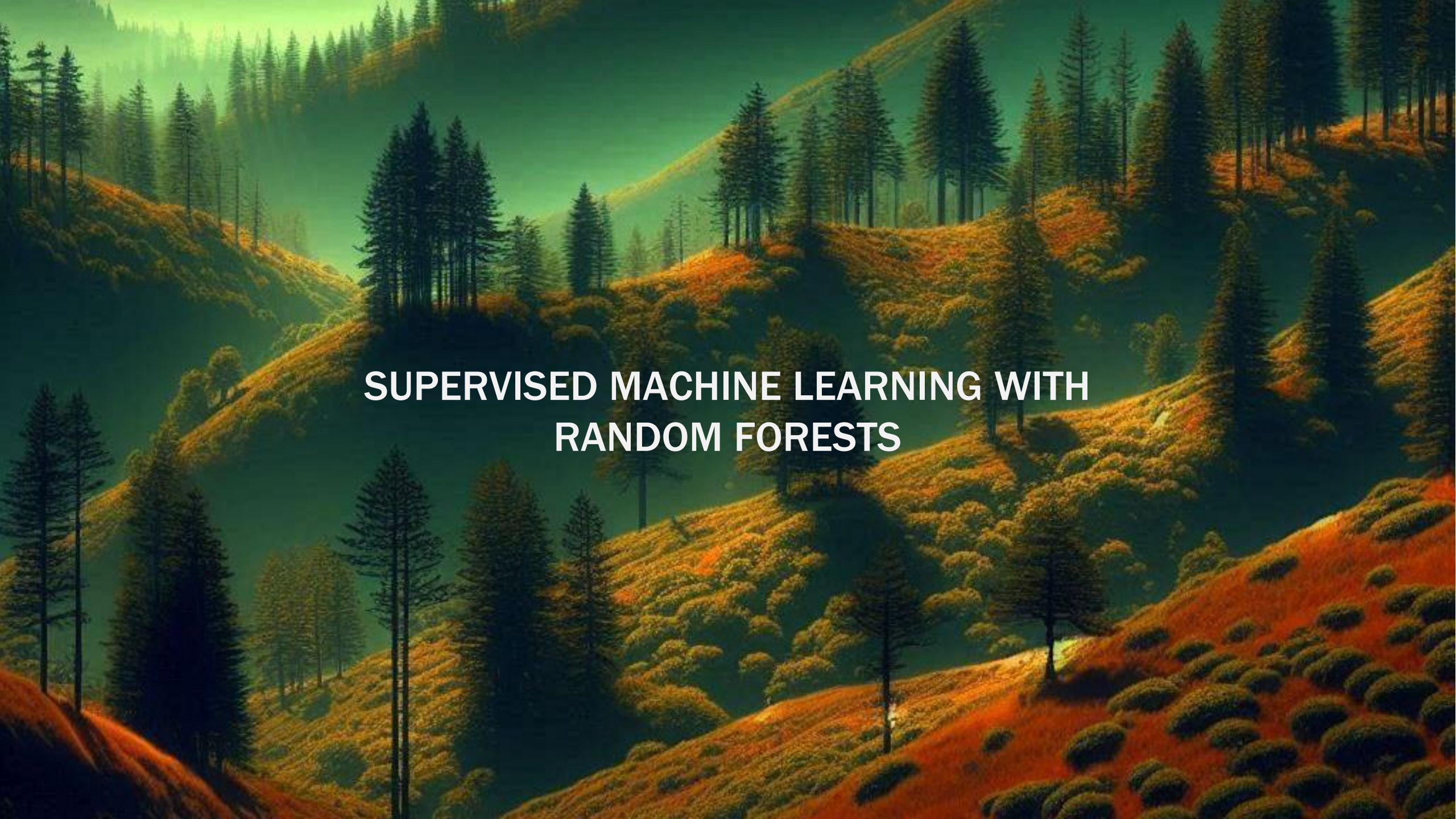
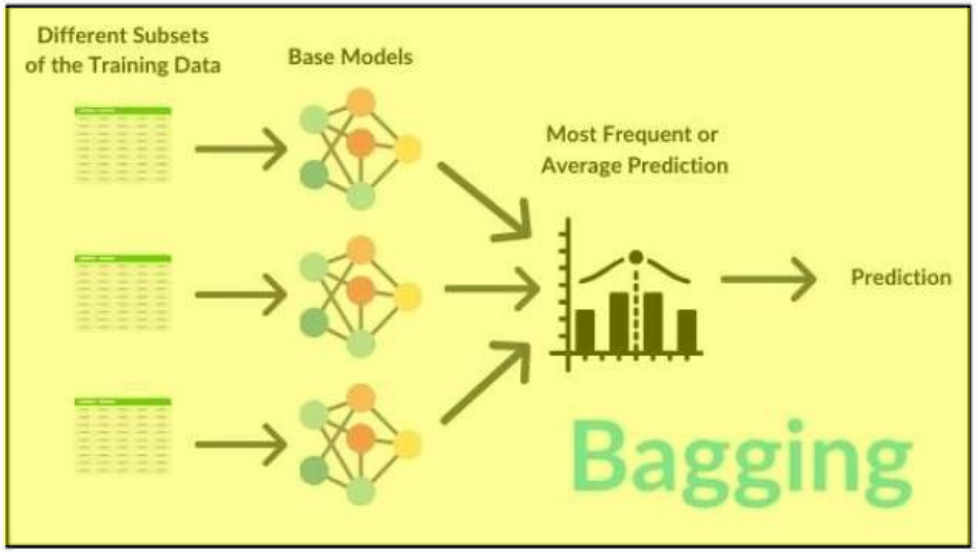

