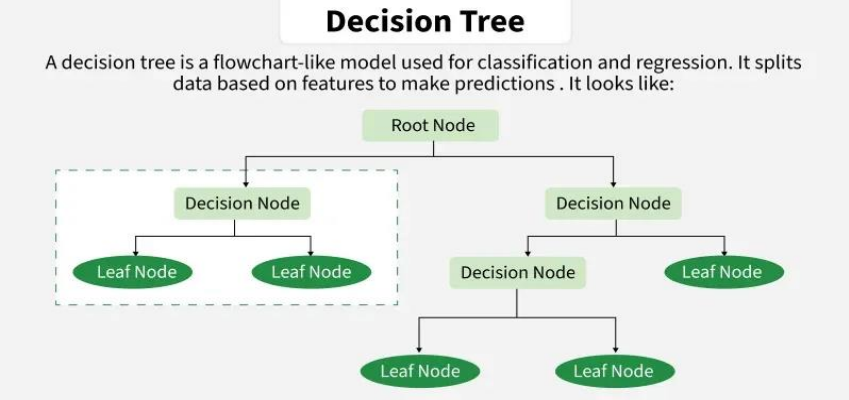
UNIT-III

Decision Tree in Machine Learning:

A decision tree is a supervised learning algorithm used for both classification and regression tasks. It has a hierarchical tree structure which consists of a root node, branches, internal nodes and leaf nodes. It works like a flowchart help to make decisions step by step where:

- Internal nodes represent attribute tests
- Branches represent attribute values
- Leaf nodes represent final decisions or predictions.

Decision trees are widely used due to their interpretability, flexibility and low preprocessing needs.



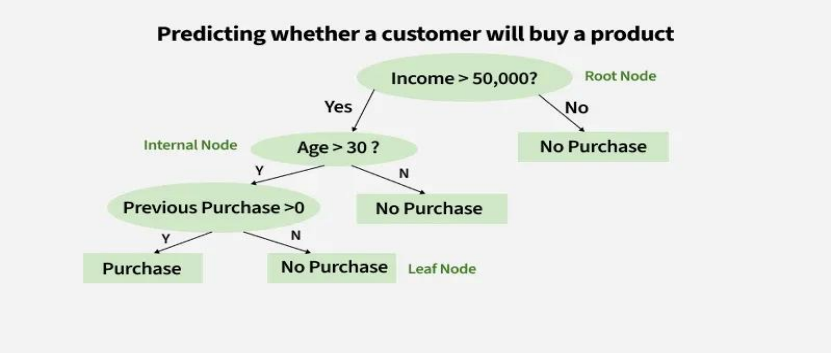
Decision tree terminology

These terms come up frequently in machine learning and are helpful to know as you embark on your machine-learning journey:

- Root node: The topmost node of a decision tree that represents the entire message or decision
- Decision (or internal) node: A node within a decision tree where the prior node branches into two or more variables
- Leaf (or terminal) node: The leaf node is also called the external node or terminal node, which means it has no child—it's the last node in the decision tree and furthest from the root node
- Splitting: The process of dividing a node into two or more nodes; the part at which the decision branches off into variables
- Pruning: The opposite of splitting, the process of going through and reducing the tree to only the most important nodes or outcomes

How Does a Decision Tree Work

A decision tree splits the dataset based on feature values to create pure subsets ideally all items in a group belong to the same class. Each leaf node of the tree corresponds to a class label and the internal nodes are feature-based decision points.



1. Root Node (Income)

First Question: "Is the person's income greater than $50,000?"

- If Yes, proceed to the next question.
- If No, predict "No Purchase" (leaf node).

2. Internal Node (Age):

If the person's income is greater than $50,000, ask: "Is the person's age above 30?"

- If Yes, proceed to the next question.
- If No, predict "No Purchase" (leaf node).

3. Internal Node (Previous Purchases):

If the person is above 30 and has made previous purchases, predict "Purchase" (leaf node).

If the person is above 30 and has not made previous purchases, predict "No Purchase" (leaf node).

Advantages of Decision Trees

Easy to Understand: Decision Trees are visual which makes it easy to follow the decision-making process.

Versatility: Can be used for both classification and regression problems.

No Need for Feature Scaling: Unlike many machine learning models, it don't require us to scale or normalize our data.

Handles Non-linear Relationships: It capture complex, non-linear relationships between features and outcomes effectively.

Interpretability: The tree structure is easy to interpret helps in allowing users to understand the reasoning behind each decision.

Handles Missing Data: It can handle missing values by using strategies like assigning the most common value or ignoring missing data during splits.

Disadvantages of Decision Trees

- **Overfitting:** They can overfit the training data if they are too deep which means they memorize the data instead of learning general patterns. This leads to poor performance on unseen data.
- **Instability:** It can be unstable which means that small changes in the data may lead to significant differences in the tree structure and predictions.
- **Bias towards Features with Many Categories:** It can become biased toward features with many distinct values which focuses too much on them and potentially missing other important features which can reduce prediction accuracy.
- **Difficulty in Capturing Complex Interactions:** Decision Trees may struggle to capture complex interactions between features which helps in making them less effective for certain types of data.
- **Computationally Expensive for Large Datasets:** For large datasets, building and pruning a Decision Tree can be computationally intensive, especially as the tree depth increases.

