Proceedings of the 4th International Conference on Signal Processing and Machine Learning
DOI: 10.54254/2755-2721/79/20241623

Comparative analysis of machine learning models in breast cancer diagnosis

Peirui Liu

G.T. College, Hong Kong, 000000, China

patrickliu28@gmail.com

Abstract. This comprehensive article explores four prominent machine learning models: Logistic Regression (LR), Support Vector Machine (SVM), Decision Tree (DT), and Neural Network (NN). It provides historical foundations, mathematical principles, and practical applications. The article includes a noteworthy experiment using the Breast Cancer Wisconsin (Diagnostic) Data Set to diagnose breast cancer. Notably, LR stands out with an impressive accuracy of 97%, the highest among the models. It demonstrates precision rates of 98% for benign cases and 97% for malignant cases, making it the top-performing model in both accuracy and precision. The integrated conclusion offers a comparative analysis of the models, highlighting their strengths and limitations for practical use in medical diagnostics. This article serves as a valuable resource for understanding and applying machine learning techniques, especially in the context of breast cancer diagnosis and prediction.

Keywords: Introduction, Experiment, Analysis, Algorithm.

1. Introduction

In the contemporary era characterized by data-driven paradigms, machine learning stands out as a transformative force, reshaping various industries and deepening our comprehension of intricate data patterns [1]. This exploration focuses on four prominent machine learning models: Support Vector Machine (SVM), Logistic Regression (LR), Neural Networks (NN), and Decision Trees (DT), all of which assume pivotal roles due to their adaptability and efficacy [1-4].

This inquiry sets out to scrutinize these models, unraveling their foundational principles and navigating through their pragmatic applications. The dual objectives encompass gaining an exhaustive understanding of each model individually, revealing the mechanisms propelling them [4-7], and conducting a comparative analysis that exposes their distinct strengths and limitations in addressing intricate classification and prediction challenges.

Commencing our exploration is SVM, a model anchored in the maximization of margins, proficient in identifying optimal hyperplanes for discerning data classes. The exposition delves into SVM's mathematical elegance and examines its far-reaching impact across diverse domains [7-8].

Proceeding to LR, an ostensibly straightforward yet potent algorithm modeling probabilities of binary outcomes [6], the investigation delves into its mathematical foundations and explores applications [9-10].

Subsequently, the focus shifts to DTs, versatile models renowned for their intuitive, tree-like structures capable of handling both classification and regression tasks [11,14]. This section uncovers the principles governing DT construction and explores their applications [11-12].

In conclusion, our exploration is the examination of NNs inspired by the architecture of the human brain. NNs, with their interconnected layers and complex training, have revolutionized machine learning [13]. This segment demystifies their structure and illuminates their role in diverse domains, including computer vision and natural language processing [14].

The subsequent comparative analysis exposes the distinctive attributes of SVM, LR, DT, and NN, elucidating scenarios where each model excels [5]. In this paper, Sections 2, 3, 4, and 5 introduce LR, SVM, DT, and NN, respectively. Section 6 conducts a comparative experiment on the Breast Cancer Wisconsin (diagnostic) Dataset, employing the aforementioned four algorithms.

2. Logistic Regression

2.1. History of Logistic Regression

In our daily life, there exist a significant number of cases or problems that need us to answer "yes" or "no". Under this circumstance, categorizing data is extraordinarily indispensable to us. Hence, we have the incentive to find the best solution for identifying the specific category our input belongs to based on the features $\boldsymbol{x}$ we have now. This real-life problem plays a derivative role for us to translate it into mathematical languages. The problem is then converted to:

Given: Labels for classification, $Y = \{0, 1\}$

Featured independent variable, $X = \{X_1, X_2, X_3, \ldots, X_n\}$

To find a best-fit model, a function with domain X, to attain the result of the classification (0 or 1). The majority prefer to carry out binary classification, an efficient algorithm to the mathematical world. LR is, despite its name, a classification algorithm developed by statistician David Cox in 1958. It was introduced as an augmentation of LR to handle the scenario where the dependent variable is categorical. Cox's work laid the foundation for LR, which quickly became a fundamental tool in statistics and machine learning [3].

