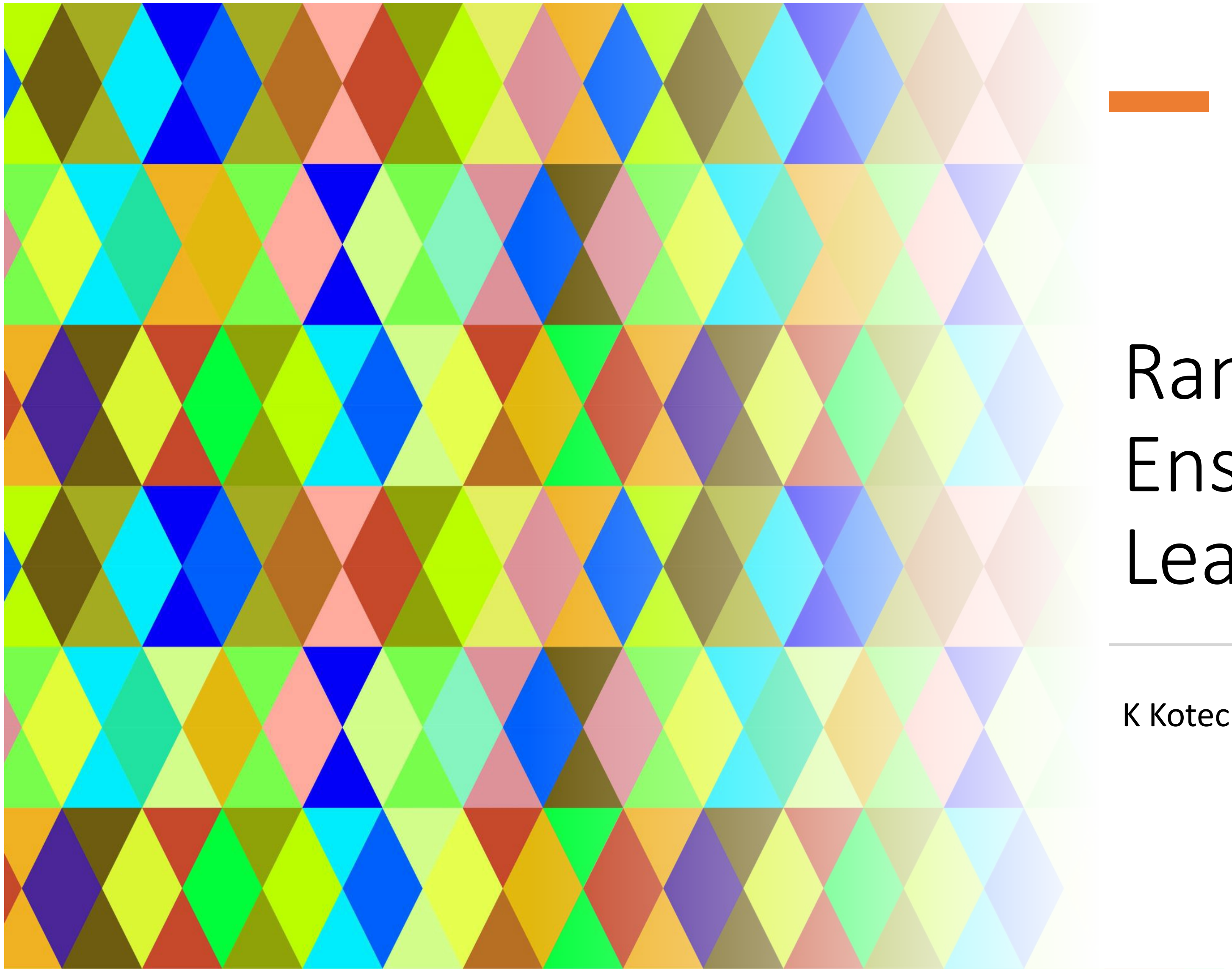
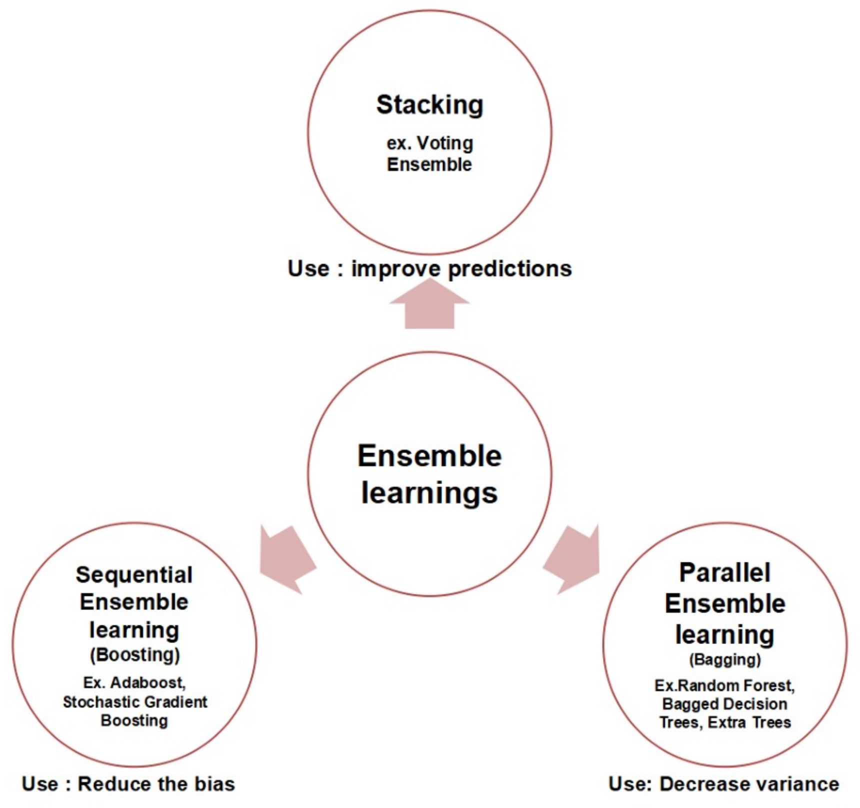