Applications of Decision Trees

1. **Loan Approval in Banking:** Banks use Decision Trees to assess whether a loan application should be approved.
3. **Medical Diagnosis:** In healthcare they assist in diagnosing diseases. For example, they can predict whether a patient has diabetes based on clinical data like glucose levels, BMI and blood pressure.
4. **Predicting Exam Results in Education:** Educational institutions use to predict whether a student will pass or fail based on factors like attendance, study time and past grades.
5. **Customer Churn Prediction:** Companies use Decision Trees to predict whether a customer will leave or stay based on behavior patterns, purchase history, and interactions.
5. **Fraud Detection:** In finance, Decision Trees are used to detect fraudulent activities, such as credit card fraud. By analyzing past transaction data and patterns, Decision Trees can identify suspicious activities and flag them for further investigation.

- **Information Gain and Gini Index in Decision Tree**:
-

Till now we have discovered the basic intuition and approach of how decision tree works,

so lets just move to the attribute selection measure of decision tree. We have two popular attribute selection measures used:

- **1. Information Gain**
- Information Gain tells us how useful a question (or feature) is for splitting data into groups. It measures how much the uncertainty decreases after the split. A good question will create clearer groups and the feature with the highest Information Gain is chosen to make the decision.

- Suppose *S* is a set of instances *A* is an attribute, *Sv* is the subset of *S*, *v* represents an individual value that the attribute *A* can take and Values (*A*) is the set of all possible values of *A* then

$$\text{Information Gain}(S,a) = \text{Entropy}(S) - \sum_{v \in \text{values}(a)} \frac{|S_v|}{|S|} \text{Entropy}(S_v)$$

- Entropy: is the measure of uncertainty of a random variable it characterizes the impurity of an arbitrary collection of examples. The higher the entropy more the information content.

For example if a dataset has an equal number of "Yes" and "No" outcomes (like 3 people who bought a product and 3 who didn't), the entropy is high because it's uncertain which outcome to predict. But if all the outcomes are the same (all "Yes" or all "No") the entropy is 0 meaning there is no uncertainty left in predicting the outcome

Suppose S*S* is a set of instances, A*A* is an attribute, Sv*Sv* is the subset of S*S* with A*A*= v*v* and Values (A*A*) is the set of all possible values of A*A*, then

$$H(P) = -\sum_{x \in C} P(x) \log P(x)$$

Gini Index

Gini Index is a metric to measure how often a randomly chosen element would be incorrectly identified. It means an attribute with a lower Gini index should be preferred. Sklearn supports "Gini" criteria for Gini Index and by default it takes "gini" value.

For example if we have a group of people where all bought the product (100% "Yes") the Gini Index is 0 indicate perfect purity. But if the group has an equal mix of "Yes" and "No" the Gini Index would be 0.5 show high impurity or uncertainty. Formula for Gini Index is given by :

Impurity measures:

Decision Trees are classification models that split data into nodes based on feature values. To determine the best split, they rely on impurity metrics that evaluate how mixed a node's class distribution is. Gini Impurity and Entropy are two measures used in decision trees to decide how

to split data into branches. Both help determine how mixed or pure a dataset is, guiding the model toward splits that create cleaner groups.

Need for Impurity Measures

Some common reasons why impurity criteria are essential in decision tree learning are:

- Prevents random or uninformative splits that reduce predictive strength.
- Helps isolate class boundaries for better interpretability.
- Controls node quality and prevents over-growth of branches.
- Reduces classification ambiguity by separating noisy samples.
- Maintains model consistency across varied datasets.

1. Gini Impurity

Gini Impurity checks how often a randomly selected sample would be mislabeled if assigned by class probability. It is computationally simple and used in tree-based classifiers.

Formula:

$$Gini = 1 - \sum pi2$$

Where pi is the probability of class i*i*.

Properties:

- Lower values indicate cleaner and more homogeneous nodes.
- Nodes become pure when all samples belong to one class.
- Slightly biased toward dominant classes during split selection.

2. Entropy

Entropy measures uncertainty in a node's class distribution and originates from information theory. Higher entropy indicates greater disorder among class labels.

Formula:

$$Entropy = -\sum \log_2(pi)$$

Where pi represents the proportion of class i*i* in the node.

Properties:

- Zero entropy corresponds to perfectly pure splits.
- Sensitive to small fluctuations across class ratios.
- Often yields balanced splits with meaningful boundaries.