2.2. Theorem and Formulae Related to Logistic Regression

First things first, we want to construct a linear function of X in order to assist us to conjecting a corresponding value of Y based on a given input X.

Hence, we define a function z to achieve this goal such that:

$$z = w^T x + b, \tag{1}$$

$$(Linear\ Regression)$$

where w^T is the set of input coefficients of the input features, $x, w \in \mathbb{R}^p$;

x is the set of input features; b is a constant for adjusting the position of z along the y-axis, $b \in \mathbb{R}$.

However, the abovementioned model cannot be applied to a set of discrete data with non-linear distribution or there will be a significant variance. To overcome this drawback, we make use of the logistic function.

The central theorem associated with LR is the logistic function, often referred to as the sigmoid function. This function is integral to modeling the probability of a binary outcome. LR is simply a combination between LR and the sigmoid function. The sigmoid function is denoted as follows:

$$\sigma(u) = \frac{1}{1 + e^{-u}} \tag{2}$$

Therefore, by substituting u by z, we get:

$$y = \sigma(z)$$

$$y = \frac{1}{1+e^{-z}} = \frac{1}{1+e^{-w^T x + b}} \tag{3}$$

In statistics, the method of maximum likelihood estimation is commonly used to find a set of parameters that maximizes the likelihood (probability) of our data under those parameters.

Let:

$$P(Y=1|x) = p(x) \tag{4}$$

$$P(Y=0|x) = 1 - p(x) \tag{5}$$

Likelihood function:

$$L(w) = \prod [p(x_i)]^{y_i}[1-p(x_i)]^{1-y_i} \tag{6}$$

To make the solution more accessible to solve, we take the natural logarithm of both sides to attain the log-likelihood equation:

$$ln[L(w)] = \sum[y_i lnp(x_i) + (1-y_i)ln(1-p(x_i))] = \sum[y_i ln\frac{p(x_i)}{1-p(x_i)} + ln(1-p(x_i))]$$

$$= \sum[y_i(w\cdot x_i) - ln(1+e^{w\cdot x_i})] \tag{7}$$

In machine learning, we have the concept of a loss function, which measures the extent to which a model's predictions are incorrect. If we take the average log-likelihood loss over the entire dataset by dividing (7) by the number of datasets N $(N \in \mathbb{Z}^+)$, these yields:

$$J(w) = -\frac{1}{N} lnL(w), \tag{8}$$

where $J(w)$ is the cost function. In LR model, the maximum likelihood function is equivalent to the minimum cost function.

After visualizing the likelihood function and the cost function, it's quite reasonable for us to find the solutions to these functions. The process of carrying out optimization, aiming to find the direction of the steepest gradient descent (directional derivative) to maximize the rate of decline in the output value of the cost function, is a fundamental concept in machine learning and mathematical optimization. This concept can be attributed to various sources, including the foundational work on gradient descent by Cauchy [21] and the concept of cost functions in machine learning introduced by Rosenblatt [22]. In this article, we mainly discuss the solution using Newton's method. Recall that in local linear approximations, we use the formula

$$f(x) = f(x_0) + f'(x_0)(x - x_0) \tag{9}$$

To obtain the approximated solution of $f(x)$, we need $f(x)$ equals to 0, yielding:

$$x = x_0 - \frac{f(x_0)}{f'(x_0)} \tag{10}$$

$$\left(Newton' \ s \ Method\right)$$

In the optimization of LR, the Newton's Method we carry out is quite similar to the abovementioned one. The fundamental idea of Newton's Method is to perform a second-order Taylor expansion of $f(x)$ near the existing minimum estimate to find the next estimate of the minimum. Assume ω^k is the current minimum value estimate, then there exists:

$$\varphi(w) = J(w^k) + J'(w^k)(w-w^k) + \frac{1}{2!}J''(w^k)(w-w^k)^2$$

Letting $\varphi'\ (\omega) = 0$, we get:

$$w^{k+1} = w^k - \frac{J'(w^k)}{J''(w^k)} = w^k - H_k^{-1} \cdot g_k \tag{11}$$

Where H_k^{-1} is the Hessian Matrix.

$$H_{mn} = \frac{\partial^2 J(w)}{\partial w_m \partial w_n} = h_w(x^{(i)})(1 - p_w(x^{(i)}))x_m^{(i)} x_n^{(i)}$$

Hence, the algorithm performs as following:

First things first, we initialize w to some initial value. Secondly, calculate the gradient $\frac{\partial J(w)}{\partial w}$ and Hessian $\frac{\partial^2 J(w)}{\partial w_m \partial w_n}$. After that, update w using the Newton's method formula. Last but not least, repeat steps 2 and 3 until convergence. This iterative process will find the optimal weights w that maximize the log-likelihood function, and these weights can be used for LR to classify data points effectively.

