Ensemble Learning

Hard and soft voting, bagging, stacking

Introduction

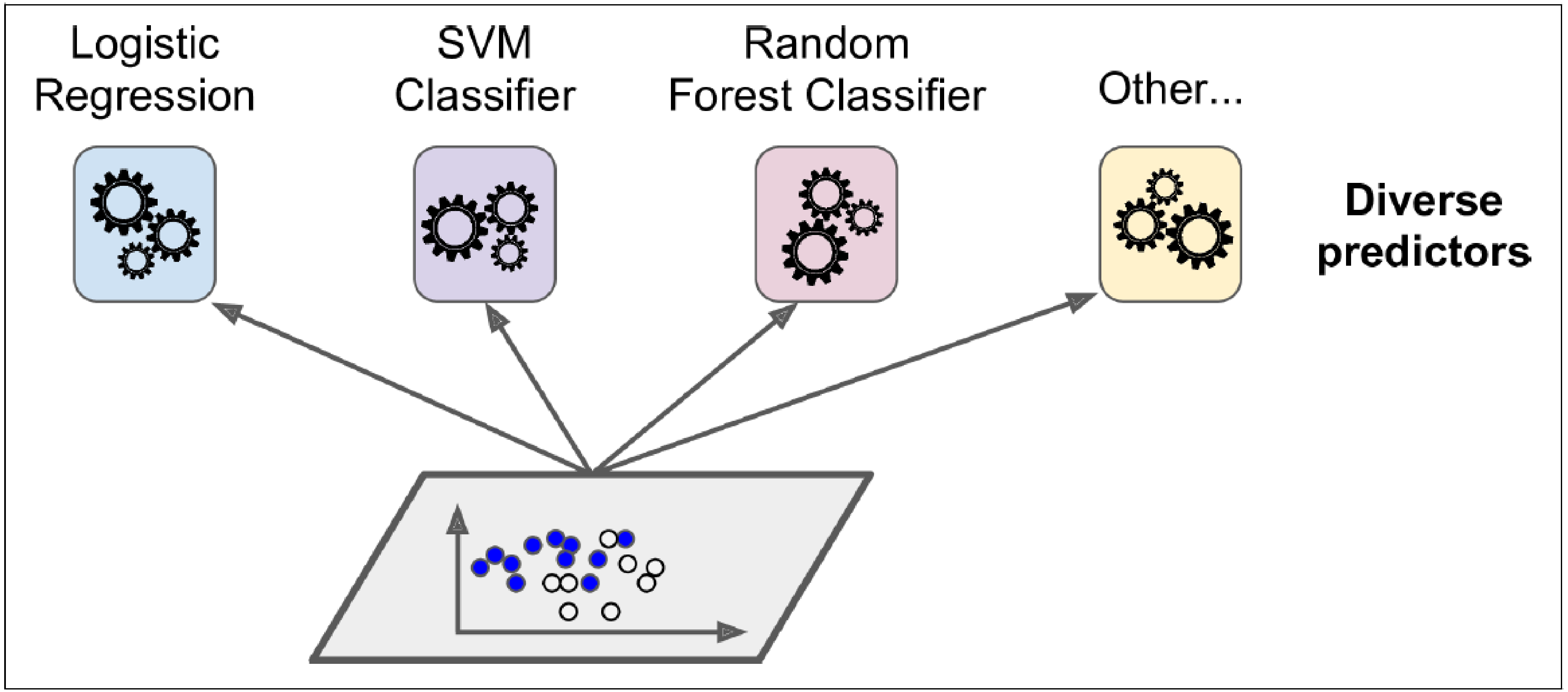
- the wisdom of the crowd.
- If you aggregate the predictions of a group of predictors (such as classifiers or regressors), you will often get better predictions than with the best individual predictor.
- A group of predictors is called an ensemble; thus, this technique is called Ensemble Learning
- An Ensemble Learning algorithm is called an Ensemble method.

Introduction(contd…)

Ensemble methods work best when the predictors are as independent from one another as possible. One way to get diverse classifiers is to train them using very different algorithms. This increases the chance that they will make very different types of errors, improving the ensemble’s accuracy.

Voting Classifiers

Training diverse classifiers



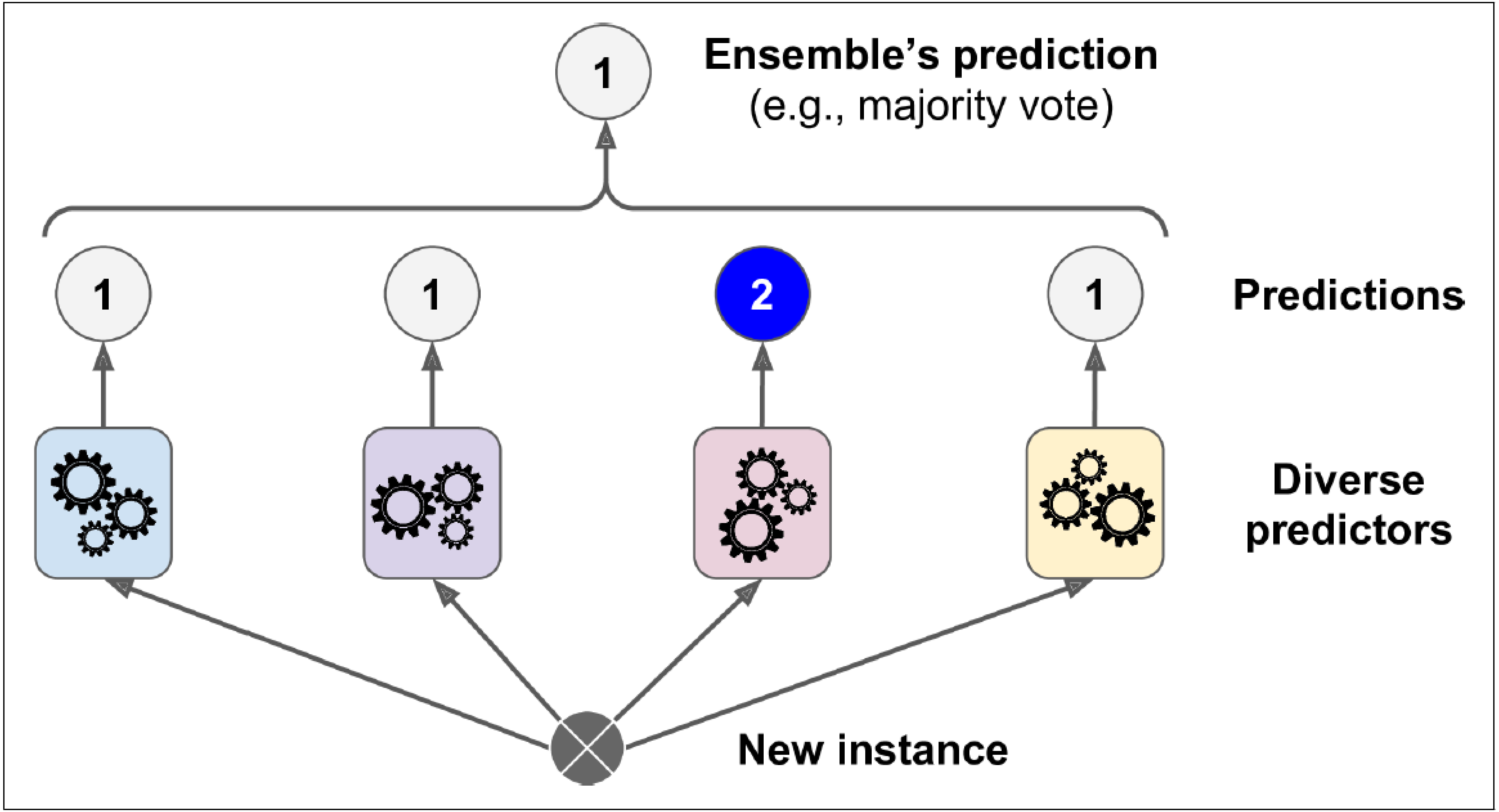
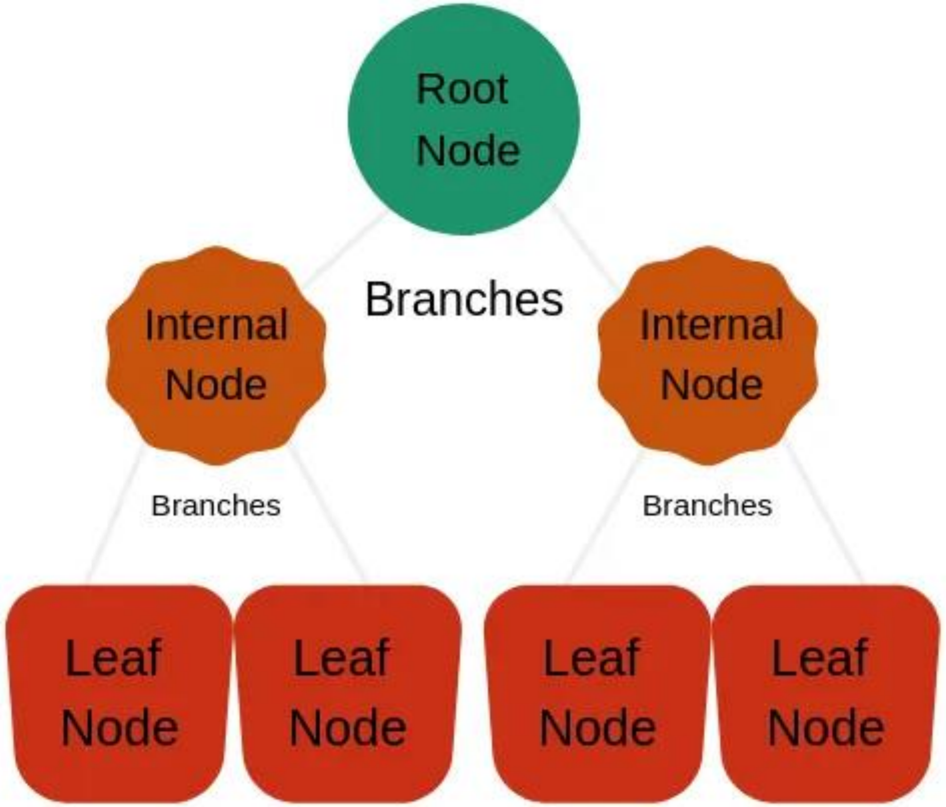


