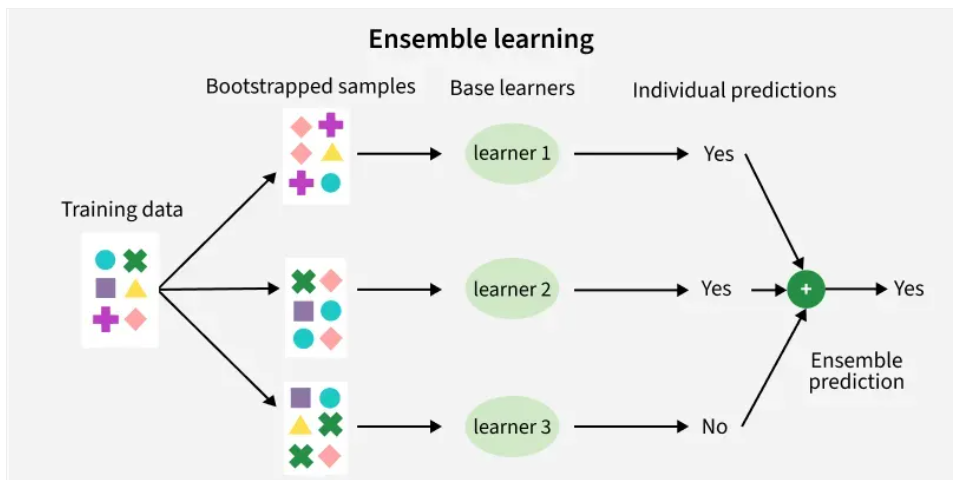
**Ensemble Learning**

**What is Ensemble Learning:**

<https://www.geeksforgeeks.org/a-comprehensive-guide-to-ensemble-learning/>

Ensemble learning is a method where we use many small models instead of just one. Each of these models may not be very strong on its own, but when we put their results together, we get a better and more accurate answer. It's like asking a group of people for advice instead of just one person—each one might be a little wrong, but together, they usually give a better answer.



**Types of Ensembles Learning in Machine Learning**

There are three main types of ensemble methods:

1. **Bagging (Bootstrap Aggregating):**  
   Models are trained independently on different random subsets of the training data. Their results are then combined—usually by averaging (for regression) or voting (for classification). This helps reduce variance and prevents overfitting.
2. **Boosting:**  
   Models are trained one after another. Each new model focuses on fixing the errors made by the previous ones. The final prediction is a weighted combination of all models, which helps reduce bias and improve accuracy.
3. **Stacking (Stacked Generalization):**  
   Multiple different models (often of different types) are trained, and their predictions are used as inputs to a final model, called a meta-model. The meta-model learns how to best combine the predictions of the base models, aiming for better performance than any individual model.

**1. Bagging Algorithm**

[Bagging classifier](https://www.geeksforgeeks.org/ml-bagging-classifier/) can be used for both regression and classification tasks. Here is an overview of Bagging classifier algorithm**:**

* **Bootstrap Sampling:** Divides the original training data into ‘N’ subsets and randomly selects a subset with replacement in some rows from other subsets. This step ensures that the base models are trained on diverse subsets of the data and there is no class imbalance.
* Base Model Training: For each bootstrapped sample we train a base model independently on that subset of data. These weak models are trained in parallel to increase computational efficiency and reduce time consumption. We can use different base learners i.e. different ML models as base learners to bring variety and robustness.
* **Prediction Aggregation:** To make a prediction on testing data combine the predictions of all base models. For classification tasks it can include majority voting or weighted majority while for regression it involves averaging the predictions.
* **Out-of-Bag (OOB) Evaluation**: Some samples are excluded from the training subset of particular base models during the bootstrapping method. These “out-of-bag” samples can be used to estimate the model’s performance without the need for cross-validation.
* **Final Prediction:** After aggregating the predictions from all the base models, Bagging produces a final prediction for each instance.

