Paradigm Academic Press
Innovation in Science and Technology
ISSN 2788-7030
JUN. 2026 VOL.5, NO.2

A Survey of Classic Machine Learning Algorithms: Principles and Applications

Xi Long[1] & Xing Zhao[2]

[1] School of Statistics and Data Science, Capital University of Economics and Business, Beijing 100071, China

[2] School of Urban Economics and Public Administration, Capital University of Economics and Business, Beijing 100071, China

Correspondence: Xing Zhao, School of Urban Economics and Public Administration, Capital University of Economics and Business, Beijing 100071, China.

doi:10.63593/IST.2788-7030.2026.06.003

Abstract

Classical machine learning algorithms constitute the fundamental cornerstone of modern data science and intelligent system development. While deep learning has achieved transformative breakthroughs across numerous fields in recent years, classical methods remain indispensable in practical scenarios characterized by limited training data, stringent interpretability requirements, or constrained computational resources. Nevertheless, existing studies generally lack a systematic, unified, and beginner-friendly comprehensive survey that integrates theoretical elaboration, multi-dimensional comparative analysis, and actionable algorithm selection guidance.

To fill this research gap, this paper presents a thorough investigation of ten representative classical machine learning algorithms: Linear Regression, Logistic Regression, Decision Tree, Random Forest, Support Vector Machine, K-Nearest Neighbors, Naive Bayes, Adaptive Boosting, K-Means, and Principal Component Analysis. Each algorithm is explicated within a consistent structural framework, covering its mathematical formulation, core working mechanism, distinct advantages, inherent limitations, and typical application scenarios. Furthermore, a horizontal comparative analysis is conducted across seven critical dimensions, and a practical algorithm selection framework with targeted recommendations for typical industrial scenarios is proposed.

This work constructs a complete and logically coherent knowledge system of classical machine learning, serving as an accessible and pragmatic reference for novice learners and engineering practitioners. It also provides insights into future integration trends between classical algorithms and emerging technologies including large language models, explainable artificial intelligence, edge intelligence, automated machine learning, and federated learning.

Keywords: machine learning, algorithm survey, comparative analysis, algorithm selection, mathematical formulation

1. Introduction

1.1 Background and Definition

Machine learning (ML) is a fundamental subfield of artificial intelligence (AI) that enables computer systems to automatically learn and improve from experience without being explicitly programmed to perform specific tasks. (Michalski, 2013)

The most rigorous and widely cited definition of machine learning was provided by Mitchell in his landmark textbook: “A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E”. (Mitchell, 1997) This definition elegantly captures the three essential components of any machine learning system: (1) Task

(T): the specific problem the system is designed to solve, such as classification, regression, or clustering; (2) Experience (E): the data used to train the system; and (3) Performance Measure (P): a quantitative metric used to evaluate how well the system performs the task.

This framework provides a unified lens through which we can understand all the classic machine learning algorithms discussed in this survey, as each algorithm represents a different approach to optimizing performance P given experience E for a particular class of tasks T.

1.2 Historical Evolution of Machine Learning

The evolution of machine learning over the past 70 years has been defined by alternating periods of rapid theoretical progress and pragmatic retrenchment, driven by the interplay between computational capabilities, mathematical insights, and real-world demand. The field traces its intellectual origins to the 1950s, when early AI researchers first proposed that machines could be designed to learn from data rather than relying solely on hand-coded rules. The seminal work of Samuel on a self-learning checkers program provided the first empirical demonstration of this concept, showing that a computer could outperform its human programmer by iteratively improving its strategy through self-play. (Samuel, 1959) This early era was dominated by symbolic approaches and optimistic claims about general AI, but these efforts ultimately stalled in the 1970s due to fundamental limitations in computational power and the inability to handle complex, noisy real-world data—a period now known as the first “AI winter”. (Toosi, 2021)

The 1980s and 1990s marked a decisive shift toward statistical learning theory, which remains the mathematical foundation of modern classical machine learning. (Wu, 1999) During this period, researchers developed rigorous theoretical frameworks for understanding generalization, overfitting, and model complexity, most notably Vapnik’s statistical learning theory, which laid the groundwork for support vector machines (SVMs) and established formal bounds on model performance. (Vapnik, 1999) This era also saw the maturation of many core algorithms that remain widely used today, including decision trees, k-nearest neighbors, and the backpropagation algorithm for neural networks. By the late 1990s, machine learning had emerged as a distinct discipline separate from general AI, with a focus on solving practical, data-driven problems.

The 2000s witnessed the rise of ensemble learning methods and the widespread adoption of machine learning in industrial applications. Algorithms such as random forests and gradient boosting machines combined multiple weak learners to achieve state-of-the-art performance on a wide range of tasks, while advances in computing infrastructure made it possible to process increasingly large datasets. During this period, classical machine learning became the dominant approach for data analysis across industries, from finance and healthcare to marketing and cybersecurity.

The past decade has been dominated by the explosive growth of deep learning, which has achieved remarkable success in areas such as computer vision, natural language processing, and speech recognition. (Zhou, 2021) This has led some to question the continued relevance of classical machine learning algorithms. However, classical methods remain indispensable in many real-world scenarios, particularly when data is limited, interpretability is critical, or computational resources are constrained. Moreover, a solid understanding of classical algorithms is essential for grasping the fundamental principles that underpin all machine learning, including deep learning. (Ezugwu, 2024)

1.3 Related Work and Research Gap

Despite the vast body of literature on machine learning, there remains a critical gap in accessible, comprehensive introductions to the core classical algorithms for beginners. Existing surveys tend to fall into one of three categories: highly theoretical treatises that are inaccessible to non-experts, domain-specific reviews that focus on particular applications rather than general algorithms, or broad overviews that sacrifice depth for breadth. Few studies provide a systematic, unified treatment of the ten most fundamental algorithms, covering their mathematical principles, advantages, limitations, and practical applications in a consistent format. Furthermore, there is a lack of clear, actionable guidance for selecting the appropriate algorithm for a given problem—a common pain point for novice practitioners.

1.4 Contributions of This Paper

This paper addresses these gaps by presenting a comprehensive, beginner-friendly survey of ten classic machine learning algorithms that form the foundation of modern data science. The contributions of this work are threefold: (1) A systematic introduction to the core principles and mathematical formulations of each algorithm in a unified framework; (2) A rigorous comparative analysis of their strengths, weaknesses, and applicable scenarios; (3) A practical algorithm selection guide to help practitioners make informed decisions.

2. Fundamental Concepts of Machine Learning

2.1 Major Learning Paradigms

Machine learning algorithms are primarily categorized into two fundamental paradigms based on the type of training data and learning objectives.

Supervised learning is the most widely used paradigm, where algorithms learn from labeled datasets. (Liu, 2011) Each sample in the training data consists of input features paired with a corresponding output label. The algorithm learns a mapping function from features to labels, enabling it to predict labels for new, unseen samples. Supervised learning addresses two core task types:

(1) Classification: Predicting discrete categorical labels, such as identifying whether an email is spam or legitimate.

(2) Regression: Predicting continuous numerical values, such as forecasting monthly sales or estimating house prices.

Unsupervised learning operates on unlabeled datasets, where no predefined output labels are provided. (Barlow, 1989) The goal is to discover hidden patterns, structures, or inherent relationships within the data. The two primary unsupervised learning tasks are:

Clustering: Grouping similar samples into distinct clusters based on their feature similarity.

Dimensionality reduction: Reducing the number of input features while preserving the most critical information in the dataset.

Other advanced learning paradigms, including semi-supervised learning and reinforcement learning, are beyond the scope of this survey.

2.2 Standard Machine Learning Workflow

A typical machine learning project follows a standardized five-stage workflow that ensures systematic development and deployment of reliable models (Zhou, 2021):

Data collection and preprocessing: Gathering relevant data and cleaning it to handle missing values, outliers, and inconsistent formats.

Feature engineering: Transforming raw data into meaningful features that better represent the underlying problem to the model.

Model selection and training: Choosing an appropriate algorithm based on the problem type and data characteristics, then training it on the training dataset to learn the feature-label mapping.

Model evaluation and tuning: Assessing model performance on a separate validation dataset and adjusting hyperparameters to optimize generalization ability.

