Lecture 18 pre-video

Ensembles of weak learners

What is a kernel?

Like a metric plus a defined dot product

A **metric** on a set X is a function (called *distance function* or simply **distance**)

$$d : X \times X \to [0, \infty),$$

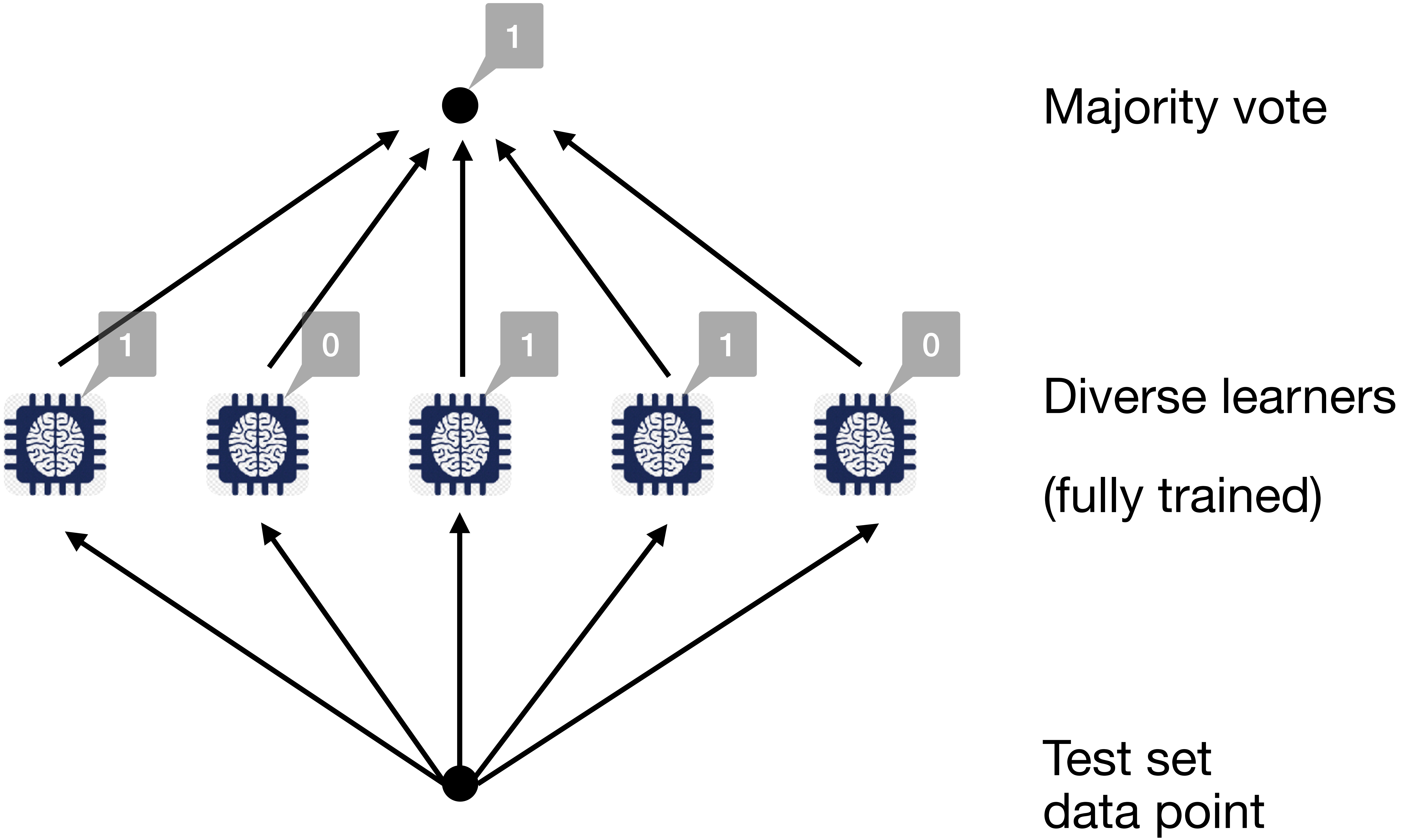
where $[0, \infty)$ is the set of non-negative real numbers and for all $x, y, z \in X$, the following three axioms are satisfied:

1. $d(x, y) = 0 \Leftrightarrow x = y$ identity of indiscernibles
2. $d(x, y) = d(y, x)$ symmetry
3. $d(x, y) \leq d(x, z) + d(z, y)$ subadditivity or triangle inequality

What is a kernel?

Like a metric plus a defined dot product

- For some vector spaces $\mathscr{X} \in \mathbb{R}^n, \mathscr{V} \in \mathbb{R}^m$ a kernel is a function, $k : \mathscr{X} \times \mathscr{X} \mapsto \mathbb{R}$
- The word "kernel" is used in mathematics to denote a weighting function for a weighted sum or integral
- The computation is made much simpler if the kernel can be written in the form of a "feature map" $\phi : \mathscr{X} \mapsto \mathscr{V}$ that moves the problem from its original vector space to one where it is easier to solve the problem
- The feature map must satisfy $k(\mathbf{x}, \mathbf{x}') = \langle \varphi(\mathbf{x}), \varphi(\mathbf{x}') \rangle_{\mathscr{V}}$ where the angle brackets mean this is a proper inner product on the space $\mathscr{V}$


1
Majority vote
1
0
1
1
0
Diverse learners
(fully trained)
Test set
data point

- assume *n* independent classifiers with a base error rate ϵ
- here, independent means that the errors are uncorrelated
- assume a binary classification task
- assume the error rate is better than random guessing (i.e., lower than 0.5 for binary classification)

$$\forall \epsilon_i \in \{\epsilon_1, \epsilon_2, \ldots, \epsilon_n\}, \epsilon_i < 0.5$$

The probability that we make a wrong prediction via the ensemble if *k* classifiers predict the same class label

$$P(k) = \binom{n}{k} \epsilon^k (1 - \epsilon)^{n-k} \qquad k > \lceil n/2 \rceil$$

(Probability mass func. of a binomial distr.)

The probability that we make a wrong prediction via the ensemble if *k* classifiers predict the same class label

$$P(k) = \binom{n}{k} \epsilon^k (1-\epsilon)^{n-k} \qquad k > \lceil n/2 \rceil$$

Ensemble error:

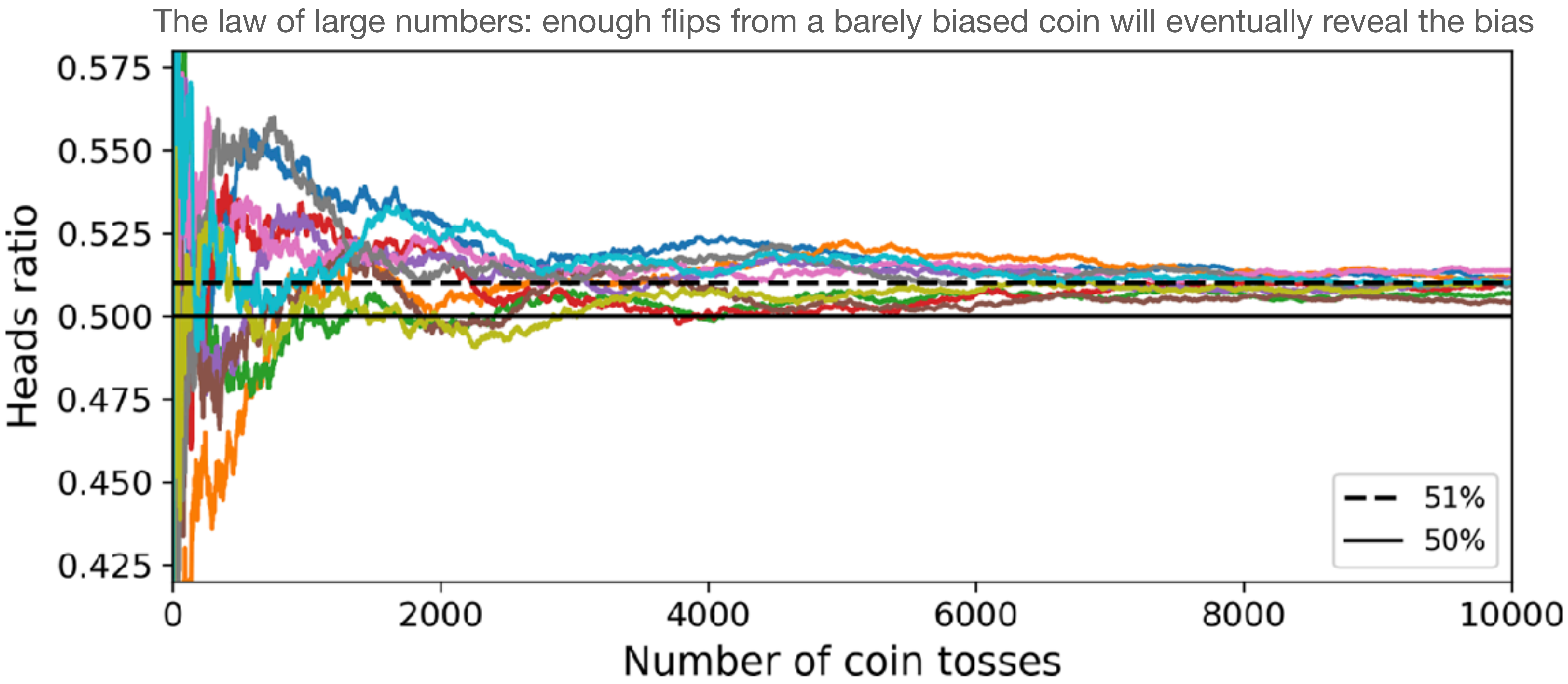
$$\epsilon_{ens} = \sum_{k}^{n} \binom{n}{k} \epsilon^k (1-\epsilon)^{n-k}$$

For instance what if there were 11 classifiers that on average are 75% correct

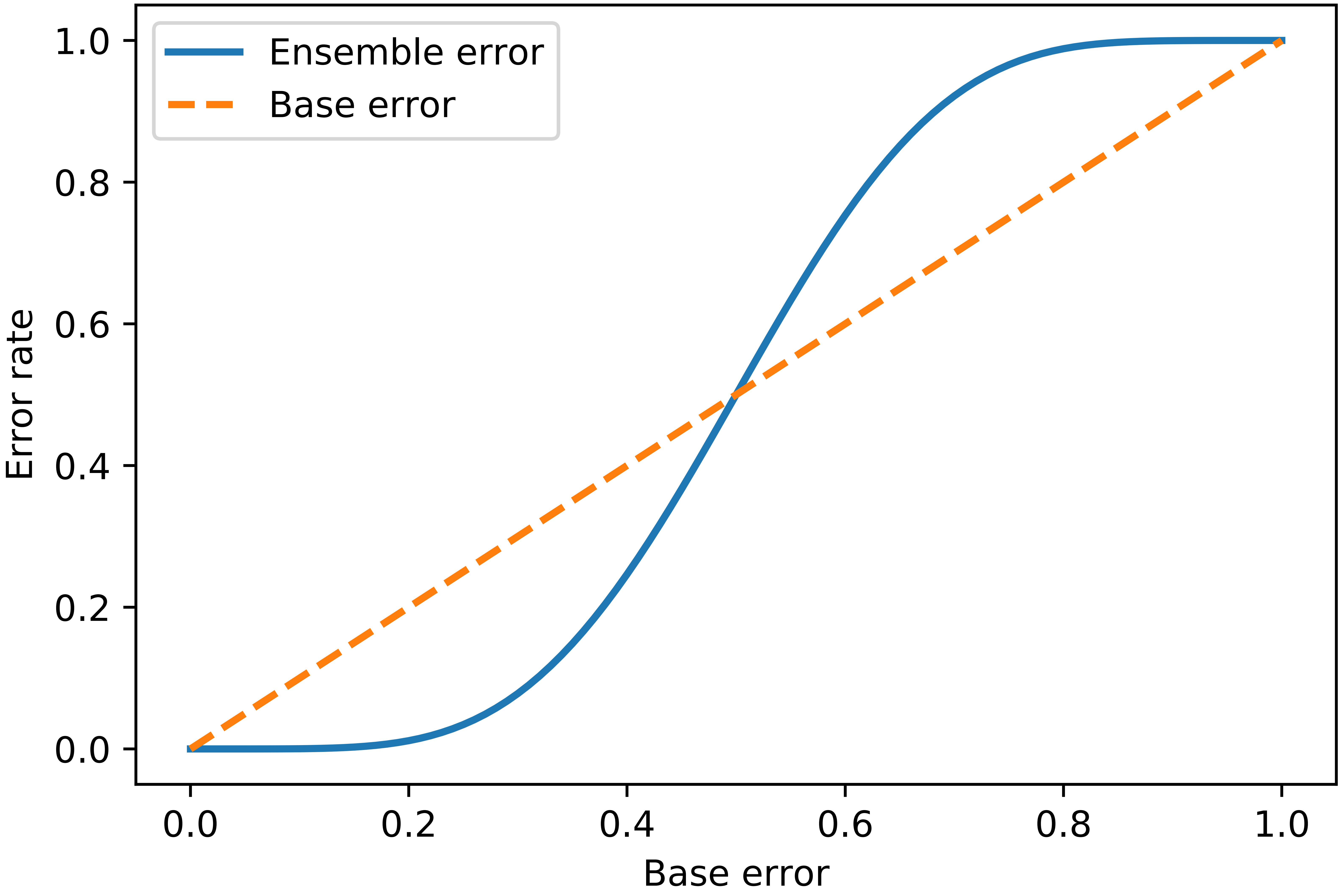
$$\epsilon_{ens} = \sum_{k=6}^{11} \binom{11}{k} 0.25^k (1-0.25)^{11-k} = 0.034$$

