Analysis of classification algorithms: Insights from MNIST and WDBC datasets

Jiyue Zhao[1,6,*], Tony Yuxiang Pan[2,7], Weibo Yao[3,8], Hongwei Lu[4,9], Zihan Liu[5,10]

[1]Beijing National Day School, Beijing 100039, China
[2]Southlands Schools International, Qingdao Weiming School, Qingdao, Shandong 266400, China
[3]Beijing New Oriental Yangzhou Foreign Language School, Yangzhou, Jiangsu 225006, China
[4]Chengdu University, Chengdu 610000, China
[5]Emory University, Atlanta 30322, Georgia, United States

[6]zhaojiyue2@gmail.com
[7]TonyPan072006@163.com
[8]1019849647@qq.com
[9]2779379954@qq.com
[10]Laujason498@gmail.com
*corresponding author

Abstract. Various classification algorithms applied to sophisticated datasets have seen significant development over the years, which involves dealing with the growing complexities of real-world data and providing efficient solutions for numerous domains like healthcare and data analysis. There is a critical need to identify the most effective algorithms to deliver high precision and generalizability. This study intends to assess diverse models, including Support Vector Machine (SVM), Multi-Layer Perceptron (MLP), DTs (DT), and Random Forests (RF), on Modified National Institute of Standards and Technology (MNIST) and Wisconsin Diagnostic Breast Cancer (WDBC) datasets, utilizing metrics like Overall Accuracy (OA), Average Accuracy (AA), and Cohen's kappa. The study has shown that the performance of the algorithms is mainly determined by the dataset's features. Additionally, insights into the strengths and limitations of each model are provided.

Keywords: Classification algorithms, Machine Learning, Deep Learning, Applications.

1. Introduction

In the area of machine learning, the classification problem holds significant importance, enabling us to categorize and organize complex data into different groups. The process involves assigning labels or categories to data points based on their features or attributes [1,2]. As datasets grow in complexity, particularly like the MNIST and WDBC datasets, the need for accurate classification algorithms becomes crucial.

The extensive range of traditional classification algorithms further increases the complexity of choosing the most suitable one for a particular dataset and problem. With various options available, selecting the most appropriate algorithm requires careful consideration of different factors such as data structure, dimensionality, and the nature of the problem. Moreover, the choice of an algorithm significantly impacts the accuracy and efficiency of the classification process, making it become a more and more crucial decision in solving classification problems.

For this report, an extensive analysis was conducted to evaluate the performance of diverse classification algorithms, including Support Vector Machine (SVM) with linear, polynomial, and Radial Basis Function (RBF) kernels, MLP, Decision Trees (DT), and Random Forest (RF) [3-7]. The experiments involved an in-depth examination of the algorithms' capabilities in accurately categorizing the complex data points within the two datasets, focusing on assessing the robustness and generalizability of these models in applications.

This report's structure is arranged as follows. The following section provides a comprehensive insight into the mathematical foundations of each of the chosen algorithms, emphasizing their unique attributes and theoretical applications. In Section 3, we introduce the MNIST and WDBC datasets and discuss the computation of key metrics, including OA, AA, and kappa. We then present the experimental results for each model. Subsequently, we conduct a comprehensive analysis comparing the results, explaining individual advantages and drawbacks of the models and giving suggestions for choosing a suitable classification model that satisfies the specific requirements and constraints of the given application domain.

2. Methodology

The classification problem is one of the supervised learning problems, which can be understood as assigning data points to distinct categories or classes based on their features [1,2,8]. There are four main classification tasks in Machine learning: binary, multi-class, multi-label, and imbalanced classifications [9]. By providing a detailed analysis of the mathematical foundations, strengths, and limitations of these algorithms, this report seeks to offer profound insights into the complexities of classification problems and the process of algorithm selection. We handled different classification problems with different datasets, and we utilized various algorithms, specifically SVM, MLP, DT, and RF, to analyze and give results separately and give insight into the election of models.

2.1. Support Vector Machine (SVM)

2.1.1. Linearly Separable Classification. We are given a dataset of n samples, each has p inputs and an output y, which can be written in the form of

$$(x_1, y_1), (x_2, y_2) \dots (x_n, y_n) \tag{1}$$

$$x_i =, \left(x_{i1}, x_{i2}, \dots, x_{ip}\right)^T, i = 1, 2, \dots, n \tag{2}$$

and the y_i is either 1 or -1, specifying the categories to which x_i pertains.

For example, Figure 1 is a 2-dimensional space, there are two datasets, for each data, $p = 2$, which are represented by the x_1 and x_2 axes. The "linear function", or the hyperplane, here is a line. In n-dimensional space, it will be a hyperplane with $n - 1$ dimension, which can be represented by: $w_1x_1 + w_2x_2 + \cdots w_nx_n + b = 0$. In a matrix equivalently expressed as $W^TX + b = 0$, $W^T = (w_1, w_2, \dots, w_n\ )$, where w_i is the corresponding coefficient of x_i. The margin is decided by support vectors, for example, in Figure 1, support vectors are those points on the dotted lines. The solid line in the middle is the hyperplane, the optimal hyperplane is decided by support vectors. In a 2-dimensional plane, the distance between the point (x_0, y_0) and line $Ax + By + C = 0$ is expressed as: $d = \frac{|Ax_0 + By_0 + C|}{\sqrt{A^2 + B^2}}$. With hyperplane $W^TX + b = 0$ and point x_i, where $\|W\| = \sqrt{w_1{}^2 + w_2{}^2 + \cdots + w_n{}^2}$, $d_i = \frac{|W^Tx_i + b|}{\|W\|_2}$, $\|W\|_2 \Leftrightarrow \|W\|$.

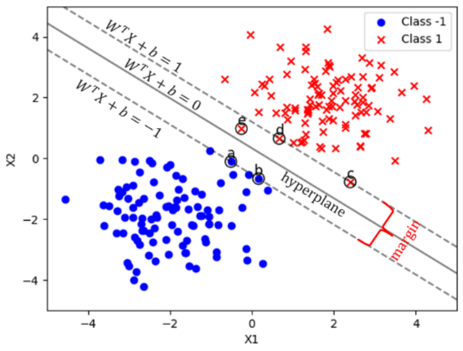


Figure 1. Linear SVM Model. Two classes were classified

Let $W^TX + b = +1$ denotes the line parallel to the middle line while passing through the support vectors in the class with $y_i = 1$, and $W^TX + b = -1$ denotes the similar line in the class with $y_i = -1$. (1 and -1 are easy for calculation because we want W and b, multiplying the equation doesn't change the result). Then,

$$\begin{cases} W^Tx_i + b \geq 1 \ \ y_i = 1 \\ W^Tx_i + b \leq -1 \ \ y_i = -1 \end{cases} \Longrightarrow y_i(W^Tx_i + b) \geq 1, \tag{3a}$$

$$|y_i| = 1 \Longrightarrow d_i = \frac{|W^Tx_i + b|}{\|W\|} = \frac{y_i(W^Tx_i + b)}{\|W\|}, \tag{3b}$$

$$2d_i = 2 \cdot \frac{y_i(W^Tx_i + b)}{\|W\|}. \tag{3c}$$

For support vectors, $2d = \frac{2}{\|W\|}$, to maximize the margin, $max\frac{2}{\|W\|} \Leftrightarrow min\|W\| \Leftrightarrow min\frac{\|W\|^2}{2}$. We need to make sure that

$$\frac{|W^Tx_i + b|}{\|W\|} \geq d = \frac{1}{\|W\|} \Leftrightarrow y_i(W^Tx_i + b) \geq 1. \tag{4}$$

$$min\frac{\|W\|^2}{2} \quad s.t. y_i(W^Tx_i + b) \geq 1, i = 1,2,\ldots,n \tag{5}$$

