Machine Learning

Ensemble Methods

Topics of previous lectures

- ✓ Ingredients of Machine Learning
- ✓ Classification Basics
- ✓ Basic Linear Classifier
- ✓ K-Nearest Neighbours Classifier
- ✓ Naive Bayes Classifier
- ✓ Linear and Quadratic Discriminant Analysis
- ✓ Support Vector Machines (SVM)
- ✓ Decision Trees

Topics of today's lecture

- Bagging
- Random Forest
- Weighted Voting
- Stacking

Motivation for Ensembles

In 1907, 787 villagers tried to guess the weight of **ox**

Motivation for Ensembles

In 1907, 787 villagers tried to guess the weight of **ox**

None of them guessed it correctly, but the **average** guess (**542.9** kg) was very close to actual weight of ox (**543.4** kg)

What are Ensemble Methods?

Definition 1

Combinations of different models are called ensembles.

What are Ensemble Methods?

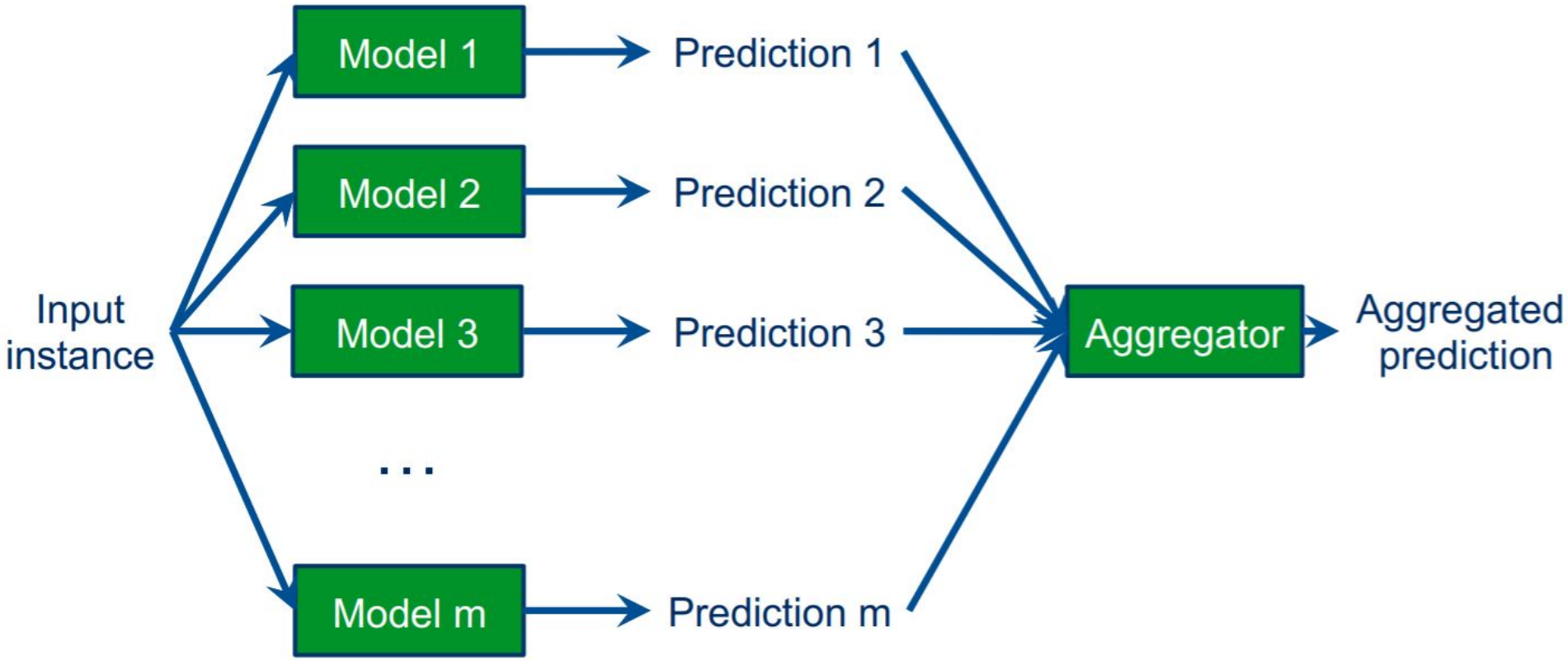
Definition 1

Combinations of different models are called ensembles.

Definition 2

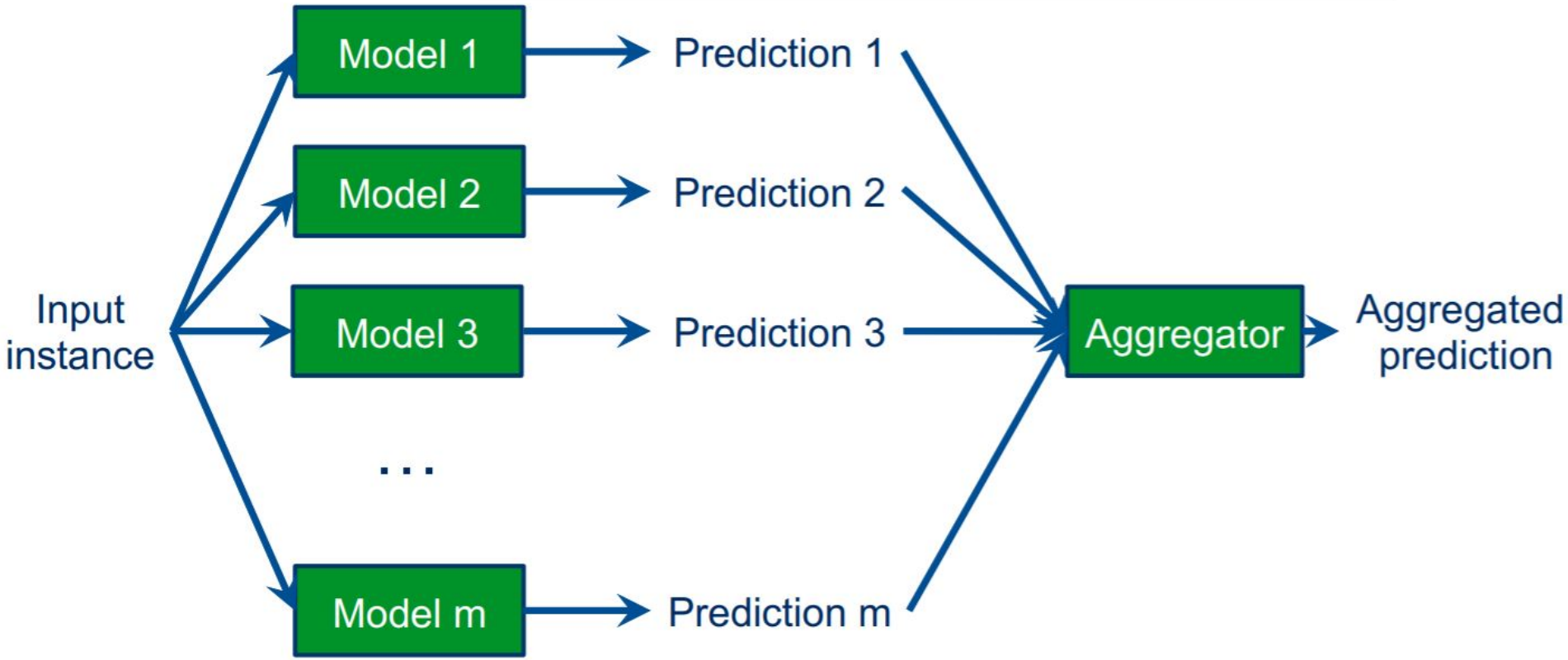
Ensemble method is the one that makes predictions by aggregating predictions from multiple models (ensemble).

Aggregation of predictions



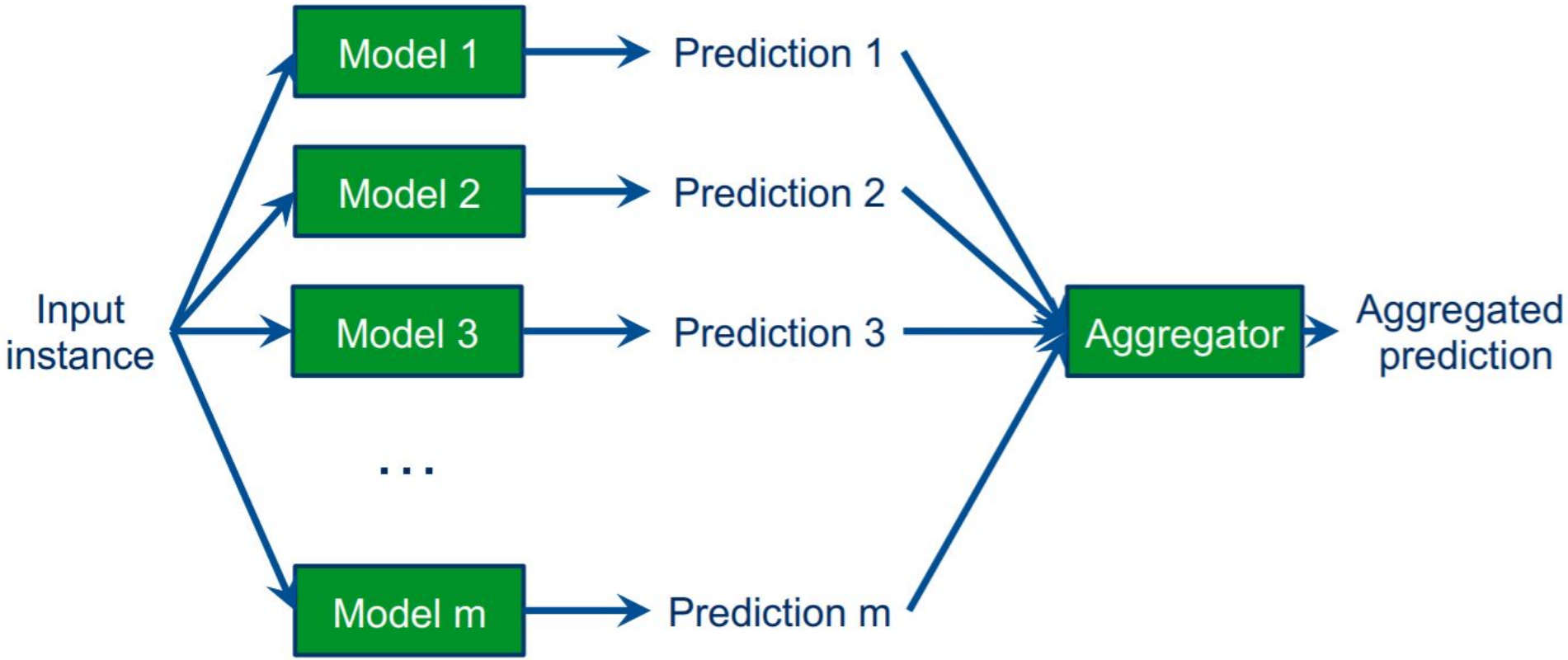
Aggregation of predictions

- How to get multiple models?

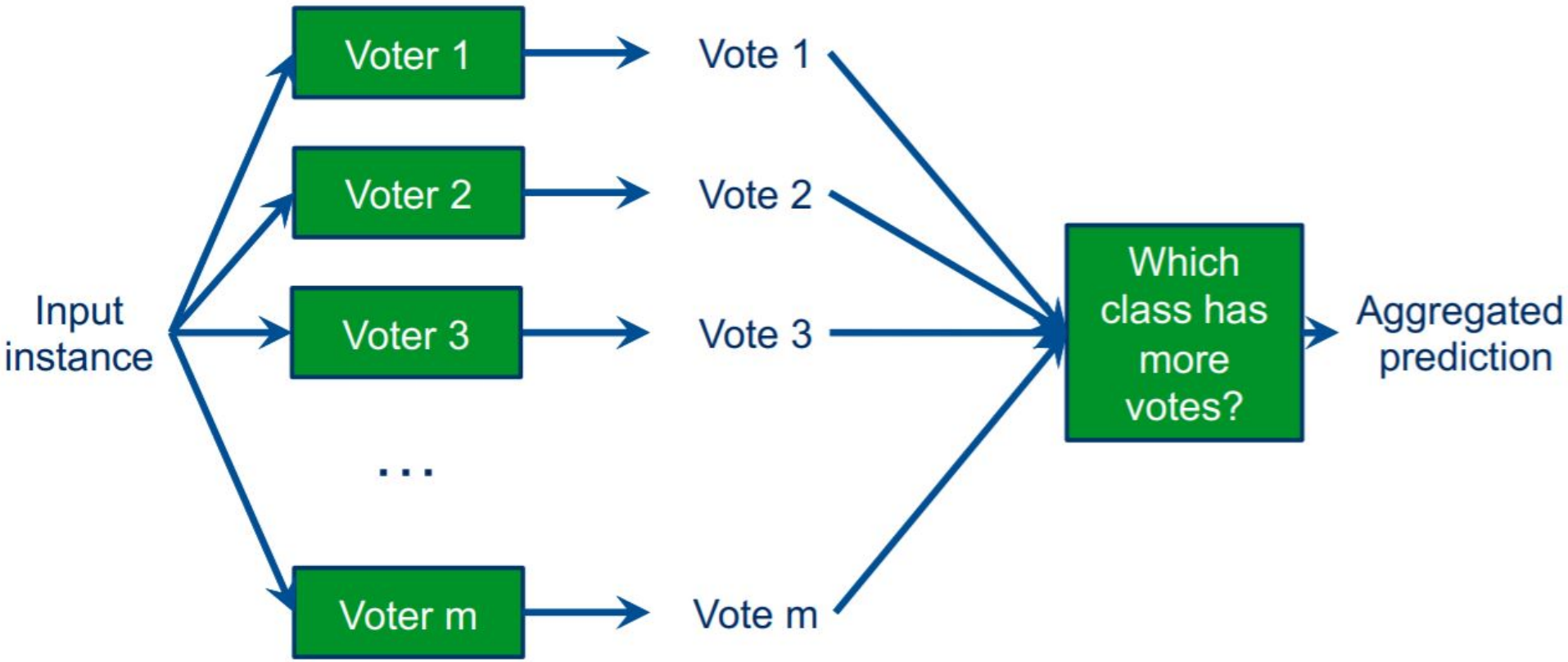


Aggregation of predictions

- How to get multiple models?
- How to aggregate the predictions?



Voting in binary classification



Theory for voting (binary task)

- Originates from Marquis de Condorcet 1785:
  "Essay on the Application of Analysis to the Probability of Majority Decisions"

Theory for voting (binary task)

- Originates from Marquis de Condorcet 1785:
  "Essay on the Application of Analysis to the Probability of Majority Decisions"
- Suppose there are M voters, each having independent errors, and the individual error probability is ϵ

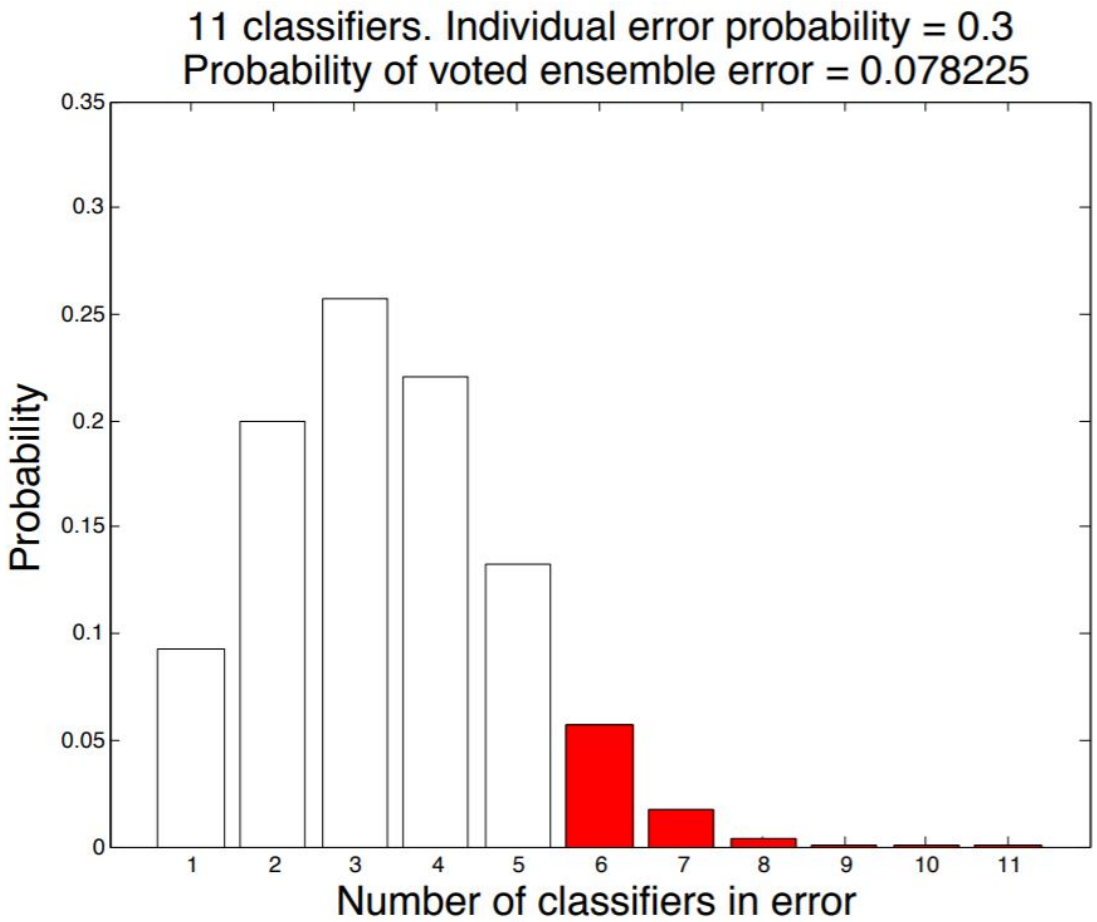
Theory for voting (binary task)

- Originates from Marquis de Condorcet 1785:
  "Essay on the Application of Analysis to the Probability of Majority Decisions"
- Suppose there are M voters, each having independent errors, and the individual error probability is ϵ
- Majority vote is wrong with probability:

$$\mathbb{P}(\text{majority vote error}) = \sum_{k \geq \lceil \frac{M+1}{2} \rceil} C_M^k \epsilon^k (1-\epsilon)^{M-k}$$

