Bagging, Boosting, and Voting are **Ensemble Learning** techniques used to improve the performance of machine learning models by combining multiple weak models (base learners).

1️⃣ Bagging (Bootstrap Aggregating)

- **Idea:** Train multiple models independently on different subsets of the data and average their predictions.
- **How it works:**
    1. Take multiple random samples **with replacement** (bootstrap sampling) from the dataset.
    2. Train a model (e.g., Decision Tree) on each sample.
    3. Combine predictions by averaging (for regression) or majority voting (for classification).
- **Popular Algorithm:** Random Forest (uses multiple Decision Trees).
- **Used when:** You want to **reduce variance** and avoid overfitting.

🔹 **Example Code (Bagging Classifier in Sklearn):**

```
from sklearn.ensemble import BaggingClassifier

from sklearn.tree import DecisionTreeClassifier

from sklearn.datasets import make_classification

from sklearn.model_selection import train_test_split

from sklearn.metrics import accuracy_score


# Create dataset

X, y = make_classification(n_samples=1000, n_features=20, random_state=42)

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)


# Bagging Classifier with Decision Trees

bagging = BaggingClassifier(base_estimator=DecisionTreeClassifier(), n_estimators=10,
random_state=42)

bagging.fit(X_train, y_train)


# Predict and evaluate

y_pred = bagging.predict(X_test)
```

print("Bagging Accuracy:", accuracy_score(y_test, y_pred))

2️⃣ Boosting (Sequential Learning)

- **Idea:** Train models sequentially, where each model corrects errors made by the previous one.
- **How it works:**
    1. Train a weak model (e.g., a small Decision Tree).
    2. Give more weight to incorrectly predicted samples.
    3. Train the next model on the updated dataset.
    4. Combine all models' predictions with weighted voting.
- **Popular Algorithms:** AdaBoost, Gradient Boosting, XGBoost, LightGBM, CatBoost.
- **Used when:** You want to **reduce bias** and make a strong model from weak learners.

🔹 **Example Code (AdaBoost in Sklearn):**

```
from sklearn.ensemble import AdaBoostClassifier
from sklearn.tree import DecisionTreeClassifier

# AdaBoost Classifier
adaboost = AdaBoostClassifier(base_estimator=DecisionTreeClassifier(max_depth=1), n_estimators=50, random_state=42)
adaboost.fit(X_train, y_train)

# Predict and evaluate
y_pred = adaboost.predict(X_test)
print("Boosting (AdaBoost) Accuracy:", accuracy_score(y_test, y_pred))
```

3️⃣ Voting (Majority Voting or Averaging)

- **Idea:** Combine multiple models (e.g., Logistic Regression, SVM, Decision Tree) and take their majority vote (classification) or average (regression).
- **How it works:**
    1. Train multiple different models on the same dataset.

2. For classification:
    - **Hard Voting:** Choose the class that gets the most votes.
    - **Soft Voting:** Average the probability scores and choose the highest.
3. For regression, take the average of predictions.

- **Used when:** You want to **combine diverse models** for better performance.

🔹 **Example Code (Voting Classifier in Sklearn):**

```
from sklearn.ensemble import VotingClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier

# Define base models
log_clf = LogisticRegression()
svm_clf = SVC(probability=True)
tree_clf = DecisionTreeClassifier()

# Voting Classifier (Hard Voting)
voting_clf = VotingClassifier(estimators=[('lr', log_clf), ('svm', svm_clf), ('tree', tree_clf)], voting='hard')
voting_clf.fit(X_train, y_train)

# Predict and evaluate
y_pred = voting_clf.predict(X_test)
print("Voting Accuracy:", accuracy_score(y_test, y_pred))
```

Summary Table

Technique	Approach	Works By	Goal	Example Model
Bagging	Parallel	Bootstrapping + Aggregation	Reduce variance	Random Forest
Boosting	Sequential	Correcting previous errors	Reduce bias	AdaBoost, XGBoost

Technique	Approach	Works By	Goal	Example Model
Voting	Parallel	Combining multiple models	Improve generalization	VotingClassifier

---

- **Key Differences:**
  - **Bagging** → Models are trained **independently** on different subsets.
  - **Boosting** → Models are trained **sequentially**, improving on previous mistakes.
  - **Voting** → Combines **completely different** models and takes a final decision.