Model deployment and monitoring: Deploying the trained model to production environments and continuously monitoring its performance over time to detect degradation.

2.3 Brief Overview of Performance Measure

Evaluating the generalization performance of a learner requires not only effective and feasible experimental estimation methods but also evaluation criteria to measure the generalization ability of the model, which is referred to as performance measure. Quantitative performance measures are essential for objectively evaluating and comparing different models.

The choice of metric depends on the specific task type:

The evaluation metrics for regression tasks focus on the difference between the predicted value and the actual value. For regression tasks, the most commonly used measures are:

(1) Mean Squared Error (MSE): Measures the average squared difference between predicted and actual values, penalizing larger errors more heavily. (Fahrmeir, 2022)

Given a dataset D containing n samples. Given a sample set, where y is the true label of example x_i. To evaluate the performance of learner f, we need to compare the learner's prediction $f(x)$ with the true label y. The most commonly used performance metric for regression tasks is Mean Squared Error.

$$E(f;D) = \frac{1}{n}\sum_{i=1}^{n}(f(x_i) - y_i)^2$$

(2) Mean Absolute Error (MAE): Measures the average absolute difference between predicted and actual values, providing a more robust measure of error magnitude. (Fahrmeir, 2022)

For classification tasks, let's illustrate with a simple binary classification example. The first concept to clarify is that the system needs to classify samples into two categories: positive and negative. (Rupp, 2008)

The box in the figure below represents all the samples that need to be predicted. The samples within the blue circle represent those that the system predicts as positive, while the samples within the green circle represent those that are actually positive.

So the part where blue and green intersect represents True Positive (TP), where the system predicts true and the actual situation is also true, indicating that the system's prediction is correct (True);

The green part outside the blue circle represents samples that the system predicted as false but are actually true. These are False Negatives (FN), where the system made an error in prediction (False);

The blue part outside the green circle represents samples that the system predicts as true but are actually false. These are False Positives (FP), where the system made an error in prediction;

Finally, the area outside the two circles represents samples that the system predicts as negative and are indeed negative, namely, correctly predicted True Negatives (TN).

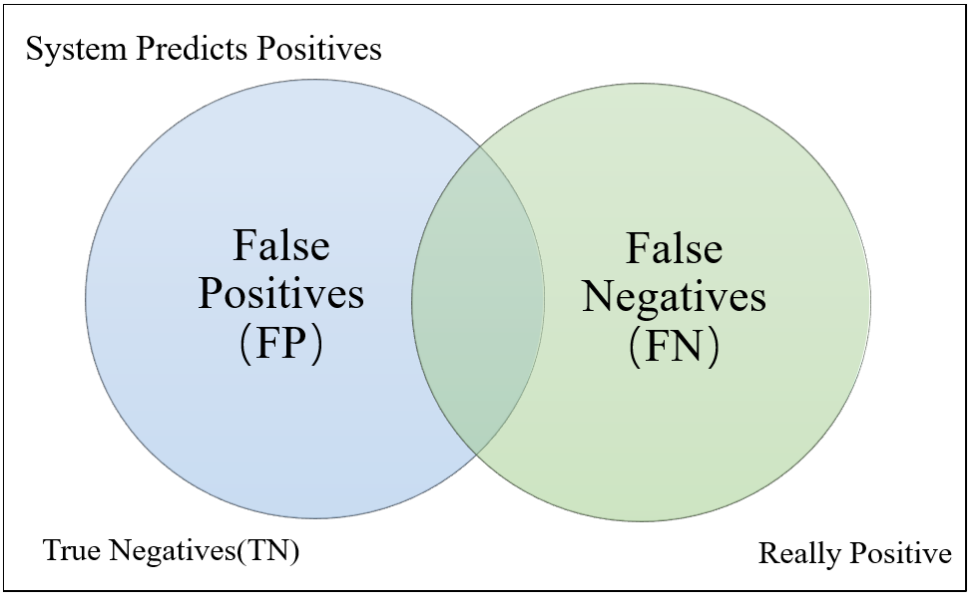

Figure 1.

The above figure can be summarized by the following table, which is called a confusion matrix.

Table 1.

	YES	NO
YES	TP	FN
NO	FP	TN

For binary classification problems, the confusion matrix is a 2×2 table, where the rows represent the true labels of the samples and the columns represent the predicted labels of the model. The first row indicates samples that are truly positive, while the second row indicates samples that are truly negative; the first column indicates samples that are predicted as positive, while the second column indicates samples that are predicted as negative.

It can be seen that the elements on the main diagonal (TP and TN) correspond to correctly predicted samples, while the elements on the subdiagonal (FN and FP) correspond to incorrectly predicted samples. Therefore, a classifier with good performance should have a larger value on the main diagonal and a smaller value on the subdiagonal.

The standard measures include:

(1) Error Rate and Accuracy: The ratio of the number of misclassified samples to the total number of samples. The overall proportion of correct predictions, suitable for balanced datasets.

$$ErrorRate = \frac{error}{total} = \frac{FP + FN}{TP + TN + FP + FN}$$

$$Accuracy = (1 - error) = \frac{correct}{total} = \frac{TP + TN}{TP + TN + FP + FN}$$

(2) Precision and Recall: Provide detailed insights into performance on specific classes, critical for imbalanced datasets where misclassification costs vary.

Precision(P):

$$P = \frac{TP}{TP + FP}$$

Recall(R):

$$R = \frac{TP}{TP + FN}$$

(3) F-score: Other fields use different criteria. For example, F-score is commonly used in information retrieval and represents the harmonic mean of Recall and Precision, which are combined as follows:

$$F_\beta = \frac{(\beta^2 + 1) \cdot P \cdot R}{\beta^2 P + R}$$

The F-score has a parameter β, which assigns different weights to Recall and Precision by adjusting this parameter. When this parameter is set to 1, it becomes F_1, indicating that Recall and Precision have equal weights.

$$F_1 = \frac{2PR}{P + R}$$

It is easy to see that when the value of Recall or Precision is small, the F-score will also be small. Therefore, regardless of how high one of these values is, as long as the other is small, the F-score will be small.

3. Ten Classic Machine Learning Algorithms

Next, we will introduce ten classic machine learning algorithms, and explain the basic principles and applications of each algorithm.

3.1 Linear Regression Algorithms

Linear regression is a fundamental predictive method in machine learning. It is based on the assumption that there is a linear relationship between the input (features) and the output (target).

The core principle of linear regression is to find a set of weights (coefficients) that make the linear combination of these weights and features as close as possible to the target value. During the training process, these weights are determined by minimizing the difference between the predicted values and the actual values, which is usually the sum of squared errors.

In regression analysis, if only one independent variable and one dependent variable are included, and the relationship between them can be approximately represented by a straight line, then this type of regression analysis is called simple linear regression analysis. (Montgomery, 2021) It usually takes the form of:

$$y = \beta_0 + \beta_1 x + \varepsilon$$

Where y is the target variable, x is the feature, β_0 is the intercept, β_1 is the slope, and ε is the error term.

If regression analysis involves two or more independent variables, and there is a linear relationship between the dependent variable and the independent variables, it is referred to as multiple linear regression analysis. The general form is as follows:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_n x_n + \varepsilon$$

In this formula, $x_1, x_2, \ldots x_n$ represent the features, while $\beta_0, \beta_1, \beta_2, \ldots \beta_n$ are the coefficients of these features.

The linear regression model possesses the following advantages:

(1) Simplicity and intuitiveness: The linear regression model is easy to understand and implement, making it a suitable learning tool for beginners in machine learning.

(2) High interpretability: The results of a linear regression model are easy to interpret, clearly showing how features affect the prediction.

(3) Fundamental status: As a fundamental model in machine learning, it serves as a cornerstone for understanding more complex models.

(4) Computational efficiency: Compared to more complex models, linear regression models have high computational efficiency and are easy to handle large-scale data processing.

Although the linear regression model is a relatively basic algorithm, it plays a significant role in our lives. In cases where there is a clear linear relationship between features and the target variable, linear regression serves as a powerful predictive tool. For instance, it can be applied to predicting housing prices and consumer spending. Additionally, linear regression can be used to analyze the impact of different features on the target variable, such as analyzing the effects of various factors on outcomes in economics and social sciences.