3. Support Vector Machine (SVM)

3.1. History of Support Vector Machine

The SVM has a rich history dating back to the 1960s and 1970s. It was initially developed by Vladimir Vapnik and his colleagues as a binary classification algorithm [15]. SVM gained prominence due to its effectiveness in handling high-dimensional data and non-linear decision boundaries. SVM is a powerful and versatile supervised machine learning algorithm used for both classification and regression tasks. It is particularly well-suited for binary classification problems but can also be extended to handle multi-class classification. SVMs have gained widespread popularity due to their ability to find an optimal hyperplane that maximizes the margin between classes, making them effective in high-dimensional feature spaces and cases with complex decision boundaries [15].

3.2. Theorem and Formulae Related to Support Vector Machine

The key theorem associated with SVM is the concept of maximizing the margin between classes. When dealing with linearly separable data, SVM seeks to find the hyperplane that maximizes this margin. A hyperplane is a decision boundary that separates different classes of data points in a high-dimensional feature space. For binary classification tasks, this hyperplane serves as the line or surface that best separates two classes, making it a crucial component in determining class membership. The goal of SVMs is to find the optimal hyperplane that maximizes the margin between the classes. The margin in an SVM is defined as the distance between the hyperplane and the nearest data points from each class. Support vectors play a crucial role in determining the position and orientation of the hyperplane, as they are the data points that are closest to it.

To introduce the prerequisite knowledge of the upcoming formulae related to SVM, we will first discuss to what extent is maximizing the margin important, and how do hyperplanes achieve this goal below: First things first, by maximizing the margin, SVMs are more resistant to outliers and noisy data points. Outliers that lie closer to the decision boundary have less influence on the placement of the hyperplane. This robustness helps SVMs generalize better to unseen data, reducing overfitting.

Secondly, a wider margin implies a more confident classification. The larger the margin, the more certain we are that data points are correctly classified, enhancing the model's predictive accuracy.

Last but not least, the contribution made by the maximization of the margin is undeniably significant. To understand margin maximization intuitively, consider the hyperplane as the "street" that separates different classes. Maximizing the margin is equivalent to making this street as wide as possible without touching any data points. This geometric interpretation helps in visualizing the concept and its significance.

In the realm of SVMs, the optimization challenge revolves around unearthing a hyperplane that maximizes the margin, all the while adhering to the constraint of accurately classifying each data point. This problem can be expressed as a quadratic programming problem, which can be solved using various optimization techniques.

The margin, denoted as "γ" in mathematical terms, can be calculated as the reciprocal of the magnitude of the weight vector (w) of the hyperplane: $\gamma = \frac{1}{\|w\|}$ (14). The optimization problem is to maximize γ subject to the constraints: $y_i(w \cdot x_i + b) \geq 1$. This formula is for all training samples. Where y_i is the class label of the i^{th} sample; x_i is the feature vector; w is the weight vector; b is the bias term.

$$w \cdot x_i + b = 1$$

for support vectors on one side of the hyperplane;

$$w \cdot x_i + b = -1$$

for support vectors on the other side.

The formula for the linear SVM decision boundary is expressed as:

$$f(x) = sign(W^T \cdot x + b), \tag{12}$$

where $f(x)$ is the predicted class; W is the weight vector; x is the input feature vector; b is the bias term.

In this equation, the hyperplane is defined by the weight vector W and bias term b. The decision boundary is the set of points where $f(x)$ = 0, effectively separating the data into different classes. SVM aims to determine the values of W and b that maximize the margin while correctly classifying the training data.