SUPERVISED MACHINE LEARNING WITH
RANDOM FORESTS

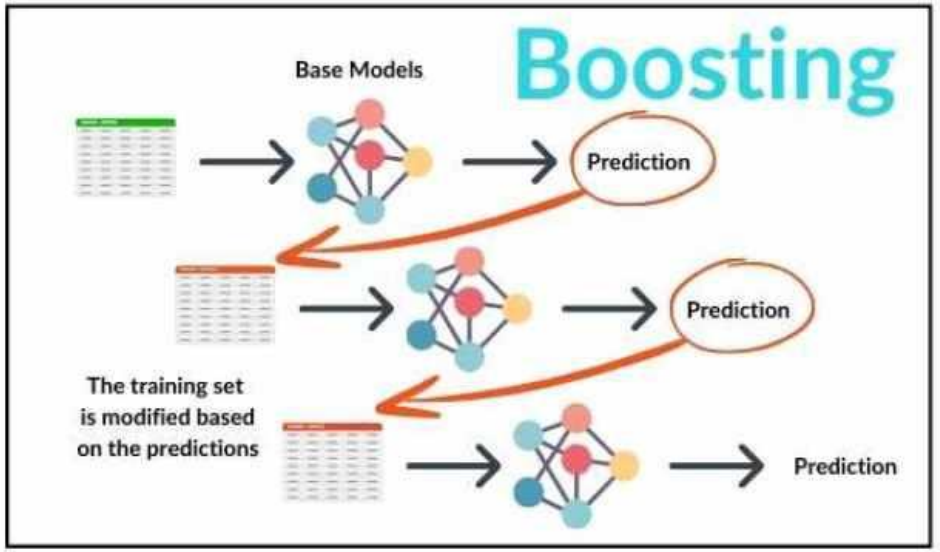
ENSEMBLE LEARNING

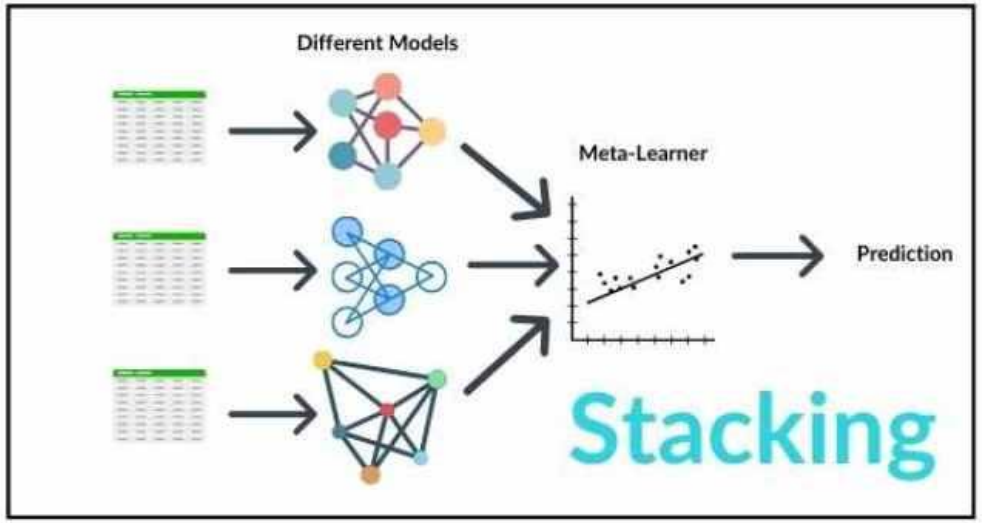
Ensemble learning is a machine learning paradigm where multiple models, often referred to as "weak learners," are combined to produce a more robust and accurate predictive model. The fundamental idea is that by aggregating the predictions of several models, the ensemble can outperform any individual constituent model. This approach leverages the diversity among models to reduce errors, enhance generalization, and improve performance on unseen data.