Here we start with a special case in 2-dimensional space, like Figure 1. We know that the distance equation and the problem can be generalized to n-dimensional space. Then we first look at the problem with equality constraint. Let $x \in \mathbb{R}^n$, we want to find x^*, so that $f(x^*) = minf(x)$ and satisfy the constraints of $g(x) = 0$. With the constraint, geometrically, we want to find a point in $n-1$-dimensional space to minimize the objective function. At the point where $f(x^*)$ is at its highest or lowest under the constraints, the contour lines of the objective function and the constraint touch each other, and the gradients of both functions are perpendicular to these contour lines. Thus, $\nabla f(x^*) + \lambda\nabla g(x^*) = 0$. Based on this, the Lagrange function is defined as $\mathcal{L}(x,\lambda) = f(x) + \lambda g(x)$ [10]. Finding the stationary points of $\mathcal{L}$ we can get this:

$$\begin{cases} \frac{\partial\mathcal{L}}{\partial x} = \frac{\partial f}{\partial x} + \lambda\frac{\partial g}{\partial x} = 0 \\ \frac{\partial\mathcal{L}}{\partial\lambda} = g(x) = 0 \end{cases}. \tag{6}$$

For problems with inequality constraints: For example, $minf(x)$ subject to $g(x) \leq 0$. For $g(x) = 0$, the direction of $\nabla f(x^*)$ and $\nabla g(x^*)$ must be opposite, since $f(x^*)$ is the minimum we want,

$\nabla f(x^*)$ need to point towards inside the feasible region, but $\nabla g(x^*)$ need to point towards outside the feasible region, which is $g(x) > 0$. Thus, $\lambda > 0$. For x^* in $g(x) < 0$, we can directly let $\nabla f(x) = 0$, which means $\lambda = 0$. Thus, the problem can be transformed into minimizing Lagrange function with constraints:

$$\begin{cases} g(x) \le 0 \\ \lambda \ge 0 \\ \lambda_i g_i(x) = 0 \end{cases} \tag{7}$$

Here is the Lagrange function for optimization problems with multiple constraints:

$$min_x f(x) \ s.t. \ \begin{matrix} g_i(x) \le 0, i = 1,2,\dots,n \\ h_j(x) = 0, j = 1,2,\dots,m \end{matrix} \tag{8}$$

$$\lambda = (\lambda_1, \lambda_2, \dots, \lambda_n),^T \mu = (\mu_1, \mu_2, \dots, \mu_m)^T, \mathcal{L}(x,\lambda,\mu) = f(x) + \sum_{i=1}^{n} \lambda_i g_i(x) + \sum_{j=1}^{m} \mu_j h_j(x) \tag{9}$$

Let $minf(W) = min\frac{\|W\|^2}{2}, g_i(W) = 1 - y_i(W^T x_i + b) \le 0, i = 1,2,\dots,n,\ \lambda \ge 0$.

$$\mathcal{L}(W,b,\lambda) = \frac{\|W\|^2}{2} - \sum_{i=1}^{n} \lambda_i \, y_i(W^T x_i + b) + \sum_{i=1}^{n} \lambda_i \tag{10}$$

We can find W and b which minimizes $\mathscr{L}(W,b,\lambda)$ by differentiating $\mathscr{L}(W,b,\lambda)$ respect to W and b with their derivatives zero.

$$\frac{\partial \mathcal{L}}{\partial W} = \frac{\partial f}{\partial W} + \sum_{i=1}^{n} \lambda_i \frac{\partial g_i}{\partial W} = 0 \Longrightarrow W = \sum_{i=1}^{n} \lambda_i y_i x_i \tag{11a}$$

$$\frac{\partial \mathcal{L}}{\partial b} = 0 \Longrightarrow \sum_{i=1}^{n} \lambda_i y_i = 0 \tag{11b}$$

We know that $g_i(W) \le 0, \lambda_i \ge 0$. So, $\sum_{i=1}^{n} \lambda_i g_i(w_i) \le 0 \ \Longrightarrow \mathcal{L}(W,b,\lambda) \le f(W) \Longrightarrow max_\lambda \mathcal{L}(W,b,\lambda)$. Suppose we already found the optimal solution, and $f(W^*) = min_W f(W)$. To make sure $g_i(W) \ \le 0$,

$$max_\lambda \mathcal{L}(W,b,\lambda) = \begin{cases} \infty, g_i(w_i) > 0 \\ f(W^*), g_i(W) \ \le 0 \end{cases} \tag{12}$$

$$min_{w,b} max_\lambda \mathcal{L}(W,b,\lambda) = \ f(W^*) \tag{13}$$

$$min_{w,b} max_\lambda \mathcal{L}(W,b,\lambda) \ s.t. \ \lambda_i \ge 0. \tag{14}$$

Here is the relationship between the dual objective function [11] and the primal objective function:

$$min_{w,b} \mathcal{L}(W,b,\lambda) \le \mathcal{L}(W^*,b,\lambda) \le f(W^*) \ \Longrightarrow \ max_\lambda min_{w,b} \mathcal{L}(W,b,\lambda) \le min_{w,b} max_\lambda \mathcal{L}(W,b,\lambda). \tag{15}$$

For $min_{w,b} \mathcal{L}(W,b,\lambda)$, substituting $(11a)$ and $(11b)$ into (10),

$$\mathcal{L}(W,b,\lambda)_D = \sum_{i=1}^{n} \lambda_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \lambda_i \lambda_j y_i \, y_j \left(x_i \cdot x_j\right) \ s.t. \ \lambda_i \ge 0, \sum_{i=1}^{n} \lambda_i y_i = 0. \tag{16}$$

Here we have a dual problem, we need to find:

$$max_\lambda \left[\sum_{i=1}^{n} \lambda_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \lambda_i \lambda_j y_i \, y_j \left(x_i \cdot x_j\right) \right] s.t. \ \lambda_i \ge 0, \sum_{i=1}^{n} \lambda_i y_i = 0 \ (i = 1,2,\dots,n) \tag{17}$$

We can use a QP solver, and we will get λ, and from $(11a)$ we can get W. Here we have only b to calculate. For support vectors x_s, $g_s(W) = 1 - y_s(W^T x_s + b) = 0 \Longrightarrow y_s(W^T x_s + b) = 1$. Considering$(11a)$, $y_s(\sum_{m\in S} \lambda_m y_m x_m \cdot x_s + b) = 1$. Since $\lambda_i \geq 0$, considering (10), when $\lambda_i > 0$, the constraint works, which means $g_i(W) = 0$, where S is the collections of indices for the support vectors, which is determined by i, $\lambda_i > 0$. Then, considering $(3a)$, we can get W and b.

$$y_s{}^2\left(\sum_{m\in S} \lambda_m y_m x_m \cdot x_s + b\right) = y_s, b = \frac{1}{N_s}\sum_{s\in S}\left(y_s - \sum_{m\in S} \lambda_m y_m x_m \cdot x_s\right) \tag{18}$$

2.1.2. Classification with Regularization. For data that is not fully linearly separable, we introduce a slack variable ξ_i to allow misclassified points. Considering $(3a), y_i(W^T x_i + b) - 1 + \xi_i \geq 0,\ \xi_i \geq 0, i = 1,2,\ldots,n$. In this SVM with soft margin, we introduce a penalty, different from (5), here we get: $min\frac{\|W^2\|}{2} + C\sum_{i=1}^{n}\xi_i, s.t.\ y_i(W^T x_i + b) - 1 + \xi_i \geq 0, \xi_i \geq 0,\ i = 1,2,\ldots,n$. For this optimization problem, we reformulate the Lagrange function:

$$\mathcal{L}(W,b,\xi,\lambda,\mu) = \frac{\|W^2\|}{2} + C\sum_{i=1}^{n}\xi_i - \sum_{i=1}^{n}\lambda_i[y_i(W^T x_i + b) - 1 + \xi_i] - \sum_{i=1}^{n}\mu_i\xi_i,\ \lambda_i \geq 0, \mu_i \geq 0. \tag{19}$$

Differentiating respect to $W, b, and\ \xi_i$, we can get $W = \sum_{i=1}^{n}\lambda_i y_i x_{i,}, \sum_{i=1}^{n}\lambda_i y_i = 0$, $C = \lambda_i + \mu_i, \mu_i \geq 0 \Rightarrow C \geq \lambda_i$. Substituting these into (19), we get $max_\lambda\left[\sum_{i=1}^{n}\lambda_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\lambda_i\lambda_j y_i\, y_j\,(x_i \cdot x_j)\right]$ $s.t.\ \sum_{i=1}^{n}\lambda_i y_i = 0, \lambda_i \geq 0, 0 \leq \lambda_i \leq C$. With QP solver we can get λ, and $W = \sum_{i=1}^{n}\lambda_i y_i x_i, b = \frac{1}{N_s}\sum_{s\in S}(y_s - \sum_{m\in S}\lambda_m y_m x_m \cdot x_s)$. The difference compared to (18) is that here we need to find vectors where $0 < \lambda_i < C$ to calculate b. Here we got our hyperplane. For nonlinear classification problems, we can add a kernel function. There are several popular kernels, such as Polynomial Kernel [12] : $k(x_i, x_j) = (x_i \cdot x_j + a)^b$. Gaussian Radial Basis Function (RBF) *Kernel,* where σ is a free parameter [12]: $k(x_i, x_j) = \exp\left(-\frac{\|x_i - x_j\|^2}{2\sigma^2}\right)$.

