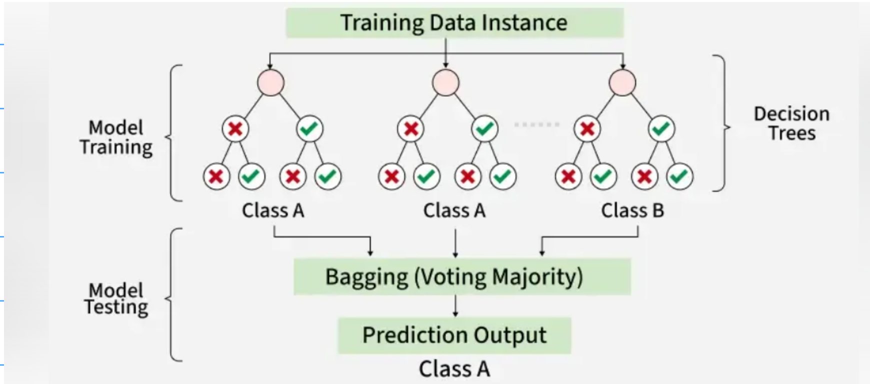
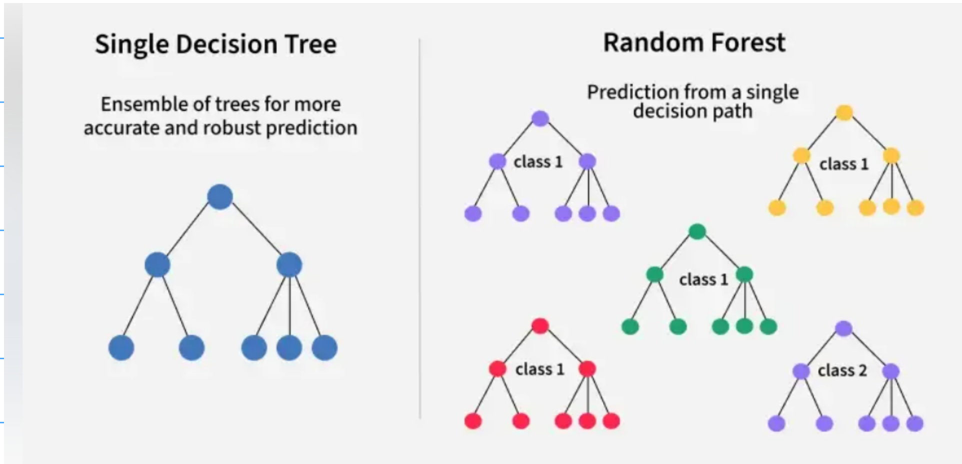
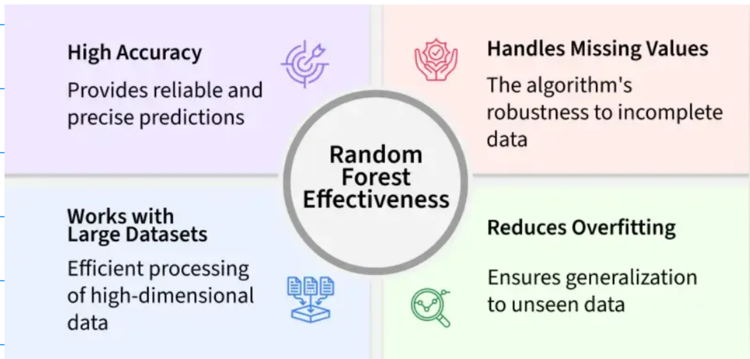
Random Forest-

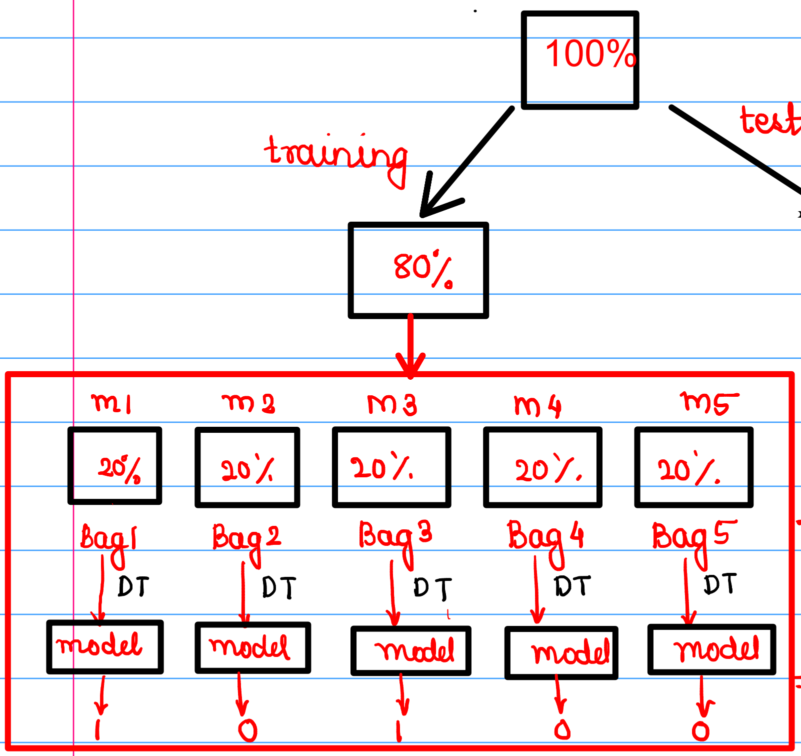
Random Forest works by building many decision trees on random subsets of data and features, then combining their individual predictions through majority voting (classification) or averaging (regression) to create a single, more accurate, and robust model, preventing overfitting and improving performance







age	score
20	30
⋮	⋮



Random Forest model

⇒ Mode [all answers]

⇒ 0

= (voting feature)

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.

The Random Forest algorithm can be used for identifying the most important features from the training dataset, in other words, feature engineering

earlier we used to take correlation or corefficient

but now if you want it in much better way then go for this bagging,boosting techniques (ensemble learning) a technique to simplify complex decision trees by removing unnecessary branches or nodes, reducing overfitting to improve generalization on new data, making the model smaller, faster, and more accurate for real-world predictions

Why Pruning is Needed (The Overfitting Problem)
Overfitting: Decision trees, if grown too deep, can memorize noise in the training data instead of learning general patterns, leading to poor performance on unseen data.
Complexity: Overly complex trees are computationally expensive and harder to interpret.

weak learner ==>model which has not learned much from training
(low performance)

classification type to have more than two classes;
this is known as multiclass classification or multinomial classification.
In multiclass classification, each data sample is assigned to exactly one category from a predefined set of three or more possible classes.

Key Concepts
Binary Classification: Involves only two possible classes
(e.g., "spam" or "not spam", "apple" or "not apple").

Multiclass Classification: Involves three or more classes where each instance belongs to a single, mutually exclusive class
(e.g., classifying an image of a fruit as either "apple", "peach", or "orange").

Multi-label Classification: A related but different concept where a single instance can be assigned to multiple labels simultaneously
(e.g., a movie tagged as both "action" and "comedy").

Common Algorithms
Many machine learning algorithms natively support multiclass classification, while others can be extended to handle it using specific strategies:
Native Multiclass Classifiers:
Decision Trees
k-Nearest Neighbors (kNN)
Naive Bayes
Random Forest
Neural Networks (often using a softmax output layer to assign probabilities)

Multiclass classification is a very common task in real-world applications, such as handwriting recognition, medical diagnosis, and product categorization.
Multiclass Classification in Machine