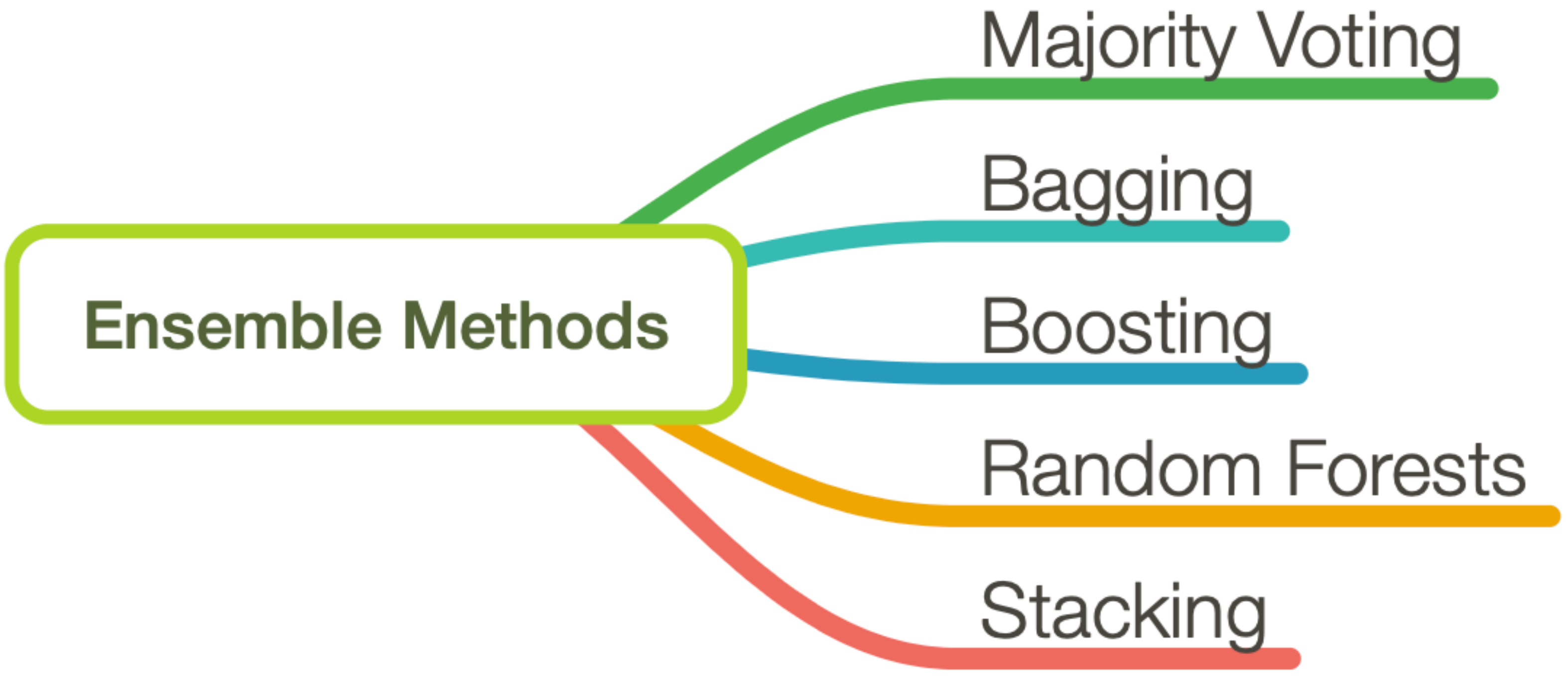
Random Forest Ensemble Learning

K Kotecha

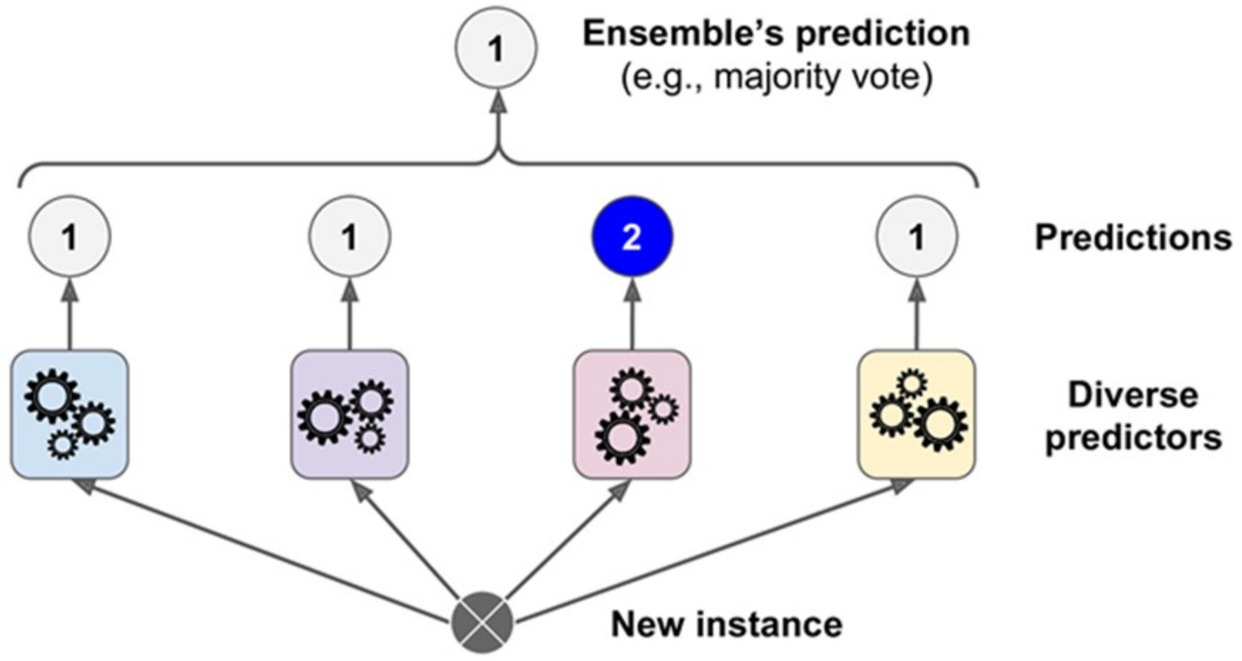
Material, slides and figs are taken from various internet sources for teaching purpose.

Ensemble learning is a technique that combines multiple machine learning algorithms to produce one optimal predictive model with **reduced variance** (using bagging), bias (using boosting) and improved predictions (using stacking).

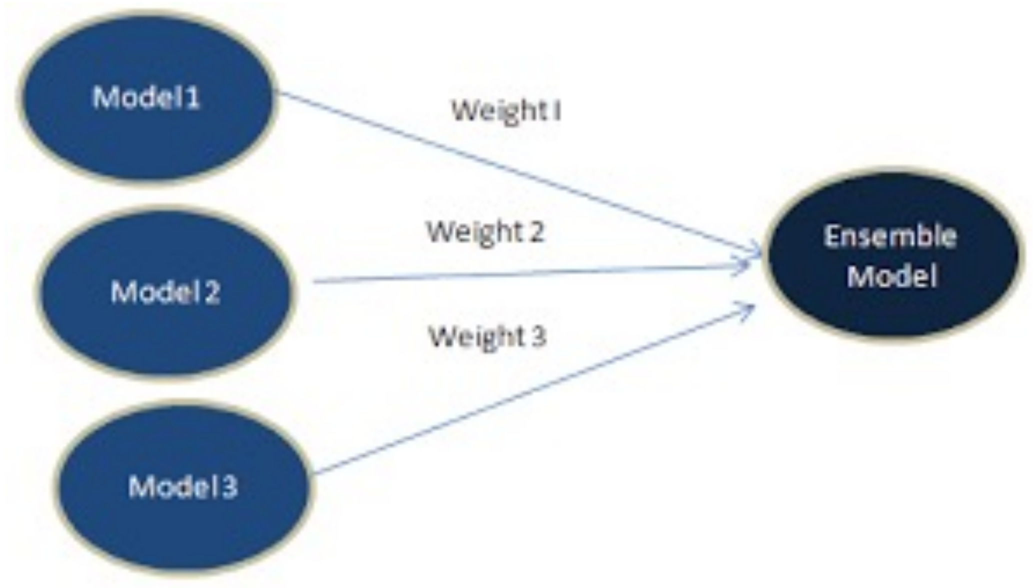

Stacking
ex. Voting Ensemble
Use : improve predictions
Ensemble learnings
Sequential Ensemble learning
(Boosting)
Ex. Adaboost, Stochastic Gradient Boosting
Use : Reduce the bias
Parallel Ensemble learning
(Bagging)
Ex.Random Forest, Bagged Decision Trees, Extra Trees
Use: Decrease variance


