these important real-world problems is to classify people based on their diabetes test results into positive and negative categories. Early and accurate diagnosis of diabetes is critical because it allows for timely intervention, management, and treatment, which can significantly reduce the risk of severe complications such as heart disease, kidney failure, stroke, and blindness. This highlights the importance of developing and comparing various machine learning algorithms to identify the most effective method for diabetes diagnosis.

In this paper, we address a notable gap in the existing literature: the comparative analysis of Bayesian and boosting classification methods specifically for diabetes diagnosis. Despite extensive research on machine learning applications in healthcare, there is a lack of studies directly comparing the efficacy of Bayesian classifiers against boosting methods in the context of diabetes, which is critical given the complexity and variability of the disease's clinical manifestations. Our objective is to rigorously compare these methods to identify the most effective approach for early and accurate diabetes diagnosis, potentially contributing to better patient management and treatment outcomes.

Recent advancements in artificial intelligence (AI) have opened new avenues for managing chronic diseases, including diabetes. AI-powered applications have shown the potential to improve patient outcomes by providing continuous monitoring and personalized treatment plans, as discussed in the work by Arefin [1] on chronic disease management. These AI tools actively engage patients in their healthcare, which is crucial for managing chronic conditions effectively. Furthermore, AI's role in enhancing disease diagnosis is demonstrated by Abdollahi and Safa [2], who explored its applications in Parkinson’s disease diagnosis, showing how machine-learning techniques can lead to early and accurate detection. This highlights the potential for similar AI-driven methods to be applied to diabetes diagnosis.

The potential of AI in healthcare extends beyond direct diagnosis and management. Khalifa and Albadawy [3] emphasized AI's transformative role in diabetes care, including prevention, diagnosis, and effective management through predictive modeling and personalized care. Mackenzie et al. [4] further explore AI's broader applications in diabetes, beyond commonly discussed closed-loop systems, highlighting its role in patient education, self-management, and clinical decision support systems.

A lot of research has been carried out in the diagnosis of diabetes in the past, using machine learning algorithms such as logistic regression, Naive Bayes, and K-Nearest Neighbor (KNN) to diagnose diabetes, concluding that logistic regression has high accuracy for classifying people [5]. Choudhury and Gupta [6] predict diabetes during the design of medical diagnosis software. They used machine learning algorithms such as

neural network, decision tree (DT), Naive Bayes, random forest (RF), KNN, support vector machine (SVM), and logistic regression.

Khanam and Foo [7] have used six machine learning algorithms, such as support vector machine, KNN, gradient boosting, decision tree, random forest, and logistic regression, to detect patterns and risk factors in India's Pima diabetes dataset. Sonar and JayaMalini [8] used classification methods such as Naive Bayes, SVM, artificial neural network (ANN), and decision trees to predict the diabetes risk level of patients. They concluded that the accuracy of the decision tree is higher than the rest of the algorithms. After preprocessing the data and selecting the critical features, Sivaranjani and coworkers [9] used principal component analysis (PCA) as a method to reduce the dimension and finally used random forest algorithms and support vector machine to diagnose diabetes and concluded that the accuracy of the random forest algorithm is more than the support vector machine. Algorithms such as KNN, random forest, logistic regression, decision tree, and the Ensemble approach to classifying type 2 diabetes datasets are used by Roobini and Mohandoss [10].

Rajput and Alashetty [11] analyzed PIMA diabetes data, using several classification algorithms such as SVM, logistic regression, decision tree, and random forest, and concluded that SVM and random forest have high accuracy. Patel and Briskilal [12] used machine learning techniques such as KNN, logistic regression, decision tree, SVM, Light Gradient Boosting Machine (LightGBM), and random forest for early diagnosis of diabetes. They concluded that random forest is more accurate than other algorithms.

Louk et al. [13] focused on PE Malware Analysis, using various tree-based ensemble learning methods such as random forest, Extreme Gradient Boosting (XGBoost), Categorical Boosting (Cat Boost), Gradient Boosting Machine (GBM), and LightGBM. Wee et al. [14] provided a comprehensive review of diabetes detection using machine learning and deep learning approaches. Their study emphasized the potential of non-invasive and anthropometric measurements in creating cost-effective and high-performance diabetes detection systems. They also discussed the impacts of oversampling techniques and data dimensionality reduction through feature selection approaches, pointing out the future direction for improving accuracy and reliability in diabetes identification.

The comparison of different machine learning methods for predicting diabetes has shown varying results. For instance, the study by Yulia Resti et al. [15] found that Multinomial Naive Bayes outperformed other methods like Fisher Discriminant Analysis and Logistic Regression, achieving performance measures exceeding 93%. Additionally, G. Parthiban et al. [16] demonstrated the use of the Naive Bayes classifier in predicting heart disease in diabetic patients, showcasing the effectiveness of Bayesian methods in medical diagnostics.

Moreover, Chou et al. [17] explored predicting the onset of diabetes using machine learning methods. Their study, based on outpatient examination data from a Taipei Municipal Medical Center, highlighted the efficacy of neural networks and boosted decision trees in predicting diabetes. Their study comparing various BERT models from Hugging Face with traditional machine learning techniques demonstrated the superiority of these models in predicting mental health disorders, reinforcing the broader applicability of machine learning in healthcare diagnostics.

In another study, Ebrahimzadeh and Safa [18] discussed the transformative potential of the Metaverse in healthcare, emphasizing how machine learning can use data generated

within this virtual environment to improve medical services. Finally, a study by Pourkeyvan and colleagues [19] highlights the application of machine learning techniques for predicting mental health disorders using social media data. This approach underscores the potential of using diverse data sources and advanced machine learning models to enhance early detection and intervention in healthcare, drawing a parallel to this work in diabetes diagnosis.

By reviewing past research, we find that different algorithms have been compared in the diagnosis of diabetes, and it can be argued that boosting algorithms have been compared with most classification methods in machine learning. However, the vital point in this article is the comparison between boosting classification algorithms and Bayesian classification methods, which has not been done so far, highlighting the uniqueness of this study. By identifying the most effective classification method for diabetes diagnosis, this study aims to contribute to the development of more accurate and reliable diagnostic tools, which can ultimately enhance clinical decision-making and patient outcomes.

The rest of the article is organized as follows, Section 2 explains the algorithms and methods used. Section 3 defines model evaluation metrics such as confusion matrix, ROC curve, accuracy, precision, recall, and F-measure. Section 4 deals with implementing the methods mentioned in Section 2 on the data. Section 5 deals with the conclusion and summary.

2. MATERIALS AND ALGORITHMS

This section will be dedicated to the detailed examination and discussion of several algorithms, namely, CatBoost, AdaBoost, XGBoost, Bayesian support vector machine (BSVM), Bayesian K-nearest neighbor (BKNN), and Bayesian decision tree (BDT).

2.1. Categorical Boosting (CatBoost)

Before defining and examining the CatBoost algorithm, we will give a general explanation of ensemble classification methods. As the name suggests, these methods utilize a collection of learners, such as different classification algorithms and techniques, in a repetitive manner. This leads to the creating models and classifications that are significantly more accurate and have fewer weaknesses. In general, ensemble algorithms can be divided into two categories: Bagging algorithms and Boosting algorithms. The CatBoost algorithm is a subset of Boosting algorithms. CatBoost was introduced by Yandex company in 2017 [20]. This algorithm has successfully managed to handle categorical inputs without directly performing encoding operations in the preprocessing stage. It also handles numerical and textual features. The existence of appropriate encoding within the algorithm has led to its recognition and has provided more advantages compared to other classification algorithms. Furthermore, during the training process of this algorithm, a set of symmetric decision trees (symmetric tree: a tree in which each parent node has either zero or two child nodes) is sequentially built with reduced cost compared to the previous tree. This algorithm uses a novel scheme for computing leaf values when selecting the tree structure, which helps reduce overfitting. Thus, the combination of new encoding features for categorical variables (using ordered target statistics) and overfitting reduction contributes to the superiority and power of this algorithm compared to others. Figure 1 illustrates an example of the CatBoost classifier structure.

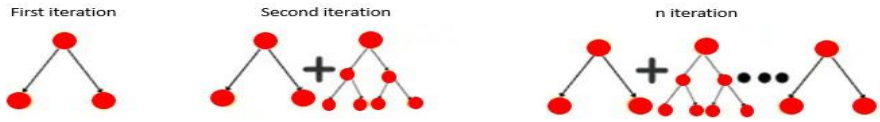

Figure 1: An example of the CatBoost classifier structure

Imagine that we observe data with samples $D = \{(X_i, y_i)\}_{i=1\dots n}$where $X_i = (x_i^1, x_i^2, \dots, x_i^m)$ is a random vector composed of m features, and $y_i \in R$ represents the target variable, which can take the form of either a binary value (i.e., positive and negative) or a numerical value (0,1). The samples (X_i, y_i) are distributed independently and identically according to an unknown probability distribution $P(\cdot,\cdot)$. The objective of the learning task is to train a function $H: R^m \to R$ that minimizes the expected loss as specified in Equation 1.

$$\mathcal{L}(M) := EL(y, M(X)) \tag{1}$$

In this context, $L(.,.)$represents a loss function with smooth properties, while $(X, y$ denotes testing data sampled from the training dataset D.