2.2. The Multilayer Perceptron (MLP) for Classification

A typical multilayer perceptron (MLP) consists of layers and a typical neuron is shown in Figure 2. The MLP consists of L layers, the first layer represents inputs, the last one is output, and the others are $L-1$ hidden layers. Let $n_l, l = 0,1,\ldots,L-1,L$ represent the number of neurons of the lth layer, x_j^l be the output of the jth $(j = 1,2,\ldots,n_l)$ node of the lth layer, then the outputs of the lth layer can be denoted as, and the feedforward process is $x^0 \to x^1 \to x^2 \to \cdots \to x^{L-1} \to x^L$.

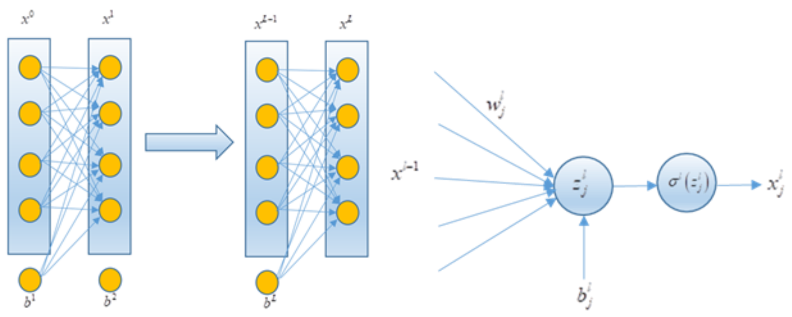


Figure 2. A Typical Multiplayer Perception and a typical Neuron

Let $w^l = \begin{bmatrix} w_1^l \\ w_1^l \\ \vdots \\ w_{n_l}^l \end{bmatrix} = \begin{bmatrix} w_{11}^l & w_{12}^l & \vdots & w_{1,n_{l-1}}^l \\ w_{21}^l & w_{22}^l & \vdots & w_{2,n_{l-1}}^l \\ \cdots & \cdots & \cdots & \cdots \\ w_{n_l,1}^l & w_{n_l,2}^l & \vdots & w_{n_l,n_{l-1}}^l \end{bmatrix}_{n_l \times n_{l-1}}$, $b^l = \begin{bmatrix} b_1^l \\ b_2^l \\ \vdots \\ b_{n_l}^l \end{bmatrix}$, σ^l denote the weights, biases, and activation function of the lth layer, the outputs of the lth layer are

$$x^l = \begin{bmatrix} \sigma^l\left(w_1^l x^{l-1} + b_1^l\right) \\ \sigma^l\left(w_2^l x^{l-1} + b_2^l\right) \\ \vdots \\ \sigma^l\left(w_{n_l}^l x^{l-1} + b_{n_l}^l\right) \end{bmatrix} = \sigma^l\left(w^l x^{l-1} + b^l\right) = \sigma^l\left(z^l\right), z^l = w^l x^{l-1} + b^l. \quad (20)$$

For each node, its output can be obtained via

$$x_j^l = \sigma^l\left(\sum_{k=1}^{n_{l-1}} w_{jk}^l x_k^{l-1} + b_j^l\right) = \sigma^l\left(\begin{bmatrix} w_{j1}^l & w_{j2}^l & \cdots & w_{jn_{l-1}}^l \end{bmatrix}\begin{bmatrix} x_1^{l-1} \\ x_2^{l-1} \\ \vdots \\ x_{n_{l-1}}^{l-1} \end{bmatrix} + b_j^l\right), \quad (21)$$
$$= \sigma^l\left(w_j^l x^{l-1} + b_j^l\right) = \sigma^l\left(z_j^l\right), z_j^l = w_j^l x^{l-1} + b_j^l, j = 1,2,...,n_l$$

For brevity, we introduce $z^l = w^l x^{l-1} + b^l = z^l\left(\theta^l, x^{l-1}\right)$, where $\theta^l = \left(w^l, b^l\right)$ are parameters of this layer are to be learned, thus,

$$x^1 = \sigma^1\left(w^1 x^0 + b^0\right) = \sigma^1\left(z^1\left(x^0, \theta^1\right)\right), \quad (22a)$$

$$x^2 = \sigma^2\left(w^2 x^1 + b^2\right) = \sigma^2\left(w^2 \sigma^1\left(w^1 x^0 + b^0\right) + b^2\right), \quad (22b)$$

Or,

$$x^2 = \sigma^2\left(z^2\left(x^1, \theta^2\right)\right) = \sigma^2\left(z^2\left(\sigma^1\left(z^1\left(x^0, \theta^1\right)\right), \theta^2\right)\right), \quad (22c)$$

$\cdots$

and the final outputs of the MLP are

$$\begin{aligned} x^L &= \sigma^L\left(w^L x^{L-1} + b^L\right) = \sigma^L\left(w^L \sigma^{L-1}\left(w^{L-1} x^{L-2} + b^{L-1}\right) + b^L\right) \\ &= \sigma^L\left(w^L \sigma^{L-1}\left(w^{L-1} \sigma^{L-2}\left(w^{L-2} x^{L-3} + b^{L-2}\right) + b^{L-1}\right) + b^L\right) \\ &= \ldots \\ &= \sigma^L\left(w^L \sigma^{L-1}\left(w^{L-1} \sigma^{L-2}\left(w^{L-2}\left(\cdots w^2 \sigma^1\left(w^1 x^0 + b^1\right) + b^2 \cdots\right) + b^{L-2}\right) + b^{L-1}\right) + b^L\right) \end{aligned}, \quad (22d)$$

or expressed in parametric form

$$\begin{aligned} x^L &= \sigma^L\left(z^L\left(x^{L-1},\theta^L\right)\right) = \sigma^L\left(z^L\left(\sigma^L\left(z^{L-1}\left(x^{L-2},\theta^{L-1}\right)\right),\theta^L\right)\right) \\ &= \ldots \\ &= \sigma^L\left(z^L\left(\sigma^L\left(z^{L-1}\left(\ldots,\sigma^1\left(z^1\left(x^0,\theta^1\right)\right)\ldots\right)\theta^{L-1}\right),\theta^L\right)\right) \\ &= \sigma^L\left(x^0,\theta^1,\theta^2,\ldots,\theta^{L-1},\theta^L\right) = \sigma^L\left(x^0,\theta\right) \end{aligned} . \tag{22e}$$

For a supervised learning problem of classification, the data pairs for training are $x^{j0},\overline{x}^j,\left(j=1,2,\ldots,N\right)$, *N* is the number of data pairs, $x^{j0},\overline{x}^j$ are input data and labels respectively and $\overline{x}_k^j \in \left\{0 \quad 1\right\}$. The activation functions of the last layer are selected as *SoftMax* function

$$x_j^L = \sigma^L\left(z_j^L\right) = \frac{e^{z_j^L}}{\sum_{k=1}^{n_L} e^{z_k^L}} \in \left(0,1\right) \ , \tag{23}$$

to realize $\sum_{j=1}^{n_L} x_j^L = 1$ unique classification. The loss function can be defined as cross entropy as

$$\begin{aligned} \mathrm{L}\left(\theta\right) &= -\frac{1}{N}\left(\sum_{j=1}^{N} \overline{x}^j \log x^{jL} + \left(1-\overline{x}^j\right)\log\left(1-x^{jL}\right)\right) \\ &= \frac{1}{N}\sum_{j=1}^{N} \mathrm{L}^j\left(\theta\right) \end{aligned} , \tag{24a}$$

$$\begin{aligned} \mathrm{L}^j\left(\theta\right) &= -\overline{x}^j \log\left(\sigma^L\left(z^L\left(x^{j,L-1},\theta^L\right)\right)\right) - \left(1-\overline{x}^j\right)\log\left(1-\sigma^L\left(z^L\left(x^{j,L-1},\theta^L\right)\right)\right) \\ &= -\overline{x}^j \log\left(\sigma^L\left(z^L\left(w^L x^{j,L-1}+b^L\right)\right)\right) - \left(1-\overline{x}^j\right)\log\left(1-\sigma^L\left(z^L\left(w^L x^{j,L-1}+b^L\right)\right)\right) \\ &= -\overline{x}^j \log\left(\sigma^L\left(x^{j0},\theta\right)\right) - \left(1-\overline{x}^j\right)\log\left(1-\sigma^L\left(x^{j0},\theta\right)\right) \end{aligned} . \tag{24b}$$

