UNIT-III

UNIT-III: Models Based on Decision Trees: Decision Trees for Classification, Impurity Measures, Properties, Regression Based on Decision Trees, Bias–Variance Trade-off, Random Forests for Classification and Regression. The Bayes Classifier: Introduction to the Bayes Classifier, Bayes' Rule and Inference, The Bayes Classifier and its Optimality, Multi-Class Classification | Class Conditional Independence and Naive Bayes Classifier (NBC)

Decision Tree Models

Decision tree models are a type of supervised learning algorithm that uses a tree-like structure to classify data or make predictions. They work by recursively partitioning the data into subsets based on the most informative features.

Types of Decision Tree Models

1. Classification Trees: Used for classification problems, where the target variable is categorical.

2. Regression Trees: Used for regression problems, where the target variable is continuous.

How Decision Tree Models Work

1. Root Node: The algorithm starts with a root node that represents the entire dataset.

2. Splitting: The algorithm selects the best feature to split the data into two subsets based on a splitting criterion (e.g., Gini impurity, entropy).

3. Child Nodes: The algorithm creates two child nodes, each representing a subset of the data.

4. Recursion: Steps 2-3 are repeated recursively until a stopping criterion is met (e.g., maximum depth, minimum samples per node).

5. Leaf Nodes: The final nodes are leaf nodes, which represent the predicted class labels or target values.

Advantages of Decision Tree Models

1. Interpretable: Decision trees are easy to visualize and interpret.

2. Handling missing values: Decision trees can handle missing values.

3. Non-parametric: Decision trees don't require any assumptions about the data distribution.

Disadvantages of Decision Tree Models

1. Over fitting: Decision trees can suffer from over fitting, especially when the trees are deep.

2. Not suitable for complex relationships: Decision trees are not suitable for modelling complex relationships between features.

Real-World Applications

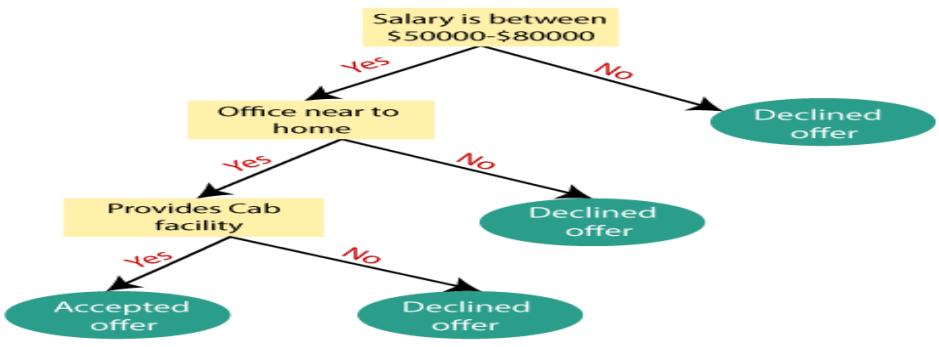
1. Credit risk assessment: Decision trees can be used to predict the likelihood of loan defaults.
2. Medical diagnosis: Decision trees can be used to diagnose diseases based on symptoms and medical history.
3. Customer segmentation: Decision trees can be used to segment customers based on demographic and behavioral features.

