Random forests

STAT 4710

November 2, 2023

Where we are

Unit 1: R for data mining

Unit 2: Prediction fundamentals

Unit 3: Regression-based methods

Unit 4: Tree-based methods

Unit 5: Deep learning

Lecture 1: Growing decision trees

Lecture 2: Tree pruning and bagging

Lecture 3: Random forests

Lecture 4: Boosting

Lecture 5: Unit review and quiz in class

Recall: Bagging

Original training data

Obs ID	X	Y
1	X_1	Y_1
2	X_2	Y_2
3	X_3	Y_3
4	X_4	Y_4
5	X_5	Y_5

Bootstrap sample 1

Obs ID	X	Y
5	X_5	Y_5
3	X_3	Y_3
2	X_2	Y_2
3	X_3	Y_3
1	X_1	Y_1

⋮

Bootstrap sample B

Obs ID	X	Y
4	X_4	Y_4
1	X_1	Y_1
1	X_1	Y_1
5	X_5	Y_5
4	X_4	Y_4

Recall: Bagging

Original training data

Obs ID	*X*	*Y*
1	X_1	Y_1
2	X_2	Y_2
3	X_3	Y_3
4	X_4	Y_4
5	X_5	Y_5

Bootstrap sample 1

Obs ID	*X*	*Y*
5	X_5	Y_5
3	X_3	Y_3
2	X_2	Y_2
3	X_3	Y_3
1	X_1	Y_1

→ T^{*1}

⋮

→ T^{*b}

⋮

Bootstrap sample *B*

Obs ID	*X*	*Y*
4	X_4	Y_4
1	X_1	Y_1
1	X_1	Y_1
5	X_5	Y_5
4	X_4	Y_4

→ T^{*B}

Recall: Bagging

Original training data

Obs ID	*X*	*Y*
1	X_1	Y_1
2	X_2	Y_2
3	X_3	Y_3
4	X_4	Y_4
5	X_5	Y_5

Bootstrap sample 1

Obs ID	*X*	*Y*
5	X_5	Y_5
3	X_3	Y_3
2	X_2	Y_2
3	X_3	Y_3
1	X_1	Y_1

$\rightarrow T^{*1}$

⋮

$\rightarrow T^{*b}$

⋮

Bootstrap sample *B*

Obs ID	*X*	*Y*
4	X_4	Y_4
1	X_1	Y_1
1	X_1	Y_1
5	X_5	Y_5
4	X_4	Y_4

$\rightarrow T^{*B}$

Regression:

$$\hat{f}(X) = \frac{1}{B}\sum_{b=1}^{B} T^{*b}(X)$$

Classification:

$$\hat{f}(X) = \text{mode}(\{T^{*b}(X)\}_{b=1}^{B})$$

Variance reduction of bagging

Variance reduction of bagging

The bagging prediction is defined by $\hat{f}(X) = \frac{1}{B}\sum_{b=1}^{B} T^{*b}(X)$.

Variance reduction of bagging

The bagging prediction is defined by $\hat{f}(X) = \frac{1}{B}\sum_{b=1}^{B} T^{*b}(X)$.

Suppose Corr$[T^{*b_1}(X), T^{*b_2}(X)] = \rho \in [0,1]$. Then, we can derive that

$$\text{Var}[\hat{f}(X)] \approx \left(\frac{1}{B} + \frac{B-1}{B}\rho\right)\text{Var}[T(X)] \approx \rho \cdot \text{Var}[T(X)],$$

where $T(X)$ is a single decision tree.

Variance reduction of bagging

The bagging prediction is defined by $\hat{f}(X) = \frac{1}{B}\sum_{b=1}^{B} T^{*b}(X)$.

Suppose $\text{Corr}[T^{*b_1}(X), T^{*b_2}(X)] = \rho \in [0,1]$. Then, we can derive that

$$\text{Var}[\hat{f}(X)] \approx \left(\frac{1}{B} + \frac{B-1}{B}\rho\right)\text{Var}[T(X)] \approx \rho \cdot \text{Var}[T(X)],$$

where $T(X)$ is a single decision tree.

- The variance is reduced by a factor of $\rho = \text{Corr}[T^{*b_1}(X), T^{*b_2}(X)]$, so the less correlated the bootstrapped trees prediction are, the better.

Variance reduction of bagging

The bagging prediction is defined by $\hat{f}(X) = \dfrac{1}{B}\sum_{b=1}^{B} T^{*b}(X)$.

Suppose Corr$[T^{*b_1}(X), T^{*b_2}(X)] = \rho \in [0,1]$. Then, we can derive that

$$\text{Var}[\hat{f}(X)] \approx \left(\frac{1}{B} + \frac{B-1}{B}\rho\right)\text{Var}[T(X)] \approx \rho \cdot \text{Var}[T(X)],$$

where $T(X)$ is a single decision tree.

- The variance is reduced by a factor of $\rho =$ Corr$[T^{*b_1}(X), T^{*b_2}(X)]$, so the less correlated the bootstrapped trees prediction are, the better.

- As long as B is large enough, the variance reduction is about the same.

Random forests: More variance reduction

Random forests: More variance reduction

Random forests are the same as bagging, but with one key modification:

Random forests: More variance reduction

Random forests are the same as bagging, but with one key modification:

At each split point of each tree:

- Randomly sample a subset of $m \leq p$ features
- Split on the best feature *among this subset*

Random forests: More variance reduction

Random forests are the same as bagging, but with one key modification:

At each split point of each tree:

- Randomly sample a subset of $m \leq p$ features
- Split on the best feature *among this subset*

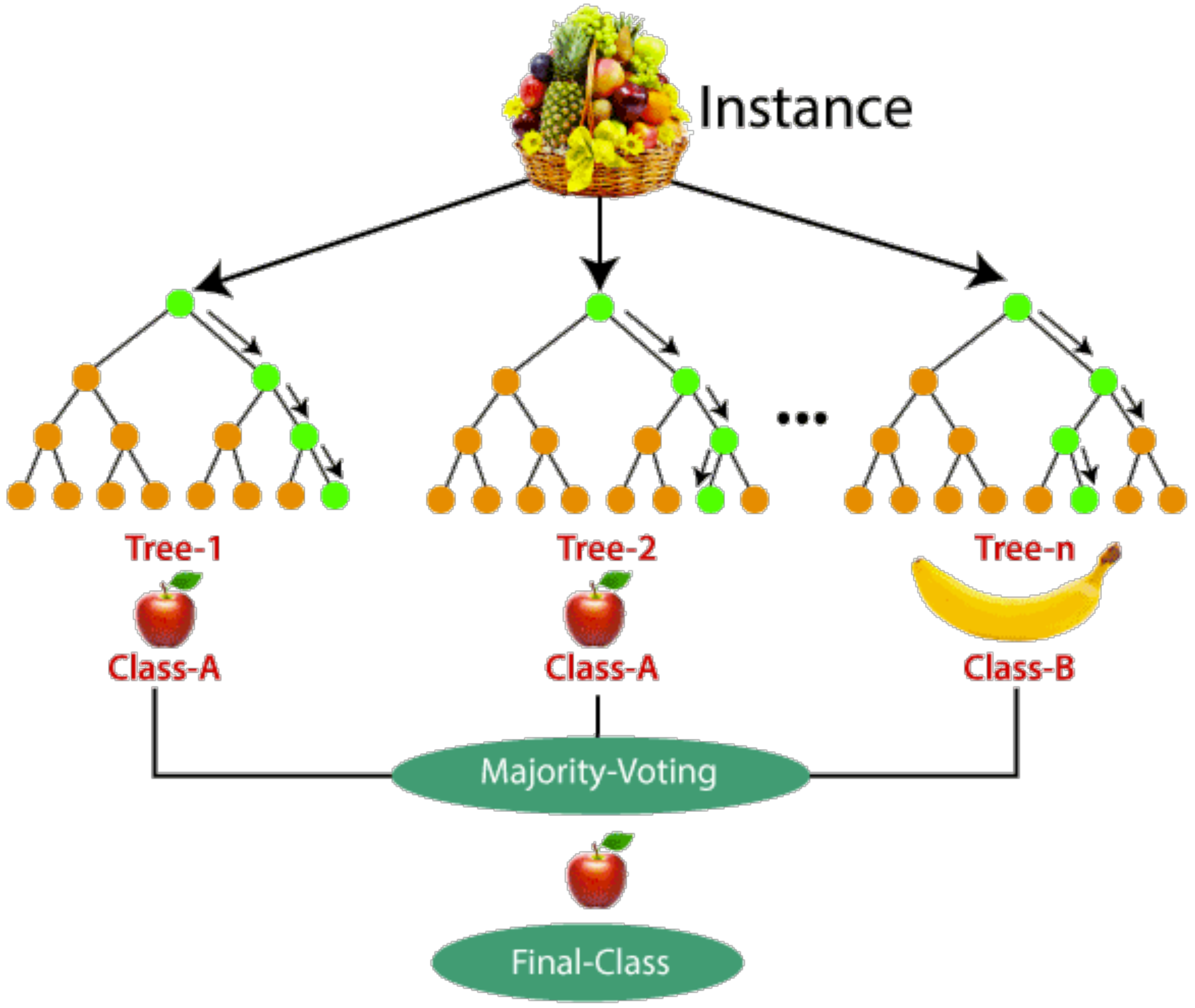

Image source: https://www.section.io/engineering-education/introduction-to-random-forest-in-machine-learning/

Random forests: More variance reduction

Random forests are the same as bagging, but with one key modification:

At each split point of each tree:

- Randomly sample a subset of $m \leq p$ features
- Split on the best feature *among this subset*

Redraw sample of features at every split point, regardless of samples at previous split points.