Here *N* and *L* are the numbers of training data and layers of the neural network respectively. (24) can be used to learn parameters using SGD (Stochastic Gradient Descent) based algorithms for

$$\min_{\theta} \mathrm{L}\left(\theta\right). \tag{25}$$

The gradient descent method is based on the gradient descent equation,

$$\frac{\partial\theta}{\partial t} = -\frac{\partial \mathrm{L}\left(\theta\right)}{\partial\theta} = -\frac{1}{N}\sum_{j=1}^{N}\frac{\partial \mathrm{L}^j\left(\theta\right)}{\partial\theta}. \tag{26a}$$

With its discrete version,

$$\theta^{k+1} = \theta^k - \frac{\Delta t}{N}\sum_{j=1}^{N}\frac{\partial \mathrm{L}^j\left(\theta^k\right)}{\partial\theta},\left(k=0,1,\ldots,K\right). \tag{26b}$$

Due to the large number of samples for training is usually large, calculations of gradients in (26b) are costly. SGD method can overcome this problem. Instead of calculating (26b) directly, the SGD method calculates only a mini-batch of size $M \ll N$ stochastically in each iterative step, i.e.

$$\theta^{k+1} = \theta^k - \frac{\Delta t}{M}\sum_{j=1}^{M}\frac{\partial \mathrm{L}^j\left(\theta^k\right)}{\partial \theta}, \left(k = 0,1,\ldots,K\right). \tag{26c}$$

Here,

$$\frac{\partial \mathrm{L}^j(\theta)}{\partial \theta} = \begin{bmatrix} \dfrac{\partial \mathrm{L}^j(\theta)}{\partial \theta^1} \\ \vdots \\ \dfrac{\partial \mathrm{L}^j(\theta)}{\partial \theta^l} \\ \vdots \\ \dfrac{\partial \mathrm{L}^j(\theta)}{\partial \theta^L} \end{bmatrix}, \quad \begin{array}{l} \dfrac{\partial \mathrm{L}^j(\theta)}{\partial \theta^L} = \dfrac{\partial z^L}{\partial \theta^L}\dfrac{\partial \sigma^L}{\partial z^L}\dfrac{\partial \mathrm{L}^j}{\partial \sigma^L} \\ \dfrac{\partial \mathrm{L}^j(\theta)}{\partial \theta^l} = \dfrac{\partial z^l}{\partial \theta^l}\dfrac{\partial \sigma^l}{\partial z^l}\cdots\dfrac{\partial z^L}{\partial \sigma^{L-1}}\dfrac{\partial \sigma^L}{\partial z^L}\dfrac{\partial \mathrm{L}^j}{\partial \sigma^L}, 1 < l < L \\ \dfrac{\partial \mathrm{L}^j(\theta)}{\partial \theta^1} = \dfrac{\partial z^1}{\partial \theta^1}\dfrac{\partial \sigma^1}{\partial z^1}\cdots\dfrac{\partial z^l}{\partial \sigma^{l-1}}\dfrac{\partial \sigma^l}{\partial z^l}\cdots\dfrac{\partial z^L}{\partial \sigma^{L-1}}\dfrac{\partial \sigma^L}{\partial z^L}\dfrac{\partial \mathrm{L}^j}{\partial \sigma^L} \end{array}, \tag{26d}$$

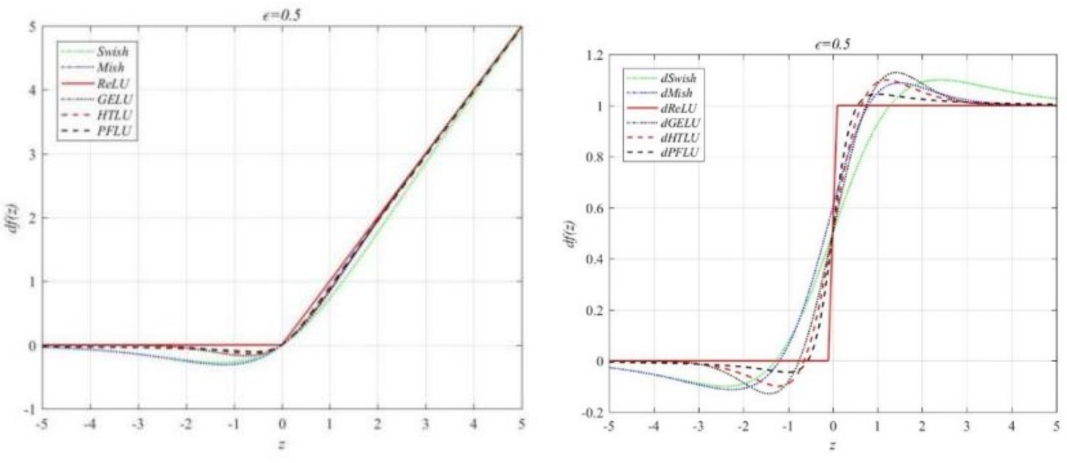
which can be calculated via automatic differentiation based on the chain rule of differentiation. n this research, we want to test different activation functions [13] for their classification performance, such as classic ones: *Sigmoid*, *Tanh,* and *ReLU*, modern ones: *Swish*, *Mish,* and *GELU*, and *PFLU* (power function linear units), *HTLU* (hyperbolic tangent linear units), where the last two ones are

$$PFLU(z) = \frac{1}{2}z\left(1 + \frac{z}{\sqrt{z^2+\varepsilon}}\right), \varepsilon = 1.9, \tag{27a}$$

$$HTLU(z) = \frac{1}{2}z\left(1 + tanh(z)\right). \tag{27b}$$

The others are summarized in [13], which were proposed by different researchers listed in references of [13].

These activation functions and their derivatives are shown in Figure 3a and Figure 3b respectively.



3a 3b

Figure 3. 3a: ReLU and its Variants, 3b: Their Derivatives

2.3. Decision Tree and Random Forest

2.3.1. Model introduction. The DT, as described in [7], is a fundamental method for classification and regression, characterized by its tree-like structure and formulation as a series of if-then rules. Notable advantages include its clear interpretability, swift training, and efficient prediction. In the training phase, the model is constructed using training data to minimize the loss function. For prediction, the model is directly employed for classification or regression tasks. DT training typically involves three stages: feature selection, DT generation, and pruning. However, a drawback of DTs is their susceptibility to overfitting, necessitating the application of pruning techniques post DT generation.

RF is a DT-based bagging ensemble algorithm, a non-parametric method [14], that can solve regression and classification problems. It overcomes the problem of DT overfitting, has a good tolerance for unbalanced samples, noise, and outliers, and has high prediction accuracy. Therefore, it has been applied by many scholars to the fields of image classification, financial decisions, and load prediction. RF uses the bootstrap method (bootstrap) to randomly extract several different sub-training sets from the original data set to build the corresponding DT, randomly including data sampling and feature selection. The test data is then input into each DT to make predictions, and the output of multiple DTs is obtained by voting.