Ensemble Methods
Majority Voting
Bagging
Boosting
Random Forests
Stacking

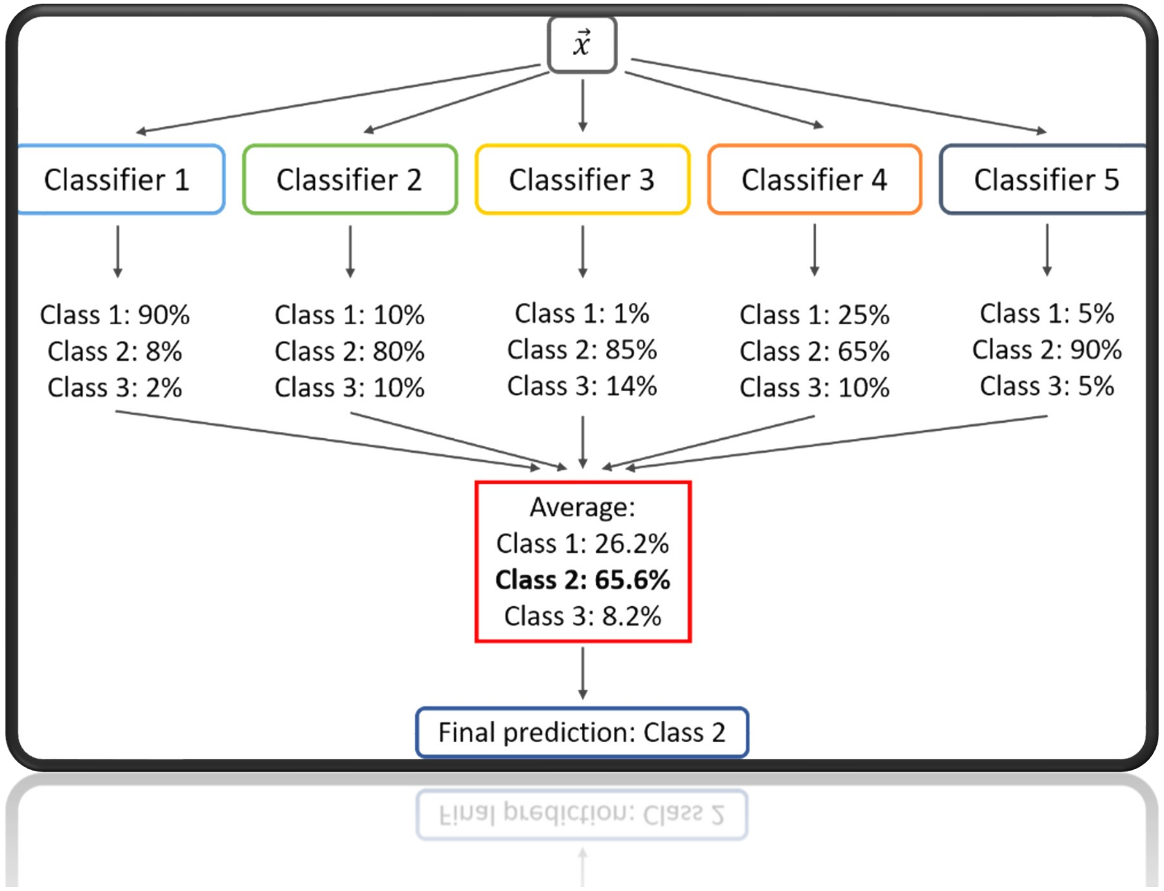
Max Voting


1
Ensemble's prediction
(e.g., majority vote)
1
1
2
1
Predictions
Diverse
predictors
New instance

Weighted Average

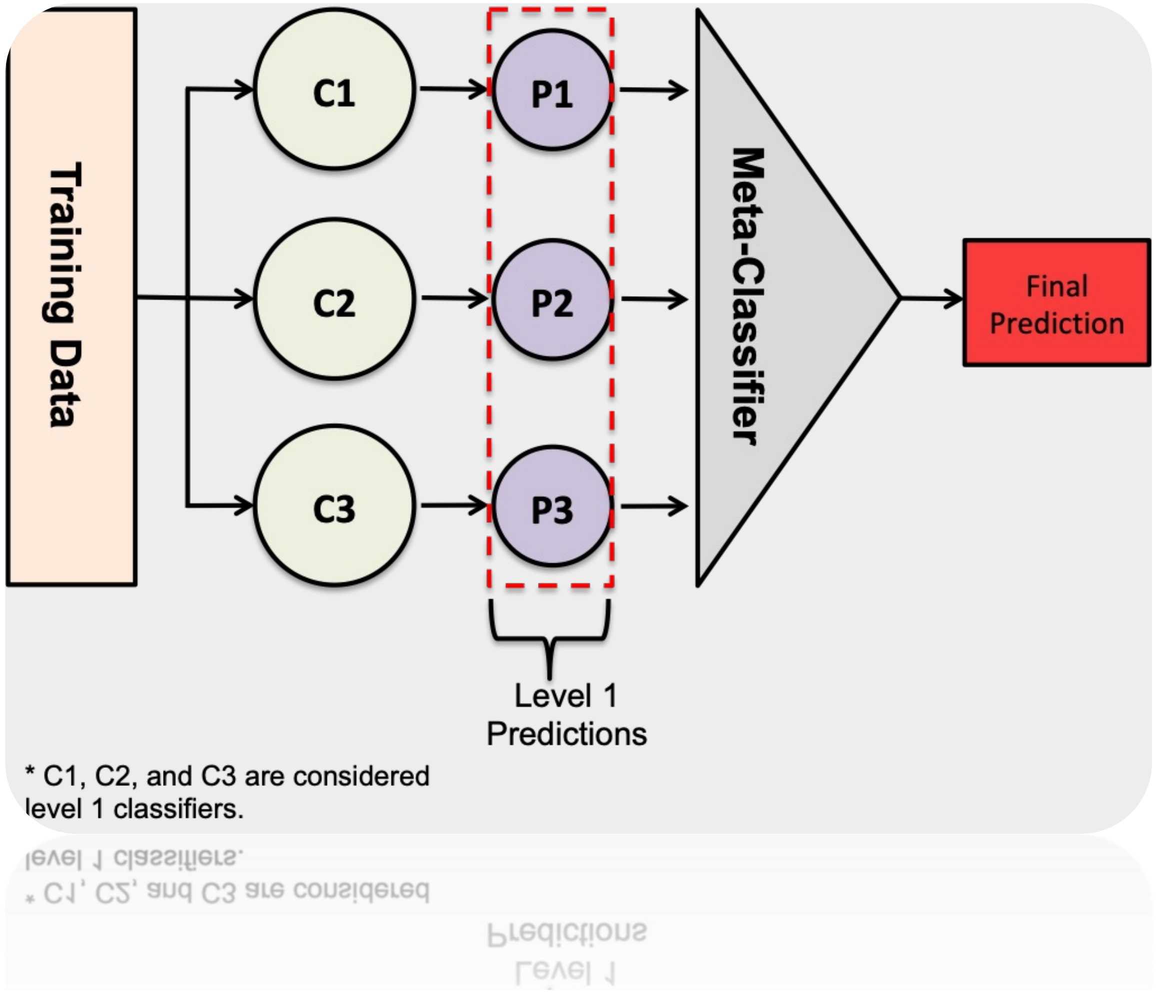


Averaging



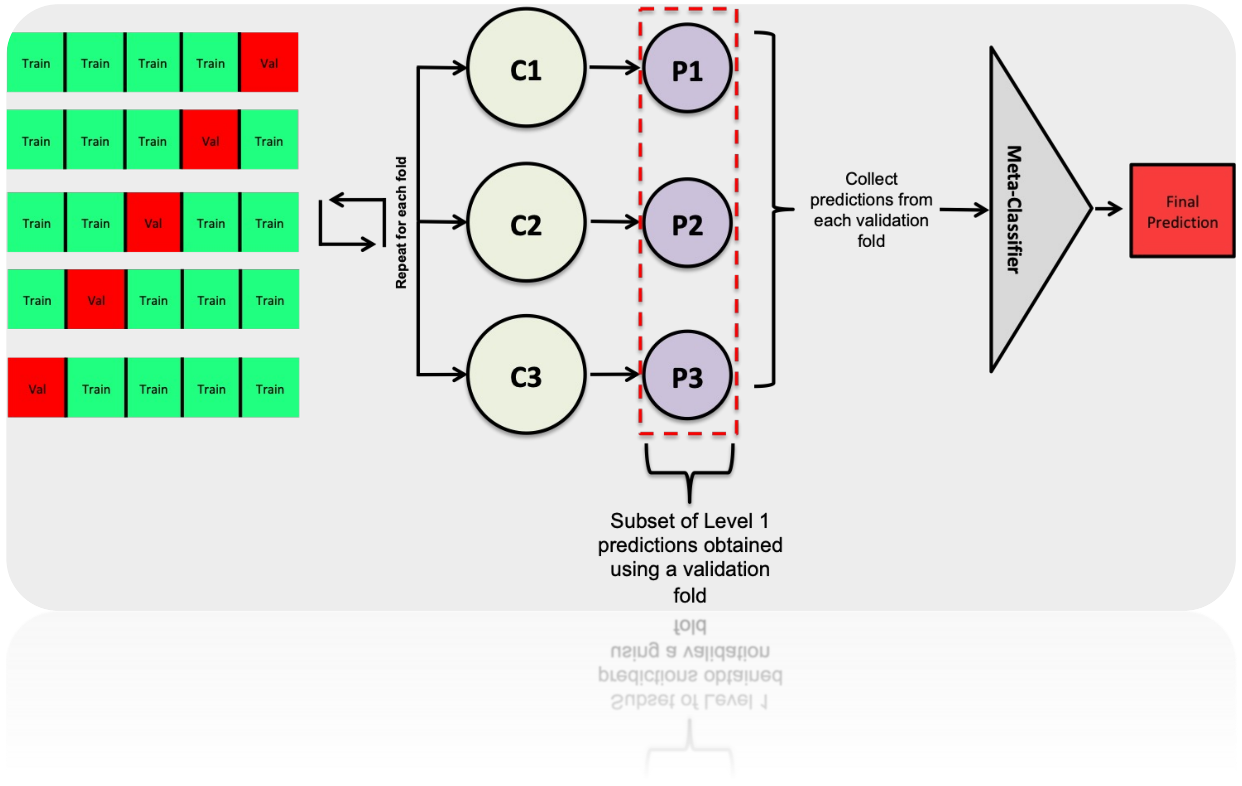
$\vec{x}$
Classifier 1
Classifier 2
Classifier 3
Classifier 4
Classifier 5
Class 1: 90%
Class 2: 8%
Class 3: 2%
Class 1: 10%
Class 2: 80%
Class 3: 10%
Class 1: 1%
Class 2: 85%
Class 3: 14%
Class 1: 25%
Class 2: 65%
Class 3: 10%
Class 1: 5%
Class 2: 90%
Class 3: 5%
Average:
Class 1: 26.2%
Class 2: 65.6%
Class 3: 8.2%
Final prediction: Class 2

Stacking



Split your training set in half. Use the first half of your training data to train the level one classifiers. Then use the trained level one classifiers to make predictions on the second half of the training data. These predictions should then be used to train meta-classifier.

