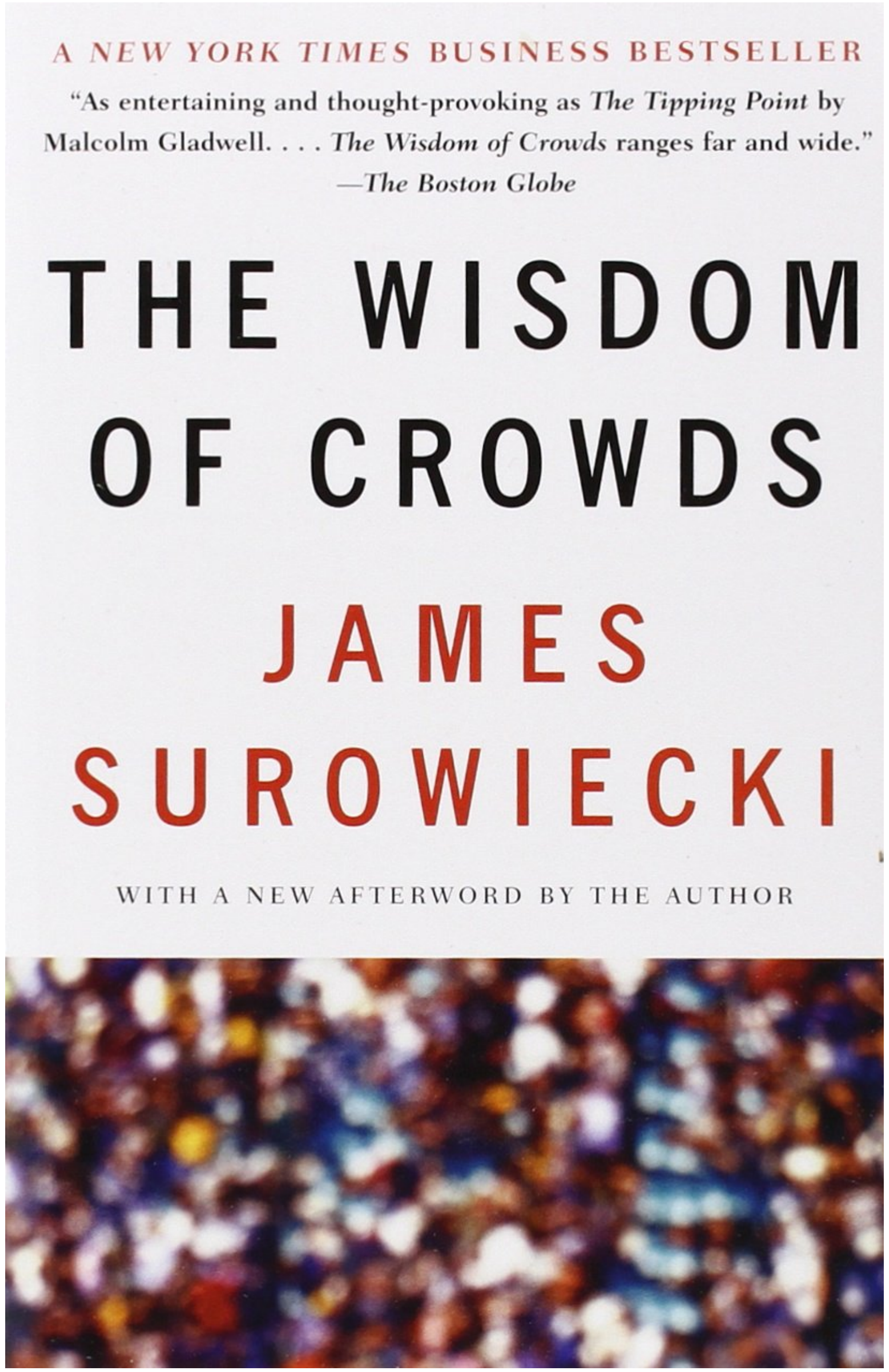
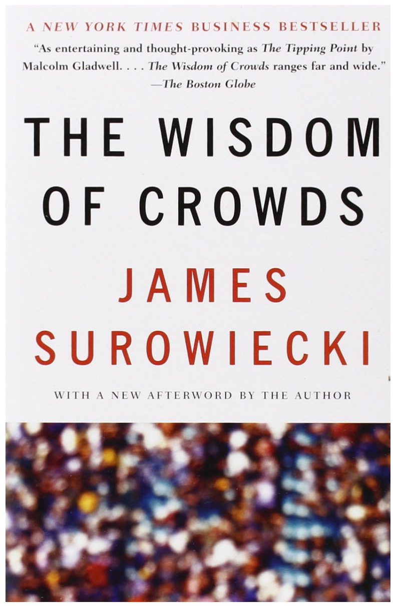
Lecture 7: Ensemble Methods

Wisdom of Crowds



- Knowledge that emerges from a collective decision
- Often better/"more accurate" than that provided by any one individual person
  - Even (individual) experts!

Wisdom of Crowds

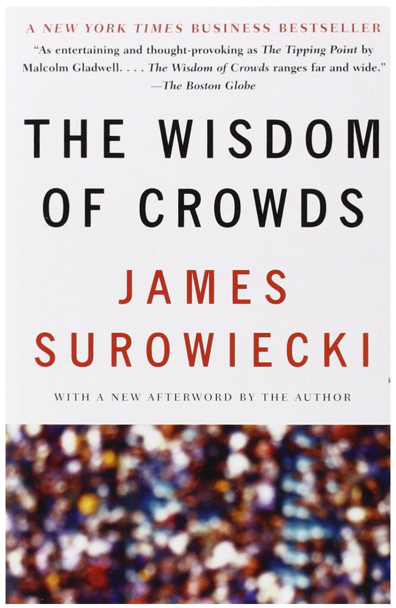


- Knowledge that emerges from a collective decision
- Often better/"more accurate" than that provided by any one individual person
  - Even (individual) experts!

Four conditions:

1. Diversity of opinion
   Each person should have private information
2. Independence
   People's opinions are not always determined by the opinions of those around them
3. Decentralization
   No one at the top dictates crowd's answer. People specialize and draw on local knowledge
4. Aggregation
   Some mechanism exists for turning private judgements into a collective decision

Wisdom of Crowds



Four conditions:

1. Diversity of opinion
   Each person should have private information
2. Independence
   People's opinions are not always determined by the opinions of those around them
3. Decentralization
   No one at the top dictates crowd's answer. People specialize and draw on local knowledge
4. Aggregation
   Some mechanism exists for turning private judgements into a collective decision

- Intuitively, in machine learning:
  - Many models/algorithms
  - Trained independently based on different information/data

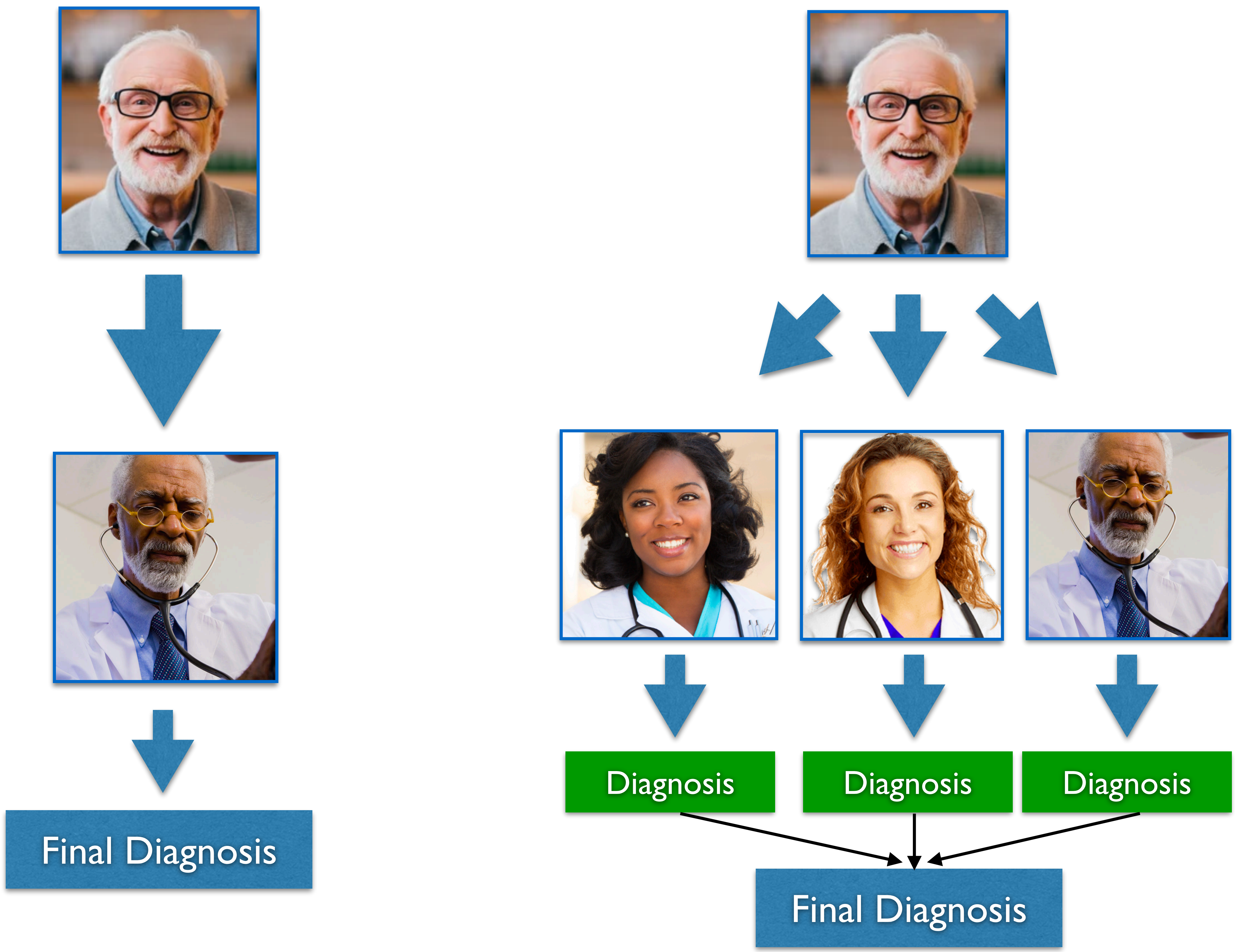
Slide from Bruno Castro da Silva

Wisdom of Crowds

- The Jelly Beans in a Jar experiment
  - Michael Mauboussin — 73 Columbia Business School students
  - *"Just by looking at the jar, can you guess how many jelly beans are there?"*
  - The jar contained 1116 jelly beans
  - Students would guess anywhere from 250 to 4100
    - Average error made by each student: 700 (62%)

Average of all 73 guesses: **1151 (3% error)**

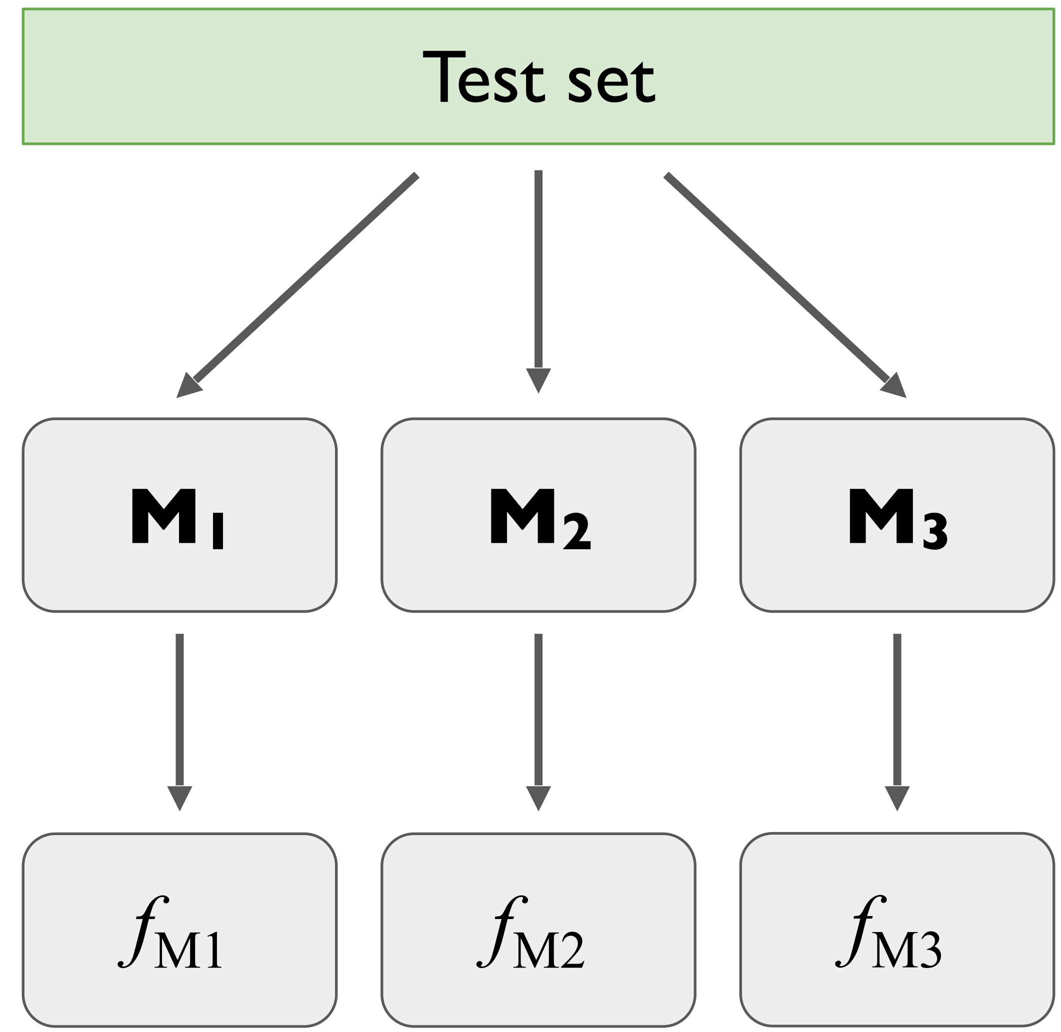
Wisdom of Crowds



Multiple Models in ML

Training set

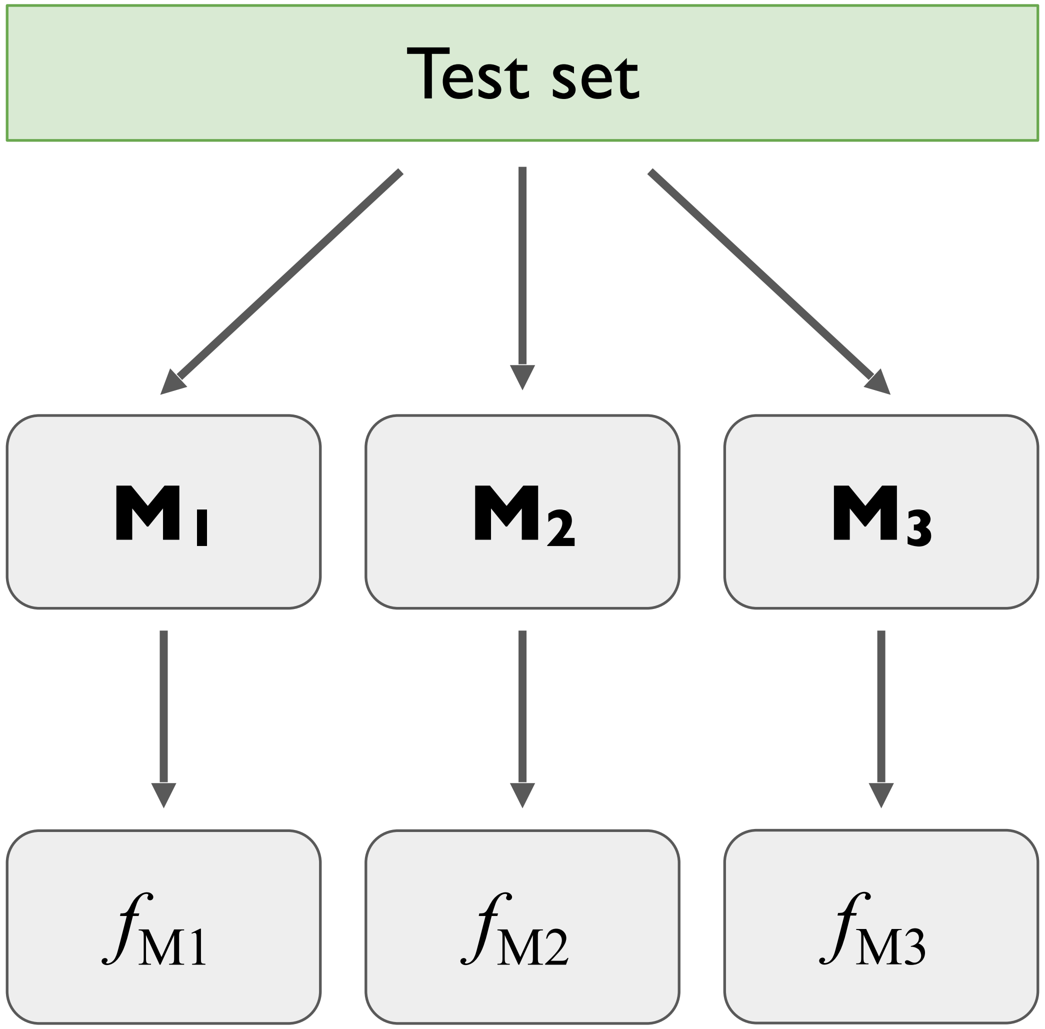
$\mathbf{x}_k$	$\mathbf{y}_k$
$\mathbf{x}_1$	1
$\mathbf{x}_2$	1
$\mathbf{x}_3$	1
$\mathbf{x}_4$	1
$\mathbf{x}_5$	1
$\mathbf{x}_6$	1
$\mathbf{x}_7$	1
$\mathbf{x}_8$	1
$\mathbf{x}_9$	1
$\mathbf{x}_{10}$	1
Accuracy	



Multiple Models in ML

Predictions made by different models
(M_1, M_2, M_3)

$\mathbf{x}_k$	$\mathbf{y}_k$	f_{M1}	f_{M2}	f_{M3}
$\mathbf{x}_1$	1	1	1	1
$\mathbf{x}_2$	1	1	0	1
$\mathbf{x}_3$	1	0	1	1
$\mathbf{x}_4$	1	1	1	1
$\mathbf{x}_5$	1	1	1	1
$\mathbf{x}_6$	1	1	1	0
$\mathbf{x}_7$	1	0	0	0
$\mathbf{x}_8$	1	1	1	0
$\mathbf{x}_9$	1	1	0	1
$\mathbf{x}_{10}$	1	0	1	1
Accuracy		70%	70%	70%

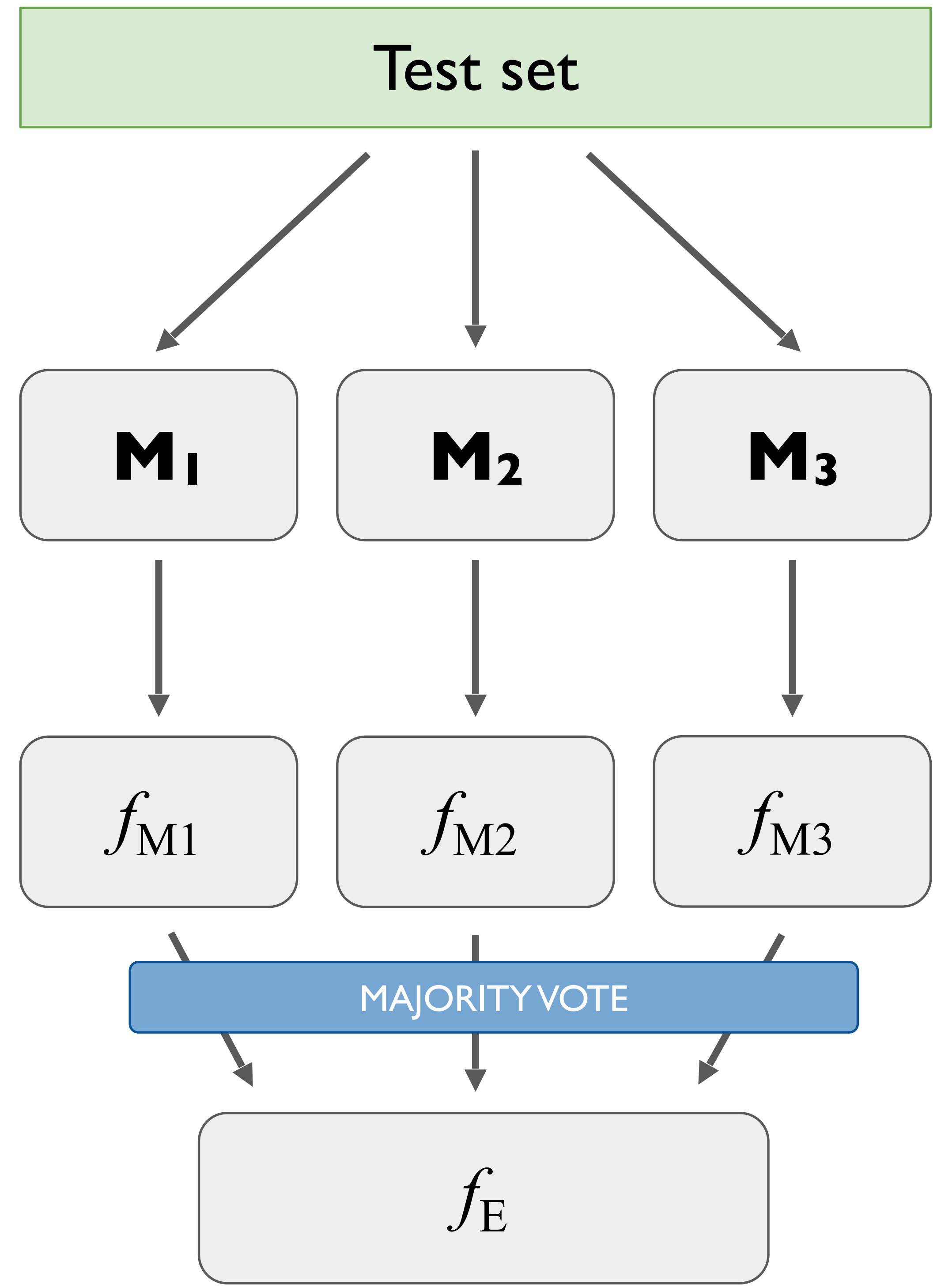


Slide from Bruno Castro da Silva

Multiple Models in ML

Combined model (via majority voting)

$\mathbf{x}_k$	$\mathbf{y}_k$	f_{M1}	f_{M2}	f_{M3}	f_E
$\mathbf{x}_1$	1	1	1	1	1
$\mathbf{x}_2$	1	1	0	1	1
$\mathbf{x}_3$	1	0	1	1	1
$\mathbf{x}_4$	1	1	1	1	1
$\mathbf{x}_5$	1	1	1	1	1
$\mathbf{x}_6$	1	1	1	0	1
$\mathbf{x}_7$	1	0	0	0	0
$\mathbf{x}_8$	1	1	1	0	1
$\mathbf{x}_9$	1	1	0	1	1
$\mathbf{x}_{10}$	1	0	1	1	1
Accuracy		70%	70%	70%	**90%**