2.3.2. Division Selection of the Decision Tree. Information entropy quantifies the uncertainty of entities based on their probabilities in mathematics. When the probability of an event is higher, indicating a lower likelihood of uncertainty, the information entropy is correspondingly lower. To calculate the information gain, we should first calculate the empirical entropy of data set D. Assuming that data set D and feature A, the empirical entropy of data set D is $H(D) = -\sum_{k=1}^{k} \frac{|C_k|}{|D|} log_2 \frac{|C_k|}{|D|}$, where $|C_k|$ is the number of samples in class k, and $|D|$ is the number of the dataset D. Then, the empirical conditional entropy $H(D|A)$ of feature A for dataset D:

$H(D|A) = \sum_{i=1}^{n} \frac{|D_i|}{|D|} H(D_i) = -\sum_{i=1}^{n} \frac{|D_i|}{|D|} \sum_{K=1}^{K} \frac{|D_{ik}|}{|D_i|} log_2 \frac{|D_{ik}|}{|D_i|}$ Finally, the information gain is calculated as $g(D,A) = H(D) - H(D|A)$. In general, the greater the information gain, the greater the "purity increase" obtained by using attribute A to divide. Therefore, we can use the information gained to make the division attribute selection of the DT.

The information gain ratio of feature A for dataset D is defined as:

$$g_R(D,A) = \frac{g(D,A)}{H_A(D)}$$

$H(D|A) = -\sum_{i=1}^{n} \frac{|D_i|}{|D|} log_2 \frac{|D_i|}{|D|}$ is called the value entropy of data set D with respect to A.

In the classification problem, the following K classes are assumed that the sample points belonging to K is P_k,then the Gini index of the probability distribution: $Gini(p) = \sum_{k=1}^{K} p_k (1 - P_k) = 1 - \sum_{k=1}^{K} P_k^2$. For dichotomy problems, the Gini index of the probability distribution is $Gini(p) = 2p(1-p)$. For the given sample set D, the Gini index: $Gini(D) = 1 - \sum_{k=1}^{K} \left(\frac{|C_k|}{|D|}\right)^2$ .When constructing DTs and using RF algorithms, the Gini coefficient is a commonly used tool for measuring data impurity. In the construction of DTs, the Gini coefficient plays a crucial role in selecting the optimal node-splitting points. This coefficient quantifies the impurity of data within a node and assists the algorithm in selecting the most suitable split points to effectively differentiate between different data categories. To achieve this, the algorithm calculates the Gini coefficient for every possible split point and ultimately selects the one with the lowest Gini coefficient.

In the context of RFs, the Gini coefficient is also employed for feature selection. RF often needs to determine which features are best suited for splitting at each node. This process involves computing the average Gini coefficient for each feature. The feature's importance score is measured based on the reduction in the Gini coefficient, where a greater reduction indicates that the feature has a more significant impact on model predictions and is, therefore, more important. This makes the Gini coefficient a critical tool in the construction of DT and RF, enabling the development of more precise models.

2.3.3. Random Forest. Random Forest (RF) is an algorithm that employs ensemble learning by integrating multiple trees, with the fundamental unit being the decision tree. Positioned within the broader domain of machine learning, specifically ensemble learning methods, RF derives its name from two key components: "random" and "forest." The term "forest" draws an analogy where a single tree represents one unit, and the amalgamation of hundreds of trees forms a forest, encapsulating the central concept of integration in RF. Intuitively, each decision tree is a classifier (assuming it's now for a classification problem and then N trees will have N classification results for an input sample. The RF integrates all the categorical voting results, and the category with the most votes is specified as the final output, which is the simplest bagging idea, and the RF algorithm is a bagging algorithm with the decision tree as the estimator.

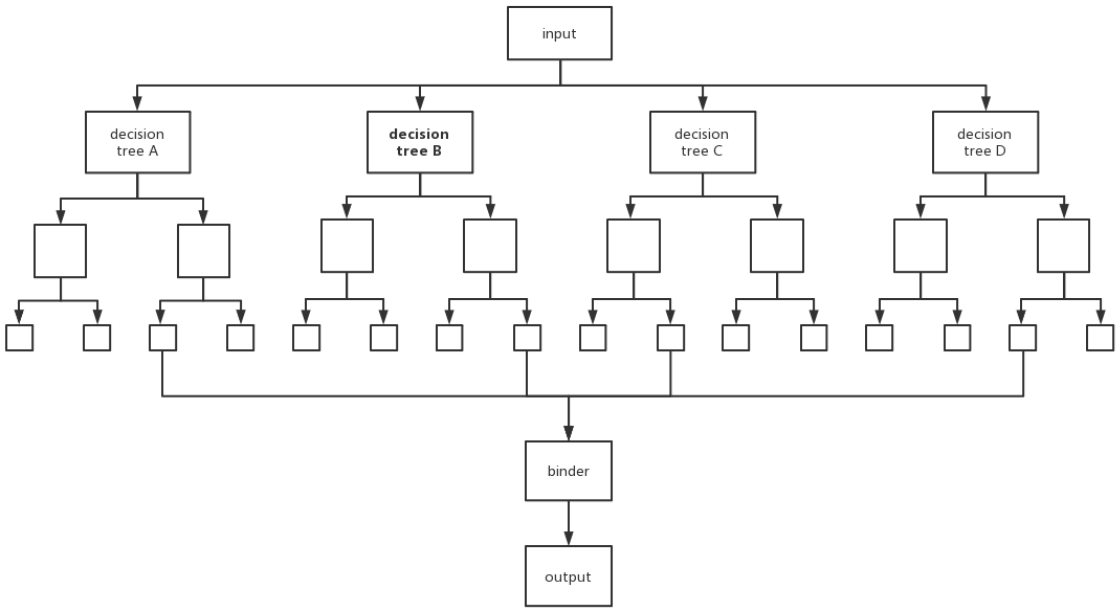


Figure 4. Process of RF

The specific process of the RF algorithm is shown above in Figure 4, in which the combiner selects the majority classification result as the final result in the classification problem and takes the average of multiple regression results as the final result in the regression problem. Assuming that the size of the training set T is N, the number of features is M, and the size of the RF is K, the specific steps of the RF algorithm are as follows: Iterate over the size of the RF K times: Randomly sample N times with replacement from the training set T to create a new sub-training set D. Randomly select m features, where m < M. Train a complete decision tree using the new training set D and the m selected features. This results in an RF.

3. Experiments

3.1. Datasets

Within the domain of machine learning, the development and application of classification algorithms have become indispensable for some complex datasets, such as the MNIST and WDBC datasets. These datasets, which respectively consist of handwritten digits and features derived from breast mass images, pose a challenge for accurate classification, necessitating the need for accurate, robust, and efficient classification algorithms [1,2].

3.1.1. MNIST. The celebrated MNIST [1,15] dataset stands for Modified National Institute Standards and Technologies databases, consists of 70000 grey value images of handwritten digits 0-9 with labels, 60000 images of them are used to train deep neural networks for deep learning, 10000 images are used to test the accuracies of the trained neural networks. Each image is 28×28 size, and the intensities of each pixel vary from $0 - 255$. Figure 5 shows some examples of this dataset and a typical handwritten digit.

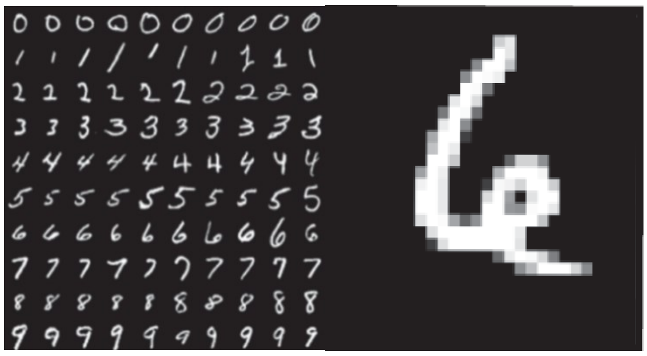


Figure 5. Partial Handwritten Digits in MNIST [16]

This large-scale dataset was set up based on the original 70000 images of 20×20 size, as a sub-set of NIST databases, through centering and interpolation manipulations in 1998 [15]. Since then, it has been widely used for testing different machine learning algorithms. It has been remarked by Geoff Hinton [17] that the MNIST database is used by neural network researchers in much the same way as biologists use fruit flies for early and quick results (before serious testing on more complex organisms) [6].