Stacking

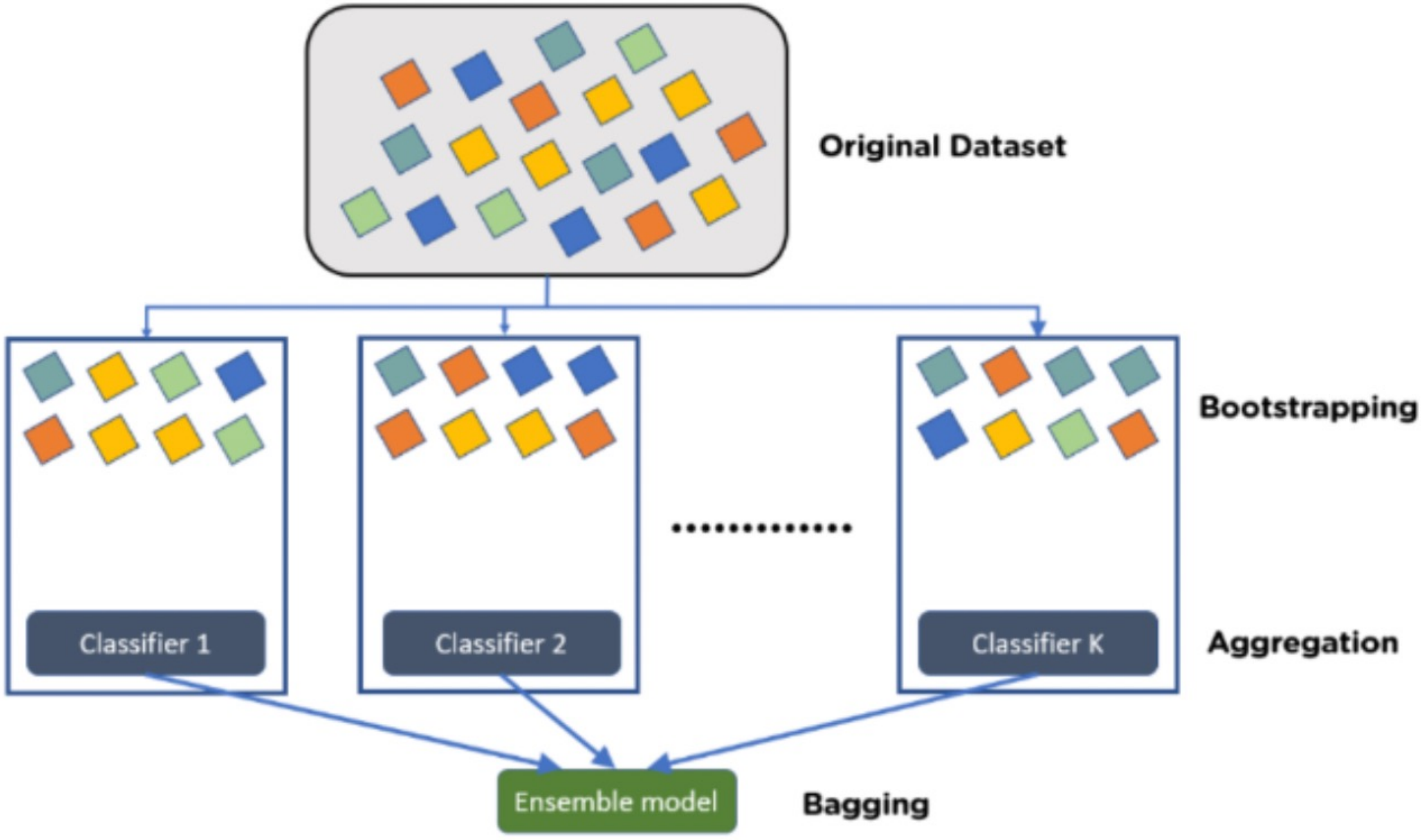


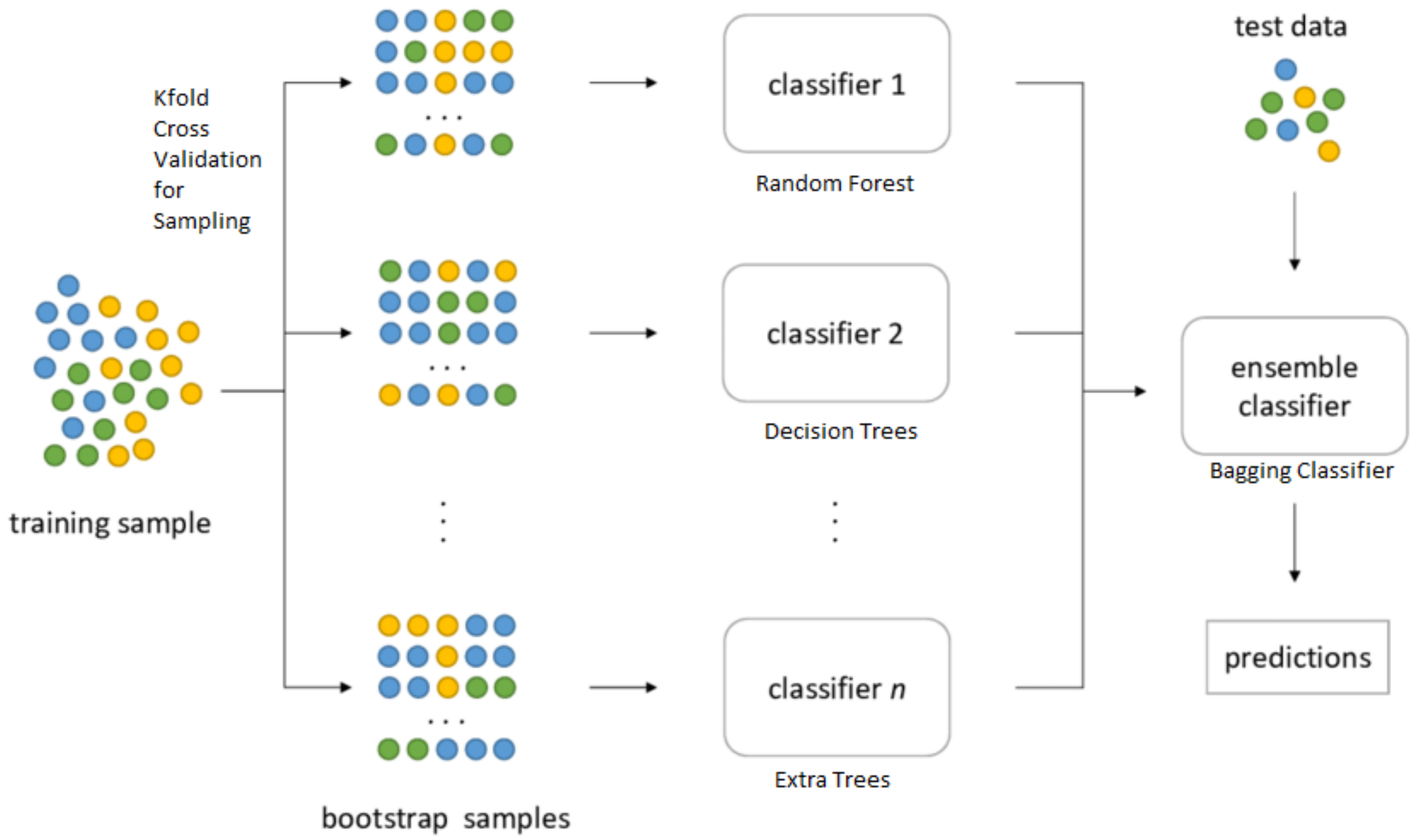
Train
Val
Repeat for each fold
C1
C2
C3
P1
P2
P3
Collect predictions from each validation fold
Meta-Classifier
Final Prediction
Subset of Level 1 predictions obtained using a validation fold

use k-fold cross validation to generate the level one predictions. Here, the training data is split into k-folds. Then the first k-1 folds are used to train the level one classifiers. The validation fold is then used to generate a subset of the level one predictions. The process is repeated for each unique group.

Algorithm 19.8 Stacking with K-fold Cross Validation

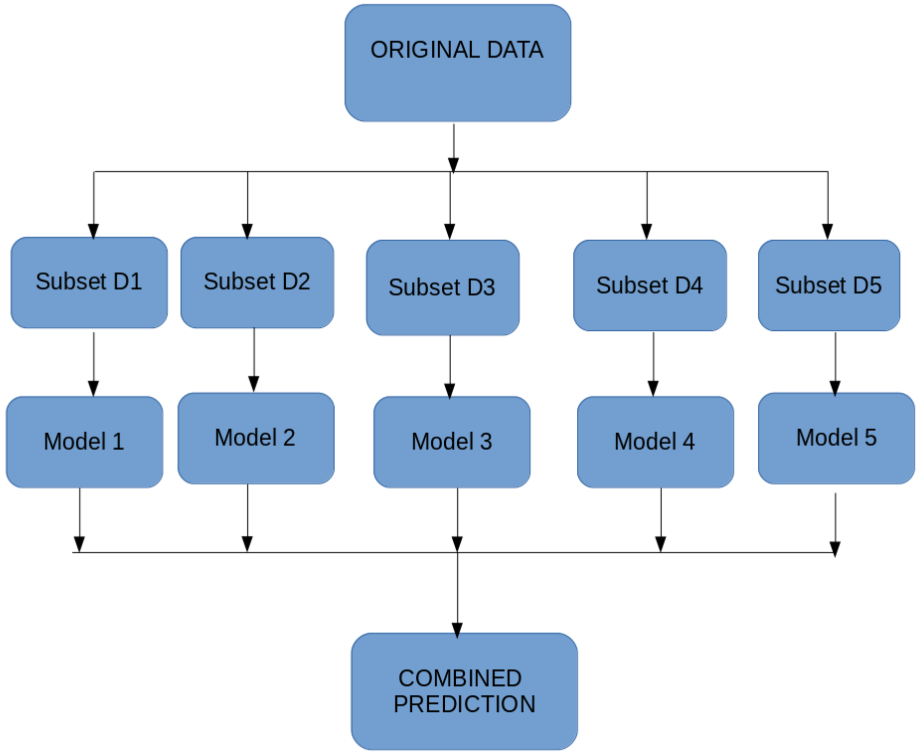
```
Input: Training data $\mathcal{D} = \{\mathbf{x}_i, y_i\}_{i=1}^{m}$ ($\mathbf{x}_i \in \mathbb{R}^n$, $y_i \in \mathcal{Y}$)
Output: An ensemble classifier $H$
 1: Step 1: Adopt cross validation approach in preparing a training set for second-level classifier