Ensemble error:

$$\epsilon_{ens} = \sum_{k}^{n} \binom{n}{k} \epsilon^k (1-\epsilon)^{n-k}$$



$$\epsilon_{ens} = \sum_{k}^{n} \binom{n}{k} \epsilon^k (1 - \epsilon)^{n-k}$$



Bias variance tradeoff in math

Properties of expectation

Linearity

When a is constant and X,Y are random variables:

$$E(aX) = aE(X)$$

$$E(X+Y) = E(X) + E(Y)$$

Constant

When c is constant:

$$E(c) = c$$

Product

When X and Y are independent random variables:

$$E(X \cdot Y) = E(X) \cdot E(Y)$$

$$
\begin{aligned}
\mathbb{E}[(y - \hat{f}(x))^2] &= \mathbb{E}[(f(x) + \epsilon - \hat{f}(x))^2] && (1) \\
&= \mathbb{E}[(f(x) - \hat{f}(x))^2] + \mathbb{E}[\epsilon^2] + 2\mathbb{E}[(f(x) - \hat{f}(x))\epsilon] \\
&= \mathbb{E}[(f(x) - \hat{f}(x))^2] + \underbrace{\mathbb{E}[\epsilon^2]}_{=\sigma_\epsilon^2} + 2\mathbb{E}[(f(x) - \hat{f}(x))] \underbrace{\mathbb{E}[\epsilon]}_{=0} && (2) \\
&= \mathbb{E}[(f(x) - \hat{f}(x))^2] + \sigma_\epsilon^2 && (3)
\end{aligned}
$$

https://towardsdatascience.com/the-bias-variance-tradeoff-8818f41e39e9

$$
\begin{aligned}
\mathbb{E}[(f(x) - \hat{f}(x))^2] &= \mathbb{E}\left[\left((f(x) - \mathbb{E}[\hat{f}(x)]) - (\hat{f}(x) - \mathbb{E}[\hat{f}(x)])\right)^2\right] && (4) \\
&= \mathbb{E}\left[\left(\mathbb{E}[\hat{f}(x)] - f(x)\right)^2\right] + \mathbb{E}\left[\left(\hat{f}(x) - \mathbb{E}[\hat{f}(x)]\right)^2\right] \\
&\quad - 2\mathbb{E}\left[\left(f(x) - \mathbb{E}[\hat{f}(x)]\right)\left(\hat{f}(x) - \mathbb{E}[\hat{f}(x)]\right)\right] && (5) \\
&= \underbrace{(\mathbb{E}[\hat{f}(x)] - f(x))}_{=\text{bias}[\hat{f}(x)]}{}^2 + \underbrace{\mathbb{E}\left[\left(\hat{f}(x) - \mathbb{E}[\hat{f}(x)]\right)^2\right]}_{=\text{var}(\hat{f}(x))} \\
&\quad - 2\left(f(x) - \mathbb{E}[\hat{f}(x)]\right)\mathbb{E}\left[\left(\hat{f}(x) - \mathbb{E}[\hat{f}(x)]\right)\right] && (6) \\
&= \text{bias}[\hat{f}(x)]^2 + \text{var}(\hat{f}(x)) \\
&\quad - 2\left(f(x) - \mathbb{E}[\hat{f}(x)]\right)\left(\mathbb{E}[\hat{f}(x)] - \mathbb{E}[\hat{f}(x)]\right) && (7) \\
&= \text{bias}[\hat{f}(x)]^2 + \text{var}(\hat{f}(x)) && (8)
\end{aligned}
$$

Combine eqns (3) + (8) $\mathbb{E}[(y - \hat{f}(x))^2] = \text{Irreducible Error} + \text{Bias}^2 + \text{Variance}.$

Lecture #13

Ensembles: Random Forest

Jason G. Fleischer, Ph.D.

Asst. Teaching Professor

Department of Cognitive Science, UC San Diego

jfleischer@ucsd.edu

@jasongfleischer

https://jgfleischer.com

Slides in this presentation are from material kindly provided by Zhuowen Tu and others credited at those slides

Decision trees

Advantages

- Interpretable
- Non-parametric method
- Able to fit arbitrary decision boundaries (not just linear!)
- Don't need to scale features to match each other
- Can be combined with techniques to make it better (like bagging and boosting)

Disadvantages

- Easy to overfit
- Needs some kind of pruning and tree-growth limits to avoid overfitting
- For regression trees, the output is bounded by the limits of the training samples

What if we trained a bunch of decision trees?