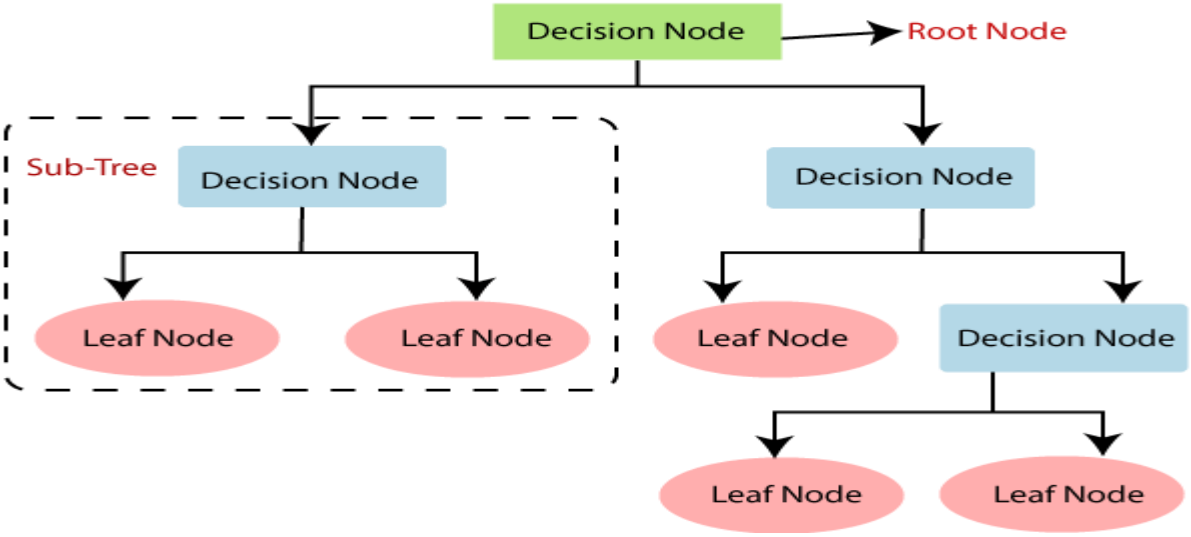
Ensemble Methods

1. Random Forests: An ensemble method that combines multiple decision trees to improve the accuracy and robustness of predictions.
2. Gradient Boosting: An ensemble method that combines multiple decision trees to improve the accuracy and robustness of predictions.

Decision Trees for Classification:

A **Decision Tree** is a supervised machine learning algorithm used for classification and regression tasks. It models decisions based on a tree-like structure, decision trees split the data into subsets based on the value of input features, creating a tree-like model of decisions. Each internal node represents a decision based on a feature, each branch represents the outcome of that decision

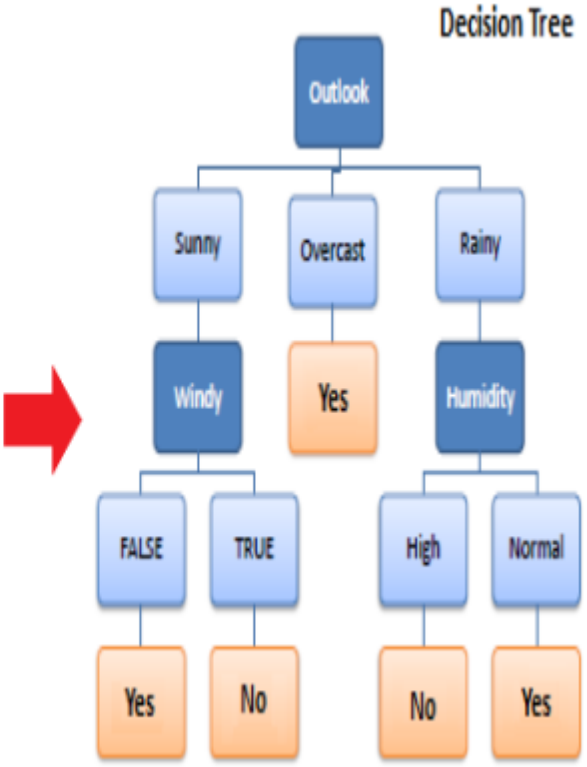




(OR)

Predictors | Target

Outlook	Temp	Humidity	Windy	Play Golf
Rainy	Hot	High	False	No
Rainy	Hot	High	True	No
Overcast	Hot	High	False	Yes
Sunny	Mild	High	False	Yes
Sunny	Cool	Normal	False	Yes
Sunny	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Rainy	Mild	High	False	No
Rainy	Cool	Normal	False	Yes
Sunny	Mild	Normal	False	Yes
Rainy	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Sunny	Mild	High	True	No



How Decision Trees Work

A decision tree follows these steps for classification:

1. **Start with the Root Node:** The dataset is split based on an attribute.
2. **Split into Branches:** Each decision leads to sub-nodes, forming branches.
3. **Continue Splitting Until a Stopping Criterion is Met:** This could be based on purity (all samples belong to one class), maximum depth, or minimum samples per leaf.

2. Key Concepts

(a) Splitting Criteria

To decide the best attribute for splitting, we use:

- **Gini Index**

  $$\text{Gini}(D) = 1 - \sum p_i^2$$

  Measures impurity: lower values mean better splits.

- **Entropy (Information Gain)**

  $$\text{Entropy}(D) = -\sum p_i \log_2 p_i$$

  Measures randomness; lower entropy means better splits.

- **Chi-Square Test**
  - Used to check the statistical significance of splits.

Formula

$$\chi^2 = \sum (\text{Observed} - \text{Expected})^2 / \text{Expected}$$

Advantages: Good for categorical data.
✕ **Disadvantages**: May not work well with small datasets.

- **Gain Ratio**
  - A normalized version of Information Gain to reduce bias.