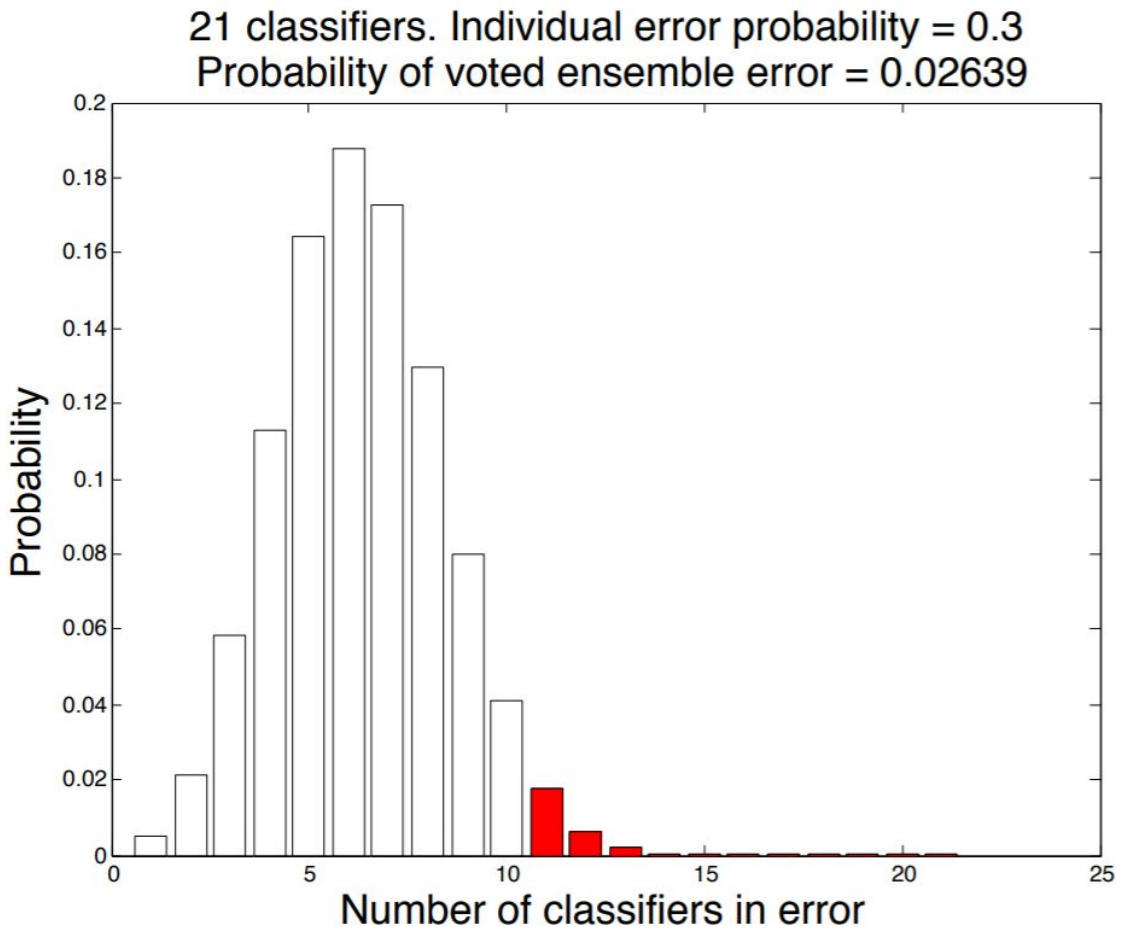
$M = 11$ Classifiers

$$\mathbb{P}(\text{majority vote error}) = \sum_{k \geq \lceil \frac{M+1}{2} \rceil} C_M^k \epsilon^k (1-\epsilon)^{M-k}$$



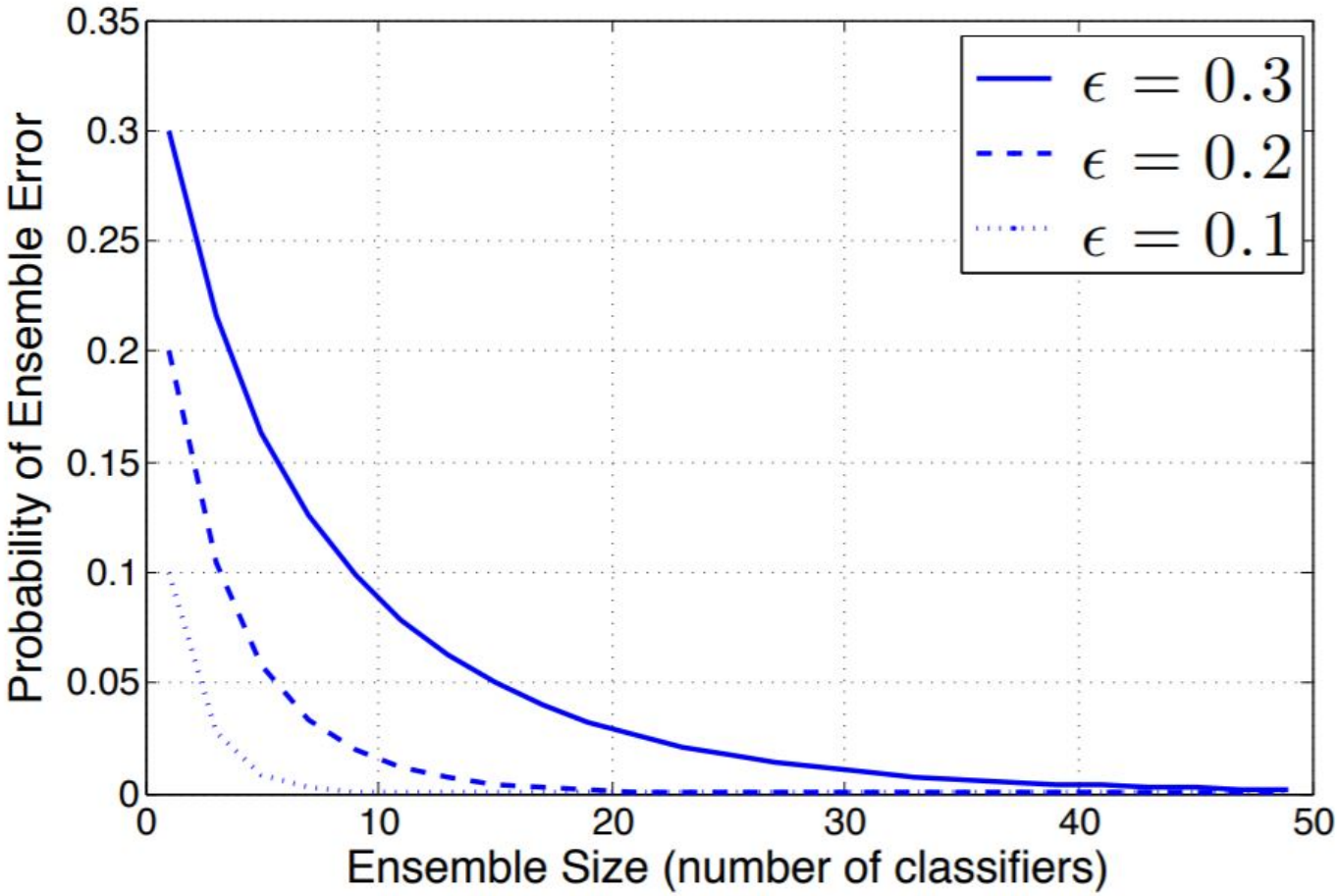
$M = 21$ Classifiers

$$\mathbb{P}(\text{majority vote error}) = \sum_{k \geq \lceil \frac{M+1}{2} \rceil} C_M^k \epsilon^k (1-\epsilon)^{M-k}$$



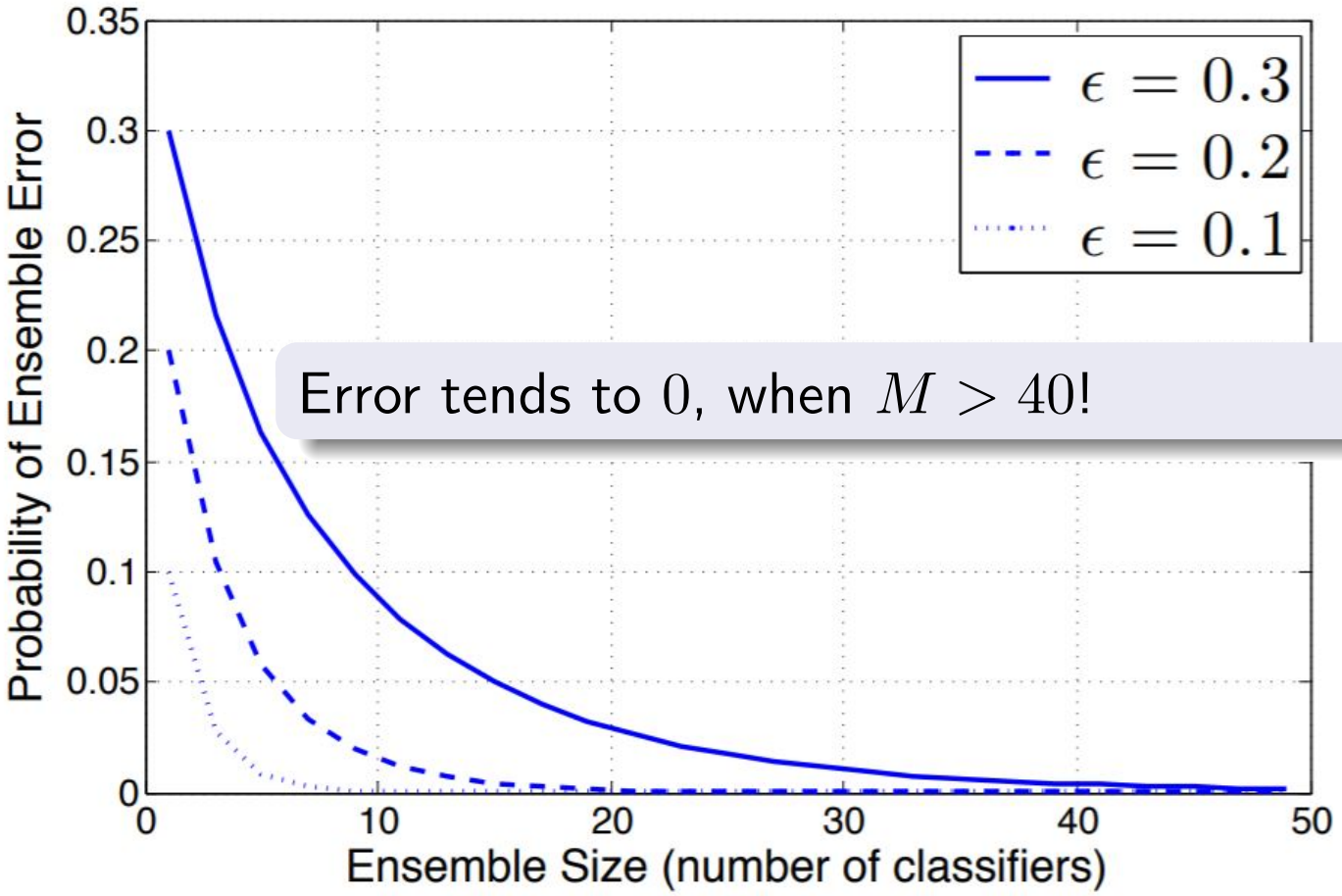
$M \uparrow$ Classifiers

$$\mathbb{P}(\text{majority vote error}) = \sum_{k \geq \lceil \frac{M+1}{2} \rceil} C_M^k \epsilon^k (1-\epsilon)^{M-k}$$



$M \uparrow$ Classifiers

$$\mathbb{P}(\text{majority vote error}) = \sum_{k \geq \lceil \frac{M+1}{2} \rceil} C_M^k \epsilon^k (1-\epsilon)^{M-k}$$



Error tends to 0, when $M > 40$!

Why does this theory not work in practice?

Errors of voters are not independent!

Key challenge of ensemble learning

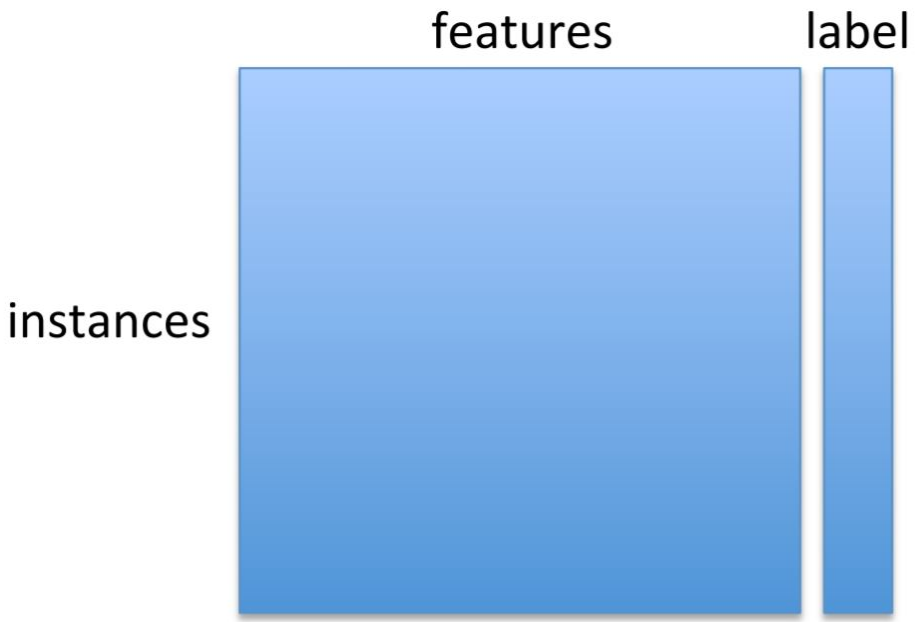
The key challenge of ensemble learning is to obtain models that are:

- reasonably accurate
- as independent as possible

How to achieve independence of models?

Idea 1: Split + Train

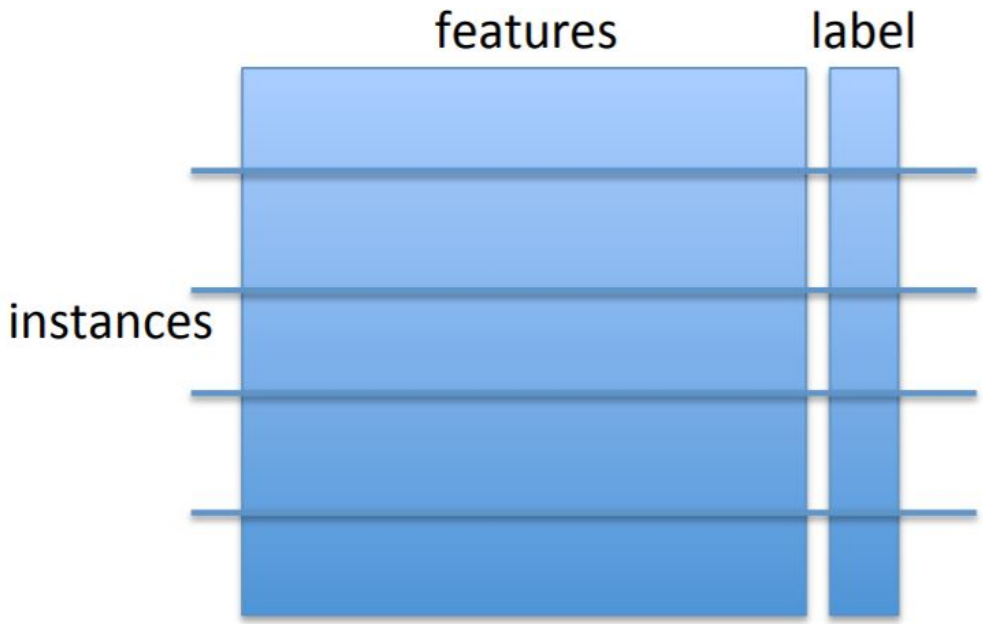
- Randomly split training data into M disjoint groups of instances, train a separate model on each group



How to achieve independence of models?

Idea 1: Split + Train

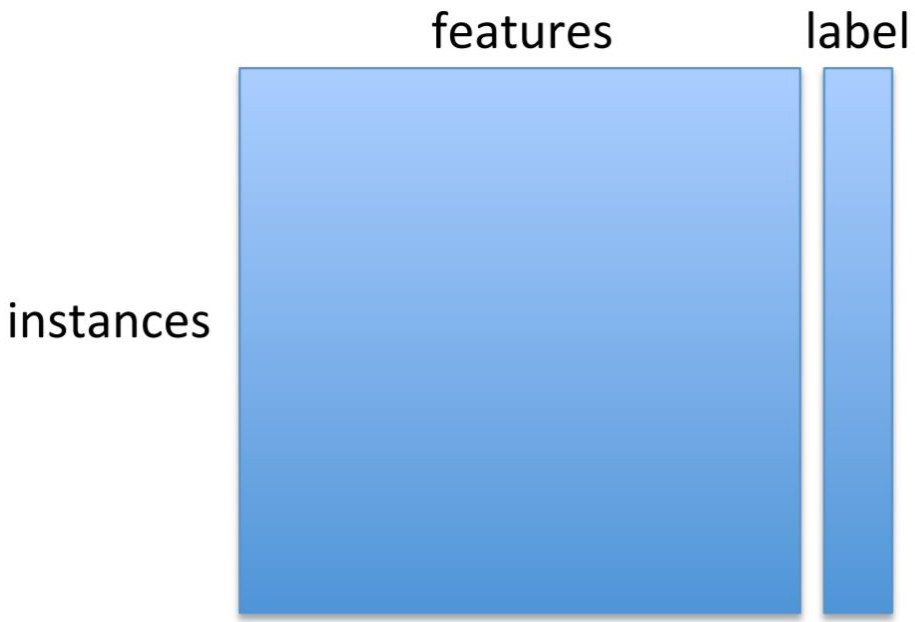
- Randomly split training data into M disjoint groups of instances, train a separate model on each group
- This is not a good idea, since groups get small and the learned models will have poor prediction quality



How to achieve independence of models?

Idea 2: Split by features + Train

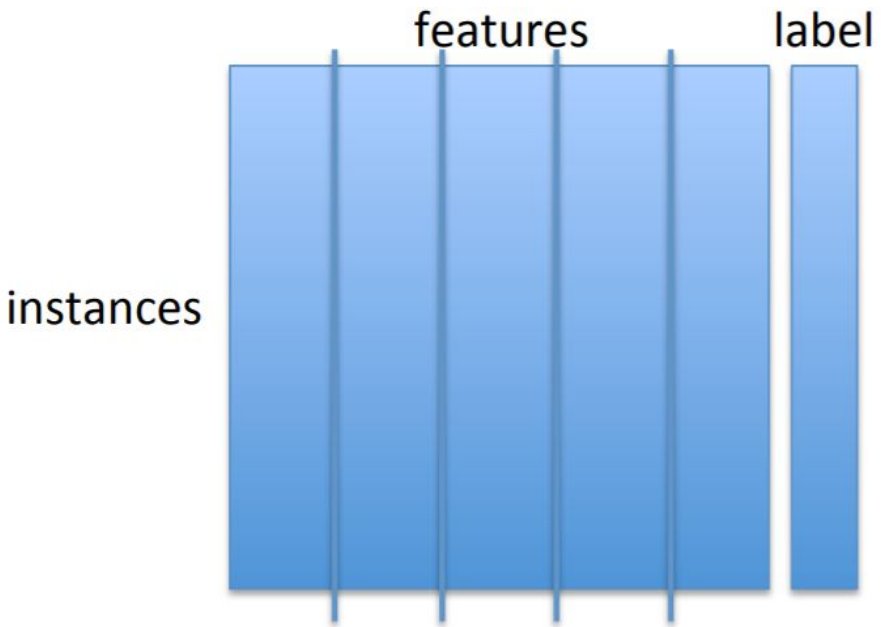
- Randomly split training data into M disjoint groups of features, train a separate model on each group



How to achieve independence of models?