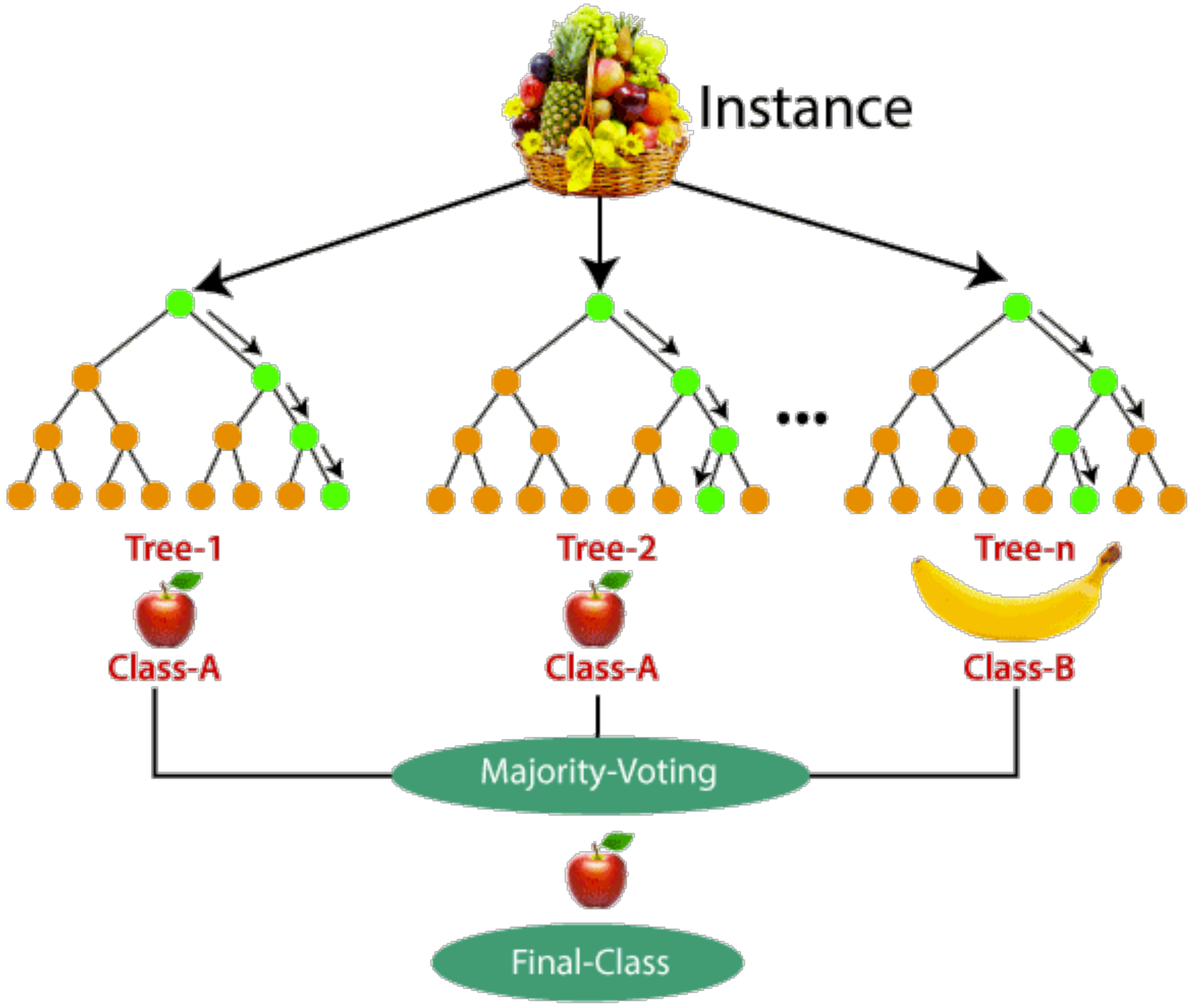

Image source: https://www.section.io/engineering-education/introduction-to-random-forest-in-machine-learning/

Random forests: More variance reduction

Random forests are the same as bagging, but with one key modification:

At each split point of each tree:

- Randomly sample a subset of $m \leq p$ features
- Split on the best feature *among this subset*

Redraw sample of features at every split point, regardless of samples at previous split points.

Intuition: Sampling features at each split decorrelates the trees, reducing variance and therefore boosting prediction performance.

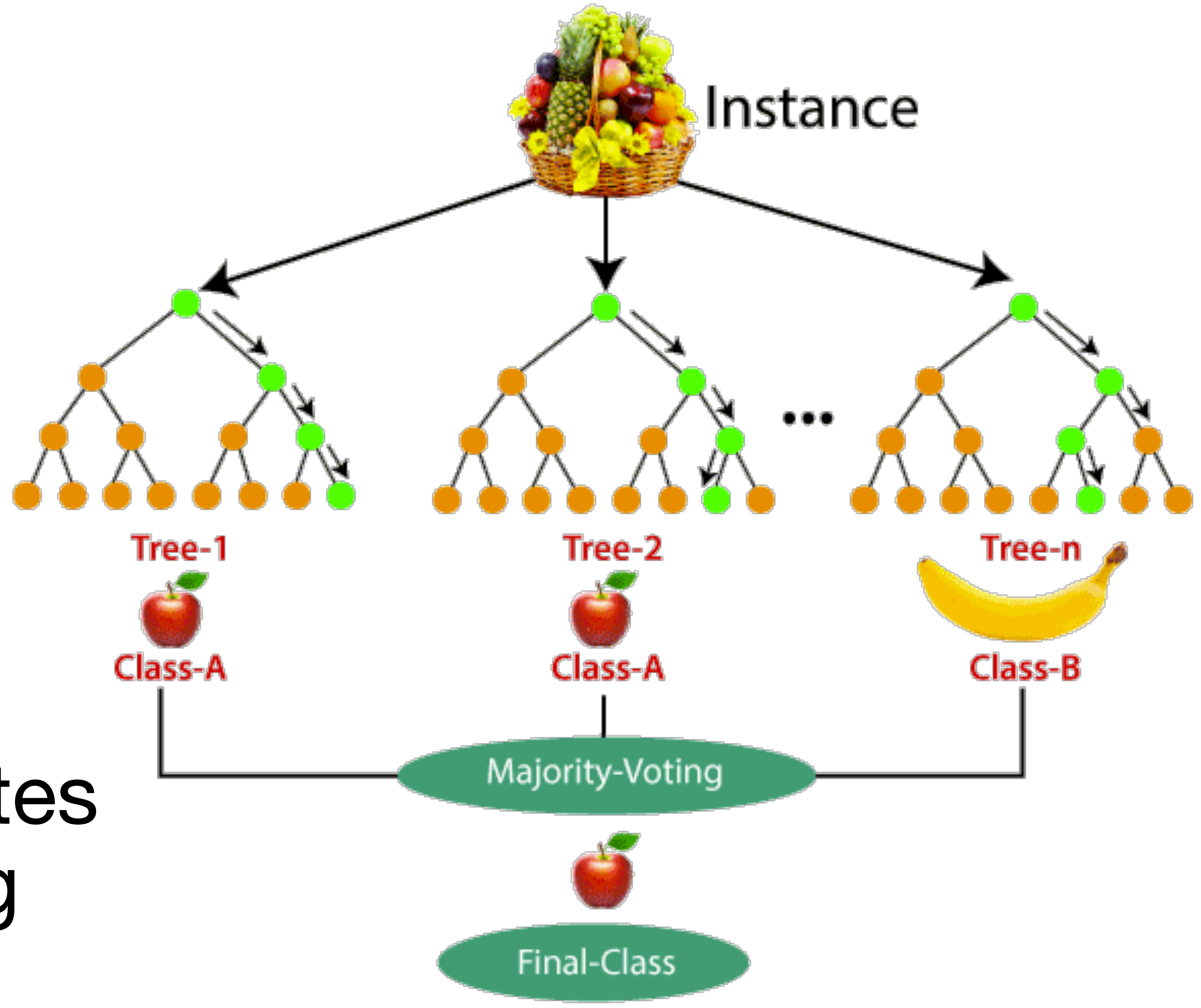

Image source: https://www.section.io/engineering-education/introduction-to-random-forest-in-machine-learning/

Random forests: More variance reduction

Random forests are the same as bagging, but with one key modification:

At each split point of each tree:

- Randomly sample a subset of $m \leq p$ features
- Split on the best feature *among this subset*

Redraw sample of features at every split point, regardless of samples at previous split points.

Intuition: Sampling features at each split decorrelates the trees, reducing variance and therefore boosting prediction performance.