(b) Stopping Conditions

- Maximum tree depth
- Minimum number of samples per leaf

- No significant improvement in split quality

3. Advantages & Disadvantages

- It is simple to understand as it follows the same process which a human follow while making any decision in real-life.
- It can be very useful for solving decision-related problems.
- It helps to think about all the possible outcomes for a problem.
- There is less requirement of data cleaning compared to other algorithms.

Disadvantages of the Decision Tree

- The decision tree contains lots of layers, which makes it complex.
- It may have an over fitting issue, which can be resolved using the **Random Forest algorithm.**
- For more class labels, the computational complexity of the decision tree may increase.

4. Pruning (Over fitting Prevention)

- **Pre-pruning:** Stops tree growth early (e.g., setting max depth).
- **Post-pruning:** Grows the full tree first, then removes non-informative branches.

Q1: Implement a Decision Tree classifier on the **Iris dataset** and visualize the tree structure. Explain the decision boundaries formed.

Q2: Compare the **Gini Index** and **Entropy** as splitting criteria. Under what conditions would you prefer one over the other?

Impurity Measures:

What is Impurity?

- Impurity refers to the level of "mixedness" or disorder within a set of data points at a node in a decision tree.
- A node with a high impurity contains a mix of different classes, while a pure node contains data points belonging to only one class.
- The goal of a decision tree algorithm is to find splits that reduce the impurity of the parent node, resulting in purer child nodes.

In decision trees, impurity measures help determine **how "mixed" a node is** — meaning, how diverse the class labels are in that node.

- The goal at each step is to **reduce impurity** — to make the child nodes as pure (homogeneous) as possible.
- A pure node has samples mostly from **one class**.

1. Entropy

Based on information theory.

Measures the amount of "uncertainty" or "disorder".

Formula:

Entropy(S)= $-i=1\sum npilog2(pi)$

2. Gini Impurity

- Used in the CART algorithm.
- Measures the probability that a randomly chosen sample is misclassified.

Gini Impurity = 1 - $\sum$ (p^2)

- **Interpretation:**
- A Gini impurity of 0 indicates a "pure" node, meaning all samples belong to the same class.
- A Gini impurity of 0.5 (in a binary classification) or higher indicates a high level of impurity, meaning the node contains a mix of classes.
- **Use in Decision Trees:**
  Decision tree algorithms use Gini impurity (or entropy) to determine the best split for each node, aiming to create nodes with the lowest impurity.

3. Classification Error

Simpler and less sensitive than the others.

Used for **pruning** rather than growing trees.

Formula: Error(S)=1−max(pi)

Characteristics of Impurity Measures

1. **Non-negativity**: Impurity measures should always be non-negative.

2. **Maximum value**: Impurity measures should have a maximum value, usually -1

3. **Minimum value**: Impurity measures should have a minimum value, usually 0.

Choosing an Impurity Measure

1. **Gini Impurity**: Suitable for binary classification problems.
2**. Entropy**: Suitable for multi-class classification problems.
3. **Variance Impurity**: Suitable for regression problems.

Difference between Gini Index and Entropy

It is the probability of misclassifying a randomly chosen element in a set.	While entropy measures the amount of uncertainty or randomness in a set.
The range of the Gini index is **[0, 0.5]**, where 0 indicates perfect purity and 0.5 indicates maximum impurity.	The range of entropy is **[0, log2(C)]**, where c is the number of classes. The range becomes **[0, 1]** for binary classification.
Gini index is a linear measure.	Entropy is a logarithmic measure.
It can be interpreted as the expected error rate in a classifier.	It can be interpreted as the average amount of information needed to specify the class of an instance.

It is the probability of misclassifying a randomly chosen element in a set.	While entropy measures the amount of uncertainty or randomness in a set.
It is sensitive to the distribution of classes in a set.	It is sensitive to the number of classes.
The computational complexity of the Gini index is O(c).	Computational complexity of entropy is O(c * log(c)).
It is less robust than entropy.	It is more robust than Gini index.
It is sensitive.	It is comparatively less sensitive.
Formula for the Gini index is Gini=1–Gini(D)=1−$\sum pi^2$.	Formula for entropy is Entropy(D)=−$\sum pi \log_2 pi$ t.
It has a bias toward selecting splits that result in a more balanced distribution of classes.	It has a bias toward selecting splits that result in a higher reduction of uncertainty.
Gini index is typically used in CART (Classification and	Entropy is typically used in ID3 and C4.5 algorithms

It is the probability of misclassifying a randomly chosen element in a set.	While entropy measures the amount of uncertainty or randomness in a set.
Regression Trees) algorithms	

Properties:

Hierarchical Structure

- A Decision Tree is a **tree-like structure** with:
  - **Root Node**: The first feature used for splitting.
  - **Internal Nodes**: Decision points based on attribute values.
  - **Leaf Nodes**: Final classification output.