Although linear regression performs well in many scenarios, its performance may not be ideal when dealing with nonlinear relationships, highly complex datasets, or data with a significant amount of noise. Therefore, it is crucial to understand the nature and requirements of the data before selecting a linear regression model.

3.2 Logistic Regression Algorithms

Logistic regression is a generalized linear regression analysis model that estimates the probability of an event occurring based on a given set of independent variable data. Since the result is a probability, the range of the dependent variable is between 0 and 1.

The core principle of logistic regression is to utilize a logistic function to map the output of a linear regression model to a value between 0 and 1. This mapping enables logistic regression to estimate the probability of an event occurring and perform classification accordingly.

In logistic regression problems, we typically employ the Sigmoid function model, which takes the form of (Zou, 2019):

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

Here, Z is typically a linear function, such as $Z = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_n x_n$, where $\beta_0, \beta_1, \beta_2 \ldots \beta_n$ are the model parameters, and $x_1, x_2, \ldots x_n$ are the features.

Based on this, we can construct a logistic regression model, which takes the form of:

$$P(Y = 1|X) = \sigma(\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_n x_n)$$

Where $P(Y = 1|X)$ represents the probability of the target variable Y being 1 given the feature X.

The logistic regression model possesses the following advantages:

(1) Probabilistic interpretation: Logistic regression not only provides classification results but also gives the probability of belonging to a certain category, which is very useful in many applications.

(2) Computational efficiency: Logistic regression typically boasts high computational efficiency, particularly when dealing with large datasets.

(3) High interpretability: Compared to some complex machine learning models, the results of logistic regression are easier to interpret.

(4) Broad applicability: Logistic regression can be extended to multi-class classification problems and has applications in many different fields.

Logistic regression models are commonly used to handle binary classification problems, such as spam detection, disease diagnosis, credit application approval, etc. Logistic regression can also be used to estimate probabilities, making it highly applicable in scenarios where the probability of an event needs to be estimated, such as banks calculating customer default probabilities. Furthermore, logistic regression performs well when there is a linear relationship between features and classification outcomes.

Although logistic regression performs well in the aforementioned problems, its effectiveness may be compromised when dealing with nonlinear relationships, complex interactions between features, or very high-dimensional data. In these cases, more complex models such as Random Forests, Support Vector Machines, or Neural networks may need to be considered.

3.3 Decision Tree Algorithms

Decision tree is a common type of machine learning method. Since the decision branches are drawn like the branches of a tree, it is called a decision tree. (Zhou, 2021) In machine learning, a decision tree is a predictive model that represents a mapping relationship between object attributes and object values.

The original decision tree algorithm, known as the Concept Learning System (CLS), was proposed by psychologist

and computer scientist E. B. Hunt in 1962 during his research on human conceptual learning processes. This algorithm established the "divide and conquer" learning strategy for decision trees. (Hunt, 1962)

The core principle of a decision tree is to decompose a complex decision-making process into a series of simpler decisions, thereby forming a tree-like structure. When constructing a decision tree, the algorithm selects the optimal feature for splitting, so as to clearly divide the dataset at each node. This process is repeated until a certain stopping criterion is met, such as the maximum depth of the tree or the minimum number of samples in a node.

It recursively partitions the feature space into mutually exclusive rectangular regions, with each region corresponding to a final prediction result. A decision tree consists of three types of nodes: the root node (representing the entire dataset), internal nodes (representing feature tests), and leaf nodes (representing final predictions). The core learning logic of the algorithm is to select the optimal feature for splitting at each node, maximizing the purity of the split child nodes, that is, ensuring that the samples within the same node belong to the same category as much as possible. (Li, 2024)

The mainstream decision tree algorithm employs three core splitting criteria:

(1) Entropy and Information Gain (ID3 Algorithm)

Entropy is used to measure the purity or uncertainty of a dataset. (Hssina, 2014) For a dataset D containing K categories, the definition of entropy is:

$$H(D) = -\sum_{i=1}^{K} p_i \, log_2(p_i)$$

Where p_i denotes the proportion of the i-th category of samples in the set D.

Information gain (IG) measures the decrease in entropy of the dataset after splitting based on feature A:

$$IG(D, A) = H(D) - H(D|A)$$

The ID3 algorithm selects the feature with the highest information gain for splitting.

(2) Information Gain Ratio (C4.5 Algorithm)

To address the issue of ID3's bias towards features with a large number of values, C4.5 employs the information gain ratio to normalize the information gain based on the inherent value of the feature (Hssina, 2014):

$$IGR(D, A) = \frac{IG(D, A)}{H_A(D)}$$

Where $H_A(D)$ represents the entropy of the dataset with respect to the values of feature A.

(3) Gini coefficient (CART algorithm)

Classification and Regression Tree (CART) uses the Gini coefficient to measure the probability of a randomly selected sample being misclassified:

$$Gini(D) = 1 - \sum_{i=1}^{K} p_i^2$$

CART is a binary tree algorithm that can handle both classification and regression tasks simultaneously. To prevent overfitting, decision trees often employ pruning techniques (either pre-pruning or post-pruning) to remove unnecessary branches. (Chipman, 1998)

The decision tree model possesses the following advantages:

(1) High interpretability: The results of decision trees are easy to understand and interpret, and even non-professionals can grasp the decision-making path.

(2) No need for data standardization: Decision trees do not have strict requirements for data standardization or normalization, unlike some other algorithms.

(3) Can handle nonlinear relationships: Decision trees can effectively handle nonlinear relationships between data.

(4) Can handle mixed-type data: Decision trees can handle both numerical and categorical features simultaneously.

Decision trees perform well in handling binary or multi-class classification problems and regression tasks involving continuous value prediction. Additionally, during its construction, a decision tree selects the most important features, making it suitable for feature selection. Furthermore, for datasets with complex relationships between features, decision trees can capture these relationships as well.

Despite the numerous advantages of decision trees, they are prone to overfitting, especially when the tree becomes deep or complex. To prevent overfitting, pruning or limiting the depth of the tree is often necessary. Furthermore,

ensemble methods based on decision trees, such as random forests and gradient boosting trees, can further enhance the performance and generalization ability of the model.

3.4 Random Forest Algorithms

Random Forest (RF) belongs to the category of ensemble learning. It improves the accuracy and stability of predictions by combining multiple decision trees.

Leo Breiman is the primary proponent and founder of the Random Forest algorithm. Together with Adele Cutler, he refined and popularized this algorithm, making it one of the most classic and widely used ensemble learning models in the field of machine learning (Breiman, 2001).

RF is an extended variant of Bagging. On the basis of constructing a Bagging ensemble with decision trees as base learners, RF further introduces random attribute selection during the training process of decision trees. Specifically, when selecting a splitting attribute, a traditional decision tree chooses the optimal attribute from the attribute set (assuming there are d attributes) of the current node; In RF, for each node of the base decision tree, a subset containing k attributes is randomly selected from the attribute set of that node, and then an optimal attribute is chosen from this subset for partitioning. The parameter k here controls the degree of randomness introduced: If we let $k = d$, the construction of a decision tree based on rules is the same as that of a traditional decision tree; If we $k = 1$, then a random attribute is selected for partitioning. Generally, the recommended value is $k = log_2$ d (Breiman, 2001).

Random forest effectively reduces the correlation between base learners through randomness at two levels: sample randomness (bootstrap sampling) and feature randomness (random attribute selection), thereby significantly enhancing the generalization ability of the ensemble model. The following provides a brief introduction to these two aspects.

(1) Bootstrap Sampling

Each base decision tree in a random forest is trained based on an independent bootstrap sampling dataset. For the original dataset D containing N samples, bootstrap sampling generates a new training set D_i by randomly drawing N samples with replacement.

In the process of bootstrap sampling, the probability that a sample remains unselected throughout N rounds of drawing is:

$$\lim_{N\to\infty}(1-\frac{1}{N})^N = \frac{1}{e} \approx 0.368$$

This means that the training set of each base decision tree contains approximately 63.2% of the samples from the original dataset, and the remaining 36.8% of samples, known as Out-of-Bag (OOB) samples, can be used to estimate the model's generalization error.