The process of gradient boosting [20] builds a sequence of successive approximations, denoted as $M^t: R^n \to R$, $t = 0,1,\dots$ with t ranging from 0 to infinity, in a step-by-step and greedy manner. Derived from the preceding approximation M^{t-1}, M^t is obtained using an additive mechanism such that $M^t = M^{t-1} + \alpha h^t$, using a step size α and function $h^t : R^m \to R$, which is a base predictor, selected from a set of functions H to reduce or minimize the expected loss defined in Equation 2.

$$h^t = argmin_{h\in H}\mathcal{L}(M^{t-1} + h) = argmin_{h\in H}EL\big(y, M^{t-1}(X) + h(X)\big) \tag{2}$$

To solve this minimization problem, gradient boosting typically employs techniques like gradient descent. Here, h^t is approximated by taking a step in the direction of the negative gradient of the loss function evaluated at M^{t-1} This process is repeated for a predefined number of steps or until the improvements become negligible [21,22]. Additional information about the CatBoost algorithm can be found in [23].

As we mentioned, the CatBoost algorithm uses the ordered target statistics method to code the data. The formula for the collected target statistic is stated in Equation 3:

$$target\ statistic = \frac{current\_count + (a \times p)}{maximum\_count + a} \tag{3}$$

In the above equation:

- a and prior (p) are the constant parameters, by default, equal to 1 and 0.5.
- current_count is the sum of all the output class values of the same category that the current row in the dataset belongs to.
- maximum_count is the sum of the same category items above the current row.

For a better understanding, we will give an example.

Example 1: Assume that we want to encode the simulated dataset using ordered target statistics in Table 1.

Table 1: Simulated example

colour	length	type
red	5	1
red	5	2
red	4	1
green	5	0
green	7	0
blue	2	1

Let's consider the situation where we are converting the item in the third row that is explicitly written in blue color. maximum_count=2, current_count=1+2=3, a=1, p=0.5.

The numeric value is:

$$\frac{(1+2)+(1\times 0.5)}{2+1}=1.166$$

The algorithm utilizes a consistent approach to label encoding for all categorical data in the dataset. It applies the same formula to handle categorical variables across the entire dataset.

Table 1 becomes Table 2.

Table 2: Simulated example

colour	length	type
red	5	1
red	5	2
1.166	4	1
green	5	0
green	7	0
blue	2	1

2.2. Adaptive Boosting (AdaBoost)

The Adaptive Boosting (AdaBoost) algorithm is a machine learning technique used to enhance the accuracy of other boosting algorithms. It operates by training weaker rules to create a boosted algorithm. In AdaBoost, the input consists of a training set $(x_1, y_1), \ldots, (x_m, y_m)$ where each x_i belongs to an instance space, and each corresponding label y_i belongs to a label set Y (in this case, assuming $Y = \{-1, +1\}$). The algorithm iteratively applies a given weak or base learning algorithm in a series of rounds denoted as $t = 1, \ldots, T$. One crucial aspect of the algorithm is the maintenance of a distribution or set of weights over the training set. The weight assigned to training sample i at round t is represented as $D_t(i)$. Initially, all weights are set equally, but in each round, the weights of misclassified samples are increased to emphasize difficult samples in the training set. The objective of the weak learner is to find a weak hypothesis $h_t: X \rightarrow \{-1, +1\}$ that is suitable for the distribution. The error of a weak hypothesis is used as a metric to assess its quality. The algorithm's procedure is outlined in Algorithm 1. More explanation can be obtained in [24].

Algorithm 1: The Adaboost algorithm

Given: $(x_1, y_1), \ldots, (x_n, y_n)$ where $x_i \in X,\ y_i \in Y = \{1, -1\}$

Set $D_1(i) = \frac{1}{n}$ (initial equal weight for all samples)

For $t = 1, \ldots T$:

- Train weak learner with distribution D_t.
- Get weak hypothesis $h_t : X \rightarrow \{-1,1\}$ with error $\varepsilon_t = P_{ri\sim Dt}[h_t(x_i) \neq y_i]$
- Choose $\alpha_t = 0.5ln\ (\frac{1-\varepsilon_t}{\varepsilon_t})$
- Update

$$D_{t+1}(i) = \frac{D_t(i)}{Z_t} \times \begin{cases} \exp(-\alpha_t) & if\ h_t(x_i) = y_i \\ exp(\alpha_t) & if\ \ h_t(x_i) \neq y_i \end{cases} \tag{4}$$

$$= \frac{D_t(i)\exp\,(-\alpha_t y_i h_t(x_i))}{Z_t}$$

Here, $D_{t+1}(i)$ is the updated weight for sample i, where Z_t is a normalization factor to ensure the weights sum to 1. The exponential factor increases the weight of misclassified samples (when $y_i \neq h_t(x)$), ensuring the next weak learner focuses more on these difficult samples. Output**:**

$$H(x) = sign(\sum_{t=1}^{T} \alpha_t h_t(x)) \tag{5}$$

The final strong classifier $H(x)$ is a weighted majority vote of the T weak classifiers, where each α_t determines the weight of the corresponding $h_t(x)$.

2.3. Extreme Gradient Boosting (XGBoost)

XGBoost is a distributed gradient boosting library specifically designed to efficiently and effectively train machine learning models. It employs an ensemble learning approach by combining the predictions of multiple weak models to generate a more robust prediction. XGBoost has gained significant popularity in machine learning due to its ability to handle large datasets and achieve state-of-the-art performance in various tasks such as classification and regression. To learn how this algorithm works, we have: Given data with m-samples and n-features, $D = \{(X_j, y_j)\}(|D| = m, X_j \in R^n, y_j \in R)$, a tree ensemble model employs a combination of L additive functions to make predictions for the output, as outlined in (6):

$$\hat{y}_j = \phi(X_j) = \sum_{l=1}^{L} h_l(X_j), \quad h_l \in H\ . \qquad , \tag{6}$$

Here, $\phi(X_j)$ is the prediction for sample j, represented as the sum of L functions $h_l(X_j)$ from the space of regression trees H. Each h_l corresponds to one tree in the ensemble. Where $H = \{h(X) = w_q(X)\}(q : R^n \rightarrow U, w \in R^n)$, the space of regression trees represents the set of all possible trees used for regression. In this context, q represents the structure of each tree, which maps a sample to its corresponding leaf index. The variable U represents the total number of leaves in the tree. Each h_l refers to the independent structure of tree q and the leaf weights w.

$$\mathcal{L}(\phi) = \sum_j l(\hat{y}_j, y_j) + \sum_l \Omega(h_l)\, \Omega(h_l) = \gamma U + \frac{1}{2}\lambda||w||^2\ , \tag{7}$$

where l is a differentiable convex loss function that measures the difference between the y_j , $\hat{y}_j$.

The loss function $L(\phi)$ measures the difference between the predicted values $\widehat{y_j}$. and the actual values y_j using a differentiable convex loss function l. The regularization term $\Omega(h_l)$ penalizes the complexity of the model to prevent overfitting, where γ and λ are regularization parameters.

To find the best tree structure, XGBoost minimizes the following objective function during training:

$$\tilde{\mathcal{L}}^{(u)}(q) = \frac{-1}{2}\sum_{j=1}^{U}\frac{(\sum_{i\in I_j} f_i)^2}{\sum_{i\in I_j} g_i+\lambda} + \gamma U \,, \tag{8}$$

where $f_i = \partial_{\hat{y}^{(u-1)}} l(y_j, \hat{y}^{(u-1)})$ and $g_i = \partial^2{}_{\hat{y}^{(u-1)}} l(y_j, \hat{y}^{(u-1)})$. In these equations, f_i and g_i are the first and second-order gradient statistics, respectively, which help in optimizing the loss function. The gradients (f_i) and (g_i) provide the necessary information to update the model parameters and improve prediction accuracy [25].

2.4. Bayesian Support Vector Machine (BSVM)

Bayesian Support Vector Machine (BSVM) extends the traditional SVM algorithm by incorporating Bayesian inference principles. It combines the strengths of SVM, known for its ability to handle high-dimensional data and nonlinear relationships, with the probabilistic framework of Bayesian methods.

In Bayesian SVM, instead of finding a single optimal hyperplane that maximally separates the data into classes, it seeks to estimate the posterior probability distribution over the hyperplane parameters. This allows for a more robust and flexible classification approach, especially in situations with limited training data or noisy training data.

The Bayesian framework introduces prior distributions over the hyperplane parameters, representing prior knowledge or assumptions about the problem. By incorporating prior information, Bayesian SVM can effectively regularize the model and handle overfitting. During the learning process, the algorithm updates the prior distribution to obtain the posterior distribution using Bayes' rule, which combines the prior knowledge with the observed data.