 2: Randomly split $\mathcal{D}$ into $K$ equal-size subsets: $\mathcal{D} = \{\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_K\}$
 3: for $k \leftarrow 1$ to $K$ do
 4:     Step 1.1: Learn first-level classifiers
 5:     for $t \leftarrow 1$ to $T$ do
 6:         Learn a classifier $h_{kt}$ from $\mathcal{D} \setminus \mathcal{D}_k$
 7:     end for
 8:     Step 1.2: Construct a training set for second-level classifier
 9:     for $\mathbf{x}_i \in \mathcal{D}_k$ do
10:         Get a record $\{\mathbf{x}'_i, y_i\}$, where $\mathbf{x}'_i = \{h_{k1}(\mathbf{x}_i), h_{k2}(\mathbf{x}_i), \ldots, h_{kT}(\mathbf{x}_i)\}$
11:     end for
12: end for
13: Step 2: Learn a second-level classifier
14: Learn a new classifier $h'$ from the collection of $\{\mathbf{x}'_i, y_i\}$
15: Step 3: Re-learn first-level classifiers
16: for $t \leftarrow 1$ to $T$ do
17:     Learn a classifier $h_t$ based on $\mathcal{D}$
18: end for
19: return $H(\mathbf{x}) = h'(h_1(\mathbf{x}), h_2(\mathbf{x}), \ldots, h_T(\mathbf{x}))$
```

