Machine Learning
Supervised Learning - Classification

Vikas Thammanna Gowda

02/27/2025

1 Classification

1.1 What is Classification?

Classification is a branch of supervised learning focused on assigning discrete labels or categories to input data. In a classification task, the model is trained on a dataset of labeled examples (where each input has a known category) and learns to predict the correct label for new, unseen instances. Unlike regression, which predicts a continuous numeric value, classification predicts a categorical outcome a definite class or category. Fundamentally, classification is about predicting a label (e.g. "spam" or "not spam"), whereas regression is about predicting a quantity (e.g. the exact price of a house). For example, determining whether an email is spam is a classification problem (output is "spam" or "not spam"), while estimating the dollar value of a house is a regression problem (output is a continuous price).

Classification plays a crucial role in machine learning and is one of the most widely used techniques. Many real-world decisions are essentially classification problems, making it highly important. In fact, classification is often one of the first ML tasks a practitioner learns and serves as a foundation for tackling more complex problems later. A broad array of real-world applications rely on classification: email spam filtering, medical diagnosis, fraud detection, image and facial recognition, speech recognition, customer churn prediction, loan approval decisions, and more. For instance, a bank may use classification to decide if a credit card transaction is fraudulent or legitimate, an autonomous car classifies objects on the road (pedestrian, vehicle, sign, etc.), and a marketing team might classify customers as likely to churn or not. In all these cases, the goal is to have the model categorize inputs into the correct discrete class based on learned patterns.

1.2 How Classification Works after Preprocessing

- **Data Preprocessing:** The initial stage includes cleaning and transforming the data. Key tasks are: handling missing values (e.g. filling with mean or using interpolation), removing or capping outliers, and ensuring consistency in data. Feature scaling is often applied so that features on different scales (e.g. age in years vs. income in dollars) are normalized or standardized to a common range. This prevents algorithms from being unduly influenced by larger-scale features. For example, one might apply standardization to convert all features to zero mean and unit variance. If the dataset contains categorical variables, these may be encoded into numeric form (e.g. one-hot encoding for nominal categories or ordinal encoding for ordered categories). All these preprocessing techniques (cleaning, scaling, encoding) help improve model performance and training stability. Finally, the data is typically split into training and test sets (and sometimes a validation set) so that model performance can be evaluated on unseen data after training.

- **Input Data:** We begin with a preprocessed dataset ready for modeling. Each instance in the dataset is described by a set of features (independent variables) along with a target label (the category for that instance). These features can be of various types – numeric (e.g. height, temperature), categorical (e.g. color, brand), ordinal, or even more complex representations like

text embeddings or image pixels. For example, in a medical dataset, features might include a patient's blood test results and symptoms, and the label might be the diagnosis (disease A, disease B, or healthy). In an image classification task, the input features are pixel values or extracted image features, and the label could be the object in the image. The quality and relevance of input features are critical, which is why feature engineering and selection are often important steps before training.

- **Training Phase:** During training, the classification algorithm learns a mapping between the input features and the output labels. The model adjusts its internal parameters to find patterns and relationships that distinguish the classes. Depending on the complexity of the data and the problem, different types of models can be used: decision trees, logistic regression, neural networks, among others.
- **Prediction Phase:** Once the model has been trained (i.e., the learning algorithm has adjusted the model parameters based on training data), we use the model to classify new, unseen inputs. For a given new sample, the model computes output scores or probabilities for each possible class.
- **Evaluation:** The model's accuracy is assessed using metrics like accuracy, precision, recall, and F1-score.