- **Bagging (Bootstrap Aggregating)**: This technique involves training multiple models on different subsets of the training data, each generated through **bootstrapping** (**sampling with replacement**). The final prediction is typically made by averaging the outputs (for regression) or taking a majority vote (for classification) of the individual models. Bagging primarily aims to reduce variance and prevent overfitting. Wikipedia
- **Boosting**: Boosting sequentially trains models, where each new model focuses on correcting the errors made by its predecessors. Models are added until no further significant improvement can be achieved. This method reduces both bias and variance, leading to strong predictive performance. Analytics Vidhya
- **Stacking (Stacked Generalization)**: In stacking, multiple diverse models are trained, and their predictions are combined using a meta-model (often a simple model) that learns to combine the base models' outputs best. This approach leverages the strengths of various models to improve overall performance. Baeldung

FLAVORS OF ENSEMBLE LEARNING: BAGGING, BOOSTING, STACKING





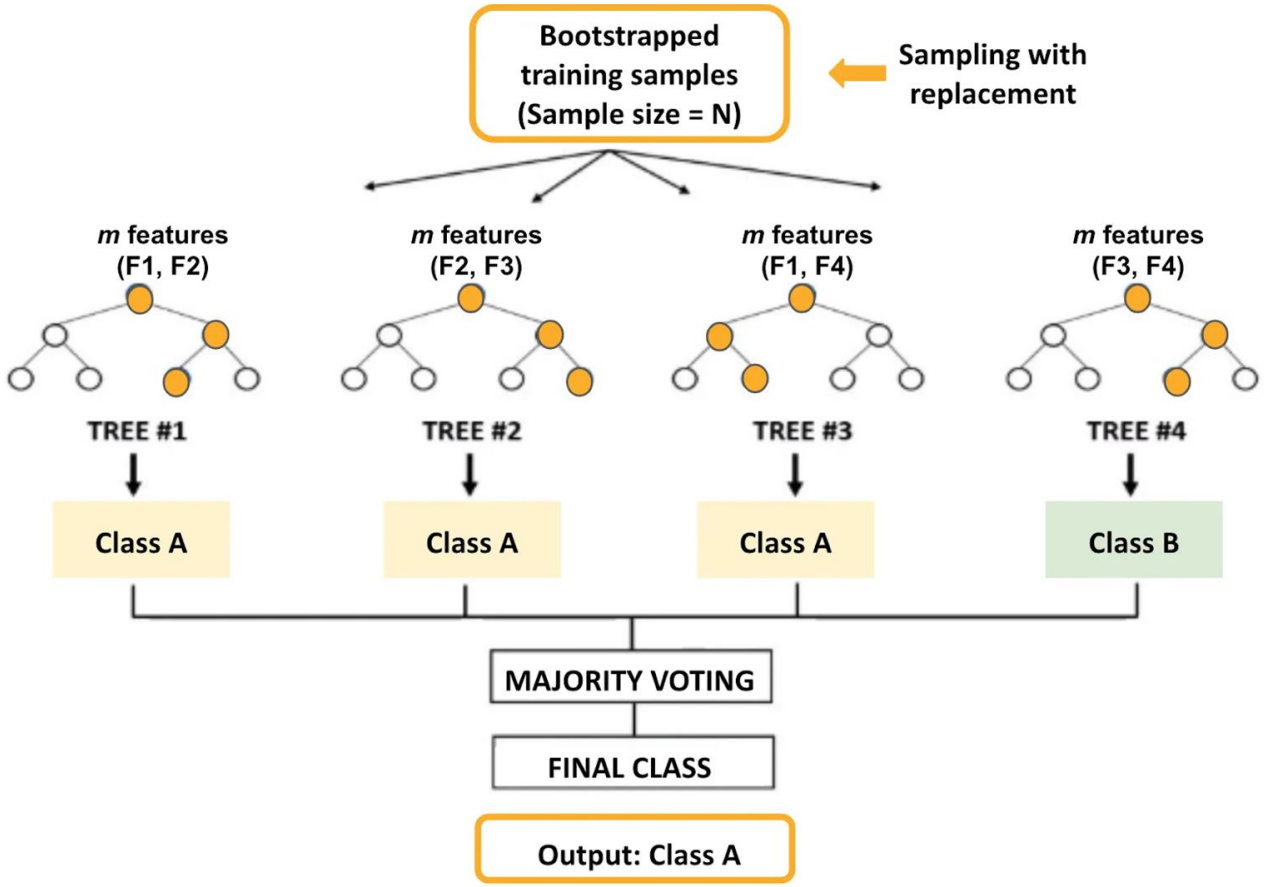


RANDOM FOREST

The **Random Forest** algorithm is an ensemble learning technique primarily used for **classification** and **regression**. It builds multiple decision trees during training and combines their outputs to improve performance and reduce overfitting.

Key Concepts

1. **Ensemble Learning**: Combines predictions from multiple models (decision trees) to produce a more robust and accurate prediction.
2. **Bagging (aka Bootstrap Aggregation)**:
   - Creates multiple subsets of the training data by sampling with replacement.
   - Each subset is used to train a decision tree.
3. **Random Feature Selection**:
   - At each split in a decision tree, a random subset of features is considered, making trees less correlated.
4. **Voting/Averaging**:
   - For classification, majority vote across trees is taken.
   - For regression, the average prediction of all trees is computed.



RANDOM FOREST PARAMETERS

Parameters

1. **n_estimators**: Number of trees in the forest.

• Larger values may improve performance but increase computation time.

2. **max_depth**: Maximum depth of the trees.