2. Recursive Partitioning

- The dataset is **recursively split** based on feature values.
- Each split aims to **reduce impurity** (using Gini Index, Entropy, etc.).
- The process continues until:
  - A stopping criterion is met (e.g., max depth).
  - Nodes become pure (only one class remains).

3. Greedy Algorithm (Top-Down Approach)

- Decision Trees use a **greedy** approach to select the best attribute for splitting at each step.
- The algorithm **does not backtrack**, meaning:
  - The best split is chosen locally, not globally.
  - This may lead to **suboptimal trees** in some cases.

4. Interpretability and Explain ability

- Decision Trees are **easy to understand and interpret**.
- The rules can be represented as **IF-THEN statements**, making them useful for **decision support systems**.

5. Handles both Numerical and Categorical Data

- Decision Trees can handle **both types** of features:
  - **Numerical Features**: Split based on thresholds.
  - **Categorical Features**: Split based on class values.

6. Non-Parametric Model

- Decision Trees do **not make assumptions** about the underlying data distribution.
- Unlike models like Logistic Regression, they **do not require features to be linearly separable**.

7. Prone to over fitting

- If a tree **grows too deep**, it can memorize training data, leading to **over fitting**.
- **Pruning techniques** (pre-pruning & post-pruning) help control over fitting.

8. Sensitive to Small Changes in Data

- A small change in data can lead to **a completely different tree structure**.
- **Solution**: Use **ensemble methods** like Random Forest to stabilize predictions.

9. Computational Efficiency

- **Training Complexity**: O(nlog⁡n)O(n \log n)O(nlogn), where nnn is the number of samples.
- Faster than some models but slower than **Naïve Bayes** or **Logistic Regression** for large datasets.

10. Feature Importance

- Decision Trees naturally provide a **ranking of feature importance**.
- Features used for earlier splits contribute more to predictions.

Regression Based on Decision Trees:

Decision Tree Regression is a machine learning technique that uses a tree-like model to predict continuous numerical values, by recursively splitting data based

on features to minimize prediction error, ultimately leading to leaf nodes that represent the predicted values.

Here's a more detailed explanation:

- **Supervised Learning:**

  Decision Tree Regression is a supervised learning algorithm, meaning it learns from labeled data where the target variable is a continuous numerical value.

- **Tree Structure:**

  The model constructs a tree-like structure where:

- **Root Node:** The starting point of the tree, representing the entire dataset.
- **Internal Nodes:** Represent decision rules or questions based on the input features.
- **Branches:** Represent the possible outcomes of the decision rules.
- **Leaf Nodes:** Represent the final predictions or predicted values.
- **Splitting Data:**

  The algorithm recursively splits the data based on the most informative features, aiming to create subsets where the predicted values are as homogeneous as possible.

- **Prediction:**

  To make a prediction for a new data point, the algorithm follows the path from the root node to a leaf node, based on the values of the input features.

- **Evaluation:**

  The performance of a Decision Tree Regression model can be evaluated using metrics like Mean Squared Error (MSE) or R-squared.

- **Advantages**

  Decision Tree Regression is easy to understand and interpret, and can handle both numerical and categorical data.

- **Disadvantages**

  Decision Tree Regression can be prone to overfitting, especially when the tree is allowed to grow too deep.
  How it works:

1. **Start with the Root Node:** The algorithm begins with the entire dataset as the root node.
2. **Find the Best Split:** It then evaluates different features and split points to find the one that results in the greatest reduction in prediction error (e.g., using Mean Squared Error).