1.3 Types of Classification

1. **Binary Classification**
   In binary classification, there are only two possible outcome classes for each input. One class is often considered the "positive" outcome and the other the "negative" outcome (this designation is arbitrary but helps in discussing metrics like precision and recall).
   Example: spam vs. not spam email classification, disease vs. no disease diagnosis, pass vs. fail outcome, or an IoT sensor reading being "normal" vs "anomaly". Many real-world decisions are binary by nature – for instance, a bank loan application can be approved or rejected, a medical test result can be positive or negative, a customer can churn (leave) or stay. Binary classification is the simplest case and is very well-studied. A lot of algorithms (like logistic regression or support vector machines) were initially designed for binary problems.
   When to use: Obviously, whenever the naturally occurring categories are two.
   Challenges: Even though conceptually simple, binary classification can be challenging if the two classes are highly imbalanced or overlapping in feature space. For example, fraud detection is a binary classification where fraud cases are rare – a model that simply predicts "not fraud" for everything will be correct most of the time (high accuracy) but is useless for detecting fraud. Thus, careful attention to metrics and algorithms (or sampling techniques) is needed in such scenarios.

2. **Multi-Class Classification**
   In multi-class classification, there are more than two possible classes, and each sample is assigned to exactly one of these classes. The classes are mutually exclusive in the sense that a single instance cannot belong to more than one class simultaneously (in contrast to multi-label, below).
   Example: Recognizing handwritten digits 0–9 (10 classes) , classifying an email as either Personal, Work, Promotions, or Spam (4 classes), or identifying the species of an animal in a photo (cat, dog, bird, etc.). Another example is sentiment analysis that classifies text into categories like Positive, Neutral, or Negative sentiment (3 classes). Multi-class problems are very common; the Iris flower classification (Setosa, Versicolor, Virginica) is a classic 3-class example, and image recognition tasks like classifying images from the CIFAR-10 dataset (which has 10 classes such as airplane, car, bird, etc.) or the very large ImageNet dataset (1000 object categories) are well-known multi-class problems.
   When to use: Any time the outcome can fall into one of N distinct categories ($N > 2$) and each instance has exactly one category.
   Challenges: Multi-class classification is generally more complex than binary because the model must distinguish among multiple classes, not just two. Some algorithms (like naive Bayes, decision

trees, neural networks) handle multi-class inherently. Others, like binary logistic regression or SVM, need to be extended (e.g. via one-vs-rest or one-vs-one schemes) to tackle multiple classes. Evaluation is also more involved: accuracy might still be used, but one must look at per-class performance (since a model might do well on some classes and poorly on others). Imbalanced class distribution (some classes having far more examples than others) can again bias the model. Another challenge is error analysis – if the model confuses Class A with B, that might be more acceptable than confusing A with C, depending on the problem (e.g., misclassifying an "airplane" as "bird" vs. as "dog" in an image classifier might have different implications). Techniques like confusion matrices are extended to multi-class to analyze such errors.

2 Model Evaluation Metrics

Evaluating a classification model involves understanding how its predictions compare to the true labels. A common tool for this is the confusion matrix, which tabulates the outcomes of predictions versus actual values. For binary classification, the confusion matrix is a 2x2 table with the following structure:

Actual \ Predicted	**Positive (Predicted 1)**	**Negative (Predicted 0)**
Positive (Actual 1)	True Positive (TP)	False Negative (FN)
Negative (Actual 0)	False Positive (FP)	True Negative (TN)

Table 1: Confusion Matrix for Binary Classification

In this table, for example, True Positives (TP) are cases where the model predicted "Positive" and the actual class was indeed "Positive". False Positives (FP) are cases where the model predicted "Positive" but the actual was "Negative" (also known as "Type I error"), and False Negatives (FN) are cases where the model predicted "Negative" but the actual was "Positive" ("Type II error"). True Negatives (TN) are correct predictions of the "Negative" class. The confusion matrix thus tells us where the model is making mistakes – is it confusing positives as negatives or vice versa – and provides the basis for many evaluation metrics. For multi-class classification, the confusion matrix extends to an N x N table (for N classes) showing counts for every actual-predicted class combination, which helps identify, say, which classes are most misclassified as others. By examining the confusion matrix, we gain insight beyond a single performance number – we can see if the model has a bias towards predicting a certain class or if some classes are systematically harder to predict than others. From the confusion matrix, we can define several evaluation metrics that summarize performance:

1. **Accuracy**
   Accuracy is the simplest metric, defined as the percentage of correctly predicted instances out of all instances. In terms of the confusion matrix, it is calculated as:

   $$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

   It represents the overall fraction of predictions the model got right. For example, if a model made 100 predictions and 90 were correct (either true positives or true negatives), the accuracy is 90%. High accuracy means the model is correct on most samples. Accuracy is a good metric when the class distribution in the dataset is roughly balanced (i.e., each class has about equal representation in the data). However, accuracy can be misleading for imbalanced datasets. Consider a dataset of 1000 samples where 950 are "Negative" and 50 are "Positive". A dummy classifier that always predicts "Negative" will be correct 950 times, yielding 95% accuracy – seemingly excellent, but it completely fails to catch any of the Positive cases. Thus, in scenarios like rare disease detection or fraud detection (where one class is rare), we look at other metrics that account for the performance on the minority class. In summary, use accuracy as a quick check, but complement it with more sensitive metrics when classes are imbalanced.

2. **Precision (Positive Predictive Value - PPV)**
   Precision measures how many of the instances that the model predicted as positive are actually positive. It is defined as:

   $$\text{Precision} = \frac{TP}{TP + FP}$$

   In other words, precision is the proportion of positive predictions that were correct. If a model labels 100 emails as "spam" (positive class) and 80 of those are truly spam while 20 are actually legitimate (false alarms), the precision is 80%. A high precision means that when the model says "Positive", it's usually correct – there are few false positives. Precision is particularly important in situations where false positives are costly. For example, in spam filtering, a false positive means

a legitimate email is incorrectly marked as spam (which could cause a user to miss important messages). In medical testing, a false positive might mean telling a healthy person they have a disease (leading to unnecessary stress and further tests). In such cases, we want a model with high precision to minimize false alarms. Precision alone, however, doesn't tell the whole story – it doesn't consider how many actual positives were missed (that's recall). A model could achieve very high precision by being extremely conservative in identifying positives (e.g., only flag the most obvious spam emails), but it might then miss a lot of spam. Thus, precision is often considered alongside recall.

3. **Recall (Sensitivity or True Positive Rate - TPR)**
   Recall measures how many of the actual positive cases the model captured by predicting positive. It is also known as sensitivity or TPR. It is defined as:

$$\text{Recall} = \frac{TP}{TP + FN}$$

   This is the fraction of positives that were successfully identified by the model. Using the email example, if there were actually 100 spam emails in the inbox and the filter caught 80 of them (true positives) but 20 spam emails slipped through as "not spam" (false negatives), the recall is 80%. High recall means the model is catching most of the positive instances (few false negatives). This is crucial when missing a positive is very costly. In medical diagnostics, recall is critical because a false negative means failing to detect a disease in a sick patient, which could be life-threatening. For a cancer screening test, you'd want as high recall as possible – better to follow up on some false positives (which precision would measure) than to miss a cancer case. However, a model that trivially labels everything as positive would have 100% recall (catching all true positives) but an awful precision (because it also flags every negative as positive). Therefore, there's often a trade-off: increasing recall may decrease precision and vice versa. Depending on the application, one might choose to prioritize recall over precision or balance them.

4. **F1-score**
   The F1-score is the harmonic mean of precision and recall, providing a single metric that balances both. It is defined as:

$$F1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

   The F1-score is high only when both precision and recall are high; if one is low, F1 will be low. It's a useful metric when we seek a balance between precision and recall and want a single number to summarize that trade-off. The harmonic mean is used (instead of a simple average) because it punishes extreme values: for instance, a precision of 1.0 and recall of 0 (or vice versa) yields an F1 of 0, indicating no overall success if one of the components is zero. When to use F1: It's especially helpful in imbalanced classification or when we care about how the classifier performs on the positive class and want to consider false positives and false negatives together. For example, in a search engine, precision is how relevant the top results are, and recall is whether it retrieved all relevant results; F1 gives a combined sense of the search quality. If we simply want an aggregate measure of a test's accuracy on the positive class (and negatives are not as important), F1 is a good choice. However, F1 can be less intuitive to interpret than precision or recall individually, so it's often reported alongside them.