Idea 2: Split by features + Train

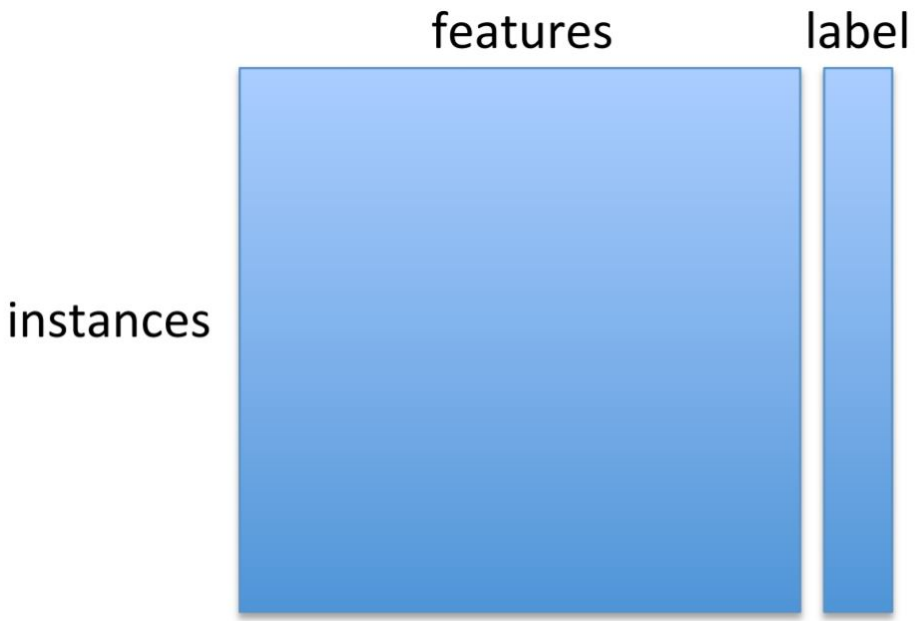
- Randomly split training data into M disjoint groups of features, train a separate model on each group
- This is not a good idea, since not having good features can be even worse than not having enough instances to train



How to achieve independence of models?

Idea 3: Overlapping subsets of instances + Train

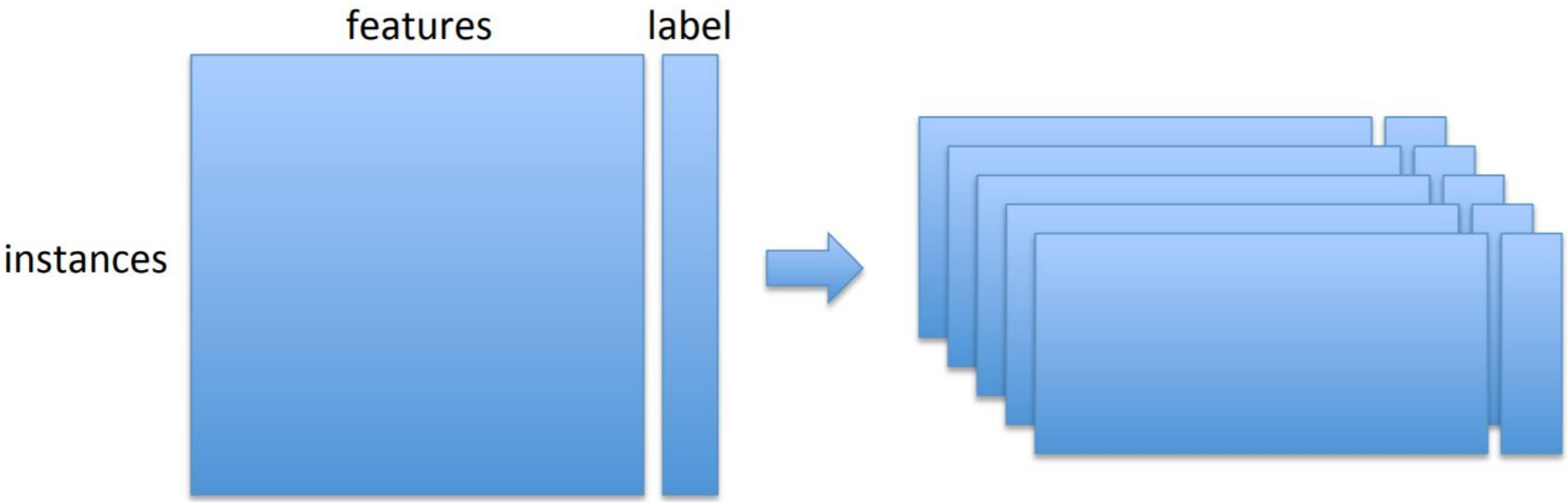
- Randomly sample M overlapping groups of instances, train a separate model on each group



How to achieve independence of models?

Idea 3: Overlapping subsets of instances + Train

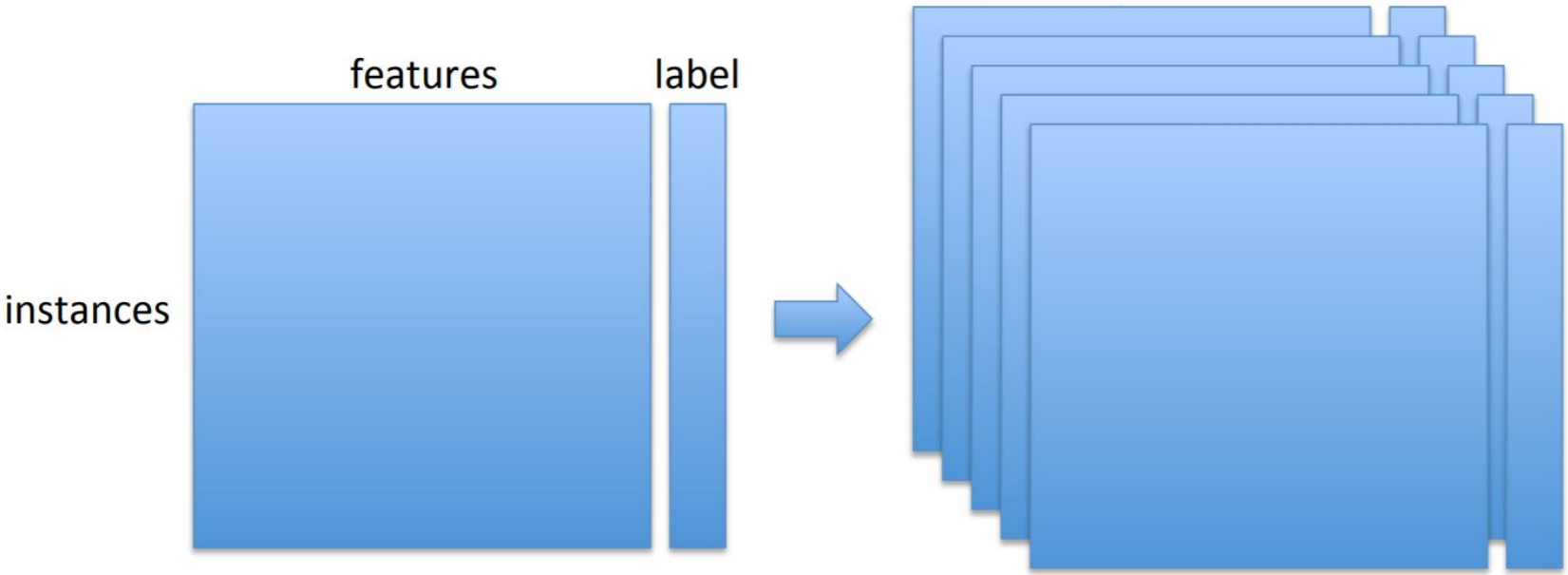
- Randomly sample M overlapping groups of instances, train a separate model on each group
- Not too bad, but still smaller training sets



How to achieve independence of models?

Idea 4: Bootstrap-sampled instances + Train

- Sample with replacement M overlapping groups of instances with same size as original dataset (bootstrapping), train a separate model on each group



Generating a new dataset by Bootstrapping

Bootstrapping:
Sample n items with replacement from the original n training instances

Original data

id	x1	x2	x3	y
1	0.18	0.45	0.8	0
2	0.11	0.82	0.07	0
3	0.87	0.3	0.21	1
4	0.34	0.49	0.18	1
5	0.95	0.64	0.63	0
6	0.03	0.59	0.15	1

Bootstrap 1

id	x1	x2	x3	y
1	0.18	0.45	0.8	0
1	0.18	0.45	0.8	0
3	0.87	0.3	0.21	1
4	0.34	0.49	0.18	1
5	0.95	0.64	0.63	0
5	0.95	0.64	0.63	0

Bootstrap 2

id	x1	x2	x3	y
1	0.18	0.45	0.8	0
2	0.11	0.82	0.07	0
3	0.87	0.3	0.21	1
6	0.03	0.59	0.15	1
6	0.03	0.59	0.15	1
6	0.03	0.59	0.15	1

Generating a new dataset by Bootstrapping

Bootstrapping:
Sample n items with replacement from the original n training instances

Original data

id	x1	x2	x3	y
1	0.18	0.45	0.8	0
2	0.11	0.82	0.07	0
3	0.87	0.3	0.21	1
4	0.34	0.49	0.18	1
5	0.95	0.64	0.63	0
6	0.03	0.59	0.15	1

Bootstrap 1

id	x1	x2	x3	y
1	0.18	0.45	0.8	0
1	0.18	0.45	0.8	0
3	0.87	0.3	0.21	1
4	0.34	0.49	0.18	1
5	0.95	0.64	0.63	0
5	0.95	0.64	0.63	0

Bootstrap 2

id	x1	x2	x3	y
1	0.18	0.45	0.8	0
2	0.11	0.82	0.07	0
3	0.87	0.3	0.21	1
6	0.03	0.59	0.15	1
6	0.03	0.59	0.15	1
6	0.03	0.59	0.15	1

Is it possible that a particular training instance is not included in the bootstrap sample?

Generating a new dataset by Bootstrapping

Bootstrapping:
Sample n items with replacement from the original n training instances

Original data

id	x1	x2	x3	y
1	0.18	0.45	0.8	0
2	0.11	0.82	0.07	0
3	0.87	0.3	0.21	1
4	0.34	0.49	0.18	1
5	0.95	0.64	0.63	0
6	0.03	0.59	0.15	1

Bootstrap 1

id	x1	x2	x3	y
1	0.18	0.45	0.8	0
1	0.18	0.45	0.8	0
3	0.87	0.3	0.21	1
4	0.34	0.49	0.18	1
5	0.95	0.64	0.63	0
5	0.95	0.64	0.63	0

Bootstrap 2

id	x1	x2	x3	y
1	0.18	0.45	0.8	0
2	0.11	0.82	0.07	0
3	0.87	0.3	0.21	1
6	0.03	0.59	0.15	1
6	0.03	0.59	0.15	1
6	0.03	0.59	0.15	1

Is it possible that a particular training instance is not included in the bootstrap sample?

It is possible with less than 50% probability

Bagging: Bootstrap AGGgregating (Breiman 1996)

- Bootstrap sampling:
  - Sample with replacement M overlapping groups of instances with the same size as original dataset

Bagging: Bootstrap AGGgregating (Breiman 1996)

- Bootstrap sampling:
  - Sample with replacement M overlapping groups of instances with the same size as original dataset
  - That is, each original instance will have 0, 1, or more copies in such a group

Bagging: Bootstrap AGGgregating (Breiman 1996)

- Bootstrap sampling:
  - Sample with replacement M overlapping groups of instances with the same size as original dataset
  - That is, each original instance will have 0, 1, or more copies in such a group
  - The probability that an instance will not be included in the bootstrap sample is $(\frac{n-1}{n})^n$, where n is the size of the dataset.

Bagging: Bootstrap AGGgregating (Breiman 1996)

- Bootstrap sampling:
  - Sample with replacement M overlapping groups of instances with the same size as original dataset
  - That is, each original instance will have 0, 1, or more copies in such a group
  - The probability that an instance will not be included in the bootstrap sample is $(\frac{n-1}{n})^n$, where n is the size of the dataset.
  - If the original dataset is big enough ($n \to \infty$), then this probability will tend to $\frac{1}{e} \approx 37\%$.

Bagging: Bootstrap AGGgregating (Breiman 1996)

- Bootstrap sampling:
  - Sample with replacement M overlapping groups of instances with the same size as original dataset
  - That is, each original instance will have 0, 1, or more copies in such a group
  - The probability that an instance will not be included in the bootstrap sample is $(\frac{n-1}{n})^n$, where n is the size of the dataset.
  - If the original dataset is big enough ($n \to \infty$), then this probability will tend to $\frac{1}{e} \approx 37\%$.
- Bagging (Bootstrap AGGregating = BAGG):
  Train a model separately on each bootstrapped dataset and then aggregate the results.

Bagging: Bootstrap AGGgregating (Breiman 1996)

- Bootstrap sampling:
  - Sample with replacement M overlapping groups of instances with the same size as original dataset
  - That is, each original instance will have 0, 1, or more copies in such a group
  - Th… the bootstrap sam…
  - If t… s probability will tend… e
- Bagging (Bootstrap AGGregating = BAGG):
  Train a model separately on each bootstrapped dataset and then aggregate the results.

> Is it possible that a particular training instance is not included in any of the M bootstrapped datasets?

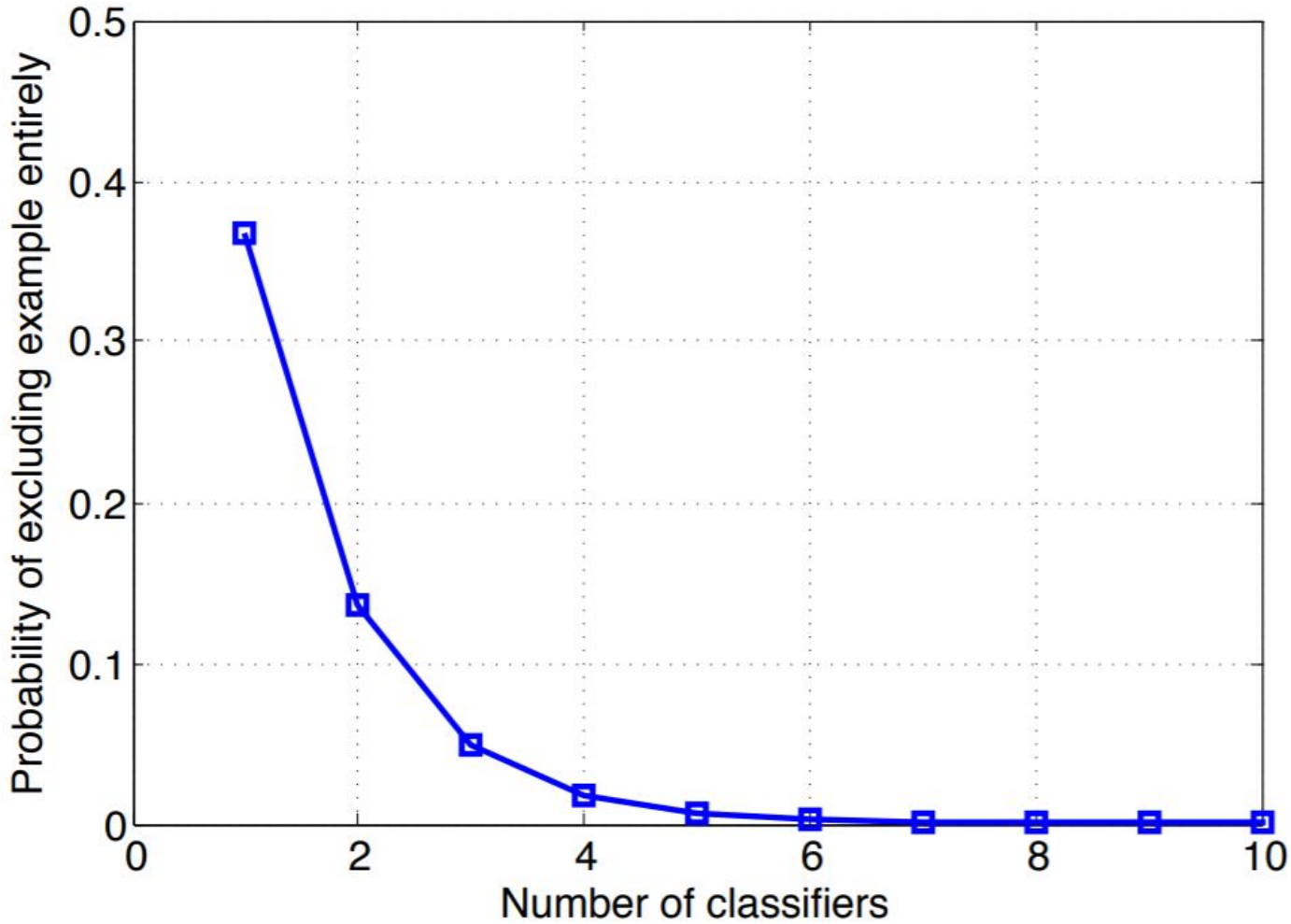
Bagging: Bootstrap AGGgregating (Breiman 1996)

- Bootstrap sampling:
  - Sample with replacement M overlapping groups of instances with the same size as original dataset
  - That is, each original instance will have 0, 1, or more copies in such a group
  - Th… the bootstrap sam…
  - If t… s probability will tend… e
- Bagging… and then aggregate…

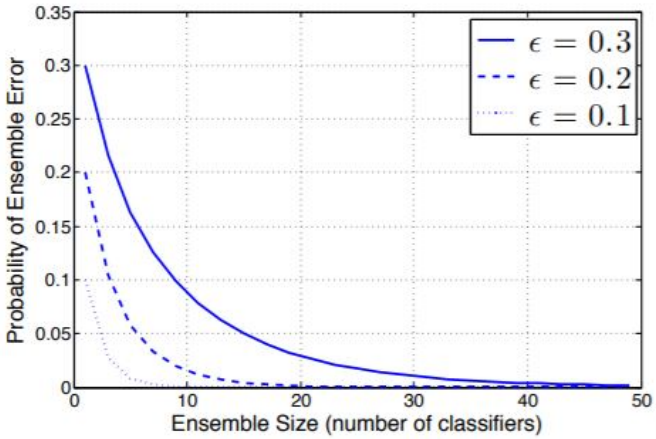
> Is it possible that a particular training instance is not included in any of the M bootstrapped datasets?