(2) Integrated prediction mechanism

For classification tasks, the majority voting method is adopted, where the category with the most votes among the prediction results of all base decision trees is taken as the final prediction result:

$$f(x) = \text{argmax}_{c\in C}\Sigma_{i=1}^{T} I(h_i(x) = c)$$

Where T is the number of decision trees, $h_i(x)$ is the prediction of the i-th decision tree for sample x, $I(\cdot)$ is an indicator function, and C is the set of all categories.

For regression tasks, the simple average method is adopted, where the average of the prediction results from all base decision trees is taken as the final prediction result:

$$f(x) = \frac{1}{T}\Sigma_{i=1}^{T} h_i(x)$$

(3) Out-of-Bag Error (OOB Error)

Random forests do not require an additional validation set and can directly estimate the generalization error through out-of-bag samples. For each sample x_j , all base decision trees that do not use it as a training sample are used for prediction, and then the error rate of these predictions is calculated, which is the out-of-bag error:

$$\text{OOB Error} = \frac{1}{N}\sum_{j=1}^{N} I(f_{OOB}(x_j) \neq y_j)$$

Where $f_{OOB}(x_j)$ is the prediction result of x_j based on out−of−bag samples, and y_j is the true label of x_j.

Random forest has the following advantages:

(1) Strong generalization ability: By integrating multiple decision trees, the problem of overfitting in a single decision tree is effectively solved, and the generalization performance is significantly better than that of a single decision tree;

(2) High robustness: insensitive to outliers and noisy data, and stable performance on complex datasets;

(3) Low requirements for data preprocessing: It inherits the advantages of decision trees, eliminating the need for data normalization or standardization;

(4) Parallel computation: The training processes of individual base decision trees are completely independent, allowing for efficient parallelization and making them suitable for handling large-scale datasets;

(5) Built-in feature importance evaluation: It can automatically calculate the contribution of each feature to the prediction results.

Random forest is one of the most commonly used machine learning algorithms in the industry, widely applied to various classification and regression tasks, such as customer churn prediction, credit scoring, disease diagnosis, image classification, and recommendation systems. Random forest excels in various classification and regression tasks. At the same time, due to its efficiency, random forest is also suitable for handling large-scale datasets. Similarly, random forest can identify important features, which is very effective for feature selection. In addition, random forest can handle complex nonlinear relationships between features.

The main drawback of Random Forest is that the model may become relatively complex, which may result in less interpretability compared to a single Decision Tree. Furthermore, although Random Forest typically has a fast training speed, the required computational resources may increase when dealing with extremely large datasets. Nevertheless, Random Forest remains a highly powerful and popular algorithm in machine learning.

3.5 Support Vector Machine Algorithms

Vladimir Naumovich Vapnik and Alexey Chervonenkis jointly developed the Vapnik-Chervonenkis theory between 1960 and 1990, laying the foundation for statistical learning theory. He is the primary inventor of the Support Vector Machine (SVM), having first proposed the original algorithm in 1964, and in the 1990s, he and his colleagues introduced the kernel trick and soft-margin SVM. (Vapnik, 1999)

For a given training sample set $D = \{(x_1, y_1), (x_2, y_2), \dots, (x_m, y_m)\}, y_i \in \{-1, +1\}$, the fundamental idea of classification learning is to find a dividing hyperplane in the sample space based on the training set D, separating samples of different categories. (Zhou,2021) The core idea of Support Vector Machine (SVM) is to seek an optimal hyperplane in the feature space, separating samples of different categories, and maximizing the minimum distance (i.e., the margin) from the hyperplane to the samples of both categories.

The core advantage of SVM lies in its ability to maximize the margin, thereby ensuring the model's generalization ability, even on small sample datasets. Based on the linear separability of data, SVM is mainly divided into three categories: linearly separable SVM (hard margin), linear SVM (soft margin), and nonlinear SVM (kernel trick). This article focuses on the most basic linearly separable SVM and briefly explains the other two extended forms:

(1) Linearly separable SVM (hard margin)

For a linearly separable binary classification dataset, there exist an infinite number of hyperplanes that can separate the two classes of samples. However, SVM selects the hyperplane with the largest margin. The mathematical expression for the hyperplane is:

$$w^T x + b = 0$$

Where $w = (w_1; w_2; \dots; w_d)$ is the normal vector, determining the direction of the hyperplane; And b is the displacement term, determining the distance between the hyperplane and the origin. Clearly, the dividing hyperplane can be determined by the normal vector w and the displacement b, which we denote as (w, b) below, The distance r from any point x in the sample space to the hyperplane (w, b) can be written as:

$$r = \frac{|w^T x + b|}{\|w\|}$$

The optimization objective of SVM is to maximize the margin $\frac{2}{\|w\|}$, which is equivalent to minimizing $\frac{1}{2}\|w\|^2$, while satisfying the constraint conditions:

$$y_i(w^T x_i + b) \geq 1, i = 1,2, \dots, N$$

Where $y_i \in \{-1,1\}$ is the label of the sample.

(2) Extended form

The first extension is the soft-margin SVM, which is designed to address linearly inseparable problems. It

introduces slack variables $\xi_i \geq 0$, allowing some samples to violate the margin constraint, while controlling the degree of penalty through the penalty parameter C.

The second extension form is nonlinear SVM, which maps low-dimensional linearly inseparable data to a high-dimensional feature space through a kernel function, making it linearly separable in the high-dimensional space. Commonly used kernel functions include linear kernel, polynomial kernel, and Gaussian kernel (RBF):

$$K(x_i, x_j) = \phi(x_i) \cdot \phi(x_j)$$

Where $\phi(\cdot)$ is a mapping function, and the kernel function avoids explicit computation of high-dimensional mappings, greatly reducing computational complexity.

SVM has the following advantages:

(1) Effectiveness: In high-dimensional spaces, especially when the boundaries between categories are clear, SVM performs very well.

(2) Strong generalization ability: SVM attempts to maximize the margin, thus it usually has good generalization ability.

(3) Scalability: By selecting an appropriate kernel function, SVM can effectively handle nonlinear problems.

(4) Strong adaptability: SVM can be used to solve various types of machine learning problems.

Based on these advantages, SVM has a wide range of applications. Initially designed to solve binary classification problems, especially when dealing with high-dimensional data, SVM can effectively handle nonlinear data through the use of kernel tricks. It performs exceptionally well when dealing with high-dimensional data such as text and images. Furthermore, although SVM is primarily used for classification, with appropriate modifications, it can also be used for regression (known as Support Vector Regression, SVR).

One of the main drawbacks of SVM is its low computational efficiency for large-scale datasets, as its training process requires a long time.

In addition, selecting appropriate kernel functions and parameters (such as regularization parameters and kernel parameters) is crucial for achieving good performance, but this often requires professional knowledge and experimental tuning.

3.6 K-Nearest Neighbor Algorithms

The k-nearest neighbor algorithm is one of the most intuitive and easily understandable machine learning algorithms, belonging to the categories of instance-based learning and lazy learning. It was originally proposed by Cover and Hart. (Cover, 1968) Unlike most algorithms that require training the model before making predictions, kNN does not have an explicit training process. It stores all training data directly and performs calculations only during prediction. kNN is simple and straightforward: given a training dataset, for a new input instance, find the k nearest instances to it in the training dataset. If the majority of these k instances belong to a certain class, then classify the input instance into that class. (Li, 2021)

Typically, “Majority Voting” can be employed in classification tasks, which involves selecting the category label that appears most frequently among the k samples as the prediction result; In regression tasks, the “average method” can be employed, which involves using the average of the real-valued output labels of these k samples as the prediction result. Alternatively, weighted average or weighted voting can be conducted based on the distance, with samples closer to the query having a higher weight.

The core of kNN lies in calculating the “similarity” between samples, with the most commonly used similarity measurement method being the distance function. Below, we introduce three of the most frequently used distance formulas:

(1) Euclidean Distance: This is the most commonly used distance metric, used to calculate the linear distance between two samples in an n-dimensional space. The mathematical formula is:

$$\mathrm{d}(x_i, x_j) = \sqrt{\sum_{k=1}^{n} (x_{ik} - x_{jk})^2}$$

Where $x_i = (x_{i1}, x_{i2}, \dots, x_{in})$ and $x_j = (x_{j1}, x_{j2}, \dots, x_{jn})$ are two n-dimensionalsample vectors.