5. **ROC-AUC (Receiver Operating Characteristic – Area Under Curve):**
   The ROC-AUC is a more holistic metric for binary classification performance, especially useful when we care about the model's performance across all classification thresholds. An ROC curve plots the True Positive Rate (Recall) against the False Positive Rate (FPR) at various threshold settings of the classifier. Instead of committing to a specific probability cutoff (like 0.5), we evaluate the model's ability to discriminate between classes at all thresholds. The Area Under the ROC Curve (AUC) is a single number summary of the ROC curve. It can be interpreted

as the probability that the classifier will rank a randomly chosen positive instance higher (i.e., more likely to be positive) than a randomly chosen negative instance. An AUC of 0.5 means the model has no discriminative power (equivalent to random guessing), while an AUC of 1.0 indicates perfect separation of classes. For example, if a spam classifier has an AUC of 0.95, it means that if you randomly pick one spam email and one legitimate email, 95% of the time the classifier will assign a higher spam-score to the spam email – a very good result. When to use ROC-AUC: It's particularly useful for imbalanced datasets and when the end application allows for selecting an optimal threshold. It evaluates the model's sensitivity vs. specificity trade-off without regard to a specific threshold or class distribution. In medical diagnostics or other decision-making scenarios, one might examine the ROC curve to choose a threshold that gives a desired balance (for instance, choosing a threshold that yields a certain minimum recall while examining how precision drops). A high AUC is generally desirable as it means the model is consistently ranking positives above negatives. However, one should be cautious: AUC is an aggregate measure and might not reflect performance at a particular operating point of interest. Also, for extremely skewed datasets, AUC can sometimes present an overly optimistic picture (since it still considers the full range of FPR down to very low values that might involve many false positives in absolute terms). Despite these nuances, ROC-AUC is widely used as a robust metric for comparing classifiers.

Choosing the Right Metric: The choice of evaluation metric should be aligned with the goals of the model and the context of the problem. For instance:

1. If you care about overall accuracy and classes are balanced, accuracy might suffice.
2. If false positives are more concerning (e.g., precision in a spam filter to not block real emails), focus on precision.
3. If false negatives are more critical (e.g., recall in an alarm system to catch all real intrusions), focus on recall.
4. The F1-score gives a balance of precision and recall, useful when you need a single measure for model selection under class imbalance.
5. ROC-AUC is great for comparing models in a threshold-agnostic way and is widely used in research and competitions to judge discriminative ability.

3 Introduction to Logistic Regression

Logistic regression is a supervised learning algorithm used for *classification* tasks, especially binary classification. It predicts the probability that an input belongs to a certain class (e.g., "yes" or "no") by fitting data to a logistic (sigmoid) function. Unlike linear regression, which outputs a continuous value, logistic regression outputs a probability between 0 and 1, which can be thresholded to predict discrete classes.

In other words, linear regression models a numeric outcome (which can produce values outside $[0, 1]$), whereas logistic regression is designed to classify by estimating the likelihood of the positive class. This makes logistic regression suitable for scenarios like predicting whether an email is spam or not, or whether a tumor is malignant or benign, where the outcome is categorical. Once the model provides a probability, we can map it to a class label (e.g., predict class 1 if probability > 0.5 and class 0 otherwise), thereby performing the classification.

3.1 Mathematical Formulation

Sigmoid (Logistic) Function

Logistic regression relies on the *sigmoid* function (also known as the logistic function) to convert a linear combination of features into a probability. The sigmoid function is defined as:

$$\sigma(t) \;=\; \frac{1}{1+e^{-t}}\,,$$

an S-shaped curve that maps any real-valued input t to the range $(0, 1)$. For instance, $\sigma(0) = 0.5$, $\sigma(t) \to 1$ as $t \to +\infty$, and $\sigma(t) \to 0$ as $t \to -\infty$. This curve ensures the model's output can be interpreted as a valid probability in $[0, 1]$.