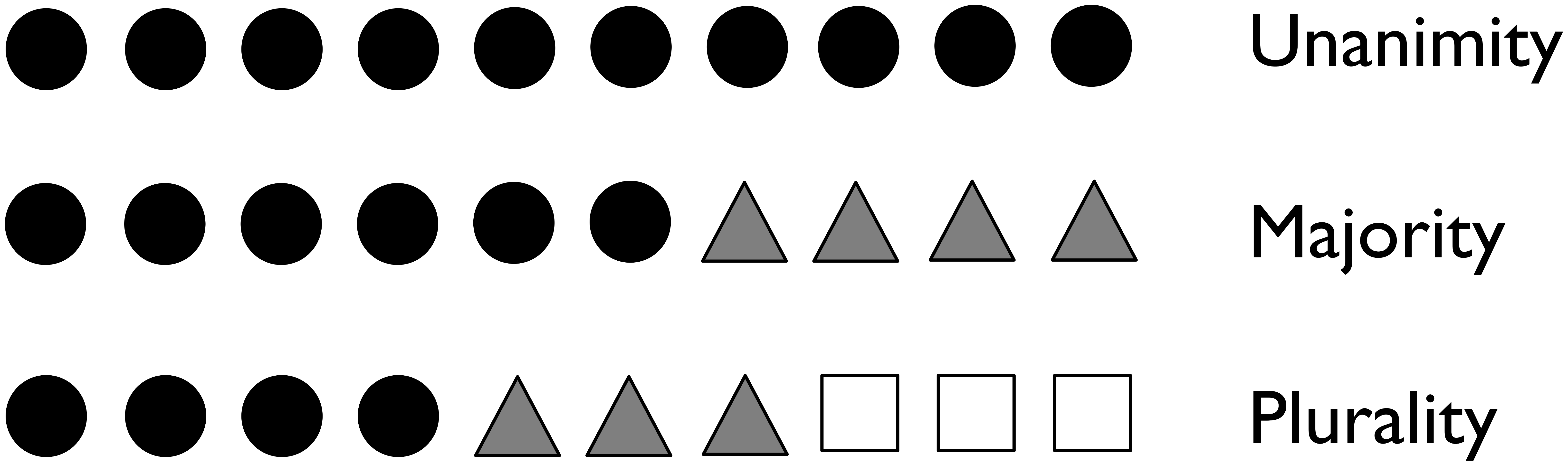
-Bagging trees

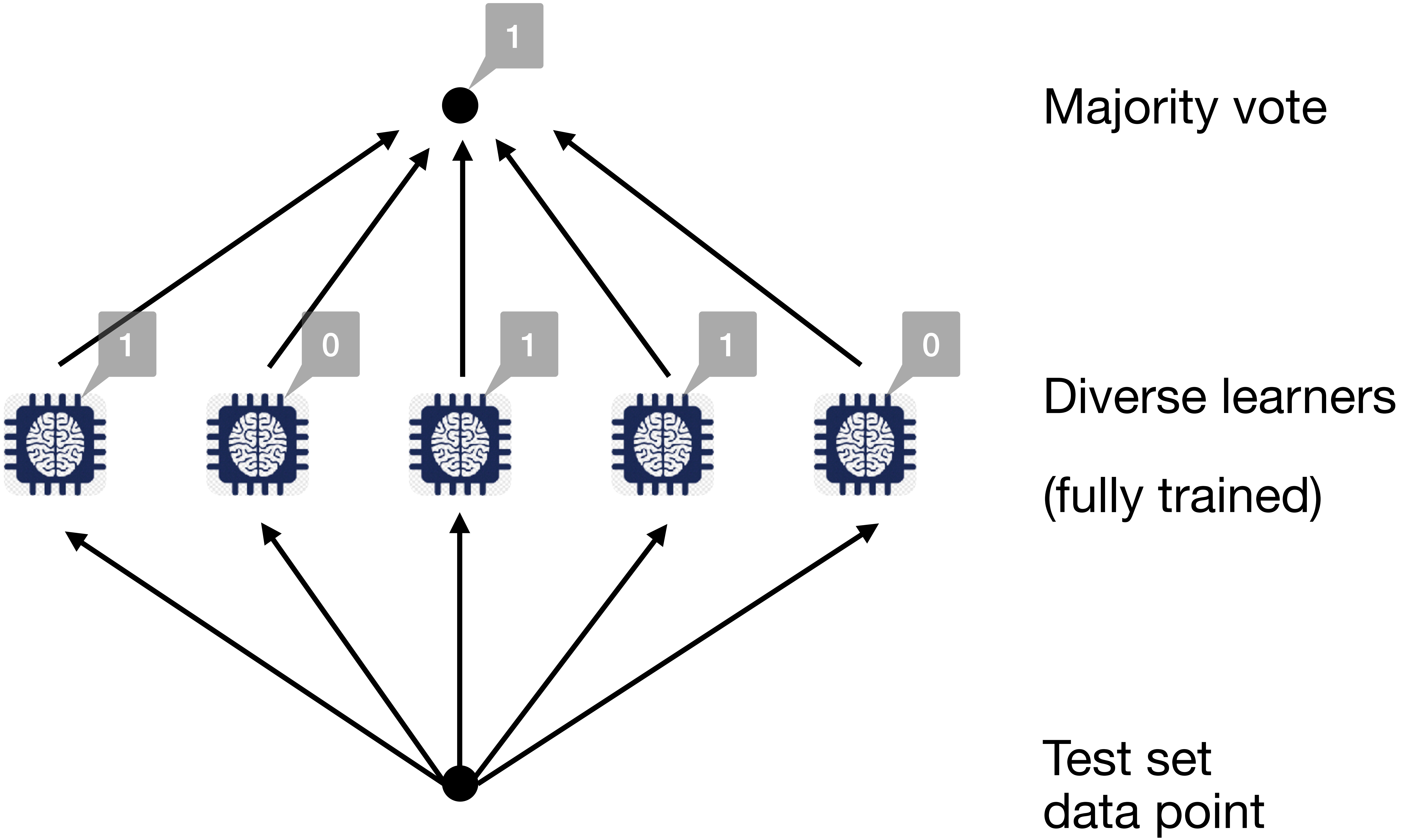
-Random forests

Ensemble methods

Voting

When we say majority we SOMETIMES actually mean plurality




1
Majority vote
1
0
1
1
0
Diverse learners
(fully trained)
Test set
data point

Why Majority Vote?

- assume *n* independent classifiers with a base error rate ϵ
- here, independent means that the errors are uncorrelated
- assume a binary classification task
- assume the error rate is better than random guessing (i.e., lower than 0.5 for binary classification)

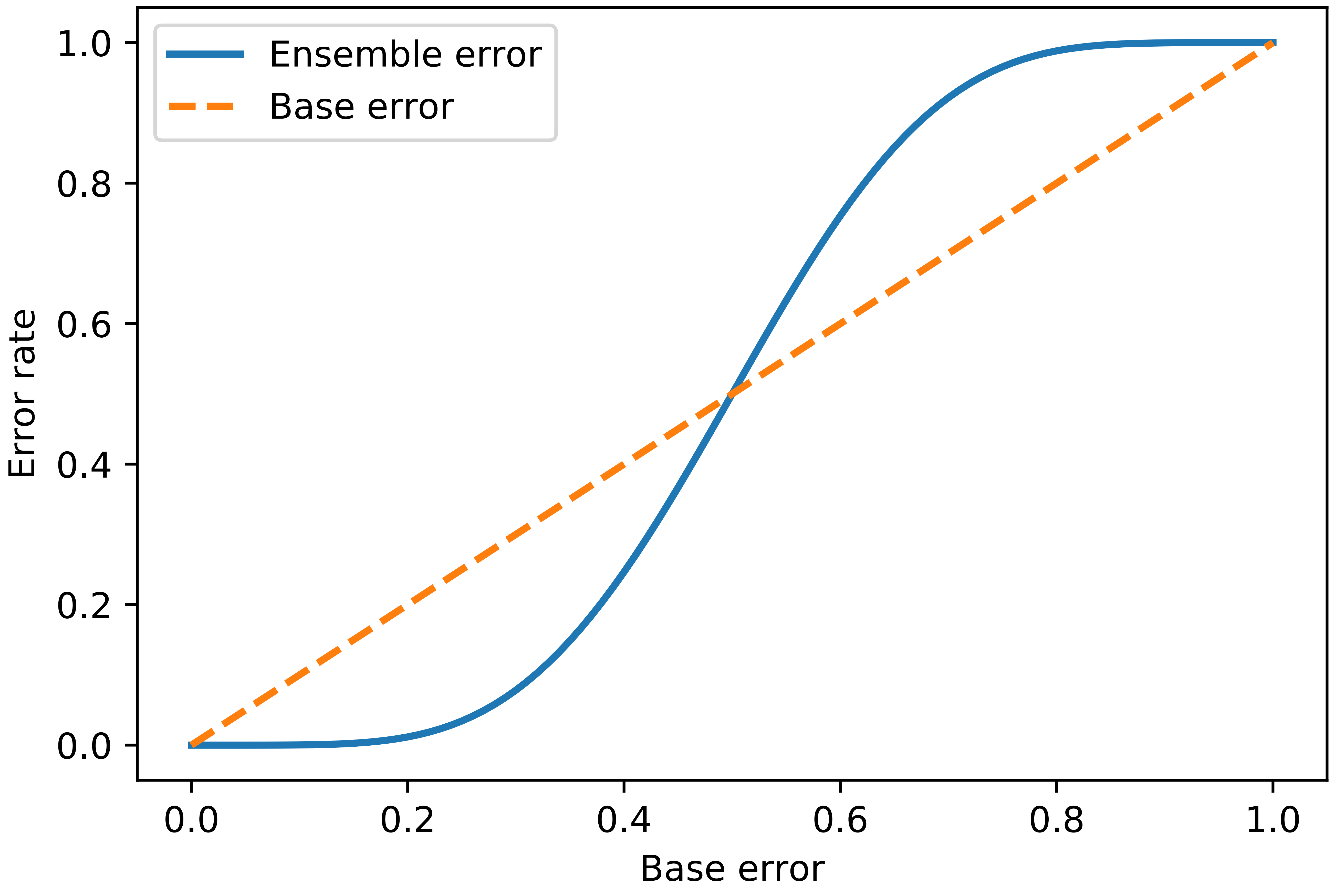
$$\forall \epsilon_i \in \{\epsilon_1, \epsilon_2, \ldots, \epsilon_n\}, \epsilon_i < 0.5$$

The probability that we make a wrong prediction via the ensemble if *k* classifiers predict the same class label

$$P(k) = \binom{n}{k} \epsilon^k (1 - \epsilon)^{n-k} \qquad k > \lceil n/2 \rceil$$

(Probability mass func. of a binomial distr.)

$$\epsilon_{ens} = \sum_{k}^{n} \binom{n}{k} \epsilon^k (1-\epsilon)^{n-k}$$



"Soft" Voting

$$\hat{y} = \arg\max_j \sum_{i=1}^{n} w_i p_{i,j}$$

$p_{i,j}$: predicted class membership probability of the *i*th classifier for class label *j*

w_j : optional weighting parameter, default $w_i = 1/n, \forall w_i \in \{w_1, \ldots, w_n\}$