(2) Manhattan Distance: Manhattan distance, also known as city block distance, is used to calculate the sum of the absolute distances between two samples in each dimension. The mathematical formula is:

$$d(x_i, x_j) = \sum_{k=1}^{n} |x_{ik} - x_{jk}|$$

(3) Minkowski Distance: The general form of Euclidean distance and Manhattan distance:

$$d(x_i, x_j) = \left(\sum_{k=1}^{n} |x_{ik} - x_{jk}|^p\right)^{\frac{1}{p}}$$

When $p = 1$, it degenerates to the Manhattan distance; when $p = 2$, it degenerates to the Euclidean distance.

When facing different task types, different methods need to be adopted. Below, we introduce the selection of methods for classification tasks and regression tasks respectively:

(1) Classification problem:

There are primarily two methods for classification problems:

The first is the majority voting method, which is the most basic classification rule. The prediction result is the category with the highest frequency among the K nearest neighbors. The mathematical formula is expressed as:

$$\hat{y} = argmax_{c \in C} \sum_{i=1}^{K} I(y_i = c)$$

Where C is the set of all categories, and $I(\cdot)$ is an indicator function that takes the value of 1 when the condition inside the parentheses holds, and 0 otherwise.

The second method is Weighted Voting. To reflect that samples closer in distance have higher importance, weights can be assigned based on the reciprocal of the distance. The mathematical formula is expressed as:

$$\hat{y} = \text{argmax}_{c \in C} \sum_{i=1}^{K} \frac{1}{\text{d}(x, x_i)^2} I(y_i = c)$$

(2) Regression problem:

There are primarily two methods for regression problems:

The first method is the average method, where the predicted result is the arithmetic mean of the values of K nearest neighbor samples. The mathematical formula is:

$$\hat{y} = \frac{1}{k} \Sigma_{i=1}^{k} y_i$$

The second method is the weighted average method, where we can also apply weights based on distance:

$$\hat{y} = \frac{\sum_{i=1}^{k} \frac{1}{d(x, x_i)^2} y_i}{\sum_{i=1}^{k} \frac{1}{\text{d}(x, x_i)^2}}$$

The K value is the most crucial hyperparameter in the kNN algorithm, thus the selection of the K value is extremely important:

(1) When the value of K is too small, the model complexity is high, prone to overfitting, and sensitive to noise and outliers;

(2) When the value of K is too large, the model becomes overly simplistic, prone to underfitting, and may consider samples that are far apart;

(3) In practical applications, the optimal value of K is typically selected through cross validation.

kNN has the following advantages:

(1) Simple and intuitive: The kNN algorithm is straightforward and easy to understand, and its implementation is relatively straightforward as well.

(2) No training required: kNN is an instance-based learning method that does not require an explicit training process.

(3) Wide applicability: It can be used for classification and regression problems, as well as multi-class problems.

(4) Adaptability: Since kNN does not rely on assumptions, it is generally effective for different types of datasets.

Based on these advantages, we can apply kNN to classification problems involving small datasets, as kNN performs well in classification tasks when the dataset is not particularly large. At the same time, kNN can also

provide effective solutions for regression problems at the basic level. In addition, as it does not require training, it is suitable for applications that require real-time decision-making. In recommendation systems, kNN can also be used to identify similar users or similar items to provide personalized recommendations.

One major drawback of the kNN algorithm is its high computational cost, especially when dealing with large-scale datasets. Furthermore, selecting appropriate K values and distance metrics is crucial for the performance of the algorithm. In addition, k NN is sensitive to noise and irrelevant features in the data. Despite these limitations, kNN is still favored in many practical applications due to its simplicity and ease of implementation.

3.7 Naive Bayes Algorithms

The Naive Bayes algorithm is a classification method based on the Bayes theorem and the assumption that features are conditionally independent of each other. For a given training dataset, the joint probability distribution between inputs and outputs is learned under the assumption of conditional independence. Then, using this model, for a given input x, the output y with the highest posterior probability is determined using the Bayes theorem. The Naive Bayes approach is simple to implement, and both learning and prediction accuracy are quite high. Therefore, it is a commonly used method. (Li, 2024)

Bayes' theorem is the foundation of the Naive Bayes algorithm. The first documented mention of Bayes' theorem can be found in an article titled *An Essay towards Solving a Problem in the Doctrine of Chance* written by Richard Price, a close friend of Thomas Bayes. This article was published after Price's death in 1763. Bayes' theorem forms the core of the Naive Bayes algorithm. It describes the relationship between conditional probabilities, and is used to calculate the probability of an event occurring under certain conditions. The mathematical formula for this theorem is as follows:

$$P(Y|X) = \frac{P(X|Y)P(Y)}{P(X)}$$

Here, $P(Y|X)$ is the posterior probability, which is the probability that a sample belongs to category Y given the feature X. $P(X|Y)$ is the likelihood probability, which is the probability that a feature X appears given that the category Y is known. $P(Y)$ is the prior probability, which is the probability that category Y appears in the training data. $P(X)$ is the evidence factor; it remains the same for all categories, so it can be ignored during classification.

Based on the assumption of independent feature conditions, the likelihood probability can be expressed as the product of individual feature condition probabilities:

$$P(X|Y = y_k) = \prod_{i=1}^{d} P(x_i|Y = y_k)$$

Here, d represents the number of features, and x_i represents the value of the i-th feature.

Therefore, the final decision-making rule for Naive Bayes is as follows:

$$y = \text{argmax}_{y_k} P(y_k) \prod_{i=1}^{d} P(x_i|Y = y_k)$$

Depending on the type of features, there are three main variants of the Naive Bayes algorithm:

(1) Gaussian Naive Bayes: Suitable for continuous features. It assumes that the features follow a normal distribution.

(2) Multinomial Naive Bayes: Suitable for discrete features, especially useful in text classification tasks.

(3) Bernoulli Naive Bayes: Suitable for binary features.

Naive Bayes has the following advantages:

(1) Extremely high computational efficiency: Both training and prediction processes are carried out very quickly. The time complexity is $O(Nd)$.

(2) Friendly to small datasets: Can still achieve good performance even when the amount of training data is limited.

(3) Robust against high-dimensional data: Particularly suitable for processing high-dimensional and sparse data such as text classification.

(4) Simple to implement: The algorithm's logic is clear and easy to understand and implement.

(5) Not sensitive to missing values: Can still function properly even when some data points are missing.

When dealing with actual problems, it's important to note that when there is strong correlation between features, the assumption of independence of features no longer holds. This is the most fundamental limitation of the algorithm. In such cases, the performance of the Naive Bayes algorithm significantly declines. Additionally, the

Naive Bayes algorithm is very sensitive to the form in which the input data is represented. Therefore, it's necessary to convert the features into a form that can be represented as probability distributions.

The most classic application of Naive Bayes is text classification, such as spam detection, sentiment analysis, news classification, etc. At the same time, it is also widely used in medical diagnosis, spam message filtering, recommendation systems and other fields. Naive Bayes is a very good choice in scenarios where a large number of high-dimensional data need to be processed quickly and the precision is not very high.

3.8 Adaptive Boosting Algorithms

Adaptive Boosting (AdaBoost) is one of the most classic representative algorithms in the field of Boosting-based ensemble learning. Boosting refers to a family of algorithms that can transform weak learners into strong learners. The working mechanism of this family of algorithms is as follows: First, a base learner is trained using the initial training dataset. Then, the distribution of training samples is adjusted based on the performance of the base learners. This adjustment ensures that the training samples that were misclassified by previous base learners receive more attention in subsequent training steps. Finally, the next base learner is trained using the adjusted distribution of samples. This process is repeated until the number of base learners reaches a predetermined value T. At that point, the T base learners are combined together using a weighted combination method. (Zhou, 2021)

The most famous representative of the AdaBoost family of algorithms is AdaBoost. (Freund & Schapire, 1997) AdaBoost differs from the parallel training methods used in Random Forest algorithms. AdaBoost employs serial training techniques, continuously correcting the errors made by previous learners in order to improve overall performance. It is the first Boosting algorithm that has been proven to have theoretical guarantees, and it also demonstrates excellent generalization capabilities in practice. The training process of AdaBoost can be divided into the following key steps:

(1) Initialization of sample weights: For a training set containing N samples, initially, the weights of each sample are equal.

$$w_{1i} = \frac{1}{N}, i = 1,2, \dots, N$$

(2) Iterative training of weak learners: For the t-th iteration:

Firstly, a weak learner $h_t(x)$ is trained based on the current sample weights w_{ti}.