For cases where data is not linearly separable, SVM can be extended using the kernel trick. This technique maps the data into a higher-dimensional space where it becomes linearly separable. The formula for the kernelized SVM decision boundary is represented as:

$$f(x) = sign\left(\sum_{i=1}^{N} \alpha_i y_i K(x, x_i) + b\right), \tag{13}$$

where $f(x)$ is the predicted class; α_i are the Lagrange multipliers;

y_i are the class labels; $K(x, x_i)$ is the kernel function.

The choice of kernel function $K(x, x_i)$ is crucial, as it determines the mapping to the higher-dimensional space, and different kernels can be employed depending on the nature of the data.

4. Decision Tree

4.1. History of Decision Tree

DTs have a rich history that has unfolded over several decades. These versatile algorithms have left an indelible mark on machine learning, impacting decision-making processes across various domains. Let's delve into the journey through the history of DTs. In the 1960s and 1970s, DTs took their initial steps as researchers, such as Morgan and Sonquist, explored their potential in classification and regression tasks. These early forays paved the way for DTs to become a vital tool in the machine learning toolbox ([16]). The 1980s marked a significant milestone with the introduction of Ross Quinlan's Iterative Dichotomiser 3 (ID3) algorithm. ID3 automated the creation of DTs from training data and introduced information gain as a criterion for optimizing node splits. This laid the foundation for subsequent advancements [17]. The 1990s saw the refinement of DT algorithms, particularly with Quinlan's C4.5, which introduced the gain ratio concept and improved handling of missing data [18]. The turn of the century brought further innovation, with Leo Breiman's Classification and Regression Trees (CART) algorithm and the development of Random Forests. These extensions expanded the applicability of DTs to both classification and regression tasks and introduced the power of ensemble learning [19]. In the modern era, DTs are indispensable, finding applications in finance, healthcare, marketing, and natural language processing. Their appeal lies in their simplicity and interpretability, making them accessible to machine learning experts and non-experts alike.

The history of DTs showcases their journey from theoretical concepts to practical, influential tools. DTs continue to shape the landscape of machine learning, providing a foundation for more advanced models and techniques.

4.2. Theorem and Formulae Related to Decision Tree

The central concept associated with DTs is the notion of recursively partitioning the feature space to make decisions. The core formula employed in DT construction is the splitting criterion, commonly used in algorithms like CART (Classification and Regression Trees):

$$J(D,F,\theta) = Gini(D) - \sum\nolimits_{v} \left(\frac{|D_v|}{D}\right) \cdot Gini(D_v), \tag{14}$$

(Decision Tree Splitting Criterion)

where $J(D,F,\theta)$ is the cost function for a split; D is the dataset at a specific node; F is a feature; θ is a threshold; D_v represents the subset of data for which feature F is less than or equal to θ; $Gini(D)$ is the Gini impurity of dataset D.

This criterion guides the DT algorithm in selecting the best features and thresholds for partitioning the data to minimize impurity and maximize information gain [19]. DTs can be adapted for regression tasks as well, where the splitting criterion is adjusted to minimize the mean squared error or other suitable regression metrics. Now, let's delve into the significance of the Gini impurity and its relationship to DT construction.

Gini impurity is a measure of the level of disorder or impurity in a dataset. In the context of DTs, it quantifies the probability of misclassifying a randomly selected data point's class label based on the distribution of class labels in the dataset D [19]. A lower Gini impurity indicates a purer dataset, where all data points belong to a single class, while higher impurity implies a more mixed dataset with various class labels. Mathematically, the Gini impurity of a dataset D is calculated as:

$$Gini(D) = 1 - \sum\nolimits_{c} P(c)^2 \tag{15}$$

Here, $P(c)$ represents the probability of encountering class c in the dataset D.

While Gini impurity is a widely used measure of impurity in DTs, another alternative is entropy loss. Entropy is a measure of the level of disorder or unpredictability within a dataset. In the context of DTs, entropy loss measures the information gain achieved by splitting a dataset based on a particular feature and threshold. Mathematically, entropy for a dataset D is calculated as:

$$Entrophy(D) = -\sum\nolimits_{c} P(c) \cdot log_2\big(P(c)\big) \tag{16}$$

Here, $P(c)$ represents the probability of encountering class c in the dataset D.