Logistic Model Equation

In a binary logistic regression model, we compute a weighted sum of the input features (plus an intercept) and then apply the sigmoid function. For input features $X = (x_1, x_2, \ldots, x_n)$, we calculate the *linear predictor*:

$$z \;=\; \beta_0 + \beta_1\, x_1 + \beta_2\, x_2 + \cdots + \beta_n\, x_n\,.$$

Then, the *predicted probability* of the positive class (class labeled "1") is:

$$p \;=\; \Pr(Y = 1 \mid X) \;=\; \sigma(z) \;=\; \frac{1}{1+e^{-z}}\,.$$

This guarantees $0 < p < 1$ for any z. Equivalently, we have a linear relationship in the *log-odds* (or logit) scale:

$$\ln\!\Big(\tfrac{p}{1-p}\Big) \;=\; \beta_0 \;+\; \beta_1\, x_1 \;+\; \cdots \;+\; \beta_n\, x_n\,.$$

This means logistic regression models the *log-odds* of the outcome as a linear function of the input features.

Decision Boundary

Once the model computes $p = \sigma(z)$, we apply a *decision threshold* to predict the class. By default, a threshold of 0.5 is used in binary classification:

$$\text{Predict class 1 if } p \geq 0.5, \quad \text{otherwise predict class 0.}$$

Geometrically, the set of points where $p = 0.5$ (i.e. $z = 0$) defines the *decision boundary*, which is a *hyperplane* given by:

$$\beta_0 + \beta_1\, x_1 + \cdots + \beta_n\, x_n \;=\; 0\,.$$

All points on one side of this boundary are classified as class 1 ($p > 0.5$), and on the other side as class 0. Hence, logistic regression yields a **linear decision boundary** in the original feature space.

3.2 Assumptions of Logistic Regression

- **Binary outcome (for basic logistic regression).** The dependent variable should be dichotomous (0 or 1). (Extensions exist for multiple classes.)
- **Independence of observations.** Each sample is assumed independent of the others. If the data points are correlated (e.g. repeated measures on the same subjects), standard logistic regression can give biased estimates.
- **No (or little) multicollinearity among predictors.** Highly correlated features can cause unstable coefficient estimates and large standard errors.
- **Linearity of independent variables and log-odds.** Logistic regression assumes a linear relationship between predictors and the *log-odds* of the outcome. If non-linear effects exist, you may need polynomial terms or other transformations.
- **Sufficient sample size.** A rule of thumb is at least 10–15 observations per predictor, particularly in the smaller class, to avoid overfitting or quasi-complete separation.

3.3 Advantages

- **Interpretability.** Coefficients can be interpreted as how each feature affects the log-odds of the outcome, and thus the probability. You can derive odds ratios as well.
- **Probabilistic outputs.** Logistic regression provides a well-calibrated probability estimate, letting you adjust classification thresholds as needed.
- **Efficiency and simplicity.** Training is relatively fast, with a convex cost function that has a unique global optimum.
- **Less prone to overfitting (if the data is suitable).** Compared to high-variance models (e.g. deep trees), logistic regression can generalize well in linear-separable tasks. Regularization (e.g. L2) further helps reduce overfitting.
- **Widely supported.** Logistic regression is a well-understood algorithm with solid theoretical background and readily available implementations (e.g. in `scikit-learn`, R, SAS, SPSS).

3.4 Limitations

- **Linear decision boundary.** Logistic regression can only separate classes via a single linear (hyperplane) boundary. Complex or non-linear class boundaries require feature engineering or more flexible models (e.g. trees, kernel SVM).
- **Mis-specification risk.** Logistic regression strictly assumes a linear relationship in log-odds. If the true relationship is highly non-linear, it may underfit unless you add non-linear terms.
- **Sensitivity to outliers.** Extreme observations can influence the model parameters significantly or lead to convergence issues (complete separation).
- **Requires careful handling of highly correlated features.** Multicollinearity can inflate the variance of coefficients and make them unstable.
- **Binary by default.** Extending logistic regression to multi-class tasks (multinomial logistic) is straightforward, but still assumes linear logit boundaries and can become more complex for multi-label or structured outputs.