Next, calculate the classification error rate of this weak learner on the weighted samples:

$$\epsilon_t = \frac{\sum_{i=1}^{N} w_{ti} I(h_t(x_i) \neq y_i)}{\sum_{i=1}^{N} w_{ti}}$$

Then, the weights of the initial weak learner are calculated again:

$$\alpha_t = \frac{1}{2} ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$$

Finally, update the weights of the samples:

$$w_{t+1,i} = \frac{w_{ti}}{z_t} \exp\left(-\alpha_t y_i h_t(x_i)\right)$$

Here, Z_t represents the normalized factor, ensuring that the total weight of all samples equals 1.

(3) Final prediction: Combine all learners' predictions by weighting them according to their importance, and then use a symbolic function to output the final prediction result:

$$f(x) = \text{sign}(\sum_{t=1}^{T} \alpha_t \, h_t(x))$$

Here, T represents the total number of weak learners.

The AdaBoost algorithm has the following advantages:

(1) Good performance: AdaBoost generally improves the accuracy of weak learners, resulting in a model with higher accuracy.

(2) Easy to implement: Compared to some more complex algorithms, AdaBoost is relatively simple to implement.

(3) Automatic feature selection: During the training process, AdaBoost can identify more important features.

(4) Not too prone to overfitting: In most cases, AdaBoost has a certain ability to resist overfitting.

AdaBoost is widely used in various classification tasks due to its outstanding performance. It is particularly effective for binary classification and multi-class classification problems. Additionally, in terms of feature selection, AdaBoost can also help improve the overall performance of the model.

AdaBoost's main limitation is its sensitivity to noise and outliers, which can lead to a decline in model performance. Additionally, for certain highly complex problems, AdaBoost may not be as effective as other more advanced algorithms, such as random forests or deep learning methods.

3.9 K-Means Clustering Algorithm

K-Means is one of the most classic and widely used unsupervised clustering algorithms. Its main goal is to divide an unlabeled dataset into k non-overlapping clusters, where the similarity between samples within the same cluster is maximized, while the similarity between samples in different clusters is minimized. (MacQueen, 1967) Unlike supervised learning algorithms, K-Means does not require any pre-labeling of data. Instead, it automatically identifies clusters based on the inherent structure of the data itself. Its core idea is iterative optimization: initially, k samples are randomly selected as initial centroids. Then, each sample is assigned to the cluster closest to its corresponding centroid. This process is repeated until the centroids no longer change significantly or until a maximum number of iterations is reached.

The core of the K-Means clustering algorithm is to minimize the Sum of Squared Errors (SSE) between all samples and the centers of their respective clusters.

$$SSE = \sum_{i=1}^{k} \sum_{x \in C_i} \|x - \mu_i\|^2$$

Here, C_i represents the i-th cluster, while μ_i represents the centroid of the i-th cluster. The distance $\|x - \mu_i\|$ denotes the Euclidean distance between the sample x and the centroid μ_i.

The specific steps of the algorithm are as follows:

(1) Initialization: Randomly select k samples from the dataset as the initial centers, $\mu_1, \mu_{2,\dots}, \mu_k$.

(2) Sample allocation: For each sample x, calculate the distance from it to all centroids. Then, assign the sample to the cluster that is closest to it.

$$C_i = \{x | \|x - \mu_i\| \leq \|x - \mu_j\|, \forall j \neq \mathrm{i}\}$$

(3) Changing the centroid: Calculate the centroid of each cluster again, which is the average of all samples within that cluster.

$$\mu_i = \frac{1}{|C_i|} \Sigma_{x \in C_i} x$$

(4) Convergence judgment: Repeat steps 2 and 3 until the change in centroid is less than the preset threshold, or until the maximum number of iterations is reached.

In this process, the choice of k value is the most critical issue in the K-Means algorithm. The commonly used method is the elbow method: by plotting the SSE curves for different k values, the k value corresponding to the "elbow" point in the curve is selected.

K-Means clustering algorithm has the following advantages:

(1) Simple and easy to understand: The algorithm's logic is clear and easy to comprehend and implement.

(2) High computational efficiency: The time complexity is $O(Nkt)$, where N represents the number of samples, k represents the number of clusters, and t represents the number of iterations. This makes it suitable for processing large datasets.

(3) Strong interpretability: The clustering results are intuitive, making it easy to explain them to people who are not technical experts.

(4) Fast convergence speed: In most practical applications, the algorithm can quickly converge to the local optimal solution.

The K-Means clustering algorithm is widely used in various unsupervised learning tasks, such as customer segmentation, image segmentation, text clustering, anomaly detection, and gene expression analysis. It is particularly suitable for scenarios where the data structure is relatively simple, and the clusters are spherical in shape. It is one of the most commonly used tools during the data exploration and preprocessing stage.

3.10 Principal Component Analysis

Finally, we would like to introduce Principal Component Analysis as the conclusion. We chose PCA as the ending

method because it is the most classic and widely used unsupervised dimensionality reduction algorithm. PCA was first proposed by the British statistician Karl Pearson in 1901. He proposed the ideological framework of transforming relevant variables through orthogonal transformation, which is regarded as the theoretical prototype of PCA. In 1933, American mathematician Harold Hotelling developed it into a systematic mathematical method and formally named it “principal component analysis”, which laid a theoretical foundation for modern PCA. (Wold, 1987)

The core idea of PCA is to use orthogonal transformation to transform the observation data represented by linearly dependent variables into a few data represented by linearly independent variables. Linearly independent variables are called main components. The number of principal components is usually smaller than the number of original variables, so principal component analysis is a dimension reduction method. (Li, 2024)

The main purposes of dimensionality reduction include solving the curse of dimensionality, reducing the amount of computation and storage costs, removing noise and redundant information from data, and visualizing high-dimensional data into 2D or 3D space. PCA reduces dimensions by finding a set of orthogonal principal components, which are the directions with the largest variance in the data and are not related to each other. The core goal of PCA is to find a linear transformation matrix W, the original d-dimensional data X is projected into the k-dimensional space $(k < d)$, so that the variance of the projected data $Y = XW$ is maximized. (Hotelling, 1936)

The specific steps of the algorithm are as follows:

(1) Data standardization: perform zero-mean centering on the original data to ensure that the average value of each feature is 0:

$$x_i' = x_i - \mu$$

Where μ is the mean vector of the feature.

(2) Calculate the covariance matrix: calculate the covariance matrix of the standardized data:

$$\Sigma = \frac{1}{N-1} X^T X$$

Where N is the number of samples.

(3) Eigenvalue decomposition: eigenvalue decomposition is performed on the covariance matrix to obtain eigenvalues $\lambda_1 \geq \lambda_2 \ldots \geq \lambda_d$ and corresponding eigenvectors $v_1, v_2, \ldots, v_d$.

(4) Select the principal component: select the eigenvector corresponding to the first k largest eigenvalues to form a transformation matrix.

$$W = [v_1, v_2, \ldots, v_k]$$

(5) Data projection: project the original data into low dimensional space: $Y = XW$.

Selection of the number of principal components: usually select the main component quantity whose cumulative variance contribution rate reaches 85%-95%, and the Cumulative Variance Ratio is defined as:

$$\text{Cumulative Variance Ratio} = \frac{\sum_{i=1}^{k} \lambda_i}{\sum_{i=1}^{d} \lambda_i}$$

PCA has the following advantages:

(1) Noise reduction: PCA can help remove the noise in the data and retain the most important signal.

(2) Visualization: PCA helps to visualize data by reducing high-dimensional data to 2 or 3 dimensions.