Figure 7-2. Hard voting classifier predictions

Soft voting

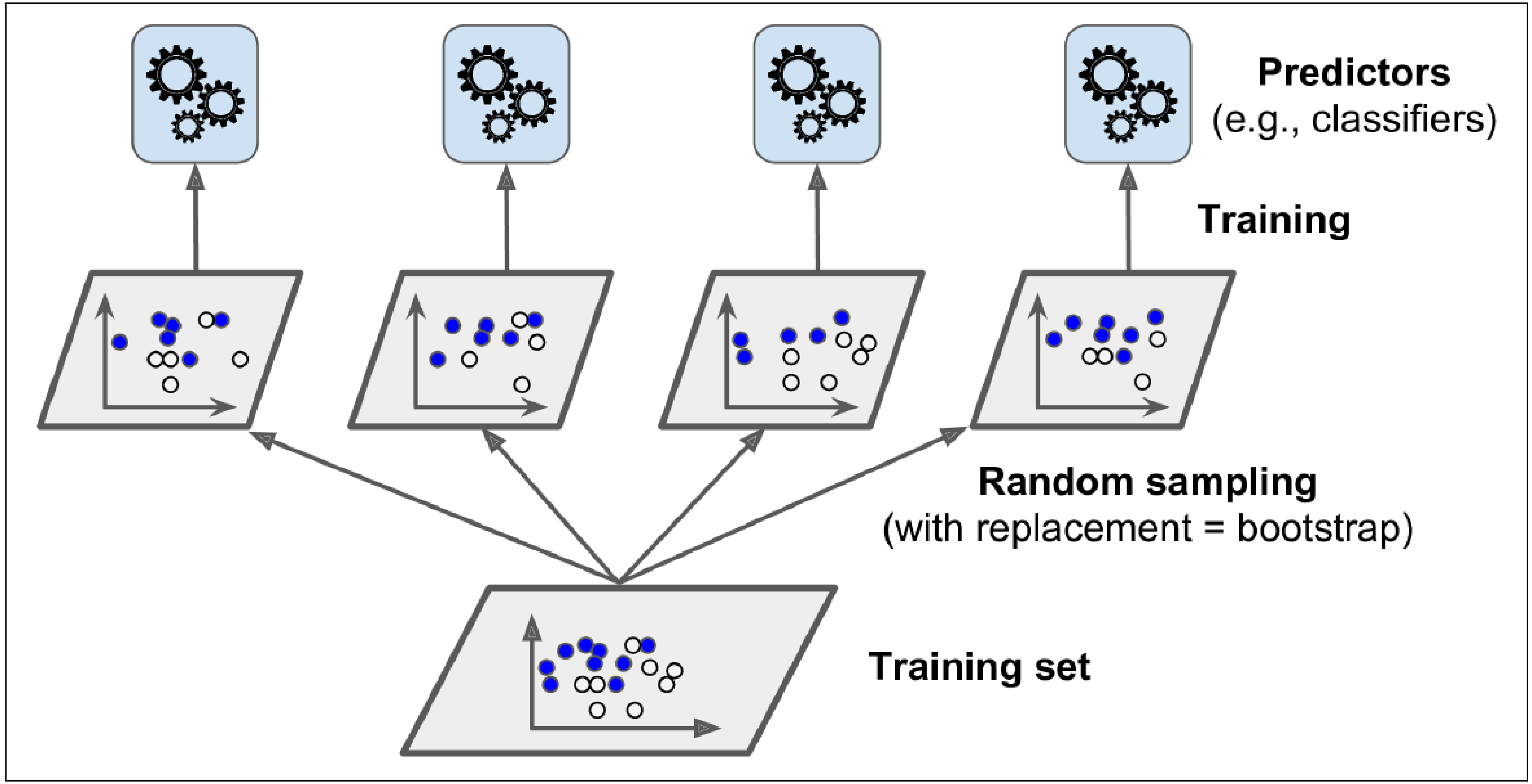
If all classifiers are able to estimate class probabilities , then you can predict the class with the highest class probability, averaged over all the individual classifiers. This is called *soft voting*. It often achieves higher performance than hard voting because it gives more weight to highly confident votes