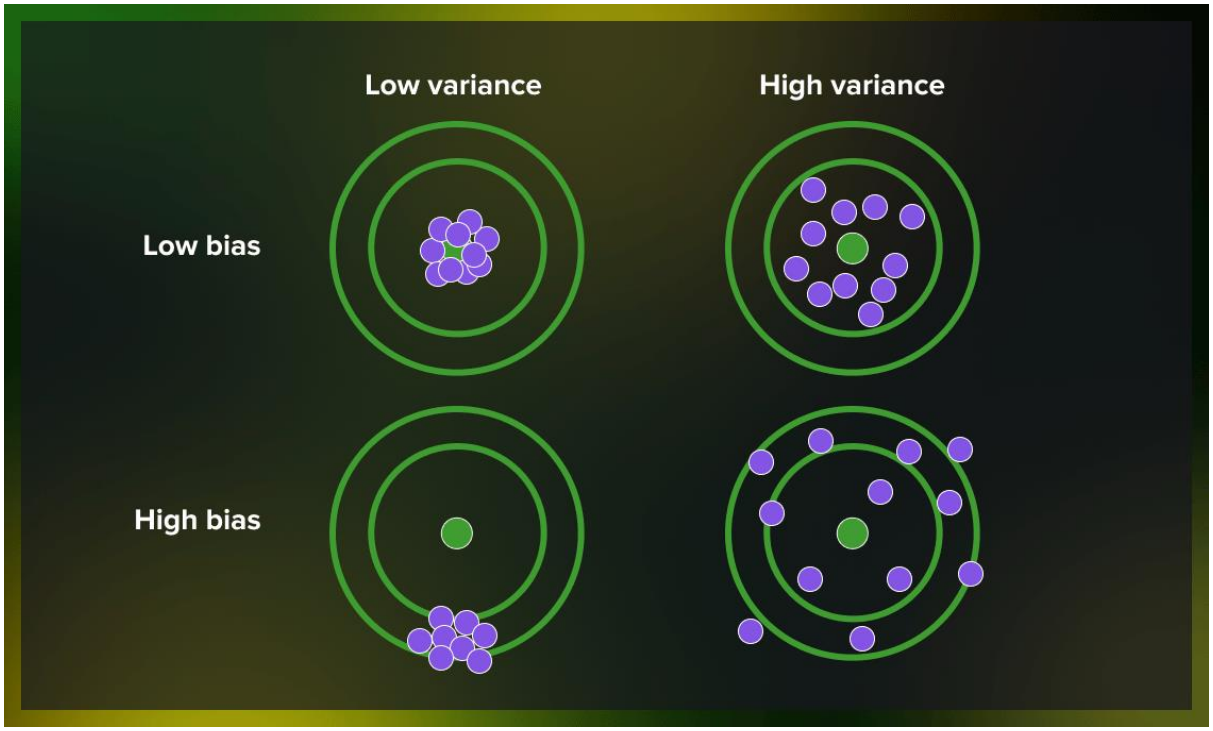
3. **Create Subtrees:** The data is split into two or more subsets based on the chosen split point, creating internal nodes and branches.
4. **Repeat:** The process is repeated for each of the newly created subsets until a stopping criterion is met (e.g., maximum tree depth, minimum number of samples per leaf).
5. **Leaf Nodes:** The leaf nodes contain the predicted values for the corresponding subsets.

Example:

Imagine you want to predict house prices based on features like size, location, and number of bedrooms. A Decision Tree Regression model could:

1. Start with the root node containing all houses.
2. Split the data based on size (e.g., small, medium, large).
3. Further split based on location (e.g., city center, suburbs).
4. Finally, split based on the number of bedrooms.
5. The leaf nodes would then contain the predicted house prices for each of these subsets.

Bias–Variance Trade-off:



What is Bias

Bias is the error introduced by a model due to **oversimplifying** the problem. A high-bias model makes strong assumptions about the data, often missing important patterns and relationships. This results in **underfitting**, where the model performs poorly on both training and test data.

Characteristics of High-Bias Models

- **Too simple**: The model cannot capture complex relationships in the data.
- **High training error**: Even on known data, the model performs poorly.
- **High test error**: The model generalizes poorly to new data.
- **Example algorithms**: Linear regression (for non-linear data), Naïve Bayes, shallow decision trees.

Imagine trying to fit a **straight line (linear regression)** to **non-linear** data (like a sine wave).
Problem: The model is too simple to capture the true trend, leading to **high bias**.

Mathematical Representation:

$\text{Bias} = E[\hat{y}] - y_{\text{true}}$

Where:

- y^ is the model's prediction.
- y_{true} is the actual value.
- E[y^] is the expected prediction over different training sets.

How to Reduce Bias?

✅ Use more complex models (e.g., decision trees, deep learning).
✅ Feature engineering (add important variables, interactions).
✅ Reduce regularization (L1/L2 penalties can oversimplify models).
✅ Use ensemble methods (random forests, boosting) to improve learning.

What is Variance?

Variance refers to the model's sensitivity to small fluctuations in the training data. A high-variance model learns noise along with the actual patterns, leading to **over fitting**. This means it performs well on the training data but poorly on unseen data.

Characteristics of High-Variance Models

🔺 **Too complex**: The model captures noise along with real patterns.
🔺 **Low training error**: It fits the training data almost perfectly.
🔺 **High test error**: Poor generalization to new data.
🔺 **Example algorithms**: Deep decision trees, high-degree polynomial regression, k-NN with low kkk.