To make predictions, Bayesian SVM considers the entire posterior distribution of the hyperplane parameters instead of relying solely on a single solution. This enables the estimation of class probabilities and provides a measure of uncertainty in the predictions. In addition, Bayesian SVM can naturally handle model selection by comparing different models through their posterior probabilities.

One common approach to implementing Bayesian SVM is using Markov chain Monte Carlo (MCMC) sampling techniques, such as Gibbs sampling or the Metropolis-Hastings algorithm, to approximate the posterior distribution. These sampling methods iteratively draw samples from the posterior, allowing for inference and prediction.

The advantage of Bayesian SVM is its ability to incorporate prior knowledge, handle uncertainty, and provide probabilistic predictions. However, it comes at the cost of increased computational complexity compared to traditional SVM, which involves sampling from the posterior distribution.

In Support Vector Machines (SVM), the objective is to minimize a cost function as follows:

$$d_\alpha(\beta, v) = \sum_{i=1}^{n} \max(1 - y_i x_i^T \beta, 0) + v^{-\alpha} \sum_{j=1}^{k} |\frac{\beta_j}{\sigma_j}|^2 \ , \quad (9)$$

where x represents the feature vector, y represents the target, β represents the coefficients, σ_j is the standard deviation of the *j-th* element of x, and v is a tuning parameter. The second part of this formula is the regularization term, which can take various forms. However, the objective of this article is to estimate the parameters of Support Vector Machines in a Bayesian framework, meaning that the estimation of parameter β will be conducted using Bayesian inference. Minimizing the cost function 9 corresponds to finding the mode of the pseudo-posterior distribution $p(\beta|v, \alpha, y)$defined by:

$$p(\beta|v, \alpha, y) \propto \exp\big(-d_\alpha(\beta, v)\big) \propto C_\alpha(v) L(y|\beta) p(\beta|v, \alpha). \quad (10)$$

The factor of $C_\alpha(v)$is a pseudo-posterior normalization constant that is absent in the classical analysis. The data-dependent factor $L(y \mid \beta)$ is a pseudo-likelihood:

$$L(\, y \mid \beta\,) = \prod_i L_i\,(y_i \mid \beta) = exp\{-2 \sum_{i=1}^{n} max\ (1 - y_i x_i^T \beta, 0)\}. \quad (11)$$

This equation is established because it demonstrates that the likelihood function of the first kind follows a mixture of normal distributions. Therefore, we will have:

$$\mathrm{L}(\mathrm{y}|\beta) = \exp\,(-2 \sum_{i=1}^{n} \max\big(1 - y_i x_i^T \beta, 0\big)) \ = \int_0^\infty \frac{1}{\sqrt{2\pi\lambda}} \exp\left(-\frac{1}{2} \frac{\big(1+\lambda_i - y_i x_i^T \beta\big)^2}{\lambda_i}\right) d\lambda$$

$$= \int_0^\infty \frac{1}{\sqrt{2\pi\lambda}} \exp(-u)\, \exp\,(-\frac{1}{2}(\lambda + u^2 \lambda^{-1}) d = \int_0^\infty \frac{1}{\sqrt{2\pi\lambda}} \exp\,(-u - \frac{\lambda}{2} - \frac{u^2 \lambda^{-1}}{2} d\lambda$$

$$= \int_0^\infty \frac{1}{\sqrt{2\pi\lambda}}\ \exp\,(\frac{-(u+\lambda)^2}{2\lambda} d\lambda = exp(-2\, max(u, 0)), \quad (12)$$

where λ is a latent variable. Based on the results obtained from 12, the pseudo-posterior distribution can be expressed as Equation 13:

$$p(\beta, \lambda, w|y, v, \alpha)$$

$$\propto \prod_{i=1}^{n} \lambda_i^{\frac{-1}{2}} \exp\,(\frac{-1}{2} \sum_{i=1}^{n} \frac{\big(1+\lambda_i - y_i x_i^T \beta\big)^2}{\lambda_i}) \prod_{j=1}^{k} w_j^{\frac{-1}{2}} \times exp\left(\frac{-1}{2v^2} \sum_{j=1}^{k} \frac{\beta_j^2}{\sigma_j^2 w_j}\right) p\big(w_j|\alpha\big), \quad (13)$$

where $p\big(w_j \mid \alpha\big) \propto w_j^{\frac{-3}{2}} St_{\frac{\alpha}{2}}^{+}\big(w_j^{-1}\big)$, which, as a result, using Markov Chain Monte Carlo (MCMC), we will obtain the value of β for parameter estimation. Further details can be found in [26].

Figure 2 illustrates the concept of Bayesian SVM by showing a hyperplane with confidence bands. The black line represents the optimal separating hyperplane determined by the Bayesian SVM, which maximizes the margin between the classes. The shaded region around the hyperplane indicates the confidence bands, reflecting the uncertainty in the classification boundary. These confidence bands represent the range within which the hyperplane could reasonably be positioned, based on the variability of the data.

The figure demonstrates that Bayesian SVM does not rigidly rely on a single hyperplane for classification but instead considers multiple potential hyperplanes. This probabilistic approach allows for more flexible decision-making, where the model can account for overlapping data and provide a measure of confidence in its predictions. This

capability to represent uncertainty makes BSVMs valuable in applications requiring reliable risk assessments and decision-making under uncertainty.

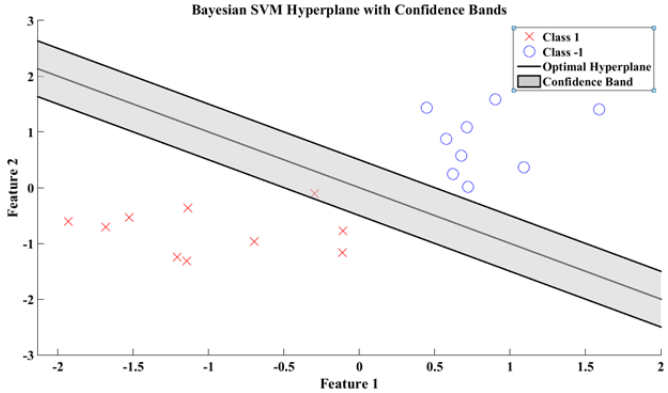

Figure 2: Bayesian SVM Hyperplane with Confidence Bands

2.5. Bayesian Decision Tree (BDT)

Bayesian Decision Tree is a specific variation of a decision tree that integrates Bayesian principles and probabilistic reasoning into its construction and interpretation. In contrast to conventional decision trees, which primarily aim to maximize accuracy, Bayesian Decision Trees consider the inherent uncertainty linked to each decision and assign probabilities to various potential outcomes. Constructing Bayesian Decision Trees involves considering prior knowledge or beliefs about the data and updating them based on observed evidence. This is done using Bayesian inference, which allows for more flexible and nuanced modeling of the relationships between variables. By incorporating prior knowledge and updating it with new information, Bayesian Decision Trees can provide more accurate predictions and account for uncertainties in the data. Bayesian Decision Trees can handle both regression and classification problems. The algorithm for constructing these trees does not rely on computationally intensive methods like Markov Chain Monte Carlo, making it more efficient and scalable. We establish D as a dataset consisting of n independent observations, denoted as $\{(x_i, y_i)$ for i ranging from 1 to n. The points $x = (x^1, \dots, x^d)$ in R^d represent the features of each observation, where the outcome y is randomly selected from Y_x.

The nature of the problem we are addressing is determined by the distribution of Y_x. If Y_x follows a discrete probability distribution, it corresponds to a classification problem. On the other hand, if Y_x follows a continuous probability distribution, it corresponds to a regression problem. The beta function will play a significant role in calculating the likelihood of the classification examples.

$$\beta(z^1, \dots, z^C) = \frac{\prod_{c=1}^{C} \tau(z^c)}{\tau(\sum_{c=1}^{C} z^c)}\ , \tag{14}$$

C is different classes, and τ is the gamma function. The dataset D is obtained by sampling from a data generation process. We can separate this process into two distinct steps:

First, a point x is randomly selected from the d-dimensional space R^d; second, the outcome Y_x is sampled given x. No prior knowledge regarding the generation of locations

x is taken into consideration. As a result, our primary focus lies in examining the distribution of Y_x. The conditional distribution is presumed to be represented within a tree structure, constructed using a set of straightforward recursive rules based on $\{x_i\}$ for i ranging from 1 to n. This tree-generation process determines the organization of the tree. From the root of the tree, we decide whether to expand the current node by creating two new leaves based on a predetermined probability. This probability may vary depending on the current depth of the tree. If expansion is not chosen, the process ends for that particular node. However, if expansion is selected, we determine which dimension within d should be utilized for the split. Once the dimension for the split is determined, we assume that the exact position of the split within the available range is distributed uniformly among all distinct points within that range. After establishing the specific location of the split, the process continues iteratively by determining whether each new leaf should be further split. This iteration repeats until there are no more nodes that require splitting or when each leaf contains only a single distinct set of observations. At this point, the tree construction process is considered complete. Considering the aforementioned generating process, it becomes evident that the fundamental element of Bayesian Decision Trees involves examining partitions of R^d that provide a more adequate explanation of the outcomes using a probabilistic approach. Within a partition Π, all points found in the same set exhibit an equivalent outcome distribution. This implies that the outcome variable Y is independent of the predictor variable x. By assuming a prior distribution for the parameters of the outcome distribution Y, it becomes possible to derive the likelihood of each set within a partition. As all observations are assumed to be independent, the overall partition likelihood $L(D|\Pi)$ is calculated by multiplying the likelihoods of each set within the partition.