Downside of Decision Tree

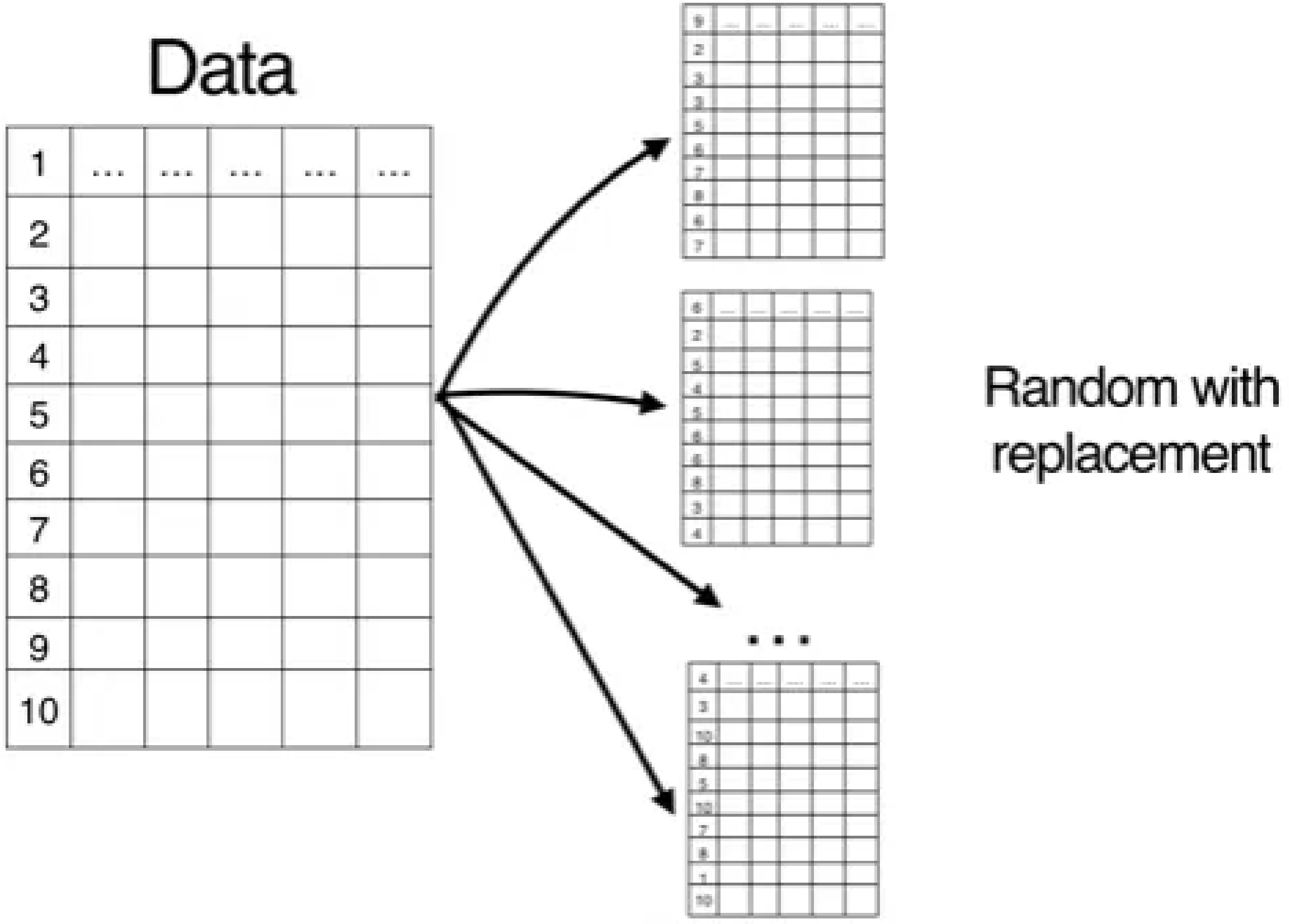


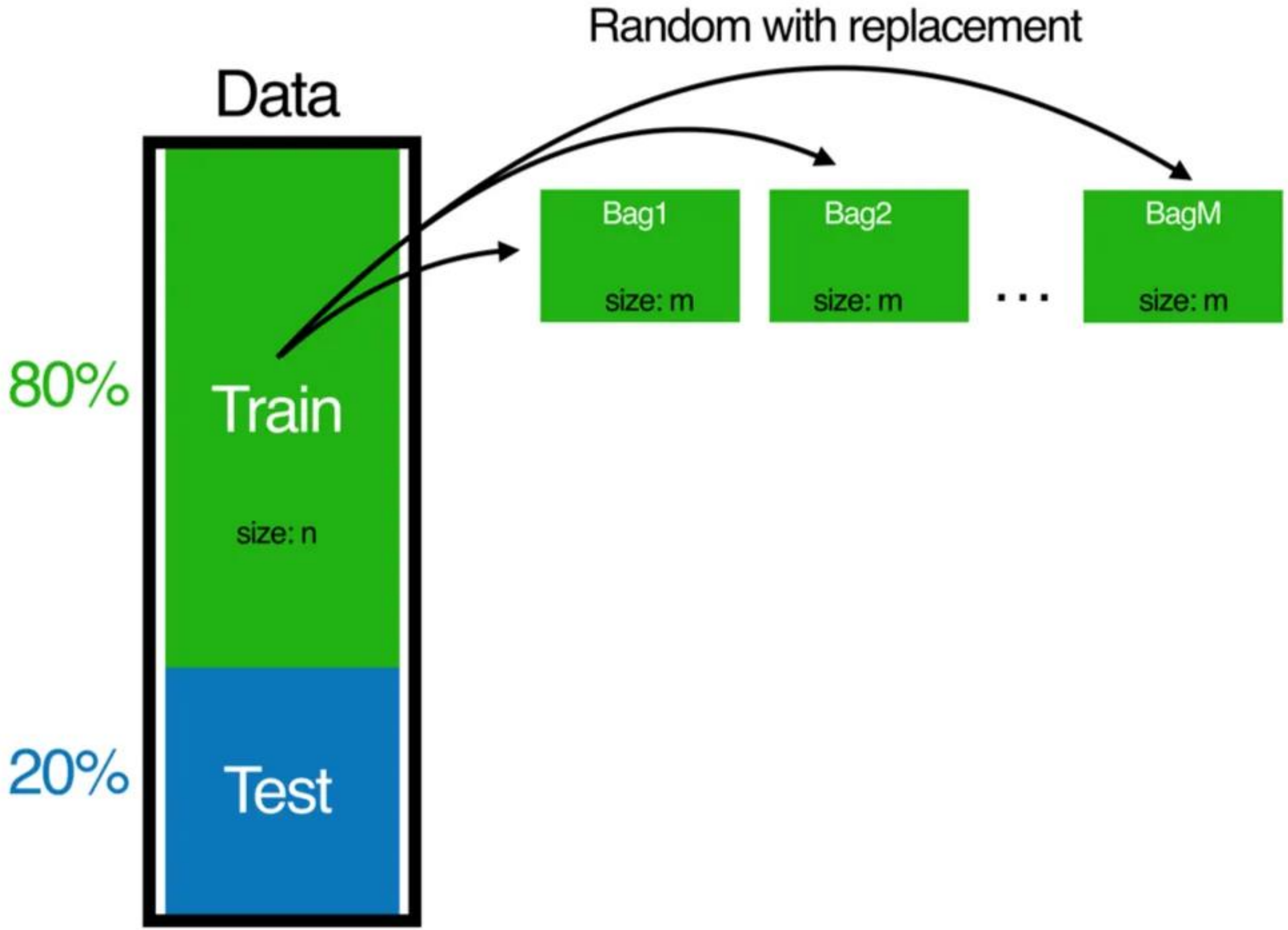
- Overfitting → High-variance
- Sensitive to data distribution
- The solution is Bagging