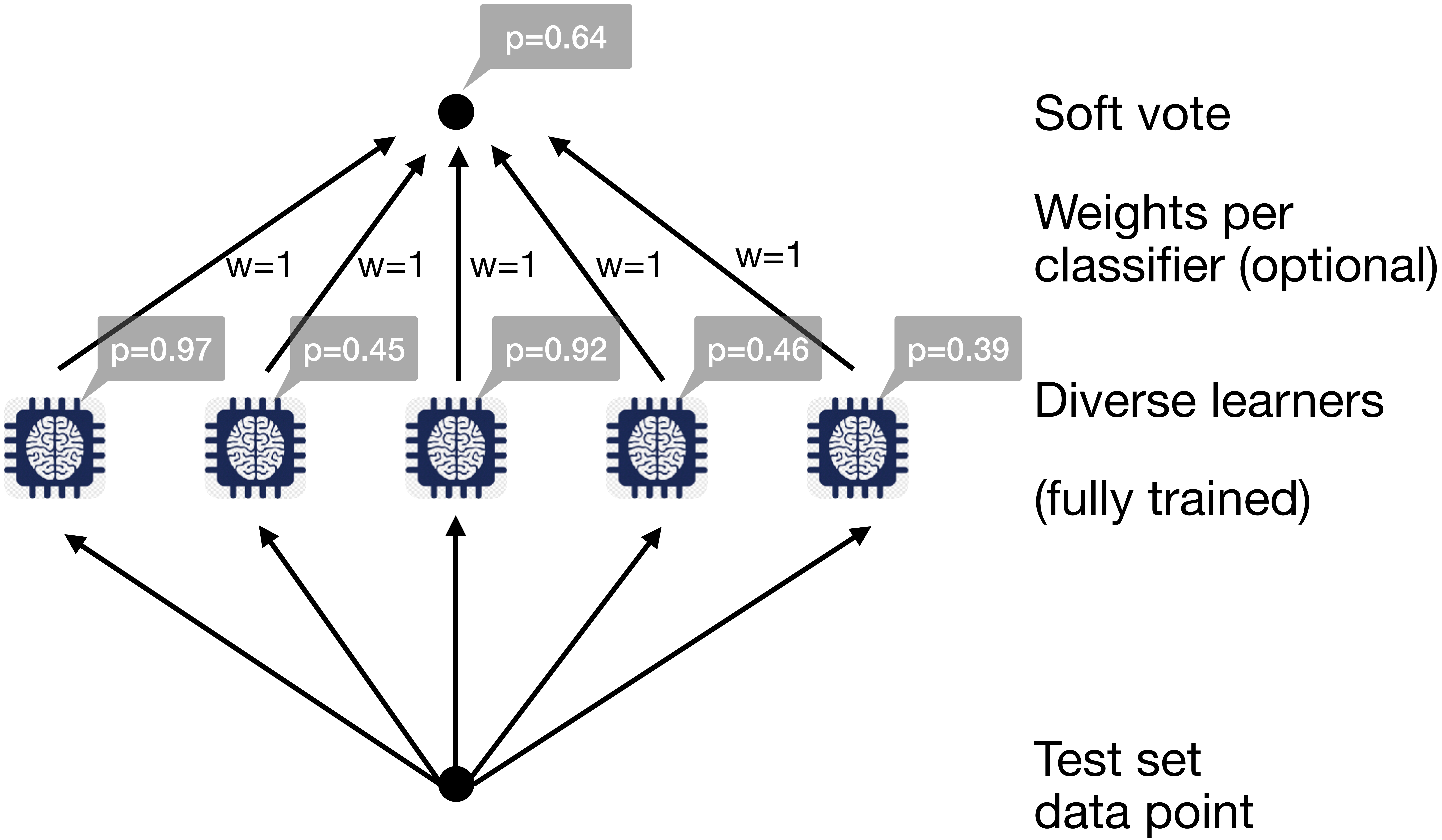
"Soft" Voting

Use only for well-calibrated classifiers!

$$\hat{y} = \arg\max_{j} \sum_{i=1}^{n} w_i p_{i,j}$$

$p_{i,j}$: predicted class membership probability of the *i*th classifier for class label *j*

w_j : optional weighting parameter, default $w_i = 1/n, \forall w_i \in \{w_1, \ldots, w_n\}$



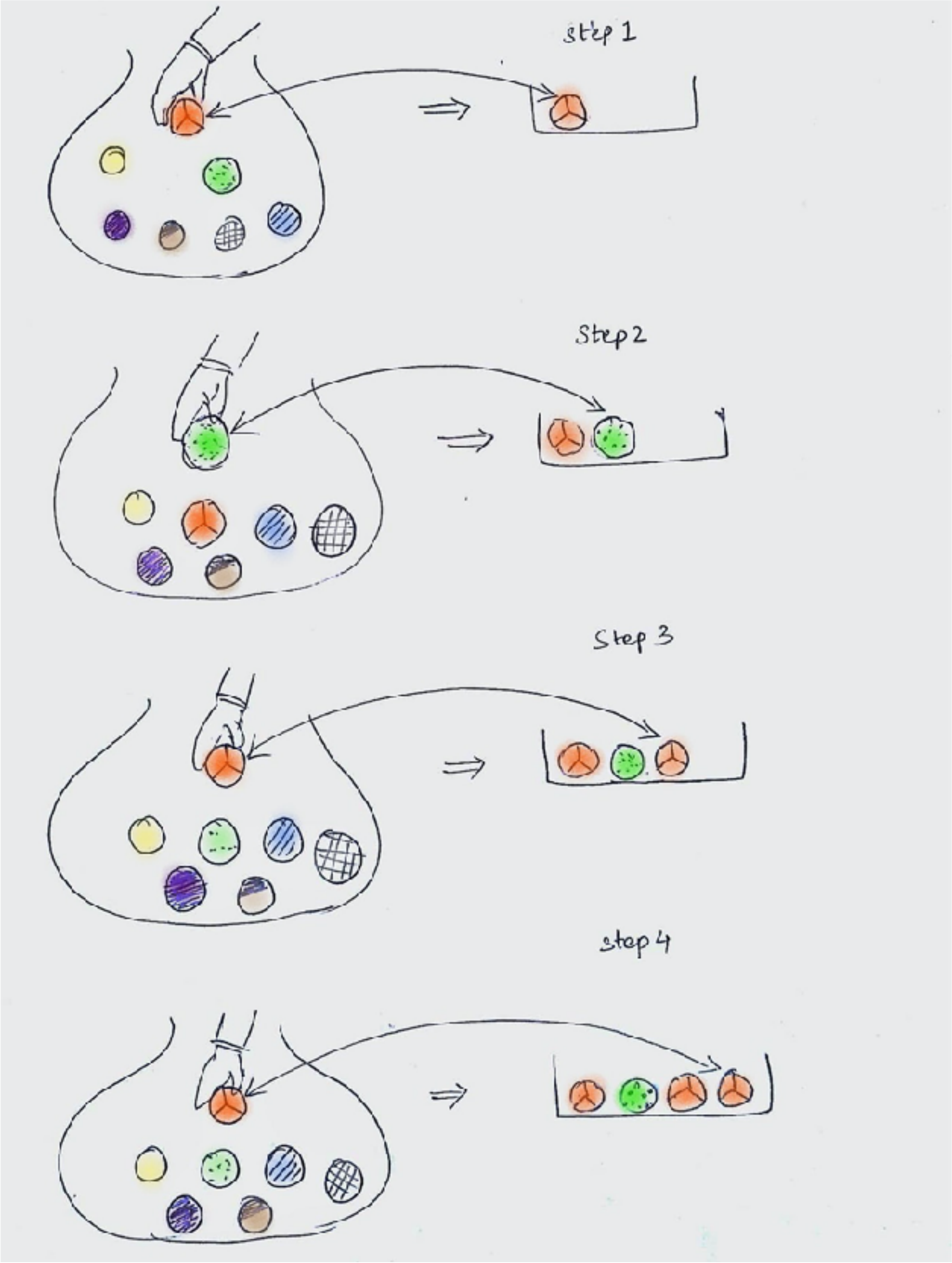
p=0.64
Soft vote
w=1
w=1
w=1
w=1
w=1
Weights per
classifier (optional)
p=0.97
p=0.45
p=0.92
p=0.46
p=0.39
Diverse learners
(fully trained)
Test set
data point