Example of Variance in Regression

Imagine fitting a **very high-degree polynomial** to a dataset.
📌 **Problem**: The model captures even small fluctuations (noise), leading to **high variance**.

Mathematical Representation

Variance measures how much the model's predictions change with different training sets:

$$\text{Variance}=E[(\hat{y}-E[\hat{y}])2]$$

Where:

- y^ is the model's prediction.
- E[y^] is the expected prediction over different training sets.

How to Reduce Variance?

✅ Simplify the model (reduce complexity, prune trees).
✅ Use regularization techniques (L1/L2, dropout in neural networks).
✅ Get more training data to reduce noise influence.
✅ Use ensemble methods (bagging, boosting) to stabilize predictions.

Bias–Variance Trade-off in Machine Learning

The **Bias–Variance Trade-off** is a fundamental concept in machine learning that helps balance model complexity and generalization. It explains the trade-off between two types of errors that affect a model's performance:

1. **Bias** (Under fitting)
2. **Variance** (Overfitting)

High Bias (Under fitting)

- Model is too simple to capture the underlying pattern.
- Example: Using a **linear regression** model to fit non-linear data.
- Results in **high training and high test error**.

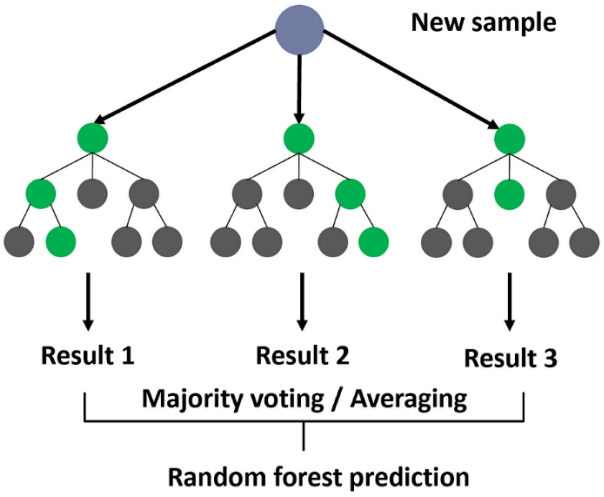
High Variance (Over fitting)

- Model is too complex and learns noise from training data.
- Example: A **deep decision tree** with no pruning.
- Results in **low training error but high test error**.

Random Forests for Classification and Regression:

What is Random Forest Classification?

Random Forest is an **ensemble learning method** that combines multiple decision trees to improve classification accuracy and reduce over fitting. It is based on the **Bagging (Bootstrap Aggregation) technique**.



How Random Forest Works for Classification

1. **Bootstrap Sampling**: Random subsets of the training data are used to train each decision tree.
2. **Random Feature Selection**: Each tree considers only a random subset of features for splitting.
3. **Training Multiple Trees**: Each tree is trained independently on its subset.
4. **Voting for Prediction**:
5. **Classification**: Uses **majority voting** from all trees.
6. **Regression**: Takes the **average** of all tree predictions.

Advantages of Random Forest Classification

Handles high-dimensional data **effectively.**
Reduces over fitting **compared to a single decision tree.**
Works well with missing data and noisy datasets.
Parallelizable**, making it efficient for large datasets.**

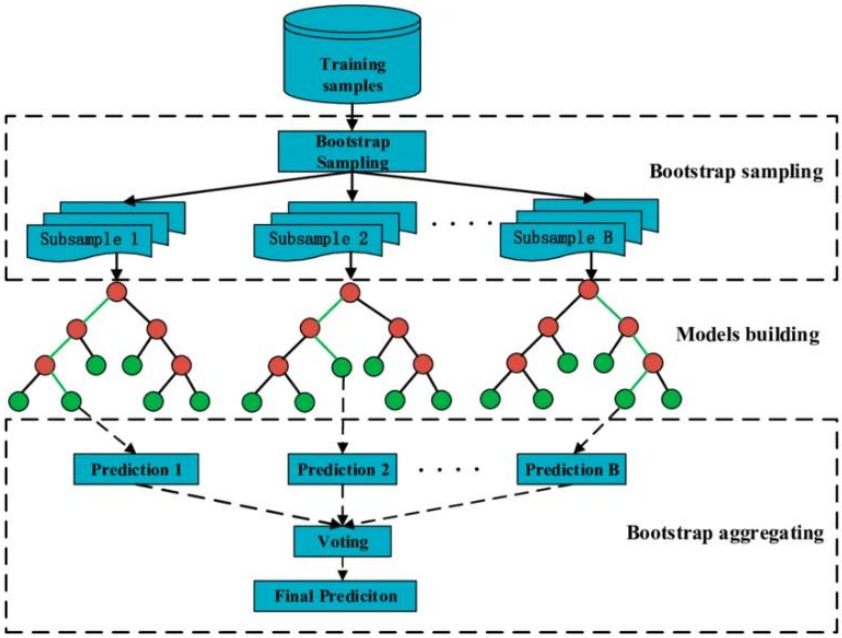
Disadvantages

Can be computationally expensive for very large datasets.
Less interpretable than a single decision tree.