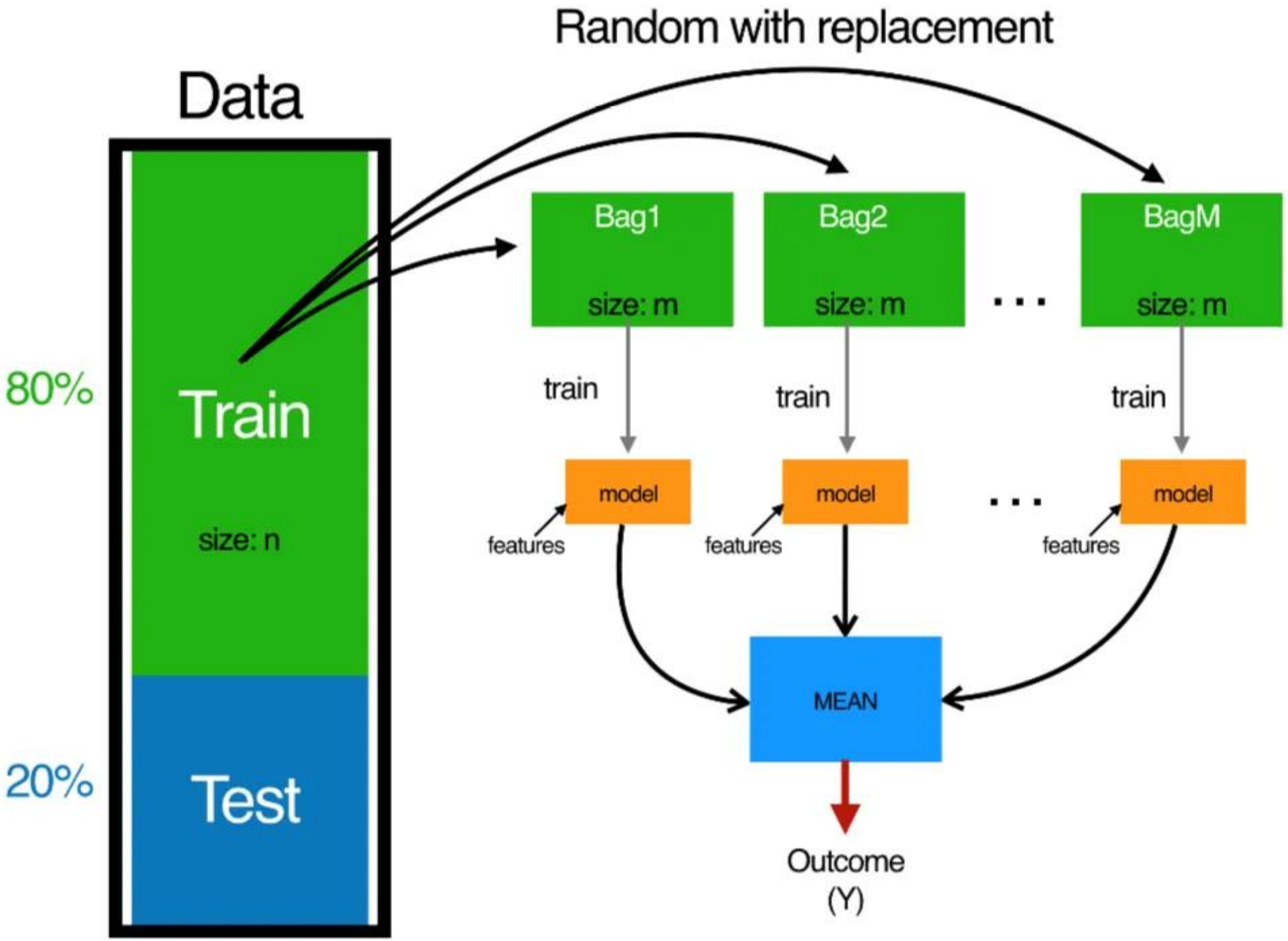
Bagging


Predictors
(e.g., classifiers)
Training
Random sampling
(with replacement = bootstrap)
Training set

Bootstrap sampling simply means sampling rows at random from the training dataset with replacement.



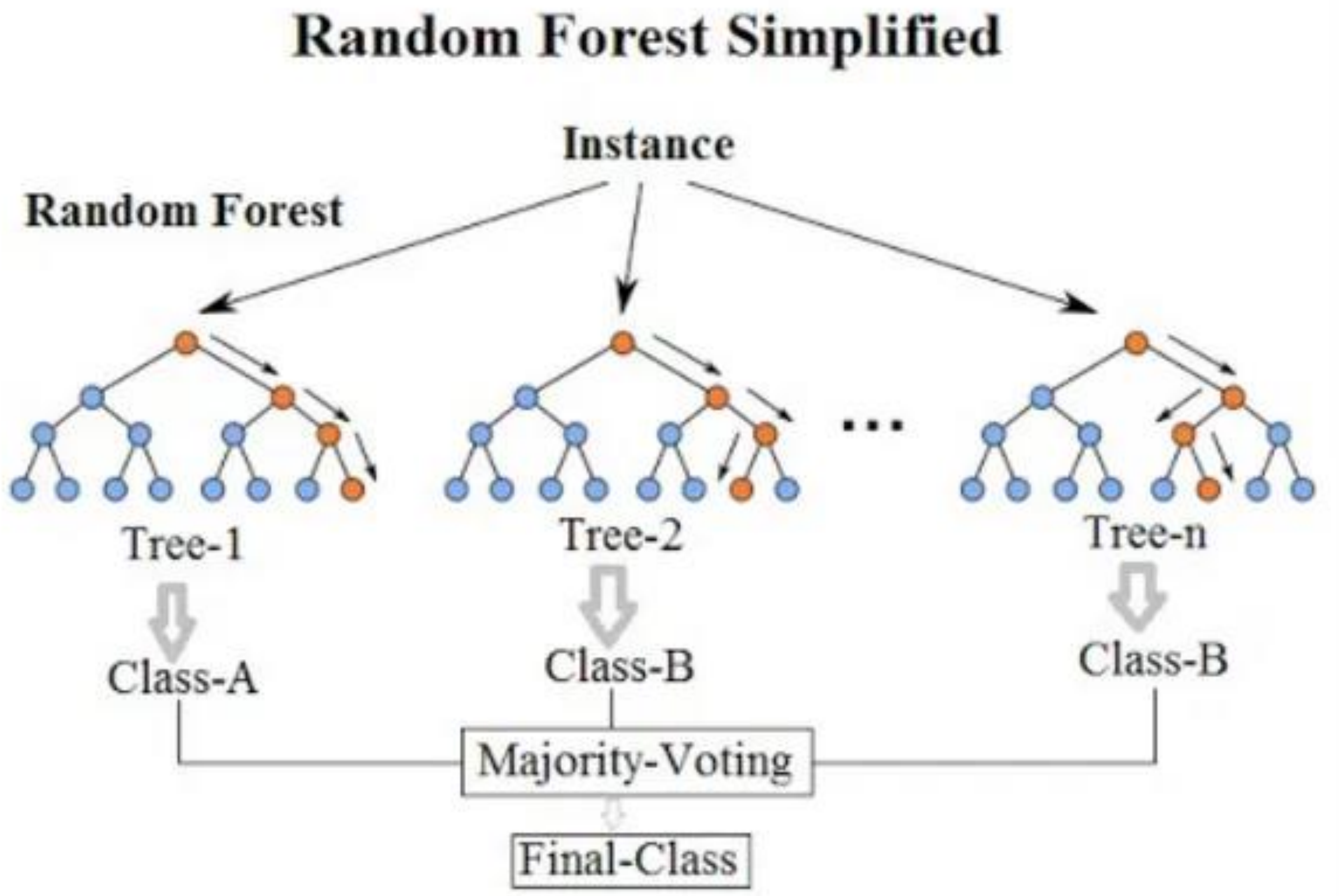

Random with replacement
Data
80%
Train
size: n
20%
Test
Bag1
size: m
Bag2
size: m
...
BagM
size: m