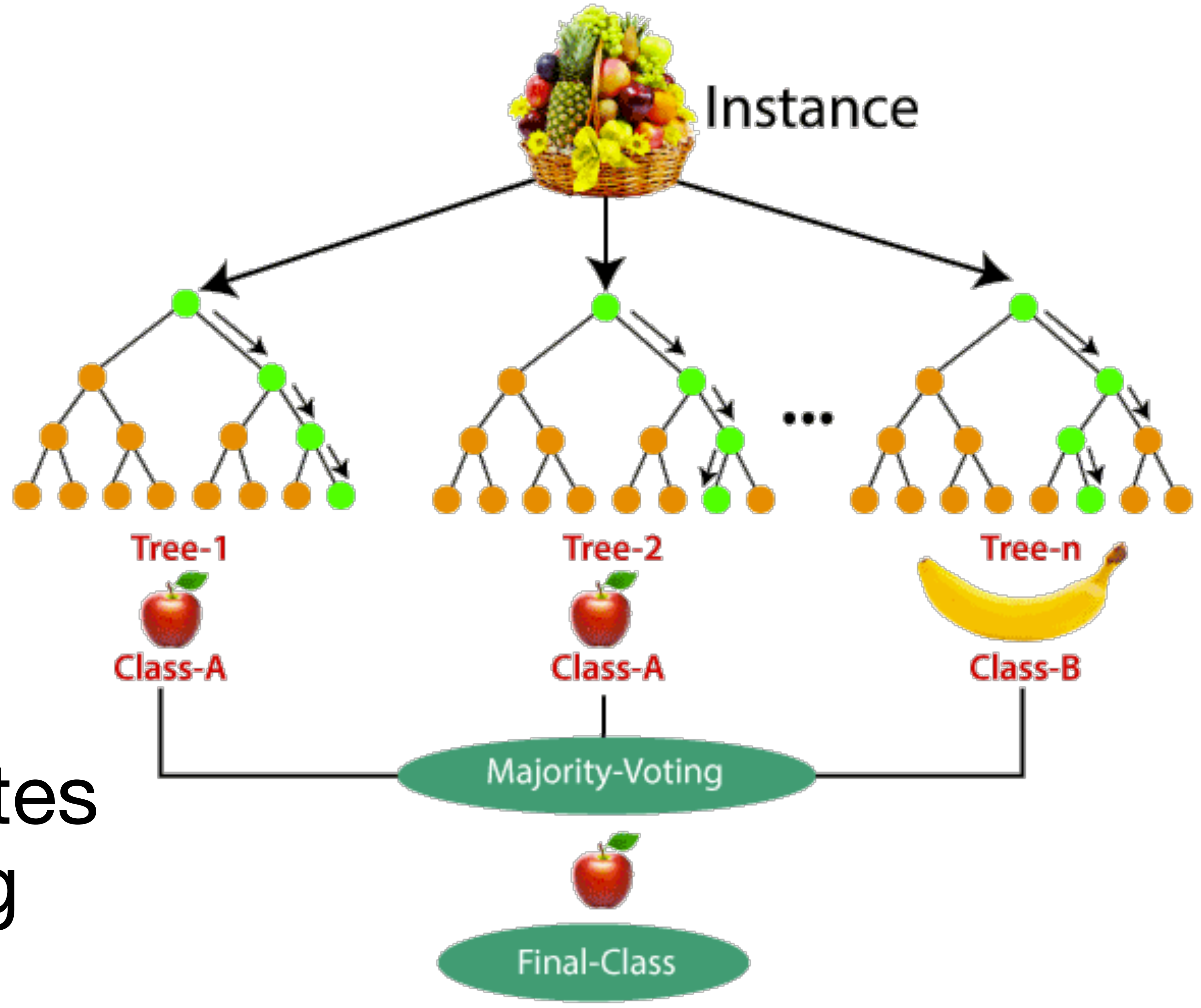

Image source: https://www.section.io/engineering-education/introduction-to-random-forest-in-machine-learning/

Note that setting $m = p$ recovers bagging.

Random forests

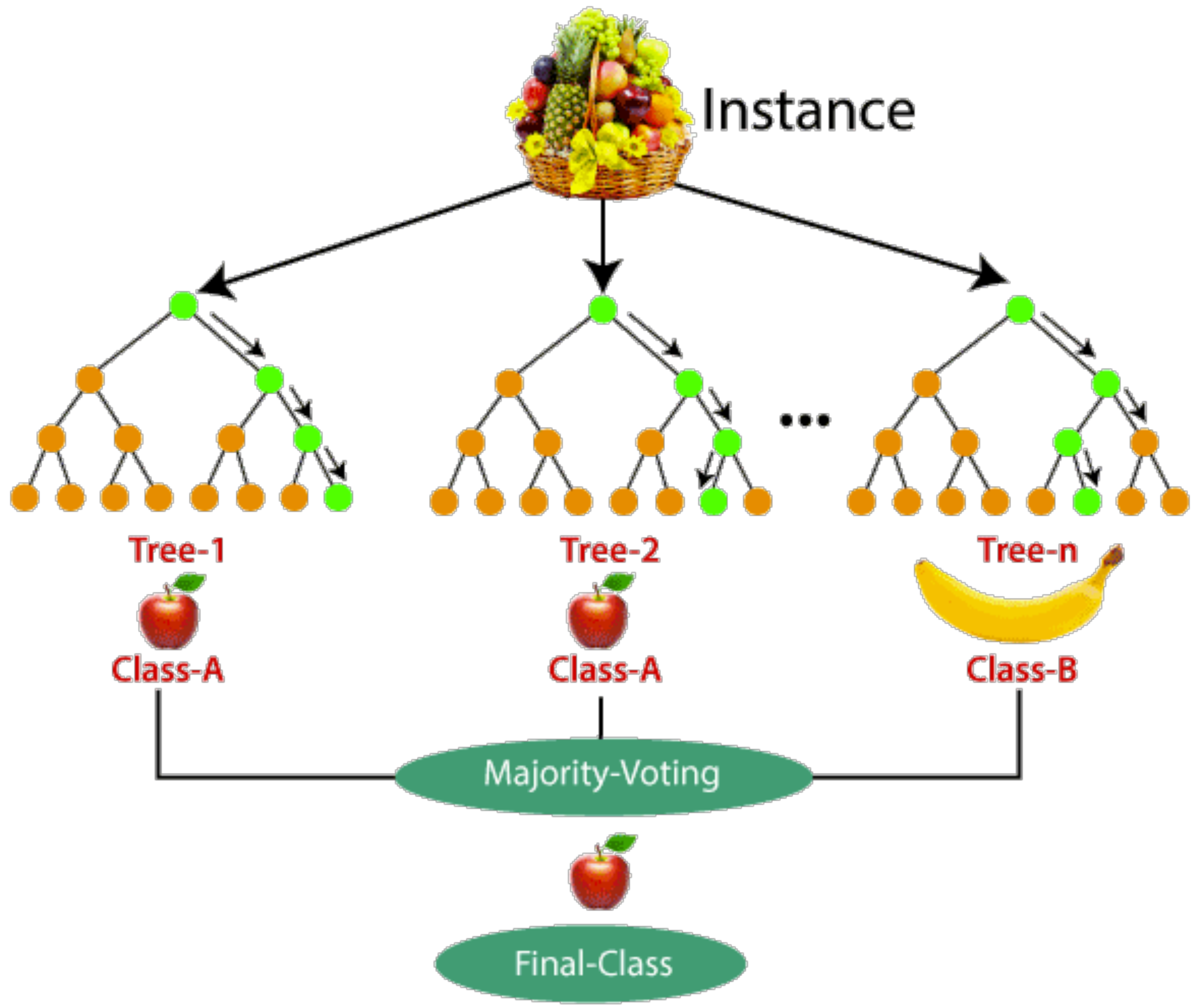

Image source: https://www.section.io/engineering-education/introduction-to-random-forest-in-machine-learning/

Random forests

Parameters:

- B: number of bootstrap samples
- m: number of variables to sample at each split
- criterion to stop splitting, like max number of nodes and/or min samples per node

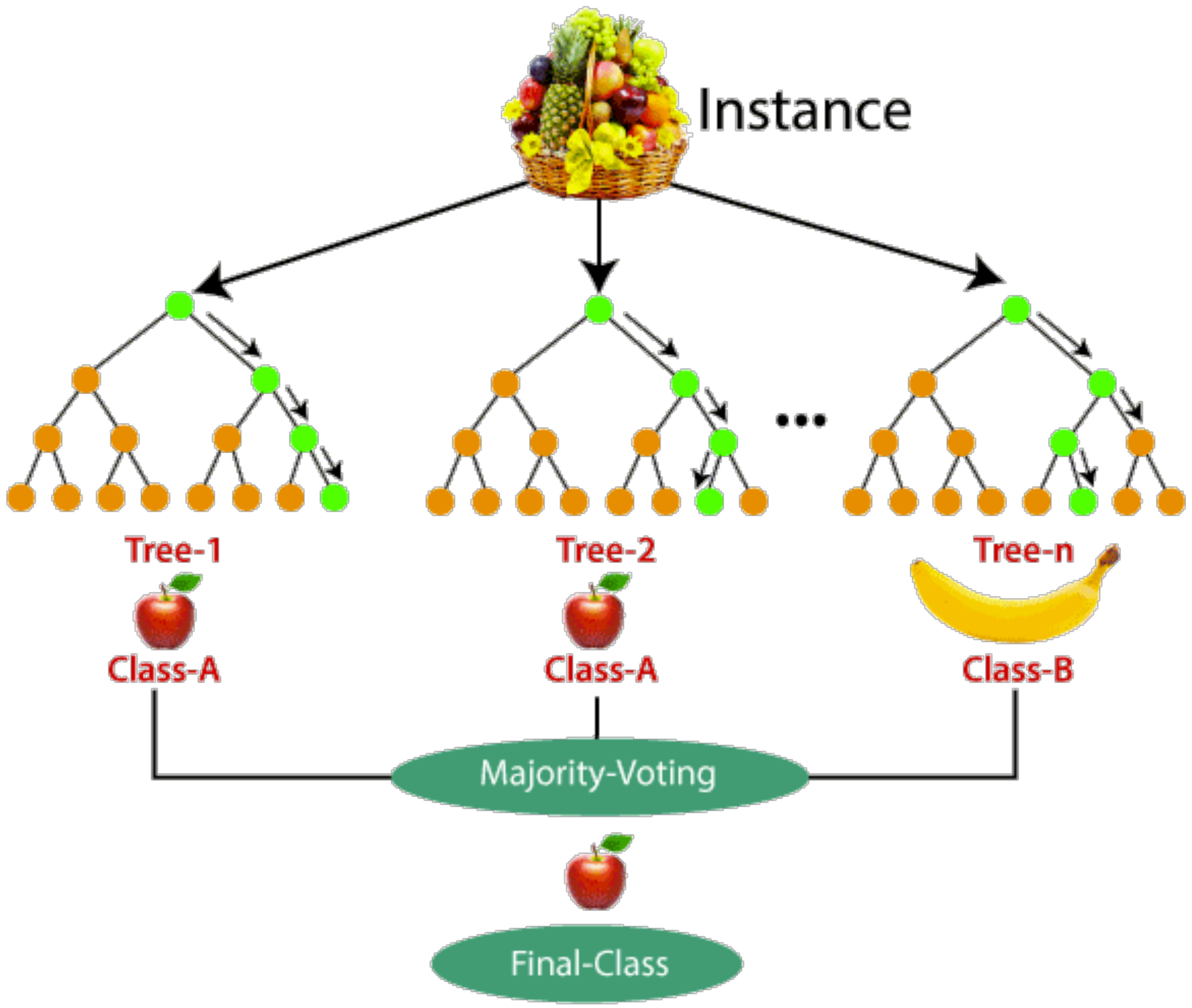

Image source: https://www.section.io/engineering-education/introduction-to-random-forest-in-machine-learning/

Random forests

Parameters:

- B: number of bootstrap samples
- m: number of variables to sample at each split
- criterion to stop splitting, like max number of nodes and/or min samples per node

Training:

- Extract B bootstrap samples from your training data

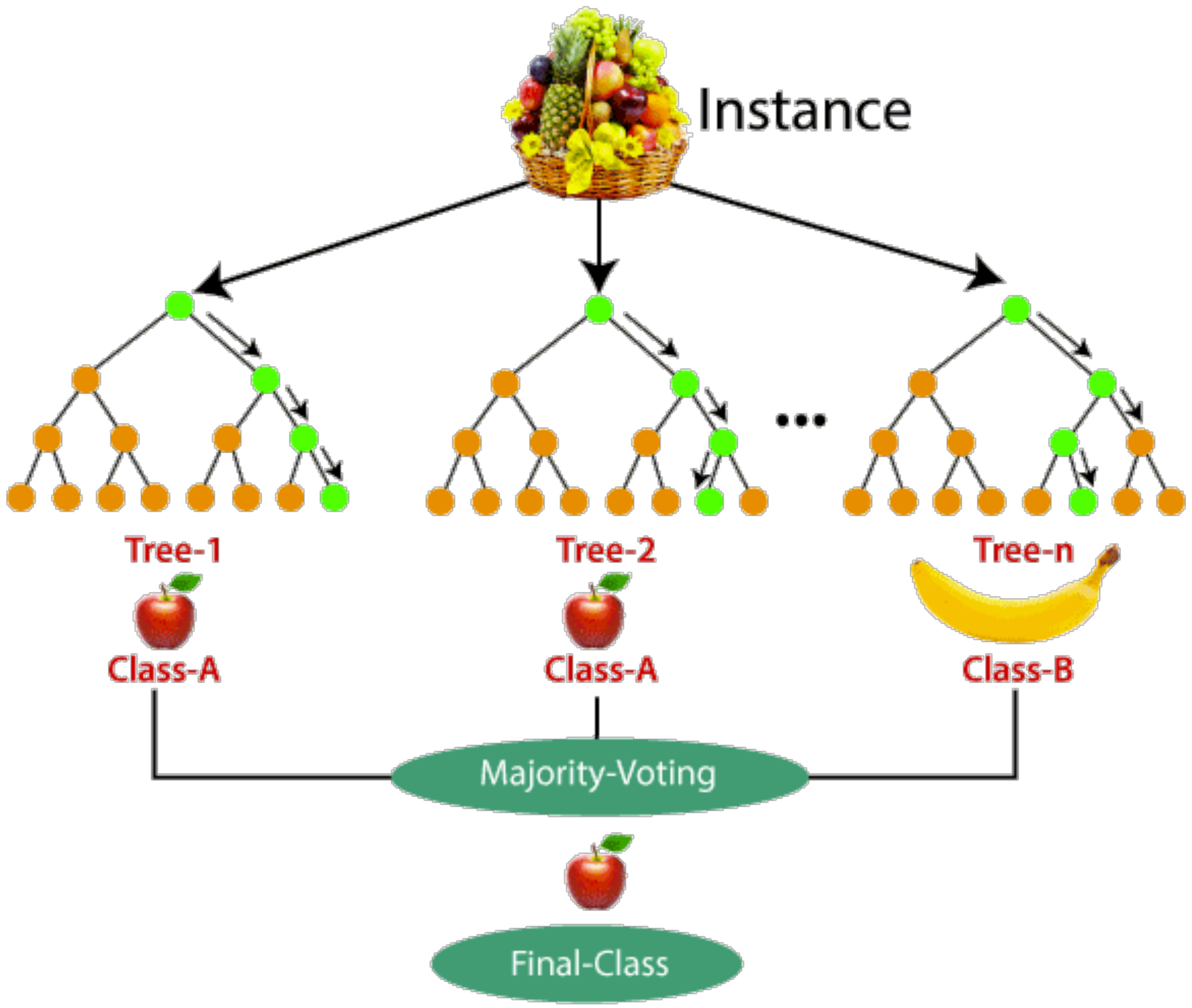

Image source: https://www.section.io/engineering-education/introduction-to-random-forest-in-machine-learning/

Random forests

Parameters:

- B: number of bootstrap samples
- m: number of variables to sample at each split
- criterion to stop splitting, like max number of nodes and/or min samples per node

Training:

- Extract B bootstrap samples from your training data
- For each bootstrap sample $b = 1,\ldots,B$,
  - Grow a decision tree based on the bootstrap sample, randomly sampling m candidate variable to split on at each step, until stopping criterion is met

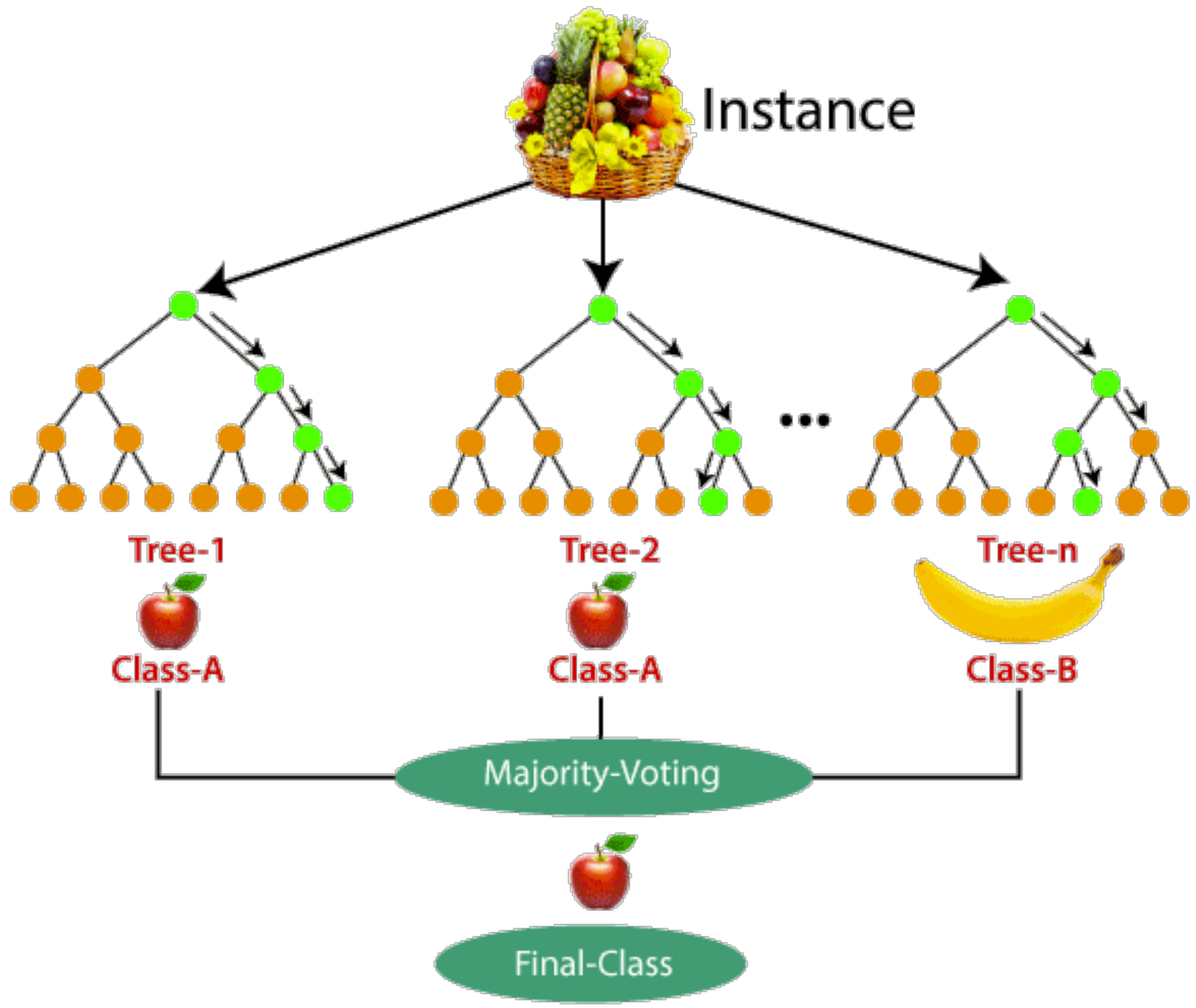

Image source: https://www.section.io/engineering-education/introduction-to-random-forest-in-machine-learning/