> **It is possible, but happens less and less frequently as M grows**

$\mathbb{P}$(excluding an instance from the whole ensemble)

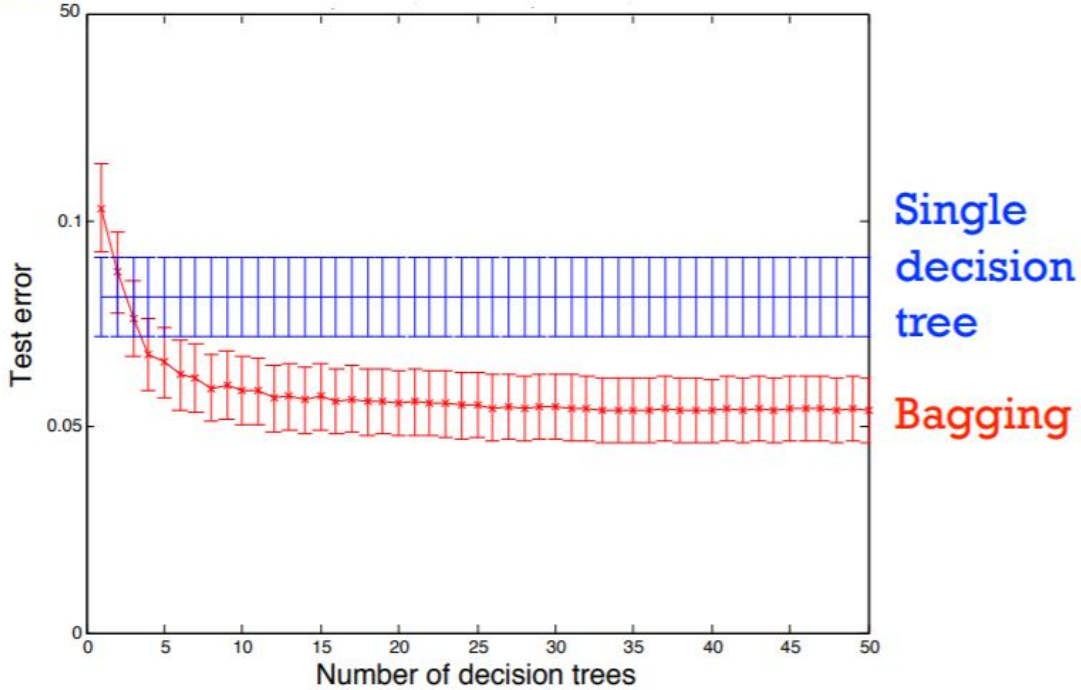


Practical Considerations



Dream!

Reality!



Bagging

Algorithm Bagging($D, T, \mathscr{A}$) – train an ensemble of models from bootstrap samples.

Input : data set D; ensemble size T; learning algorithm $\mathscr{A}$.

Output : ensemble of models whose predictions are to be combined by voting or averaging.

1 **for** $t = 1$ to T **do**

2 build a bootstrap sample D_t from D by sampling $|D|$ data points with replacement;

3 run $\mathscr{A}$ on D_t to produce a model M_t;

4 **end**

5 **return** $\{M_t | 1 \leq t \leq T\}$

When is bagging useful?

- Bagging is bad if models are very similar (not independent enough)

When is bagging useful?

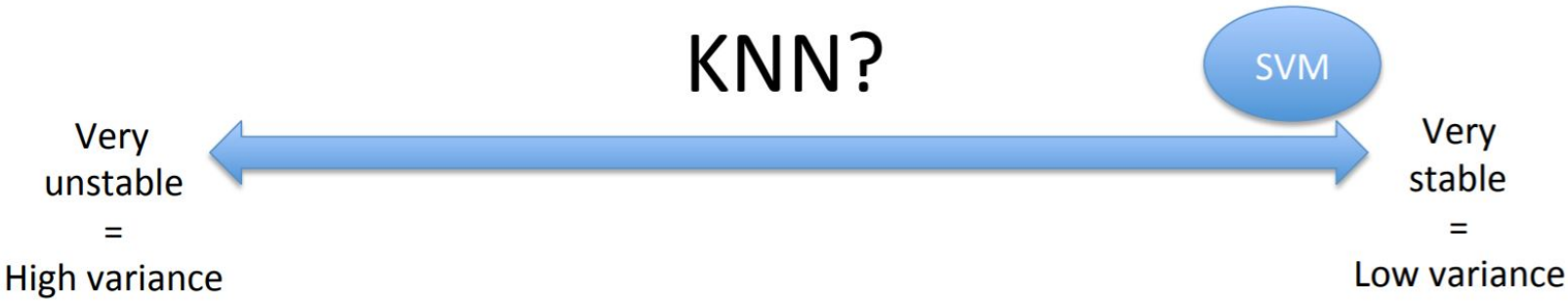
- Bagging is bad if models are very similar (not independent enough)
- This happens if the learning algorithm is stable, that is, model does not usually change much after changing a few training instances

When is bagging useful?

- Bagging is bad if models are very similar (not independent enough)
- This happens if the learning algorithm is stable, that is, model does not usually change much after changing a few training instances

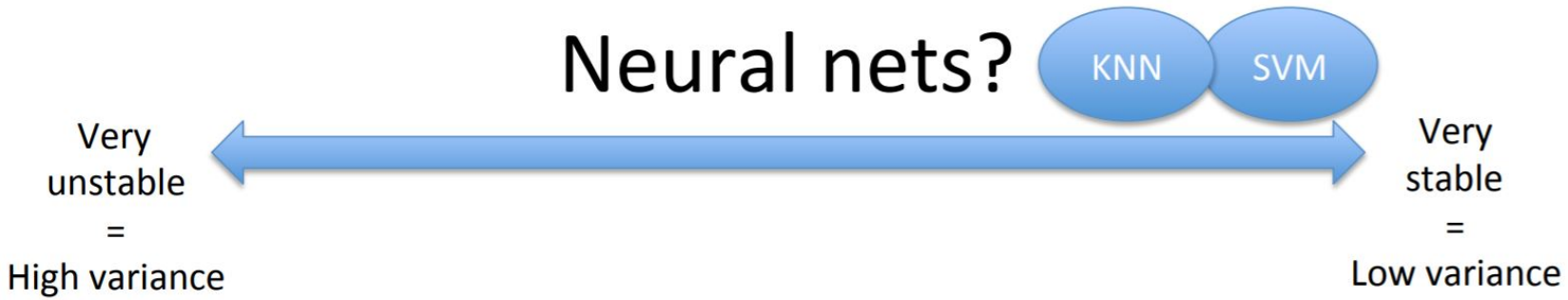
When is bagging useful?

- Bagging is bad if models are very similar (not independent enough)
- This happens if the learning algorithm is stable, that is, model does not usually change much after changing a few training instances



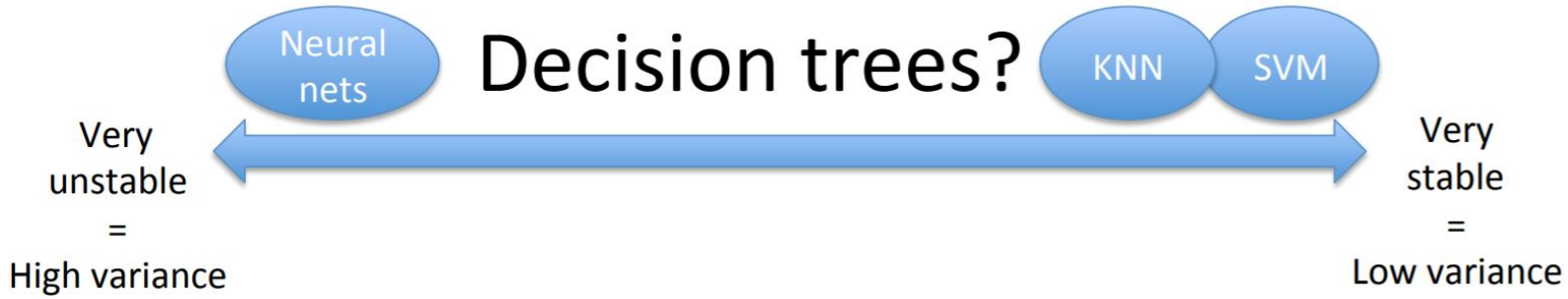
When is bagging useful?

- Bagging is bad if models are very similar (not independent enough)
- This happens if the learning algorithm is stable, that is, model does not usually change much after changing a few training instances



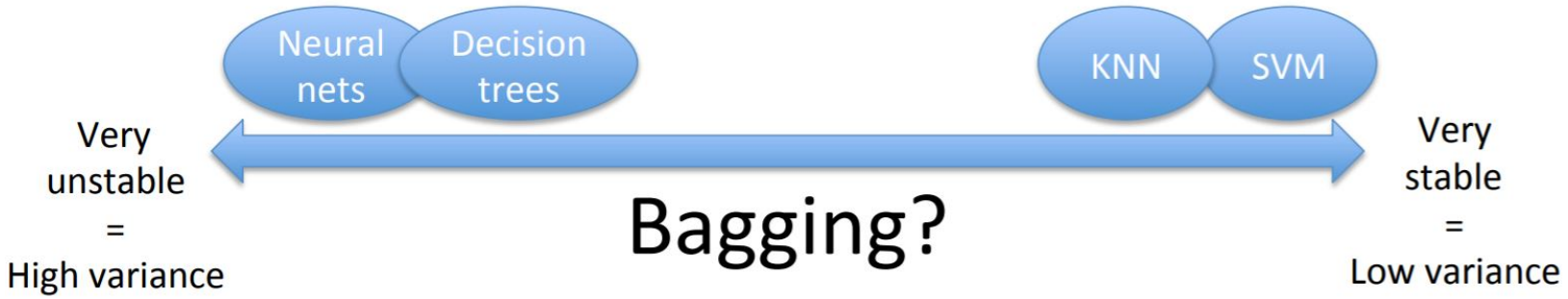
When is bagging useful?

- Bagging is bad if models are very similar (not independent enough)
- This happens if the learning algorithm is stable, that is, model does not usually change much after changing a few training instances



When is bagging useful?

- Bagging is bad if models are very similar (not independent enough)
- This happens if the learning algorithm is stable, that is, model does not usually change much after changing a few training instances



When is bagging useful?

- Bagging is bad if models are very similar (not independent enough)
- This happens if the learning algorithm is stable, that is, model does not usually change much after changing a few training instances

Summary of Bagging

- Individual models trained on bootstrap-sampled instances, predictions are aggregated

Summary of Bagging

- Individual models trained on bootstrap-sampled instances, predictions are aggregated
- Bagging is useful when the algorithm to learn individual models is:
  - Relatively accurate

Summary of Bagging

- Individual models trained on bootstrap-sampled instances, predictions are aggregated
- Bagging is useful when the algorithm to learn individual models is:
  - Relatively accurate
  - Relatively unstable (high variance)

Summary of Bagging

- Individual models trained on bootstrap-sampled instances, predictions are aggregated
- Bagging is useful when the algorithm to learn individual models is:
  - Relatively accurate
  - Relatively unstable (high variance)
- The aggregated model is then usually better than the original model trained on full dataset

Random Forest (Breiman 2000)

- Random forest: similar to bagged decision trees but different in using features

Random Forest (Breiman 2000)

- Random forest: similar to bagged decision trees but different in using features
- **In each recursive step** of learning decision trees:

Random Forest (Breiman 2000)

- Random forest: similar to bagged decision trees but different in using features
- **In each recursive step** of learning decision trees:
  - **Randomly select F features** out of all P given features

Random Forest (Breiman 2000)

- Random forest: similar to bagged decision trees but different in using features
- **In each recursive step** of learning decision trees:
  - **Randomly select F features** out of all P given features
  - Find the best split among these features

Random Forest (Breiman 2000)

- Random forest: similar to bagged decision trees but different in using features
- **In each recursive step** of learning decision trees:
  - **Randomly select F features** out of all P given features
  - Find the best split among these features
- Parameter F is usually fixed to be $F = \sqrt{P}$ for classification

Random Forest

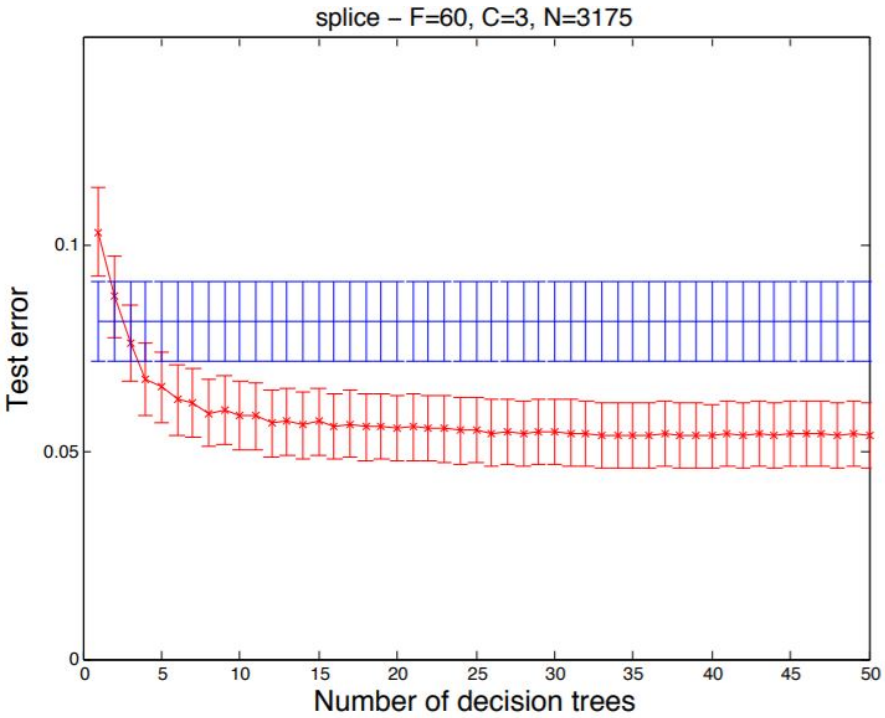
Algorithm RandomForest(D, T, d) – train an ensemble of tree models from bootstrap samples and random subspaces.

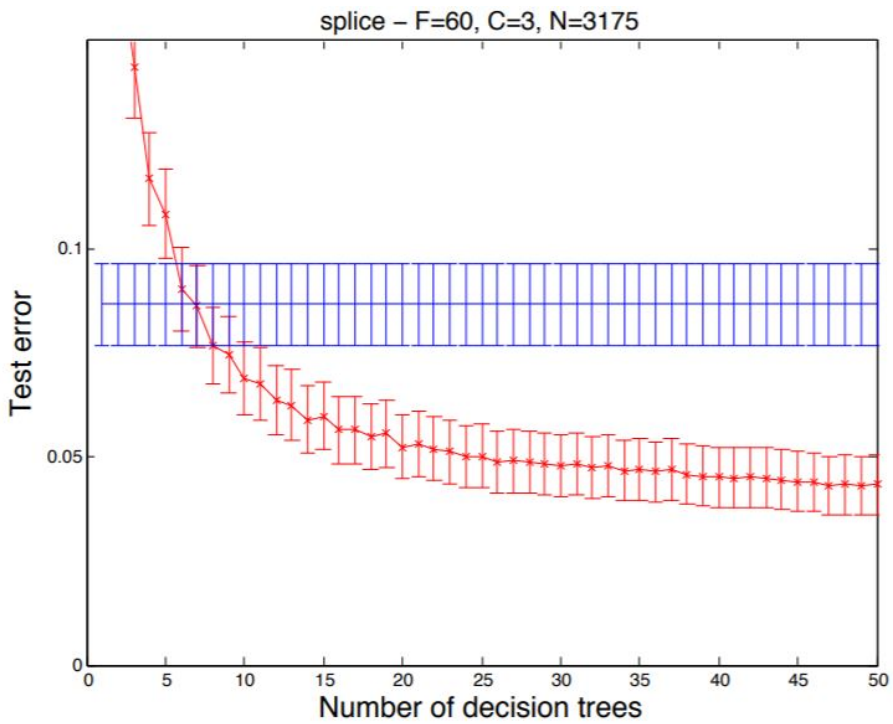
Input : data set D; ensemble size T; subspace dimension d.

Output : ensemble of tree models whose predictions are to be combined by voting or averaging.

1 **for** $t = 1$ to T **do**
2 build a bootstrap sample D_t from D by sampling $|D|$ data points with replacement;
3 select d features at random and reduce dimensionality of D_t accordingly;
4 train a tree model M_t on D_t without pruning;
5 **end**
6 **return** $\{M_t | 1 \leq t \leq T\}$

Bagging vs Random Forest





Bagging (LEFT) vs Random Forests (RIGHT) on the Splice dataset.

How many trees in a forest?

- The more the better!

How many trees in a forest?

- The more the better!
- How to know that there are enough?

How many trees in a forest?

- The more the better!
- How to know that there are enough?
- Out-of-bag (OOB) error

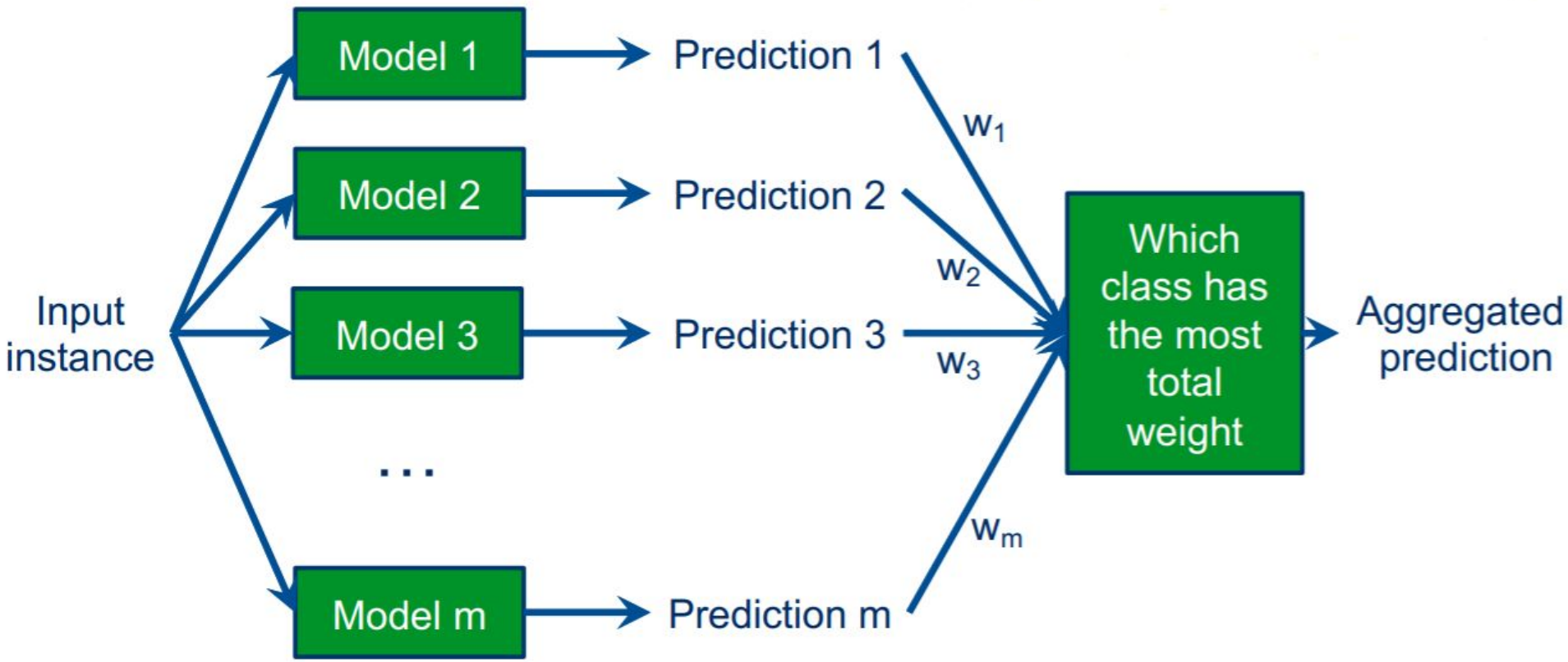
How many trees in a forest?

- The more the better!
- How to know that there are enough?
- Out-of-bag (OOB) error
  - For each training instance make a prediction using trees that do not use that instance and evaluate

How many trees in a forest?