Low variance

Low bias

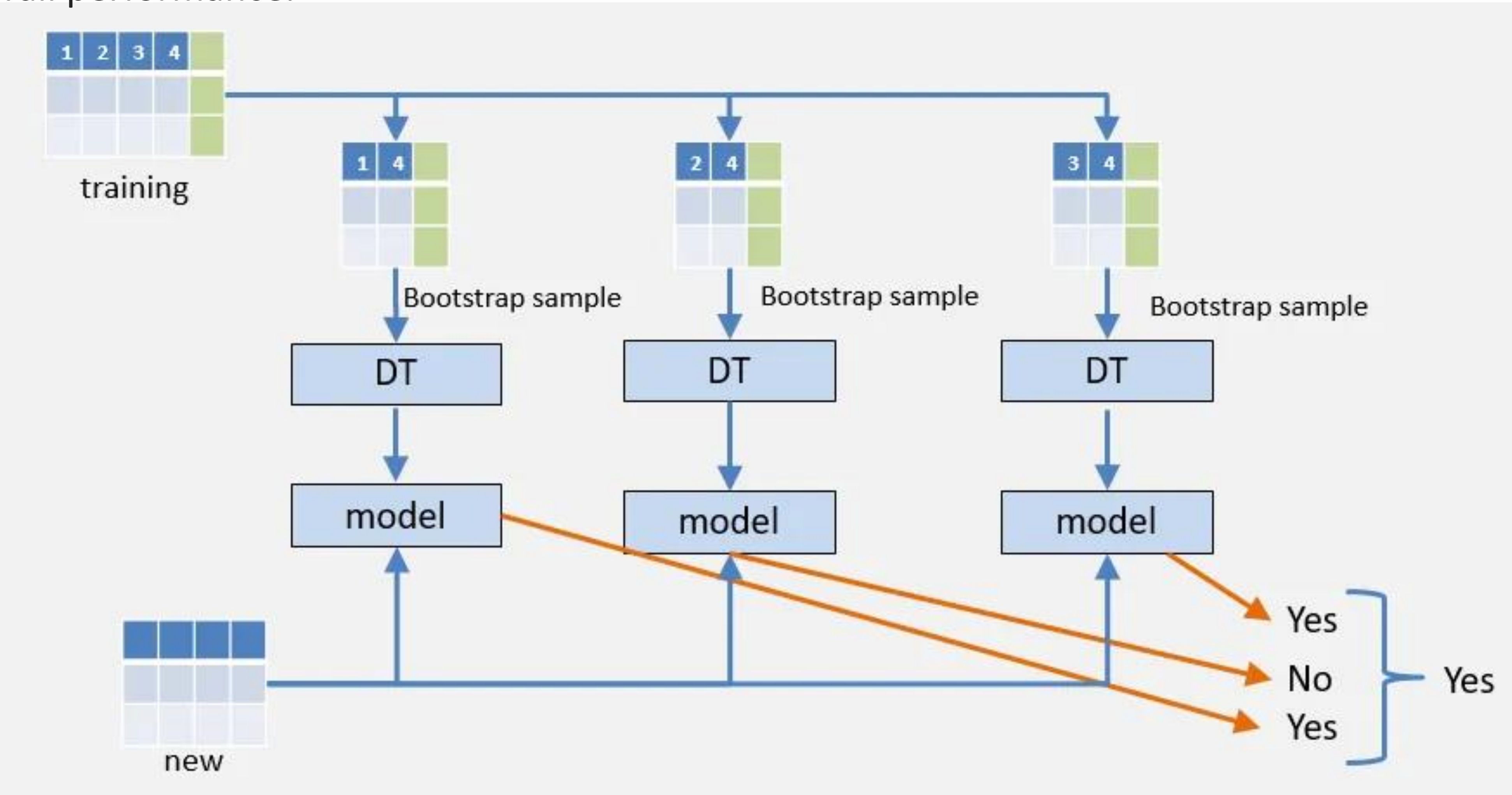
Parallel training and parallel inferencing



From Bagging to Random Forests

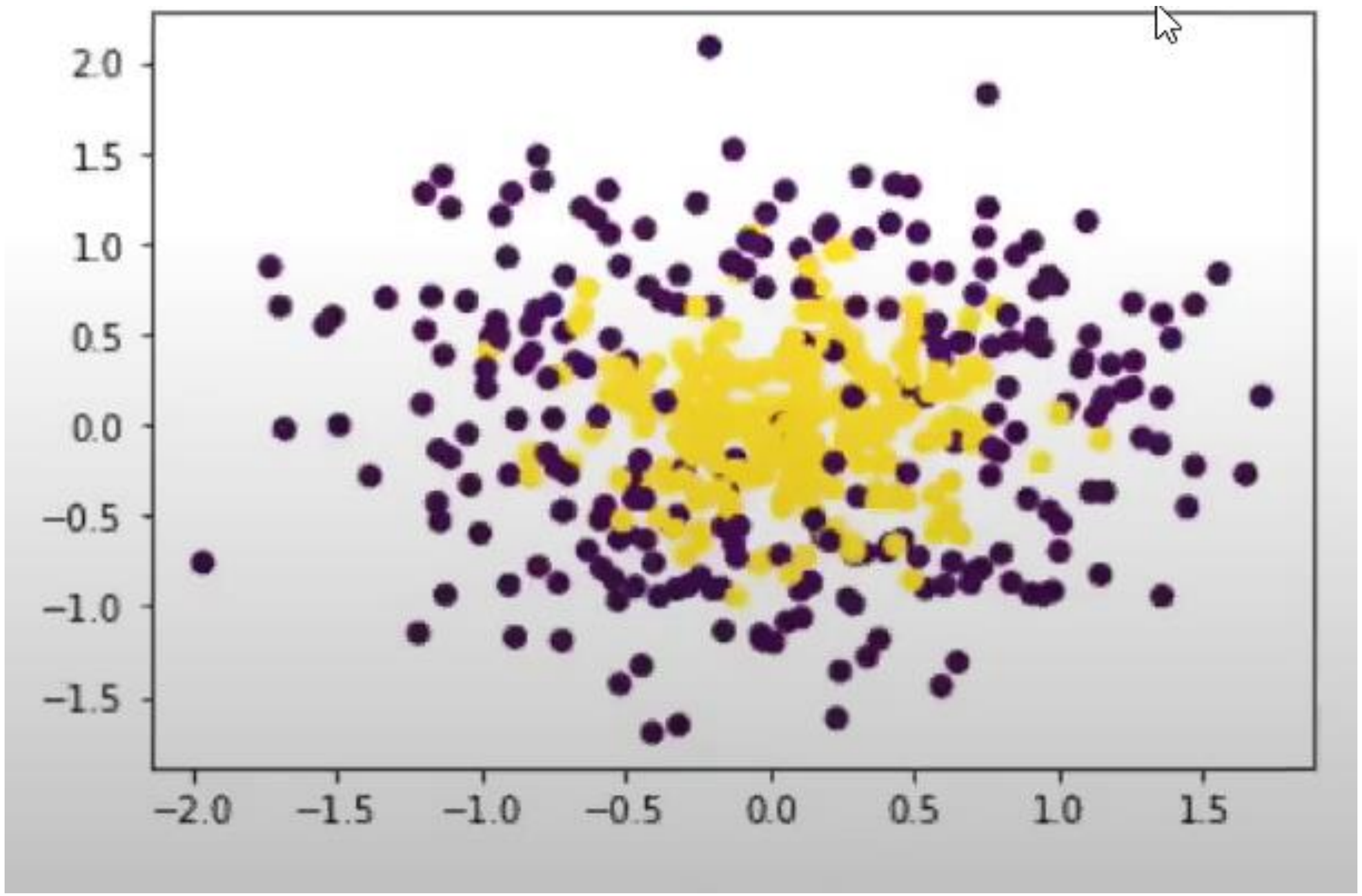
- Random forests include another type of bagging scheme: they consider only a **random subset of the features** when searching for the best split at each decision tree node.

- This process further reduces the correlation between the trees in the forest, decreasing the model's variance and improving its overall performance.