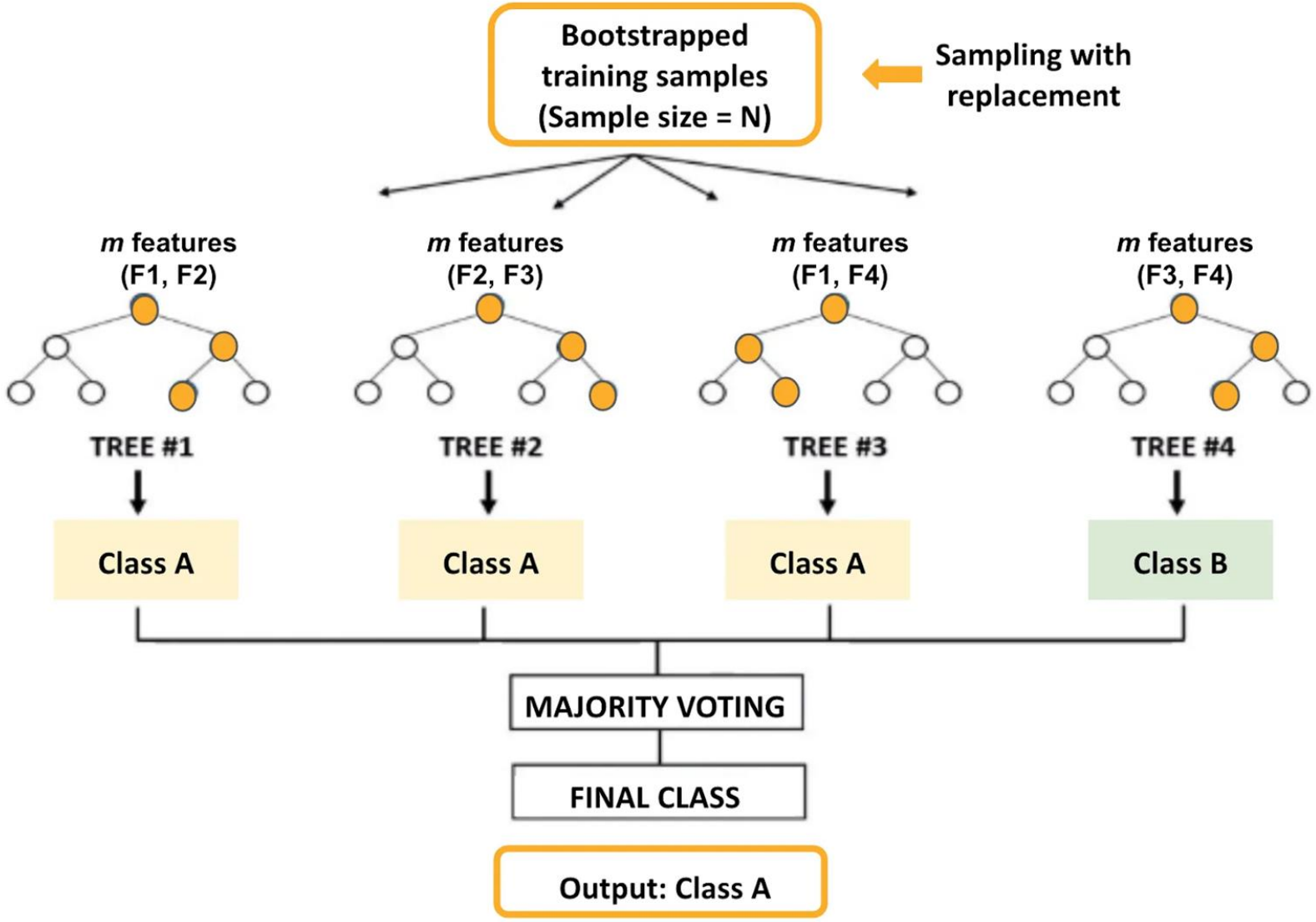
• Controls overfitting; smaller values reduce overfitting but may underfit.

3. **min_samples_split**: Minimum number of samples required to split a node.

4**. min_samples_leaf**: Minimum number of samples required to be at a leaf node.

5. **max_features**: Number of features considered at each split.

• Options: “sqrt”, “log2”, or a fixed number of features.



FEATURE IMPORTANCE

Random Forest provides a mechanism to measure the importance of each feature in making predictions. Two primary methods are employed to calculate feature importance in Random Forests:

1. Mean Decrease in Impurity (MDI): [This is the default]

MDI evaluates the importance of a feature based on the total reduction of a chosen impurity measure (such as Gini impurity or entropy) brought by that feature across all trees in the forest. The process involves:

- **Impurity Reduction Calculation:** For each tree, whenever a feature is used to split a node, the algorithm computes the decrease in impurity resulting from that split.
- **Summation Across Trees:** These impurity reductions are summed for each feature across all trees in the forest.
- **Normalization:** The summed values are then normalized to provide a relative importance score for each feature.

Features that result in larger impurity reductions are considered more important. However, MDI can be biased towards features with more categories or continuous features with many unique values. Wikipedia

2. Permutation Importance:

Permutation importance assesses the impact of each feature by measuring the change in the model's performance when the feature's values are randomly shuffled. The steps include:

- **Baseline Performance Measurement:** Evaluate the model's performance on a validation set to establish a baseline.
- **Feature Shuffling:** Randomly permute the values of the feature under consideration, breaking the relationship between the feature and the target variable.
- **Performance Re-evaluation:** Measure the model's performance on the modified dataset.
- **Importance Calculation:** The importance score is determined by the decrease in performance due to the shuffling. A significant drop indicates that the feature is important for accurate predictions.

Permutation importance is model-agnostic and provides a more unbiased estimate of feature importance compared to MDI. Scikit-learn

```python
# Plot feature importances
importances = rf_model.feature_importances_
indices = np.argsort(importances)[::-1]
features = [f"Feature {i}" for i in range(X.shape[1])]

plt.figure(figsize=(10, 6))
plt.title("Feature Importance")
plt.bar(range(X.shape[1]), importances[indices], align="center")
plt.xticks(range(X.shape[1]), [features[i] for i in indices], rotation=45)
plt.xlabel("Feature")
plt.ylabel("Importance")
plt.show()
```