Now we explain the partition space, which is the building block for the Bayesian Decision Trees. Let $D = (x_i, y_i), i = 1, \dots, n$ be a dataset consisting of n independent observations in $M \times R$, $M \subseteq R^d$. A partition $\Pi = \{M_1, M_2, \dots, M_k\}$ of M divides M into disjoint subsets $M_1, M_2, \dots, M_k$ such that $=\cup M_w$. As a result, the dataset D will be split into $D_{M_1}, \dots., D_{M_k}$, where the observation (x_i, y_i) belongs to D_{M_w} if and only if $x_i \in M_w$. All Y_{M_w} sampled within region M_w follows the same distribution with probability measure ρ_q and parameters q in R^s. We assume a prior distribution γM_w for parameters q. Then, the likelihood of our data given the prior is:

$$L\left(D_{M_w}\right) = \begin{cases} \int \left(\prod_{y_{i\in D_{M_w}}} \rho_q(y_i)\right)\gamma M_w(q)\, dq & D_{M_w} \neq \emptyset \\ 1 & D_{M_w} = \emptyset \end{cases} . \tag{15}$$

The likelihood of our data given the partition is:

$$\mathrm{L(D|}\ \Pi) = \prod_{w=1}^{k} L\left(D_{M_w}\right). \tag{16}$$

Considering $p(\Pi)$ *over* Ω_Π, the posterior distribution of our data is:

$$p(\Pi|D) = \frac{L(D|\ \Pi)\times p(\Pi)}{p(D)} = \frac{L(D|\ \Pi)\times p(\Pi)}{\int L(D|\ \Pi)\times p(\Pi)} \tag{17}$$

Finally, the posterior distribution of q in each region $M_w$is:

$$\mu(q) = \frac{L(y_i \in D_{M_w}|q)\gamma M_w(q)}{\int L\left(y_i \in D_{M_w}|q\right)\gamma M_w(q)dq}. \tag{18}$$

Step-by-step explanation for Bayesian Decision Tree construction:

1. **Data Collection and Initial Setup**: Gather the dataset D consisting of n independent observations (x_i, y_i).
2. **Partitioning the Feature Space**: Divide the feature space $M \subseteq R^d$ into disjoint subsets $\{M_1, M_2, \dots, M_k\}$.
3. **Prior Distribution Assumption**: Assume a prior distribution γM_w for the parameters q of the outcome distribution Y.
4. **Likelihood Calculation for Each Partition**: For each partition M_w, calculate the likelihood $L(D_{M_w})$ using the (16) formula.
5. **Overall Likelihood Calculation**: Combine the likelihoods of each partition to get the overall likelihood equation (17).
6. **Posterior Distribution Calculation**: Calculate the posterior distribution of the partitions using equation (18).
7. **Parameter Estimation**: Finally, estimate the parameters q for each region M_w, using the posterior distribution equation (18).

Further details can be found in [27].

2.6. Bayesian K-Nearest Neighbor (BKNN)

Bayesian k-nearest neighbor (BKNN) is an enhanced version of the conventional k-nearest neighbor (KNN) algorithm, integrating Bayesian inference to improve prediction capabilities. KNN is a non-parametric classification algorithm that assigns a class label to an unseen data point by considering the majority class labels of its k nearest neighbors in the training dataset. In BKNN, Bayesian inference is employed to estimate the posterior probability of each class label, going beyond the simple majority vote of the k nearest neighbors. This is achieved by taking into account the prior probabilities of the classes and evaluating the likelihood of observing the class labels in the training data based on the neighbors' features. The steps involved in BKNN are as follows:

1. **Data Collection**: Gather the dataset D in blocks, each containing pairs of features X and labels Y.
2. **Nearest Neighbor Identification:** Identify the k nearest neighbors of the test data point by utilizing a distance metric (e.g., Euclidean distance) in the feature space. These neighbors correspond to the training instances with the closest feature values to the test instance.
3. **Likelihood Calculation:** Calculate the likelihood of observing each class label in the training data, given the features of the neighbors. This involves counting the occurrences of each class label among the neighbors.
4. **Bayes' Theorem Application:** Utilize Bayes' theorem to compute the posterior probability of each class label, incorporating the prior probabilities and the calculated likelihoods. The posterior probability indicates the probability of a class label considering the observed features of the neighbors.
5. **Prediction:** Finally, assign the class label with the highest posterior probability as the predicted class label for the test data point.

The Bayesian approach in BKNN enables the integration of prior knowledge and facilitates handling uncertainty by accounting for the probabilistic nature of the problem. It offers a more robust and flexible prediction method compared to the traditional KNN

algorithm, particularly in scenarios involving imbalanced class distributions or limited training data.

It's important to note that the performance of BKNN relies on the selection of prior probabilities, distance metrics, and the value of k. These choices should be made thoughtfully, taking into consideration the characteristics of the dataset and the specific problem being addressed.

Let's get a little more familiar with the formula: Imagine that we are sequentially receiving data D in blocks, with $D = \{(Y_1, X_1), (Y_2, X_2), \dots, (Y_k, X_k)\}$,where $Y_h = \{y_1^{(h)}, \dots, y_{n_h}^{(h)}\}$ and $X_h = \{x_1^{(h)}, \dots, x_{n_h}^{(h)}\}$. One common scenario could arise when the value of k=2 and we have a training set (Y_1, X_1) of n_1 points and a separate evaluation set (X_2) consisting of n_2 data points with unknown corresponding Y_2 values. Another usual scenario involves the sequential arrival of single observations over time, $D = \{(y_1^{(1)}, x_1^{(1)}), (y_1^{(2)}, x_1^{(2)}), \dots, (y_1^{(k)}, x_1^{(k)})\}$, with $n_h = 1$ for all h. Let $Y = (Y_1, \dots, Y_k)$=$(y_1, \dots, y_n)$ represent the collection of combined responses and $X = (X_1, \dots, X_k) = (x_1, x_2, \dots, x_n)$, represent the set of combined predictors, where $n = \sum_{h=1}^{k} n_h$, and let $T = (t_1, \dots, t_n)$ be a set of indicator variables, where $t_i = j$ indicates that the i-th data point in Y originated from the j-th block, $j \in \{1, \dots, k\}$.

Now, we establish the joint prior distribution on Y given X, β, and r to be:

$$p(Y \mid X, \beta, r) = \prod_{i=1}^{n} \frac{\exp\left[\beta\left(\frac{1}{r}\right)\sum_{j \underset{\sim}{r} i} \delta_{y_i y_j}\right]}{\sum_{f=1}^{F} \exp\left[\beta\left(\frac{1}{r}\right)\sum_{j \underset{\sim}{r} i} \delta_{f y_j}\right]}, \tag{19}$$

where δ_{ab} is the Dirac function ($\delta_{\mathrm{ab}} = 1$ if $\mathrm{a} = \mathrm{b}$, and 0 otherwise). β is an interaction parameter that controls the magnitude of the relationship between the neighboring $y_i h$ and $\sum_{j \underset{\sim}{r} i}$.denotes that the summation is over the r nearest neighbors of x_i in the set $\{X_1, \dots, X_{ti}\} \backslash x_i$ is given the distance metric $\rho(.,.)$, as previously described, t_i represents the index of the block containing x_i. The expression $\left({}^{1}/_{r}\right)\sum_{j \underset{\sim}{r} i} \delta_{f y_j}\}$ indicates the proportion of points belonging to class f in the r nearest neighbors of x_i. Using this approach, the probability of y_i is conditioned solely on the points in blocks up to and including the $t_i th$ block.

The predictive distribution for a new observation is:

$$p(y_{n+1} \mid x_{n+1}, Y, X, \beta, r) = \frac{\exp\left[\beta\left(\frac{1}{r}\right)\sum_{j \underset{\sim}{r} i} \delta_{y_{n+1} y_j}\right]}{\sum_{f=1}^{F} \exp\left[\beta\left(\frac{1}{r}\right)\sum_{j \underset{\sim}{r} i} \delta_{f y_j}\right]}, \tag{20}$$

Therefore, the most probable class for y_{n+1} is determined by the most frequent class observed among its r nearest neighbors. By employing sequential conditioning in the neighborhood, the requirement for an additional term in distribution 20 concerning the points in the original dataset x_i, for which x_{n+1} becomes one of their r nearest neighbors, is eliminated. The joint distribution 19 resembles the priors commonly encountered in Markov random field models employed in spatial statistics. The reason behind using equation 19 is that it represents a normalized distribution, and it's normalizing constant remains independent of both β and r. This normalization significantly facilitates the analysis when we treat β and r as random variables. To assume that β and r are known

and fixed a priori is unrealistic and disregards a crucial factor of uncertainty in the model. To accommodate this, we assign prior distributions to β and r, leaving the marginal predictive distribution as:

$$(y_{n+1}|x_{n+1}, Y, X, \beta, r) = \sum_r \int p(y_{n+1}|x_{n+1}, Y, X, \beta, r)p(\beta, k|Y, X)d\beta, \quad (21)$$

where $p(\beta, k|Y, X) \propto P(Y|X, \beta, r)p(\beta, r)$[28].