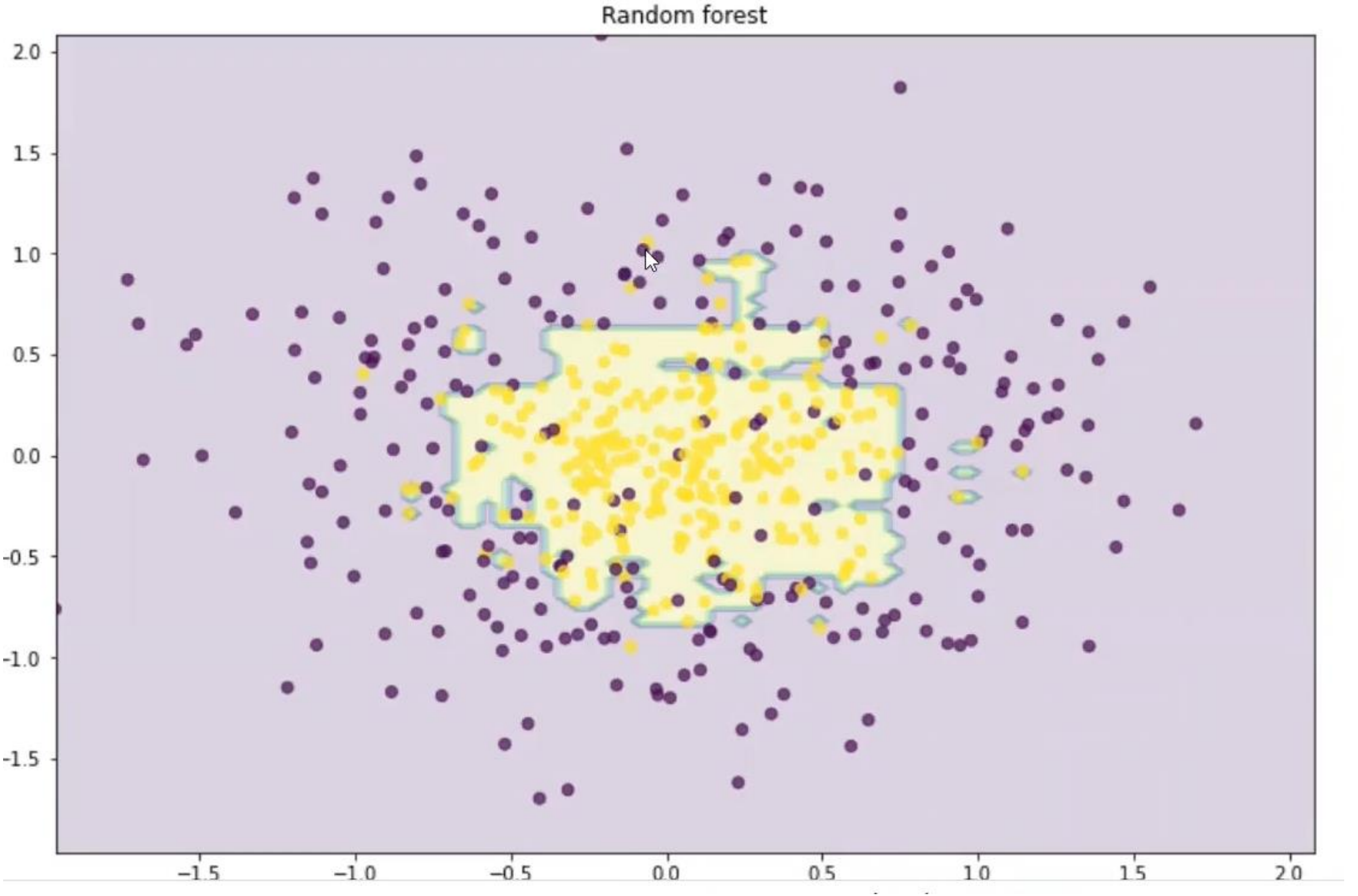
Hyperparameters of Random Forrest

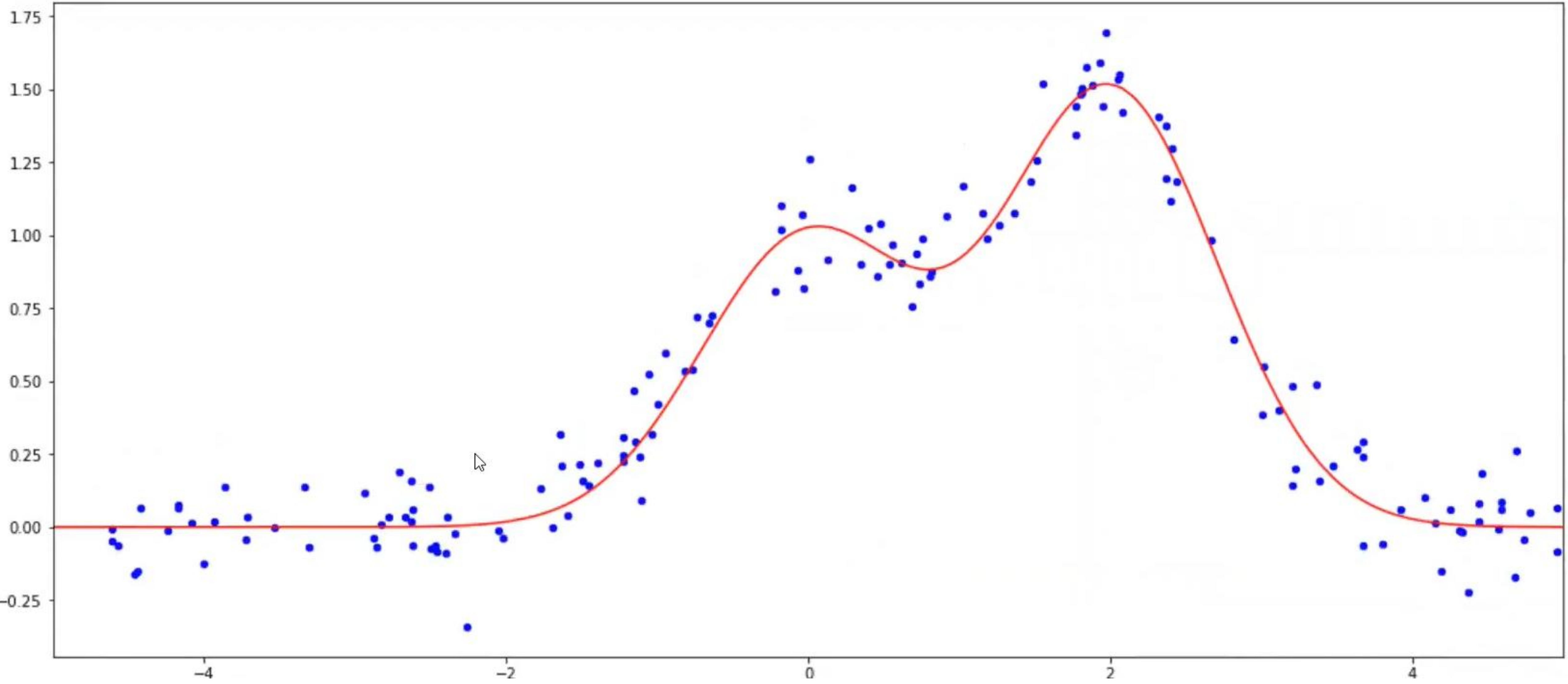
1. *n_estimators* — the number of trees in the forest (defaults to 100).
2. *max_features* — the number of features to consider when searching for the best split in each node. The options are to specify an integer for the number of features, a floating point representing the fraction of features to use, 'sqrt' (the default) for using a square root of the features, 'log2' for using a log2 of the features, and None for using all the features.
3. *max_samples* — the number of samples to draw from the training set for training each tree. The options are to specify an integer for the number of samples, a floating point representing the fraction of the training samples to use, and None (the default) for using all the training samples.