Applications

Some of the use-cases of impurity metrics are:

- **Fraud Detection:** Helps differentiate legitimate behaviors from suspicious anomalies, supporting real-time financial monitoring.
- **Customer Behavior Classification:** Enables marketing segmentation by grouping users with similar interaction patterns, improving personalized outreach.
- **Medical Predictions:** Assists in screening symptoms or diagnostic attributes, allowing healthcare systems to classify risk more accurately.
- **Quality Inspection:** Detects faulty products by separating normal sensor readings from defective patterns within manufacturing pipelines.
- **Document Categorization:** Organizes incoming text data into relevant topics, improving search and retrieval efficiency in large content systems.

Advantages

Some benefits of impurity based splitting include:

- **Clear Decision Boundaries:** Each split expresses a simple logical condition, making the model inherently explainable and easy to visualize.
- **Reduced Ambiguity:** Sibling branches become more homogeneous, improving consistency across the tree.
- **Improved Intermediate Node Quality:** Refined node splits reduce the complexity needed in later branches, enhancing learning efficiency.
- **Strong Multi-Class Handling:** Effectively separates overlapping classes in datasets with multiple labels.
- **Flexible Metric Choice:** Developers can switch between metrics depending on dataset behavior and performance observations.

Disadvantages

Some disadvantages of impurity metrics are:

- **Bias Toward Many Unique Values:** Features with many categories may appear informative artificially, leading to misleading splits.
- **Sensitivity to Noise and Imbalance:** Noisy datasets or skewed class distributions can distort impurity measurements and reduce accuracy.
- **Potential for Deep Trees:** Without constraints, impurity-driven expansion can increase depth and complexity unnecessarily.
- **Need for Pruning:** Additional regularization techniques such as pruning or depth limits are required to control overfitting and maintain generalization.
- **Types of decision trees in machine learning**

Decision trees in machine learning can be either classification trees or regression trees. Together, both algorithms fall into a category of “classification and regression trees” and are sometimes called CART. Their respective roles are to “classify” and to “predict.”

1. Classification trees

Classification trees determine whether an event happened or didn’t happen. Usually, this involves a 'yes' or 'no' outcome.

We often use this type of decision-making in the real world. Here are a few examples to help contextualise how decision trees work for classification:

- Example 1: How to spend your free time after work

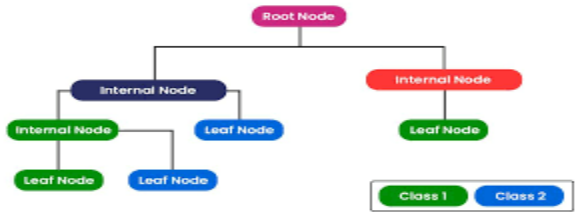
What you do after work in your free time can depend on the weather. If sunny, you can picnic with a friend, grab a drink with a colleague, or run errands. If it is raining, stay home and watch a movie instead. There is a clear outcome. That is classified as whether to “go out” or “stay in.”

- Example 2: Homeownership based on age and income

In a classification tree, the data set splits according to its variables. Two variables, age, and income, determine whether or not someone buys a house. If training data tells us that 70 percent of people over age 30 bought a house, then the data gets split there, with age becoming the first node in the tree. This split makes the data 80 percent 'pure.' The second node then addresses income.

To understand how decision trees work in machine learning, consider registering for these Guided Projects to apply your skills to real-world projects. You can complete them in two hours or less:

- Decision Tree and Random Forest Classification using Julia
- Predicting Salaries with Decision Trees



2. Regression trees

Regression trees, on the other hand, predict continuous values based on previous data or information sources. For example, they can predict the price of gasoline or whether a customer will purchase eggs (including which type of eggs and at which store).

This type of decision-making is more about programming algorithms to predict what is likely to happen, given previous behaviour or trends.

- Example 1: Housing prices in West Bengal

Regression analysis could be used to predict the price of a house in West Bengal, plotted on a graph. The regression model can predict housing prices in the coming years using data points from previous years' prices. This relationship is a linear regression since housing prices will continue to rise. Machine learning helps us predict specific prices based on a series of variables that have been true in the past.

- Example 2: Bachelor's degree graduates in 2025

A regression tree can help a university predict how many bachelor's degree students will be in 2025. On a graph, one can plot the number of degree-holding students between 2010 and 2022. If the number of university graduates increases linearly each year, then regression analysis can be used to build an algorithm that predicts the number of students in 2025.

To get started on how decision tree algorithms work in predictive machine learning models, look at these Guided Projects. Each project takes less than two hours, and they are based on real-world examples so you can elevate your skills:

- XG-Boost 101: Used Cars Price Prediction
- Decision Tree Classifier for Beginners in R

Classification and Regression Tree (CART) is a predictive algorithm used in machine learning that generates future predictions based on previous values. Decision trees are at the core of machine learning and serve as a basis for other machine learning algorithms, such as random forest, bagged decision trees, and boosted decision trees.