3.1.2. WDBC. The dataset of WDBC (Wisconsin Diagnostic Breast Cancer) was first used for nuclear feature extraction for breast tumor diagnosis in 1993 [2] by the research group of the University of Wisconsin., which can be downloaded from [18]. The dataset includes 569 instances, each consisting of 32 attributes. The first one is ID number, the second one is labeled with M-Malignant or B-Benign, and the other 30 data describe 10 features: radius (R), texture (T), perimeter (P), area (A), smoothness (SM), compactness (Com), concavity (Con), concave points (ConP), symmetry (SY), fractal dimension (FD), each feature has 3 values of the mean, standard error, and 'worst' cases (for instance, the largest radius). 70% of the dataset is used for training and the other 30% is used for testing.

3.2. Evaluation Metrics

Overall accuracy is the probability that for each random sample, the result classified is consistent with the test data type. We can use this formula to calculate it. It's the sum of the diagonal elements in the data matrix.

$$OA = \frac{(TP + TN)}{(TP + TN + FP + FN)}$$

True Positive (TP) is a predicting positive class to positive numbers. True Negative (TN) predicts the negative class as the number of negative classes. False Positive (FP) Predicts a negative class as a positive class is a false alarm (Type I error). False Negative (FN) predicts the number of positive classes as negative classes $\rightarrow$ underreporting (Type II error). Average Accuracy is the average accuracy of every class.

$$AA = \frac{1}{n}\sum_{i=1}^{n}\frac{(TP+TN)_n}{(TP+TN+FP+FN)_n}$$

n is the sum of the number of classes. $(TP+TN)_n$ is the number of all true numbers of nth class. $(TP+TN+FP+FN)_n$ is the number of all numbers of nth class.

The kappa coefficient is a metric used for consistency testing and can also be used to measure the effectiveness of classification. The kappa coefficients take values between -1 and 1, and they are usually greater than 0. The kappa coefficient can be calculated by the formula [19]:

$$\kappa = \frac{Po - Pe}{1 - Pe}$$

Po is the parameter that equals the sum of the elements of the diagonal divided by the sum of the elements of the whole matrix which is OA. Pe is the parameter that equals the sum of the sum of all elements in nth row add the sum of all elements in nth column in every row and column of the matrix divided by the square of the sum of the elements of the whole matrix.

3.3. Experimental Settings and Separate Result

3.3.1. SVM.

a) Experiment based on MNIST

The one-against-one classifier trains binary classifiers for N-classes data sets. Each classifier separates a pair of classes. On the other hand, a one-versus-rest [20] classifier allows us to train a single classifier per class while repeating N times. Here for linear support vectors classifier (SVC), we use the "one-versus-rest" strategy for multiclass classification. We train the model on the combined training and validation data. We iterate through a list of different values of the cost parameter "c". We calculate the accuracy and display the confusion matrix. We observe a delicate balance between bias and variance in the table of different parameters and corresponding results. We used grid search [21] with c across the range of $(0.001, 0.01, 0.1, 1, 10)$, and we finally got the testing accuracy of 91.8% with $c = 1$.

For SVC with RBF kernel, we take a random 10% sample from the training data and test data to reduce computation time. We run the SVC for multiple combinations of cost factors "c" and Gamma. We perform the grid search with the same range of c across the range of $(1, 5, 10, 15)$ and gamma across the range of $(0.001, 0.01, 0.01, 0.1, 1, 10)$. The model gets the best performance with $c = 15$ and $gamma = 0.01$, which gives the accuracy of 96.6% while avoiding overfitting.

As for SVC with the poly kernel, we run the SVC for multiple combinations of different cost factors and degrees, where the degrees parameter in the polynomial kernel determines the degree of the polynomial used in the mapping. As the cost and degree increase, the testing accuracy decreases, indicating overfitting, and we see low bias and high variance. With the cost keeping constant, increasing degree results in immediate overfitting. Here we get the best performance with $c = 10$ and $degree = 2$, which gives us a testing accuracy of 96.2%.

b) Experiments based on WDBC

Considering features of WDBC with varying measurement scales, and two classes (malignant and benign), we reprocess the dataset with standardization [22] to ensure features are on the same scale. We split data points into training and testing sets. Given the complexity of WDBC, we consider the cost parameter (c) tuning. Here we set c parameter varied across the range of $(0.001, 0.01, 0.1, 1, 10)$ to find the best performing one. As shown in Figure 6, we can see the imbalance as the cost parameter increases. We used the grid search with cross-validation to reduce the risk of overfitting. We finally approach the testing accuracy of 98.6%.

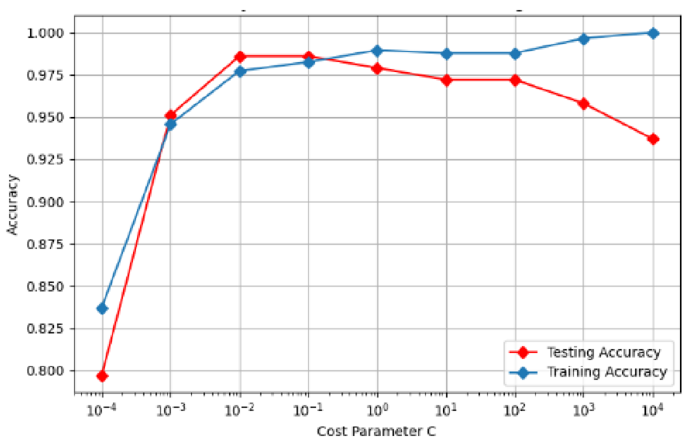


Figure 6. Accuracy versus the Cost Parameter C (log-scale) in Linear SVM for WDBC

For SVM with RBF kernel function, gamma has the range of $(0.0001, 0.001, 0.01, 0.1, 1, 10)$, and c varies across the same range as that of linear one. Eventually, we get an accuracy of 98.6%, with the best c of 10 and $gamma$ of 0.01. The observation suggests that the dataset might be inherently well-suited for linear separation, leading to similar outcomes between the linear and RBF kernels.

With the same reprocessing method, degrees vary across the range of $(2, 3, 4)$, and the range of c is unchanged. The chosen degrees allow the model to capture more complex relationships between features and nonlinear patterns. The optimal accuracy of 96.5% is achieved with $c = 10$ and degrees of 3.

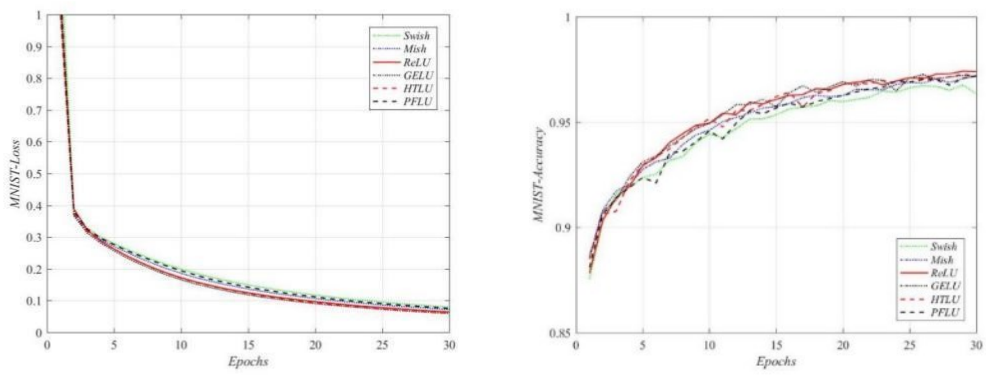
3.3.2. MLP Network setup, training, and testing.

a) Experiments based on MNIST

In this experiment, we construct a simple MLP neural network with five layers. The first layer has 784 nodes corresponding with pieces of an image of 28×28 size. The next three layers have 128 nodes, 64 nodes, and 10 nodes respectively. The last layer has 10 nodes also for outputs corresponding with a one-hot vector. The activation functions in hidden layers are *ReLU*, *Swish*, *Mish*, *GELU*, *HTLU*, *and PFLU* functions for comparisons, the *SoftMax* functions are employed in the last layer.

The 60000 images are fed into the network in batches of 64 images stochastically in each epoch. We set 30 epochs in training processes based on the SGD method with a learning rate of 0.01. The initial values of parameters are selected randomly based on seeds. The other 10000 images are used to test for accuracies with the ratio of the correct result number and the number of tested samples.

Figure 7a and Figure 7b show the changes in loss functions and accuracy with epochs by using these activation functions respectively.