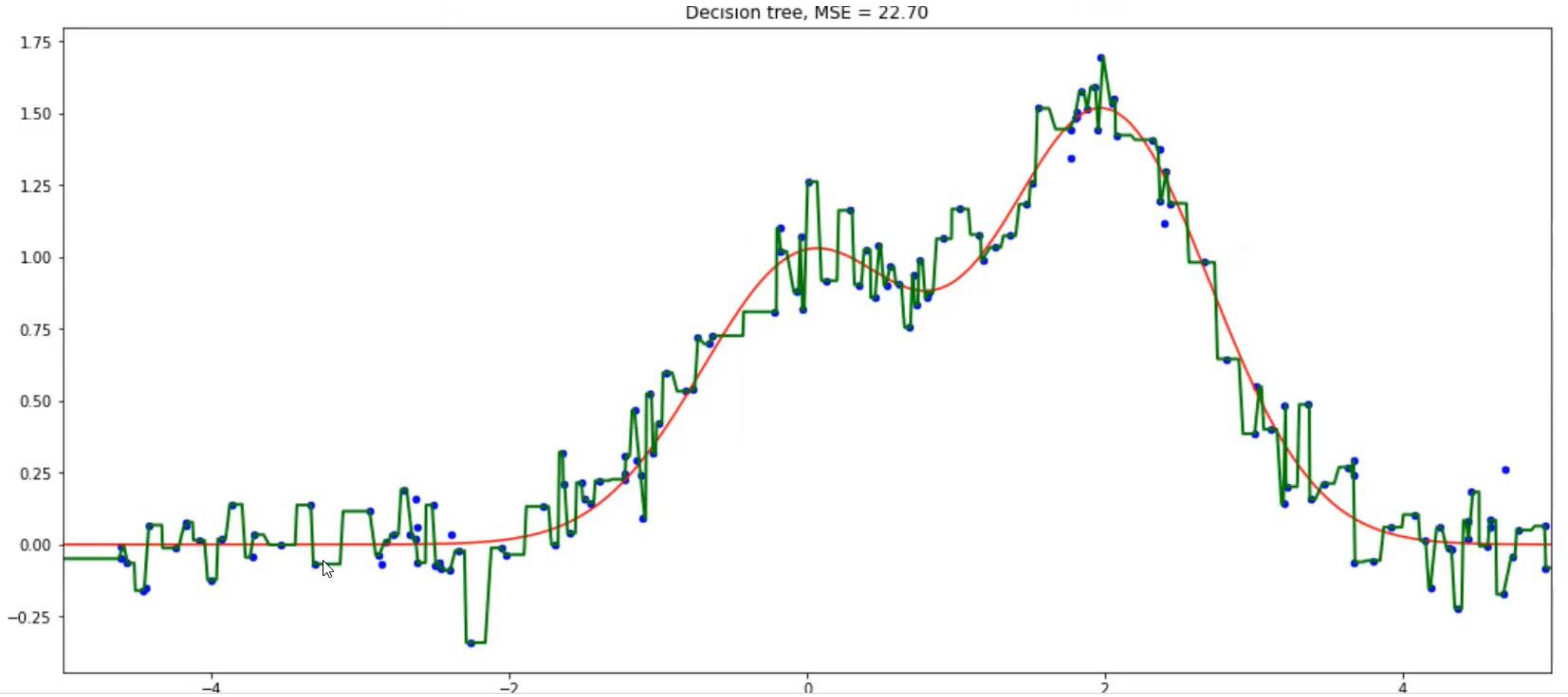

2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5


Decision tree
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0

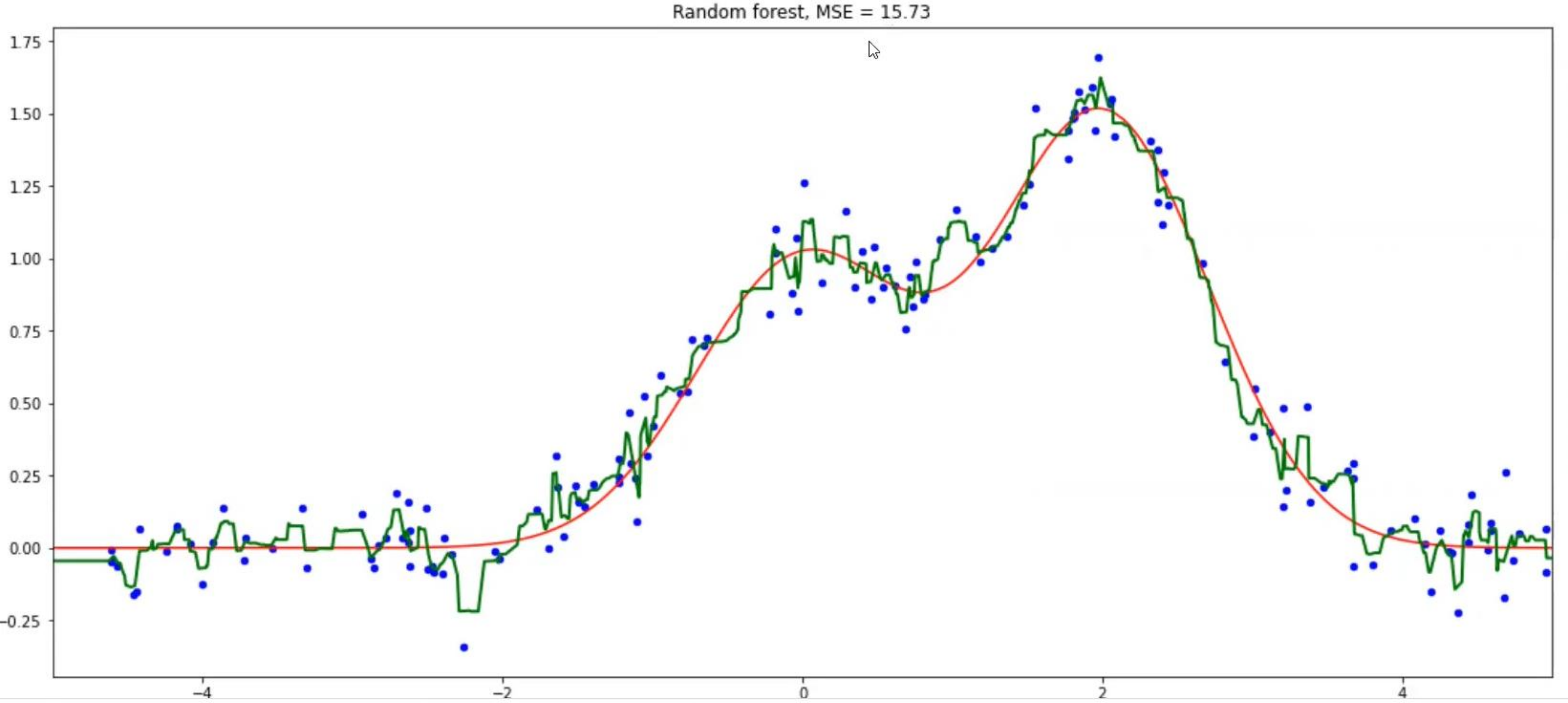
500 Trees




1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
−0.25
−4
−2
0
2
4

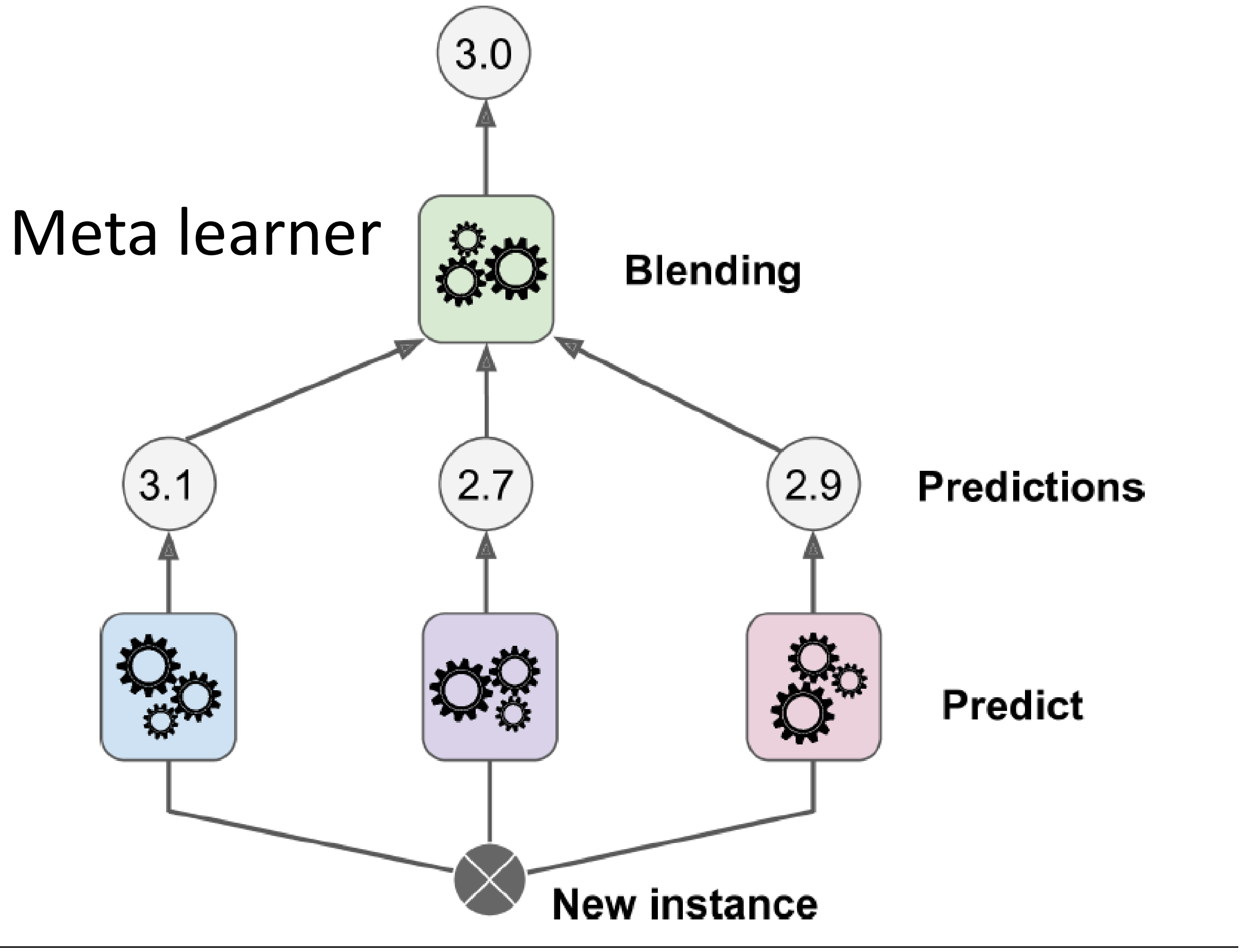


Decision tree, MSE = 22.70
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
−0.25