Classification vs Regression in Machine Learning

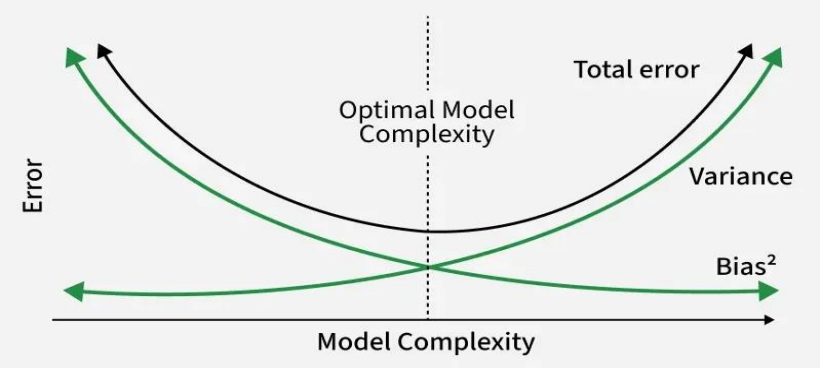
Difference Between Regression and Classification

Here we compare Regression and Classification in machine learning

Features	Regression	Classification
Output Type	Predicts a continuous numeric value	Predicts a categorical label
Goal	Fitting the best-fit line or curve	Drawing a decision boundary between classes
Error Measure	Uses metrics like MSE, MAE, RMSE	Uses metrics like Accuracy, Precision, Recall, F1 Score
Use Cases	Forecasting, price prediction, risk estimation	Image recognition, sentiment analysis, fraud detection
Handling Outliers	Highly sensitive	Less sensitive

Bias and Variance in Machine Learning

Bias and Variance are two fundamental concepts that help explain a model's prediction errors in machine learning. Bias refers to the error caused by oversimplifying a model while variance refers to the error from making the model too sensitive to training data.



Bias

Bias is the error that occurs when a model is too simple to capture the true patterns in the data.

1. **High bias:** The model oversimplifies, misses patterns and underfits the data.
2. **Low bias:** The model captures patterns well and is closer to the true values.

$$\text{Bias} = E[\hat{\theta}] - \theta.$$

How to Reduce Bias?

Some methods to lower bias in models are:

1. **Use More Complex Models:** Use models capable of capturing non-linear relationships such as neural networks or ensemble methods.
2. **Add Relevant Features:** Include additional informative features in the training data to give the model for capturing underlying patterns.
3. **Adjust Regularization Strength:** Reduce regularization to allow the model more flexibility in fitting the data.

Variance

Variance arises when a model becomes too sensitive to training data and it captures noises in data too. It fails to give prediction on unseen new data.

1. **High variance:** The model is too sensitive to small changes and may overfit.
2. **Low variance:** The model is more stable but might miss some patterns.

$$\sigma^2 = \frac{\sum(xi - \bar{x})^2}{N}$$

Bias Variance Tradeoff

The total prediction error depends on the tradeoff between bias and variance:

Model Type	Bias	Variance	Result
Underfitting	High	Low	Poor training and test performance
Optimal	Moderate	Moderate	Best generalization
Overfitting	Low	High	Poor test performance

Applications

Some of the applications of bias and variance analysis are:

1. **Model Selection:** Helps determine whether a simple or complex model is best suited for the task ensuring good generalization.
2. **Hyperparameter Tuning:** Guides fine tuning parameters such as learning rate, regularization strength or tree depth to reduce errors.
3. **Model Evaluation:** Assists in identifying underfitting or overfitting by comparing training and test performance.
4. **Error Analysis:** Helps pinpoint the main causes of prediction errors and refine model strategies accordingly.
5. **Ensemble Learning:** Balances bias and variance effectively by combining multiple models to enhance stability and accuracy.

Advantages

Some of the advantages of understanding bias and variance are:

1. **Improves Model Accuracy:** Enables building models that perform consistently well on unseen data.
2. **Supports Efficient Training:** Saves computational resources by avoiding unnecessarily complex or overfitted models.

3. **Enhances Interpretability:** Makes it easier to understand and explain the reasons behind model errors.
4. **Guides Model Complexity:** Helps find the optimal level of model complexity for different data sizes and problems.

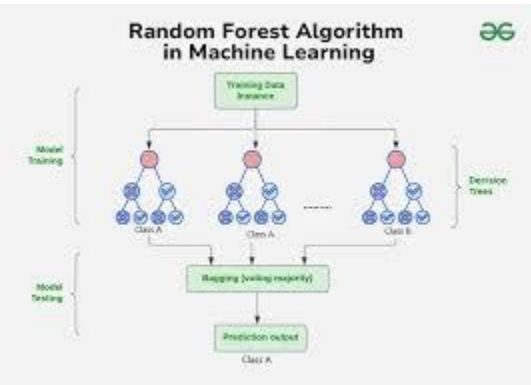
Limitations

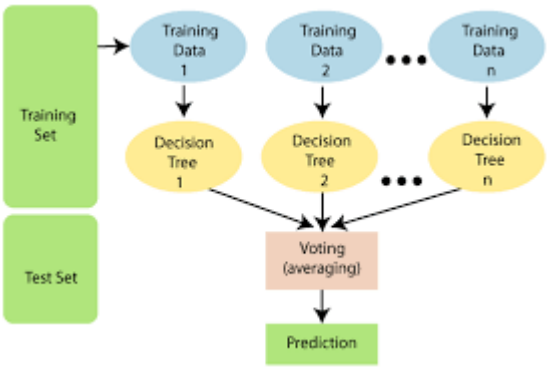
Some of the limitations of bias and variance concepts are:

1. **Difficult to Quantify Precisely:** Measuring exact bias and variance in modern complex models can be challenging.
2. **Highly Data Dependent:** Model behavior may vary significantly across datasets with different characteristics.
3. **Unpredictable in Deep Learning:** Deep neural networks can display unexpected bias-variance dynamics due to non-convex optimization.
4. **Tradeoff Challenge:** Minimizing one often increases the other requiring careful experimentation and balance.

Random forest for classification and regression in machine learning

Random Forest is a machine learning algorithm that uses many decision trees to make better predictions. Each tree looks at different random parts of the data and their results are combined by voting for classification or averaging for regression which makes it as ensemble learning technique. This helps in improving accuracy and reducing errors.





Working of Random Forest Algorithm

- **Create Many Decision Trees:** The algorithm makes many decision trees each using a random part of the data. So every tree is a bit different.
- **Pick Random Features:** When building each tree it doesn't look at all the features (columns) at once. It picks a few at random to decide how to split the data. This helps the trees stay different from each other.
- **Each Tree Makes a Prediction:** Every tree gives its own answer or prediction based on what it learned from its part of the data.

- **Combine the Predictions:** For **classification** we choose a category as the final answer is the one that most trees agree on i.e majority voting and for **regression** we predict a number as the final answer is the average of all the trees predictions.
- **Why It Works Well:** Using random data and features for each tree helps avoid overfitting and makes the overall prediction more accurate and trustworthy.

Key Features of Random Forest

- **Handles Missing Data:** It can work even if some data is missing so you don't always need to fill in the gaps yourself.
- **Shows Feature Importance:** It tells you which features (columns) are most useful for making predictions which helps you understand your data better.
- **Works Well with Big and Complex Data:** It can handle large datasets with many features without slowing down or losing accuracy.
- **Used for Different Tasks:** You can use it for both **classification** like predicting types or labels and **regression** like predicting numbers or amounts.

Assumptions of Random Forest

- **Each tree makes its own decisions**: Every tree in the forest makes its own predictions without relying on others.
- **Random parts of the data are used**: Each tree is built using random samples and features to reduce mistakes.
- **Enough data is needed**: Sufficient data ensures the trees are different and learn unique patterns and variety.
- **Different predictions improve accuracy**: Combining the predictions from different trees leads to a more accurate final result.

Advantages of Random Forest

- Random Forest provides very accurate predictions even with large datasets.
- Random Forest can handle missing data well without compromising with accuracy.
- It doesn't require normalization or standardization on dataset.
- When we combine multiple decision trees it reduces the risk of overfitting of the model.

Limitations of Random Forest

- It can be computationally expensive especially with a large number of trees.
- It's harder to interpret the model compared to simpler models like decision trees.

Random Forest is a versatile and widely used **ensemble learning** algorithm capable of performing both **classification** and **regression** tasks. It operates by constructing a multitude of

decision trees during training and merging their outputs to produce a more accurate and stable final prediction than any single tree could.

Classification vs. Regression

The primary difference between using a random forest for classification versus regression lies in how the predictions from individual trees are aggregated.

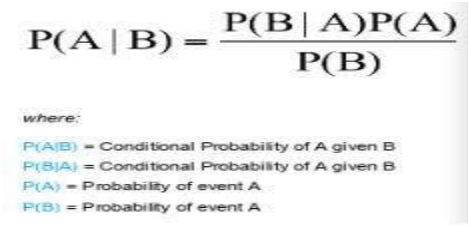
Feature	Classification	Regression
Target Variable	Categorical/Discrete labels (e.g., "spam" or "not spam", "disease" or "no disease")	Continuous numerical values (e.g., house prices, stock prices, temperature)
Aggregation	Uses **majority voting**, where the final output is the class predicted most frequently by the individual trees.	Uses the **mean (average)** of the predictions made by all individual trees.
Splitting Criterion	Typically uses metrics like Gini impurity or information gain to evaluate splits.	Typically uses metrics based on variance, such as Mean Squared Error (MSE).

Introduction to bayes classifier in machine learning:

A Bayes classifier is a supervised machine learning algorithm based on Bayes' Theorem, used for classification tasks by calculating the probability of a data point belonging to a specific class. It selects the label with the highest Maximum a Posteriori (MAP) probability, often assuming conditional independence between features (the "Naive" assumption) to simplify computation.