4 Introduction to Decision Trees

A **decision tree** is a supervised learning method commonly used for *classification* (and also regression) tasks. It builds a hierarchical tree-like model that progressively splits the feature space to separate classes. Each internal node in the tree corresponds to a *feature test*, and each branch to a possible outcome of that test; leaf nodes provide the final class label (or distribution). Decision trees are popular due to their *interpretability* (they resemble a flowchart of if-then rules), relatively *low computational cost* to make predictions, and their ability to handle both numerical and categorical data.

Unlike linear models (e.g. logistic regression) that produce a linear decision boundary, decision trees can *split* the feature space multiple times to capture complex, non-linear decision boundaries. However, they can also *overfit* if grown too deep. Various tree-building algorithms exist—such as *ID3*, *C4.5*, and *CART*—differing in the splitting criteria and handling of attributes, but the general principles remain the same.

4.1 Mathematical Formulation of Tree Induction

Recursive Partitioning

The general idea behind *tree induction* is **recursive partitioning** of the training set. Suppose we have a training set

$$\left\{ \left(\mathbf{x}^{(i)}, y^{(i)}\right) \right\}_{i=1}^{N} \quad \text{where} \quad y^{(i)} \in \{1, \ldots, K\} \quad (\text{for } K\text{-class classification}),$$

and $\mathbf{x}^{(i)}$ contains the features for instance i. Initially, the root node contains all N training examples. We then select a *feature test* that splits these examples into two or more child nodes to *increase purity* (i.e., to reduce the mixture of classes).

This process repeats recursively: in each child node, choose the best feature test and partition again. Eventually, the recursion stops, and we assign a *leaf node* with a predicted class (or a class distribution) to the partition.

Splitting Criteria

A central question is how to pick the "best" feature test at each node. Different algorithms use different measures of *impurity* or *information gain*. Two common metrics in classification are:

- **Gini Impurity (CART style):** For a node with class distribution $p_1, p_2, \ldots, p_K$ (where p_j is the fraction of samples belonging to class j), the *Gini impurity* is

  $$G = \sum_{j=1}^{K} p_j \left(1 - p_j\right) \;=\; 1 - \sum_{j=1}^{K} p_j^2.$$

  A node is pure (only one class) if $G = 0$. When splitting a node into subnodes, the algorithm computes the *weighted sum* of Gini impurities in each child, aiming to **minimize** it.

- **Entropy / Information Gain (ID3/C4.5 style):** For the same class distribution p_j, the *entropy* is

  $$H \;=\; -\sum_{j=1}^{K} p_j \, \log_2(p_j).$$

  A node is pure if $H = 0$. The algorithm looks for splits that yield the largest *information gain*, i.e. the greatest decrease in entropy from parent to children.

Binary vs. Multi-way Splits

Some algorithms split each node *binary* (e.g. $x < \theta$ vs. $x \geq \theta$), while others do multi-branch splits (one branch per category or per interval). The approach depends on the type of feature (numeric or categorical) and the particular tree induction algorithm.

4.2 Stopping Criteria and Pruning

A naive tree-building procedure might continue until each leaf is *completely pure* (all samples belong to the same class), but this frequently leads to **overfitting**. To mitigate this, one can:

- **Stop early**: enforce a maximum depth, a minimum number of samples per leaf, or stop if the node cannot be split to improve purity.
- **Prune after growing**: grow the tree fully, then prune back less-important branches by using a validation set or a complexity parameter (like cost-complexity pruning in CART).

The final result is a tree of manageable depth that balances *accuracy* and *generalization*.

4.3 Assumptions of Decision Trees

While flexible, decision trees have implicit assumptions:

- **Samples are independent.** Similar to other supervised models, each instance is assumed independent of others.
- **Features are relevant and well-expressed.** A tree only splits on the features provided. If crucial information is missing or not well-encoded in the features, the model may fail to separate classes.
- **No strong assumption about linearity.** Unlike logistic regression, a decision tree does *not* assume a linear boundary; it learns piecewise-constant regions. This can be an advantage for non-linear data, but also means the tree can become very complex.
- **Sufficient data in each node for reliable splits.** If data is very sparse or if many features are noisy, the tree can make splits that don't generalize. Regularization (max depth, pruning, etc.) is crucial to avoid overfitting.