Random forests

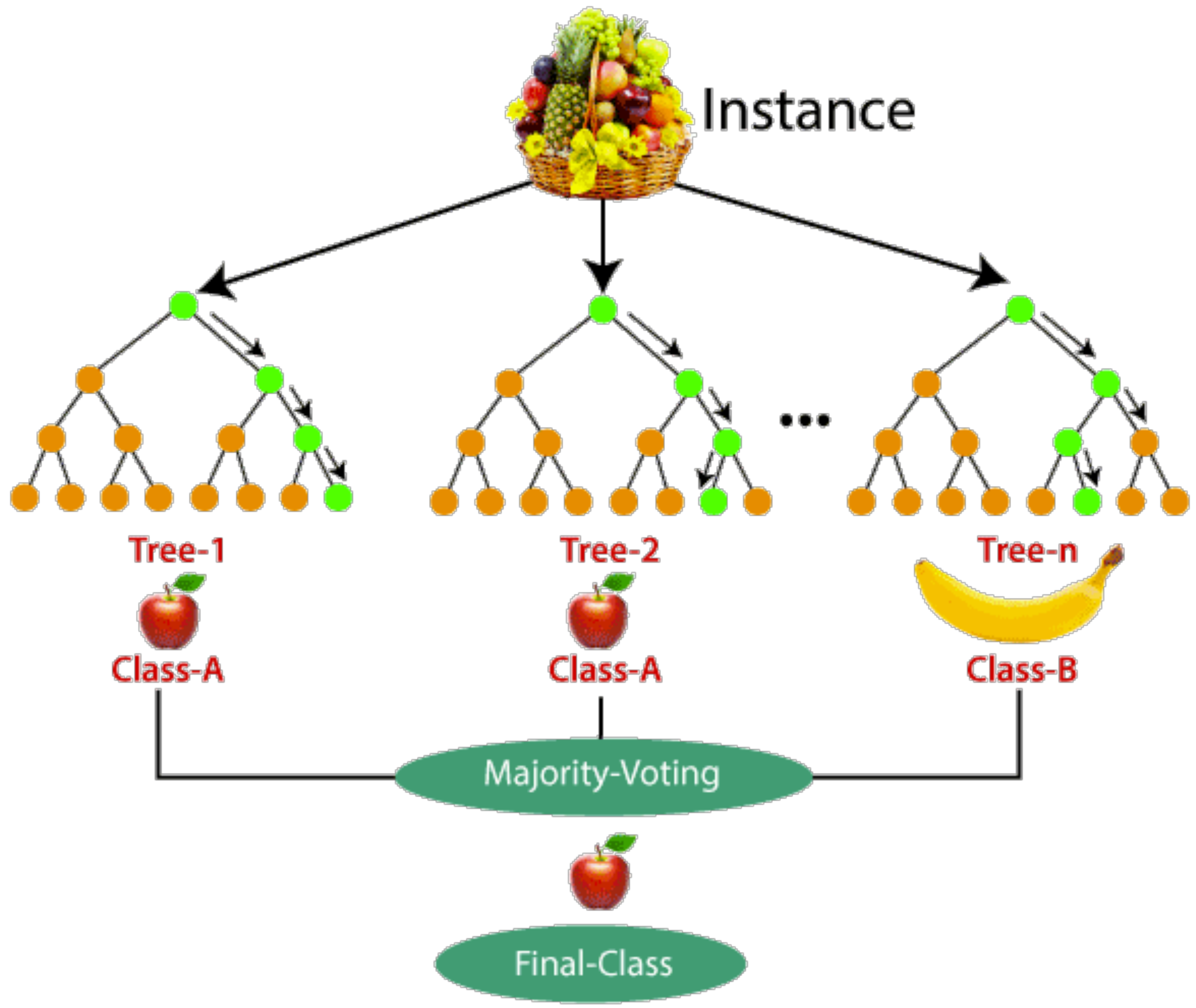

Image source: https://www.section.io/engineering-education/introduction-to-random-forest-in-machine-learning/

Parameters:

- B: number of bootstrap samples
- m: number of variables to sample at each split
- criterion to stop splitting, like max number of nodes and/or min samples per node

Training:

- Extract B bootstrap samples from your training data
- For each bootstrap sample $b = 1, \ldots, B$,
  - Grow a decision tree based on the bootstrap sample, randomly sampling m candidate variable to split on at each step, until stopping criterion is met

Prediction:

- aggregate the decision trees using the mean (for regression) or mode (for classification)

A bias-variance trade-off in choosing m

A bias-variance trade-off in choosing m

If m is larger, the random forest will have lower bias (it can better fit the underlying trend) but higher variance (more correlated trees).

A bias-variance trade-off in choosing m

If m is larger, the random forest will have lower bias (it can better fit the underlying trend) but higher variance (more correlated trees).

If m is smaller, the random forest will have higher bias (it might not be able to fit the underlying trend as well) but lower variance (less correlated trees).

A bias-variance trade-off in choosing m

If m is larger, the random forest will have lower bias (it can better fit the underlying trend) but higher variance (more correlated trees).

If m is smaller, the random forest will have higher bias (it might not be able to fit the underlying trend as well) but lower variance (less correlated trees).

Default choices: $m = p/3$ for regression and $m = \sqrt{p}$ for classification.

A bias-variance trade-off in choosing m

If m is larger, the random forest will have lower bias (it can better fit the underlying trend) but higher variance (more correlated trees).

If m is smaller, the random forest will have higher bias (it might not be able to fit the underlying trend as well) but lower variance (less correlated trees).

Default choices: $m = p/3$ for regression and $m = \sqrt{p}$ for classification.

For best predictive performance, m should be tuned.

Tuning random forests via out-of-bag error

Tuning random forests via out-of-bag error

We usually tune prediction methods via cross-validation. For random forests, there is a clever and computationally faster alternative: out-of-bag error.

Tuning random forests via out-of-bag error

We usually tune prediction methods via cross-validation. For random forests, there is a clever and computationally faster alternative: out-of-bag error.

The idea behind cross-validation is that we want to using parts of our training data as validation sets. By bootstrapping, random forests already do this!

Tuning random forests via out-of-bag error

We usually tune prediction methods via cross-validation. For random forests, there is a clever and computationally faster alternative: out-of-bag error.

The idea behind cross-validation is that we want to using parts of our training data as validation sets. By bootstrapping, random forests already do this!

For each bootstrap sample, define the “bag” to be the set of unique training observations in the sample. Then, predictions based on that tree can be made on the out-of-bag (OOB) samples.

Tuning random forests via out-of-bag error

We usually tune prediction methods via cross-validation. For random forests, there is a clever and computationally faster alternative: out-of-bag error.

The idea behind cross-validation is that we want to using parts of our training data as validation sets. By bootstrapping, random forests already do this!

For each bootstrap sample, define the “bag” to be the set of unique training observations in the sample. Then, predictions based on that tree can be made on the out-of-bag (OOB) samples.

Bootstrap sample *b*

Obs ID	X	Y
5	X_5	Y_5
3	X_3	Y_3
2	X_2	Y_2
3	X_3	Y_3
1	X_1	Y_1

$\longrightarrow T^{*b}$

Tuning random forests via out-of-bag error

We usually tune prediction methods via cross-validation. For random forests, there is a clever and computationally faster alternative: out-of-bag error.

The idea behind cross-validation is that we want to using parts of our training data as validation sets. By bootstrapping, random forests already do this!

For each bootstrap sample, define the "bag" to be the set of unique training observations in the sample. Then, predictions based on that tree can be made on the out-of-bag (OOB) samples.

Bootstrap sample *b*

Obs ID	***X***	***Y***
5	X_5	Y_5
3	X_3	Y_3
2	X_2	Y_2
3	X_3	Y_3
1	X_1	Y_1

$\longrightarrow T^{*b}$

Tuning random forests via out-of-bag error

We usually tune prediction methods via cross-validation. For random forests, there is a clever and computationally faster alternative: out-of-bag error.

The idea behind cross-validation is that we want to using parts of our training data as validation sets. By bootstrapping, random forests already do this!

For each bootstrap sample, define the "bag" to be the set of unique training observations in the sample. Then, predictions based on that tree can be made on the out-of-bag (OOB) samples.

Bootstrap sample *b*

Obs ID	***X***	***Y***
5	X_5	Y_5
3	X_3	Y_3
2	X_2	Y_2
3	X_3	Y_3
1	X_1	Y_1
4	X_4	Y_4

$\longrightarrow T^{*b}$

4, X_4, Y_4: out-of-bag *b*

Tuning random forests via out-of-bag error

We usually tune prediction methods via cross-validation. For random forests, there is a clever and computationally faster alternative: out-of-bag error.

The idea behind cross-validation is that we want to using parts of our training data as validation sets. By bootstrapping, random forests already do this!

For each bootstrap sample, define the “bag” to be the set of unique training observations in the sample. Then, predictions based on that tree can be made on the out-of-bag (OOB) samples.

Bootstrap sample *b*

Obs ID	X	Y
5	X_5	Y_5
3	X_3	Y_3
2	X_2	Y_2
3	X_3	Y_3
1	X_1	Y_1
4	X_4	Y_4

out-of-bag *b*

$$\longrightarrow T^{*b} \longrightarrow T^{*b}(X_4)$$

Out of bag error

Original training data

Obs ID	*X*	*Y*
1	X_1	Y_1
2	X_2	Y_2
3	X_3	Y_3
4	X_4	Y_4
5	X_5	Y_5

Bootstrap sample 1

Obs ID	*X*	*Y*
5	X_5	Y_5
3	X_3	Y_3
2	X_2	Y_2
3	X_3	Y_3
1	X_1	Y_1

$\longrightarrow T^{*1}$

⋮

Bootstrap sample *B*

Obs ID	*X*	*Y*
4	X_4	Y_4
1	X_1	Y_1
1	X_1	Y_1
5	X_5	Y_5
4	X_4	Y_4

$\longrightarrow T^{*B}$

Out of bag error

Original training data

Obs ID	*X*	*Y*
1	X_1	Y_1
2	X_2	Y_2
3	X_3	Y_3
4	X_4	Y_4
5	X_5	Y_5

Bootstrap sample 1

Obs ID	*X*	*Y*
5	X_5	Y_5
3	X_3	Y_3
2	X_2	Y_2
3	X_3	Y_3
1	X_1	Y_1

→ T^{*1}

4	X_4	Y_4

out-of-bag 1

⋮

Bootstrap sample *B*

Obs ID	*X*	*Y*
4	X_4	Y_4
1	X_1	Y_1
1	X_1	Y_1
5	X_5	Y_5
4	X_4	Y_4

→ T^{*B}

Out of bag error

Original training data

Obs ID	X	Y
1	X_1	Y_1
2	X_2	Y_2
3	X_3	Y_3
4	X_4	Y_4
5	X_5	Y_5

Bootstrap sample 1

Obs ID	X	Y
5	X_5	Y_5
3	X_3	Y_3
2	X_2	Y_2
3	X_3	Y_3
1	X_1	Y_1

→ T^{*1}

out-of-bag 1

Obs ID	X	Y
4	X_4	Y_4

⋮

Bootstrap sample B

Obs ID	X	Y
4	X_4	Y_4
1	X_1	Y_1
1	X_1	Y_1
5	X_5	Y_5
4	X_4	Y_4

→ T^{*B}

out-of-bag B

Obs ID	X	Y
2	X_2	Y_2
3	X_3	Y_3

Out of bag error

Original training data

Obs ID	*X*	*Y*
1	X_1	Y_1
2	X_2	Y_2
3	X_3	Y_3
4	X_4	Y_4
5	X_5	Y_5

Bootstrap sample 1

Obs ID	*X*	*Y*
5	X_5	Y_5
3	X_3	Y_3
2	X_2	Y_2
3	X_3	Y_3
1	X_1	Y_1

→ T^{*1}

out-of-bag 1

4	X_4	Y_4

⋮

Bootstrap sample *B*

Obs ID	*X*	*Y*
4	X_4	Y_4
1	X_1	Y_1
1	X_1	Y_1
5	X_5	Y_5
4	X_4	Y_4

→ T^{*B}

out-of-bag *B*

2	X_2	Y_2
3	X_3	Y_3

OOB predictions

Obs ID	*X*	*Y*	$T^{*1}(X)$	…	$T^{*B}(X)$	$\hat{Y}^{OOB}$
1	X_1	Y_1	—	…	—	$\hat{Y}_1^{OOB}$
2	X_2	Y_2	—	…	$T^{*B}(X_2)$	$\hat{Y}_2^{OOB}$
3	X_3	Y_3	—	…	$T^{*B}(X_3)$	$\hat{Y}_3^{OOB}$
4	X_4	Y_4	$T^{*1}(X_4)$	…	—	$\hat{Y}_4^{OOB}$
5	X_5	Y_5	—	…	—	$\hat{Y}_5^{OOB}$