7a 7b

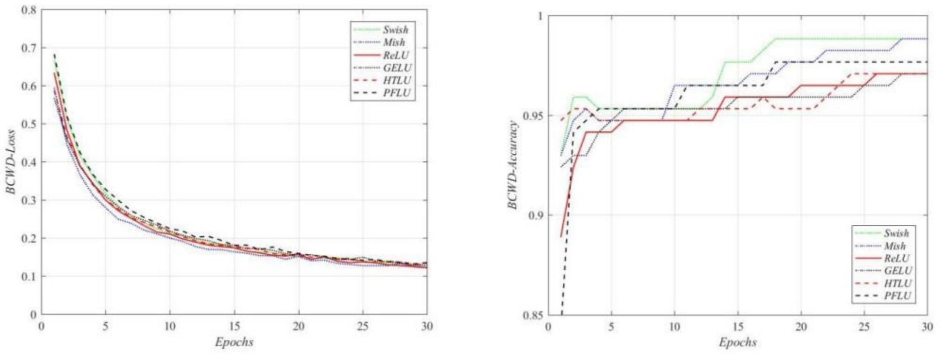
Figure 7. 7a: Changes of Loss Function via Epochs for Different Activation Functions, 7b: Changes of Accuracy via Epochs for Different Activation Functions

b) Experiments based on WDBC

In this experiment, we construct a simple MLP neural network with four layers. The first layer has 30 nodes corresponding with 30 features. The next two layers have 128 nodes, and 2 nodes respectively. The last layer has 2 nodes for outputs corresponding with a one-hot vector. The activation functions in hidden layers are *ReLU*, *Swish*, *Mish*, *GELU*, *HTLU*, *and PFLU* functions for comparisons, the *SoftMax* functions are employed in the last layer.

The data of 400 persons are fed into the network in batches of 64 stochastically in each epoch. We set 30 epochs in training processes based on the SGD method with a learning rate of 0.01. The initial values of parameters are selected randomly based on seeds. The other data of 169 persons are used to test for accuracies with the ratio of correct result number and the number of tested samples.

Figure 8a and Figure 8b show the changes in loss functions and accuracies with epochs by using these activation functions respectively.



8a 8b

Figure 8. 8a: Changes of loss function via epochs for different activation functions, 8b: Changes of accuracy via epochs for different activation functions

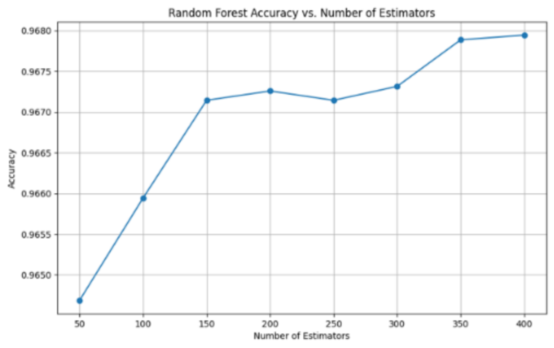
3.3.3. Decision Tree, Random Forest. In this study, we first utilized the fetch_openml function from Scikit-Learn to load the MNIST dataset as well as the WDBC (Wisconsin Diagnostic Breast Cancer) dataset. Subsequently, we conducted data preprocessing by dividing the datasets into training and testing sets to streamline the processes of model training and evaluation.

Next, we delved into the training process of the DT classifier. The DT is a supervised learning algorithm that relies on feature-based segmentation for classification. We employed the decision tree classifier from Scikit-Learn to construct and train the DT model. The model training process encompasses feature selection, tree growth, and node splitting, all aimed at maximizing classification accuracy.

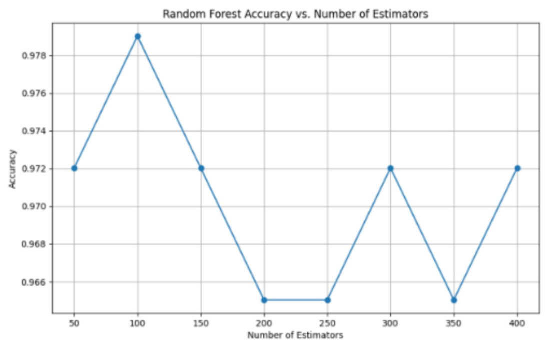
Following that, our focus shifted to the training of the RF classifier, an ensemble learning method that enhances classification performance by combining multiple DTs. We used the Random-Forest-Classifier in Scikit-Learn to build the RF model. During model construction, we had to make critical choices regarding hyperparameters for instance, the number of DTs (n_estimators) and the maximum depth of each tree (max_depth). The selection of these hyperparameters is pivotal to the model's performance and therefore necessitates careful tuning.

To identify the optimal combination of hyperparameters, we employed GridSearchCV for hyperparameter grid search. GridSearchCV allowed us to conduct cross-validation within the specified hyperparameter combinations, to identify the combination that delivers the best performance. In our study, we considered various values for n_estimators and max_depth, employing 2-fold cross-validation to assess the performance of each combination. Ultimately, we selected the hyperparameter combination that yielded the highest performance to construct the final RF classifier model. In the MNIST dataset, different numbers of decision trees were used to draw a graph showing the corresponding accuracy of random forests. It can be observed in Figure 9 that the accuracy is highest when the number of decision trees is 400.

In the WDBC dataset, different numbers of decision trees were used to draw a graph showing the corresponding accuracy of random forests. It can be observed that the accuracy is highest when the number of decision trees is 100. Figure 9a and Figure 9b respectively demonstrate the trend of random forest accuracy varying with the number of decision trees in the MNIST and WDBC datasets. Table 1 and Table 2 respectively present the accuracy of decision trees, random forests, Gini coefficients, and Entropy Loss Rate under the MNIST and WDBC datasets.



9a



9b

Figure 9.

Table 1. Experiments based on MNIST

DT accuracy	RF Accuracy	Gini Index	Entropy Loss Rate
0.8716	0.9678	0.0623	0.2052

Table 2. Experiments based on WDBC

DT accuracy	RF Accuracy	Gini Index	Entropy Loss Rate
0.9510	0.9791	0.0675	0.2187

3.4. Compare and Contrast

Table 3. Metrics Comparison for MNIST

Algorithms		Kappa	OA	AA	Time (training)/s	Time (testing)/s
SVM	***Linear***	0.908	0.918	0.926	19.1	0.184
	RBF	0.962	0.966	0.966	453.03	**0.0596**
	Poly	0.932	0.939	**0.983**	2297	2.62
DT	***Gini Index = 0.0623 Entropy Loss Rate = 0.2052***	0.854	0.871	0.856	8.31	0.0728
RF	***Gini Index = 0.0623 Entropy Loss Rate = 0.2052***	0.960	0.968	0.964	**4.39**	0.200
MLP	***Swish***	0.964	0.968	0.968	37.7	0.357
	Mish	0.970	0.973	0.973	41.1	0.0934
	ReLU	0.967	0.970	0.970	34.1	0.0598
	GELU	**0.971**	**0.974**	0.974	35.8	0.0768
	HTLU	0.970	0.973	0.973	40.1	0.0868
	PFLU	0.966	0.969	0.967	47.390	0.0769

Table 4. Metrics Comparison for WDBC

Algorithms		Kappa	OA	AA	Time(training)/s	Time(testing)/s
SVM	***Linear***	0.972	0.986	0.986	0.00598	0.000998
	RBF	0.972	0.986	0.986	0.00299	0.000997
	Poly	0.930	0.965	0.965	0.00403	0.000526
DT	***Gini Index = 0.0675 Entropy Loss Rate = 0.2187***	0.853	0.951	0.951	**0.001**	0.001
RF	***Gini Index = 0.0675 Entropy Loss Rate = 0.2187***	0.925	0.965	0.961	0.24	0.003
MLP	***Swish***	0.924	0.965	0.965	0.231	0.00200
	Mish	**0.975**	**0.988**	**0.987**	0.290	**0.000100**
	ReLU	0.962	0.983	**0.987**	0.219	**0.000100**
	GELU	0.937	0.971	0.970	0.251	0.00200
	HTLU	0.924	0.965	0.965	0.280	0.00300
	PFLU	0.925	0.965	0.962	0.291	0.000100

In Table 3, we find that the Kappa values of all the methods to classify MNIST are high. They are all above 0.81. This indicates that the results of all the methods are almost perfect. However, the Kappa coefficient is generally greater when using MLP to classify MNIST which has an average value of 0.97. Apart from it, the Kappa values sorted from largest to smallest are RF, SVM and DT. The value of OA and AA is great when using RF and MLP. It shows that the number of true samples is heavily weighted in the total sample number. However, in general, the OA and AA of MLP appeared to be more stable and highly. However, the accuracy of DT is much higher than that of MLP, which is 12% lower in general.