Key Concepts and Components

- Bayes' Theorem: The foundation, calculated as:



- Naive Assumption: The "Naive" Bayes classifier assumes all features are independent of each other, which simplifies calculations significantly even if this assumption is violated in real-world data.
- MAP (Maximum a Posteriori): The classifier assigns the class y that maximizes the probability

Types of Naive Bayes Classifiers

- Gaussian Naive Bayes: Assumes continuous features follow a normal (Gaussian) distribution.
- Multinomial Naive Bayes: Used for discrete counts, such as text classification (word frequencies).
- Bernoulli Naive Bayes: Suitable for binary/boolean features.

Advantages and Applications

- Speed: Very fast to train and predict because it only requires calculating probabilities.
- Efficiency: Handles high-dimensional data well, making it ideal for text classification, spam filtering, and sentiment analysis.
- Minimal Data: Requires less training data compared to complex algorithms.

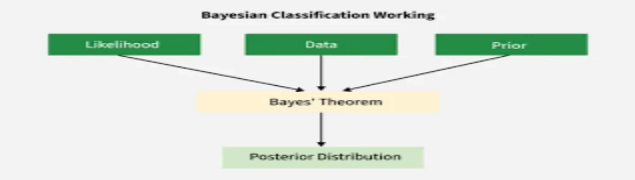
Limitations

- Independence Assumption
- Zero Frequency

Applications

- Spam Filtering: Identifying unwanted emails based on keywords.
- Sentiment Analysis: Categorizing text (like reviews) as positive, negative, or neutral.
- Medical Diagnosis: Predicting the likelihood of a disease based on symptoms.
- Recommendation Systems: Suggesting content or products based on previous user behavior.

Bayes classifier rule and inference:



Bayes Classifier Rule (Formula)

The core rule is to choose the class C (out of all possible classes) that maximizes the posterior probability given input features x:

$$\text{Class} = \arg\max_C P(C|x) = \arg\max_C \frac{P(x|C)P(C)}{P(x)}$$

Since the denominator $P(x)$ (evidence) is constant for all classes, the rule is typically applied as:

$$\text{Class} = \arg\max_C P(x|C)P(C)$$

Components:

- **Posterior ($P(C|x)$):** The probability of class C given features x.
- **Likelihood ($P(x|C)$):** The probability of features x given class C.
- **Prior ($P(C)$):** The initial probability of class C (before seeing data).
- **Evidence ($P(x)$):** The overall probability of features x.

Bayesian Inference and Application

Bayesian inference is the process of updating the probability of a hypothesis (C) as more evidence (x) becomes available. It is a learning procedure that combines prior beliefs with new data.

1. **Prior Belief ($P(C)$):** Establish initial knowledge about the class distribution.
2. **Likelihood ($P(x|C)$):** Determine the probability of observing the data features x under each possible class.
3. **Posterior Update ($P(C|x)$):** Multiply likelihood by prior to find the updated, posterior probability of each class.
4. **Decision:** Select the class with the maximum posterior probability.

Multiclass classification in machine learning

Multiclass Classification and Multi-Label Classification are two important approaches used to categorize data but they differ in how many classes an instance can belong to. In multiclass classification, each input is assigned to only one class, while in multi-label classification, an input can be associated with multiple classes at the same time. Understanding this distinction is essential for choosing the right model for real-world tasks.

Multiclass Classification

Multiclass classification is a type of supervised learning problem where each data instance is assigned to exactly one class out of three or more mutually exclusive classes. In this setting, an instance cannot belong to more than one class at the same time. The model learns to discriminate among all possible classes and predicts the single most probable class for each input.

For example, in a handwritten digit recognition task, an image can represent only one digit (0–9) at a time. Even though multiple classes exist, the prediction is restricted to exactly one label.

- Each instance has one and only one correct label.
- Classes are mutually exclusive.
- Prediction output is typically a single class index or label.
- Often implemented using softmax activation for probabilistic outputs.

Use Cases

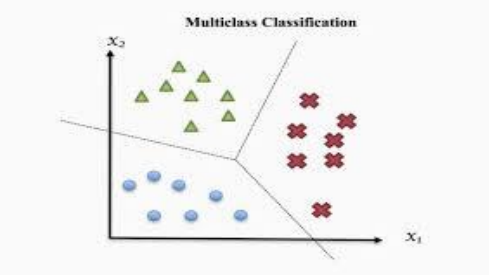
- **Digit recognition**: Identifying handwritten digits (0–9).
- **Document categorization**: Assigning a news article to one category such as politics, sports or technology.
- **Image classification**: Classifying an image as cat, dog or bird.
- **Medical diagnosis (single condition)**: Diagnosing a patient with one disease among multiple possible diseases.

Advantages

- Conceptually simple and easy to interpret.
- Wide algorithm support, including Logistic Regression, Decision Trees, SVMs and Neural Networks.
- Efficient training and inference compared to multi-label setups.
- Evaluation metrics like accuracy are straightforward and intuitive.

Limitations

- Cannot model overlapping categories, which limits applicability in complex domains.
- Assumes strict class boundaries, which may not reflect real-world ambiguity.
- Poor fit for problems where instances naturally belong to multiple categories.