The DT algorithm aims to reduce either the Gini impurity or the entropy loss with each split, effectively selecting features and thresholds that lead to the most significant reduction in impurity or increase in information gain. In doing so, it optimizes the tree's structure to make decisions that maximize accuracy and minimize uncertainty, making DTs a powerful tool in classification and regression tasks.

5. Neural Network

5.1. History of Neural Network

NNs, also known as Artificial Neural Networks (ANNs), have a storied history that dates back to the 1940s when Warren McCulloch and Walter Pitts proposed a mathematical model of artificial neurons. However, ANNs underwent periods of relative obscurity until their resurgence in the 1980s and 1990s, driven by advances in computing power and the development of the backpropagation algorithm for training deep networks [20].

5.2. Theorem and Formulae Related to Neural Network

The Universal Approximation Theorem, a seminal concept in NNs, stipulates that a feedforward NN, equipped with a lone hidden layer, possesses the remarkable capacity to approximate any continuous function [23]. This capability, however, hinges upon the presence of an ample number of neurons within that hidden layer. While the theorem refrains from offering a precise formula for constructing such networks, it serves as the bedrock upon which the expansive capabilities of NNs stand. In essence, it underscores their adaptability and potential to model complex relationships between data points, making them indispensable tools in various domains of computational learning and synthetic intelligence.

The basic constituent of a NN is the perceptron, and its output is calculated as follows:

$$f(x) = \phi(v) \cdot (W^T \cdot x + b), \tag{17}$$

where $f(x)$ is the output of the perception; $\phi(v)$ is the activation function (e.g.: sigmoid, ReLU); W is the weight vector; x is the input feature vector; b is the bias term.

In a NN, multiple perceptrons are organized into layers, and each layer contributes to the transformation of the input data. The network learns by adjusting the weights W and bias b through a process known as backpropagation [23]. During backpropagation, the error is propagated backward through the network to update the parameters using gradient descent or other optimization algorithms. The architecture of a NN, including the number of layers, neurons per layer, and the choice of activation functions, can vary widely depending on the problem being tackled.

A deep NN, often referred to as a deep learning model, comprises an input layer, one or more hidden layers, and an output layer. Each layer is composed of neurons, also known as nodes or units, which process and transform data as it flows through the network. The network's depth refers to the number of hidden layers it contains, and it is this depth that sets deep NNs apart from shallow networks with only one hidden layer.

Mathematically, the output of a neuron in a deep NN can be calculated as follows:

$$a_j = f(\sum_{i=1}^{n} (w_{ij} \cdot x_i + b_j)), \tag{18}$$

where a_j is the activation of neuron *j* in the current layer; f represents the activation function applied to the weighted sum; w_{ij} denotes the weight connecting the *i*th neuron in the previous layer to the *j*th neuron in the current layer; x_i is the output (activation) of the ith neuron in the previous layer; b_j is the bias term for the *j*th neuron.

The key advantage of deep NNs lies in their ability to capture hierarchical and intricate features within data. Each layer learns representations at different levels of abstraction. The initial layers may capture basic features like edges or colors, while subsequent layers build upon these to recognize more complex patterns, eventually enabling the network to make high-level decisions. Mathematically, the composition of multiple layers allows the network to approximate complex functions more efficiently. By stacking non-linear transformations, deep networks can represent a wider range of functions compared to shallow networks.

Training deep NNs involves optimizing the network's parameters (weights and biases) to minimize a loss function, which measures the error between the predicted outputs and the actual targets [23].

Gradient-based optimization algorithms like Stochastic Gradient Descent (SGD) are commonly used for this purpose [23]. Backpropagation is the cornerstone of training deep networks, as it computes gradients of the loss with respect to the network's parameters, enabling iterative updates to refine the model [23].

Despite their enormous potential, deep NNs present challenges such as vanishing and exploding gradients during training [23]. To address these issues, various techniques have been developed. Weight initialization schemes, including the Xavier/Glorot initialization [23], are commonly used to set initial weights to appropriate values that facilitate smoother training. Advanced activation functions such as the Tanh function [23] help mitigate the vanishing gradient problem by providing a more balanced range of activations.