The value of time used to train and to test of the RF is the minimum, which is 4.39s and 0.200s respectively. The second fastest is DT, the third fastest is SVM, and the slowest is MLP. Although the testing time of MLP is very fast, its training time is very long, which is 40 seconds on average. And when using SVM, only linear kernel gives the result quickly.

In Table 4, All Kappa values are almost perfect too. Overall, SVM has a somewhat higher Kappa coefficient, and the second greatest Kappa value is the value when using MLP to classify. Next is RF and the lowest value is from DT.

OA and AA of the classification of the dataset WDBC is very high when using SVM. Based on the comparison of the overall data, we found that OA and AA measured with SVM were generally higher than MLP by 0.1. However, the DT has the lowest accuracy which is 3.5% lower than SVM. The value of time used to train and to test the SVM when using RBF is the minimum, which is 0.00299s and 0.000997s respectively. The second fastest is DT, the third fastest is MLP and the slowest is RF. Same to Table 3, the testing time of MLP is speedy, and its training time is very long.

4. Conclusion

This paper provides an in-depth examination of diverse algorithms, such as SVM with varying kernels, DT, RF, and MLP, as applied to MNIST and WDBC. The experiments yield diverse results and efficiencies, facilitating a comprehensive comparison of metrics.

Roughly summarizing, the standout performer across both datasets was the linear SVM. Its adeptness in capturing the nuances of handwritten digits and discerning benign from malignant tumors shows its polyfunctionality. Also, it achieves a precise equilibrium between precision and computational efficiency. The risk of overfitting is reduced because of the simplicity of the linear kernel, making it more resilient when dealing with limited datasets.

However, the choice of the optimal algorithm is complicated and context dependent. The poly kernel SVM, with an AA of 0.983 for MNIST, highlights the importance of aligning algorithmic choices with specific application requirements. DT always gives the result quickly but with low accuracy. RF is the fastest way and gives the second most accurate result when classifying MNIST, but it is slow and not very accurate when classifying WDBC. MLP always gives an accurate result and takes a lot of time. When classifying the dataset of MNIST, if you want the most accurate results, using MLP is the best choice as it has generally high accuracy and the greatest value of the Kappa coefficient. If you want to get the result promptly, you should use the RF in future investigations. When classifying the dataset WDBC, if you want the most accurate results, using SVM is better as it has the best accuracy and the highest Kappa coefficient. If you want to get the result promptly, you should use the SVM, it requires less running time.

Despite the comprehensive analysis and success, potential errors might arise from overfitting or imbalanced datasets. Also, the quality and representativeness of the datasets, MNIST and WDBC, are important factors influencing model performance. Any biases or noise in the datasets could impact the generalizability of our models. The model's hyperparameters, including regularization terms and kernel parameters, may not be optimized to their fullest potential. Furthermore, the study focused on individual algorithms, neglecting the potential benefits of ensemble methods. Future directions involve rigorous error analysis, hyperparameter optimization, and exploring ensemble methods.

Our study serves as a foundational exploration, highlighting the polyfunctionality of the linear SVM while acknowledging the potential for further optimization. In the face of time constraints, optimizing hyperparameters for each algorithm and conducting a more exhaustive search for optimal configurations could be beneficial. With additional time and resources, an in-depth investigation into potential errors, such as dataset quality and hyperparameter fine-tuning, should be conducted. Moreover, exploring the integration of different methods, diverse neural network architectures, and validating models on real-world datasets will likely lead to improved model performance.

References

[1] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to docu ment recognition," Proceedings of the IEEE, vol. 86, no. 11, pp. 2278–2324, 1998. doi: 10.1109/5.726791

[2] W. H. Wolberg, W. N. Street, and O. L. Mangasarian, "Machine learning techniques to diagnose breast cancer from image-processed nuclear features of fine needle aspirates," Cancer Letters, vol. 77, no. 2–3, pp. 163–171, 1995. doi:10.1016/0304-3835(94)90099-x

[3] C. Cortes and V. Vapnik, "Support-Vector Networks," Machine Learning, vol. 20, no. 3, pp. 273–297, 1995. doi:10.1007/bf00994018

[4] L. Breiman, Machine Learning, vol. 45, no. 1, pp. 5–32, 2001. doi:10.1023/a:1010933404324

[5] C. M. Bishop, Pattern Recognition and Machine Learning. Springer New York, 2016.

[6] C. C. Aggarwal, Neural Networks and Deep Learning: A Textbook. Springer, 2023.

[7] Z. Wang, J. Xu, J. Ma, and Z. Cai, "A novel combined intelligent algorithm prediction model for the risk of the coal and gas outburst," Scientific Reports, vol. 13, no. 1, 2023. doi:10.1038/s41598-023-43013-0

[8] J. Kumar. Mandal and D. Bhattacharya, " Supervised Classification Algorithms in Machine Learning: A Survey and Review," in Emerging technology in modelling and graphics proceedings of IEM Graph 2018, Singapore: Springer Singapore, 2020

[9] Z. Keita, "Classification in machine learning: A guide for beginners," DataCamp, https://www.datacamp.com/blog/classification-machine-learning (accessed Nov. 4, 2023).

[10] D. P. Bertsekas, *Constrained Optimization and Lagrange Multiplier Methods*. Belmont, MA: Athena Scientific, 1996.

[11] S. P. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge: Cambridge Univ. Pr., 2011.

[12] N. Cristianini and J. Shawe-Taylor, *An Introduction to Support Vector Machines: And Other Kernel-Based Learning Methods*. Cambridge: Cambridge University Press, 2006.

[13] A. D. Jagtap and G. E. Karniadakis, "How important are activation functions in regression and classification? A survey, performance comparison, and Future Directions," Journal of

Machine Learning for Modeling and Computing, vol. 4, no. 1, pp. 21–75, 2023. doi:10.1615/jmachlearnmodelcomput.2023047367

[14] X. Liu, H. Yang, J. Yang, and F. Liu, “Application of RF model integrated with feature reduction for biomass torrefaction,” Sustainability, vol. 14, no. 23, p. 16055, 2022. doi:10.3390/su142316055

[15] “The mnist database,” MNIST handwritten digit database, Yann LeCun, Corinna Cortes and Chris Burges, http://yann.lecun.com/exdb/mnist/ (accessed Oct. 18, 2023).

[16] R. C. Gonzalez and R. E. Woods, Digital Image Processing. Pearson, 2019.

[17] “Lecture 1/16 : Introduction,” YouTube, https://www.youtube.com/watch?v=2fRnHVVLf1Y&list=PLiPvV5TNogxKKwvKb1RKwkq2hm7ZvpHz0 (accessed Oct. 18, 2023).

[18] “Breast cancer wisconsin (Diagnostic),” UCI Machine Learning Repository, https://archive.ics.uci.edu/dataset/17/breast+cancer+wisconsin+diagnostic. (accessed Oct. 18, 2023).

[19] W. Tang, J. Hu, H. Zhang, P. Wu, and H. He, “Kappa coefficient: A popular measure of Rater Agreement,” Shanghai archives of psychiatry, https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4372765/ (accessed Oct. 23, 2023).

[20] C. M. Bishop, Pattern Recognition and Machine Learning. MTM, 2023.

[21] I. Syarif, A. Prugel-Bennett, and G. Wills, “SVM parameter optimization using grid search and genetic algorithm to improve classification performance,” TELKOMNIKA (Telecommunication Computing Electronics and Control), vol. 14, no. 4, p. 1502, 2016. doi:10.12928/telkomnika.v14i4.3956

[22] X. Wan, “Influence of feature scaling on convergence of gradient iterative algorithm,” Journal of Physics: Conference Series, vol. 1213, no. 3, p. 032021, 2019. doi:10.1088/1742-6596/1213/3/032021