- The more the better!
- How to know that there are enough?
- Out-of-bag (OOB) error
  - For each training instance make a prediction using trees that do not use that instance and evaluate
  - Stabilisation of OOB error suggests that there are enough trees

Weighted voting



Weighted voting

- Ensembles can be
  - Homogeneous - all individual models are obtained with the same learning algorithm, on slightly different datasets

Weighted voting

- Ensembles can be
  - Homogeneous - all individual models are obtained with the same learning algorithm, on slightly different datasets
  - Heterogeneous – individual models are obtained with different algorithms

Weighted voting

- Ensembles can be
  - Homogeneous - all individual models are obtained with the same learning algorithm, on slightly different datasets
  - Heterogeneous – individual models are obtained with different algorithms
- Classification can be performed by

Weighted voting

- Ensembles can be
  - Homogeneous - all individual models are obtained with the same learning algorithm, on slightly different datasets
  - Heterogeneous – individual models are obtained with different algorithms
- Classification can be performed by
  - Voting

Weighted voting

- Ensembles can be
  - Homogeneous - all individual models are obtained with the same learning algorithm, on slightly different datasets
  - Heterogeneous – individual models are obtained with different algorithms
- Classification can be performed by
  - Voting
  - Weighted voting

Weighted voting

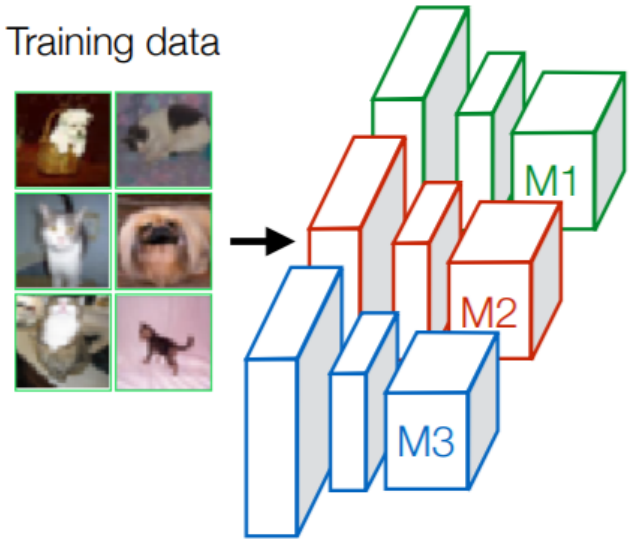
- Ensembles can be
  - Homogeneous - all individual models are obtained with the same learning algorithm, on slightly different datasets
  - Heterogeneous – individual models are obtained with different algorithms
- Classification can be performed by
  - Voting
  - Weighted voting
- Better models should have higher weights

Weighted voting

- Ensembles can be
  - Homogeneous - all individual models are obtained with the same learning algorithm, on slightly different datasets
  - Heterogeneous – individual models are obtained with different algorithms
- Classification can be performed by
  - Voting
  - Weighted voting
- Better models should have higher weights
- How to obtain weights?

