Bagging is a technique that **reduces variance** and improves model performance by combining multiple models trained on different subsets of the data. Each model is trained independently, and their predictions are aggregated.

- For regression tasks, predictions are averaged;
- For classification, majority voting is used.

Steps of Bagging

1. Bootstrap Sampling:
   - Randomly sample subsets of the data with replacement to create multiple training datasets.
2. Train Models:
   - Train separate models (e.g., decision trees) on these datasets.
3. Aggregate Results:
   - Combine predictions (average for regression or majority vote for classification).

Example: Bagging with Scikit-learn

We will use the Iris dataset for a classification example. The Bagging classifier will be applied with Decision Trees as base learners.

```
# Import required libraries
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.ensemble import BaggingClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
```

```
# Load the Iris dataset
iris = load_iris()
X, y = iris.data, iris.target
```

```
# Split data into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
```

```
# Initialize base model (Decision Tree)
base_model = DecisionTreeClassifier(random_state=42)
```

```
# Create a BaggingClassifier with Decision Trees as base learners

bagging_model = BaggingClassifier(
    estimator=base_model,      # Updated argument name
    n_estimators=10,           # Number of trees. Each tree uses 80% of the training data.
    max_samples=0.8,           # Fraction of data sampled for each tree
    bootstrap=True,            # Enable bootstrap sampling
    random_state=42
)
```

In the context of bagging and bootstrap sampling, sampling with replacement means that when creating a new subset of data, each sample (or data point) from the original dataset can be selected multiple times. This is because, after selecting a sample, it is "put back" into the pool of data, making it available for selection again.

Key Points of Sampling with Replacement:

1. Duplication Possible: A single sample from the original dataset may appear multiple times in the new subset.
2. Random Selection: Each sample is chosen randomly, ensuring diversity among subsets.
3. Subset Size: The subset size is typically the same as the original dataset, but it contains duplicate samples.

Why Use Sampling with Replacement?

1. Increased Model Diversity: Each model sees a slightly different dataset, leading to diverse predictions and reducing the chance of overfitting.
2. Effective Use of Data: Even with small datasets, sampling with replacement ensures variability in training subsets.

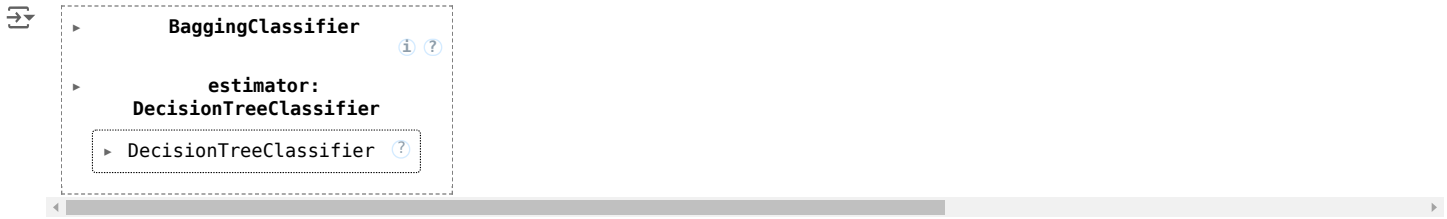
Example of Sampling with Replacement

Imagine a dataset: Original Dataset = [A, B, C, D] If we create a bootstrap sample of size 4 with replacement:

- Sample 1: [A, C, C, D]
- Sample 2: [B, B, C, A] Here, samples C and B appear multiple times due to replacement.

1. bootstrap=True, the training subsets for each base model (e.g., each decision tree) are created using bootstrap sampling.
2. bootstrap=False, subsets are created without replacement, meaning each data point appears only once in the subset (no duplicates).

```
# Train the BaggingClassifier
bagging_model.fit(X_train, y_train)
```




```
# Make predictions

y_pred = bagging_model.predict(X_test)
```

```
# Evaluate accuracy

accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy of Bagging Classifier: {accuracy:.2f}")
```

```
Accuracy of Bagging Classifier: 1.00
```

The BaggingClassifier trains multiple trees independently and aggregates their results.