Figure 3 shows the Bayesian KNN classification process with probability contours. In the figure, data points from two different classes are plotted in a two-dimensional feature space, represented by different markers. The black contour line represents the decision boundary, indicating the threshold where the probability of belonging to one class equals the probability of belonging to the other class.

The probability contours illustrate how the posterior probability of class membership changes across the feature space. Areas close to the class centers have higher confidence, whereas regions near the decision boundary reflect uncertainty. By visualizing these contours, we can see how BKNN dynamically adjusts its decision boundaries based on the distribution and proximity of the neighboring data points. This capability enables BKNN to provide not only class predictions but also the likelihood of those predictions, making it a powerful tool in applications where understanding uncertainty is crucial.

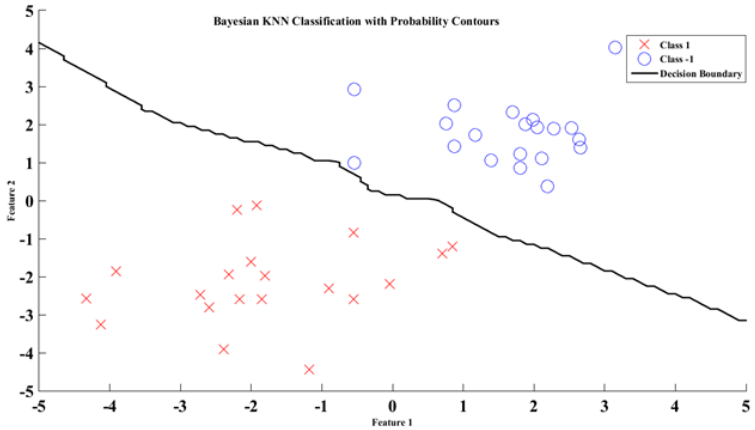

Figure 3: Bayesian KNN Classification with Probability Contours

3. MODEL EVALUATION METRICS

In this section, we will explain model evaluation metrics such as the confusion matrix, ROC curve, accuracy, precision, recall, and F-measure.

3.1. Confusion Matrix

The confusion matrix illustrates the performance of a binary classifier (Figure 4). The true (1) and false (0) actual values are compared to the positive (1) and negative (0) predictions. The confusion matrix contains the values **TP**, **TN**, **FP**, and **FN**, representing the true positive, true negative, false positive, and false negative, respectively. These values are used to estimate the classification model's capabilities.

Class designation		Actual class	
		True(1)	False(0)
Predicted class	Positive(1)	TP	FP
	Negative(0)	FN	TN

Figure 4: Confusion matrix for the binary classification problem [30]

We explain each of the components of Figure 4:

- **TP (True Positive):** In the context of the confusion matrix, a data point is considered a True Positive (TP) when it is predicted as a positive outcome, and the actual outcome confirms the prediction.
- **FP (False Positive):** When a positive outcome is predicted in the confusion matrix, but the actual outcome is negative, the data point is considered a false positive. This scenario is known as a **Type 1 Error**, and it can be seen as an instance of good intentions leading to incorrect predictions.
- **FN (False Negative):** When a negative outcome is predicted in the confusion matrix, but the actual outcome is positive, the data point is classified as a false negative. This situation is commonly referred to as a **Type 2 Error**, which is considered as equally perilous as a Type 1 Error.
- **TN (True Negative):** The data point in the confusion matrix is categorized as True Negative (TN) when a negative outcome is predicted and aligns with the actual outcome.
-

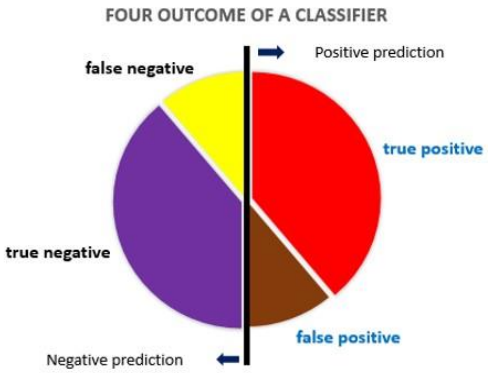

Figure 5: The elliptical form of classification result [29]

3.2. Accuracy

Accuracy is determined by adding the number of correct predictions (TP + TN) and dividing it by the total number of data points (P + N).

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \ . \tag{22}$$

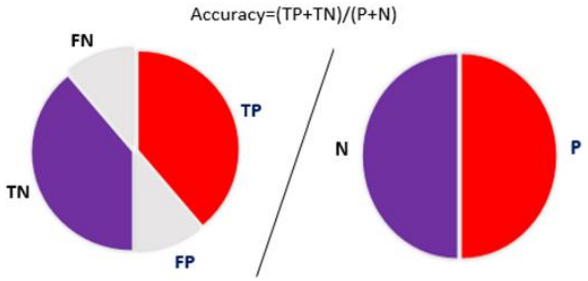

Figure 6: Two ellipses visually demonstrate the process of calculating accuracy [29]

3.3. Sensitivity or Recall

To calculate the True Positive Rate (also called Sensitivity or Recall), the number of correctly predicted positive instances (TP) is divided by the total number of positive cases (P) and also called Sensitivity or Recall (REC).

$$sensitivity\ or\ recall = \frac{TP}{TP+FN}\ . \tag{23}$$

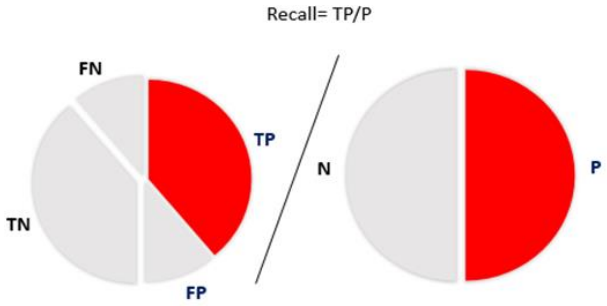

Figure 7: Two ellipses visually represent calculating Recall (True Positive Rate) [29]

3.4. Precision

Precision is computed by dividing the number of correct positive predictions (TP) by the total number of positive predictions (TP + FP).

$$Precision = \frac{TP}{TP+FP}. \tag{24}$$

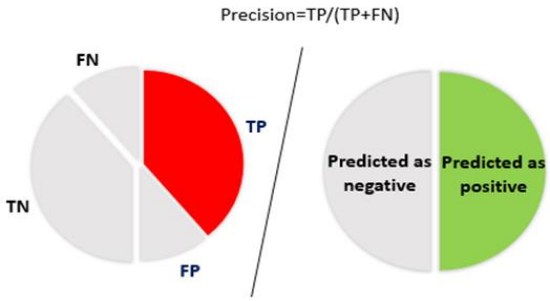

Figure 8: Two ellipses show how precision is calculated [29]

3.5. F-Measure

The F-score is an accuracy measure for a test, calculated using precision and recall. It is determined by a specific formula that considers both precision and recall.

$$F - measure = \frac{2 \times precision \times recall}{precision \times recall}. \quad (25)$$

3.6. ROC Area

Another important tool for data evaluation is the ROC (Receiver Operating Characteristic) curve. The ROC curve shows the trade-off between the true positive rate (TPR) and the false positive rate (FPR) across different threshold levels. True Positive Rate (TPR) is calculated as:

$$TPR = \frac{TP}{TP+FN}. \quad (26)$$

This metric indicates the proportion of actual positives correctly identified by the model.

False Positive Rate (FPR) is calculated as:

$$FPR = \frac{FP}{FP+TN}. \quad (27)$$

The ROC curve plots TPR on the vertical axis and FPR on the horizontal axis. The area under the ROC curve (AUC) quantifies the overall ability of the model to discriminate between positive and negative classes. An AUC closer to 1 indicates a better-performing model.

These evaluation metrics and visual representations help in understanding and interpreting the performance of classification models, enabling better decision-making based on the specific requirements of the task at hand. By providing a detailed step-by-step explanation of these metrics, readers can better comprehend how each metric contributes to evaluating and improving model performance.

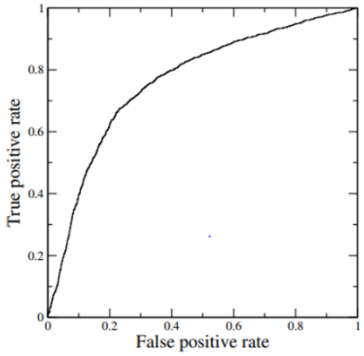

Figure 9: ROC curve [30]

3.7. Nemenyi post-hoc test

The Nemenyi post-hoc test is a statistical test used to perform pairwise comparisons following a Friedman test when there are significant differences between multiple groups. This test helps to determine which specific groups (or algorithms, in the context of machine

learning performance comparison) differ from each other. The Nemenyi test is designed to control the family-wise error rate when making multiple comparisons. It ensures that the likelihood of making one or more type I errors (false positives) is kept within a specified limit.