**Python pseudo code for Bagging Estimator implementing libraries:**

**1. Importing Libraries and Loading Data**

* **BaggingClassifier:** for creating an ensemble of classifiers trained on different subsets of data.
* **DecisionTreeClassifier:** the base classifier used in the bagging ensemble.
* **load\_iris:** to load the Iris dataset for classification.
* **train\_test\_split:** to split the dataset into training and testing subsets.
* **accuracy\_score**: to evaluate the model’s prediction accuracy.

from sklearn.ensemble import BaggingClassifier

from sklearn.tree import DecisionTreeClassifier

from sklearn.datasets import load\_iris

from sklearn.model\_selection import train\_test\_split

from sklearn.metrics import accuracy\_score

**2. Loading and Splitting the Iris Dataset**

* **data = load\_iris():**loads the Iris dataset, which includes features and target labels.
* **X = data.data:** extracts the feature matrix (input variables).
* **y = data.target:** extracts the target vector (class labels).
* **train\_test\_split(...):**splits the data into training (80%) and testing (20%) sets, with random\_state=42 to ensure reproducibility.

data = load\_iris()

X = data.data

y = data.target

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

**3. Creating a Base Classifier**

**Decision tree is chosen as the base model. They are prone to overfitting when trained on small datasets making them good candidates for bagging.**

* **base\_classifier = DecisionTreeClassifier()**: initializes a Decision Tree classifier, which will serve as the base estimator in the Bagging ensemble.

base\_classifier = DecisionTreeClassifier()

**4. Creating and Training the Bagging Classifier**

* A **BaggingClassifier**is created using the decision tree as the base classifier.
* **n\_estimators = 10** specifies that 10 decision trees will be trained on different bootstrapped subsets of the training data.

bagging\_classifier = BaggingClassifier(base\_classifier, n\_estimators=10, random\_state=42)

bagging\_classifier.fit(X\_train, y\_train)

**5. Making Predictions and Evaluating Accuracy**

* The trained bagging model predicts labels for test data.
* The accuracy of the predictions is calculated by comparing the predicted labels (**y\_pred**) to the actual labels (**y\_test**).

y\_pred = bagging\_classifier.predict(X\_test)

accuracy = accuracy\_score(y\_test, y\_pred)

print("Accuracy:", accuracy)

**Output:**

*Accuracy: 1.0*

**2. Boosting Algorithm**

[Boosting](https://www.geeksforgeeks.org/boosting-in-machine-learning-boosting-and-adaboost/) is an ensemble technique that combines multiple weak learners to create a strong learner. Weak models are trained in series such that each next model tries to correct errors of the previous model until the entire training dataset is predicted correctly. One of the most well-known boosting algorithms is [AdaBoost (Adaptive Boosting).](https://www.geeksforgeeks.org/implementing-the-adaboost-algorithm-from-scratch/) Here is an overview of Boosting algorithm:

* **Initialize Model Weights**: Begin with a single weak learner and assign equal weights to all training examples.
* **Train Weak Learner**: Train weak learners on these dataset.
* **Sequential Learning**: Boosting works by training models sequentially where each model focuses on correcting the errors of its predecessor. Boosting typically uses a single type of weak learner like decision trees.
* **Weight Adjustment**: Boosting assigns weights to training datapoints. Misclassified examples receive higher weights in the next iteration so that next models pay more attention to them.

**Python pseudo code for boosting Estimator implementing libraries:**

**1. Importing Libraries and Modules**

* **AdaBoostClassifier from sklearn.ensemble:** for building the AdaBoost ensemble model.
* **DecisionTreeClassifier from sklearn.tree:** as the base weak learner for AdaBoost.
* **load\_iris from sklearn.datasets:**to load the Iris dataset.
* **train\_test\_split from sklearn.model\_selection:**to split the dataset into training and testing sets.
* **accuracy\_score from sklearn.metrics:**to evaluate the model’s accuracy.