(3) Decorrelation: PCA eliminates the correlation between data features through orthogonal transformation.

(4) Improve algorithm efficiency: The reduced dimension data can improve the computing efficiency of machine learning algorithm and reduce resource consumption.

PCA can be used as a preprocessing step before more complex machine learning tasks, such as applying classification or regression algorithms to high-dimensional data. At the same time, data visualization can be realized to help understand the internal structure of high-dimensional data. In addition, feature extraction can also be carried out in pattern recognition and signal processing to extract useful features and remove the noise part of the data to extract the main signal.

The main limitation of PCA is that it depends on linear assumptions, which may not be effective for nonlinear data structures. In addition, PCA is very sensitive to abnormal values, so it is necessary to carefully consider data cleaning and pre-processing before applying PCA.

4. Comparison and Selection Guide of Algorithms

To further clarify the characteristics and applicable boundaries of each algorithm, this section conducts a multi-dimensional horizontal comparison of the ten classic machine learning algorithms from seven core dimensions, including learning paradigm, task adaptability, interpretability, computational efficiency, data scale adaptability, robustness to outliers, and overfitting risk.

4.1 Algorithm Taxonomy and Task Applicability

The ten classic algorithms introduced above can be clearly divided into two categories according to their learning paradigms: supervised learning algorithms and unsupervised learning algorithms. Among them, supervised learning includes 8 algorithms, covering the two core tasks of machine learning: classification and regression. Linear Regression is the only basic algorithm dedicated to regression tasks; Logistic Regression and Naive Bayes are mainly applied to classification tasks; Decision Tree, Random Forest, Support Vector Machine (SVM), K-Nearest Neighbors (KNN) and AdaBoost are general-purpose algorithms that can address both classification and regression problems. Unsupervised learning consists of two algorithms corresponding to two core tasks respectively: K-Means serves as the industry standard for clustering tasks, while Principal Component Analysis (PCA) is the preferred tool for dimensionality reduction and data preprocessing. Such a clear task division constitutes the primary principle of algorithm selection, which directly defines the scope of alternative algorithms.

4.2 Interpretability Tiers and Scenario Fit

Algorithm interpretability is one of the core considerations in industrial applications, and the ten algorithms present a clear gradient distribution. Linear Regression, Logistic Regression, Decision Tree and KNN fall into the first tier with extremely high interpretability. The coefficients of linear models directly quantify the marginal impact of each feature on the outcome; rules generated by decision trees can be described directly in natural language; and the prediction results of KNN can be intuitively explained through nearest neighbor samples. Random Forest and K-Means belong to the second tier with moderate interpretability: Random Forest can output feature importance ranking, and the clustering results of K-Means can be interpreted via cluster center characteristics. SVM, AdaBoost and PCA are in the third tier with low interpretability, which are typical “black-box models” whose internal decision-making process is difficult to explain intuitively. (Jordan, 2015) In highly regulated fields such as medical diagnosis, financial risk control and judicial assistance, white-box models in the first tier must be prioritized; in scenarios such as recommendation systems and image preprocessing, black-box models can be adopted on the premise of accuracy priority.

4.3 Computational Efficiency and Resource Constraints

The computational efficiency of different algorithms varies greatly, which directly determines their applicability under different data scales. Naive Bayes boasts the highest computational efficiency, with extremely fast training and prediction speeds and almost no computational resource consumption. Linear Regression, Logistic Regression and K-Means follow closely, with fast training speeds and suitability for large-scale datasets. Decision Tree, PCA and Random Forest are at a medium level, among which Random Forest can further improve efficiency through parallel training. AdaBoost and SVM have relatively low computational efficiency, and their training time increases significantly with the growth of sample size. KNN is a special case: it requires almost no training process but has extremely slow prediction speed, since it needs to calculate the distance from each test sample to all training samples. In resource-constrained scenarios such as embedded devices and real-time inference systems, Naive Bayes and linear models should be given priority; for large-scale datasets with more than one million samples, SVM and KNN should be avoided. (Havenstein, 2018)

4.4 Robustness and Data Adaptability

Algorithms differ significantly in their adaptability to different data characteristics. SVM and Naive Bayes have strong adaptability to small-sample data, and can still achieve satisfactory generalization performance even when the number of samples is smaller than the feature dimension. Random Forest and AdaBoost perform better on large-sample datasets, as they can continuously improve performance by integrating more weak learners. In terms of robustness to outliers, Decision Tree and Random Forest perform the best, as their splitting mechanism is naturally insensitive to outliers. (Guetari, 2023) By contrast, Linear Regression, Logistic Regression, K-Means and PCA are highly sensitive to outliers, which can seriously distort the parameter estimation of the models. For high-dimensional sparse data such as text data, Naive Bayes and linear SVM have irreplaceable advantages; for non-linear data, tree models, ensemble models and kernel SVM deliver better performance.

4.5 Generalization Performance and Overfitting Risk

Overfitting is one of the core challenges in machine learning, and different algorithms have significantly different overfitting risks. Decision Tree has the highest overfitting risk, and an unpruned decision tree will almost certainly overfit the training data. Random Forest, by integrating multiple decision trees, effectively reduces the overfitting

risk and is recognized as one of the algorithms with the strongest generalization ability. (Halabaku, 2024) Logistic Regression, Naive Bayes and KNN have low overfitting risks and can maintain stable performance even on small-sample datasets. SVM and AdaBoost have a moderate level of overfitting risk, which can be effectively controlled through appropriate hyperparameter tuning. It should be specially noted that, as unsupervised learning algorithms, K-Means and PCA do not have the overfitting problem in the traditional sense.

4.6 Selection Principles and Scenario Recommendations

Based on the above multi-dimensional comparison, algorithm selection should follow the core priority of "determine the task first, then evaluate the data, weigh interpretability, and finally consider resources". In practical applications, Naive Bayes and linear SVM are the first choice for text classification tasks; K-Means is preferred for customer segmentation; Logistic Regression and Decision Tree are recommended for credit scoring and medical diagnosis; Linear Regression and Random Forest are suitable for regression tasks such as housing price prediction; SVM and Naive Bayes are optimal for small-sample classification tasks; and PCA is the primary option for data preprocessing and dimensionality reduction. For most general scenarios, Random Forest is an excellent default choice, as it achieves the best balance among accuracy, robustness, computational efficiency and interpretability.

5. Conclusion and Future Work

Through the systematic research on ten classic machine learning algorithms, this thesis fully elaborates the mathematical principles, implementation logic, advantages, limitations and application boundaries of each algorithm, and constructs a comprehensive, clear, hierarchical and practically instructive knowledge system of classic machine learning. These classic algorithms, which have undergone decades of theoretical verification and industrial practice, are still the preferred solutions for solving most practical problems today. The statistical learning ideas, optimization methods and modeling logic contained in them have not become obsolete with the development of technology, but are constantly glowing with new vitality in the integration with emerging technologies.

Looking forward to the future, the field of machine learning is in an unprecedented period of rapid development. Emerging technologies such as large language models and generative artificial intelligence are profoundly changing the pattern of the entire industry. (Zhao, 2026) In this context, classic algorithms will not be eliminated, but will play a more important role in the new technological ecosystem, presenting a trend of multi-dimensional integration and innovative development. First, the deep integration of classic algorithms and large models is one of the most active research directions at present. Large models have shown amazing capabilities in processing unstructured data such as text, images and audio, but still have obvious shortcomings in structured data processing, logical reasoning, interpretability and computational efficiency, which classic algorithms can just make up for.

Second, the rapid development of Explainable Artificial Intelligence (XAI) will further highlight the value of classic algorithms. With the wide application of artificial intelligence technology in high-risk fields such as medical diagnosis, financial risk control, autonomous driving and judicial assistance, the interpretability of models is no longer an optional additional attribute, but has become the core bottleneck determining whether the technology can be implemented. Classic algorithms themselves are the benchmark of interpretability. The coefficients of linear models and the rules of decision trees have clear physical meanings and can directly explain the basis of decisions to users and regulatory agencies. At the same time, the current mainstream interpretability methods, such as LIME, SHAP, and Anchors, are essentially based on the ideas of classic algorithms. (Adadi, 2018) They approximate the decision-making behavior of complex models by building a simple interpretable model (such as a linear model or a decision tree) locally in the complex model. In the future, classic algorithms will become the core pillar of the XAI field, promoting the transformation of artificial intelligence from "usable" to "trustworthy" and "reliable", and providing a guarantee for the healthy development and responsible application of artificial intelligence.