Methodology:

- **Ranking of Data:** Each group (e.g., algorithm) is ranked according to its performance across multiple datasets. In cases of tied ranks, average ranks are assigned.
- **Calculation of Average Ranks:** For each group, compute the average rank across all datasets. The average rank R_j for the j-th group is calculated as:

$$R_j = \frac{1}{N}\sum_{i=1}^{N} R_{ij}\ , \tag{28}$$

where N represents the number of datasets, and R_{ij} denotes the rank of the j-th group on the i-th dataset.

- **Friedman Test Statistic:** The Friedman test statistic Q is calculated using the following formula:

$$Q = \frac{12N}{k(k+1)}\left(\sum_{j=1}^{k} {R_j}^2 - \frac{k(k+1)^2}{4}\right), \tag{29}$$

where k is the number of groups.

- **Critical Difference (CD):** To determine whether the differences between pairs of groups are significant, compute the critical difference. The formula for the critical difference CD is:

$$CD = q_\alpha\sqrt{\frac{k(k+1)}{6N}}, \tag{30}$$

where q_α is the critical value from the Studentized range distribution for the chosen significance level α.

Pairwise Comparisons:

For each pair of groups, calculate the absolute difference in their average ranks. If this difference exceeds the critical difference CD, the performance differences between the two groups are considered statistically significant:

$$\left|R_j - R_m\right| > CD, \tag{31}$$

where R_j and R_m are the average ranks of the j-th and m-th groups, respectively.

4. RESULTS AND DISCUSSION

In this section, we have used the diabetes dataset taken from the UCI dataset website to analyze and evaluate. The dataset consists of 520 patient samples, each characterized by 17 distinct features, which are pivotal for diagnosing diabetes:

- **Age**: (16 years to 90 years)
- **Gender**: Male / Female

- **Polyuria**: Yes / No
- **Polydipsia**: Yes / No
- **Sudden weight loss**: Yes / No
- **Weakness**: Yes / No
- **Polyphagia**: Yes / No
- **Genital thrush**: Yes / No
- **Visual blurring**: Yes / No
- **Itching**: Yes / No
- **Irritability**: Yes / No
- **Delayed healing**: Yes / No
- **Partial paresis**: Yes / No
- **Muscle stiffness**: Yes / No
- **Alopecia**: Yes / No
- **Obesity**: Yes / No
- **Class (target variable)**: Positive / Negative

Figure 10 is a horizontal stacked bar chart that visually represents the distribution of key features in the dataset, comparing the prevalence of each feature among individuals with positive and negative diabetes diagnoses. The chart highlights the proportion of "Yes" and "No" responses for binary features and the gender distribution.

This chart effectively emphasizes the variation in symptom occurrence between individuals with positive and negative diabetes diagnoses, providing insights into the dataset's structure and the relevance of each feature for model training.

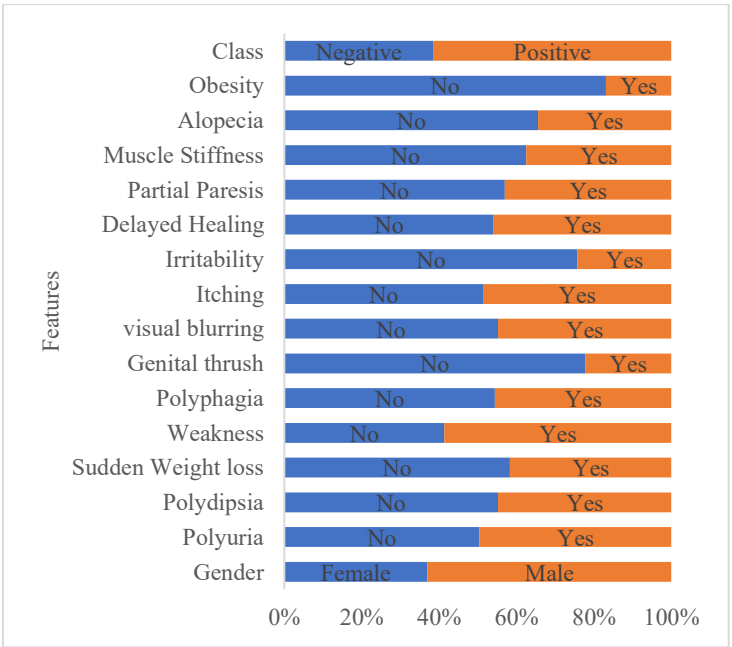

Figure 10: horizontal stacked bar chart comparing the distribution of key features

The goal is to predict the result of a person's test using the mentioned features by applying the algorithms described in Section 2 and choosing a model with the highest accuracy.

The first step of starting to examine and analyze data is data preprocessing, which is a part of data preparation and is considered a crucial preliminary step for the data mining process. Recently, data preprocessing techniques for training machine learning models and artificial intelligence models have significantly advanced.

Data preprocessing is essential for improving data quality and consistency, making it suitable for machine learning analysis. Our preprocessing steps included:

- **Checking for Missing Values:** We thoroughly checked the dataset and found no missing values, ensuring complete data entries for each feature.
- **Label Encoding:** Categorical features such as 'Gender' and binary symptoms (e.g., 'Polyuria', 'Polydipsia') were converted into a numerical format using Label Encoding. This step was necessary for algorithms that require numerical input.
- **Feature Selection:** The Chi-Square test was conducted to evaluate the significance of each feature to the target variable (diabetes diagnosis). Features with high p-values (e.g., Itching, Delayed healing, Obesity) were excluded to streamline the model training process.
- **Data Standardization:** While categorical data were encoded, continuous features (e.g., Age) were standardized using Z-score normalization to minimize the impact of differing scales.
- **Data Balancing:** Since the dataset showed class imbalance (as illustrated in Figure 9), we employed the Synthetic Minority Over-sampling Technique (SMOTE) to ensure a balanced distribution of classes, which helps prevent biased model training.

For the preprocessing of the diabetes dataset, we have checked the missing data, extracted important features, and standardized, and except for the age variable, the rest of the variables did not need to be standardized. We also used the Label Encoding method to code the categorical data. The Label Encoding is converting categorical labels or variables into numerical representations. It is commonly used when working with machine learning algorithms that require numeric inputs. However, it should be noted that during the CatBoost algorithm training, we used the data before coding because this algorithm uses special coding (ordered target statistics).

By checking the data, we found that it does not contain any missing values. For feature selection, we also used the Chi-Square Test. The Chi-Square test measures the independence between categorical variables and the target variable. It assesses whether there is a significant association between the feature and the target. Features with high chi-square statistics and p-values below a certain threshold can be considered essential and selected for further analysis.

Because the chi-square method uses classification variables, we removed the age variable and selected the characteristics with the rest of the variables. It should be noted that the age variable is considered during the analysis process. Figure 11 shows the characteristics selected by the Chi-square test. As we can see, the variables itching, delayed healing, and obesity should be removed because their p-value is more than 0.05. To check whether the response variable is balanced or not, we draw its graph (Figure 12).

According to Figure 12, we can see that the data is unbalanced, so it is necessary to balance it because unbalanced data can lead to biased model performance. Since the

majority class dominates the dataset, a model trained on unbalanced data tends to be biased toward predicting the majority class. As a result, the model's accuracy may appear high due to correctly predicting the majority class, while its performance on minority classes may be poor. There are different methods to balance the data, and we used SMOTE in this paper.

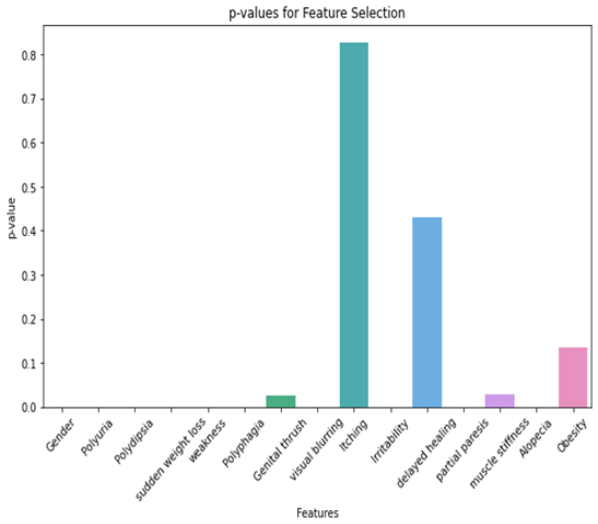

Figure 11: p-value for each feature in the Chi-square test

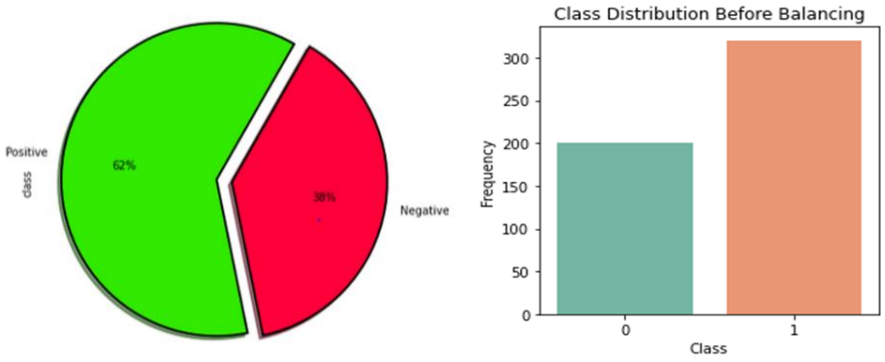

Figure 12: Response variable

SMOTE (Synthetic Minority Over-sampling Technique) is a popular algorithm that balances imbalanced datasets by generating synthetic samples for the minority class. It addresses the issue where the minority class is underrepresented compared to the majority class. Pay attention to Figure 13.