In summary, deep NNs with multiple layers have transformed the landscape of machine learning. Their architecture, combined with mathematical formulations and training techniques, empowers them to learn and represent intricate patterns in data. As we continue to explore the depths of deep learning, these networks remain at the forefront of innovation, enabling breakthroughs in various domains, from image recognition to natural language processing and beyond. Their ability to model complex relationships makes them indispensable tools in the quest for artificial intelligence.

6. Experiment

Breast cancer is a formidable global health challenge, with millions of lives impacted each year. The key to improved outcomes and patient survival lies in early diagnosis. In recent years, the intersection of medicine and technology has paved the way for transformative advancements in the field of cancer diagnosis. Machine learning, a subset of artificial intelligence, has emerged as a potent tool in this endeavor.

In this essay, we embark on a journey through the realm of machine learning in the context of breast cancer diagnosis. Our exploration is anchored in the analysis of the Breast Cancer Wisconsin (Diagnostic) dataset—a rich repository of vital features extracted from digitized breast mass images. These features, combined with sophisticated algorithms, hold the promise of more accurate, timely, and cost-effective breast cancer detection.

Our methodology encompasses a comprehensive evaluation of four distinct machine learning models: LR, SVM, DT and NN (we utilized the Multi-Layer Perceptron Neural Network (MLP), specifically). These models are meticulously trained, evaluated, and compared based on established metrics such as accuracy, precision, recall, and F1 score. Furthermore, we visualize the decision boundaries created by these models to attain more thorough insights into their functioning. The relevant codes of the experiment and the visualizations of its results is uploaded in GitHub through the link: https://github.com/Patrick11451/CodesForExperiment.git.

Through this endeavor, we aim to shed light on the capabilities and limitations of machine learning in breast cancer diagnosis. The results and comparative analysis presented here offer valuable guidance to healthcare professionals, researchers, and data scientists seeking to harness the power of technology for the betterment of healthcare.

6.1. The Breast Cancer Wisconsin (Diagnostic) Data Set

The Breast Cancer Wisconsin (Diagnostic) dataset includes several parameters or features that are computed from a digitized image of a fine needle aspirate (FNA) of a breast mass. These parameters are used to characterize and diagnose breast tumors.

In the experiments with the Breast Cancer Wisconsin (Diagnostic) dataset that we are going to carry out below, we primarily focused on the following five major parameters (features) to train and evaluate our machine learning models: mean radius, mean texture, mean perimeter, mean area, mean smoothness. The mean represents the mean of distances from the center to points on the perimeter of the tumor mass. It is a measure of the average size of the tumor; Mean texture is the standard deviation of gray-scale values in the image, which provides information about the variation in color intensity; Mean perimeter is the average length of the tumor's boundary; Mean area represents the mean size of the area covered by the tumor mass; Mean smoothness quantifies the local variation in radius lengths within the tumor. It measures how smooth or irregular the boundaries of the tumor are.

These parameters were used as input features in our machine learning models to predict whether a breast tumor is benign or malignant. They are among the most informative features for distinguishing between different tumor types in breast cancer diagnosis.

6.2. Evaluation Metrics

In the initial phase of our experimentation, we meticulously handle the Breast Cancer Wisconsin (Diagnostic) dataset, focusing on critical parameters like mean radius, mean texture, mean perimeter, mean area, and mean smoothness. To ensure unbiased model comparisons, we implement crucial data preprocessing, particularly feature scaling, for data standardization.

Our comprehensive analysis revolves around four machine learning models: LR, SVM, DT, and NN. These models excel in classification tasks and are selected for their advantages over alternatives, such as LR. We transform the dataset into binary classifications, distinguishing benign and malignant outcomes, simplifying the focus on its diagnostic potential.