Third, the explosion of edge intelligence and lightweight deployment will bring new application scenarios for classic algorithms. With the popularization of Internet of Things technology and the construction of 5G networks, more and more intelligent applications need to be migrated from the cloud to edge devices, such as smartphones, wearable devices, industrial sensors, autonomous vehicles, etc. These edge devices usually have limited computing resources, limited power consumption, and unstable network connections, and cannot run complex deep learning models. Classic algorithms have become the preferred algorithms for edge intelligence due to their small computational load, low memory usage, simple deployment, and good real-time performance.

Fourth, the popularization of Automated Machine Learning (AutoML) will further amplify the value of classic algorithms. AutoML aims to complete the entire machine learning process through automated methods, including data cleaning, feature engineering, algorithm selection, hyperparameter tuning and model deployment, thereby greatly lowering the threshold for using machine learning and allowing more non-professionals to use machine

learning technology to solve practical problems. Classic algorithms are the core components of AutoML systems. (Ferreira, 2021) Whether it is the construction of fast baseline models, the surrogate models for hyperparameter search, or the evaluation of feature importance, they are inseparable from the support of classic algorithms.

Finally, the rise of privacy computing and federated learning provides a new direction for the development of classic algorithms. Today, when data security and privacy protection are receiving increasing attention, how to maximize the value of data without disclosing original data has become the focus of the industry. (Yin, 2021) Federated learning, as an emerging distributed machine learning paradigm, allows multiple participants to collaboratively train a global model without sharing original data, perfectly solving the contradiction between data silos and privacy protection. (Wen, 2023) Classic algorithms have become the preferred algorithms for federated learning due to their simple structure, easy distributed implementation, and low communication overhead. At present, technologies such as Federated Linear Regression, Federated Logistic Regression, Federated K-Means, and Federated Decision Trees have been widely used in finance, healthcare, government affairs and other fields. In the future, how to further optimize the performance and security of classic algorithms in federated scenarios will be an important research topic in the field of machine learning.

In summary, classic algorithms are not outdated technologies, but the eternal cornerstone of the field of machine learning. They have withstood the test of time, condensed the wisdom of countless researchers, and have unique advantages that deep learning cannot replace. In the future, classic algorithms will be deeply integrated and mutually promoted with emerging technologies such as large models, XAI, edge computing, AutoML, and privacy computing, jointly promoting the continuous advancement of artificial intelligence technology in a more reliable, efficient and inclusive direction, and providing continuous technical impetus for the digital transformation and intelligent upgrading of human society.

References

Adadi, A., & Berrada, M. (2018). Peeking inside the black-box: a survey on explainable artificial intelligence (XAI). *IEEE access*, *6*, 52138-52160.

Barlow, H. B. (1989). Unsupervised learning. *Neural computation*, *1*(3), 295-311.

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

Chipman, H. A., George, E. I., & McCulloch, R. E. (1998). Bayesian CART model search. *Journal of the American Statistical Association*, *93*(443), 935-948.

Cover, T. (1968). Estimation by the nearest neighbor rule. *IEEE Transactions on Information Theory*, *14*(1), 50-55.

Ezugwu, A. E., Ho, Y. S., Egwuche, O. S., Ekundayo, O. S., Van Der Merwe, A., Saha, A. K., & Pal, J. (2024). Classical machine learning: seventy years of algorithmic learning evolution. *arXiv preprint arXiv:2408.01747*.

Fahrmeir, L., Kneib, T., Lang, S., & Marx, B. D. (2022). Regression models. In *Regression: Models, methods and applications* (pp. 23-84). Berlin, Heidelberg: Springer Berlin Heidelberg.

Ferreira, L., Pilastri, A., Martins, C. M., Pires, P. M., & Cortez, P. (2021, July). A comparison of AutoML tools for machine learning, deep learning and XGBoost. In *2021 International joint conference on neural networks (IJCNN)* (pp. 1-8). IEEE.

Freund, Y., & Schapire, R. E. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, *55*(1), 119-139.

Guetari, R., Ayari, H., & Sakly, H. (2023). Computer-aided diagnosis systems: a comparative study of classical machine learning versus deep learning-based approaches. *Knowledge and Information Systems*, 1.

Halabaku, E., & Bytyçi, E. (2024). Overfitting in Machine Learning: A Comparative Analysis of Decision Trees and Random Forests. *Intelligent Automation & Soft Computing*, *39*(6).

Havenstein, C., Thomas, D., & Chandrasekaran, S. (2018). Comparisons of performance between quantum and classical machine learning. *SMU Data Science Review*, *1*(4), 11.

Hotelling, H. (1936). Simplified calculation of principal components. *Psychometrika*, *1*(1), 27-35.

Hssina, B., Merbouha, A., Ezzikouri, H., & Erritali, M. (2014). A comparative study of decision tree ID3 and C4. 5. *International Journal of Advanced Computer Science and Applications*, *4*(2), 13-19.

Hunt, D. E. (1966). A conceptual systems change model and its application to education. In *Experience Structure & Adaptability* (pp. 277-302). Berlin, Heidelberg: Springer Berlin Heidelberg.

Jordan, M. I., & Mitchell, T. M. (2015). Machine learning: Trends, perspectives, and prospects. *Science*, *349*(6245), 255-260.

Li, H., Lin, L., & Zeng, H. (2024). *Machine learning methods* (pp. 1-532). Singapore: Springer.

Liu, B. (2011). Supervised learning. In *Web Data Mining: Exploring Hyperlinks, Contents, and Usage Data* (pp. 63-132). Berlin, Heidelberg: Springer Berlin Heidelberg.

MacQueen, J. (1967). Multivariate observations. In *Proceedings of the 5th Berkeley symposium on mathematical statistics and probability* (Vol. 1, pp. 281-297). Oakland, CA, USA: University of California press.

Michalski, R. S., Carbonell, J. G., & Mitchell, T. M. (Eds.). (2013). *Machine learning: An artificial intelligence approach*. Springer Science & Business Media.

Montgomery, D. C., Peck, E. A., & Vining, G. G. (2021). *Introduction to linear regression analysis*. John Wiley & Sons.

Rupp, A. A., & Templin, J. L. (2008). Unique characteristics of diagnostic classification models: A comprehensive review of the current state-of-the-art. *Measurement*, *6*(4), 219-262.

Samuel, A. L. (1959). Some Studies in Machine Learning Using the Game of Checkers. *IBM Journal of Research and Development*, *3*(3), 535-554.

Toosi, A., Bottino, A. G., Saboury, B., Siegel, E., & Rahmim, A. (2021). A brief history of AI: how to prevent another winter (a critical review). *PET clinics*, *16*(4), 449-469.

Vapnik, V. N. (1999). An overview of statistical learning theory. *IEEE transactions on neural networks*, *10*(5), 988-999.

Wen, J., Zhang, Z., Lan, Y., Cui, Z., Cai, J., & Zhang, W. (2023). A survey on federated learning: challenges and applications. *International journal of machine learning and cybernetics*, *14*(2), 513-535.

Wold, S., Esbensen, K., & Geladi, P. (1987). Principal component analysis. *Chemometrics and intelligent laboratory systems*, *2*(1-3), 37-52.

Wu, Y. (1999). Statistical learning theory.

Yin, X., Zhu, Y., & Hu, J. (2021). A comprehensive survey of privacy-preserving federated learning: A taxonomy, review, and future directions. *ACM Computing Surveys (CSUR)*, *54*(6), 1-36.

Zhao, Wayne Xin, et al. (2026). A survey of large language models. *Frontiers of Computer Science*, *20*, 2012627.

Zhou, Z. H. (2021). *Machine learning*. Springer nature.

Zou, X., Hu, Y., Tian, Z., & Shen, K. (2019, October). Logistic regression model optimization and case analysis. In *2019 IEEE 7th international conference on computer science and network technology (ICCSNT)* (pp. 135-139). IEEE.