Out of bag error

Original training data

Obs ID	X	Y
1	X_1	Y_1
2	X_2	Y_2
3	X_3	Y_3
4	X_4	Y_4
5	X_5	Y_5

Bootstrap sample 1

Obs ID	X	Y
5	X_5	Y_5
3	X_3	Y_3
2	X_2	Y_2
3	X_3	Y_3
1	X_1	Y_1

→ T^{*1}

out-of-bag 1

Obs ID	X	Y
4	X_4	Y_4

⋮

Bootstrap sample *B*

Obs ID	X	Y
4	X_4	Y_4
1	X_1	Y_1
1	X_1	Y_1
5	X_5	Y_5
4	X_4	Y_4

→ T^{*B}

out-of-bag *B*

Obs ID	X	Y
2	X_2	Y_2
3	X_3	Y_3

OOB predictions

Obs ID	X	Y
1	X_1	Y_1
2	X_2	Y_2
3	X_3	Y_3
4	X_4	Y_4
5	X_5	Y_5

$T^{*1}(X)$	…	$T^{*B}(X)$	$\hat{Y}^{OOB}$
—	…	—	$\hat{Y}_1^{OOB}$
—	…	$T^{*B}(X_2)$	$\hat{Y}_2^{OOB}$
—	…	$T^{*B}(X_3)$	$\hat{Y}_3^{OOB}$
$T^{*1}(X_4)$	…	—	$\hat{Y}_4^{OOB}$
—	…	—	$\hat{Y}_5^{OOB}$

Regression:

$$\hat{Y}_i^{\text{OOB}} = \text{mean}\{T^{*b}(X_i)\}_{i \in \text{OOB}_b}$$

$$\text{OOB err} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i^{\text{OOB}})^2$$

Out of bag error

Original training data

Obs ID	*X*	*Y*
1	X_1	Y_1
2	X_2	Y_2
3	X_3	Y_3
4	X_4	Y_4
5	X_5	Y_5

Bootstrap sample 1

Obs ID	*X*	*Y*
5	X_5	Y_5
3	X_3	Y_3
2	X_2	Y_2
3	X_3	Y_3
1	X_1	Y_1

→ T^{*1}

out-of-bag 1

Obs ID	*X*	*Y*
4	X_4	Y_4

⋮

Bootstrap sample *B*

Obs ID	*X*	*Y*
4	X_4	Y_4
1	X_1	Y_1
1	X_1	Y_1
5	X_5	Y_5
4	X_4	Y_4

→ T^{*B}

out-of-bag *B*

Obs ID	*X*	*Y*
2	X_2	Y_2
3	X_3	Y_3

OOB predictions

Obs ID	*X*	*Y*	$T^{*1}(X)$	…	$T^{*B}(X)$	$\hat{Y}^{OOB}$
1	X_1	Y_1	—	…	—	$\hat{Y}_1^{OOB}$
2	X_2	Y_2	—	…	$T^{*B}(X_2)$	$\hat{Y}_2^{OOB}$
3	X_3	Y_3	—	…	$T^{*B}(X_3)$	$\hat{Y}_3^{OOB}$
4	X_4	Y_4	$T^{*1}(X_4)$	…	—	$\hat{Y}_4^{OOB}$
5	X_5	Y_5	—	…	—	$\hat{Y}_5^{OOB}$

Regression:

$$\hat{Y}_i^{\text{OOB}} = \text{mean}\{T^{*b}(X_i)\}_{i \in \text{OOB}_b}$$

$$\text{OOB err} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i^{\text{OOB}})^2$$

Classification:

$$\hat{Y}_i^{\text{OOB}} = \text{mode}\{T^{*b}(X_i)\}_{i \in \text{OOB}_b}$$

$$\text{OOB err} = \frac{1}{n}\sum_{i=1}^{n} I(Y_i \neq \hat{Y}_i^{\text{OOB}})$$

Parameters to tune (or not)

Parameters to tune (or not)

Random forests generally work pretty well even if not tuned (i.e. if default parameter choices are used).

Parameters to tune (or not)

Random forests generally work pretty well even if not tuned (i.e. if default parameter choices are used).

However, parameters can be tuned using OOB error to improve performance:

Parameters to tune (or not)

Random forests generally work pretty well even if not tuned (i.e. if default parameter choices are used).

However, parameters can be tuned using OOB error to improve performance:

- m: most important tuning parameter

Parameters to tune (or not)

Random forests generally work pretty well even if not tuned (i.e. if default parameter choices are used).

However, parameters can be tuned using OOB error to improve performance:

- m: most important tuning parameter
- criteria to stop splitting: can be tuned but growing trees about as deep as possible generally works pretty well

Parameters to tune (or not)

Random forests generally work pretty well even if not tuned (i.e. if default parameter choices are used).

However, parameters can be tuned using OOB error to improve performance:

- m: most important tuning parameter
- criteria to stop splitting: can be tuned but growing trees about as deep as possible generally works pretty well
- B: least necessary to tune; just choose a large value like 100-1000.

Interpretability and variable importance measures

Interpretability and variable importance measures

Compared to trees, main drawback of random forests is reduced interpretability.

Interpretability and variable importance measures

Compared to trees, main drawback of random forests is reduced interpretability.

However, variable importance measures can help improve the interpretability.

Interpretability and variable importance measures

Compared to trees, main drawback of random forests is reduced interpretability.

However, variable importance measures can help improve the interpretability.

Two types of variable importance measures are used for random forests:

Interpretability and variable importance measures

Compared to trees, main drawback of random forests is reduced interpretability.

However, variable importance measures can help improve the interpretability.

Two types of variable importance measures are used for random forests:

- purity based importance: how much improvement in node purity results from splitting on a feature

Interpretability and variable importance measures

Compared to trees, main drawback of random forests is reduced interpretability.

However, variable importance measures can help improve the interpretability.

Two types of variable importance measures are used for random forests:

- purity based importance: how much improvement in node purity results from splitting on a feature
- OOB prediction based importance: how much deterioration in prediction accuracy results from scrambling a feature out of bag

Purity-based variable importance

Consider the construction of one tree. For each split, note the feature that was split on and resulting reduction in RSS or Gini index (i.e. improvement in purity).

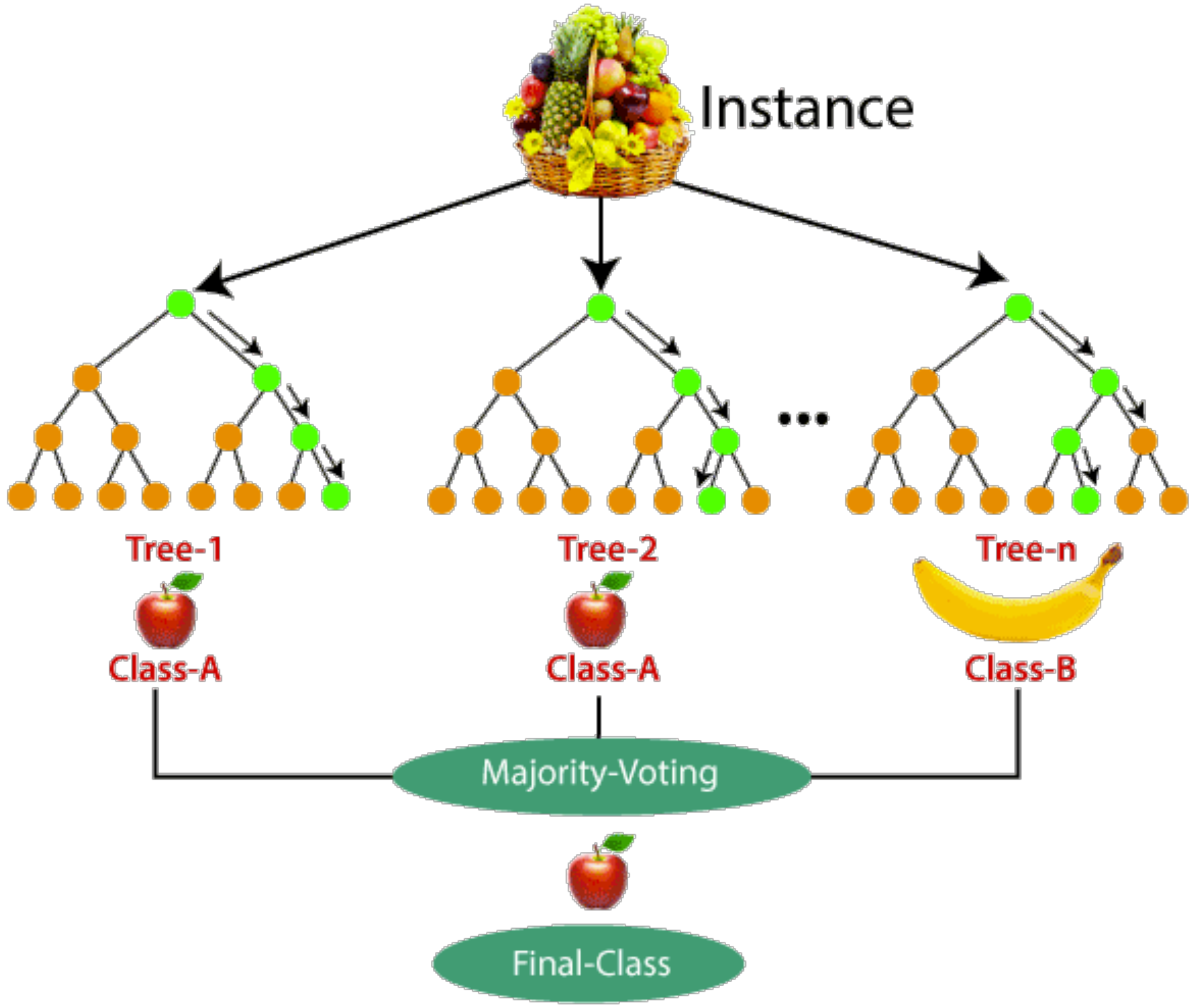

Image source: https://www.section.io/engineering-education/introduction-to-random-forest-in-machine-learning/

Purity-based variable importance

Consider the construction of one tree. For each split, note the feature that was split on and resulting reduction in RSS or Gini index (i.e. improvement in purity).