RANDOM FORESTS FOR REGRESSION:

What is Random Forest Regression?

Random Forest Regression is an **ensemble learning method** that uses multiple decision trees to make predictions for **continuous target variables**. It is based on the **Bagging (Bootstrap Aggregation) technique**.



How Random Forest Works for Regression

1. **Bootstrap Sampling**: Random subsets of training data are used to train each decision tree.
2. **Random Feature Selection**: Each tree considers a random subset of features for splitting.
3. **Training Multiple Trees**: Each tree learns independently on its data subset.
4. **Averaging for Prediction**:
    - **Regression**: The final prediction is the **average** of all individual tree predictions.

Advantages of Random Forest Regression

Reduces over fitting compared to a single decision tree.
Handles non-linear relationships in data.
Works well with missing values and noisy data.
Supports high-dimensional datasets with many features.

Disadvantages

Can be **computationally expensive** for large datasets.
Less interpretable compared to a single decision tree.

THE BAYES CLASSIFIER: INTRODUCTION TO THE BAYES CLASSIFIER:

What is the Bayes Classifier?

The **Bayes Classifier** is a **probabilistic model** that classifies data points based on **Bayes' Theorem**. It predicts the **most probable class** for a given data point based on prior knowledge.

Bayes' Theorem

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

- P(A|B): This is the probability that event A is true given that B is true. In other words, if we know B has occurred (e.g., the email contains the word "offer"), we want to determine how likely it is that A is true (the email is spam).
- P(B|A): This is the probability that B is true if A is true. So, if we know the email is spam (A), this probability tells us how likely it is to contain the word "offer" (B).
- P(A): This represents the prior probability of A, meaning how likely it is that an email is spam before considering specific features (B).
- P(B): This is the prior probability of B, which refers to how likely it is to observe feature B in general ("offer"), regardless of A.

Key Features of Naive Bayes Classifiers

- The Naive Bayes Classifier is a simple probabilistic classifier and it has very few number of parameters which are used to build the ML models that can predict at a faster speed than other classification algorithms.

- It is a probabilistic classifier because it assumes that one feature in the model is independent of existence of another feature. In other words, each feature contributes to the predictions with no relation between each other.

- Naïve Bayes Algorithm is used in spam filtration, Sentimental analysis, classifying articles and many more.

BAYES' RULE AND INFERENCE:

Bayes' rule, a cornerstone of Bayesian inference, allows us to update our beliefs about a hypothesis (or event) based on new evidence, moving from a prior probability to a posterior probability.

Here's a breakdown:

- **Bayes' Rule:**
- The formula is P(A|B) = [P(B|A) * P(A)] / P(B), where:
- P(A|B) is the posterior probability of A given B (updated belief).
- P(B|A) is the likelihood of B given A (probability of evidence given hypothesis).
- P(A) is the prior probability of A (initial belief).
- P(B) is the probability of the evidence B.
- **Bayesian Inference:**

  This is a statistical method that uses Bayes' rule to update probabilities based on new evidence, leading to a more informed understanding of a hypothesis.
- **Key Concepts:**
- **Prior Probability:** Your initial belief about the likelihood of an event before observing any evidence.
- **Likelihood:** The probability of observing the evidence, assuming the hypothesis is true.
- **Posterior Probability:** The updated belief about the hypothesis after considering the evidence.
- **How it Works:**

Bayesian inference starts with a prior probability, then uses Bayes' rule to calculate the posterior probability, which is the updated belief after observing the evidence.

- **Applications:**

  Bayesian inference is used in various fields, including:

- **Machine Learning:** Building predictive models.
- **Medicine:** Diagnosing diseases.
- **Science:** Analysing data and drawing conclusions.
- **Finance:** Assessing risks and making investment decisions.

THE BAYES CLASSIFIER AND ITS OPTIMALITY:

The Bayes classifier is a theoretically optimal classifier that minimizes the probability of misclassification by selecting the class with the highest posterior probability, serving as a benchmark for other classification algorithms.

Here's a more detailed explanation:

- **Theoretical Foundation:**

  The Bayes classifier is based on Bayes' theorem, which describes how to update the probability of a hypothesis given new evidence.