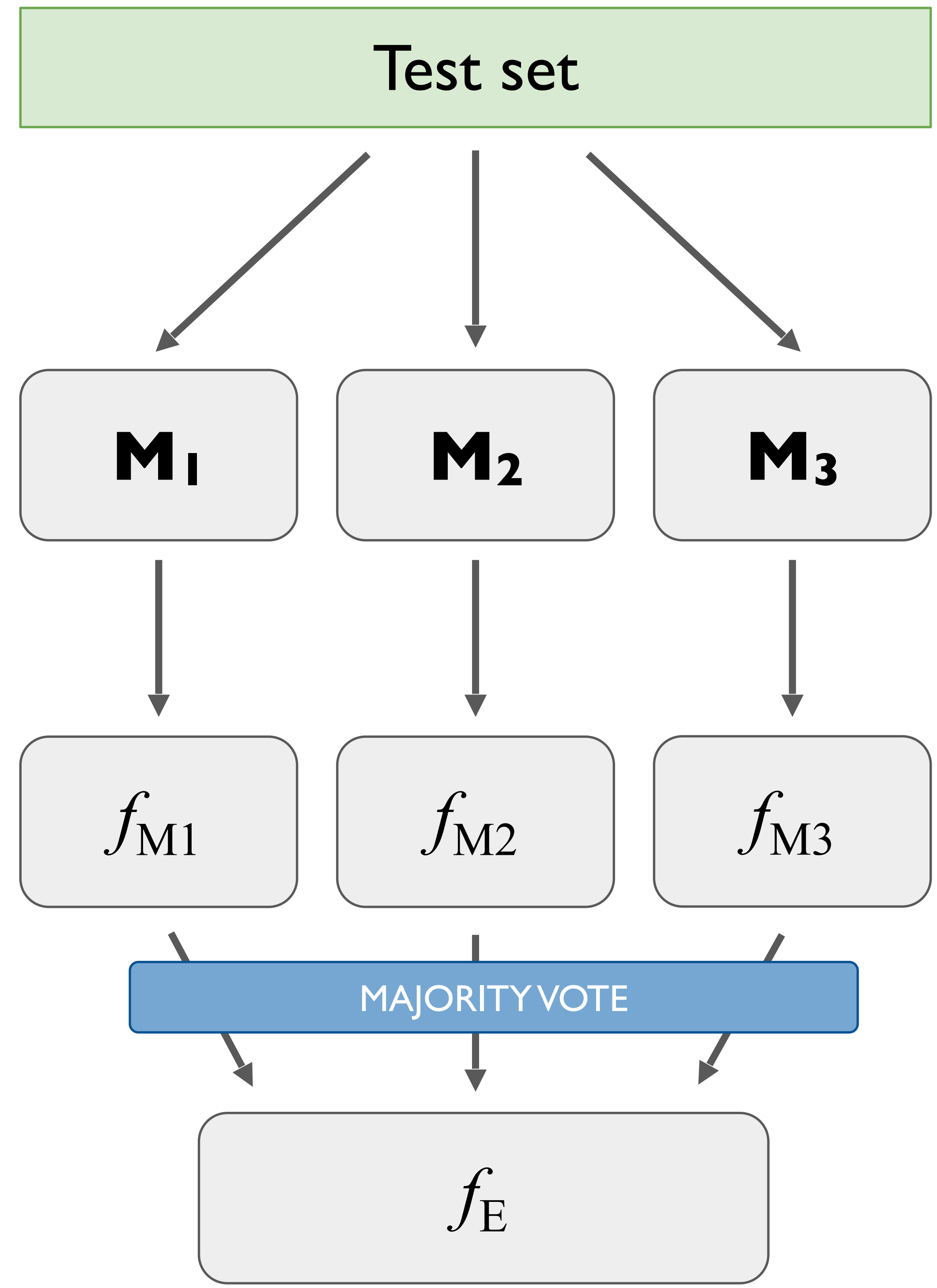


Multiple Models in ML

Combined model (via majority voting)

$\mathbf{x}_k$	$\mathbf{y}_k$	f_{M1}	f_{M2}	f_{M3}	f_E
$\mathbf{x}_1$	1	1	1	1	1
$\mathbf{x}_2$	1	1	0	1	1
$\mathbf{x}_3$	1	0	1	1	1
$\mathbf{x}_8$	1	1	1	0	1
$\mathbf{x}_9$	1	1	0	1	1
$\mathbf{x}_{10}$	1	0	1	1	1
Accuracy		70%	70%	70%	**90%**

The "consensus" resulting from combining each model's predictions tends to have higher accuracy than each individual classifier

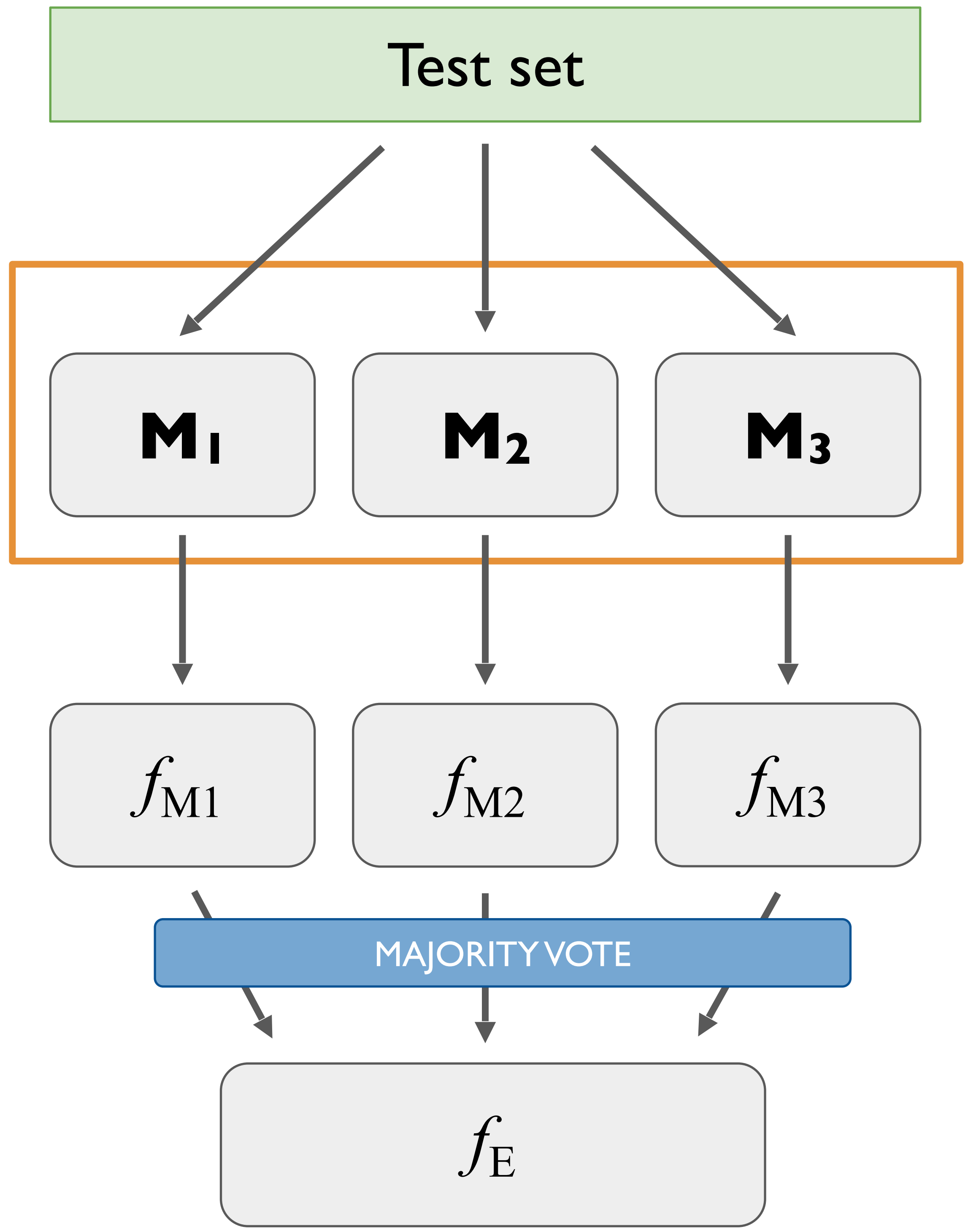


Multiple Models in ML

"The ensemble consensus tends to have higher accuracy than that of its individual classifiers"

Conditions for *ensembles* to perform well:
"**accuracy and diversity**"

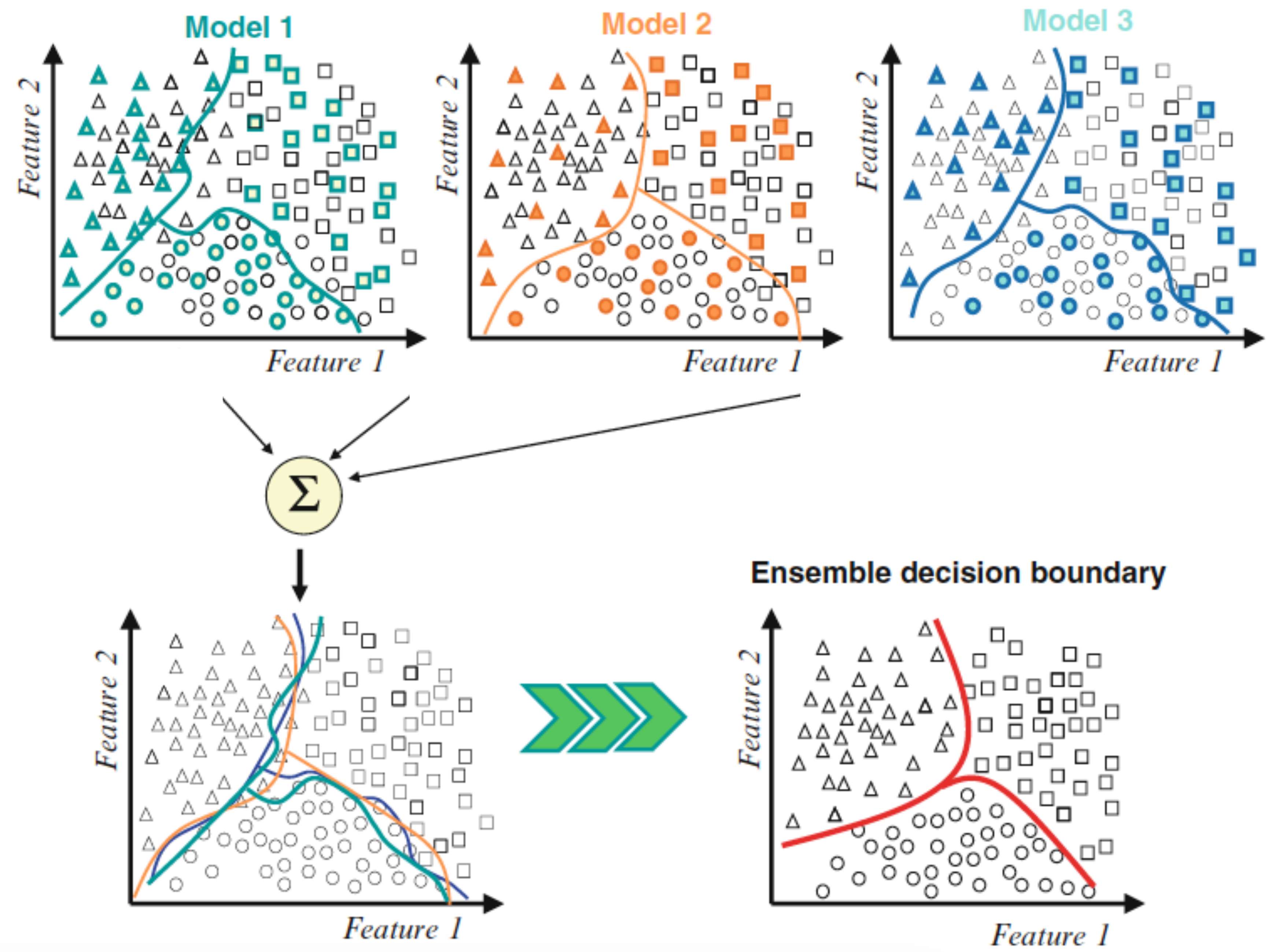
- **Error rate** of individual classifiers **< 50%**
- **Errors made by classifiers are independent**
- Under these conditions:
  - if classifiers have similar error rate (e.g., 45%)
  - the expected error rate of the ensemble decreases linearly with the number of models



Slide from Bruno Castro da Silva

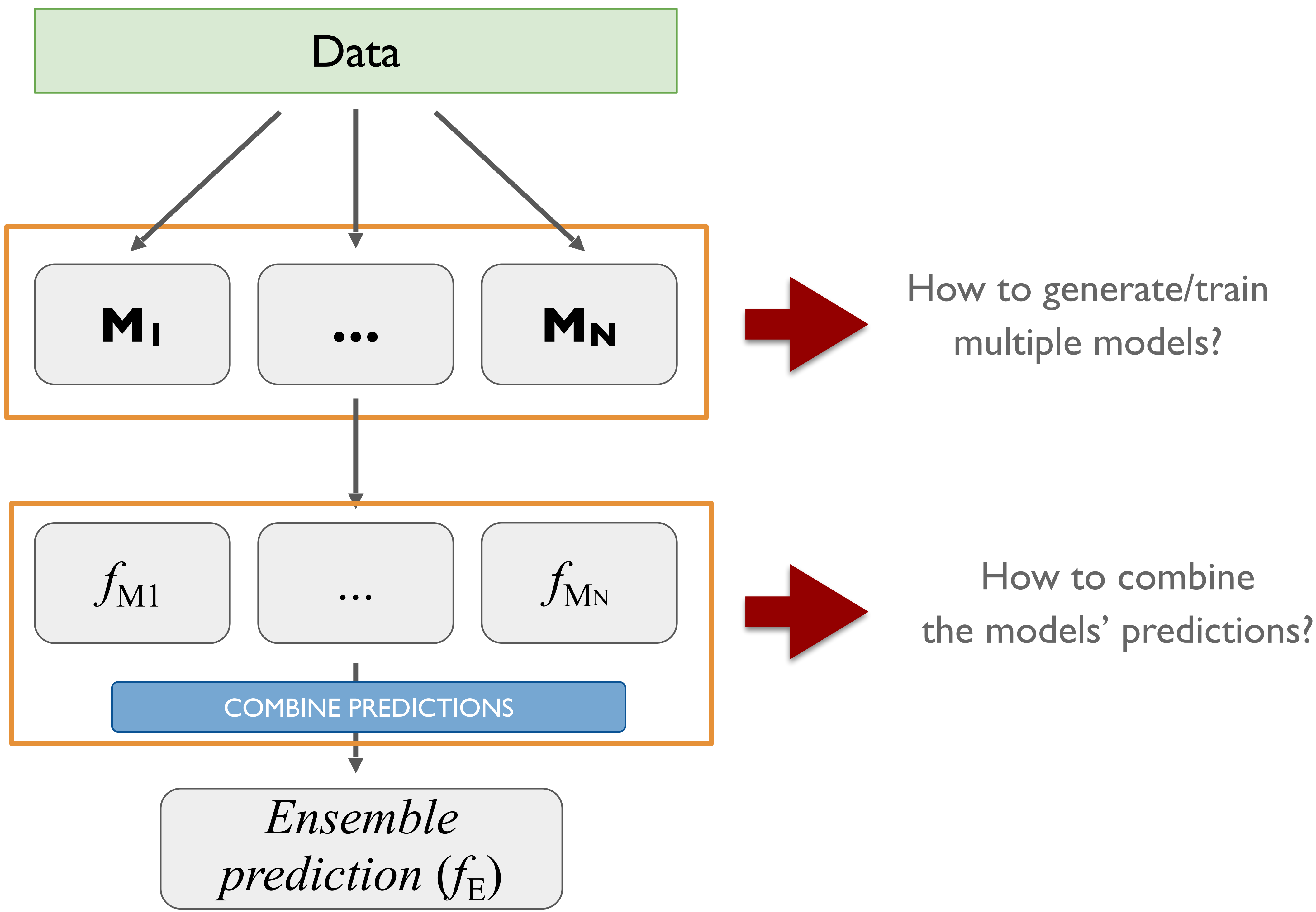
Ensemble Learning

How is the decision boundary of the combined model influenced by the use of multiple models?



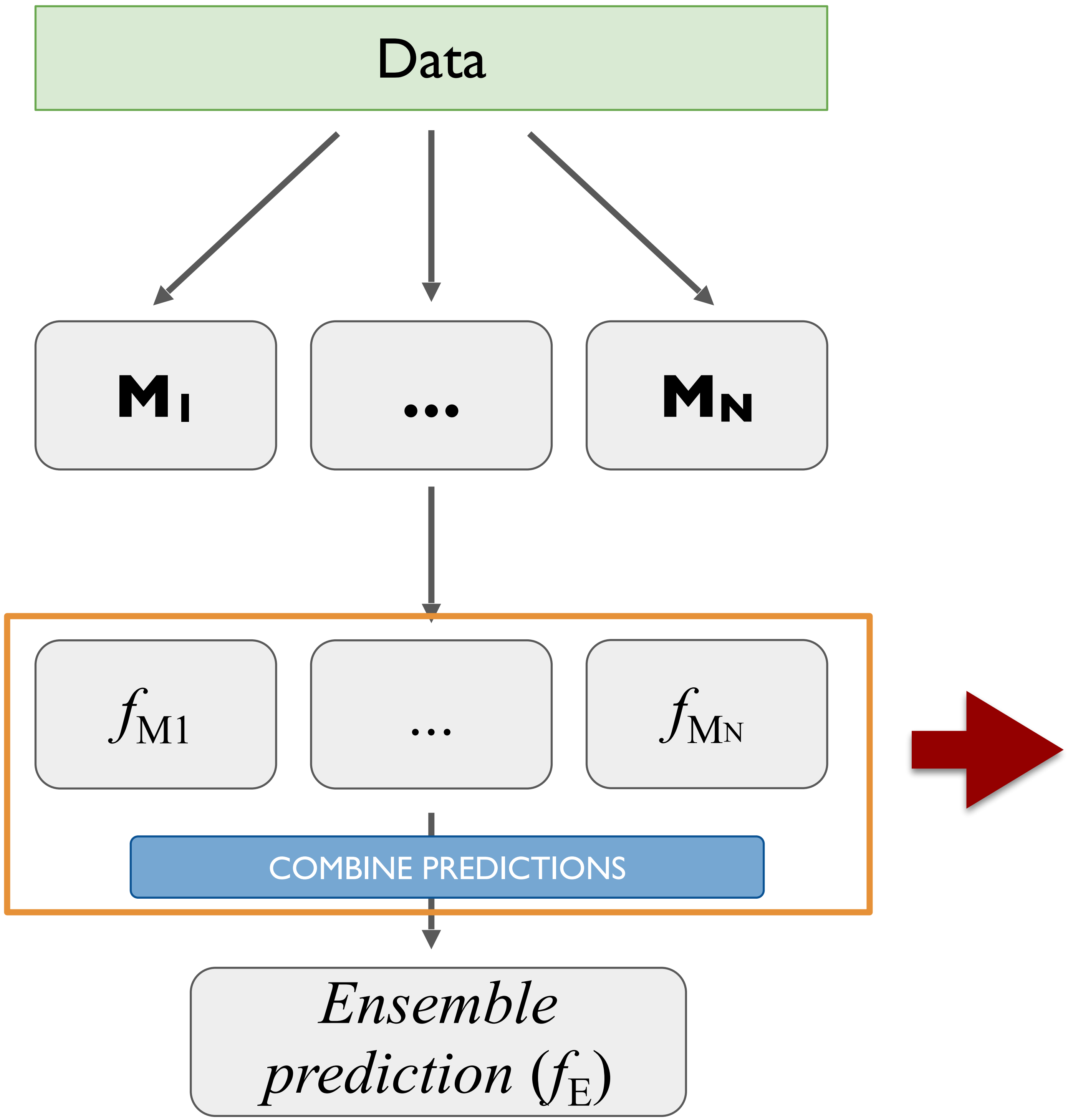
Source: Polikar, 2006

Multiple Models in ML



Slide from Bruno Castro da Silva

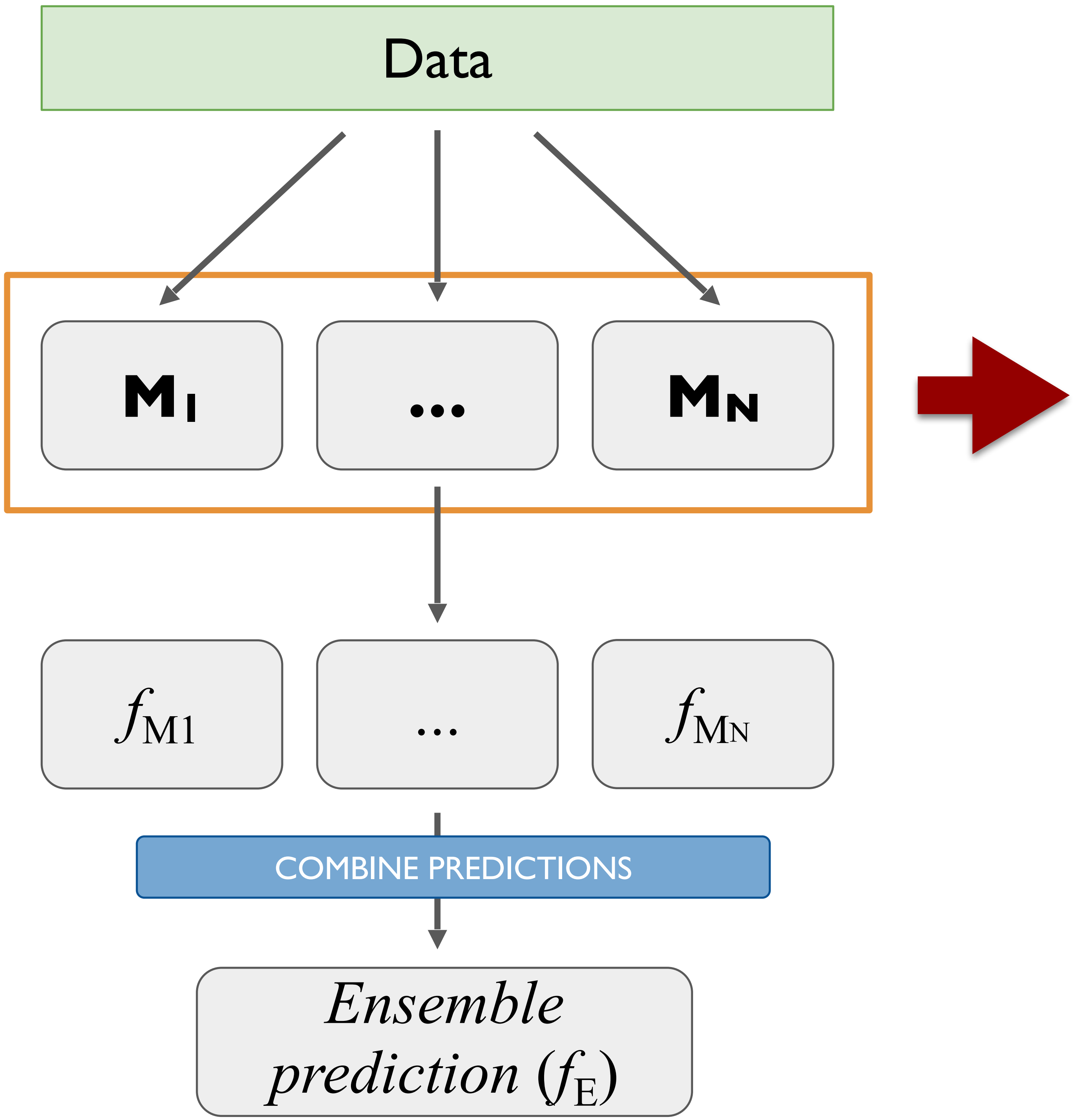
Multiple Models in ML



How to combine
the models' predictions?