Define the importance of each feature in this single tree by summing up the improvement in purity for all splits based on this feature.

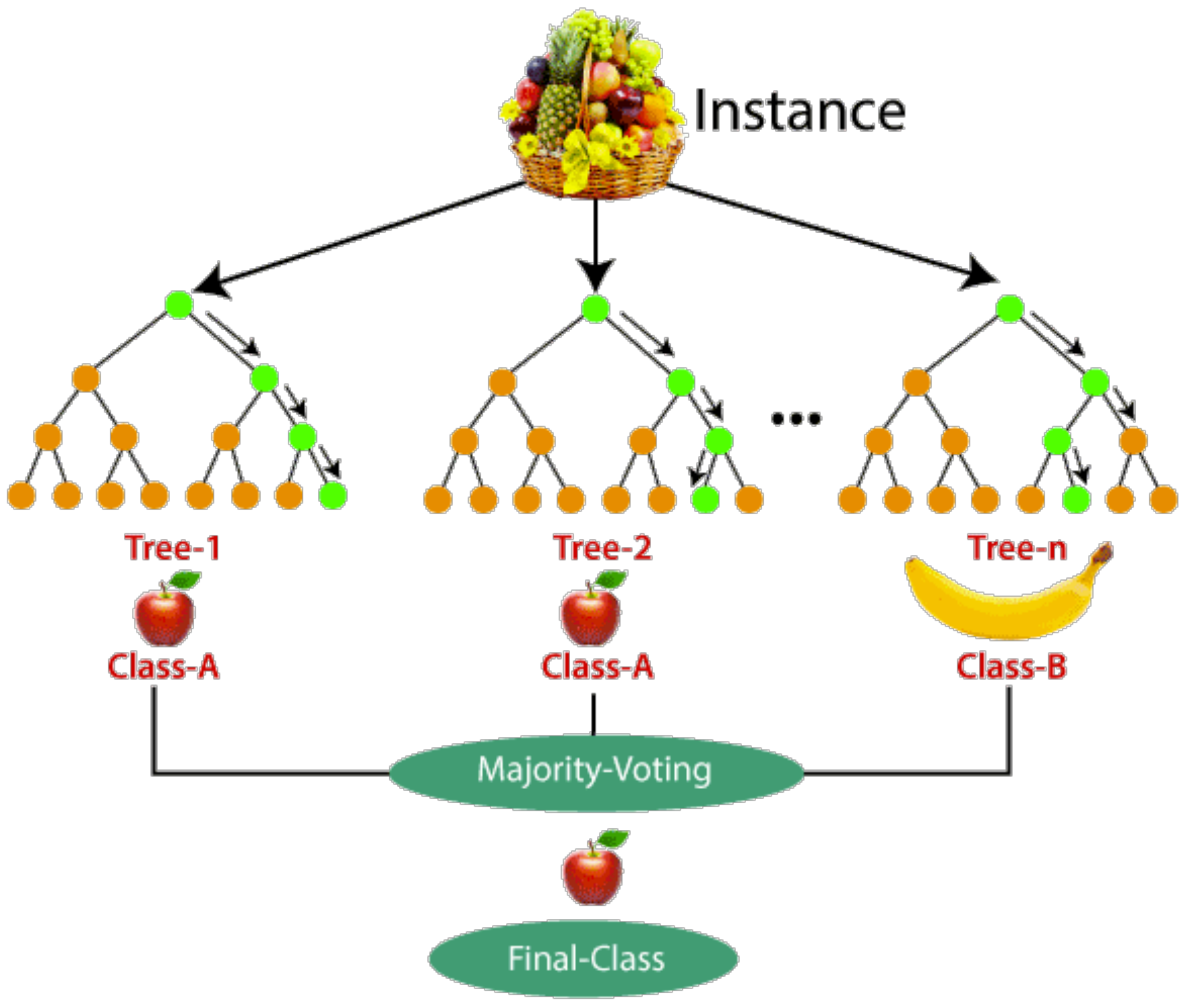

Image source: https://www.section.io/engineering-education/introduction-to-random-forest-in-machine-learning/

Purity-based variable importance

Consider the construction of one tree. For each split, note the feature that was split on and resulting reduction in RSS or Gini index (i.e. improvement in purity).

Define the importance of each feature in this single tree by summing up the improvement in purity for all splits based on this feature.

For random forests, we can average this quantity over all of the trees to get a purity-based variable importance metric.

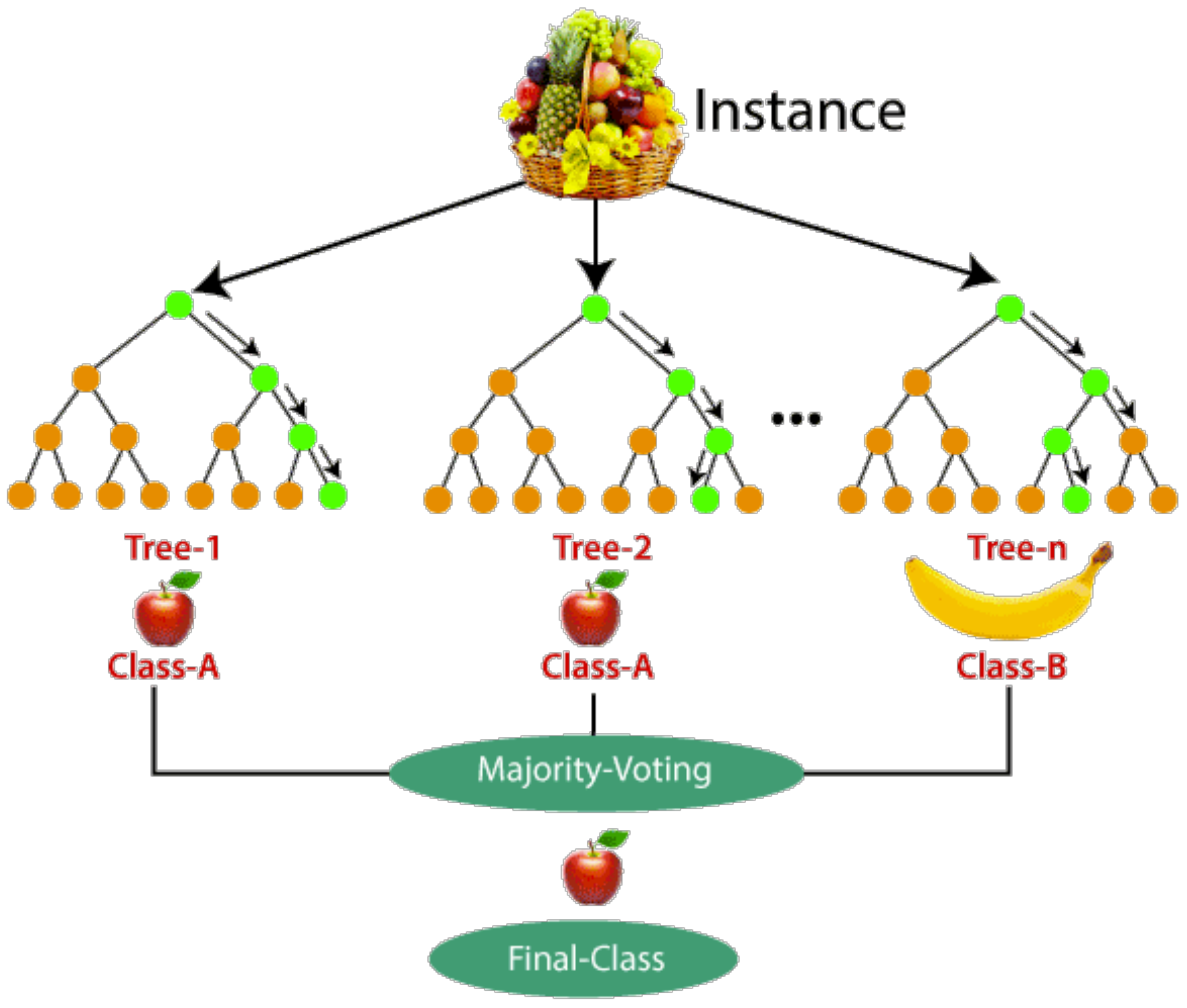

Image source: https://www.section.io/engineering-education/introduction-to-random-forest-in-machine-learning/

OOB prediction based variable importance

OOB prediction based variable importance

Recall the OOB error introduced a few slides ago.

OOB prediction based variable importance

Recall the OOB error introduced a few slides ago.

$\boldsymbol{X}$ =

	x_0	x_1	...	x_j	...	x_{p-1}
X_1	12	0		a		1.5
X_2	-3	1		b		-0.7
X_3	5	0		c		0.2
X_4	16	0		d		-3.5
X_5	-7	1		e		0.9

OOB prediction based variable importance

Recall the OOB error introduced a few slides ago.

Regular OOB predictions

$T^{*1}(X)$	…	$T^{*B}(X)$	$\hat{Y}^{OOB}$
—	…	—	$\hat{Y}_1^{OOB}$
—	…	$T^{*B}(X_2)$	$\hat{Y}_2^{OOB}$
—	…	$T^{*B}(X_3)$	$\hat{Y}_3^{OOB}$
$T^{*1}(X_4)$	…	—	$\hat{Y}_4^{OOB}$
—	…	—	$\hat{Y}_5^{OOB}$

→ Regular OOB error

X =

	x_0	x_1	…	x_j	…	x_{p-1}
X_1	12	0		a		1.5
X_2	-3	1		b		-0.7
X_3	5	0		c		0.2
X_4	16	0		d		-3.5
X_5	-7	1		e		0.9

OOB prediction based variable importance

Recall the OOB error introduced a few slides ago.