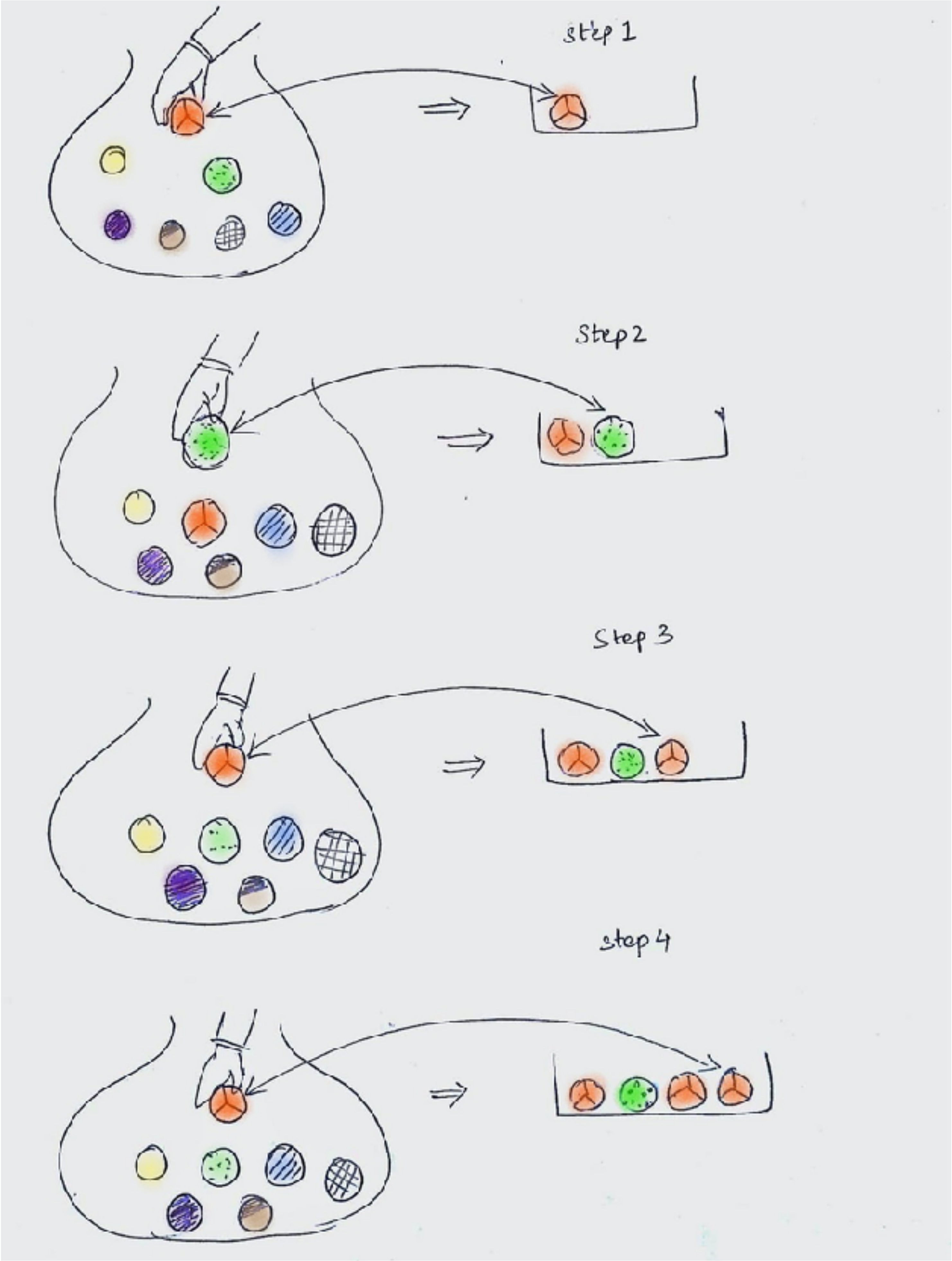
The Bootstrap

Circa 1900, to pull (oneself) up by (one's) bootstraps was used figuratively of an impossible task (Among the "practical questions" at the end of chapter one of Steele's "Popular Physics" schoolbook (1888) is, "30. Why can not a man lift himself by pulling up on his boot-straps?"). By 1916 its meaning expanded to include "better oneself by rigorous, unaided effort." The meaning "fixed sequence of instructions to load the operating system of a computer" (1953) is from the notion of the first-loaded program pulling itself, and the rest, up by the bootstrap.

(Source: Online Etymology Dictionary)