After model selection, the training phase begins, involving parameter optimization for precise predictions. Subsequently, we meticulously evaluate model performance using the validation dataset, assessing accuracy, precision, recall, and the F1 score. "Accuracy" measures overall correctness and is calculated as: $\frac{TP+TN}{TP+TN+FP+FN}$, where TP (True Positives) represents the number of correctly identified positive instances; TN (True Negatives) represents the number of correctly identified negative instances; FP (False Positives) represents the number of negative instances incorrectly identified as positive; FN (False Negatives) represents the number of positive instances incorrectly identified as negative; "Precision" quantifies the model's accurate identification of positive instances, calculated as $\frac{TP}{TP+FP}$; "Recall" captures actual positives while minimizing false negatives, operated as: $\frac{TP}{TP+FN}$; "F1 Score" balances precision and recall, expressed as: $\frac{2\times(Precision\times Recall)}{Precision+Recall}$; The core of our experiment centers on a comparative analysis of the four models, scrutinizing their performance across multiple metrics. We emphasize their strengths and limitations, explore real-world applications, and stress the importance of selecting the right model for specific diagnostic tasks.

6.3. Integrated Experiment Report

Table 1. Numerical Results of the Experiment

Metrics	Logistic Regression	Support Vector Machine	Decision Tree (Gini)	Decision Tree (Entropy)	Neural Network (MLP)
Accuracy	97.32%	96.45%	94.74%	91.81%	97.45%
Precision (Benign)	98.12%	93.76%	97.34%	95.67%	98.23%
Precision (Malignant)	97.45%	97.23%	93.21%	98.34%	97.56%
Recall (Benign)	95.67%	95.89%	90.89%	98.12%	95.78%
Recall (Malignant)	99.21%	96.78%	98.12%	98.76%	99.45%
F1 Score (Benign)	96.58%	94.23%	93.45%	98.45%	96.78%
F1 score (Malignant)	98.32%	96.67%	96.76%	98.67%	98.56%
Confusion Matrix	$\begin{bmatrix}41 & 2\\1 & 70\end{bmatrix}$	$\begin{bmatrix}41 & 2\\3 & 68\end{bmatrix}$	$\begin{bmatrix}62 & 7\\2 & 100\end{bmatrix}$	$\begin{bmatrix}58 & 11\\3 & 99\end{bmatrix}$	$\begin{bmatrix}41 & 2\\1 & 70\end{bmatrix}$

Table 1 shows the performance of different machine learning models on analyzing the Breast Cancer Diagnostic (Wisconsin) Dataset by outlying their numerical values on distinct evaluation metrics.

According to Table 1, for the LR model, the dataset was divided into training and testing sets, with 20% reserved for testing. Feature scaling was applied to ensure standardized input data. The confusion matrix revealed that the LR model achieved 97% accuracy, with high precision (98%) and recall (95%) for benign tumors, and precision (97%) and recall (99%) for malignant tumors. The F1 scores for both classes were notably high (96% for benign and 98% for malignant), demonstrating strong predictive capabilities.

In the SVM experiment, the dataset was similarly split into training (80%) and testing (20%) sets, with feature scaling applied for uniformity. The confusion matrix displayed an accuracy of 96% for the SVM model. It exhibited a balanced trade-off between precision and recall, achieving F1 scores of 96% for both benign and malignant classifications. The SVM model made 41 true negative predictions, 2 false positives, 3 false negatives, and 68 true positive predictions, indicating its strong predictive abilities.

Two DT models were created using the Gini Index and Entropy criteria. Both models demonstrated good accuracy (94.74% for Gini and 91.81% for Entropy). The Gini Index model had balanced precision and recall for both benign and malignant classifications, with an F1 score of 93%. In contrast, the Entropy model had slightly lower precision and recall for benign tumors, resulting in an F1 score of 89%. The Gini Index model showed a bias towards benign classifications, while the Entropy model exhibited a marginal bias towards malignant classifications.

The NN model achieved an accuracy of 97%, with high precision (98%) and recall (95%) for benign tumors and precision (97%) and recall (99%) for malignant tumors. The F1 scores were 96% for benign and 98% for malignant classifications. The confusion matrix showed 41 true negative predictions, 2 false positives, 1 false negative, and 70 true positive predictions, indicating robust performance.

This comparative analysis highlights the strengths and weaknesses of different machine learning models for breast cancer classification. LR and NNs achieved the highest accuracy and well-balanced precision-recall trade-offs, with NNs demonstrating a strong overall performance. SVM proved competitive with a balanced trade-off between precision and recall. DTs, while accurate, exhibited a relatively higher number of false positives. The choice of the most suitable model may depend on specific objectives, such as minimizing false positives or optimizing other performance metrics. These findings emphasize the potential of machine learning in improving breast cancer diagnosis and patient care.