−4
−2
0
2
4

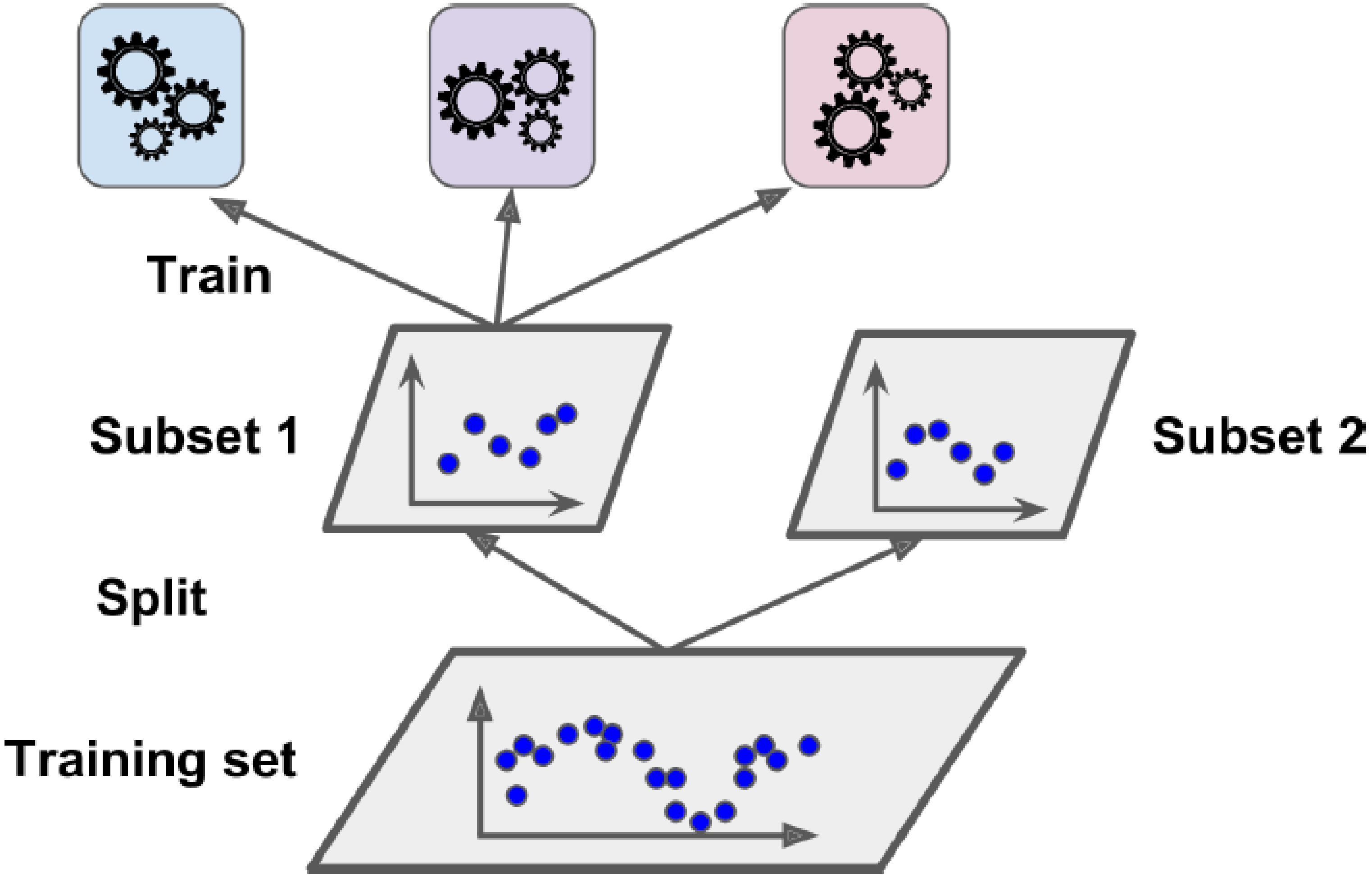

Random forest, MSE = 15.73
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
−0.25
−4
−2
0
2
4

Stacking

- A simple idea: instead of using trivial functions (such as hard voting) to aggregate the predictions of all predictors in an ensemble, why don’t we train a model to perform this aggregation?



Step -1: Training base learner



Step -2: Training meta learner