Slide from Bruno Castro da Silva

Multiple Models in ML



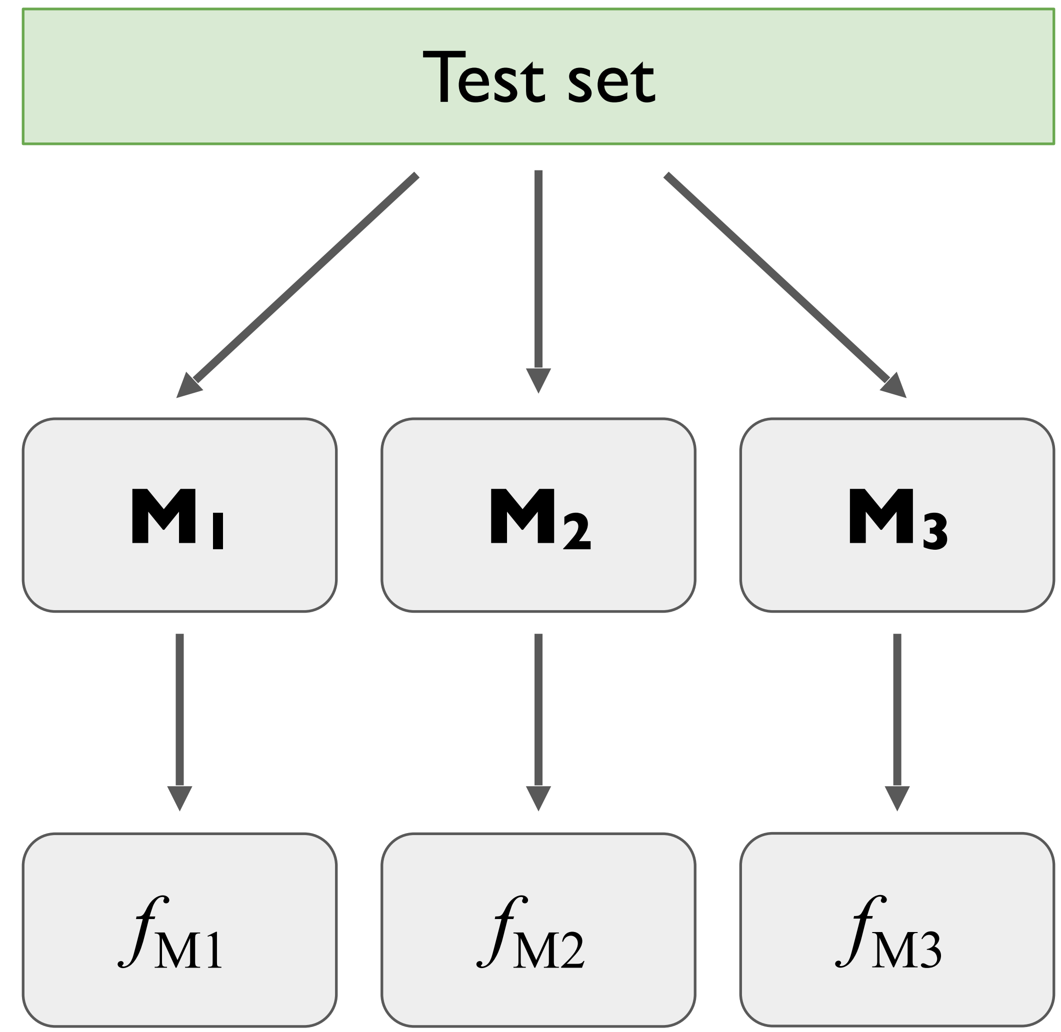
How to generate/train multiple models?

Slide from Bruno Castro da Silva

Multiple Models in ML

Training set

$\mathbf{x}_k$	$\mathbf{y}_k$
$\mathbf{x}_1$	1
$\mathbf{x}_2$	1
$\mathbf{x}_3$	1
$\mathbf{x}_4$	1
$\mathbf{x}_5$	1
$\mathbf{x}_6$	1
$\mathbf{x}_7$	1
$\mathbf{x}_8$	1
$\mathbf{x}_9$	1
$\mathbf{x}_{10}$	1
Accuracy	

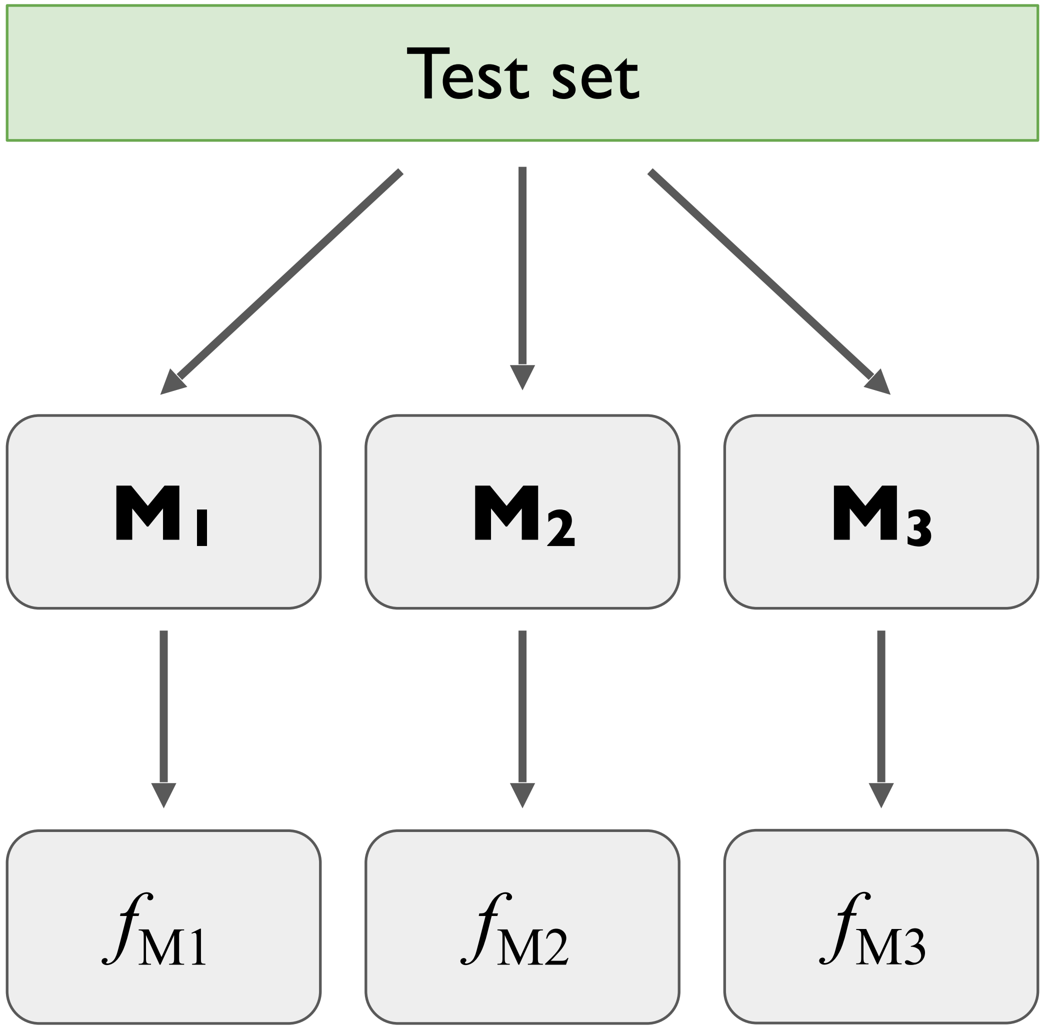


Slide from Bruno Castro da Silva

Multiple Models in ML

Predictions made by different models (M_1, M_2, M_3)

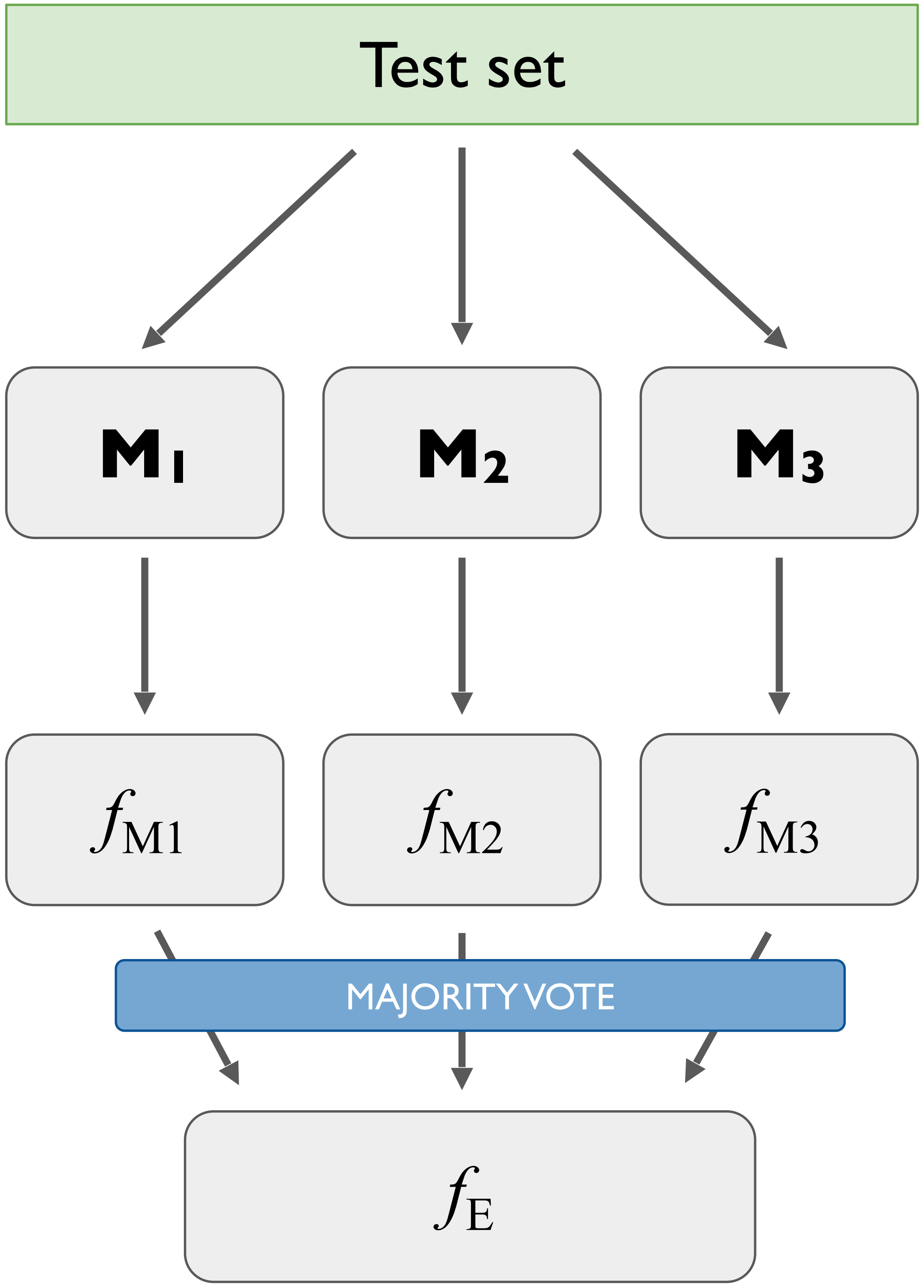
$\mathbf{x}_k$	$\mathbf{y}_k$	f_{M1}	f_{M2}	f_{M3}
$\mathbf{x}_1$	1	1	1	1
$\mathbf{x}_2$	1	1	0	1
$\mathbf{x}_3$	1	0	1	1
$\mathbf{x}_4$	1	1	1	1
$\mathbf{x}_5$	1	1	1	1
$\mathbf{x}_6$	1	1	1	0
$\mathbf{x}_7$	1	0	0	0
$\mathbf{x}_8$	1	1	1	0
$\mathbf{x}_9$	1	1	0	1
$\mathbf{x}_{10}$	1	0	1	1
Accuracy		70%	70%	70%



Multiple Models in ML

Combined model (via majority voting)

$\mathbf{x}_k$	$\mathbf{y}_k$	f_{M1}	f_{M2}	f_{M3}	f_E
$\mathbf{x}_1$	1	1	1	1	1
$\mathbf{x}_2$	1	1	0	1	1
$\mathbf{x}_3$	1	0	1	1	1
$\mathbf{x}_4$	1	1	1	1	1
$\mathbf{x}_5$	1	1	1	1	1
$\mathbf{x}_6$	1	1	1	0	1
$\mathbf{x}_7$	1	0	0	0	0
$\mathbf{x}_8$	1	1	1	0	1
$\mathbf{x}_9$	1	1	0	1	1
$\mathbf{x}_{10}$	1	0	1	1	1
Accuracy		70%	70%	70%	**90%**



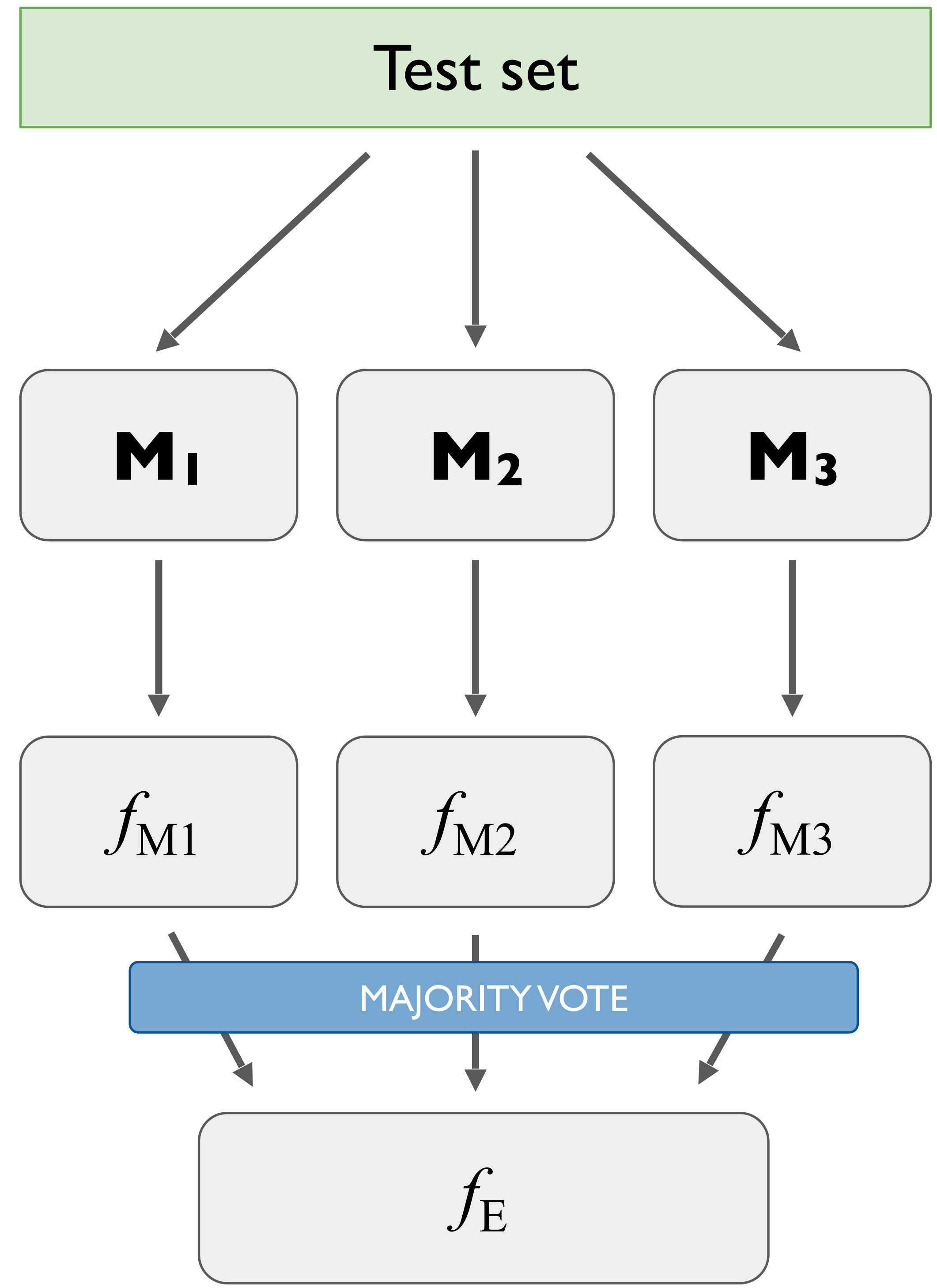
Slide from Bruno Castro da Silva

Multiple Models in ML

Combined model (via majority voting)

$\mathbf{x}_k$	$\mathbf{y}_k$	f_{M1}	f_{M2}	f_{M3}	f_E
$\mathbf{x}_1$	1	1	1	1	1
$\mathbf{x}_2$	1	1	0	1	1
$\mathbf{x}_3$	1	0	1	1	1
$\mathbf{x}_8$	1	1	1	0	1
$\mathbf{x}_9$	1	1	0	1	1
$\mathbf{x}_{10}$	1	0	1	1	1
Accuracy		70%	70%	70%	**90%**

The "consensus" resulting from combining each model's predictions tends to have higher accuracy than each individual classifier

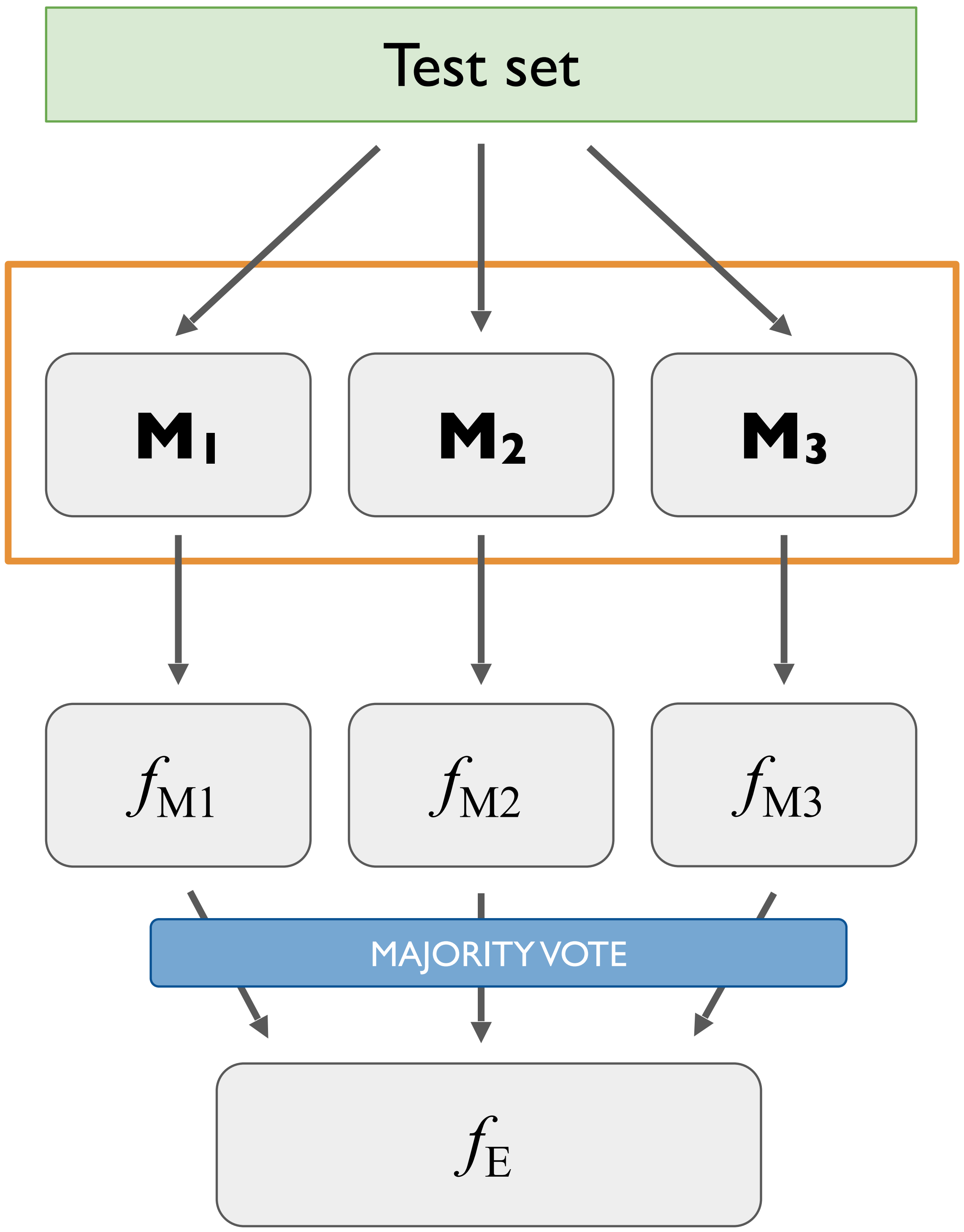


Multiple Models in ML

"The ensemble consensus tends to have higher accuracy than that of its individual classifiers"

Conditions for *ensembles* to perform well:
"**accuracy and diversity**"

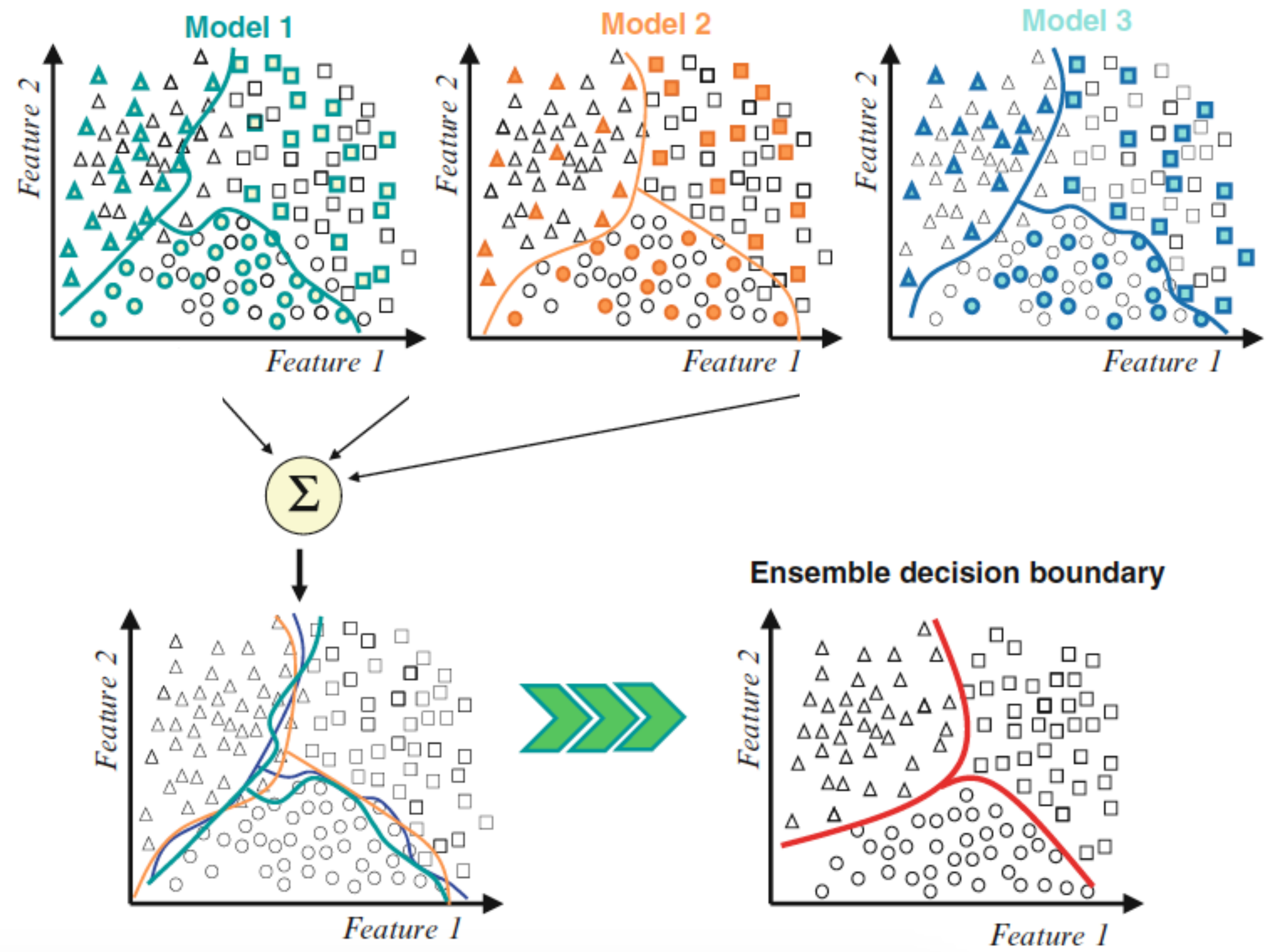
- **Error rate** of individual classifiers **< 50%**
- **Errors made by classifiers are independent**
- Under these conditions:
  - if classifiers have similar error rate (e.g., 45%)
  - the expected error rate of the ensemble decreases linearly with the number of models