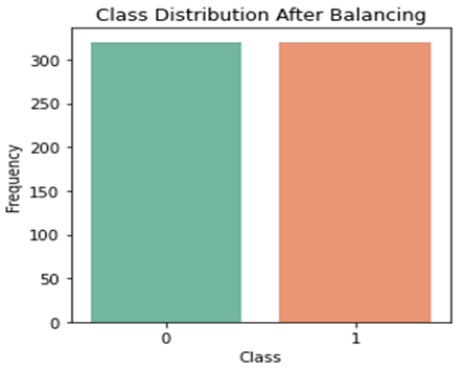

Figure 13: Response variable after balancing

We want to visualize the variables and get to know the features a little more.

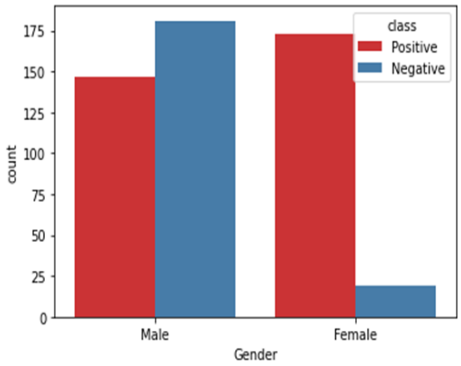

Figure 14: Gender versus class

As we can see, women have diabetes more than men.

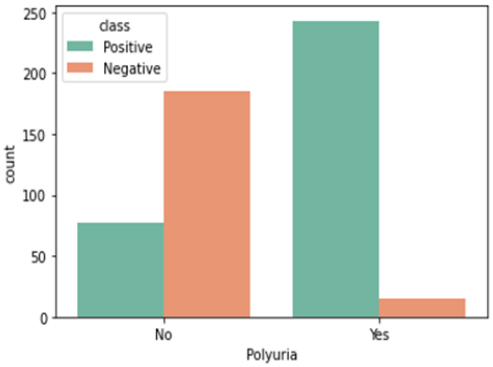

Figure 15: Polyuria versus class

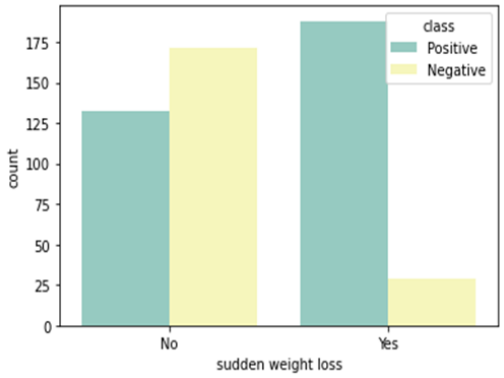

Figure 16: Sudden weight loss versus class

From Figure 16, we can see that people who have suddenly lost weight get sick more often.

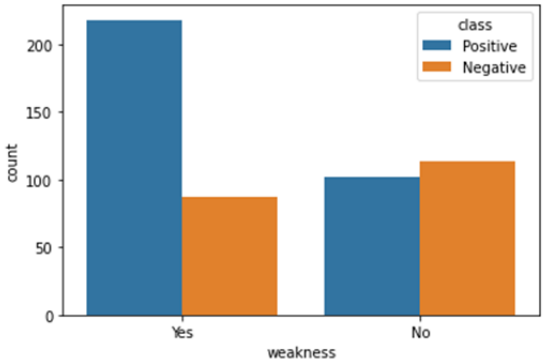

Figure 17: Weakness versus class

After preprocessing and visualizing the data, we now want to implement the mentioned models on the data and compare their accuracy with each other.

Table 3: Performance Comparison of the Algorithms

Algorithm	Accuracy	Precision	Recall	F1-score
CatBoost	0.93	0.94	0.93	0.93
AdaBoost	0.90	0.90	0.90	0.90
XGBoost	0.89	0.91	0.88	0.89
BSVM	**0.97**	0.97	0.97	0.97
BKNN	**0.95**	0.95	0.95	0.95
BDT	**0.96**	0.96	0.96	0.96

According to Table 3 and the analysis shown in the ROC curve of the algorithms (Figure 18), it can be concluded that Bayesian classification methods exhibit higher accuracy than boosting classification methods. This superior performance implies that

Bayesian algorithms are particularly reliable for predictive tasks, such as forecasting the results of new individuals' tests.

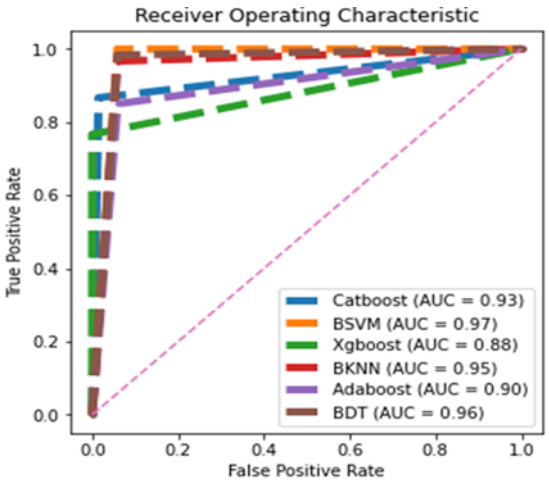

Figure 18: ROC Curves for the Algorithms [31]

Positive Aspects:

1. High Accuracy: Bayesian classification methods' higher accuracy means they can make more reliable predictions, reducing the likelihood of errors in test results. This high accuracy is especially beneficial in fields where precision is critical, such as in medical diagnostics or financial forecasting.
2. Robust Performance: The robustness of these algorithms leads to better generalization to new data, making them valuable in real-world applications where new data points frequently appear. This robustness ensures that the algorithms maintain performance even when encountering previously unseen data.
3. Decision Support: These methods can support decision-making processes in various fields. In medical diagnostics, for example, accurate predictions can lead to better patient outcomes by enabling earlier and more precise interventions.
4. Data Utilization: Bayesian methods effectively incorporate prior knowledge and update predictions as new data becomes available, enhancing their adaptability. This continuous learning process allows these algorithms to remain relevant and accurate over time.
5. Consistency: Consistent and precise predictions can build trust in automated systems among users and stakeholders. This trust is crucial for the broader acceptance and integration of these algorithms into critical decision-making processes.
6. Improved Predictive Accuracy: The higher accuracy of Bayesian methods can lead to more reliable predictions in practical applications, such as predicting patient outcomes or financial trends.
7. Increased Trust: The consistent performance of Bayesian methods can increase user confidence in automated systems, leading to greater adoption of these technologies.

These positive aspects highlight why Bayesian classification methods are particularly advantageous and why they can be confidently used to predict the test results of new

individuals. After concluding that Bayesian classification methods are more accurate than boosting classification methods, the question arises whether there is a statistically significant performance difference between the algorithms. To address this, we applied the Nemenyi Post Hoc statistical test, yielding the following results:

Based on the results of the Nemenyi test, it can be seen that BSVM, BKNN, and BDT show significant differences compared to the other algorithms.

Table 4 shows the accuracy of each algorithm and the algorithms with which it has statistically significant differences in performance, based on the Nemenyi post-hoc test.