- **Optimality:**

  It's considered the "best" classifier in a theoretical sense because it achieves the lowest possible error rate (also known as Bayes risk) for a given classification problem.

- **How it Works:**

  For a given input, the Bayes classifier assigns the input to the class with the highest posterior probability (the probability of the class given the input).

- **Practical Limitations:**

  In practice, the Bayes classifier is difficult to implement because it requires knowing the true underlying probability distributions of the classes, which are often unknown.

- **Importance as a Benchmark:**

  Despite its impracticality, the Bayes classifier is a valuable concept because it provides a theoretical lower bound on the achievable error rate, allowing researchers to evaluate the performance of other classification algorithms.

- **Naive Bayes:**

  A simplified version of the Bayes classifier, the Naive Bayes classifier, makes the assumption that features are conditionally independent given the class, which simplifies the calculations but can also lead to a loss of accuracy.

- **Zero-One Loss:**

  The Bayes classifier is optimal under zero-one loss, which means it minimizes the misclassification rate.

- **Applications:**

  While the Bayes classifier itself is not directly used in practice, the underlying principles are used in various machine learning algorithms and applications

MULTI CLASS CLASSIFICATION:

Naive Bayes algorithms can be effectively used for multi-class classification by calculating the probability of each class label and then predicting the class with the highest probability.

Here's a more detailed explanation:

- **Probabilistic Classifier:**

  Naive Bayes is a probabilistic classifier, meaning it uses Bayes' Theorem to calculate the probability of a data point belonging to a particular class.

- **Multi-class Extension:**

  The core concept of Naive Bayes extends naturally to multi-class classification. Instead of predicting a binary outcome (e.g., spam/not spam), it predicts the probability of belonging to any of the multiple classes.

- **Class with Highest Probability:**

  For each data point, the algorithm calculates the probability of it belonging to each class. The class with the highest probability is then assigned as the prediction.

- **Naive Assumption:**

  The algorithm is "naive" because it assumes that the presence of a particular feature in a class is unrelated to the presence of any other feature. This is a simplifying assumption that makes the calculations tractable.

- **Types of Naive Bayes:**

  There are different types of Naive Bayes classifiers, such as Gaussian Naive Bayes (for continuous data) and Multinomial Naive Bayes (for discrete data like text).

- **Applications:**

  Naive Bayes is commonly used in tasks like text classification, spam detection, and sentiment analysis.

- **Advantages:**

  It's a simple, fast, and relatively accurate algorithm, especially for high-dimensional data.

- **Disadvantages:**

  The strong independence assumption can be a limitation in some cases, but it often still performs well in practice.

CLASS CONDITIONAL INDEPENDENCE AND NAIVE BAYES CLASSIFIER (NBC)

The Naive Bayes Classifier (NBC) relies on the assumption of class-conditional independence, meaning that each feature's presence (or absence) in a class is independent of other features, given the class label. This simplifies the classification process, allowing for efficient and often accurate predictions.

Here's a more detailed explanation:

- **Class-Conditional Independence:**

  This is the core assumption of NBC. It means that the probability of a feature being a certain value, given a specific class, is independent of the values of other features, within that same class.

- **Example:** In a spam filter, the presence of the word "discount" is assumed to be independent of the presence of the word "free" (given that the email is spam), even though they might occur together.

- **How it Simplifies the Model:**

- The assumption allows the model to calculate the probability of a data point belonging to a class by simply multiplying the probabilities of each feature being present (or absent) in that class.
- This greatly simplifies the calculation compared to considering the complex interactions between features.

Why it's "Naive":

- The assumption of conditional independence is often unrealistic in real-world scenarios, as features are frequently correlated.
- This is why the algorithm is considered "naive" - it makes a strong, simplifying assumption that might not always hold true.
- **Despite the Assumption:**
- Despite the unrealistic nature of the assumption, Naive Bayes classifiers often perform surprisingly well in practice, especially for tasks like text classification and spam filtering.
- This is because the algorithm is computationally efficient and can be trained quickly, even with large datasets.
- **Key Concepts:**
- **Bayes' Theorem:** NBC is based on Bayes' theorem, which describes how to update the probability of a hypothesis (class) given new evidence (feature values).
- **Prior Probability:** The probability of a class before considering any evidence.
- **Likelihood:** The probability of the evidence given the hypothesis (class).
- **Posterior Probability:** The updated probability of the hypothesis (class) after considering the evidence.

class conditional independence, we simplify $P(X|C_k)$

$P(C_k|X1,X2,...,Xn) \propto P(C_k) \prod^{n} P(X_i|C_k)$