Slide from Bruno Castro da Silva

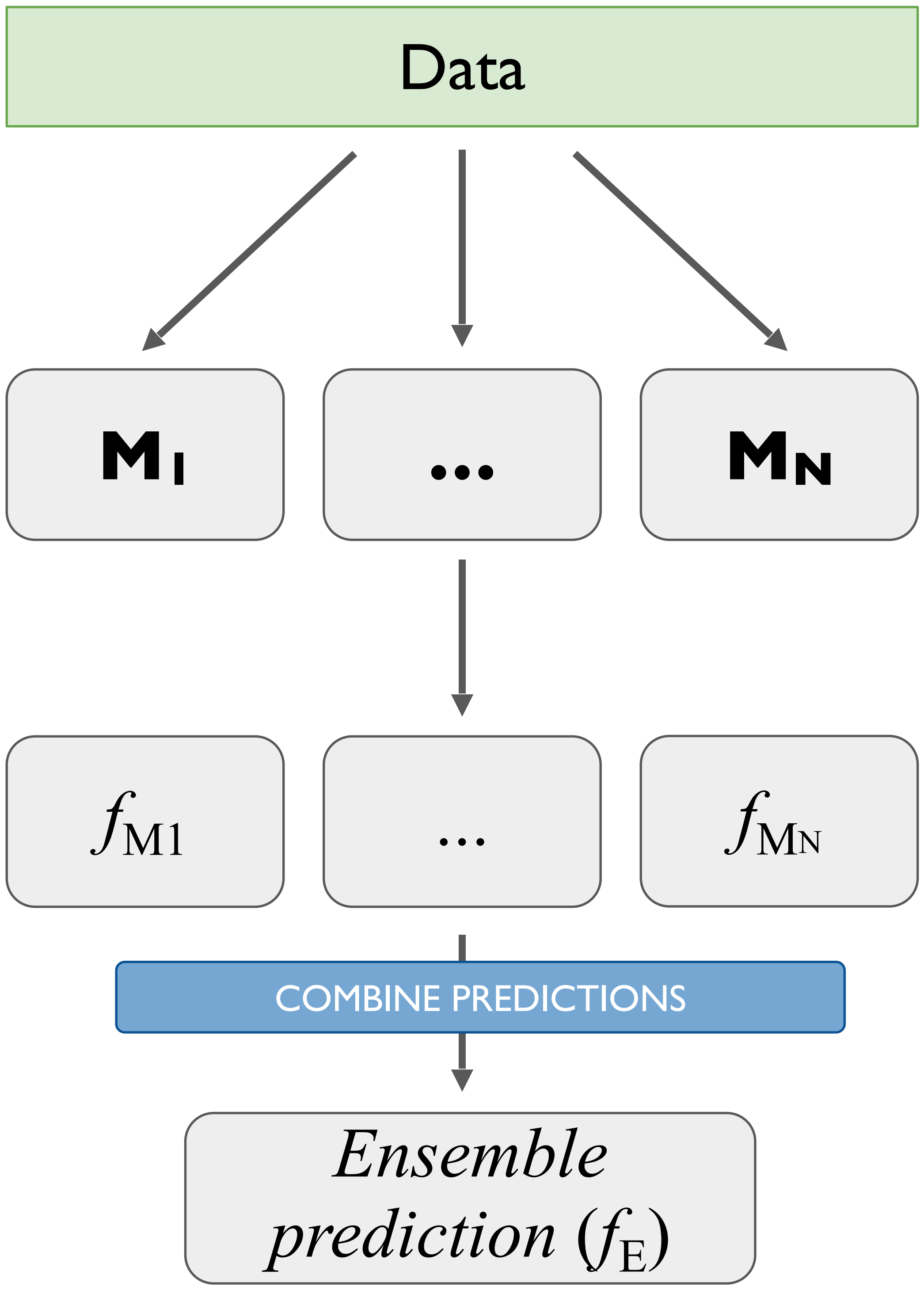
Ensemble Learning

How is the decision boundary of the combined model influenced by the use of multiple models?

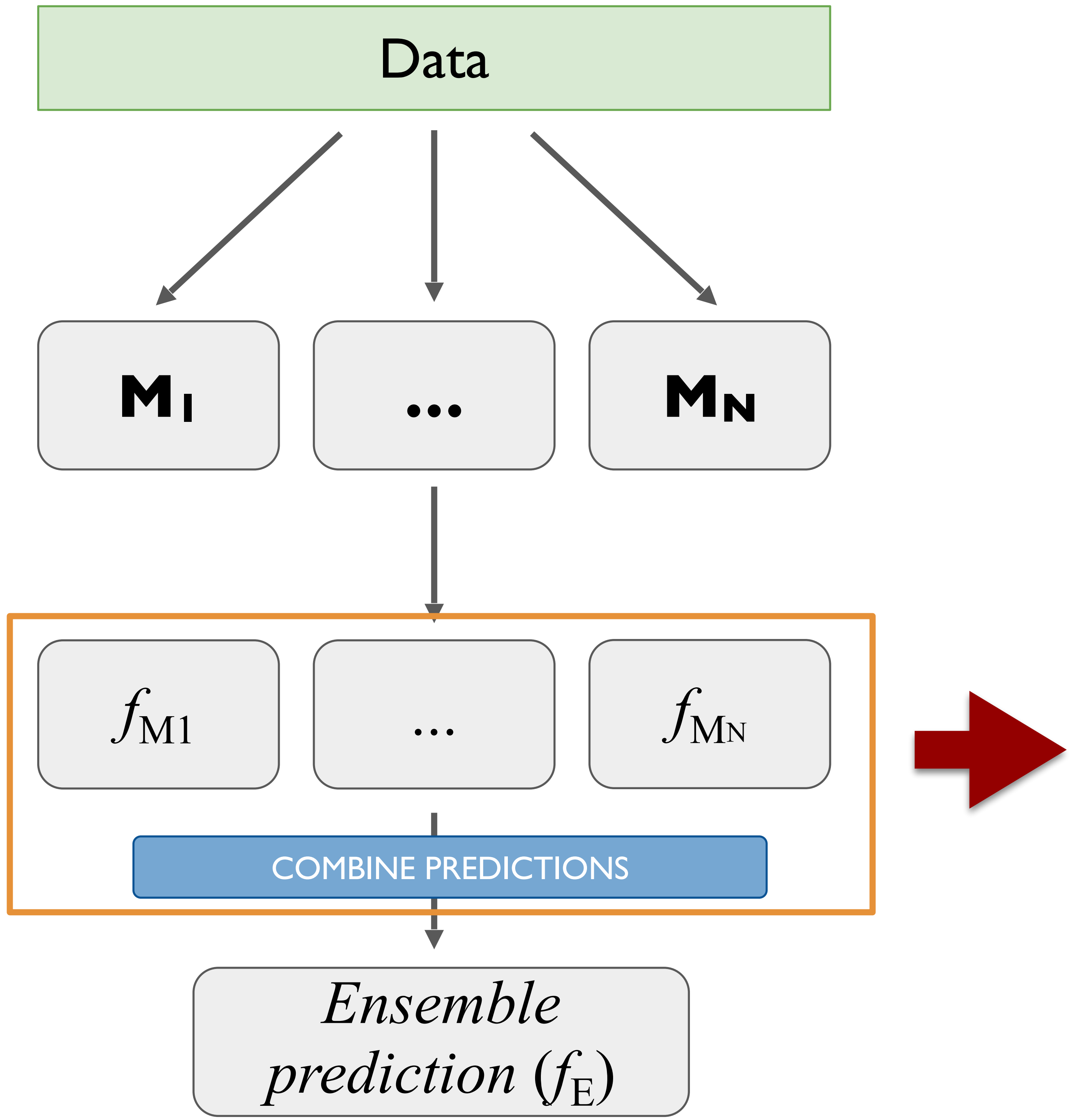


Source: Polikar, 2006

Multiple Models in ML



Multiple Models in ML



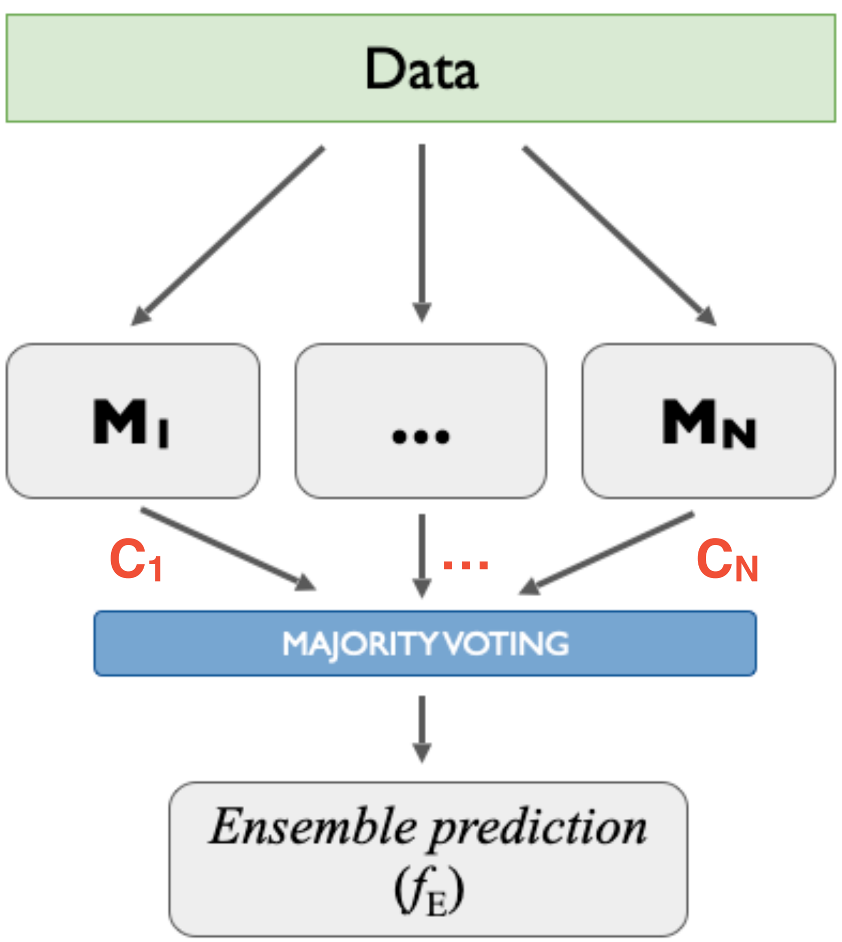
How to combine
the models' predictions?

Slide from Bruno Castro da Silva

Combining Models' Predictions

Voting methods vs. Averaging methods

Based on the type of data being predicted
discrete values (or classes) vs. continuous values



- Each classifier predicts a label/class ($C_1, C_2, \dots, C_N$)
- **Uniform, majority voting**: predictions made by all classifiers contribute equally to the final decision
- Final prediction: the class with the most votes
- Let $d_{k,j} = 1$ if the k-*th* classifier predicted class C_j (and $d_{k,j} = 0$ otherwise)

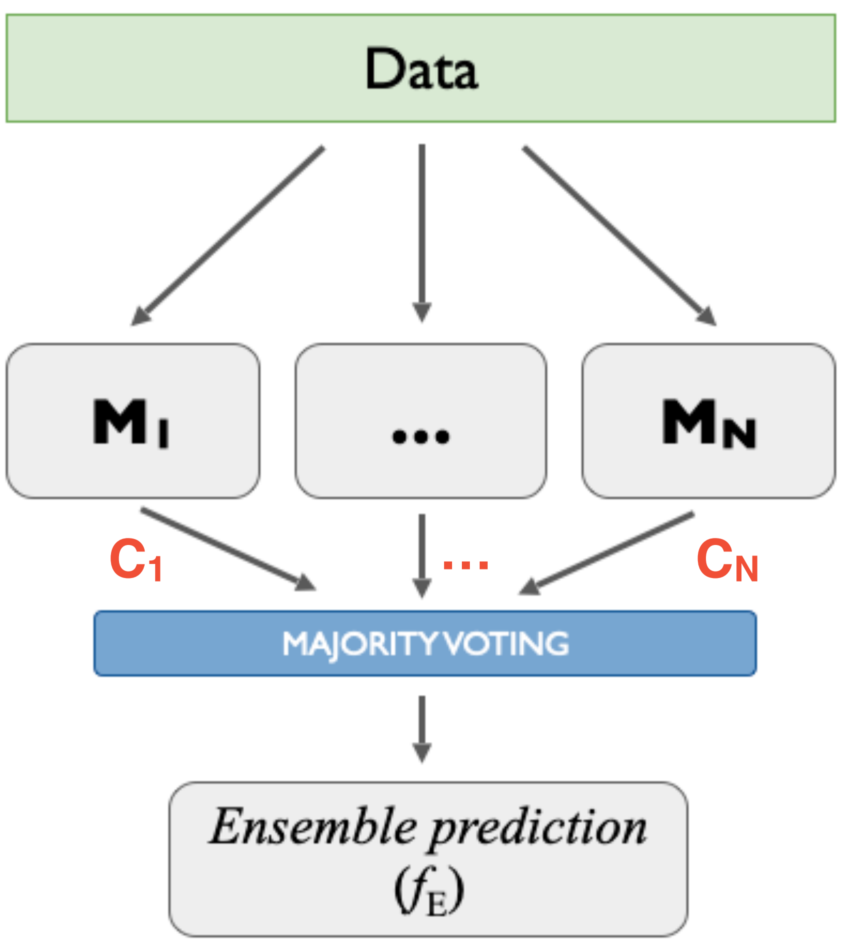
$$\text{Final predicted class} = \max_{j=\{1,\dots,C\}} \sum_{k=1}^{N} d_{k,j}$$

Slide from Bruno Castro da Silva

Combining Models' Predictions

Voting methods vs. Averaging methods

Based on the type of data being predicted
discrete values (or classes) vs. continuous values



- Each classifier predicts a label/class ($C_1, C_2, \ldots, C_N$)
- **Weighted, majority voting**: weights can assigned to each classifier's predictions if some classifiers are "better" than others
- Weights w_k, for each k-th classifier, can be computed by evaluating the classifier's performance on training/validation datasets
- Let $d_{k,j} = 1$ if the k-*th* classifier predicted class C_j (and $d_{k,j} = 0$ otherwise)

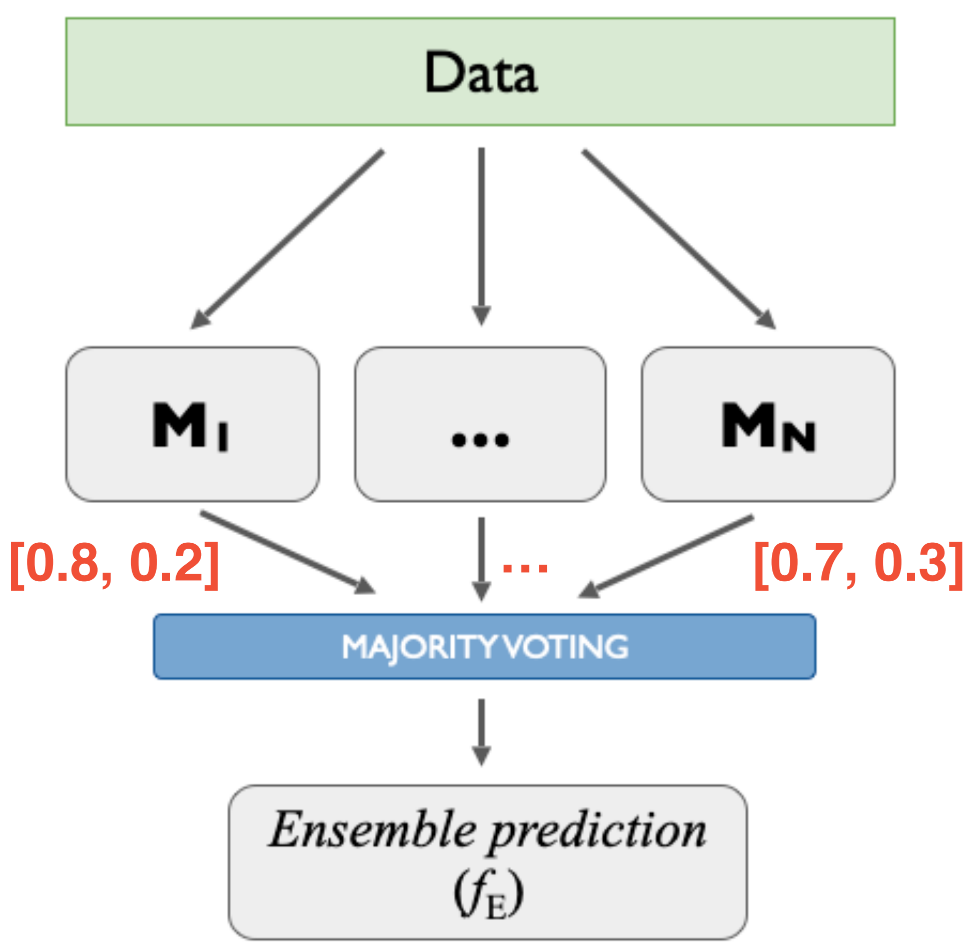
Final predicted class $= \max_{j=\{1,\ldots,C\}} \sum_{k=1}^{N} w_k \, d_{k,j}$

Slide from Bruno Castro da Silva

Combining Models' Predictions

Voting methods vs. Averaging methods

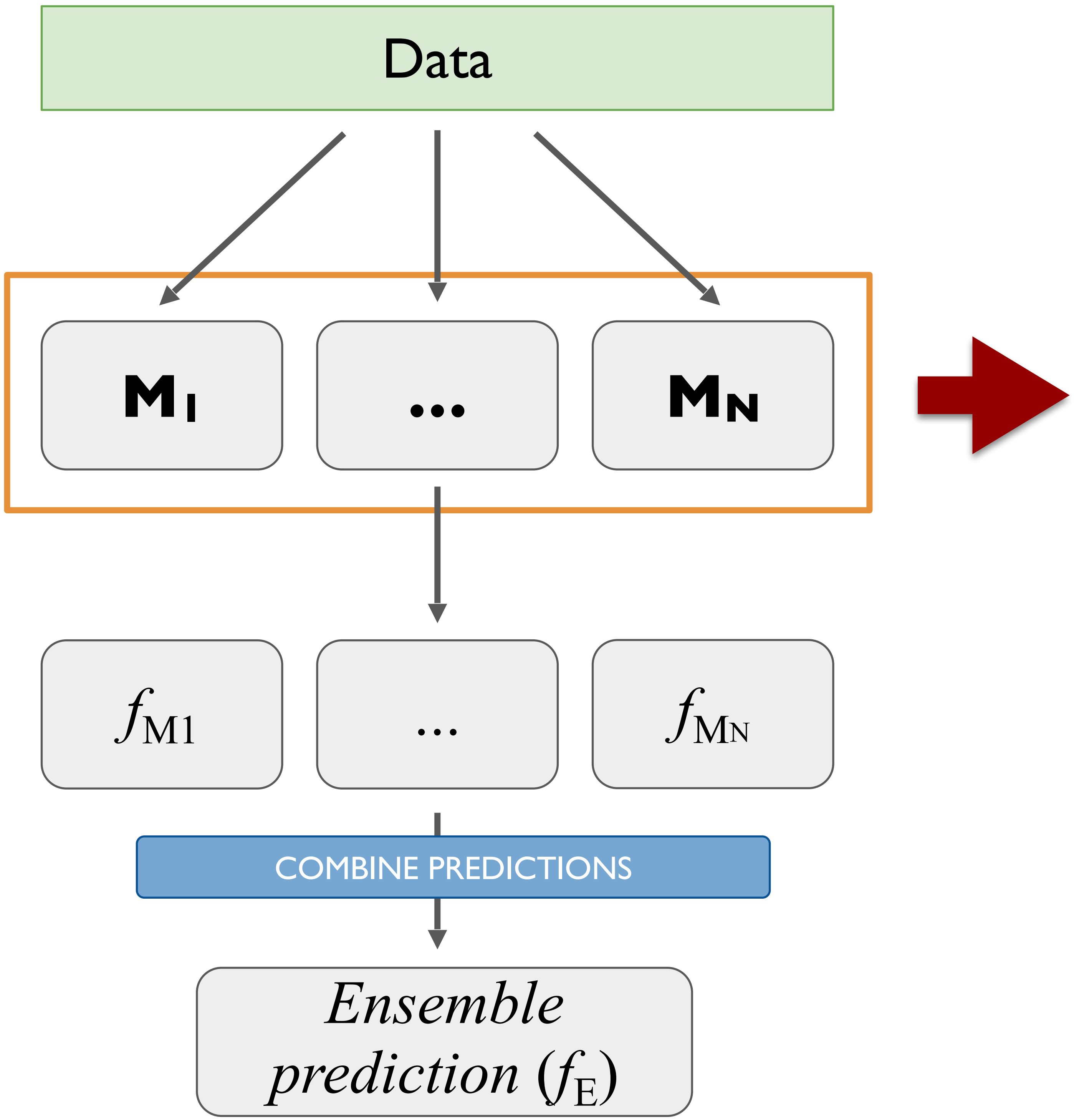
Based on the type of data being predicted
discrete values (or classes) vs. <u>continuous values</u>



- Each classifier outputs the probability that the input belongs to each possible class (Pr[1], Pr[2], … , Pr[N])
- **Different ways of combining such predictions:**
  - e.g., average, product, maximum, minimum, median, etc.
- Final predicted class:
  - the one the maximizes the quantity as defined above

Slide from Bruno Castro da Silva

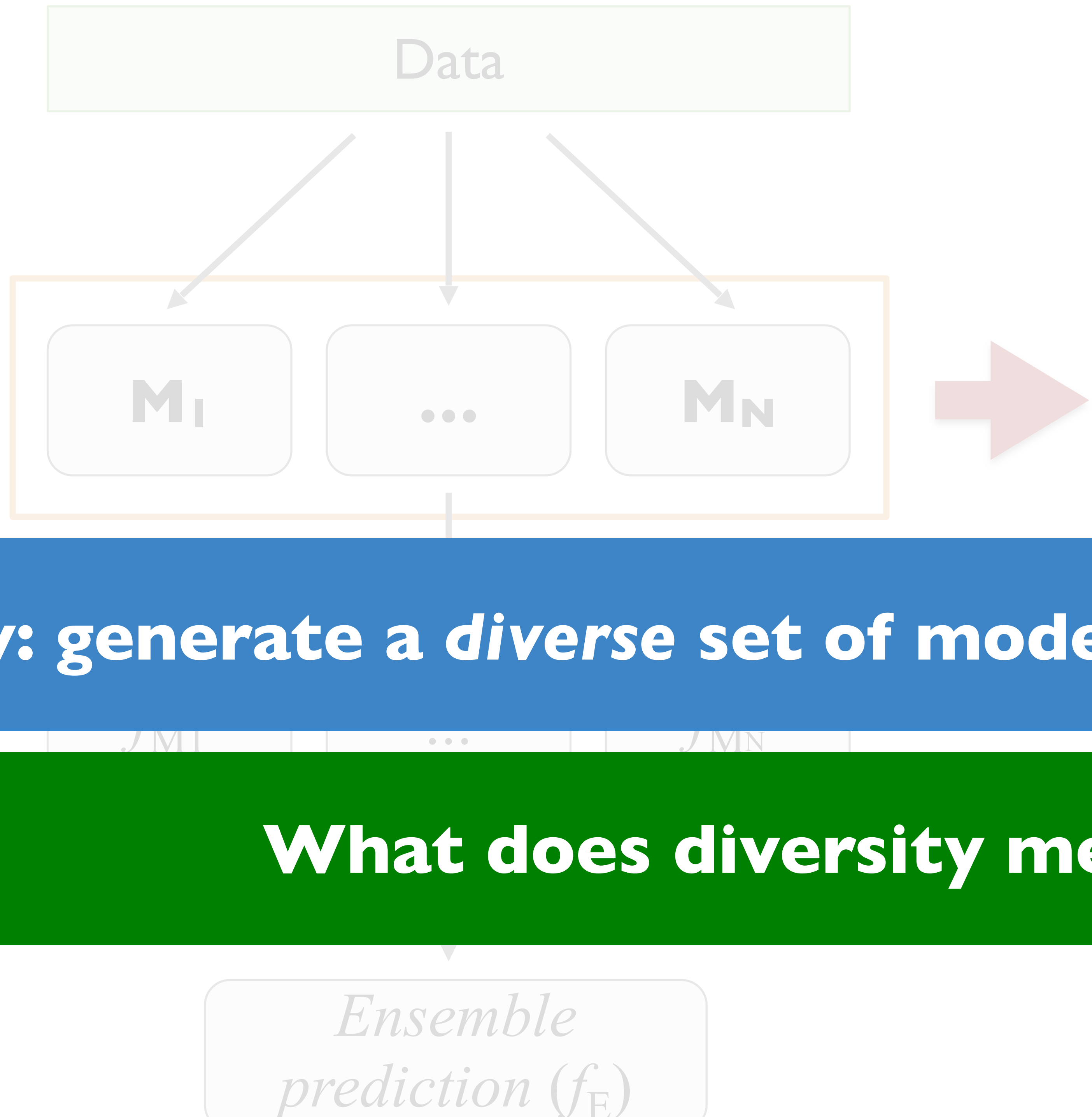
Multiple Models in ML



How to generate/train multiple models?