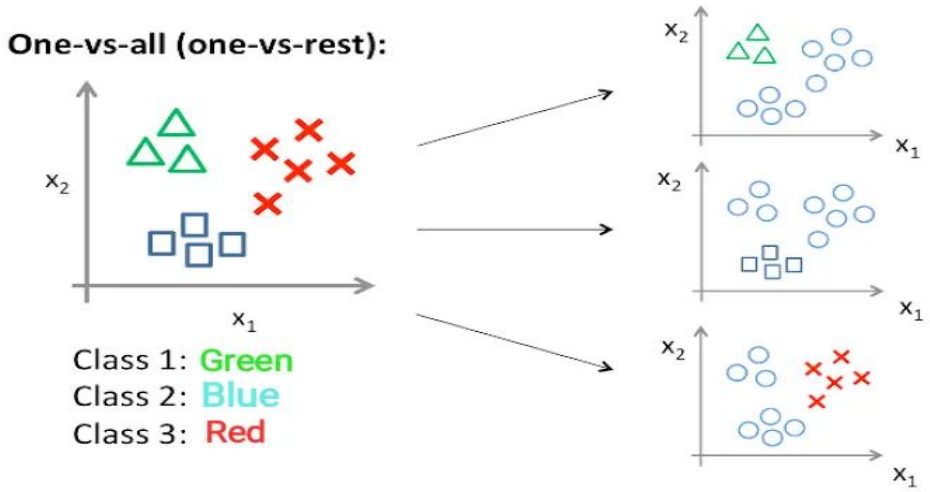


There are mainly two types of multi-class classification techniques:-

- **One vs. All (one-vs-rest)[N-class instances then N binary classifier models]**
- **One vs. One[N-class instances then N* (N-1)/2 binary classifier models]**

One vs. All (One-vs-Rest)

In one-vs-All classification, for the N-class instances dataset, we have to generate the N-binary classifier models. The number of class labels present in the dataset and the number of generated binary classifiers must be the same.



we have to generate the same number of classifiers as the class labels are present in the dataset, So we have to create three classifiers here for three respective classes.

Classifier 1:- [Green] vs [Red, Blue]

Classifier 2:- [Blue] vs [Green, Red]

Classifier 3:- [Red] vs [Blue, Green]

Now to train these three classifiers, we need to create three training datasets. So let's consider our primary dataset is as follows,

Features			Classes
x1	x2	x3	G
x4	x5	x6	B
x7	x8	x9	R
x10	x11	x12	G
x13	x14	x15	B
x16	x17	x18	R

Class 1 :- Green
Class 2 :- Blue
Class 3 :- Red

Main Dataset

Features			Classes
x1	x2	x3	G
x4	x5	x6	B
x7	x8	x9	R
x10	x11	x12	G
x13	x14	x15	B
x16	x17	x18	R

Class 1 :- Green Class 2 :- Blue Class 3 :- Red

Training Dataset 1
Class :- Green

Features			Green
x1	x2	x3	**+1**
x4	x5	x6	-1
x7	x8	x9	-1
x10	x11	x12	**+1**
x13	x14	x15	-1
x16	x17	x18	-1

Figure 6: Training dataset for **Green** class

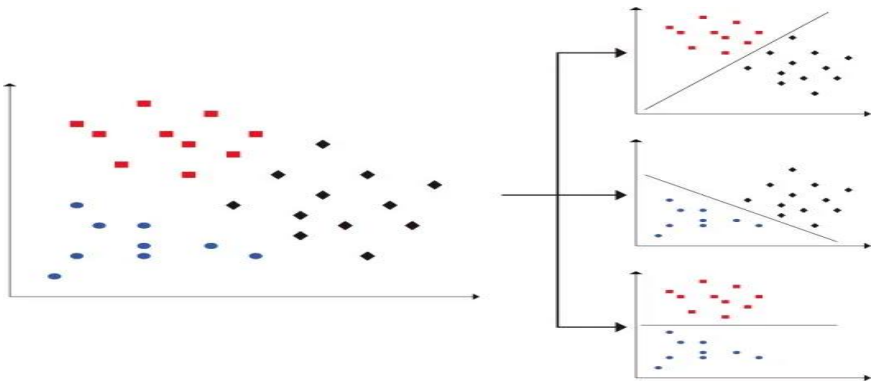
Training Dataset 2
Class :- Blue

Features			Blue
x1	x2	x3	-1
x4	x5	x6	**+1**
x7	x8	x9	-1
x10	x11	x12	-1
x13	x14	x15	**+1**
x16	x17	x18	-1

Training Dataset 3
Class :- Red

Features			Red
x1	x2	x3	-1
x4	x5	x6	-1
x7	x8	x9	**+1**
x10	x11	x12	-1
x13	x14	x15	-1
x16	x17	x18	**+1**

One vs. One (OvO)



In One-vs-One classification, for the **N-class** instances dataset, we have to generate the **N* (N-1)/2** binary classifier models. Using this classification approach, we split the primary dataset into one dataset for each class opposite to every other class.