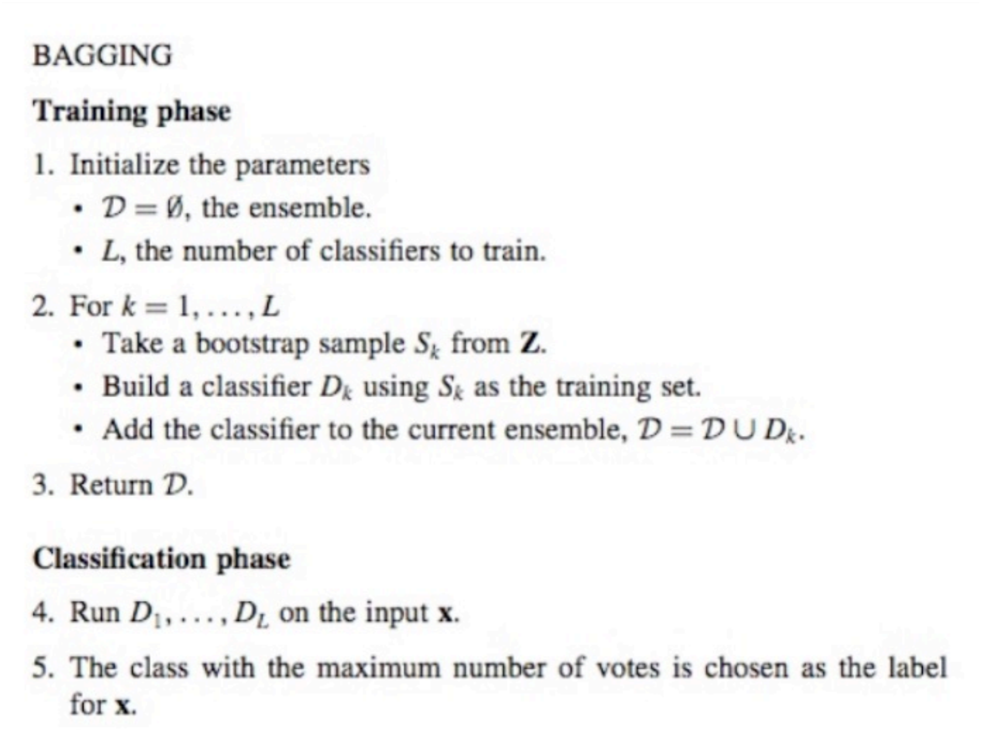

Original Dataset
Bootstrapping
Classifier 1
Classifier 2
Classifier K
Aggregation
Ensemble model
Bagging



Bagging Classifier Process Flow

Bagging - Bootstrap Aggregation





BAGGING

Training phase

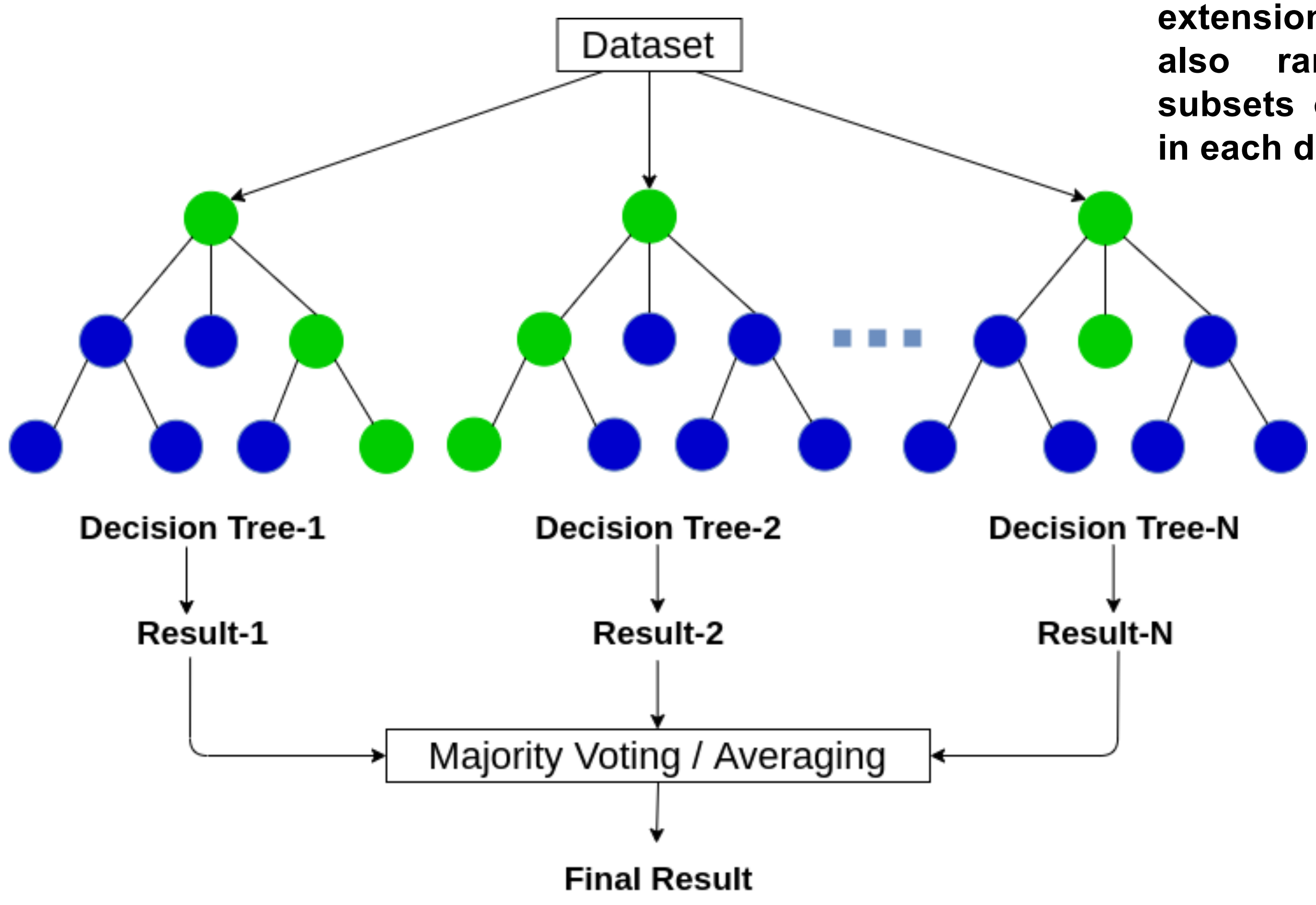
```
1. Initialize the parameters
    • $\mathcal{D} = \emptyset$, the ensemble.
    • $L$, the number of classifiers to train.
2. For $k = 1, \dots, L$
    • Take a bootstrap sample $S_k$ from $\mathbf{Z}$.
    • Build a classifier $D_k$ using $S_k$ as the training set.
    • Add the classifier to the current ensemble, $\mathcal{D} = \mathcal{D} \cup D_k$.
3. Return $\mathcal{D}$.
```