4.4 Advantages

- **Interpretability.** A decision tree can be seen as a flowchart of if-else questions, making it more understandable to non-experts. Each path to a leaf represents a *logical rule* for classification.
- **Non-linear boundaries.** Splits in the feature space can produce complex decision boundaries, enabling the model to capture intricate patterns that linear methods might miss.
- **Handles mixed data types.** Decision trees naturally handle numerical and categorical features without extensive preprocessing. They can also handle missing values (depending on the implementation).
- **Invariance to feature scaling.** Unlike some models, a tree does not require feature scaling since it uses thresholds and comparisons rather than distances or dot products.
- **Fast inference.** Making a prediction requires traversing a single path down the tree from root to leaf, which is generally quite fast.

4.5 Limitations

- **Overfitting.** A fully grown tree can perfectly classify the training data but generalize poorly. Pruning or limiting depth is essential.
- **High variance.** Small changes (e.g. in the training set) can alter splits near the top of the tree, yielding very different subtrees. This can be mitigated by *ensemble methods* like Random Forests or Gradient Boosted Trees.

- **No global optimum guarantee.** Greedy splitting at each node can lead to suboptimal overall trees. ID3, C4.5, and CART do local optimization, not global.
- **Less expressive probability estimates.** Probabilities at a leaf are often the fraction of samples of each class in that leaf. These can be blocky and unstable, especially with small leaf sizes.
- **Can be outperformed by other methods.** Complex non-linear models (neural nets, ensembles) often achieve higher accuracy if the data is large and varied.

1. What is the primary objective of classification in machine learning?
   (a) To predict continuous numeric values
   (b) To cluster data into groups
   (c) To assign discrete labels to input data
   (d) To generate new features
2. Which of the following is an example of a classification problem?
   (a) Predicting the price of a house
   (b) Forecasting daily temperature
   (c) Determining if an email is spam or not
   (d) Estimating the amount of rainfall in a region
3. What is the key difference between classification and regression?
   (a) Classification predicts a continuous value, while regression predicts a categorical outcome
   (b) Regression predicts a continuous value, while classification predicts a categorical outcome
   (c) Classification and regression are the same
   (d) Regression always provides better accuracy
4. What preprocessing step is often applied to numerical features in classification?
   (a) Removing all numerical values
   (b) Feature scaling (normalization or standardization)
   (c) Converting numbers into text
   (d) Ignoring missing values
5. Which of the following is a binary classification problem?
   (a) Predicting whether a tumor is malignant or benign
   (b) Identifying the breed of a dog from an image
   (c) Classifying emails into Work, Personal, and Promotions categories
   (d) Predicting the price of a car based on its mileage
6. What is a key challenge of binary classification?
   (a) Handling missing values
   (b) Handling highly imbalanced datasets
   (c) Feature extraction
   (d) Data visualization
7. Which algorithm is commonly used for binary classification?
   (a) Linear Regression
   (b) Logistic Regression
   (c) k-Means Clustering
   (d) Principal Component Analysis

8. What is multi-class classification?

   (a) A classification problem where there are only two possible output labels
   (b) A problem where a data point belongs to multiple labels at once
   (c) A classification problem with more than two possible output classes
   (d) A method used for regression

9. In a classification task, what is the purpose of feature engineering?

   (a) To make the dataset larger
   (b) To improve model performance by selecting or transforming relevant features
   (c) To remove irrelevant instances from the dataset
   (d) To increase the computational complexity

10. What is the purpose of splitting data into training and test sets?

    (a) To reduce the dataset size
    (b) To evaluate model performance on unseen data
    (c) To increase training accuracy
    (d) To reduce computation time