The Bootstrap: sampling with replacement

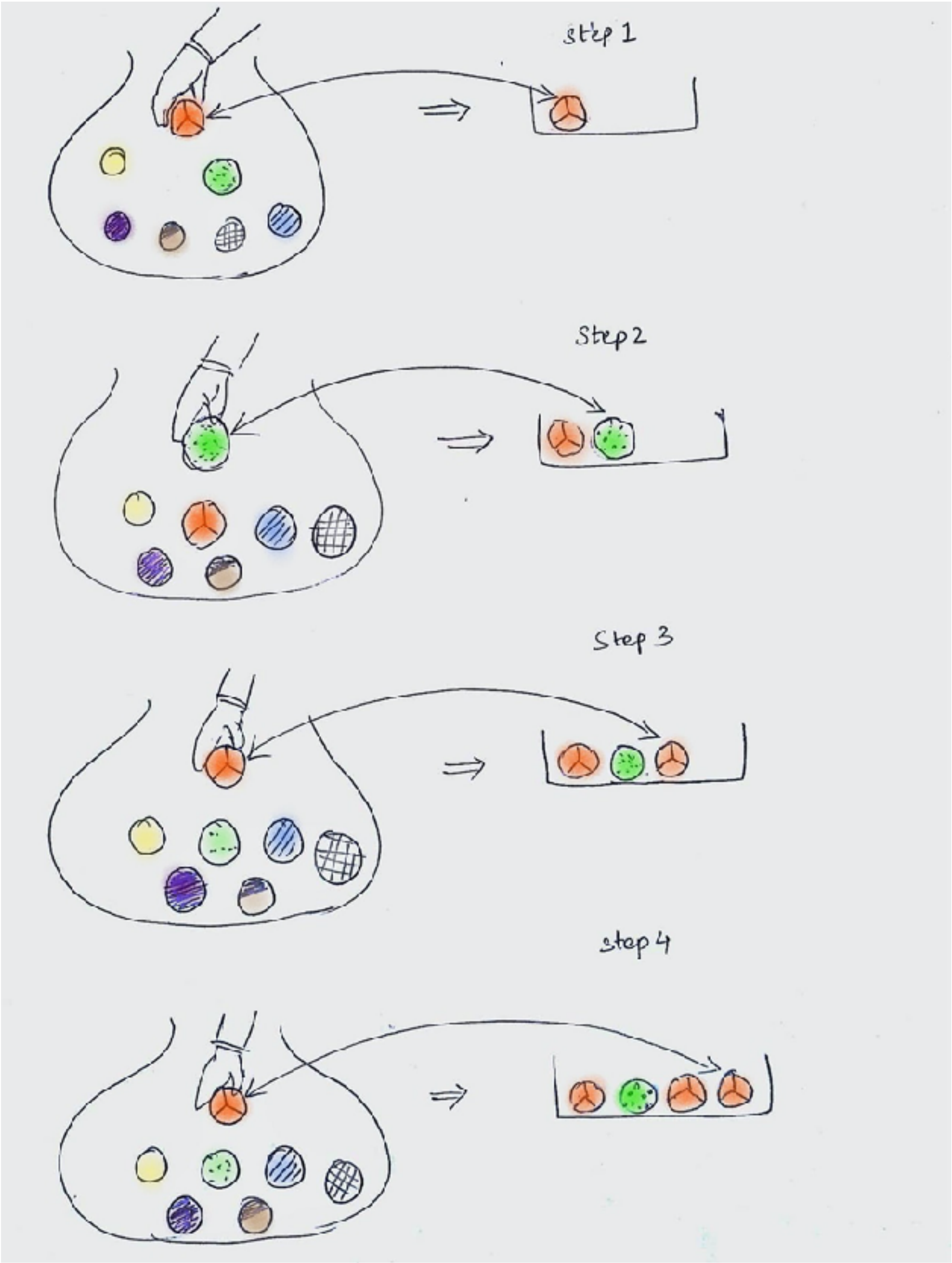




Bootstrap Sampling

$$P(\textbf{not chosen}) = \left(1 - \frac{1}{n}\right)^n,$$

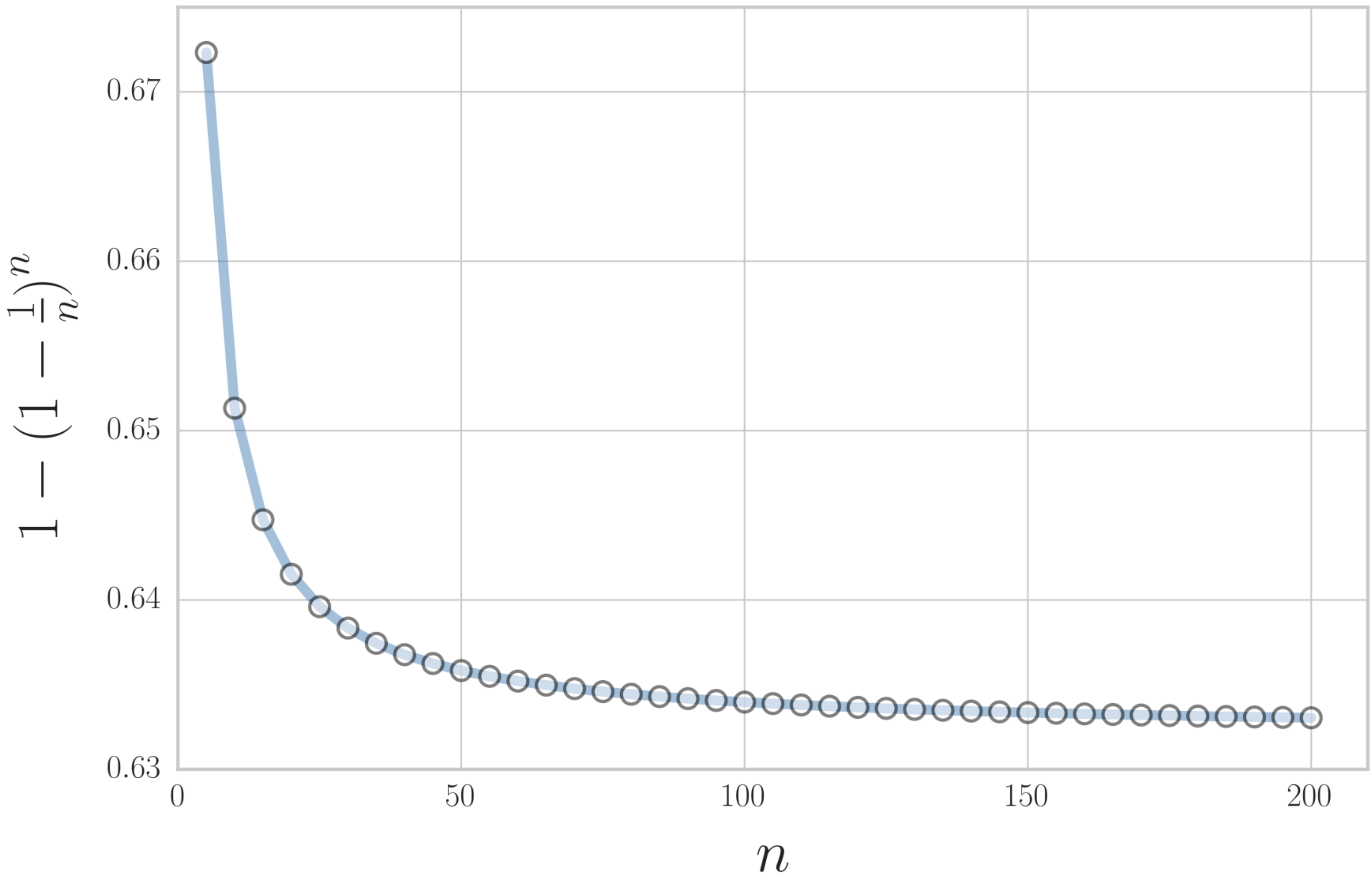
$$\frac{1}{e} \approx 0.368, \quad n \to \infty\,.$$