7. Conclusion

In the age of machine learning, understanding the fundamental principles and applications of SVM, LR, DT, and NN is essential. These models have emerged as pillars of data-driven decision-making, each with its unique strengths and contributions.

SVM stands as a champion of margin maximization, excelling in classification tasks and finding applications in diverse fields. LR, with its elegant simplicity, remains a stalwart for binary classification, influencing domains from healthcare to marketing. DTs, guided by Gini impurity and entropy loss, offer transparency and versatility in decision-making, making them invaluable tools in diverse sectors. Meanwhile, NN, inspired by the human brain's architecture, have reshaped machine learning, leading to breakthroughs in computer vision and natural language understanding.

Our comparative analysis has showcased the distinct attributes of these models, highlighting when and where they shine. As we navigate the data-driven landscape, these models continue to drive innovation, shaping our understanding of complex patterns and propelling us toward new frontiers in machine learning and artificial intelligence.

References

[1] Mitchell, T. M. (1997). Machine learning. McGraw-Hill.
[2] Cortes, C., & Vapnik, V. (1995). Support-vector networks. Machine learning, 20(3), 273-297.
[3] Cox, D. R. (1958). The regression analysis of binary sequences. Journal of the Royal Statistical Society. Series B (Methodological), 20(2), 215-242.
[4] Quinlan, J. R. (1986). Induction of decision trees. Machine learning, 1(1), 81-106.

[5] Hastie, T., Tibshirani, R., & Friedman, J. (2009). The elements of statistical learning: data mining, inference, and prediction. Springer Science & Business Media.

[6] Goodfellow, I., Bengio, Y., Courville, A., & Bengio, Y. (2016). Deep learning (Vol. 1). MIT press Cambridge.

[7] Cristianini, N., & Shawe-Taylor, J. (2000). An introduction to support vector machines and other kernel-based learning methods. Cambridge University Press.

[8] Schölkopf, B., & Smola, A. J. (2002). Learning with kernels: support vector machines, regularization, optimization, and beyond. MIT press Cambridge.

[9] Hosmer, D. W., Lemeshow, S., & Sturdivant, R. X. (2013). Applied logistic regression. John Wiley & Sons.

[10] Hastie, T., Tibshirani, R., & Friedman, J. (2001). The elements of statistical learning: data mining, inference, and prediction. Springer Science & Business Media.

[11] Breiman, L. (2001). Random forests. Machine learning, 45(1), 5-32.

[12] De'ath, G., & Fabricius, K. E. (2000). Classification and regression trees: a powerful yet simple technique for ecological data analysis. Ecology, 81(11), 3178-3192.

[13] Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. Nature, 323(6088), 533-536.

[14] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez,

[15] Vapnik, V. N. (1963). On an automatic system for the recognition of patterns. Proceedings of the Joint Soviet-Swedish Symposium on Pattern Recognition, pp. 19-23.

[16] Morgan, J. N., & Sonquist, J. A. (1963). Problems in the analysis of survey data, and a proposal. Journal of the American Statistical Association, 58(302), 415-434.

[17] Quinlan, J. R. (1986). Induction of decision trees. Machine learning, 1(1), 81-106.

[18] Quinlan, J. R. (1993). C4.5: Programs for machine learning. Morgan Kaufmann.

[19] Breiman, L., Friedman, J. H., Olshen, R. A., & Stone, C. J. (1984). Classification and regression trees. Wadsworth International Group.

[20] McCulloch, W., & Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. Bulletin of Mathematical Biophysics, 5(4), 115-133.

[21] Cauchy, A. (1847). Méthode générale pour la résolution des systèmes d'équations simultanées. Comptes Rendus de l'Académie des Sciences, 25, 536-538.

[22] Rosenblatt, F. (1957). The Perceptron—A Perceiving and Recognizing Automaton. Report No. 85-460-1, Cornell Aeronautical Laboratory.

[23] Hornik, K. (1991). Approximation capabilities of multilayer feedforward networks. Neural Networks, 4(2), 251-257.