11. Which metric is used to evaluate classification models?

    (a) Mean Squared Error (MSE)
    (b) Accuracy, Precision, Recall, F1-score
    (c) R-Squared (R^2)
    (d) Root Mean Squared Error (RMSE)

12. What does a confusion matrix represent?

    (a) A table summarizing correct and incorrect predictions
    (b) A method for normalizing data
    (c) A feature extraction technique
    (d) A probability distribution

13. In a confusion matrix, what does True Positive (TP) mean?

    (a) A correct prediction of the negative class
    (b) A misclassification of a positive class
    (c) A correct prediction of the positive class
    (d) A misclassification of the negative class

14. Which metric measures the proportion of actual positives correctly identified?

    (a) Precision
    (b) Recall (Sensitivity)
    (c) Accuracy
    (d) ROC-AUC

15. What is the formula for precision?

    (a) TP / (TP + FN)
    (b) TP / (TP + FP)
    (c) (TP + TN) / (TP + TN + FP + FN)
    (d) TP / (TP + TN)

16. What does the F1-score measure?

    (a) The geometric mean of precision and recall
    (b) The balance between precision and recall
    (c) The percentage of correctly classified instances
    (d) The number of features used in a model

17. When should you use precision instead of recall?

    (a) When false positives are more costly than false negatives
    (b) When false negatives are more costly than false positives
    (c) When the dataset is perfectly balanced
    (d) When accuracy is already high

18. Which metric is used to evaluate classification models across different thresholds?

    (a) Confusion Matrix
    (b) Precision
    (c) ROC-AUC
    (d) Mean Squared Error

19. What is the purpose of the logistic function in logistic regression?

    (a) To normalize feature values
    (b) To convert any real-valued number into a probability between 0 and 1
    (c) To maximize model complexity
    (d) To cluster data points

20. What is the mathematical formula for the logistic function?

    (a) $\sigma(t) = \frac{1}{1+e^{-t}}$
    (b) $\sigma(t) = e^t$
    (c) $\sigma(t) = t^2$
    (d) $\sigma(t) = t + 1$

21. What is the decision boundary in logistic regression?

    (a) A threshold where the predicted probability is exactly 0.5
    (b) The maximum number of features used in the model
    (c) The point where classification fails
    (d) A method to handle missing values

22. What is an assumption of logistic regression?

    (a) The dataset must contain only categorical features
    (b) The dependent variable is binary
    (c) The model requires deep learning techniques
    (d) There are no independent variables

23. What does the Gini Impurity measure in a decision tree?

    (a) The probability of an incorrect classification at a node
    (b) The variance in numerical features
    (c) The correlation between features
    (d) The entropy of a dataset

24. What is entropy in decision trees?

    (a) A measure of randomness or impurity in a dataset
    (b) A type of loss function
    (c) A data normalization technique
    (d) A metric for regression models

25. What is the stopping criterion for growing a decision tree?

    (a) When all nodes have equal numbers of samples
    (b) When further splitting does not improve classification
    (c) When the dataset contains more than 100 features
    (d) When there are no categorical variables

26. How does pruning help decision trees?

    (a) It removes unnecessary branches to reduce overfitting
    (b) It increases tree depth for better classification
    (c) It replaces missing values in the dataset
    (d) It improves feature scaling

27. Which of the following is NOT an advantage of decision trees?

    (a) Interpretability
    (b) Handling both categorical and numerical data
    (c) Sensitivity to noisy data
    (d) Fast inference

28. Which of the following methods helps reduce overfitting in decision trees?

    (a) Increasing the depth of the tree
    (b) Using ensemble methods like Random Forest
    (c) Removing the training set
    (d) Using logistic regression

29. How do decision trees handle categorical data?

    (a) By assigning numerical values to categories
    (b) By creating branches based on categorical feature values
    (c) By converting categorical data into text format
    (d) By removing categorical features from the dataset

30. What is the main limitation of decision trees?

    (a) They are always less accurate than neural networks
    (b) They require feature scaling
    (c) They tend to overfit without pruning
    (d) They cannot handle categorical features