Slide from Bruno Castro da Silva

Multiple Models in ML

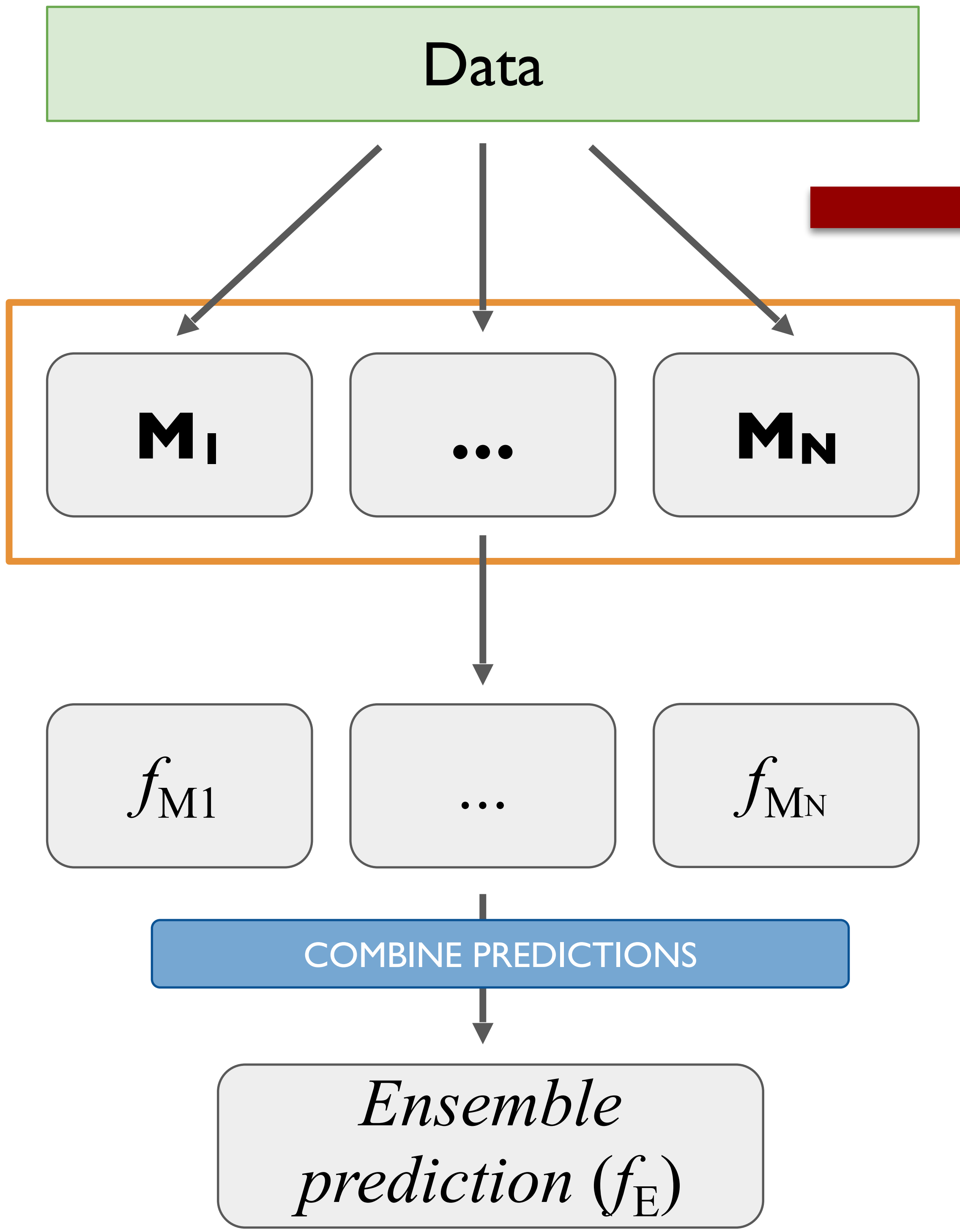


How to generate/train multiple models?

Ideally: generate a *diverse* set of models in the *ensemble*

What does diversity mean?

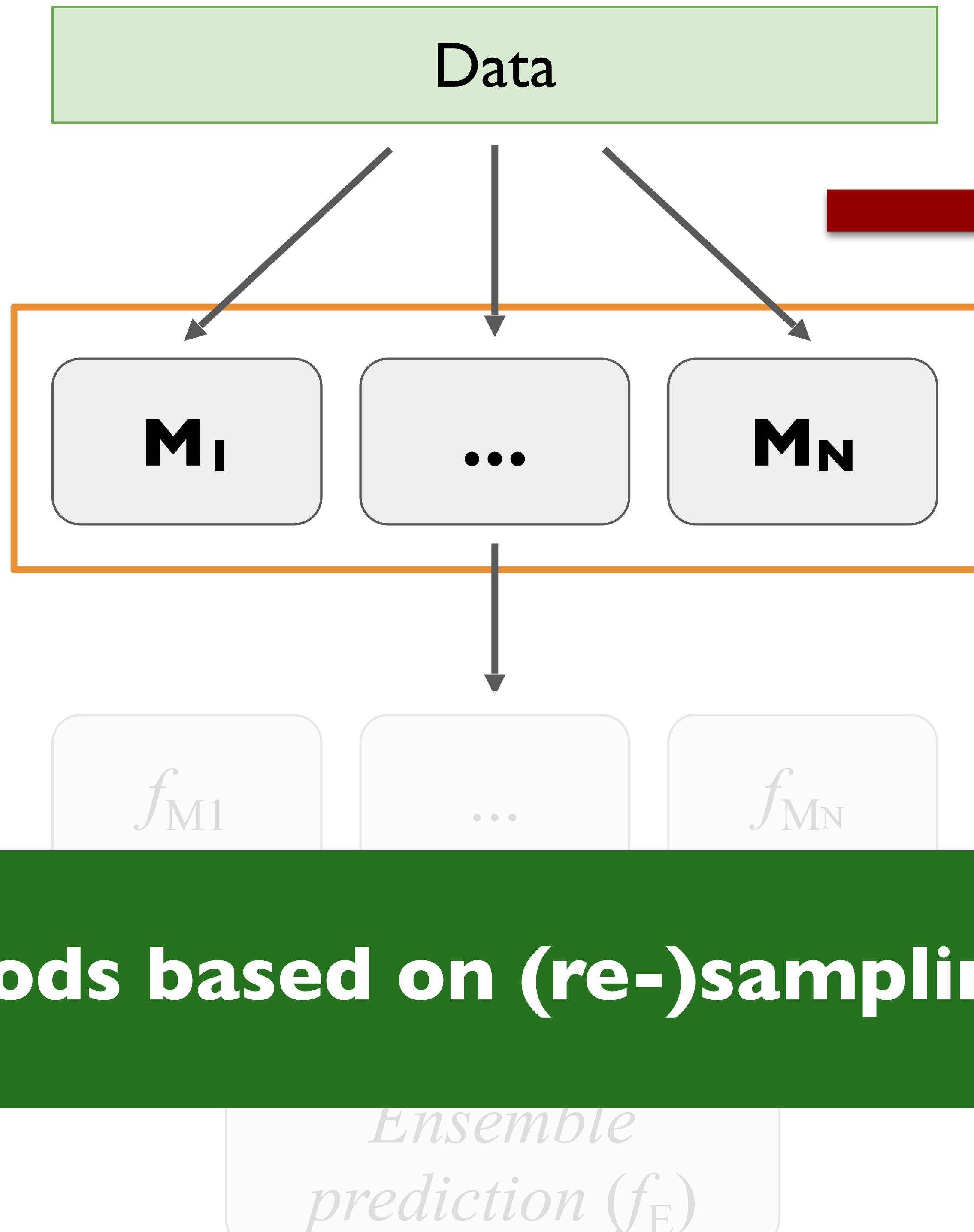
Multiple Models in ML



Should use different datasets to train each model

- But all models are homogeneous
  - e.g., all of them are neural networks;
- In this first approach
  - we do not mix different types of algorithms/machine learning models

Multiple Models in ML



Should use different datasets to train each model

- But all models are homogeneous
  - e.g., all of them are neural networks;
- In this first approach
  - we do not mix different types of algorithms/machine learning models

Methods based on (re-)sampling from the training set

Bagging (a.k.a., Bootstrap Aggregating)

- Diversity: achieved by generating many "synthetic" datasets based on the original training set
  - Each new "synthetic" dataset is known as a ***bootstrap*** dataset
  - Bootstrap datasets are created by sampling (with replacement) from the original/complete dataset
  - If the original dataset has size *N*, each bootstrap dataset should also have size *N*

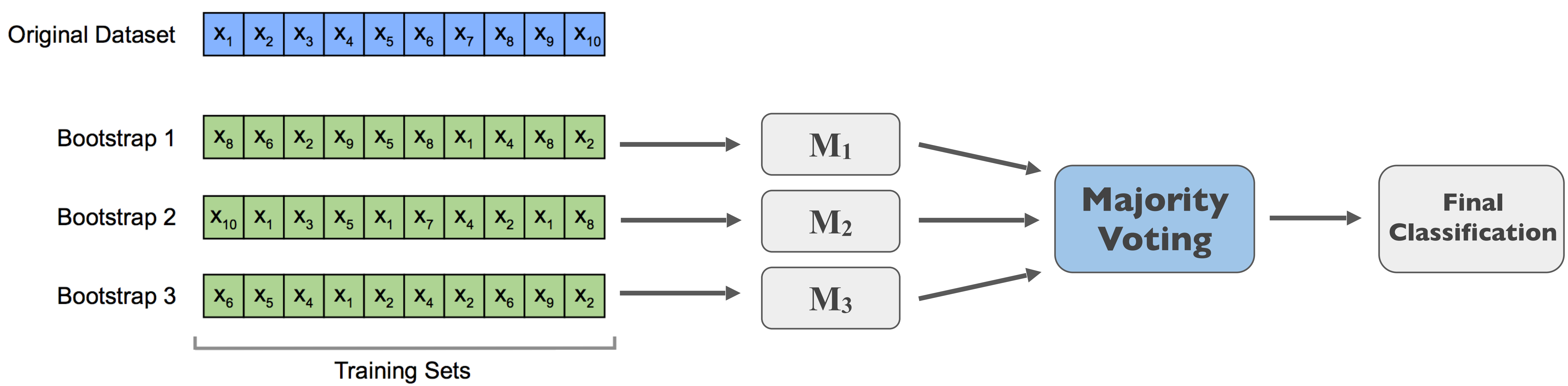
Out-of-bag instances
(~ ⅓)

Bagging (a.k.a., Bootstrap Aggregating)

- Diversity: achieved by generating many "synthetic" datasets based on the original training set
  - Each new "synthetic" dataset is known as a ***bootstrap*** dataset
  - Bootstrap datasets are created by sampling (with replacement) from the original/complete dataset
  - If the original dataset has size *N*, each bootstrap dataset should also have size *N*

  - One classifier (same algorithm, same hyper-parameters) is trained based on each bootstrap

  - To classify new instances, perform majority voting considering all models in the ensemble



Slide from Bruno Castro da Silva

Bagging (a.k.a., Bootstrap Aggregating)

Input:

- Training data S with correct labels $\omega_i \in \Omega = \{\omega_1,\ldots,\omega_C\}$ representing C classes
- Weak learning algorithm **WeakLearn,**
- Integer T specifying number of iterations.
- Percent (or fraction) F to create bootstrapped training data

Do $t = 1, \ldots, T$

1. Take a bootstrapped replica S_t by randomly drawing F percent of S.
2. Call **WeakLearn** with S_t and receive the hypothesis (classifier) h_t.
3. Add h_t to the ensemble, $\mathtt{E}$.

End

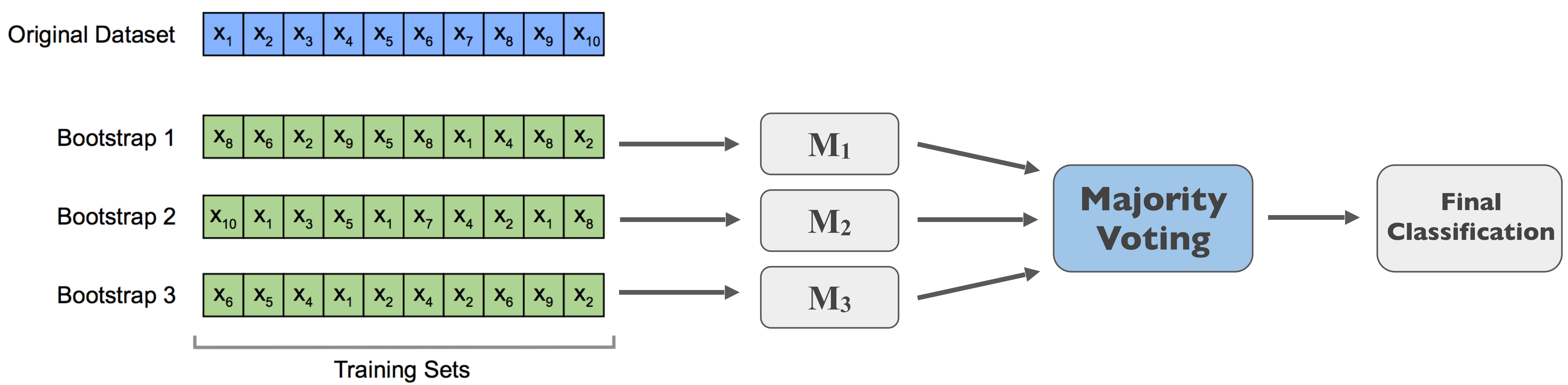
Test: Simple Majority Voting – Given unlabeled instance $\mathbf{x}$

1. Evaluate the ensemble $\mathtt{E} = \{h_1, \ldots, h_T\}$ on $\mathbf{x}$.
2. Let $$v_{t,j} = \begin{cases} 1, & \text{if } h_t \text{ picks class } \omega_j \\ 0, & \text{otherwise} \end{cases} \tag{8}$$
   be the vote given to class ω_j by classifier h_t.
3. Obtain total vote received by each class
   $$V_j = \sum_{t=1}^{T} v_{t,j}, \quad j = 1,\ldots,C \tag{9}$$
4. Choose the class that receives the highest total vote as the final classification.

Source: Polikar, 2006

Bagging (a.k.a., Bootstrap Aggregating)

- Usually performs well when we have small datasets
- Bagging helps decrease the *variance* of the resulting classifier, by computing the "average" output of multiple models
- Does not help decrease the *bias* of the algorithm, though:
  - if each individual model in the ensemble is strongly biased towards a "wrong way of explaining the training set"
  - combining their outputs will not help…



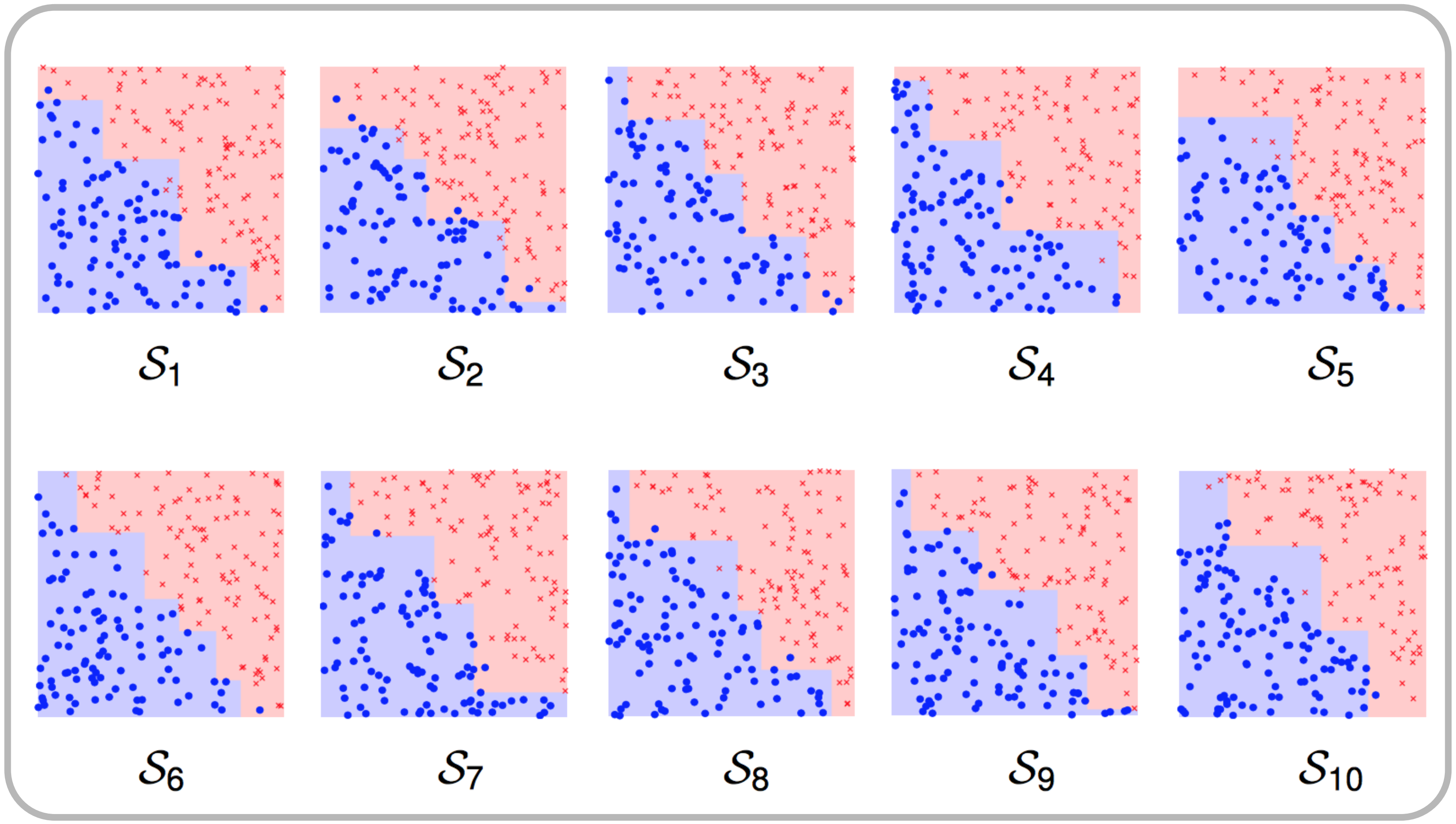
Slide from Bruno Castro da Silva

Bagging (a.k.a., Bootstrap Aggregating)

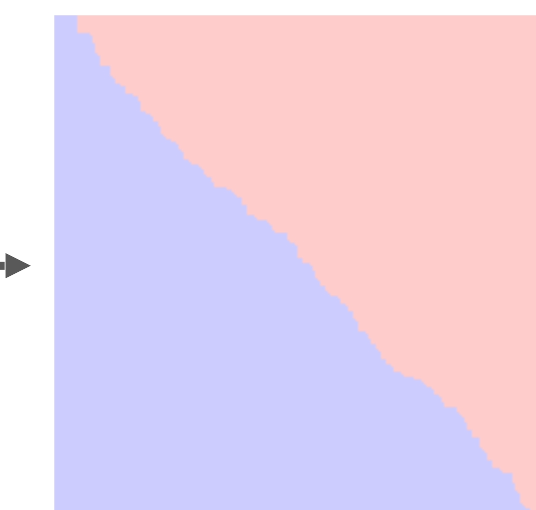
- How would Bagging work if combined with the Decision Tree algorithm?
  - Train multiple decision trees
  - Each one based on a different bootstrap dataset (e.g., bootstrap datasets $\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_{10}$)
  - Because data given to each tree is slightly different…
    - the trees will be slightly different (e.g., will perform different tests/attribute splits)
  - Each tree will thus result in a slightly different decision boundary

Bagging (a.k.a., Bootstrap Aggregating)

- How would Bagging work if combined with the Decision Tree algorithm?
  - Train multiple decision trees
  - Each one based on a different bootstrap dataset (e.g., bootstrap datasets $\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_{10}$)
  - Because data given to each tree is slightly different…
    - the trees will be slightly different (e.g., will perform different tests/attribute splits)
  - Each tree will thus result in a slightly different decision boundary