Table 4: Nemenyi Post-Hoc test and accuracy

Algorithm	Accuracy (%)	Significant differences (p<0.05)
Catboost	93	Adaboost, XGboost, BSVM, BKNN, BDT
Adaboost	90	Catboost, XGboost, BSVM, BKNN, BDT
XGboost	89	Catboost, Adaboost, BSVM, BKNN, BDT
BSVM	97	Catboost, Adaboost, XGboost, BKNN, BDT
BKNN	95	Catboost, Adaboost, XGboost, BSVM, BDT
BDT	96	Catboost, Adaboost, XGboost, BSVM, BKNN

Figure 19 illustrates the heatmap of p-values from the Nemenyi post-hoc test, comparing the performance of various machine learning algorithms, including CatBoost, AdaBoost, XGBoost, BSVM, BKNN, and BDT. The color intensity represents the significance level of the differences between each pair of algorithms, with darker colors indicating lower p-values and thus more statistically significant differences. In this heatmap, we observe the following:

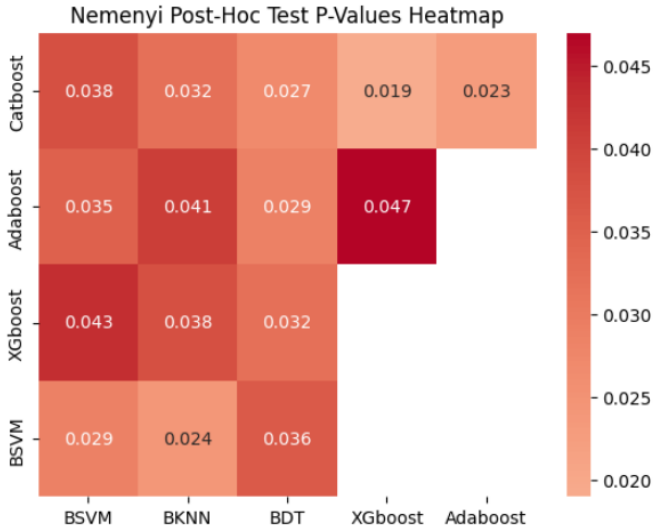

Figure 19: The pairwise statistical significance of differences between CatBoost, AdaBoost, XGBoost, BSVM, BKNN, and BDT based on the Nemenyi Post-Hoc test

CatBoost exhibits statistically significant differences when compared with BSVM, BKNN, BDT, XGBoost, and AdaBoost, as indicated by p-values ranging from 0.019 to 0.038.

AdaBoost also shows significant differences in performance with other models, particularly with XGBoost, where the p-value is the highest at 0.047, indicating a marginally significant difference.

XGBoost and BSVM display the most differences from other algorithms, with p-values ranging from 0.024 to 0.043, reinforcing the observation that these algorithms differ in terms of accuracy and other performance metrics.

This visualization effectively highlights the pairwise statistical differences between the algorithms, demonstrating which models have significantly different performance outcomes according to the Nemenyi test.

5. CONCLUSION

Since early diagnosis of diabetes and its treatment save people's lives, it is vital to use accurate methods with low error to classify people. Different methods and algorithms in the field of classical machine learning deal with classification, and the primary goal of all researchers is to use the methods that have the most accuracy, in line with this, different types of classification methods have been implemented. However in this article, Bayesian approaches are used to obtain higher accuracy for classification, and these methods are compared with boosting classification methods. So far, many types of research have shown that boosting methods have higher accuracy than classification methods. However, comparing boosting methods with Bayesian classification methods has not been made. In this study, by comparing boosting classification methods and Bayesian classification methods, we have shown that Bayesian classification methods have higher accuracy for classifying people because they take uncertainty into account when classifying. The future scope of this work includes expanding the dataset to include a more diverse population and exploring the integration of other advanced machine learning techniques, such as deep learning models, to further enhance diagnostic accuracy. Additionally, real-time implementation in clinical settings and the development of personalized treatment plans based on predictive models can be considered.

Funding: This research received no external funding

REFERENCES

[1] S. Arefin, "Chronic Disease Management through an AI-Powered Application,"*Journal of Service Science and Management,* vol. 17, no. 4, pp. 305-320, 2024.

[2] S. Abdollahi and R. Safa, "Machine learning and AI for advancing Parkinson's disease diagnosis: exploring promising applications," *Big Data and Computing Visions,* vol. 4, no. 1, pp. 12-21, 2024.

[3] M. Khalifa and M. Albadawy, "Artificial intelligence for diabetes: Enhancing prevention,diagnosis, and effective management," *Computer Methods and Programs in Biomedicine Update,* 100141, 2024.

[4] S.C. Mackenzie, C.A.R. Sainsbury, D.J. Wake, "Diabetes and artificial intelligence beyond the closed loop: a review of the landscape, promise and challenges," *Diabetologia,* vol. 67, no. 2, pp. 223-235, 2024.

[5] Khaleel, F.A., Al-Bakry, A.M.: "Diagnosis of diabetes using machine learning algorithms," *Materials Today: Proceedings 80*, 3200–3203 (2023).

[6] A. Choudhury and D. Gupta, "A survey on medical diagnosis of diabetes using machine learning techniques," *In: Recent Developments in Machine Learning and Data Analytics*, IC3, pp. 67–78, 2019.

[7] J.J. Khanam and S.Y. Foo, "A comparison of machine learning algorithms for diabetes prediction," *Ict Express,* vol. 7, pp. 432–439, 2021.
[8] P. Sonar and K. JayaMalini, "Diabetes prediction using different machine learning approaches," *In: 2019 3rd International Conference on Computing Methodologies and Communication, (ICCMC),* pp. 367–371, 2019.
[9] 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),*1, pp. 141–146, 2021.
[10] M. Roobini, M. Lakshmi, R. Rajalakshmi, L. Sujihelen and K. Babu, "Type 2 diabetes mellitus classification using predictive supervised learning model," *Soft Computing,* pp. 1–15, 2023.
[11] G. Rajput, G and A. Alashetty, "Diabetes classification using ml algorithms," *In: Inven tive Systems and Control: Proceedings of ICISC 2023*, pp. 867–877, Springer, 2023.
[12] H. Patel, H and J. Briskilal, "Prediction of Diabetes using Machine Learning Algorithm,"
[13] M.H.L. Louk and B.A. Tama, "Tree-based classifier ensembles for pe malware analysis: A performance revisit," *Algorithms,* vol. 15, no. 9, p.332, 2022.
[14] B.F. Wee, S. Sivakumar, K.H. Lim, W.K. Wong and F.H. Juwono, "Diabetes detection based on machine learning and deep learning approaches," *Multimedia Tools and Applications,* vol. 83, no. 8, pp. 24153-24185, 2024.
[15] Y. Resti, E.S. Kresnawati, N.R. Dewi and N. Eliyati, " Diagnosis of diabetes mellitus in women of reproductive age using the prediction methods of naive bayes, discriminant analysis, and logistic regression," *Science and Technology Indonesia*, vol. 6. No. 2, pp. 96–104, 2021.
[16] G. Parthiban, A. Rajesh and S.K. Srivatsa, " Diagnosis of heart disease for diabetic patients using naive bayes method," *International Journal of Computer Applications*, vol. 24, no. 3, pp. 7–11, 2011.
[17] C.Y. Chou, D.Y. Hsu and C.H. Chou, "Predicting the onset of diabetes with machine learning methods," *Journal of Personalized Medicine*, vol. 13, no. 3, p. 406, 2023.
[18] F. Ebrahimzadeh, and R. Safa, "Unlocking the Potential of the Metaverse for Innovative and Immersive Digital Care," *arXiv preprint arXiv:2406.07114*, 2024.
[19] A. Pourkeyvan, R. Safa, and A. Sorourkhah, "Harnessing the power of hugging face transformers for predicting mental health disorders in social networks," *IEEE Access*, vol. 12, pp. 28025–28035, 2024.
[20] J.H. Friedman, "Greedy function approximation: a gradient boosting machine," *Annals of statistics*, vol. 29, no. 5, pp. 1189–1232, 2001.
[21] J. Friedman, T. Hastie and R. Tibshirani, "Additive logistic regression: a statistical view of boosting (with discussion and a rejoinder by the authors)," *The annals of statistics,* vol. 28, no. 2, pp. 337–407, 2000.
[22] L. Mason, J. Baxter, P. Bartlett and M. Frean, "Boosting algorithms as gradient descent," *Advances in neural information processing systems*, vol. 12, 1999.
[23] L. Prokhorenkova, G. Gusev, A. Vorobev, A.V. Dorogush and A. Gulin, "CatBoost: unbiased boosting with categorical features," *Advances I n neural information processing systems*, vol. 31, 2018.
[24] Y. Freund, R. Schapire and N. Abe, "A short introduction to boosting," *Journal Japanese Society For Artificial Intelligence,* vol. 14, no. 5, pp. 771-780, 1999.
[25] T. Chen and C. Guestrin, "Xgboost: Reliable large-scale tree boosting system," *In: Proceedings of the 22nd SIGKDD Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA*, pp.13–17, 2015.
[26] N.G.Polson and S.L. Scott, "Data augmentation for support vector machines," *Bayesian Analysis*, vol. 6, no. 1, pp.1–24, 2011.
[27] G. Nuti, L.A.J. Rugama and A.I. Cross, "A Bayesian decision tree algorithm," *arXiv preprint arXiv:1901.03214*, 2019.

[28] C. Holmes and N. Adams, "A probabilistic nearest neighbor method for statistical pattern recognition," *Journal of the Royal Statistical Society Series B: Statistical Methodology,* vol. 64, no. 2, pp.295–306, 2002.
[29] Ž. Vujović, et al., "Classification model evaluation metrics," *International Journal of Advanced Computer Science and Applications,* vol. 12, no. 6, pp. 599–606, 2021.
[30] T. Fawcett, "Roc graphs: Notes and practical considerations for researchers," *Machine learning,* vol. 31, no. 1, pp. 1–38, 2004.