from sklearn.ensemble import AdaBoostClassifier

from sklearn.tree import DecisionTreeClassifier

from sklearn.datasets import load\_iris

from sklearn.model\_selection import train\_test\_split

from sklearn.metrics import accuracy\_score

**2. Loading and Splitting the Dataset**

* **data = load\_iris(): loads the Iris dataset, which includes features and target labels.**
* **X = data.data: extracts the feature matrix (input variables).**
* **y = data.target: extracts the target vector (class labels).**
* **train\_test\_split(...): splits the data into training (80%) and testing (20%) sets, with random\_state=42 to ensure reproducibility.**

data = load\_iris()

X = data.data

y = data.target

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

**3. Defining the Weak Learner**

We are creating the base classifier as a decision tree with maximum depth 1 (a decision stump). This simple tree will act as a weak learner for the**AdaBoost algorithm**, which iteratively improves by combining many such weak learners.

base\_classifier = DecisionTreeClassifier(max\_depth=1)

**4. Creating and Training the AdaBoost Classifier**

* **base\_classifier: The weak learner used in boosting.**
* **n\_estimators = 50: Number of weak learners to train sequentially.**
* **learning\_rate = 1.0: Controls the contribution of each weak learner to the final model.**
* **random\_state = 42: Ensures reproducibility.**

adaboost\_classifier = AdaBoostClassifier(

base\_classifier, n\_estimators=50, learning\_rate=1.0, random\_state=42

)

adaboost\_classifier.fit(X\_train, y\_train)

**5. Making Predictions and Calculating Accuracy**

We are calculating the accuracy of the model by comparing the true labels **y\_test**with the predicted labels **y\_pred**. The accuracy\_score function returns the proportion of correctly predicted samples. Then, we print the accuracy value.

accuracy = accuracy\_score(y\_test, y\_pred)

print("Accuracy:", accuracy)

**Output:**

*Accuracy: 1.0*

**Benefits of Ensemble Learning in Machine Learning**

Ensemble learning is a versatile approach that can be applied to machine learning model for: -

* **Reduction in Overfitting**: By aggregating predictions of multiple model's ensembles can reduce overfitting that individual complex models might exhibit.
* **Improved Generalization**: It generalizes better to unseen data by minimizing variance and bias.
* **Increased Accuracy**: Combining multiple models gives higher predictive accuracy.
* **Robustness to Noise**: It mitigates the effect of noisy or incorrect data points by averaging out predictions from diverse models.
* **Flexibility**: It can work with diverse models including decision trees, neural networks and support vector machines making them highly adaptable.
* **Bias-Variance Tradeoff**: Techniques like bagging reduce variance, while boosting reduces bias leading to better overall performance.

There are various ensemble learning techniques we can use as each one of them has their own pros and cons.

**Ensemble Learning Techniques**

| **Technique** | **Category** | **Description** |
| --- | --- | --- |
| **Random Forest** | Bagging | [Random forest](https://www.geeksforgeeks.org/random-forest-algorithm-in-machine-learning/) constructs multiple decision trees on bootstrapped subsets of the data and aggregates their predictions for final output, reducing overfitting and variance. |
| **Random Subspace Method** | Bagging | Trains models on random subsets of input features to enhance diversity and improve generalization while reducing overfitting. |
| **Gradient Boosting Machines (GBM)** | Boosting | [Gradient Boosting Machines](https://www.geeksforgeeks.org/ml-gradient-boosting/) sequentially builds decision trees, with each tree correcting errors of the previous ones, enhancing predictive accuracy iteratively. |
| **Extreme Gradient Boosting (XGBoost)** | Boosting | [XGBoost](https://www.geeksforgeeks.org/xgboost/) do optimizations like tree pruning, regularization, and parallel processing for robust and efficient predictive models. |
| **AdaBoost (Adaptive Boosting)** | Boosting | [AdaBoost](https://www.geeksforgeeks.org/implementing-the-adaboost-algorithm-from-scratch/) focuses on challenging examples by assigning weights to data points. Combines weak classifiers with weighted voting for final predictions. |
| **CatBoost** | Boosting | [CatBoost](https://www.geeksforgeeks.org/catboost-ml/) specialize in handling categorical features natively without extensive preprocessing with high predictive accuracy and automatic overfitting handling. |