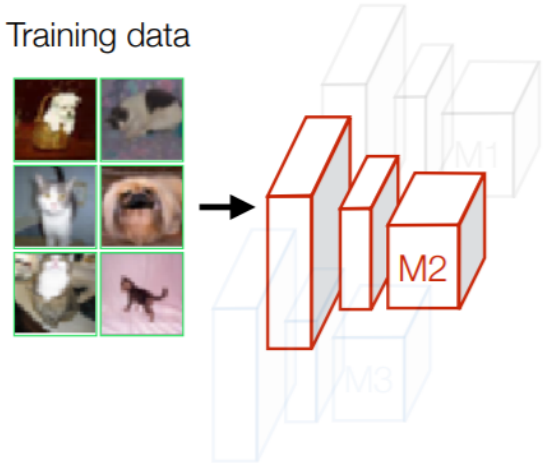
Obtaining weights

We can estimate the weight of each model based on **CV** on training data



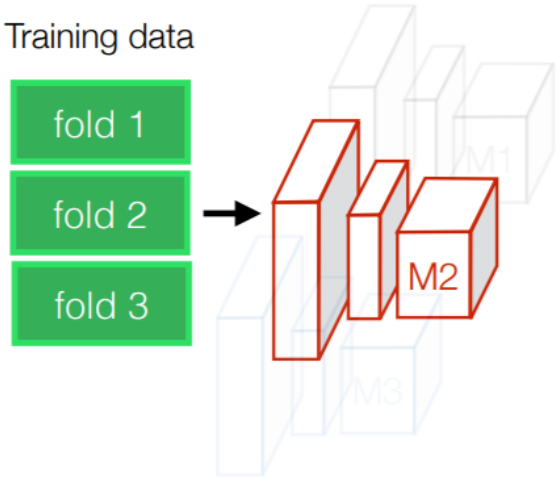
Obtaining weights

We can estimate the weight of each model based on **CV** on training data



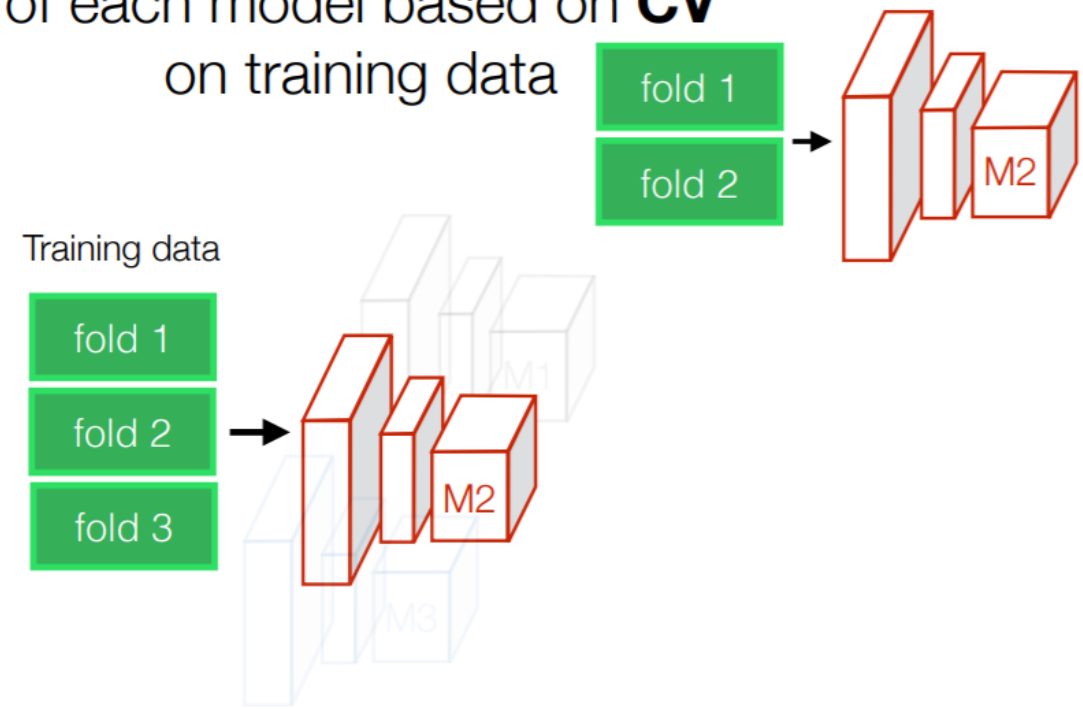
Obtaining weights

We can estimate the weight of each model based on **CV** on training data



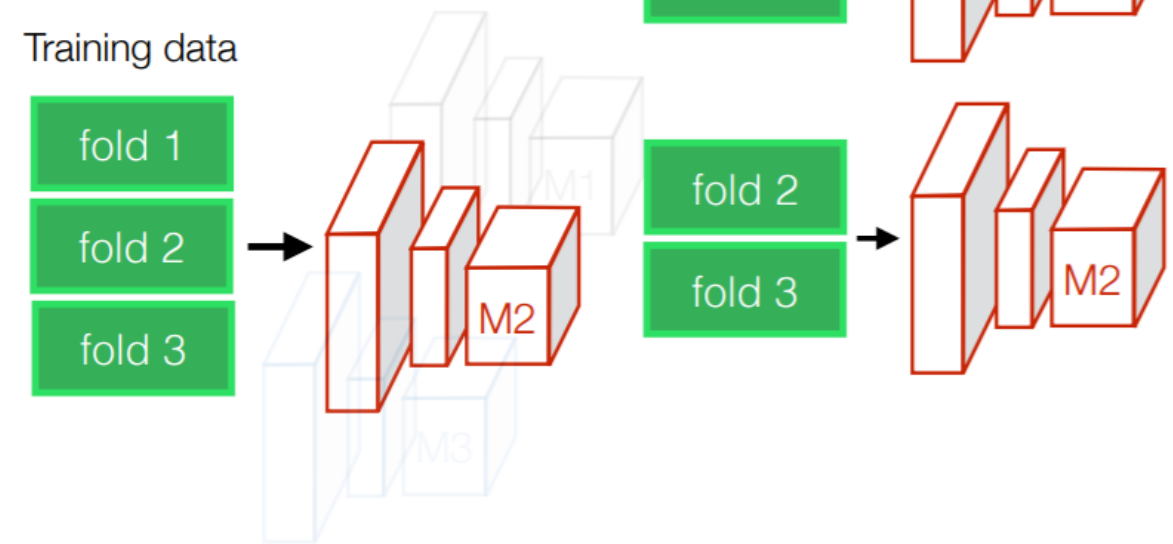
Obtaining weights

We can estimate the weight of each model based on **CV** on training data

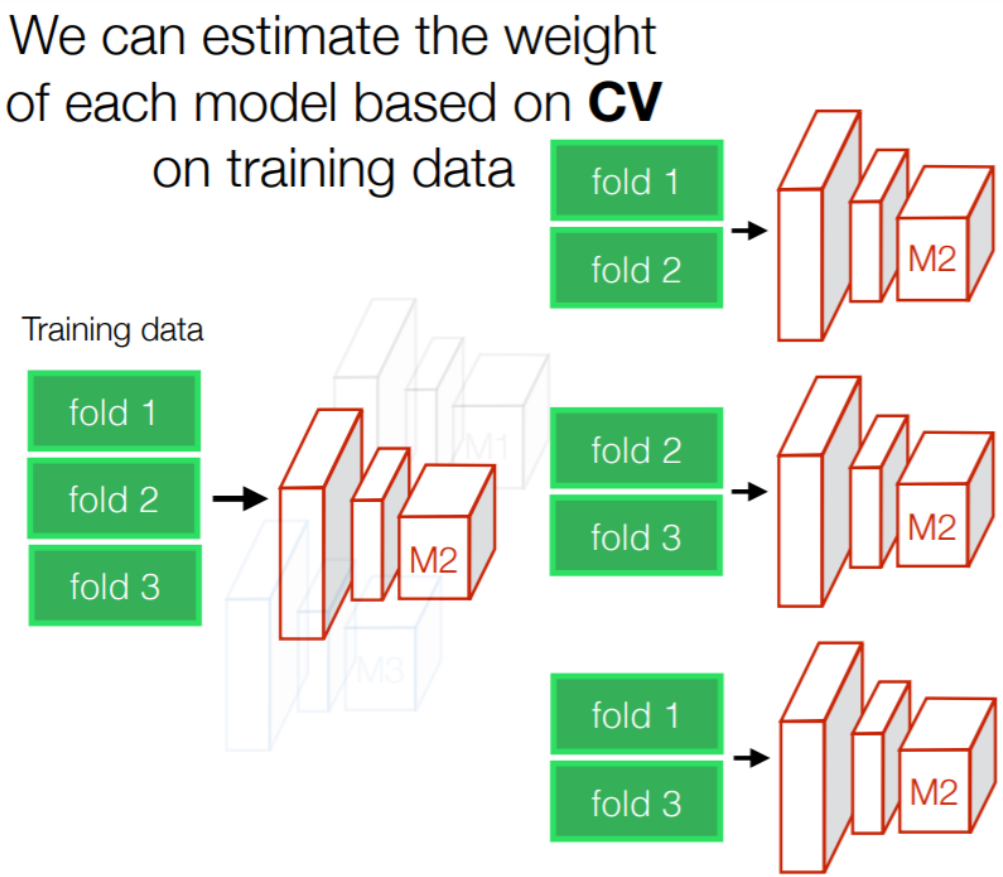


Obtaining weights

We can estimate the weight of each model based on **CV** on training data

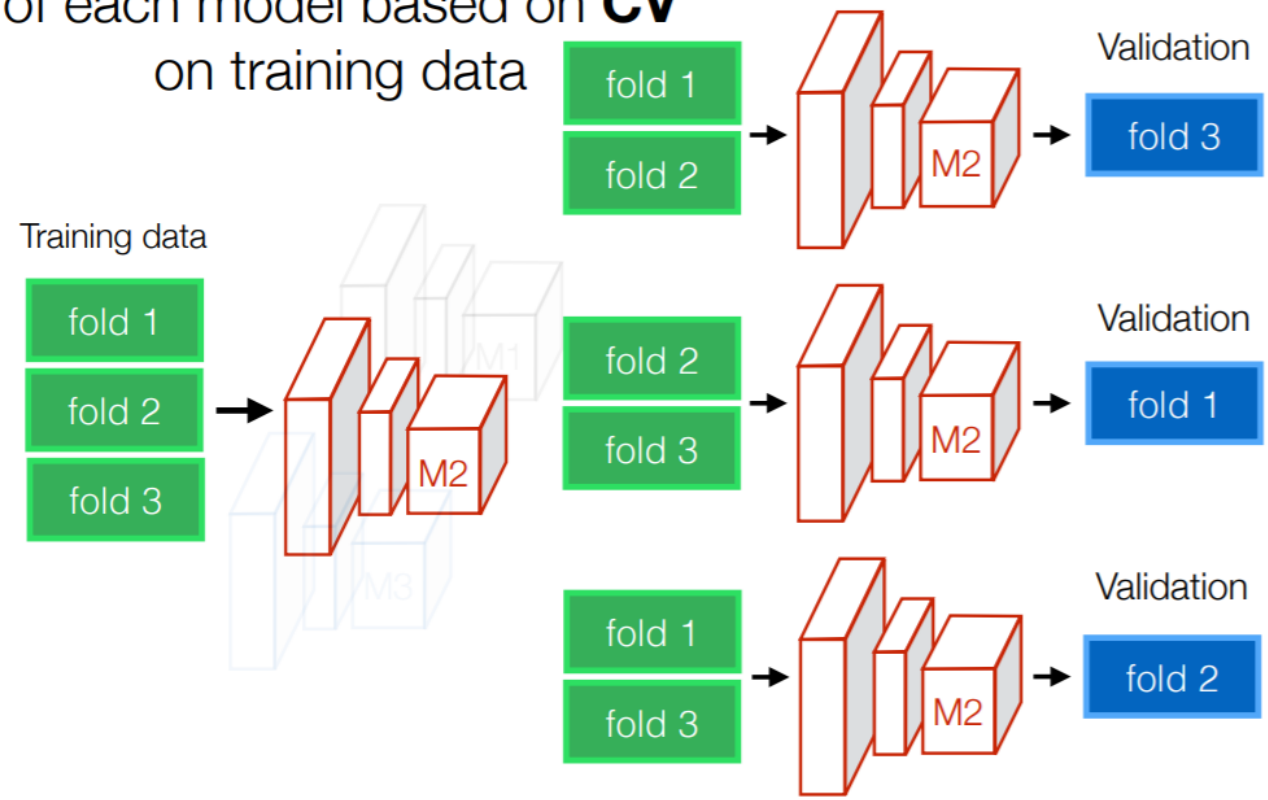


Obtaining weights

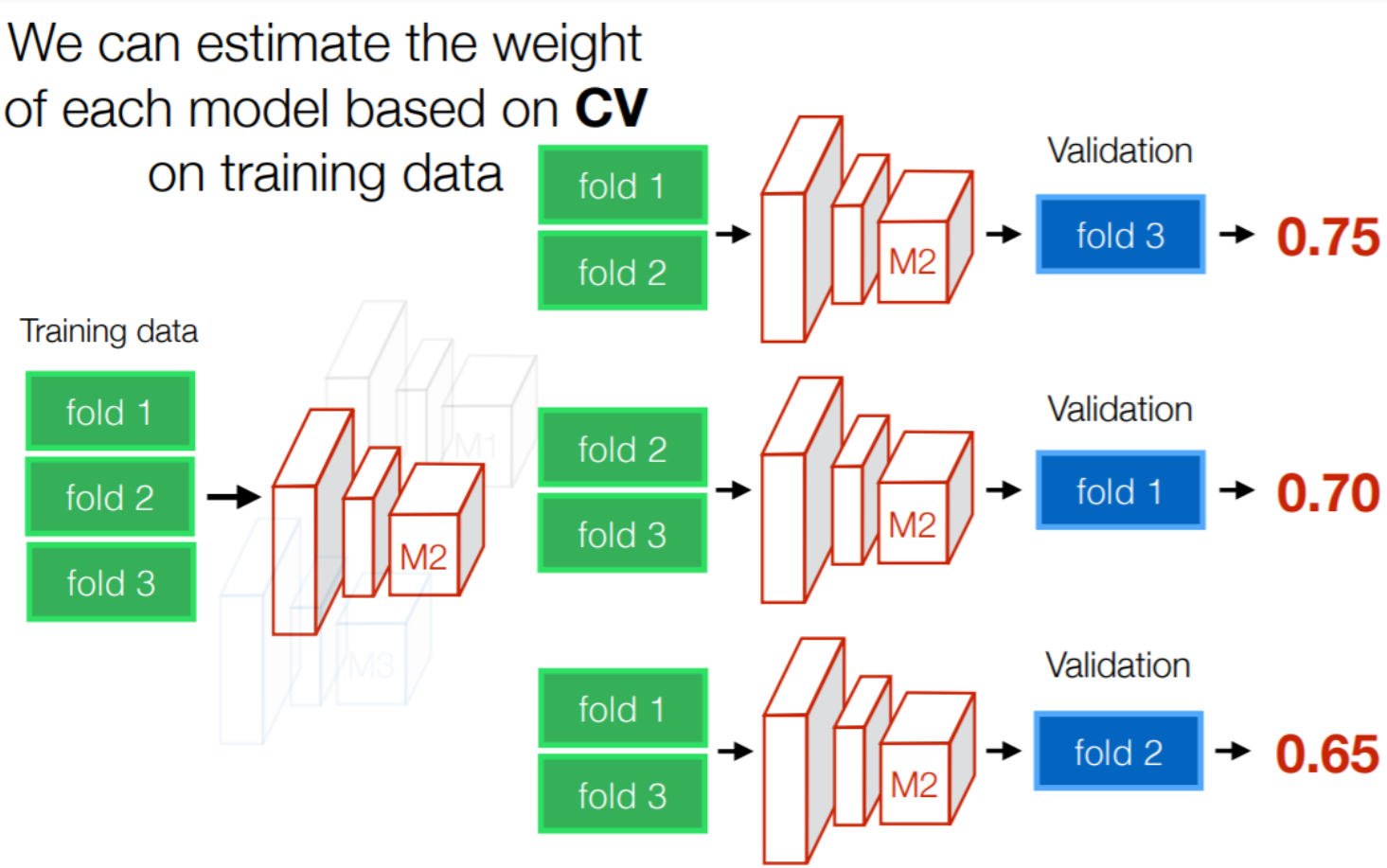


Obtaining weights

We can estimate the weight of each model based on **CV** on training data

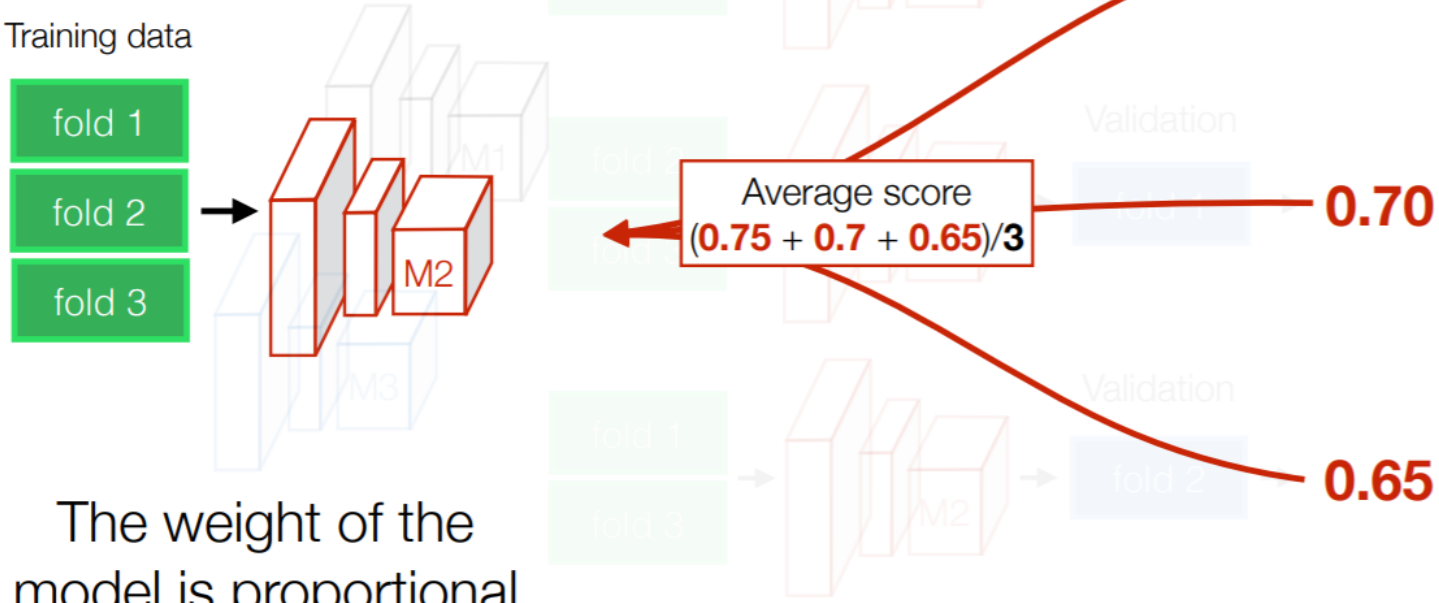


Obtaining weights



Obtaining weights

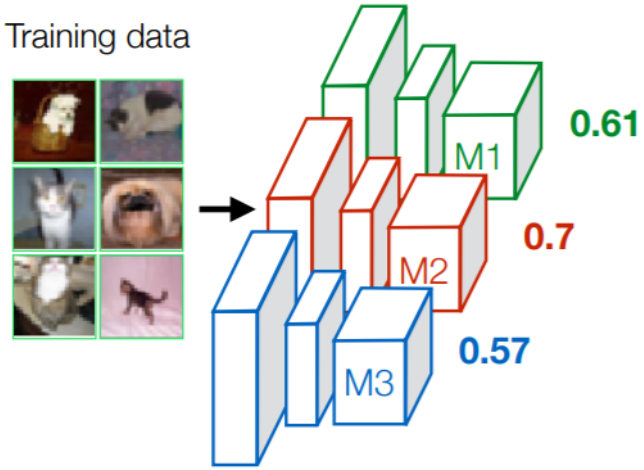
We can estimate the weight of each model based on **CV** on training data



The weight of the model is proportional to its **average score**

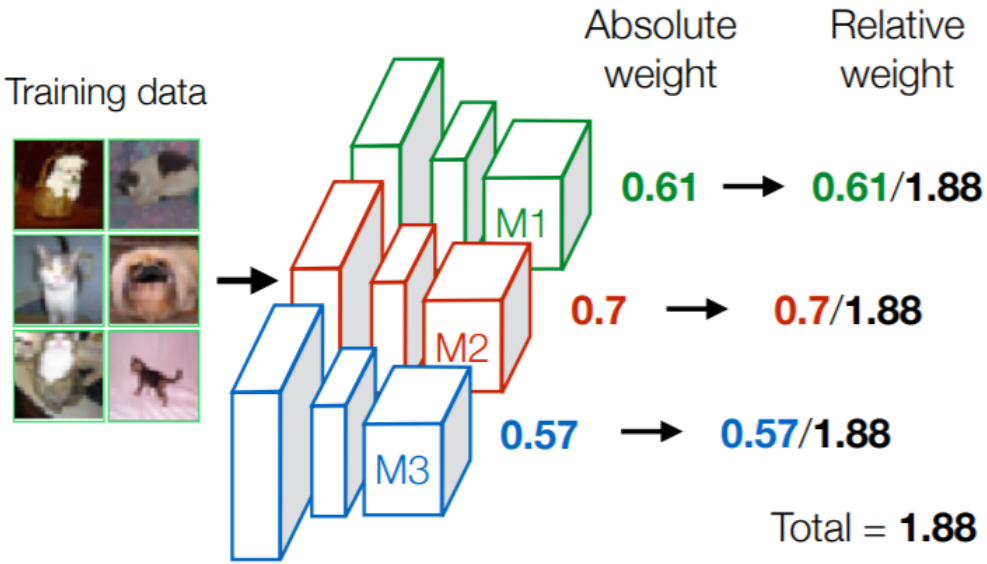
Obtaining weights

We can estimate the weight of each model based on **CV** on training data



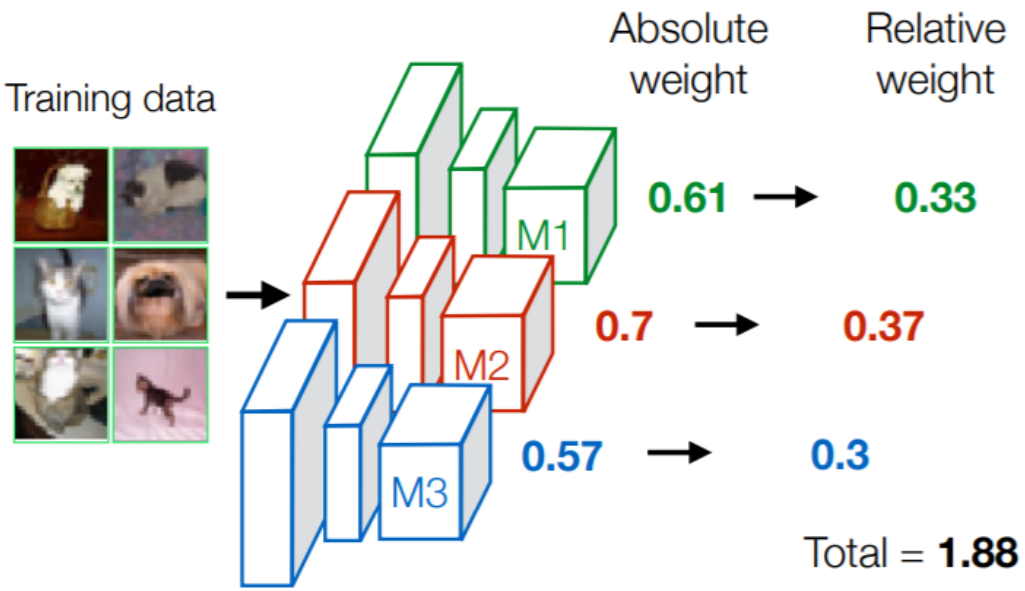
Obtaining weights

We can estimate the weight of each model based on **CV** on training data



Obtaining weights

We can estimate the weight of each model based on **CV** on training data



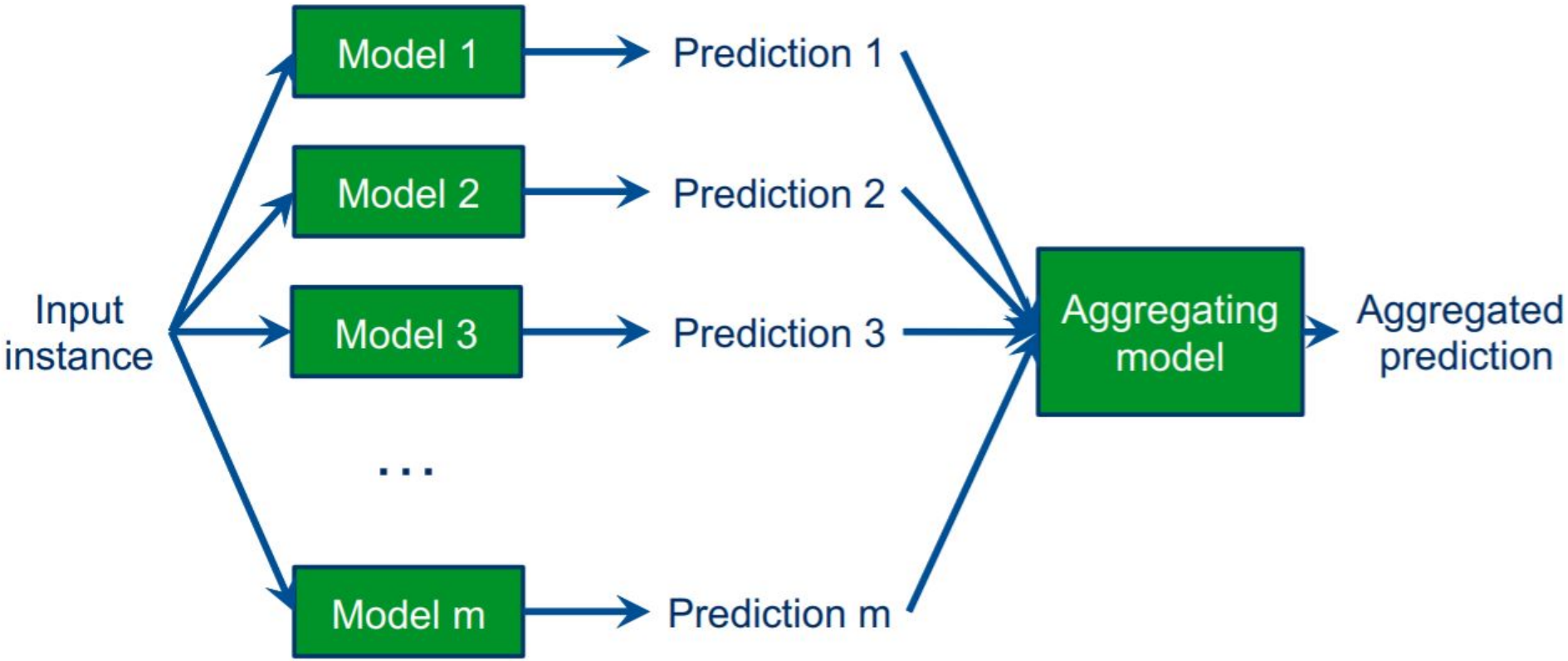
Stacking

- We can learn the weights in a linear classification task, where the individual model outputs are treated as features

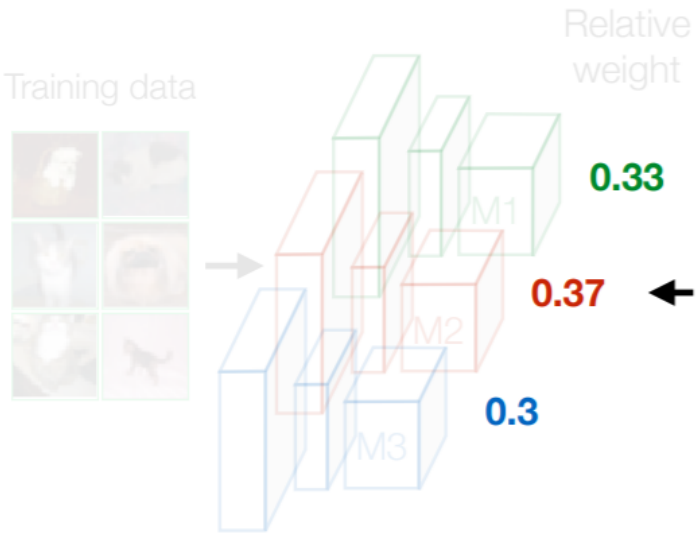
Stacking

- We can learn the weights in a linear classification task, where the individual model outputs are treated as features
- This is known as **stacking**, because we stack one classifier on top of many individual classifiers

Stacking


Input instance
Model 1
Model 2
Model 3
...
Model m
Prediction 1
Prediction 2
Prediction 3
Prediction m
Aggregating model
Aggregated prediction

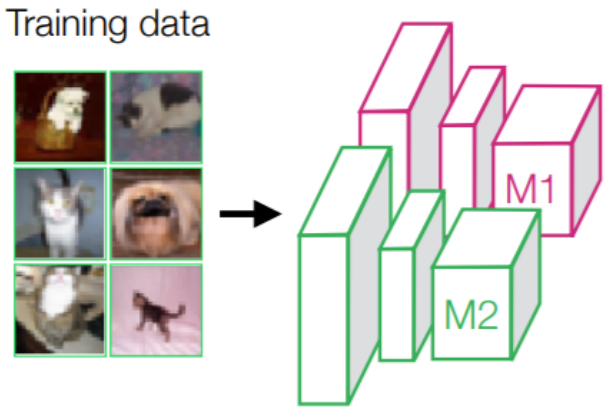
Stacking



We can train **another model** to infer these weights for **each input point separately**

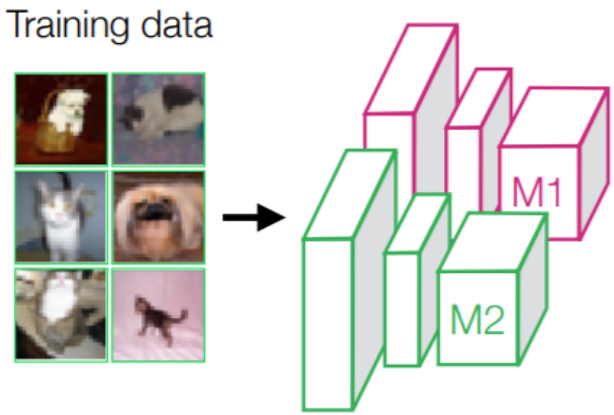
Stacking

Let's assume we have two models (**M1** and **M2**)



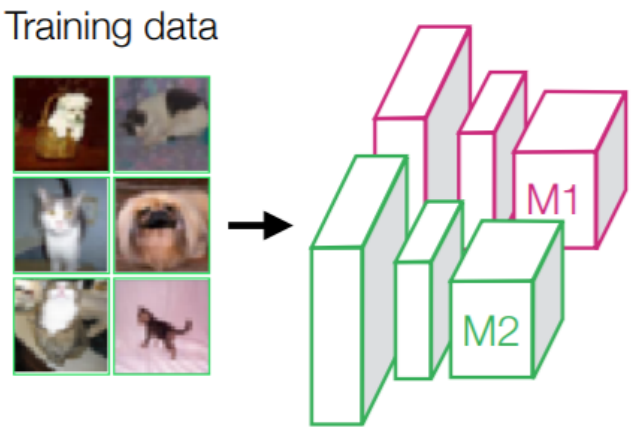
Stacking

We use models **M1** and **M2** to predict probability of class **dog** for each image

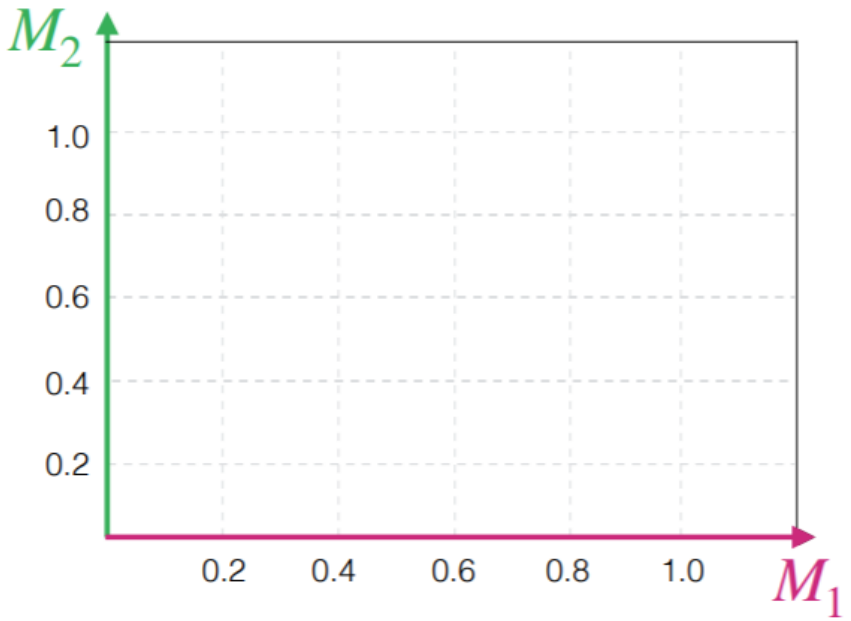


Stacking

We use models **M1** and **M2** to predict probability of class **dog** for each image



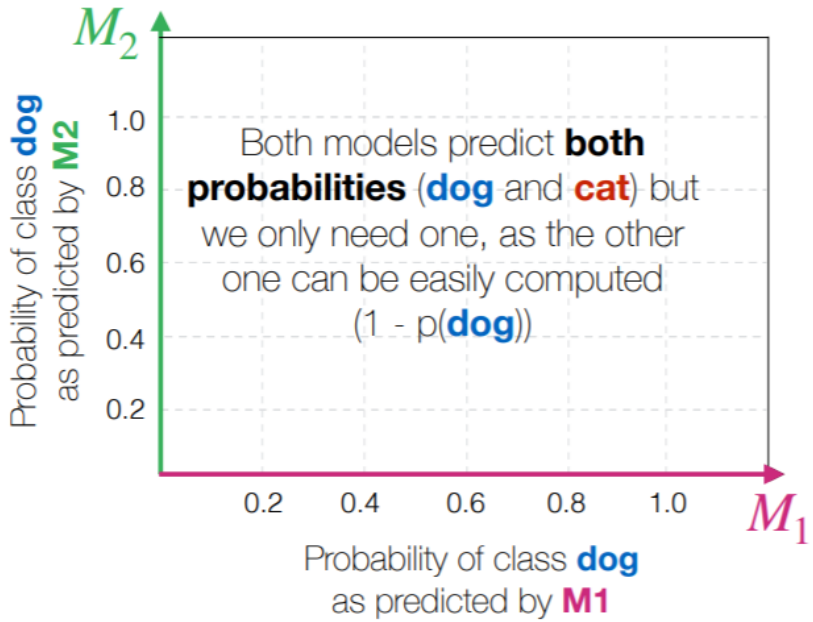
The resulting probabilities will be plotted on the graph below