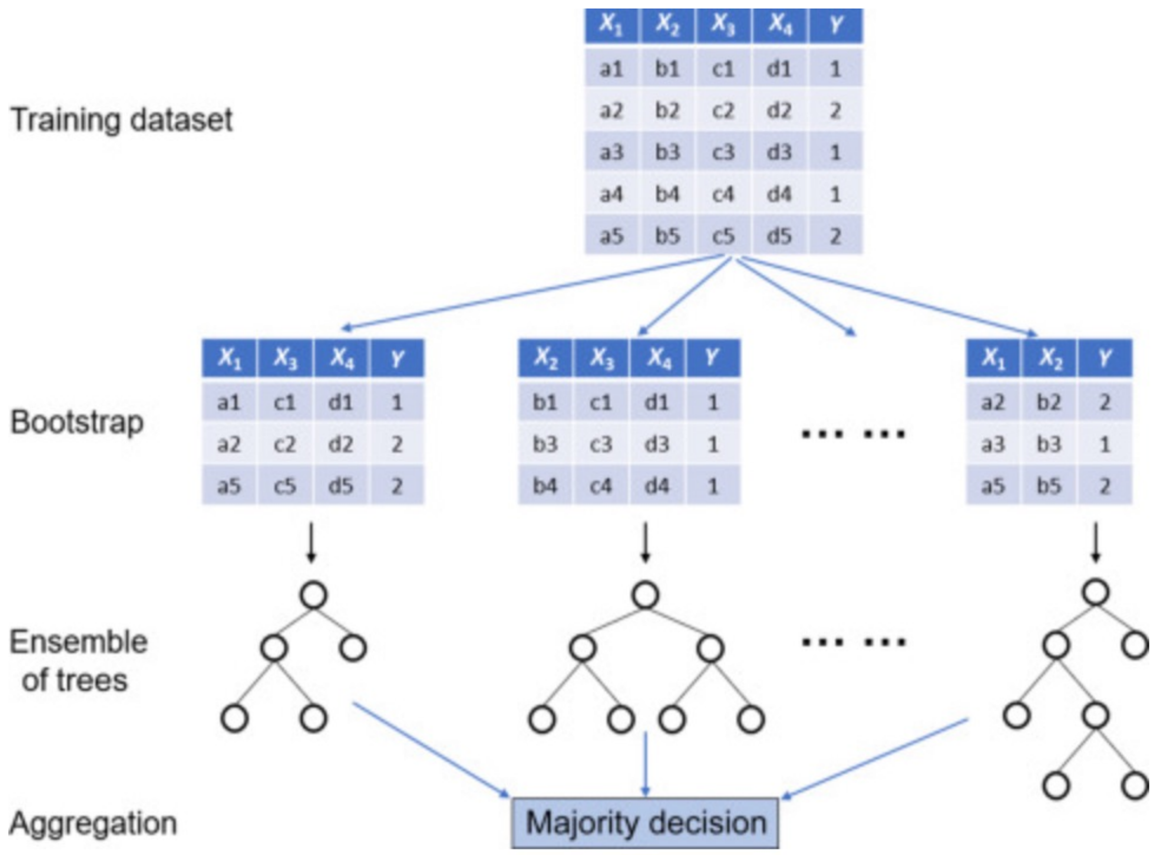
Classification phase

```
4. Run $D_1, \dots, D_L$ on the input $\mathbf{x}$.
5. The class with the maximum number of votes is chosen as the label
   for $\mathbf{x}$.
```

Random Forest

Random forest is an extension of bagging that also randomly selects subsets of features used in each data sample



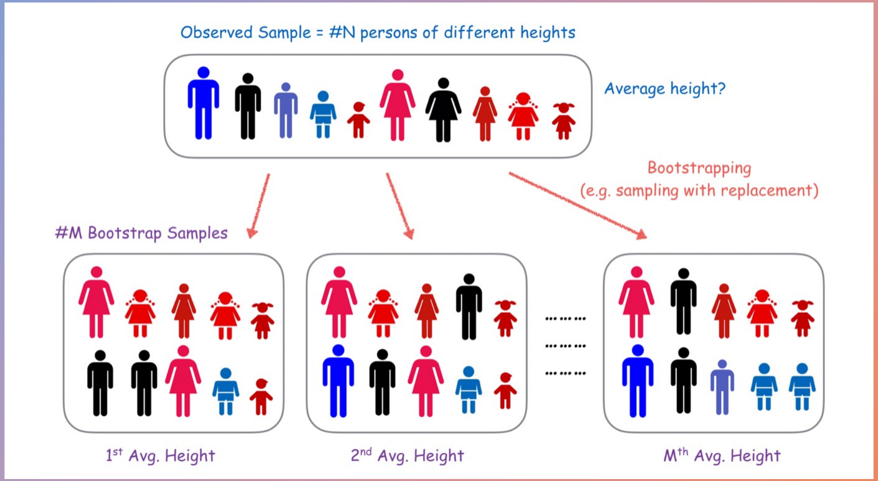


X_1	X_2	X_3	X_4	Y
a1	b1	c1	d1	1
a2	b2	c2	d2	2
a3	b3	c3	d3	1
a4	b4	c4	d4	1
a5	b5	c5	d5	2