For each feature j and each tree, consider making predictions on the OOB data after first scrambling feature j. We can therefore get a scrambled OOB error.

Regular OOB predictions

$T^{*1}(X)$	…	$T^{*B}(X)$	$\hat{Y}^{OOB}$
—	…	—	$\hat{Y}_1^{OOB}$
—	…	$T^{*B}(X_2)$	$\hat{Y}_2^{OOB}$
—	…	$T^{*B}(X_3)$	$\hat{Y}_3^{OOB}$
$T^{*1}(X_4)$	…	—	$\hat{Y}_4^{OOB}$
—	…	—	$\hat{Y}_5^{OOB}$

→ Regular OOB error

X =

	x_0	x_1	…	x_j	…	x_{p-1}
X_1	12	0		a		1.5
X_2	-3	1		b		-0.7
X_3	5	0		c		0.2
X_4	16	0		d		-3.5
X_5	-7	1		e		0.9

OOB prediction based variable importance

Recall the OOB error introduced a few slides ago.

For each feature j and each tree, consider making predictions on the OOB data after first scrambling feature j. We can therefore get a scrambled OOB error.

Regular OOB predictions

$T^{*1}(X)$	…	$T^{*B}(X)$	$\hat{Y}^{OOB}$
—	…	—	$\hat{Y}_1^{OOB}$
—	…	$T^{*B}(X_2)$	$\hat{Y}_2^{OOB}$
—	…	$T^{*B}(X_3)$	$\hat{Y}_3^{OOB}$
$T^{*1}(X_4)$	…	—	$\hat{Y}_4^{OOB}$
—	…	—	$\hat{Y}_5^{OOB}$

→ Regular OOB error

X =

	x_0	x_1	…	x_j	…	x_{p-1}
X_1	12	0		a		1.5
X_2	-3	1		b		-0.7
X_3	5	0		c		0.2
X_4	16	0		d		-3.5
X_5	-7	1		e		0.9

scramble →

X =

	x_0	x_1	…	x_j	…	x_{p-1}
X_1	12	0		b		1.5
X_2	-3	1		e		-0.7
X_3	5	0		d		0.2
X_4	16	0		c		-3.5
X_5	-7	1		a		0.9

OOB prediction based variable importance

Recall the OOB error introduced a few slides ago.

For each feature j and each tree, consider making predictions on the OOB data after first scrambling feature j. We can therefore get a scrambled OOB error.

Regular OOB predictions

$T^{*1}(\mathbf{X})$	…	$T^{*B}(\mathbf{X})$	$\hat{Y}^{OOB}$
—	…	—	$\hat{Y}_1^{OOB}$
—	…	$T^{*B}(X_2)$	$\hat{Y}_2^{OOB}$
—	…	$T^{*B}(X_3)$	$\hat{Y}_3^{OOB}$
$T^{*1}(X_4)$	…	—	$\hat{Y}_4^{OOB}$
—	…	—	$\hat{Y}_5^{OOB}$

→ Regular OOB error

Scrambled OOB predictions

$T^{*1}(\mathbf{X})$	…	$T^{*B}(\mathbf{X})$	$\hat{Y}^{OOB}$
—	…	—	$\hat{Y}_1^{OOB}$
—	…	$T^{*B}(X_2)$	$\hat{Y}_2^{OOB}$
—	…	$T^{*B}(X_3)$	$\hat{Y}_3^{OOB}$
$T^{*1}(X_4)$	…	—	$\hat{Y}_4^{OOB}$
—	…	—	$\hat{Y}_5^{OOB}$

→ Scrambled OOB error

$\mathbf{X}$ =

	x_0	x_1	…	x_j	…	x_{p-1}
X_1	12	0		a		1.5
X_2	-3	1		b		-0.7
X_3	5	0		c		0.2
X_4	16	0		d		-3.5
X_5	-7	1		e		0.9

scramble →

$\mathbf{X}$ =

	x_0	x_1	…	x_j	…	x_{p-1}
X_1	12	0		b		1.5
X_2	-3	1		e		-0.7
X_3	5	0		d		0.2
X_4	16	0		c		-3.5
X_5	-7	1		a		0.9

OOB prediction based variable importance

Recall the OOB error introduced a few slides ago.

For each feature j and each tree, consider making predictions on the OOB data after first scrambling feature j. We can therefore get a scrambled OOB error.

For each feature j, we can define an OOB-prediction-based variable importance by the difference in OOB error when this feature is scrambled and when it is not.

$\mathbf{X}$ =

	x_0	x_1	…	x_j	…	x_{p-1}
X_1	12	0		a		1.5
X_2	-3	1		b		-0.7
X_3	5	0		c		0.2
X_4	16	0		d		-3.5
X_5	-7	1		e		0.9

scramble →

$\mathbf{X}$ =

	x_0	x_1	…	x_j	…	x_{p-1}
X_1	12	0		b		1.5
X_2	-3	1		e		-0.7
X_3	5	0		d		0.2
X_4	16	0		c		-3.5
X_5	-7	1		a		0.9

Regular OOB predictions

$T^{*1}(\mathbf{X})$	…	$T^{*B}(\mathbf{X})$	$\hat{Y}^{OOB}$
—	…	—	$\hat{Y}_1^{OOB}$
—	…	$T^{*B}(X_2)$	$\hat{Y}_2^{OOB}$
—	…	$T^{*B}(X_3)$	$\hat{Y}_3^{OOB}$
$T^{*1}(X_4)$	…	—	$\hat{Y}_4^{OOB}$
—	…	—	$\hat{Y}_5^{OOB}$

→ Regular OOB error

Scrambled OOB predictions

$T^{*1}(\mathbf{X})$	…	$T^{*B}(\mathbf{X})$	$\hat{Y}^{OOB}$
—	…	—	$\hat{Y}_1^{OOB}$
—	…	$T^{*B}(X_2)$	$\hat{Y}_2^{OOB}$
—	…	$T^{*B}(X_3)$	$\hat{Y}_3^{OOB}$
$T^{*1}(X_4)$	…	—	$\hat{Y}_4^{OOB}$
—	…	—	$\hat{Y}_5^{OOB}$

→ Scrambled OOB error

OOB prediction based variable importance

Recall the OOB error introduced a few slides ago.

For each feature j and each tree, consider making predictions on the OOB data after first scrambling feature j. We can therefore get a scrambled OOB error.

For each feature j, we can define an OOB-prediction-based variable importance by the difference in OOB error when this feature is scrambled and when it is not.

$\mathbf{X}$ =

	x_0	x_1	…	x_j	…	x_{p-1}
X_1	12	0		a		1.5
X_2	-3	1		b		-0.7
X_3	5	0		c		0.2
X_4	16	0		d		-3.5
X_5	-7	1		e		0.9

scramble →

$\mathbf{X}$ =

	x_0	x_1	…	x_j	…	x_{p-1}
X_1	12	0		b		1.5
X_2	-3	1		e		-0.7
X_3	5	0		d		0.2
X_4	16	0		c		-3.5
X_5	-7	1		a		0.9

Regular OOB predictions

$T^{*1}(X)$	…	$T^{*B}(X)$	$\hat{Y}_{OOB}$
—	…	—	$\hat{Y}_1^{OOB}$
—	…	$T^{*B}(X_2)$	$\hat{Y}_2^{OOB}$
—	…	$T^{*B}(X_3)$	$\hat{Y}_3^{OOB}$
$T^{*1}(X_4)$	…	—	$\hat{Y}_4^{OOB}$
—	…	—	$\hat{Y}_5^{OOB}$

→ Regular OOB error

Scrambled OOB predictions

$T^{*1}(X)$	…	$T^{*B}(X)$	$\hat{Y}_{OOB}$
—	…	—	$\hat{Y}_1^{OOB}$
—	…	$T^{*B}(X_2)$	$\hat{Y}_2^{OOB}$
—	…	$T^{*B}(X_3)$	$\hat{Y}_3^{OOB}$
$T^{*1}(X_4)$	…	—	$\hat{Y}_4^{OOB}$
—	…	—	$\hat{Y}_5^{OOB}$

→ Scrambled OOB error

Variable Importance =
scrambled OOB err - regular OOB err

Summary

Summary

- Random forests are a fancier version of bagging based on random sub-sampling of m features at each split point.

Summary

- Random forests are a fancier version of bagging based on random sub-sampling of m features at each split point.
- They improve on bagging by de-correlating the bootstrapped decision trees and therefore reducing the variance of the method.

Summary

- Random forests are a fancier version of bagging based on random sub-sampling of m features at each split point.
- They improve on bagging by de-correlating the bootstrapped decision trees and therefore reducing the variance of the method.
- OOB error is a nice alternative to cross-validation error for random forests, and can be used to tune parameters such as m.

Summary

- Random forests are a fancier version of bagging based on random sub-sampling of m features at each split point.
- They improve on bagging by de-correlating the bootstrapped decision trees and therefore reducing the variance of the method.
- OOB error is a nice alternative to cross-validation error for random forests, and can be used to tune parameters such as m.
- Random forests usually give much better prediction performance than individual decision trees, but at the cost of interpretability.

Summary

- Random forests are a fancier version of bagging based on random sub-sampling of m features at each split point.
- They improve on bagging by de-correlating the bootstrapped decision trees and therefore reducing the variance of the method.
- OOB error is a nice alternative to cross-validation error for random forests, and can be used to tune parameters such as m.
- Random forests usually give much better prediction performance than individual decision trees, but at the cost of interpretability.
- Nevertheless, there are a couple ways to measure variable importance in random forests, giving us some interpretability.

Summary

- Random forests are a fancier version of bagging based on random sub-sampling of m features at each split point.
- They improve on bagging by de-correlating the bootstrapped decision trees and therefore reducing the variance of the method.
- OOB error is a nice alternative to cross-validation error for random forests, and can be used to tune parameters such as m.
- Random forests usually give much better prediction performance than individual decision trees, but at the cost of interpretability.
- Nevertheless, there are a couple ways to measure variable importance in random forests, giving us some interpretability.

Random forests are a state-of-the-art tool for predictive modeling.