Stacking

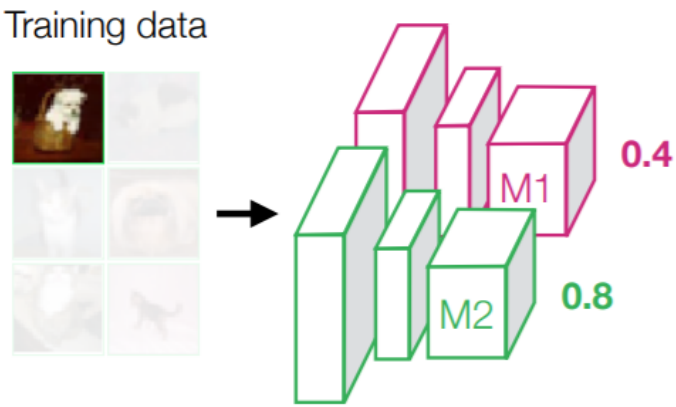
We use models **M1** and **M2** to predict probability of class **dog** for each image

The resulting probabilities will be plotted on the graph below

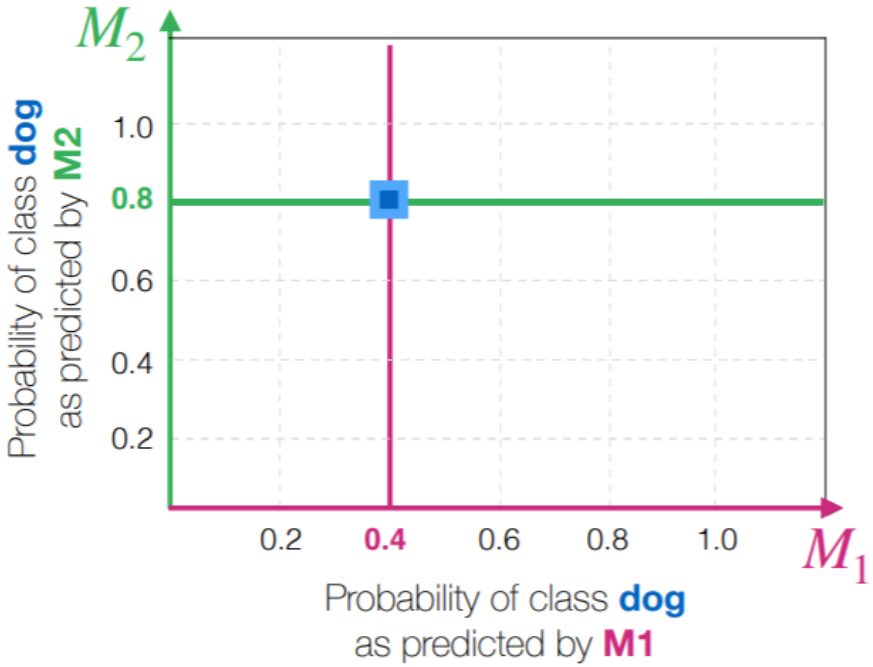


Stacking

We use models **M1** and **M2** to predict probability of class **dog** for each image

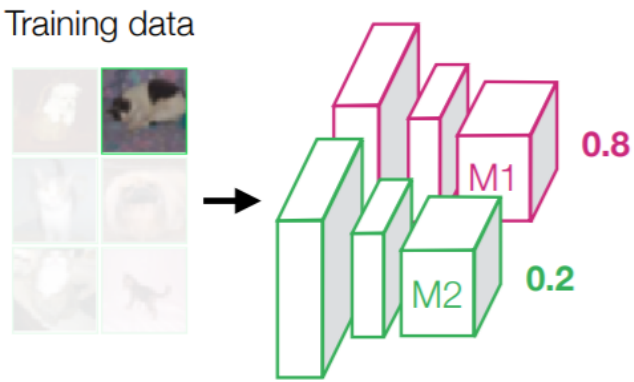


The resulting probabilities will be plotted on the graph below

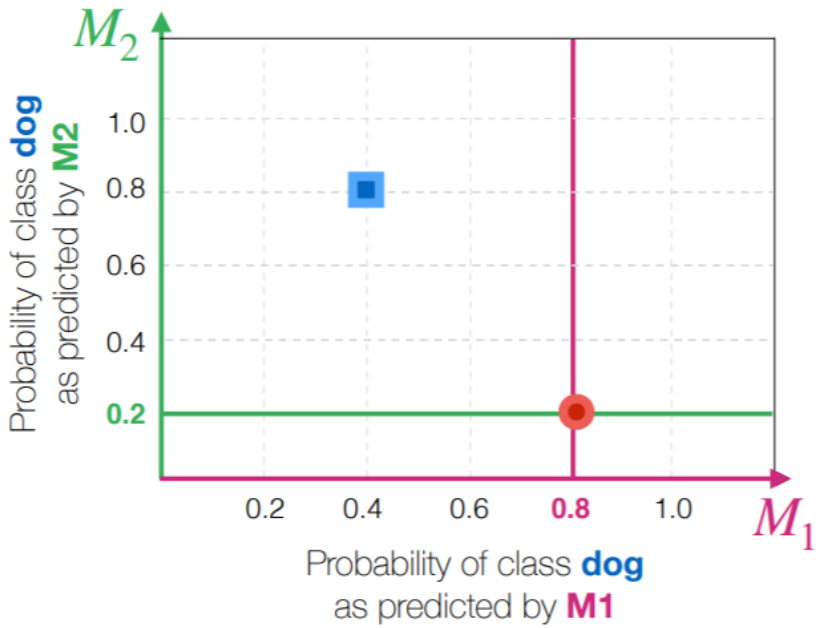


Stacking

We use models **M1** and **M2** to predict probability of class **dog** for each image

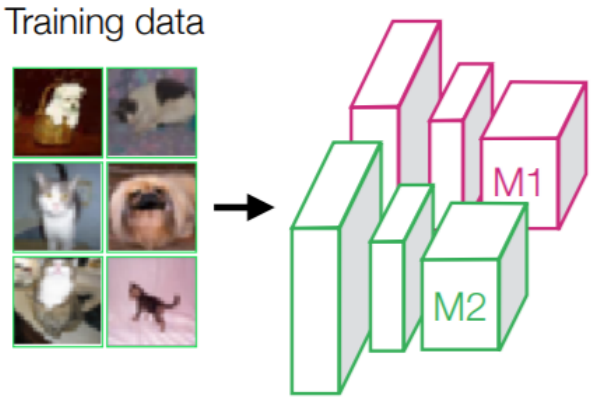


The resulting probabilities will be plotted on the graph below

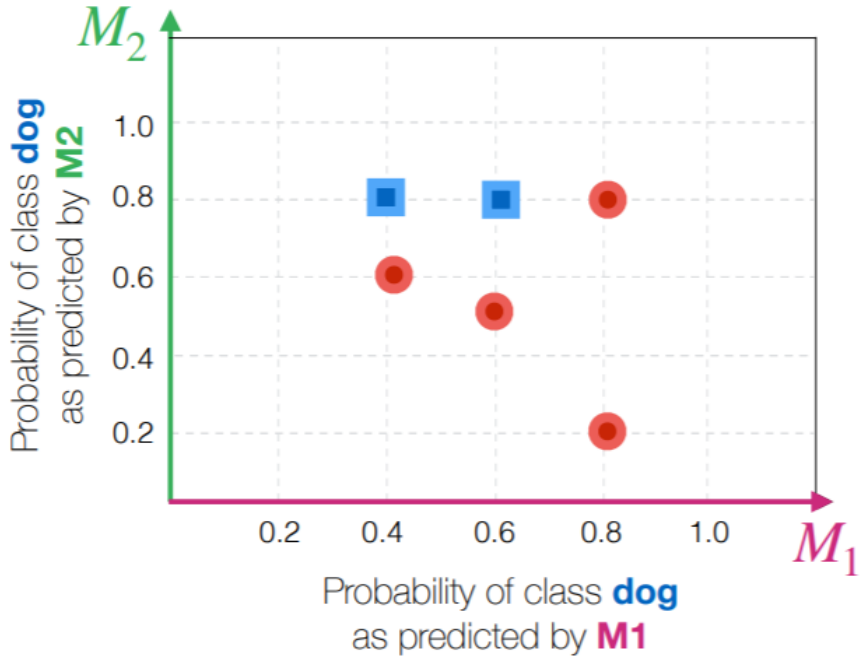


Stacking

We use models **M1** and **M2** to predict probability of class **dog** for each image

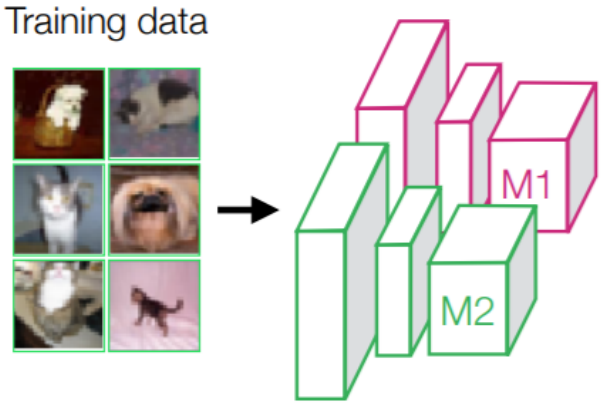


We can train a **Decision Tree** classifier on top of these results

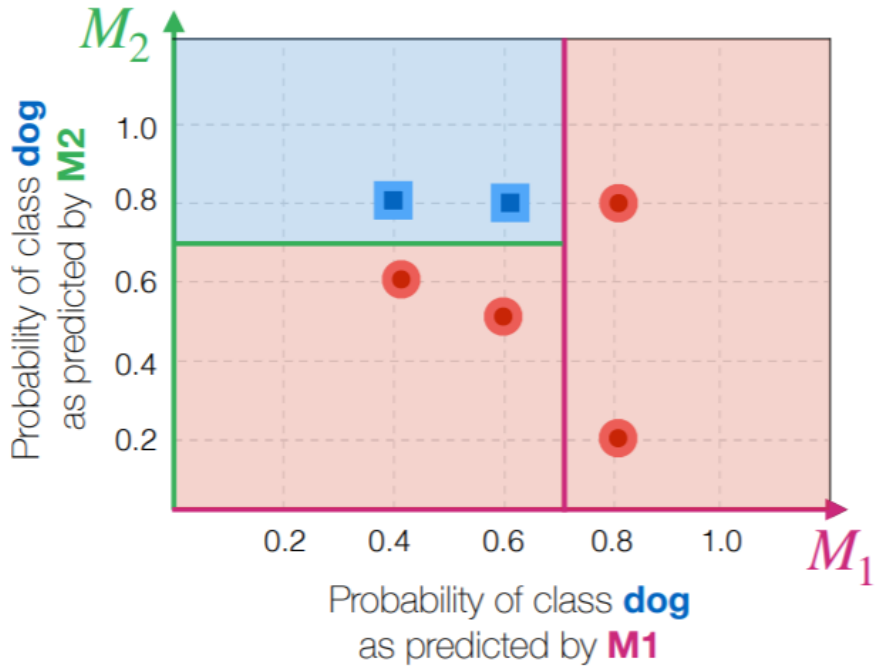


Stacking

We use models **M1** and **M2** to predict probability of class **dog** for each image

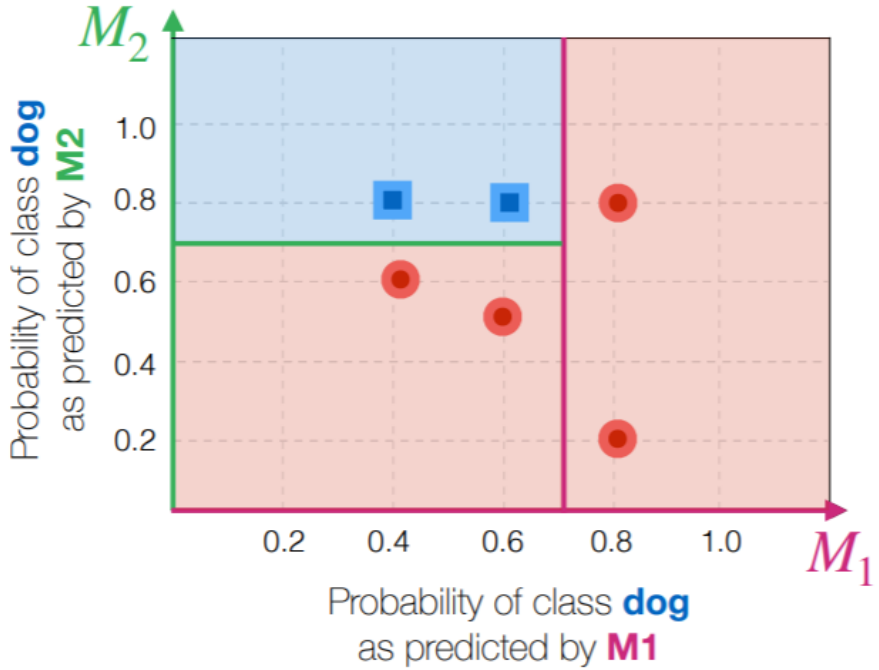


We can train a **Decision Tree** classifier on top of these results



Stacking

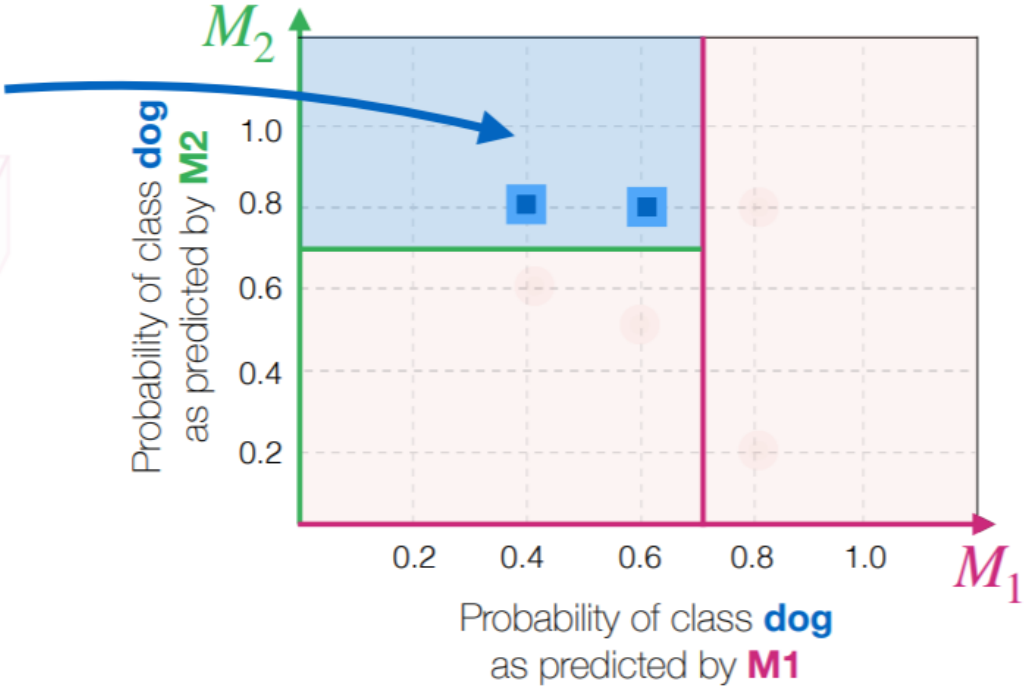
Let's see if we can interpret resulting tree (**meta-model**)



Stacking

Let's see if we can interpret resulting tree (**meta-model**)

If **M2** is confident that image is **dog** ($p > 0.7$) go with its choice

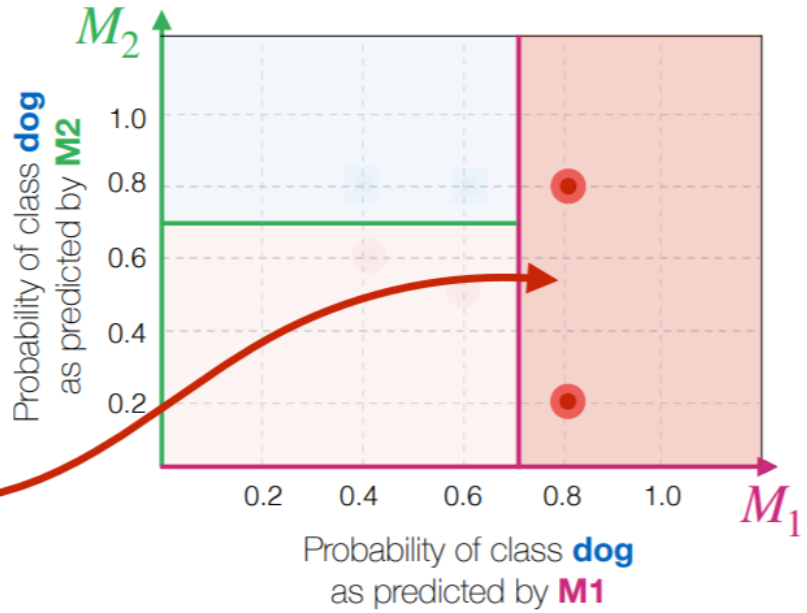


Stacking

Let's see if we can interpret resulting tree (**meta-model**)

If **M2** is confident that image is **dog** (p > 0.7) go with its choice

Unless the **M1** is also confident that this image is **dog** (p > 0.7)



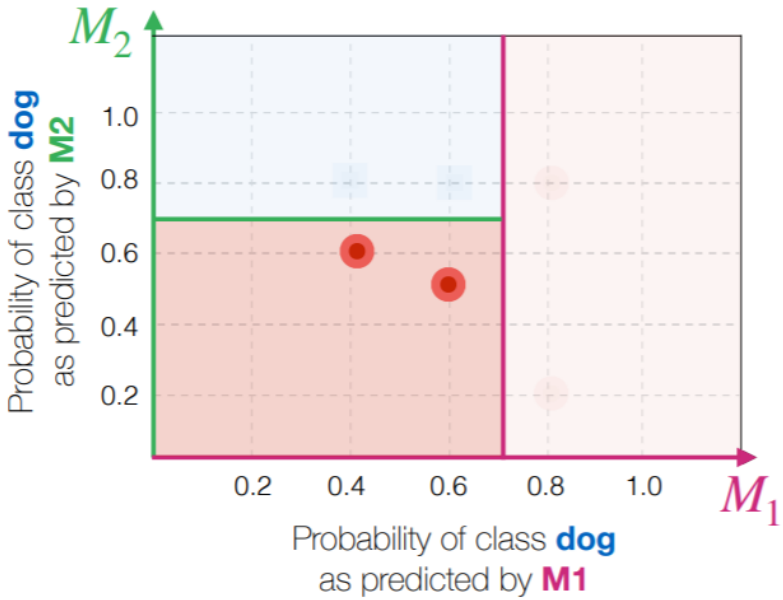
Stacking

If **M2** is confident that image is **dog** ($p > 0.7$) go with its choice

Unless the **M1** is also confident that this image is **dog** ($p > 0.7$)