Taking the above example, we have a classification problem having three types: **Green**, **Blue**, and **Red (N=3).**

We divide this problem into **N* (N-1)/2 = 3** binary classifier problems:

- Classifier 1: Green vs. Blue
- Classifier 2: Green vs. Red
- Classifier 3: Blue vs. Red

- **Binary vs Multiclass Classification**

Parameters	Binary classification	Multi-class classification
No. of classes	It is a classification of two groups, i.e. classifies objects in at most two classes.	There can be any number of classes in it, i.e., classifies the object into more than two classes.
Algorithms used	The most popular algorithms used by the binary classification are- • Logistic Regression • k-Nearest Neighbors • Decision Trees • Support Vector Machine • Naive Bayes	Popular algorithms that can be used for multi-class classification include: • k-Nearest Neighbors • Decision Trees • Naive Bayes • Random Forest. • Gradient Boosting
Examples	Examples of binary classification include- • Email spam detection (spam or not). • Churn prediction (churn or not). • Conversion prediction (buy or not).	Examples of multi-class classification include: • Face classification. • Plant species classification. • Optical character recognition.

Naive Bayes classifier

The Naïve Bayes classifier is a supervised machine learning algorithm that is used for classification tasks such as text classification. They use principles of probability to perform classification tasks.

Naïve Bayes is part of a family of generative learning algorithms, meaning that it seeks to model the distribution of inputs of a given class or category. Unlike discriminative classifiers, like logistic regression, it does not learn which features are most important to differentiate between classes.

Naïve Bayes is also known as a probabilistic classifier since it is based on Bayes' Theorem. It would be difficult to explain this algorithm without explaining the basics of Bayesian statistics. This theorem, also known as Bayes' Rule, allows us to "invert" conditional probabilities. As a reminder, conditional probabilities represent the probability of an event given some other event has occurred, which is represented with the following formula:

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

- $P(y|X)$: Posterior probability, probability of class y given features

  P(X|y): Likelihood, probability of features X given class yy

P(y): Prior probability of class

P(X): Marginal likelihood or evidence

Advantages

Easy to implement and computationally efficient.

Effective in cases with a large number of features.

Performs well even with limited training data.

It performs well in the presence of categorical features.

For numerical features data is assumed to come from normal distributions

Disadvantages

- Assumes that features are independent, which may not always hold in real-world data.
- Can be influenced by irrelevant attributes.
- May assign zero probability to unseen events, leading to poor generalization.

Applications

- **Spam Email Filtering**: Classifies emails as spam or non-spam based on features.
- **Text Classification**: Used in sentiment analysis, document categorization and topic classification.
- **Medical Diagnosis:** Helps in predicting the likelihood of a disease based on symptoms.
- **Credit Scoring:** Evaluates creditworthiness of individuals for loan approval.
- **Weather Prediction**: Classifies weather conditions based on various factors.

Example:

	Outlook	Temperature	Humidity	Windy	Play Golf
0	Rainy	Hot	High	False	Yes
1	Rainy	Hot	High	True	No
2	Overcast	Hot	High	False	Yes
3	Sunny	Mild	High	False	No
4	Sunny	Cool	Normal	False	Yes
5	Sunny	Cool	Normal	True	No
6	Overcast	Cool	Normal	True	Yes
7	Rainy	Mild	High	False	No
8	Rainy	Cool	Normal	False	Yes
9	Sunny	Mild	Normal	False	Yes
10	Rainy	Mild	Normal	True	Yes
11	Overcast	Mild	High	True	Yes
12	Overcast	Hot	Normal	False	Yes
13	Sunny	Mild	High	True	No

Let's take a dataset used for predicting if golf is played based on:

- **Outlook**: Sunny, Rainy, Overcast
- **Temperature**: Hot, Mild, Cool
- **Humidity**: High, Normal
- **Wind**: True, False

Class Probabilities:
From dataset of 14 rows:

P(Yes)=14/9

- *P*(No)=14/5

Outlook

	Yes	No	P(yes)	P(no)
Sunny	3	2	3/10	2/4
Overcast	4	0	4/10	0/4
Rainy	3	2	3/10	2/4
Total	10	4	100%	100%

Temperature

	Yes	No	P(yes)	P(no)
Hot	2	2	2/9	2/5
Mild	4	2	4/9	2/5
Cool	3	1	3/9	1/5
Total	9	5	100%	100%

Humidity

	Yes	No	P(yes)	P(no)
Hot	3	4	3/9	4/5
Normal	6	1	6/9	1/5
Total	9	5	100%	100%

Wind

	Yes	No	P(yes)	P(no)
False	6	2	6/9	2/5
True	3	3	3/9	3/5
Total	9	5	100%	100%

Play		P(Yes)/P(No)
Yes	9	9/14
No	5	5/14
Total	14	100%