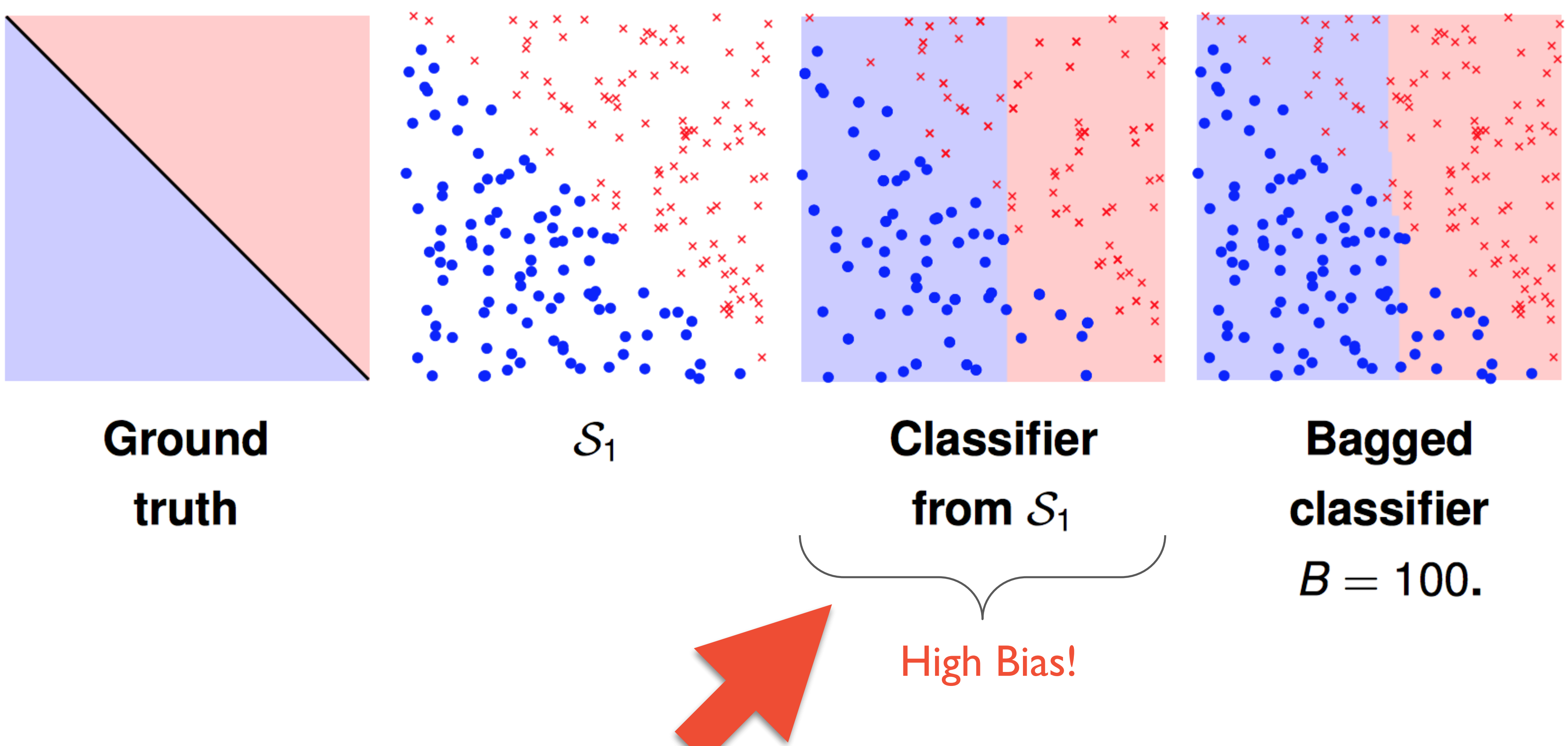


"Average" decision boundary of the ensemble



Bagging (a.k.a., Bootstrap Aggregating)

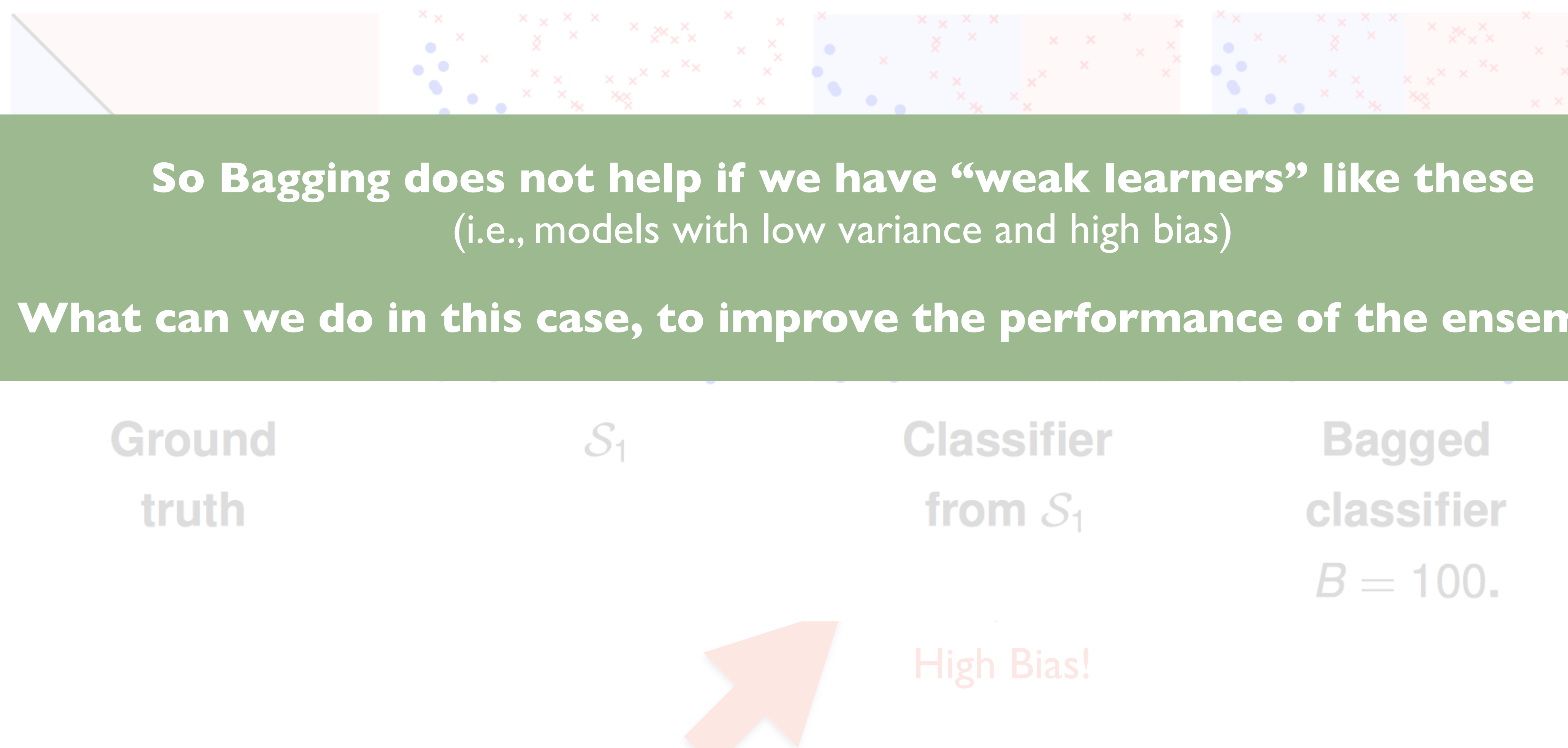
- As previously mentioned:
  - Bagging helps decrease the *variance* of the resulting classifier, by computing the "average" output of multiple models
  - But it does not help decrease the *bias* of the algorithm:
  - If each individual model in the ensemble is strongly biased towards a "wrong way of explaining the training set"
    - then combining their outputs will not help…



Source: Sullivan, Maki, 2013

Bagging (a.k.a., Bootstrap Aggregating)

- As previously mentioned:
  - Bagging helps decrease the *variance* of the resulting classifier, by computing the “average" output of multiple models
  - But it does not help decrease the *bias* of the algorithm:
  - If each individual model in the ensemble is strongly biased towards a “wrong way of explaining the training set”
    - then combining their outputs will not help…



So Bagging does not help if we have “weak learners” like these
(i.e., models with low variance and high bias)

What can we do in this case, to improve the performance of the ensemble?

Boosting

- As previously mentioned:
  - Bagging helps decrease the *variance* of the resulting classifier, by computing the "average" output of multiple models
  - But it does not help decrease the *bias* of the algorithm:
  - If each individual model in the ensemble is strongly biased towards a "wrong way of explaining the training set"
    - then combining their outputs will not help…

So Bagging does not help if we have "weak learners" like these
(i.e., models with low variance and high bias)

What can we do in this case, to improve the performance of the ensemble?

But before we talk about Boosting….

Let us present a widely-used ML algorithm based on Bagging

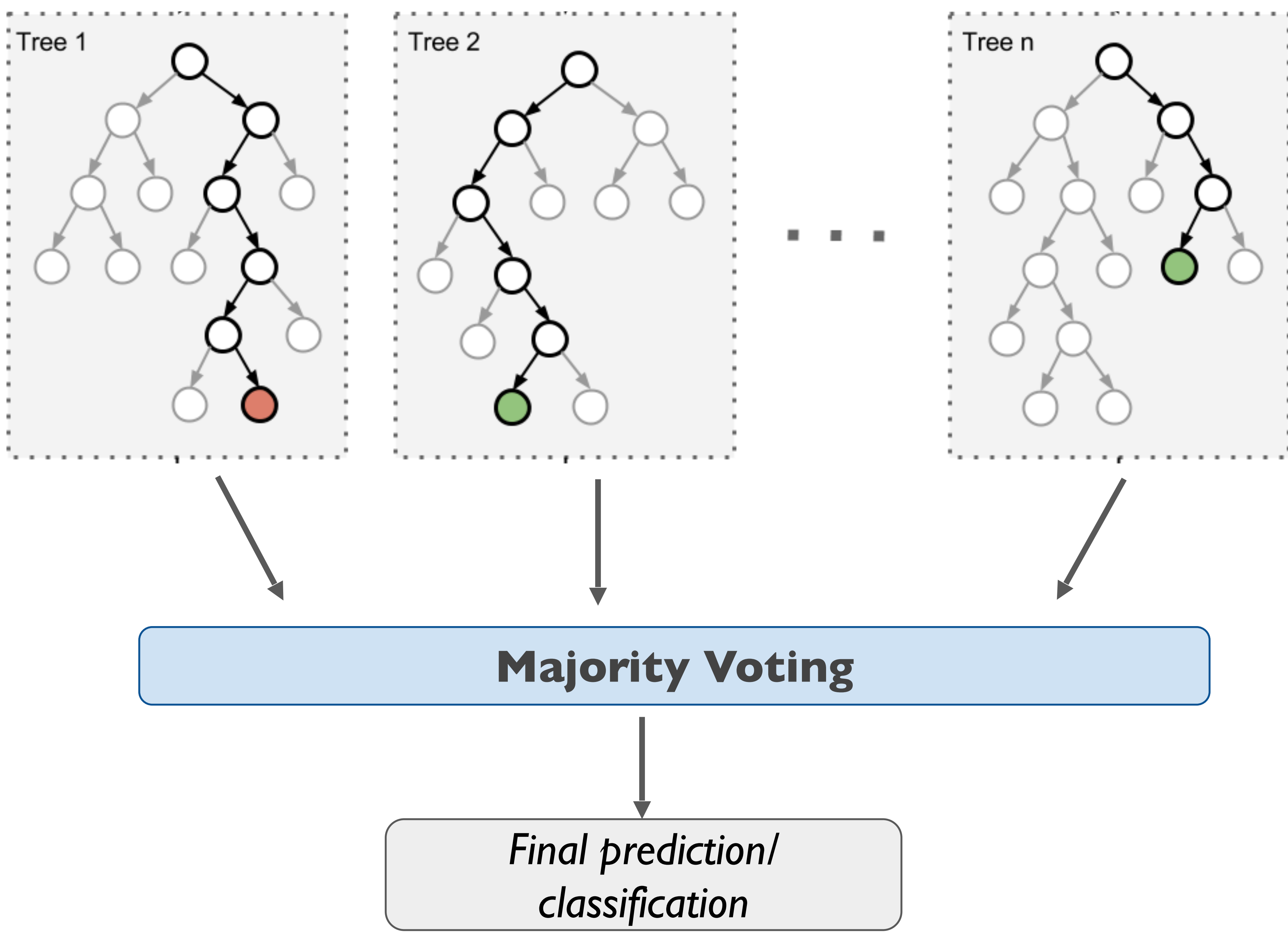
Random Forests

Slide from Bruno Castro da Silva

Random Forests

Random Forests

- Trains many (slightly different) Decision Trees
- Their predictions are combined via majority voting

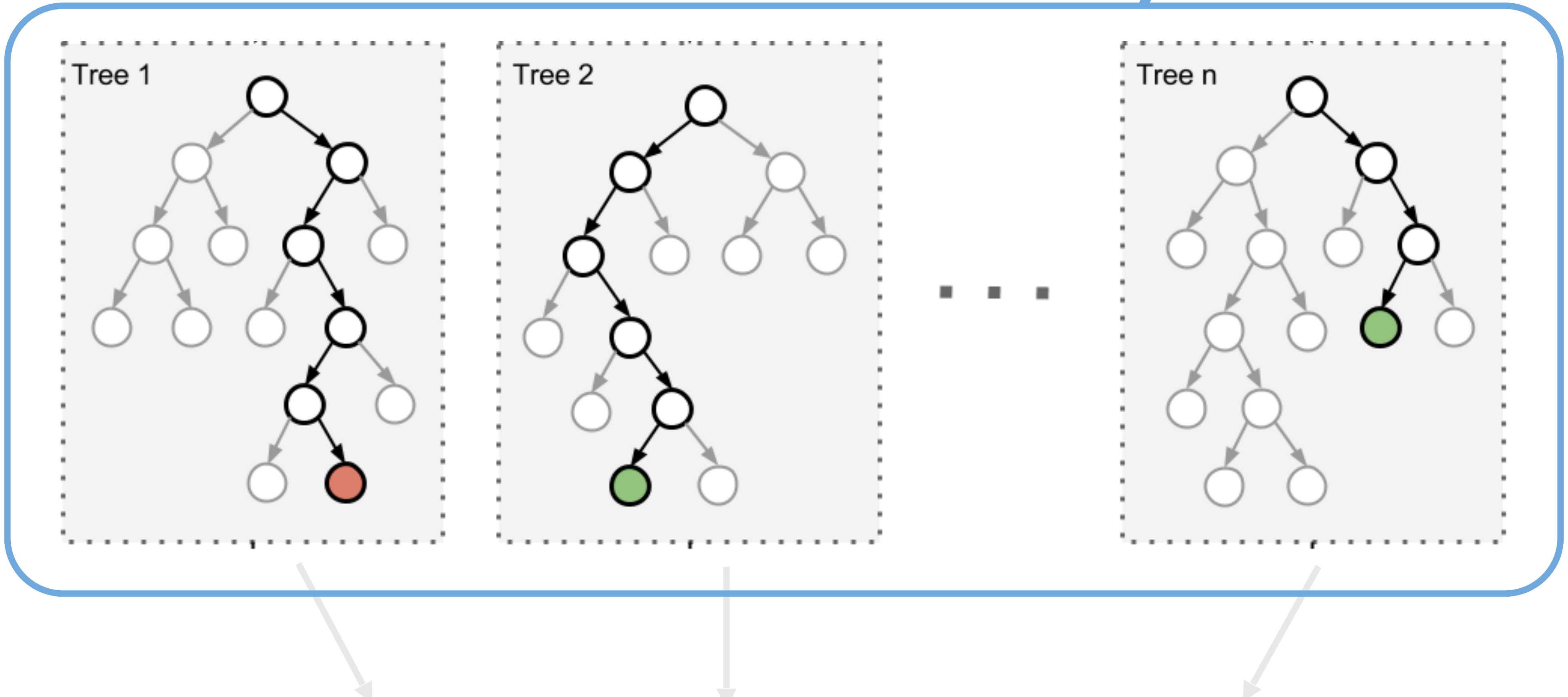


Slide from Bruno Castro da Silva

Random Forests

- Trains many (slightly different) Decision Trees
- Their predictions are combined via majority voting

How to ensure diversity?



Bagging, combined with selection of random attributes

Final prediction/ classification

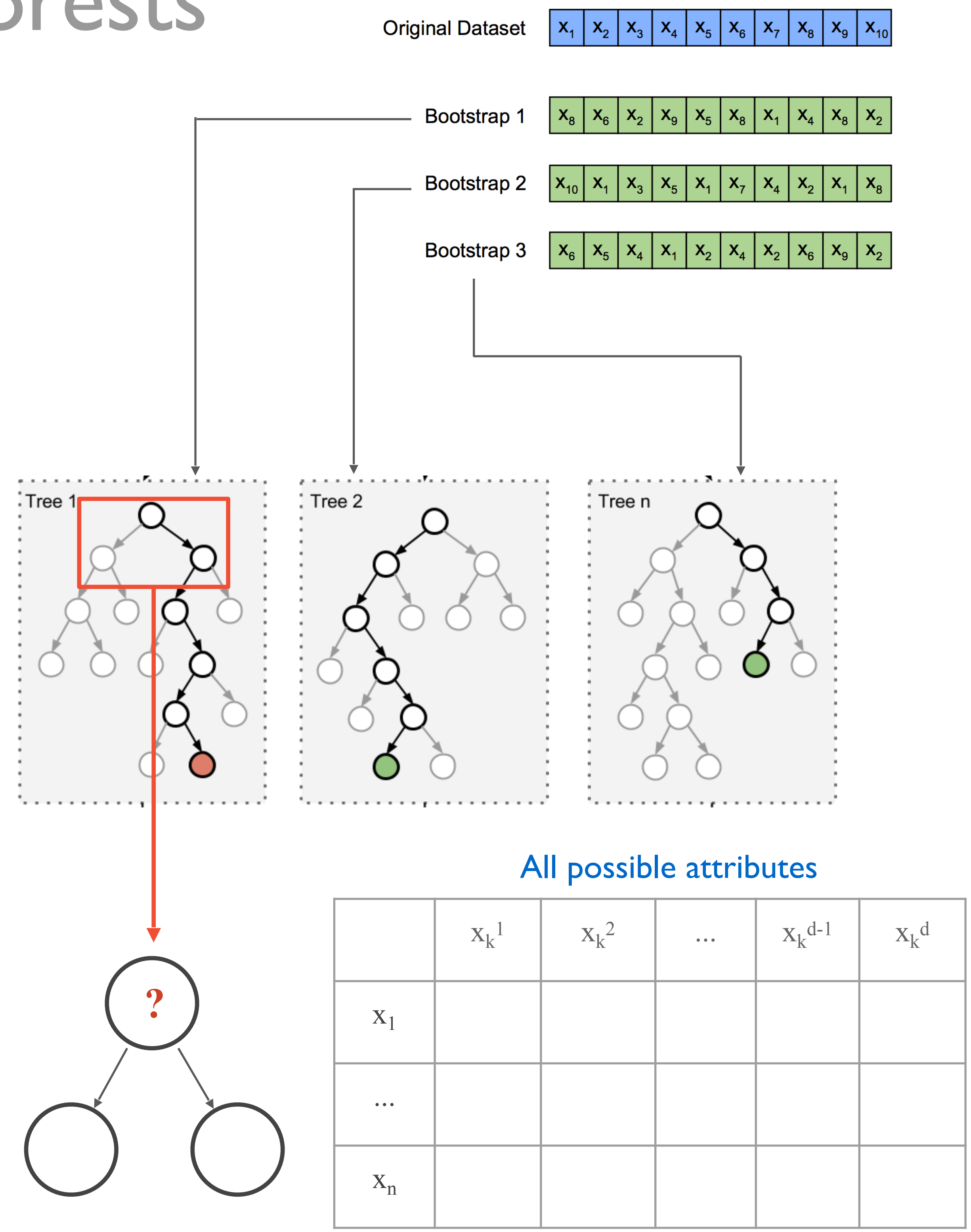
Slide from Bruno Castro da Silva

Random Forests

- **Bagging + selection of random attributes**

- **Bagging**
- Each tree is trained with a different bootstrap dataset
  - Sampling (with replacement) from original dataset

- **Selection of random attributes**
- Whenever splitting a node
  - Does not check Information Gain of *all* attributes



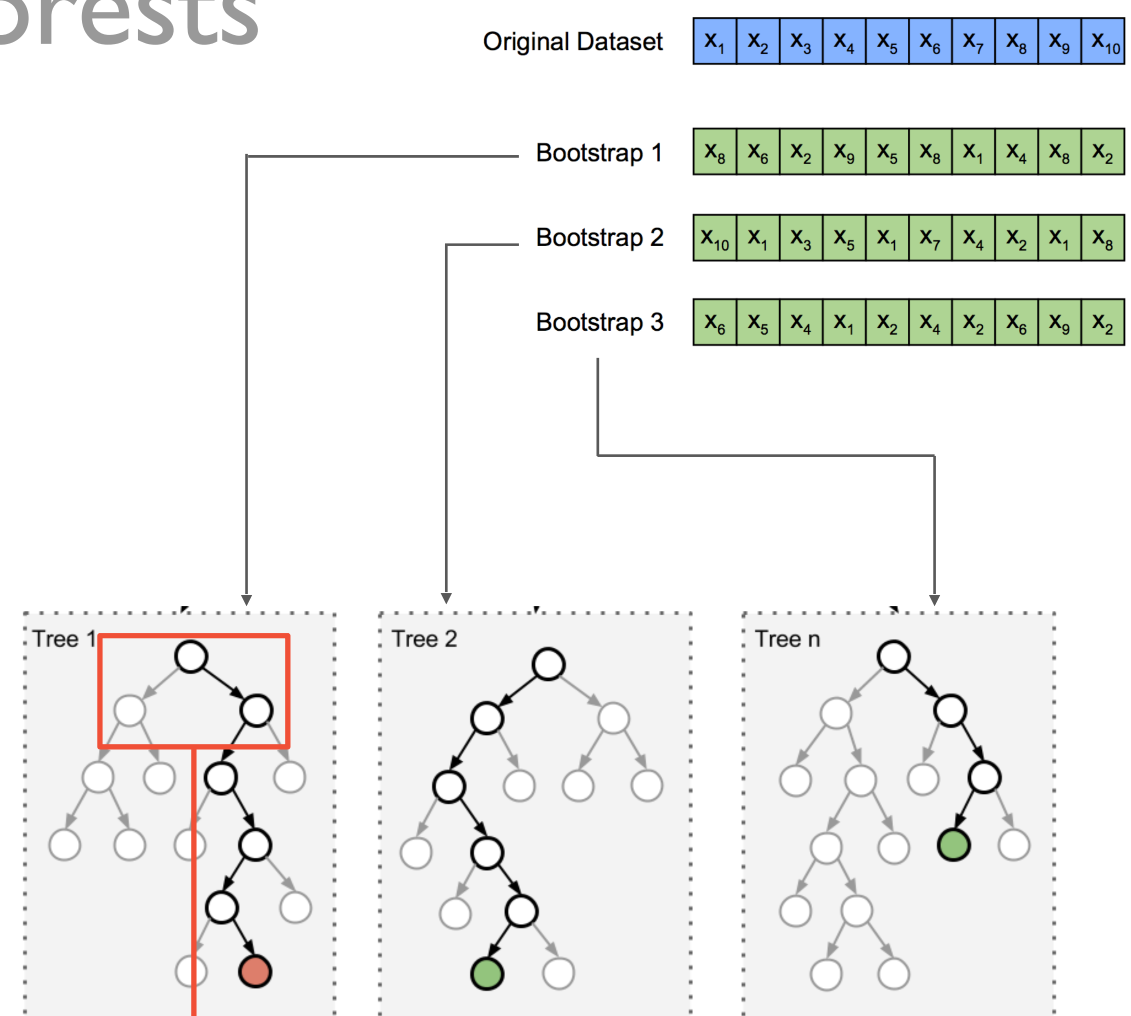
Slide from Bruno Castro da Silva

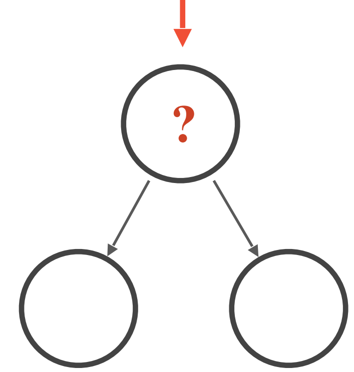
Random Forests

- **Bagging + selection of random attributes**

  - **Bagging**
  - Each tree is trained with a different bootstrap dataset
    - Sampling (with replacement) from original dataset

  - **Selection of random attributes**
  - Whenever splitting a node
    - Does not check Information Gain of *all* attributes
    - Only of a subset of m randomly selected attributes,
      - Out of all possible attributes





All possible attributes

	x_k^1	x_k^2	...	x_k^{d-1}	x_k^d
x_1					
...					
x_n					

Considers only a random subset of m attributes (shown in gray) when deciding how to split a node