If both M1 and **M2** are unconfident ($p < 0.7$), predict **cat**

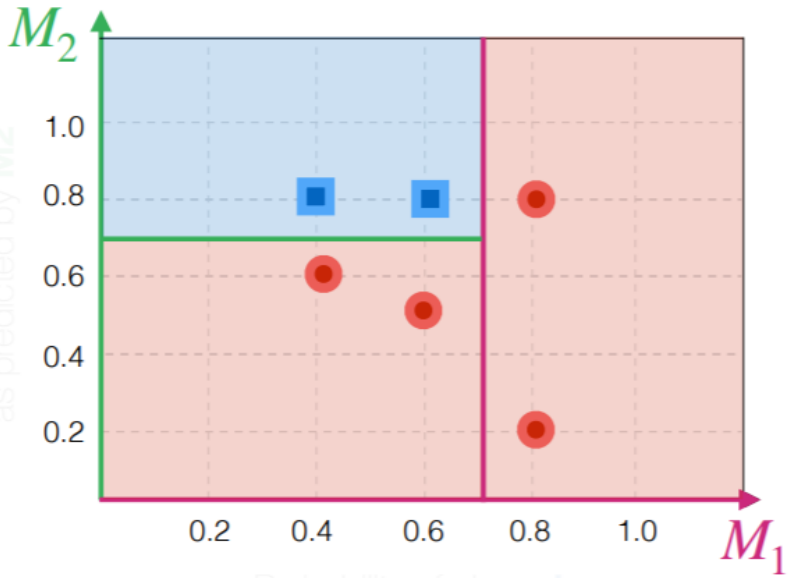
Let's see if we can interpret resulting tree (**meta-model**)



Stacking

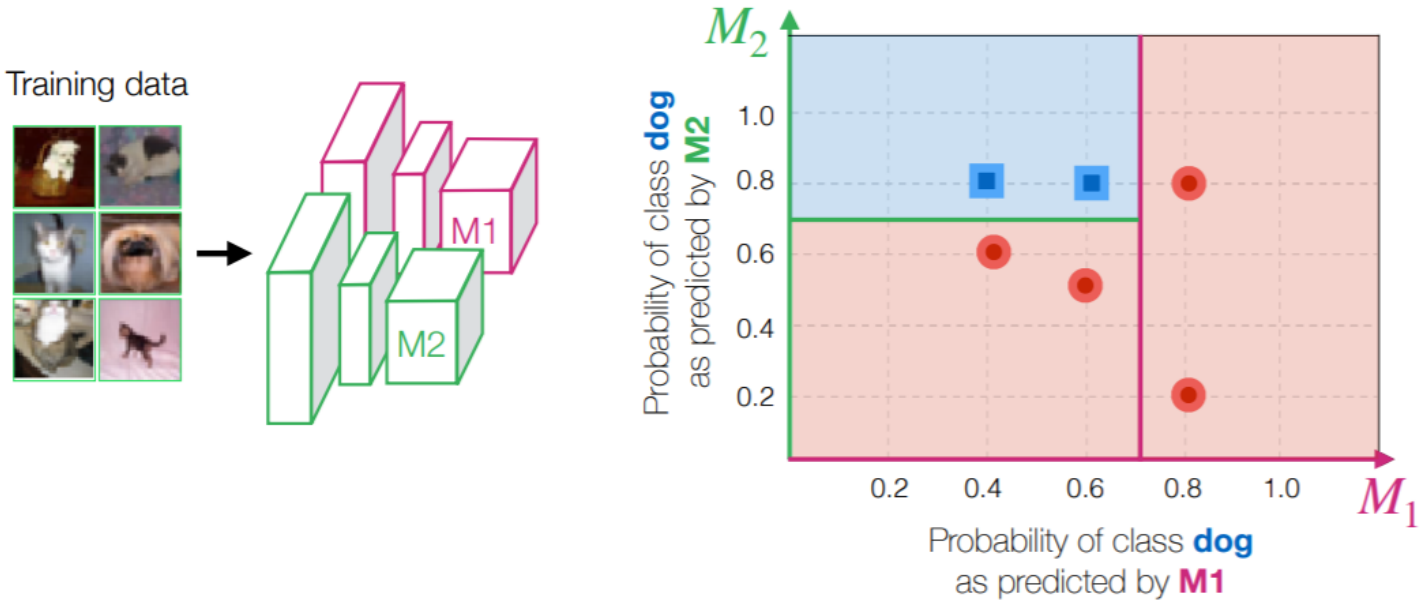
Training an additional model (**meta-model**) on top of ensemble predictions is called **stacking**

The **stacking model** can **be any model** that you find most appropriate for your case



Stacking

Training an additional model (**meta-model**) on top of ensemble predictions is called **stacking**

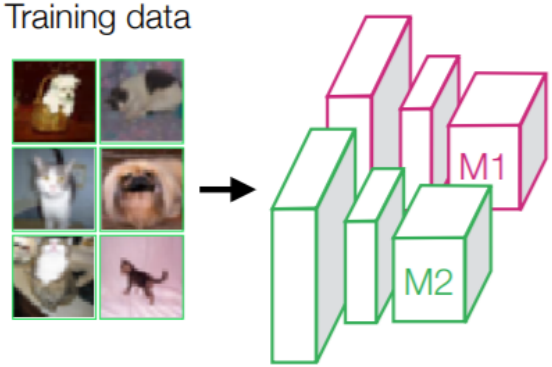


There is **one problem** with the setup I have shown you here

Stacking

Probability scores obtained on training data will be **overfitted**

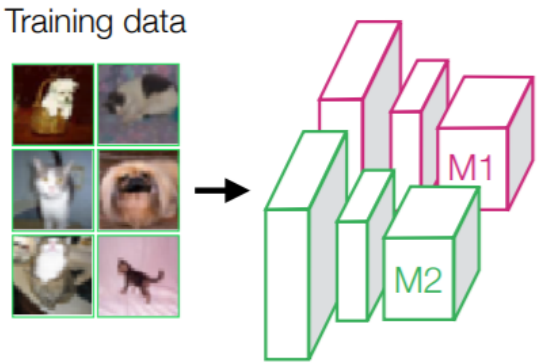
Training data



There is **one problem** with the setup I have shown you here

Stacking

Probability scores obtained on training data will be **overfitted**

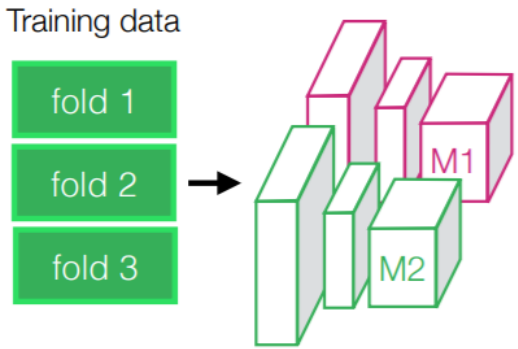


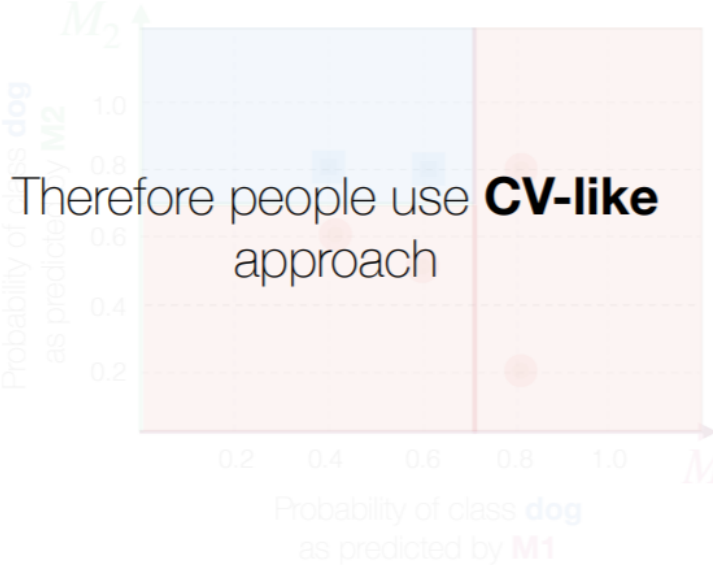
Therefore people use **CV-like** approach

There is **one problem** with the setup I have shown you here

Stacking

Training an additional model (**meta-model**) on top of
Data is randomly split into folds s is called **stacking**

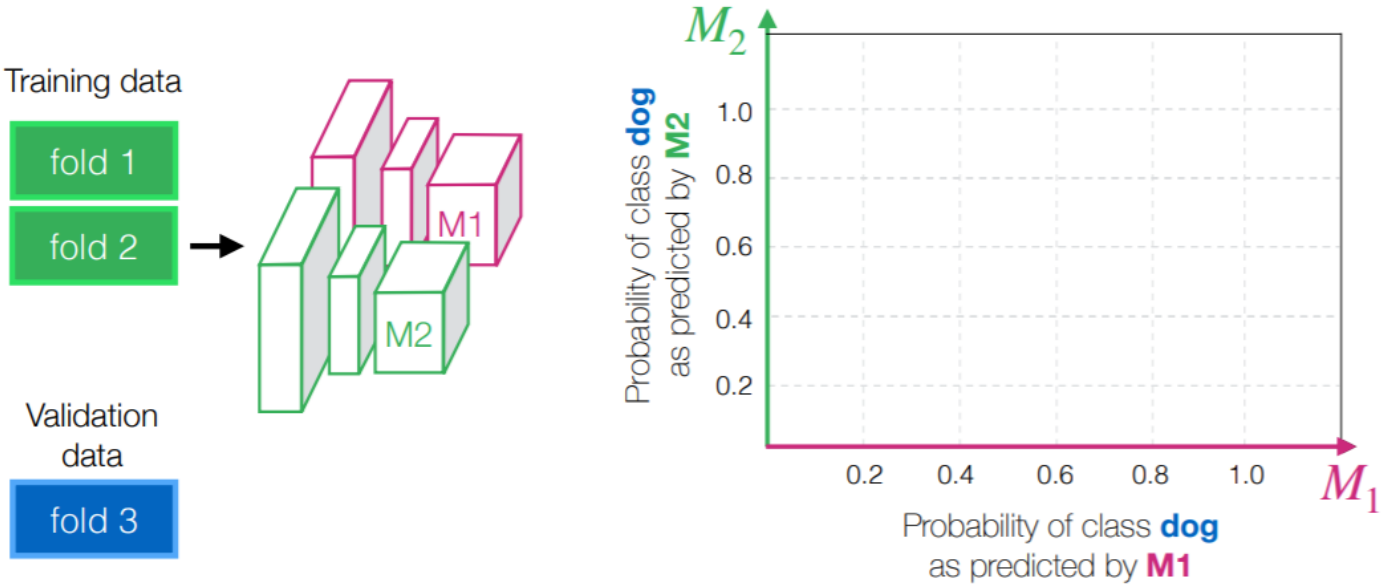




Stacking

Training an additional model (**meta-model**) on top of the predictions is called **stacking**

Data is randomly split into folds

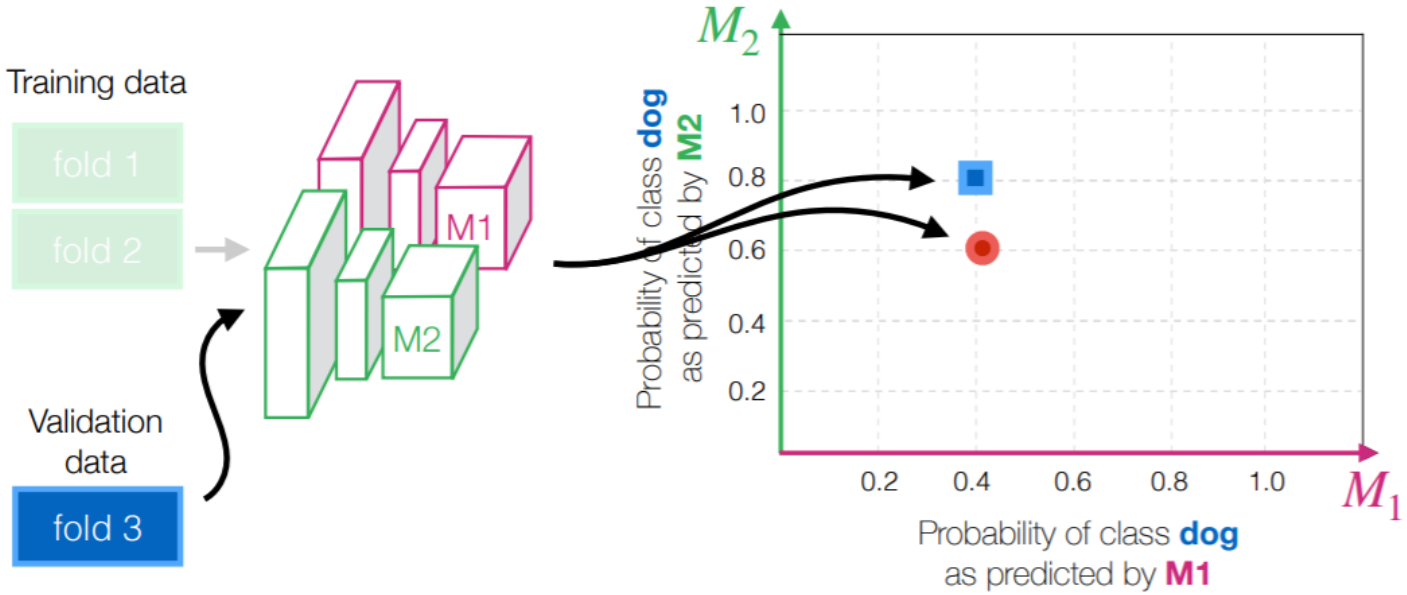


While models are trained on training data, the probability scores are generated based on validation set

Stacking

Training an additional model (**meta-model**) on top of [illegible] is called **stacking**

Data is randomly split into folds

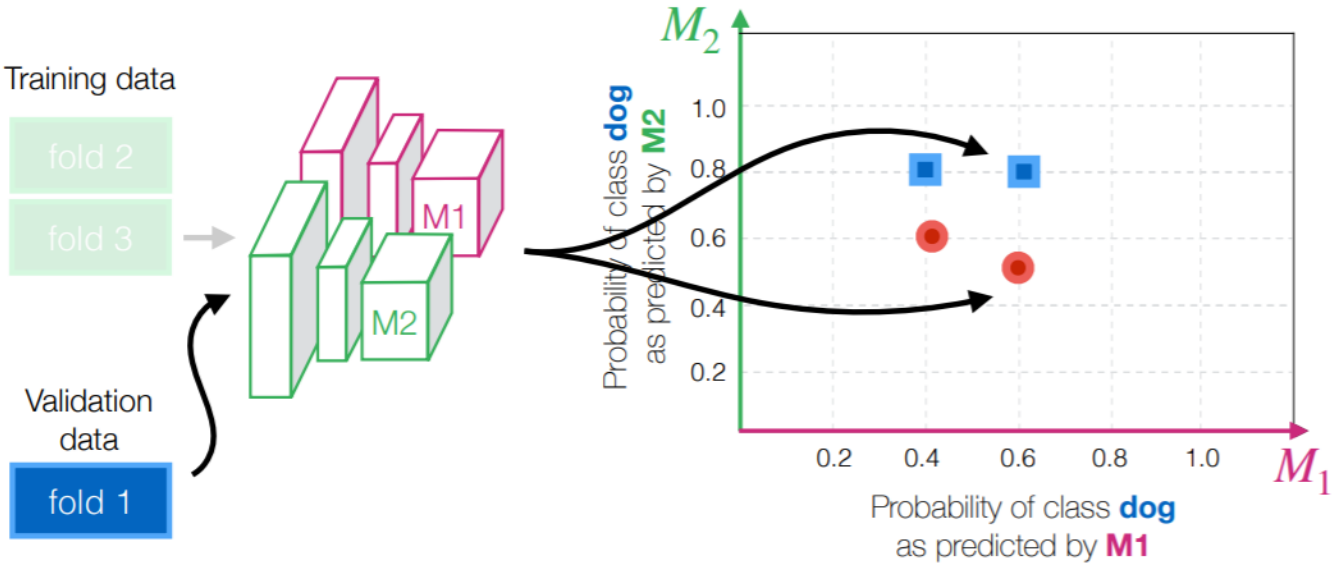


While models are trained on training data, the probability scores are generated based on validation set

Stacking

Training an additional model (**meta-model**) on top of ... is called **stacking**

Data is randomly split into folds

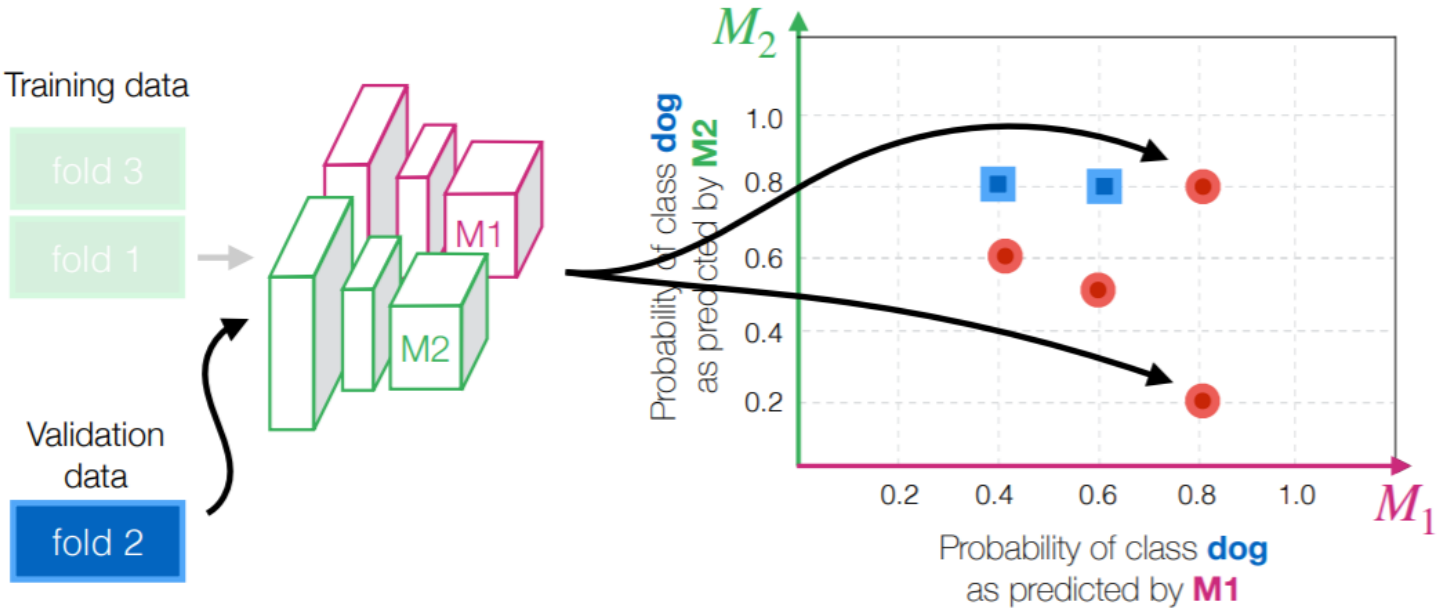


While models are trained on training data, the probability scores are generated based on validation set

Stacking

Training an additional model (**meta-model**) on top of [illegible] is called **stacking**

Data is randomly split into folds



While models are trained on training data, the probability scores are generated based on validation set

What have we learned today?

- ✓ Bagging
- ✓ Random Forest
- ✓ Weighted Voting
- ✓ Stacking