X_1	X_3	X_4	Y
a1	c1	d1	1
a2	c2	d2	2
a5	c5	d5	2

X_2	X_3	X_4	Y
b1	c1	d1	1
b3	c3	d3	1
b4	c4	d4	1

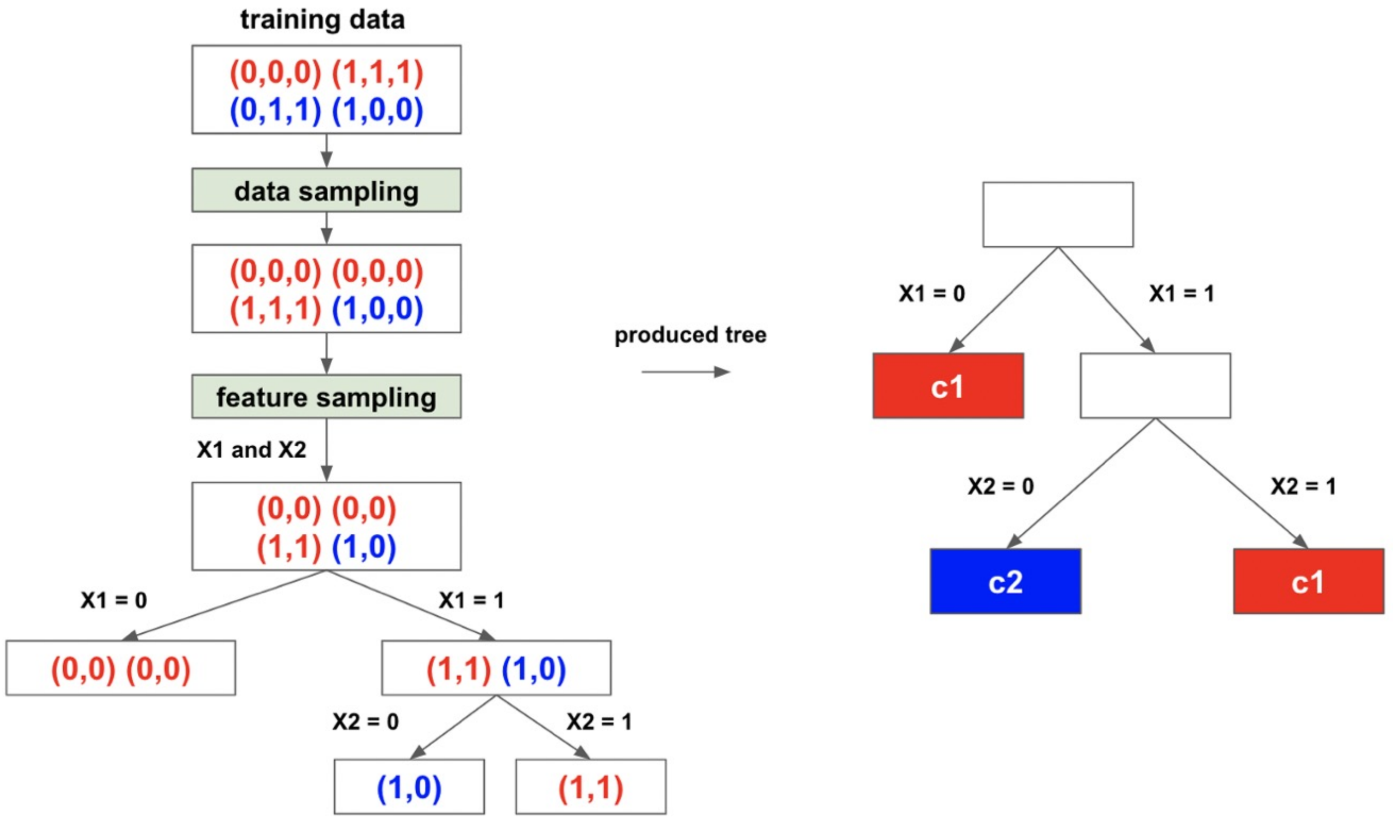
X_1	X_2	Y
a2	b2	2
a3	b3	1
a5	b5	2



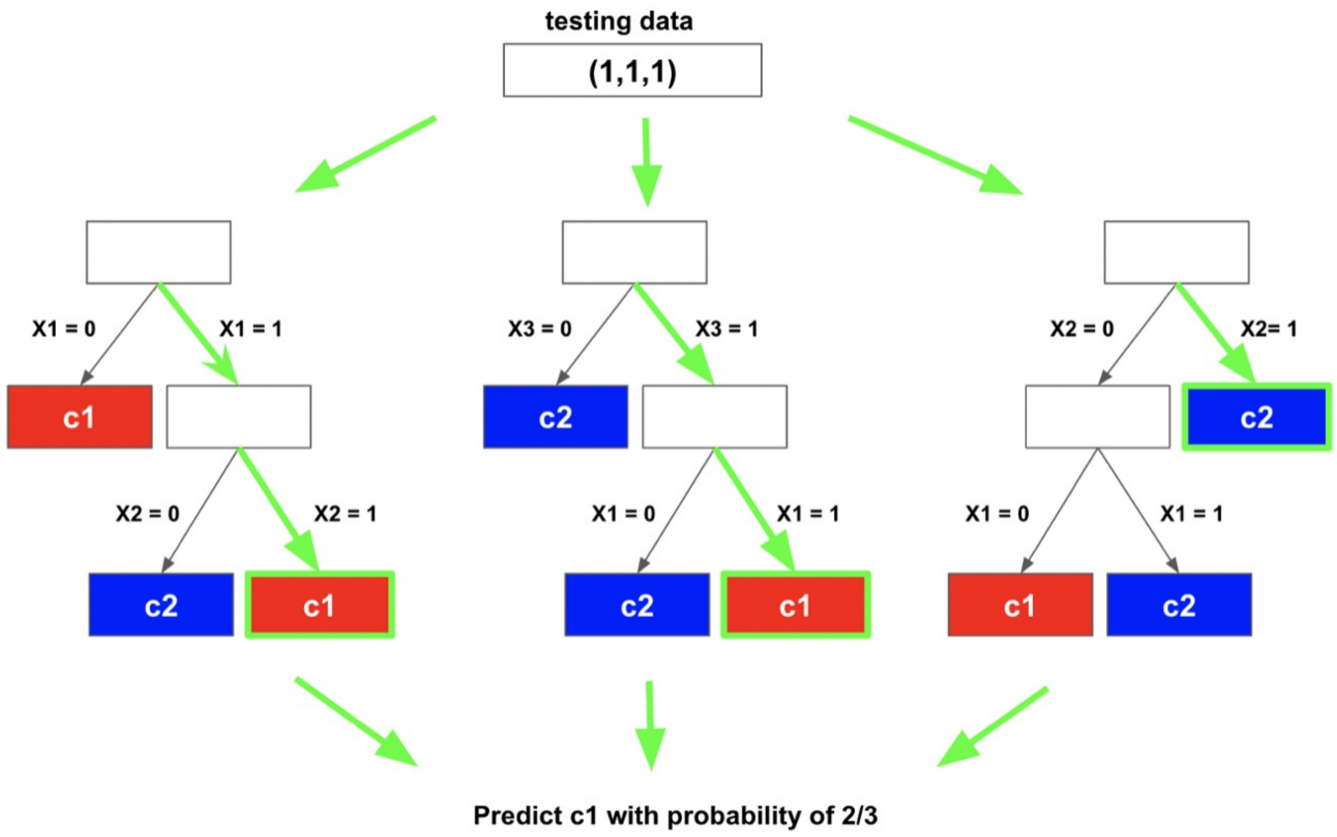
Bootstrap

- The bootstrap is a powerful statistical method for estimating a quantity from a data sample.
- This is easiest to understand if the quantity is a descriptive statistic such as a mean or a standard deviation.
- Let’s assume we have a sample of 100 values (x) and we’d like to get an estimate of the mean of the sample.
- We can calculate the mean directly from the sample as:
- mean(x) = 1/100 * sum(x)
- We know that our sample is small and that our mean has error in it.
- We can improve the estimate of our mean using the bootstrap procedure:
- Create many (e.g. 1000) random sub-samples of our dataset with replacement
- (meaning we can select the same value multiple times).
- Calculate the mean of each sub-sample.
- Calculate the average of all of our collected means and use that as our estimated mean for the data.

X1	X2	X3	Class
0	0	0	c1
1	1	1	c1
0	1	1	c2
1	0	0	c2



training data
(0,0,0) (1,1,1)
(0,1,1) (1,0,0)
data sampling
(0,0,0) (0,0,0)
(1,1,1) (1,0,0)
feature sampling
X1 and X2
(0,0) (0,0)
(1,1) (1,0)
X1 = 0
X1 = 1
(0,0) (0,0)
(1,1) (1,0)
X2 = 0
X2 = 1
(1,0)
(1,1)
produced tree
X1 = 0
X1 = 1
c1
X2 = 0
X2 = 1
c2
c1


testing data
(1,1,1)
X1 = 0
X1 = 1
c1
X2 = 0
X2 = 1
c2
c1
X3 = 0
X3 = 1
c2
X1 = 0
X1 = 1
c2
c1
X2 = 0
X2= 1
c2
X1 = 0
X1 = 1
c1
c2
Predict c1 with probability of 2/3

- **Real life applications of Random Forests**
    - Fraud detection for bank accounts, credit card
    - Detect and predict the drug sensitivity of a medicine
    - Identify a patient’s disease by analyzing their medical records
    - Predict estimated loss or profit while purchasing a particular stock