Slide from Bruno Castro da Silva

Random Forests

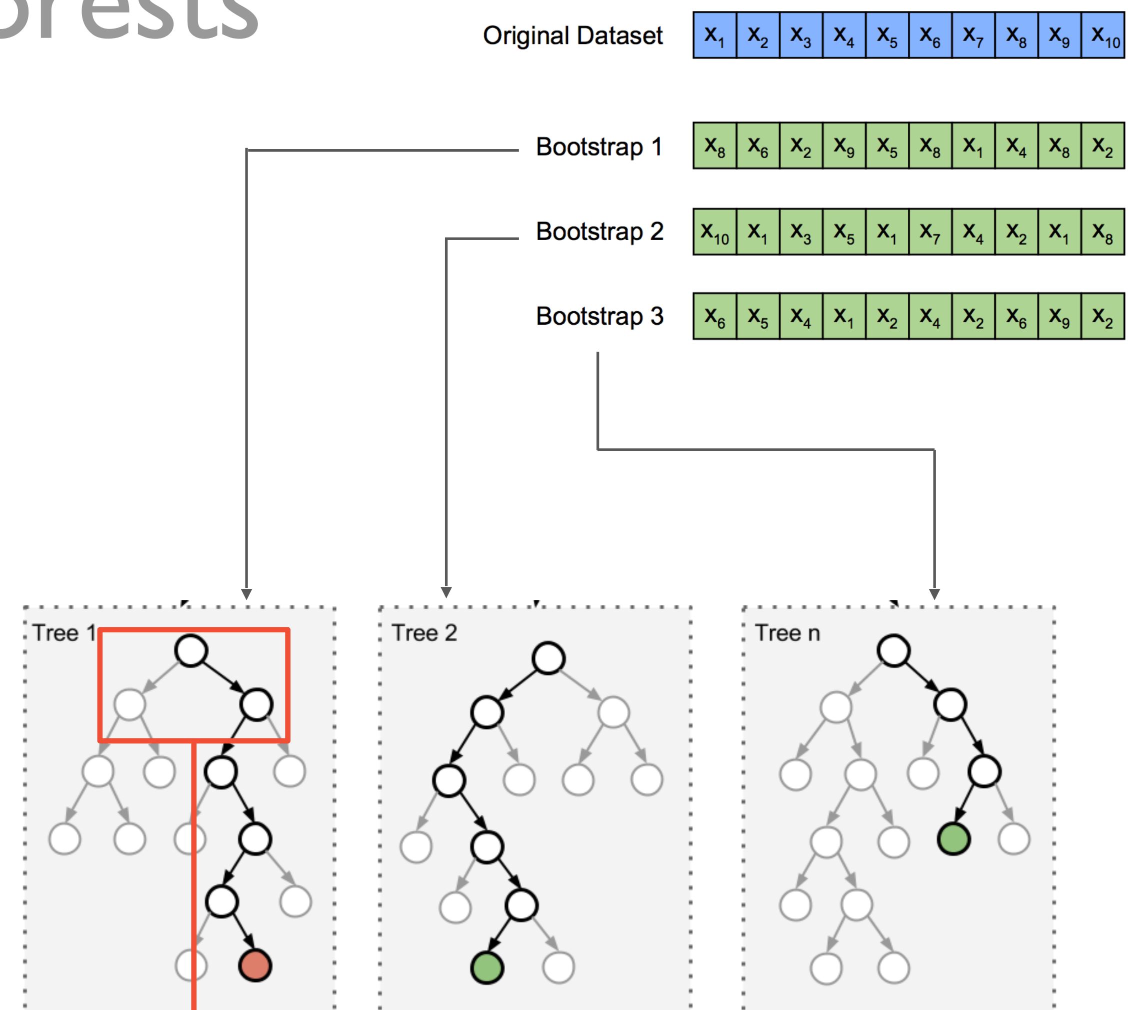
- **Bagging + selection of random attributes**

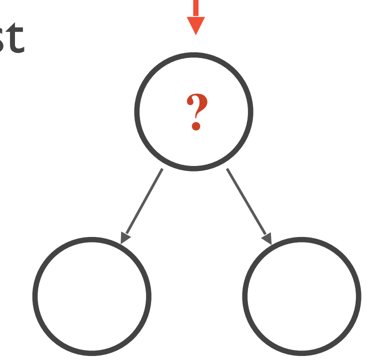
- **Bagging**
- Each tree is trained with a different bootstrap dataset
  - Sampling (with replacement) from original dataset

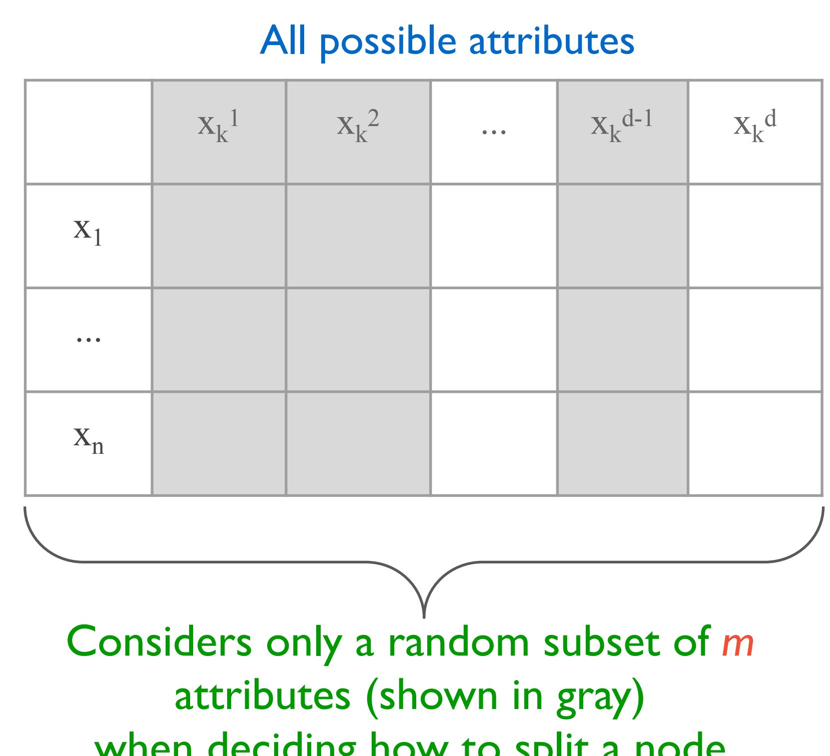
- **Selection of random attributes**
- Whenever splitting a node
  - Does not check Information Gain of *all* attributes
  - Only of a subset of m randomly selected attributes,
    - Out of all possible attributes
    - Splits node based on the attribute with best Information Gain *(or Gini index, etc)*

- **Performing these two sampling procedures**
- Highly diverse ensemble of decision trees

- **Out-of-Bag instances (test instances)**
- Approximately $1/3$ of the data
- Used to evaluate the ensemble







Slide from Bruno Castro da Silva

Random Forests

- **Bagging + selection of random attributes**
  - **Bagging**
  - Each tree is trained with a different bootstrap dataset
    - Sampling (with replacement) from original dataset

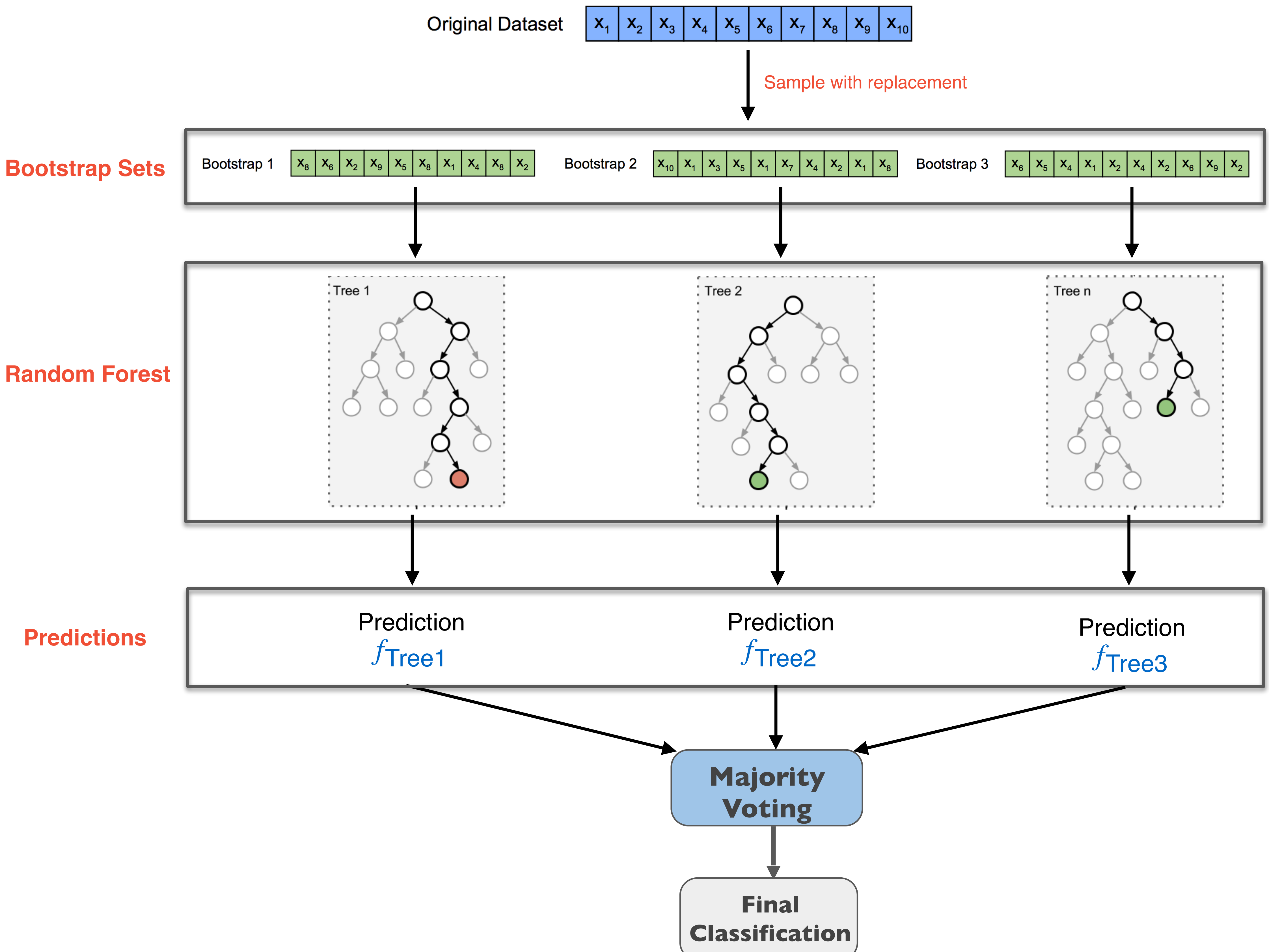
One of the most widely used (supervised) machine learning model nowadays

Often, state-of-the-art in real-life applications

- Out of all possible attributes
- Splits node based on the attribute with best Information Gain *(or Gini index, etc)*

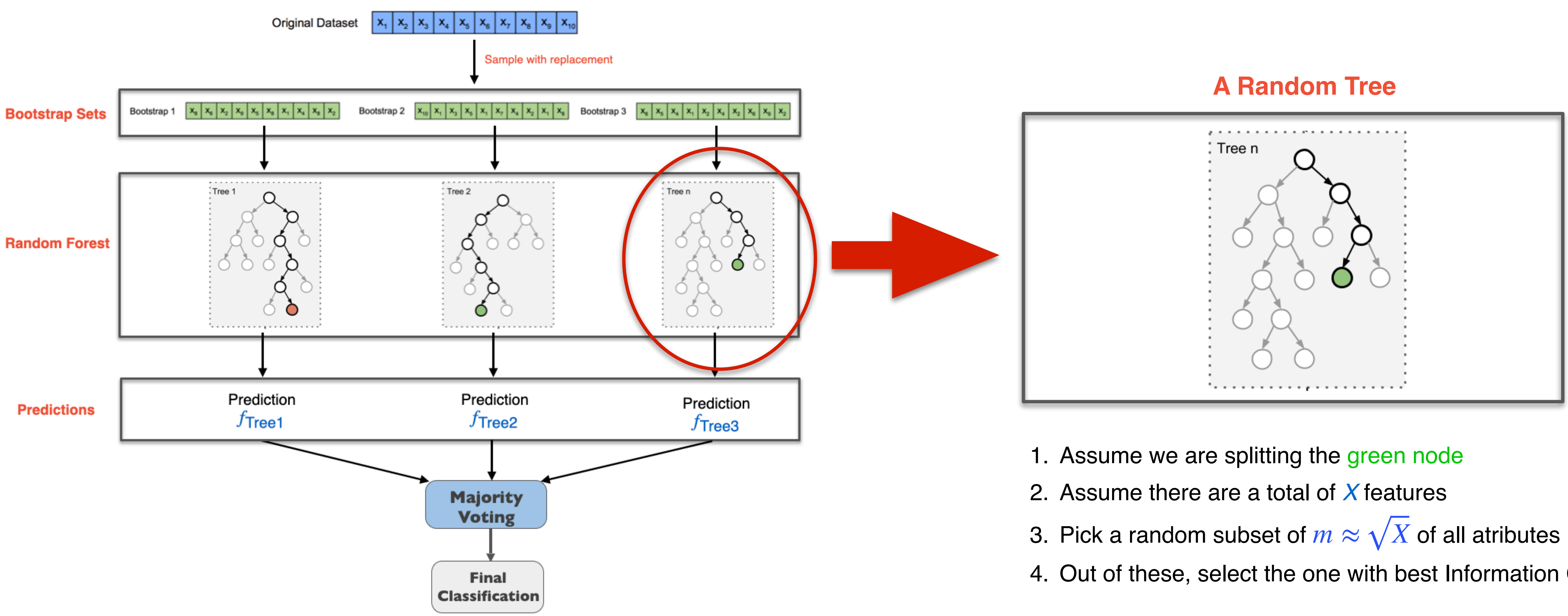
- **Performing these two sampling procedures**
- Highly diverse ensemble of decision trees
- **Out-of-Based instances (test instances)**
- Approximately 1/3 of the data
- Used to evaluate the ensemble

Random Forests



Slide from Bruno Castro da Silva

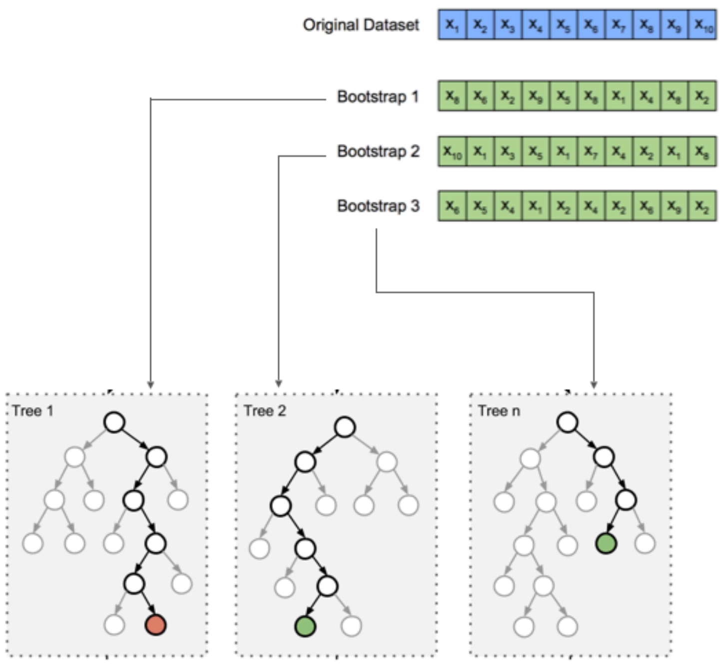
Random Forests



1. Assume we are splitting the green node
2. Assume there are a total of X features
3. Pick a random subset of $m \approx \sqrt{X}$ of all atributes
4. Out of these, select the one with best Information Gain

Slide from Bruno Castro da Silva

Random Forests: Training



1. Let *D* be the original training set
    1. *D* contains *N* training instances, each with *X* attributes

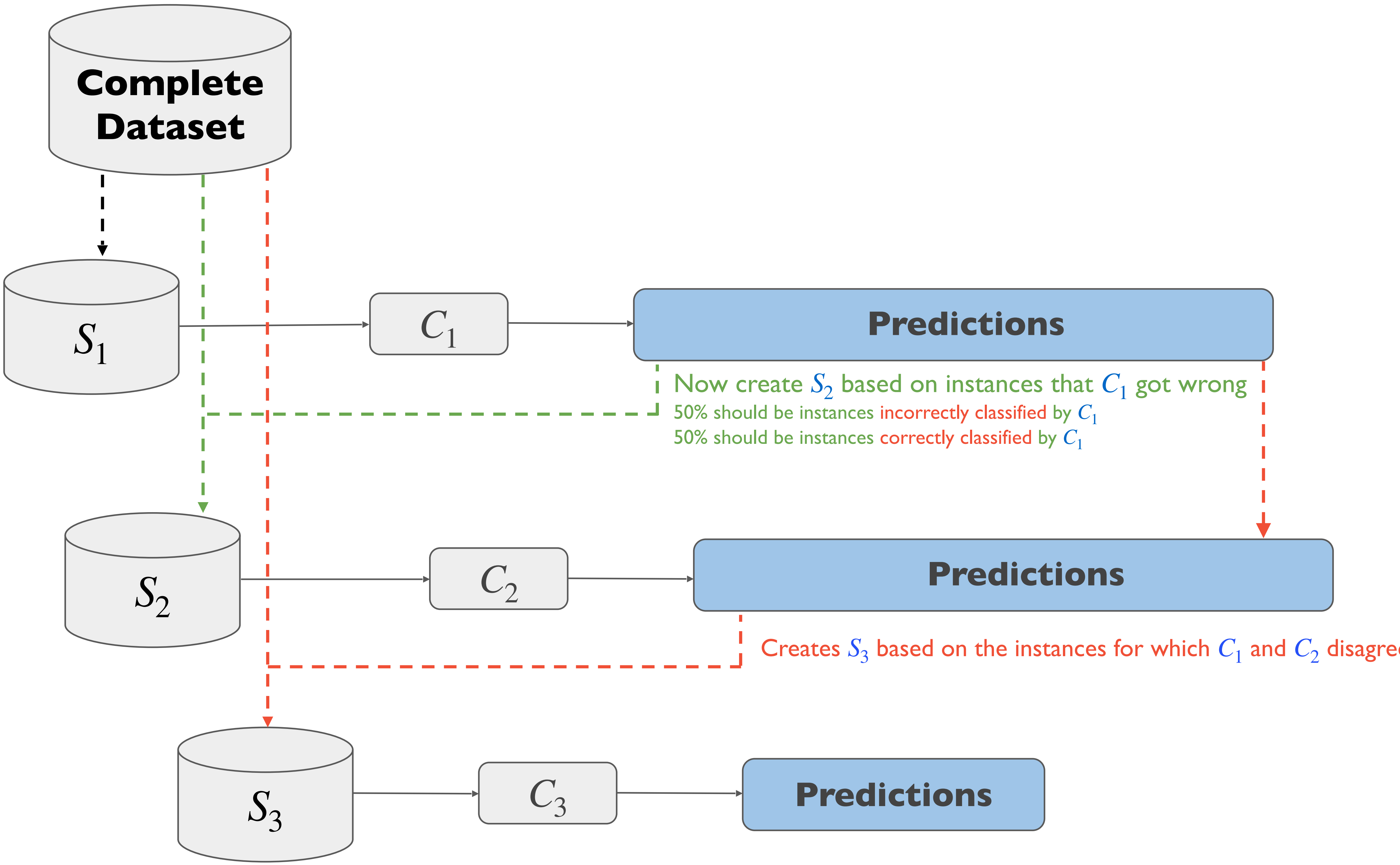
2. For each bootstrap $b = 1, \ldots, B$
    1. Construct a bootstrap dataset of size *N* by sampling from *D* with replacement
    2. Train a decision tree based on this bootstrap by recursively:
        1. Picking a random subset of $m \approx \sqrt{X}$ attributes
        2. Out of these, select the best attribute to split the current node

           (e.g., based to Information Gain)
        3. Add the node to the tree and use it to partition the data into disjoint subsets

3. Return the ensemble of learned trees → the Random Forest

Slide from Bruno Castro da Silva

Boosting: Training



Slide from Bruno Castro da Silva

Boosting

- **Boosting**
- Intuition assuming an ensemble composed of 3 classifiers

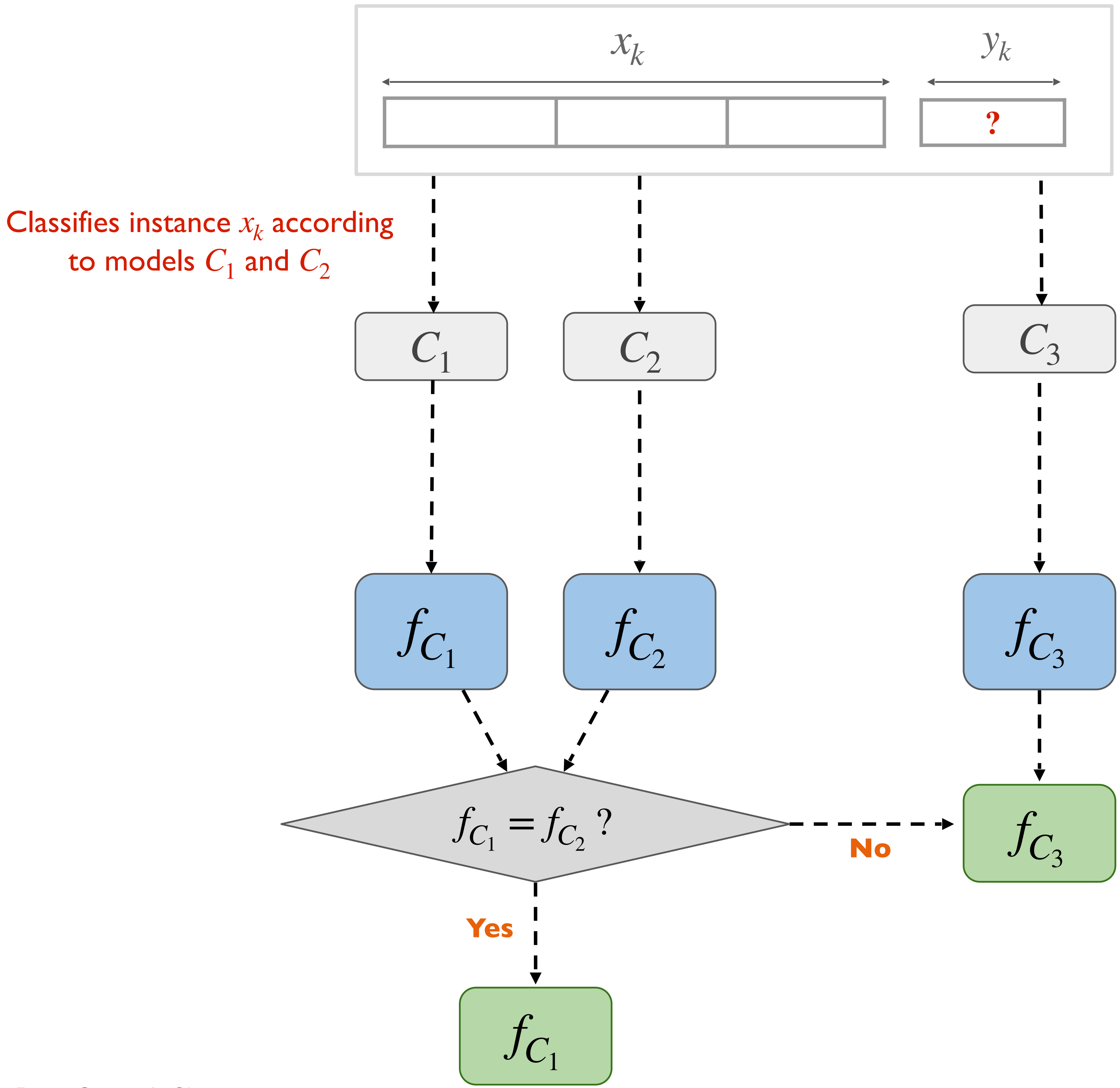
- **Training**
  - Construct a *first* random subset of instances, S_1, of the original dataset
  - Use S_1 to train a weak classifier C_1
  - Construct a *second* random subset of instances, S_2
    - These are the instances that were "challenging" for C_1
      - Half of S_2 will be composed of instances incorrectly classified by C_1
      - Half of S_2 will be composed of instances correctly classified by C_1
  - Use S_2 to train a weak classifier C_2 *(i.e., C_2 will focus on the instances that C_1 got wrong)*
  - Construct a *third* random subset of instances, S_3
    - Composed of all instances for which C_1 and C_2 disagree
  - Use S_3 to train a final weak classifier C_3

Boosting: Classifying New Instances

Intuition

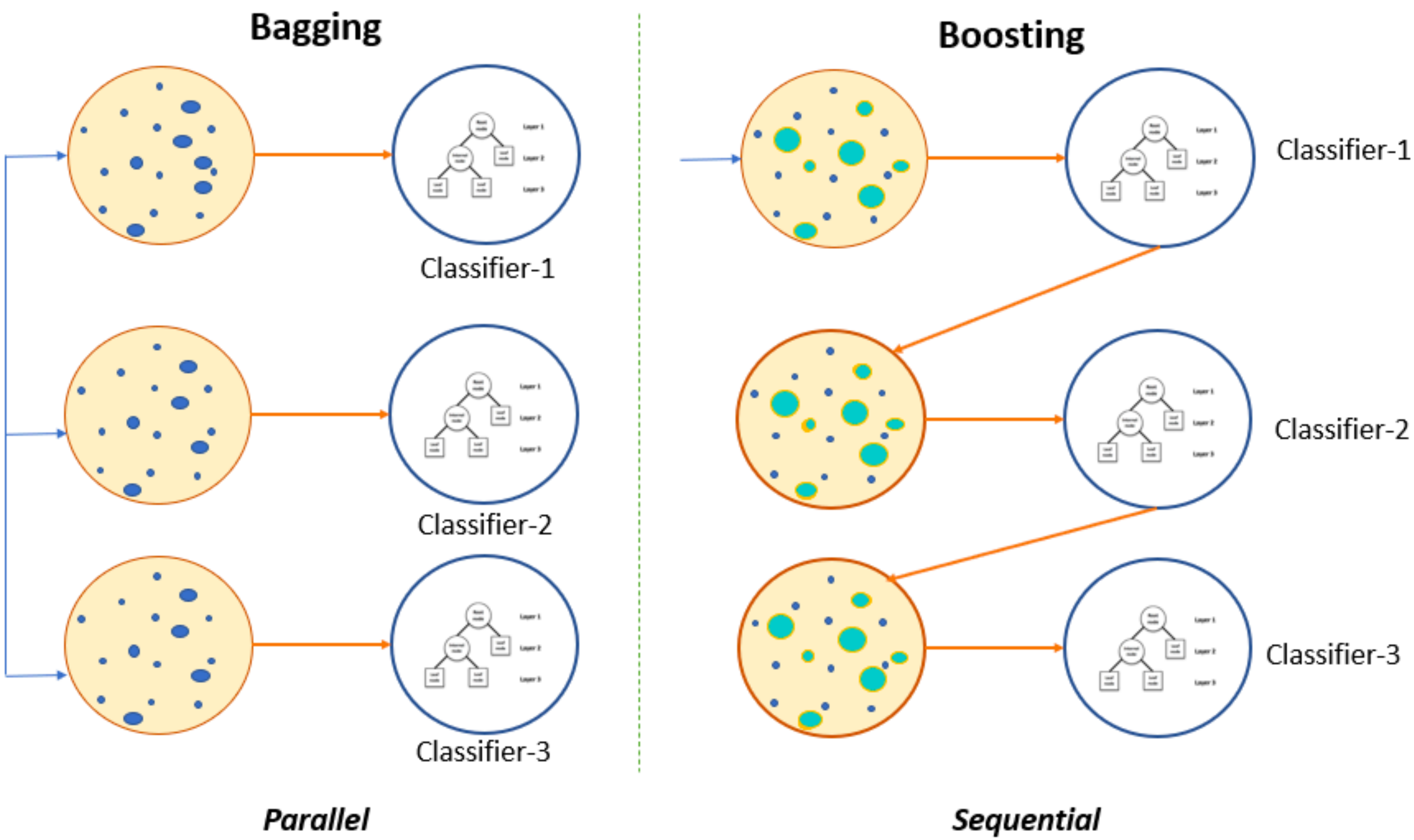
- **When classifying a new instance, check the predicted label according to C_1 and C_2**
- **If C_1 and C_2 agree regarding the predicted class, that will be the final output/prediction made by the ensemble**
- **If C_1 and C_2 disagree, use C_3's prediction as the final output/prediction made by the ensemble**

Boosting: Classifying New Instances



Slide from Bruno Castro da Silva

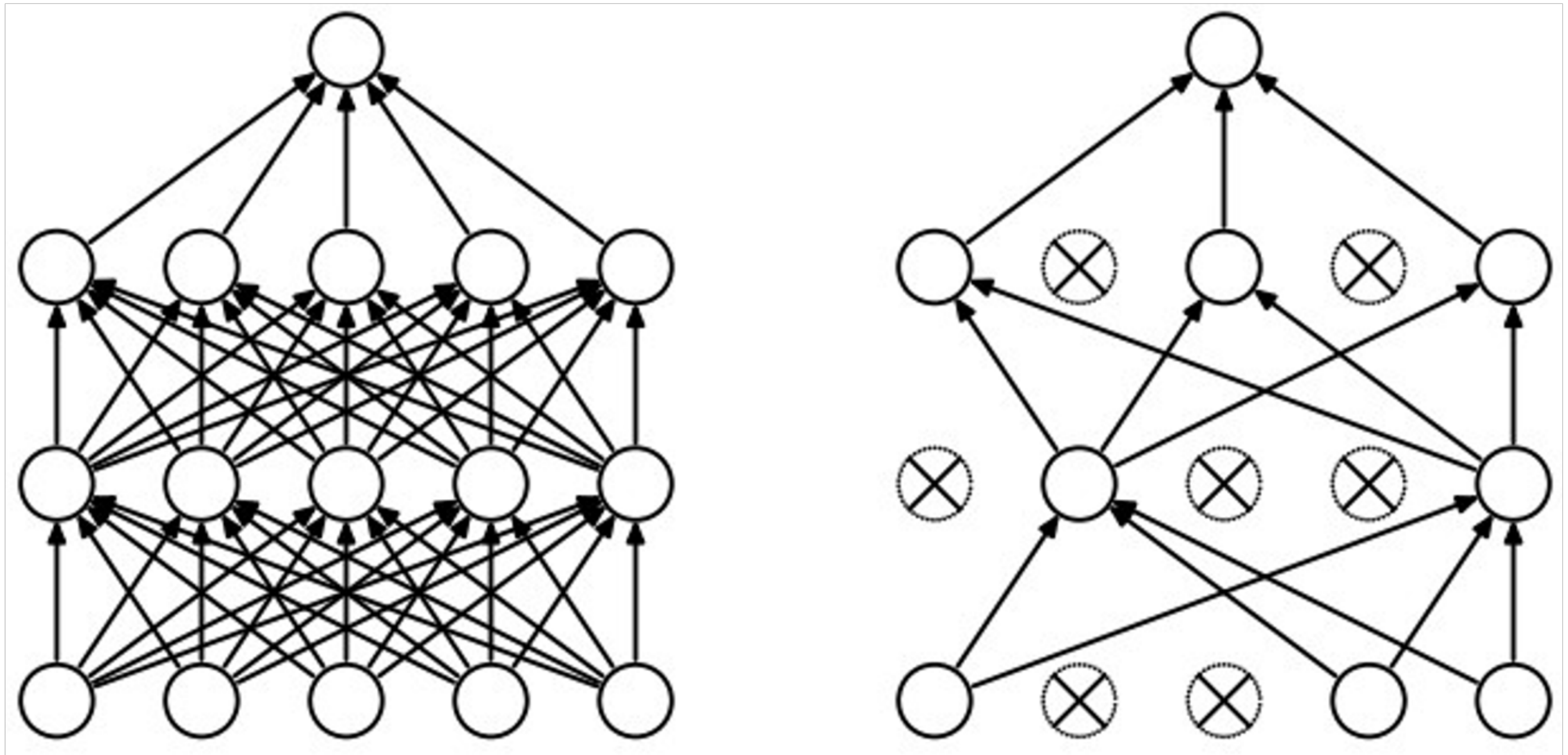
Bagging vs. Boosting



Source: https://www.pluralsight.com/guides/ensemble-methods:-bagging-versus-boosting

Regularization: **Dropout**

"randomly set some neurons to zero in the forward pass"



(a) Standard Neural Net (b) After applying dropout.

[Srivastava et al., 2014]

Model Ensemble in Neural Networks

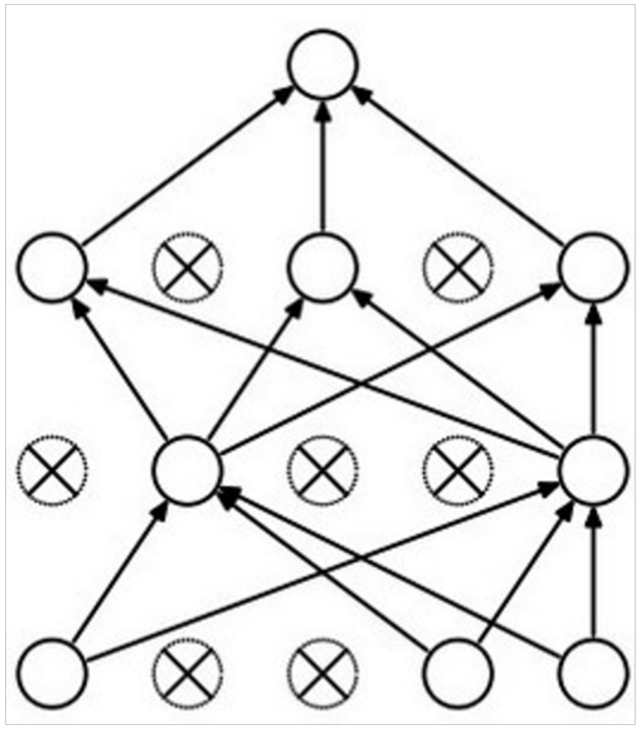
```
p = 0.5 # probability of keeping a unit active. higher = less dropout

def train_step(X):
  """ X contains the data """

  # forward pass for example 3-layer neural network
  H1 = np.maximum(0, np.dot(W1, X) + b1)
  U1 = np.random.rand(*H1.shape) < p # first dropout mask
  H1 *= U1 # drop!
  H2 = np.maximum(0, np.dot(W2, H1) + b2)
  U2 = np.random.rand(*H2.shape) < p # second dropout mask
  H2 *= U2 # drop!
  out = np.dot(W3, H2) + b3

  # backward pass: compute gradients... (not shown)
  # perform parameter update... (not shown)
```

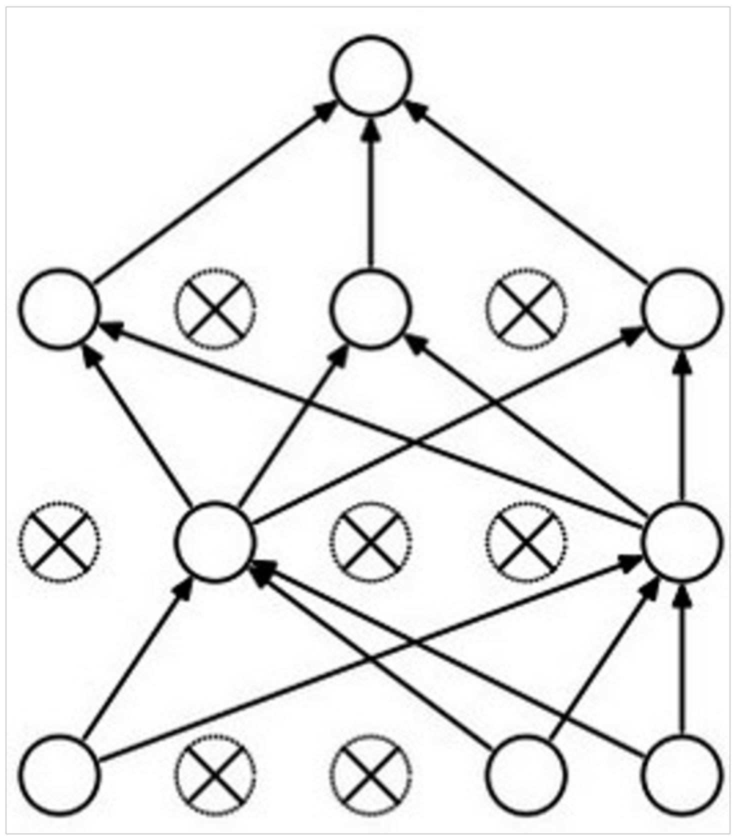
Example forward pass with a 3-layer network using dropout



Slide from Bruno Castro da Silva

Model Ensemble in Neural Networks

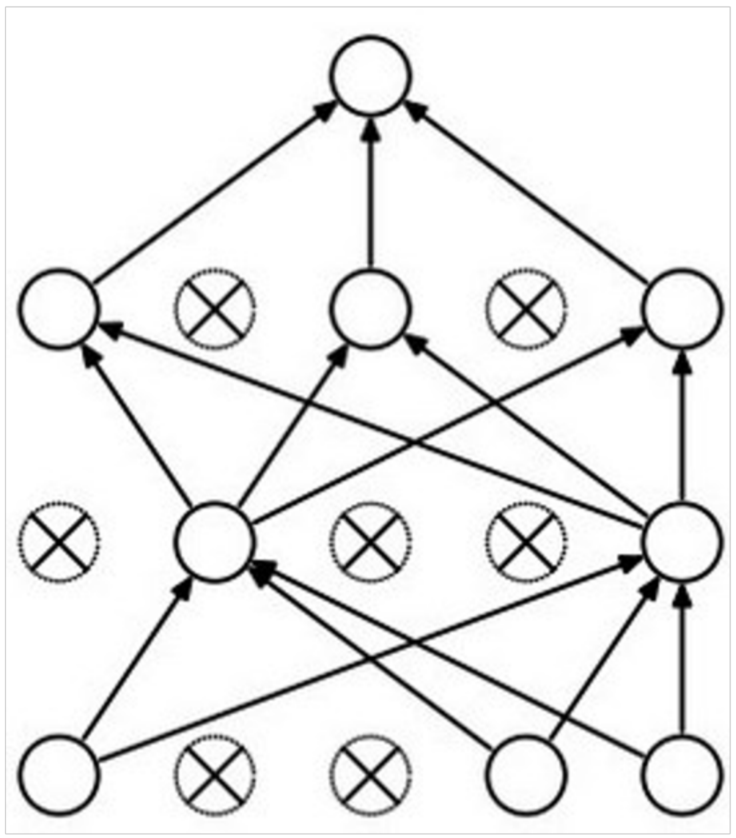
Waaaait a second…
How could this possibly be a good idea?



Forces the network to have a redundant representation.

Model Ensemble in Neural Networks

Waaaait a second…
How could this possibly be a good idea?



Another interpretation:

Dropout is training a large ensemble of models (that share parameters).

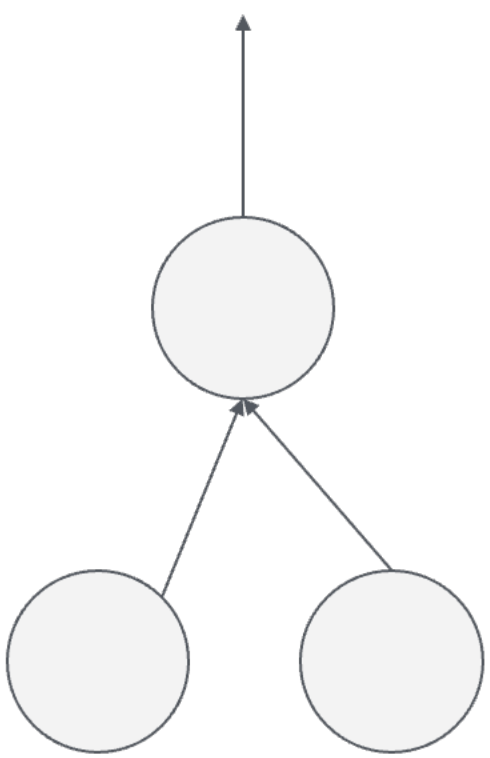
Each binary mask is one model, gets trained on only ~one datapoint.

Model Ensemble in Neural Networks

At test time….

Can in fact do this with a single forward pass! (approximately)

Leave all input neurons turned on (no dropout).



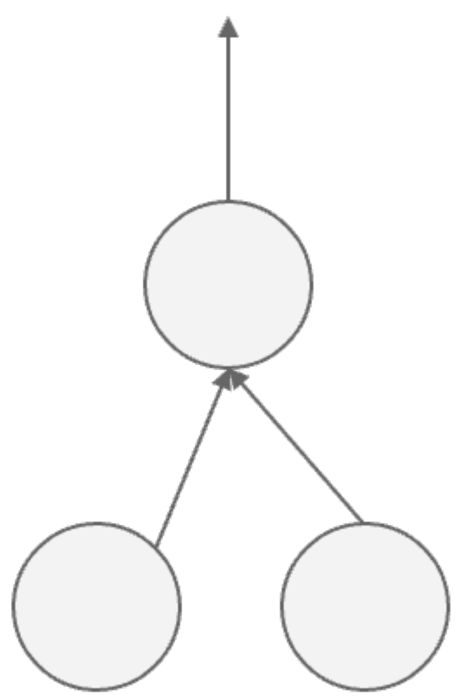
(this can be shown to be an approximation to evaluating the whole ensemble)

Model Ensemble in Neural Networks

At test time….

Can in fact do this with a single forward pass! (approximately)

Leave all input neurons turned on (no dropout).



Q: Suppose that with all inputs present at test time the output of this neuron is x.

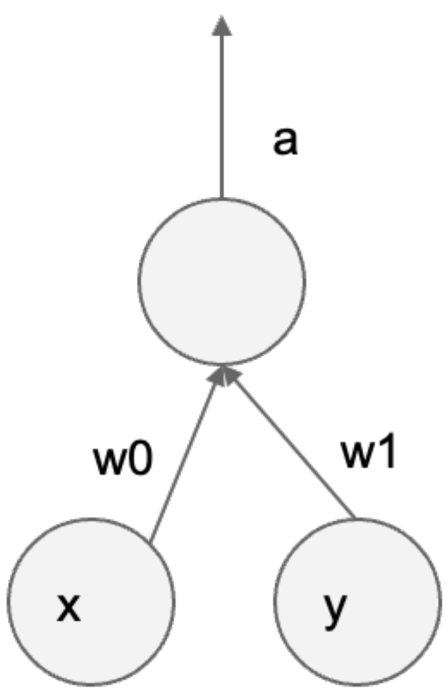
What would its output be during training time, in expectation? (e.g. if p = 0.5)

Model Ensemble in Neural Networks

At test time….

Can in fact do this with a single forward pass! (approximately)

Leave all input neurons turned on (no dropout).



during test: **$a = w0*x + w1*y$**

during train:

$$
\begin{aligned}
E[a] = \tfrac{1}{4} * (&w0*0 + w1*0\\
&w0*0 + w1*y\\
&w0*x + w1*0\\
&w0*x + w1*y)\\
&= \tfrac{1}{4} * (2\, w0*x + 2\, w1*y)\\
&= \tfrac{1}{2} * (w0*x + w1*y)
\end{aligned}
$$

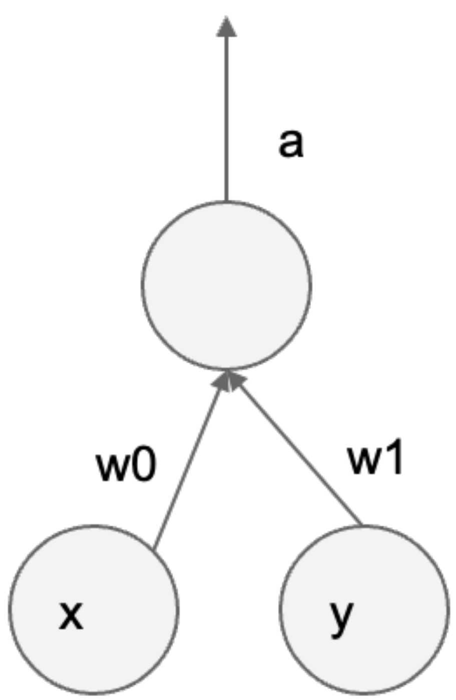
20

Model Ensemble in Neural Networks

At test time….

Can in fact do this with a single forward pass! (approximately)

Leave all input neurons turned on (no dropout).



during test: **a = w0*x + w1*y**

during train:

E[a] = ¼ * (w0*0 + w1*0
w0*0 + w1*y
w0*x + w1*0
w0*x + w1*y)
= ¼ * (2 w0*x + 2 w1*y)
= ½ * (w0*x + w1*y)

With p=0.5, using all inputs in the forward pass would inflate the activations by 2x from what the network was "used to" during training!
=> Have to compensate by scaling the activations back down by ½

Model Ensemble in Neural Networks

We can do something approximate analytically

```
def predict(X):
  # ensembled forward pass
  H1 = np.maximum(0, np.dot(W1, X) + b1) * p # NOTE: scale the activations
  H2 = np.maximum(0, np.dot(W2, H1) + b2) * p # NOTE: scale the activations
  out = np.dot(W3, H2) + b3
```

At test time all neurons are active always
=> We must scale the activations so that for each neuron:
output at test time = expected output at training time

Slide from Bruno Castro da Silva

Model Ensemble in Neural Networks

Dropout Summary

```python
""" Vanilla Dropout: Not recommended implementation (see notes below) """

p = 0.5 # probability of keeping a unit active. higher = less dropout

def train_step(X):
  """ X contains the data """

  # forward pass for example 3-layer neural network
  H1 = np.maximum(0, np.dot(W1, X) + b1)
  U1 = np.random.rand(*H1.shape) < p # first dropout mask
  H1 *= U1 # drop!
  H2 = np.maximum(0, np.dot(W2, H1) + b2)
  U2 = np.random.rand(*H2.shape) < p # second dropout mask
  H2 *= U2 # drop!
  out = np.dot(W3, H2) + b3

  # backward pass: compute gradients... (not shown)
  # perform parameter update... (not shown)

def predict(X):
  # ensembled forward pass
  H1 = np.maximum(0, np.dot(W1, X) + b1) * p # NOTE: scale the activations
  H2 = np.maximum(0, np.dot(W2, H1) + b2) * p # NOTE: scale the activations
  out = np.dot(W3, H2) + b3
```

drop in forward pass

scale at test time

Model Ensemble in Neural Networks

More common: "Inverted dropout"

```
p = 0.5 # probability of keeping a unit active. higher = less dropout

def train_step(X):
  # forward pass for example 3-layer neural network
  H1 = np.maximum(0, np.dot(W1, X) + b1)
  U1 = (np.random.rand(*H1.shape) < p) / p # first dropout mask. Notice /p!
  H1 *= U1 # drop!
  H2 = np.maximum(0, np.dot(W2, H1) + b2)
  U2 = (np.random.rand(*H2.shape) < p) / p # second dropout mask. Notice /p!
  H2 *= U2 # drop!
  out = np.dot(W3, H2) + b3

  # backward pass: compute gradients... (not shown)
  # perform parameter update... (not shown)

def predict(X):
  # ensembled forward pass
  H1 = np.maximum(0, np.dot(W1, X) + b1) # no scaling necessary
  H2 = np.maximum(0, np.dot(W2, H1) + b2)
  out = np.dot(W3, H2) + b3
```

test time is unchanged!

24

Slide from Bruno Castro da Silva