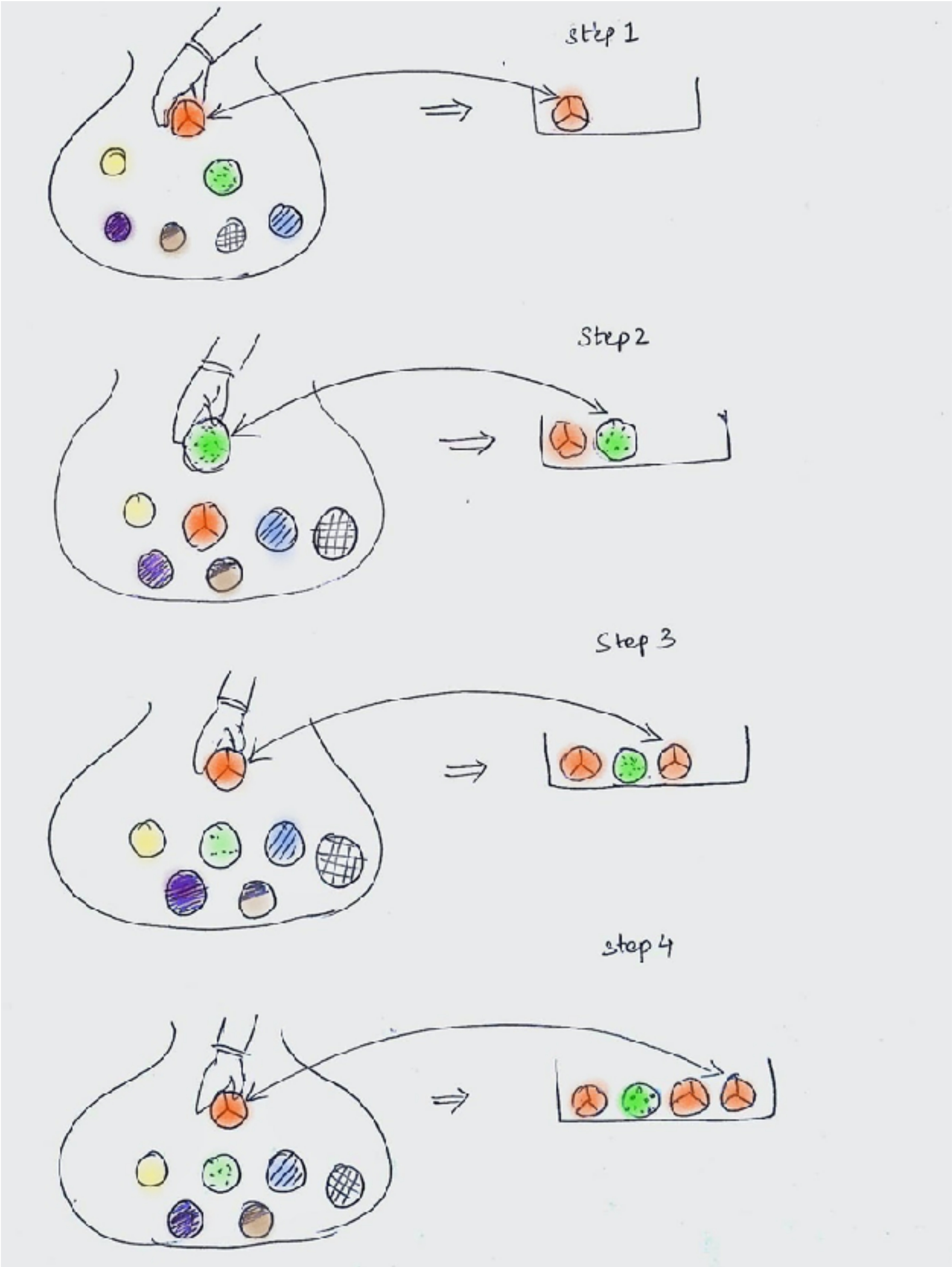


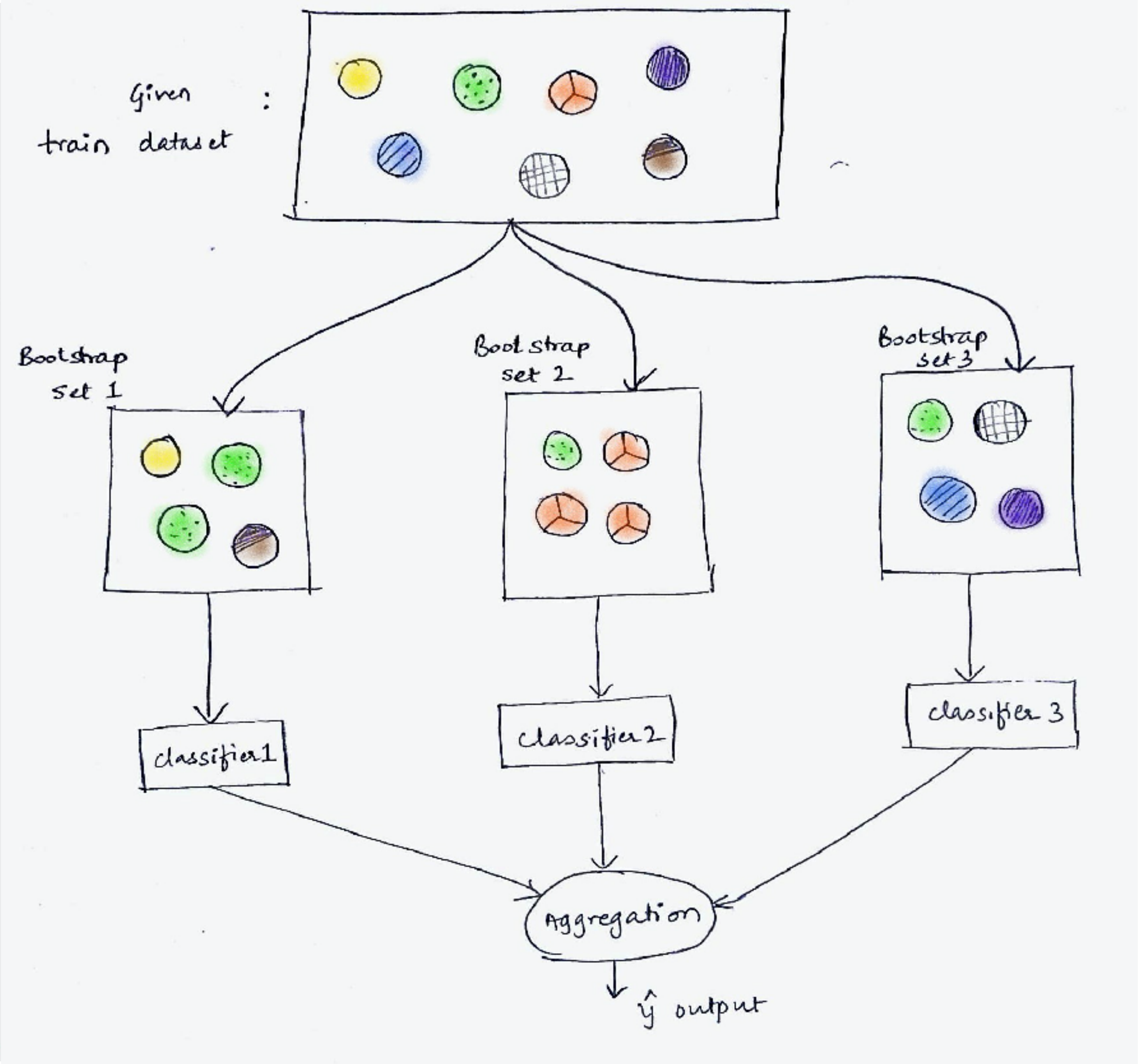
$$P(\textbf{not chosen}) = \left(1 - \frac{1}{n}\right)^n,$$

$$\frac{1}{e} \approx 0.368, \quad n \to \infty\,.$$

$$P(\textbf{chosen}) = 1 - \left(1 - \frac{1}{n}\right)^n \approx 0.632$$







Breiman, L. (1996). Bagging predictors. *Machine learning*, *24*(2), 123-140.

Drawings by Amey Naik

https://medium.com/machine-learning-through-visuals/machine-learning-through-visuals-part-1-what-is-bagging-ensemble-learning-432059568cc8

Bagging

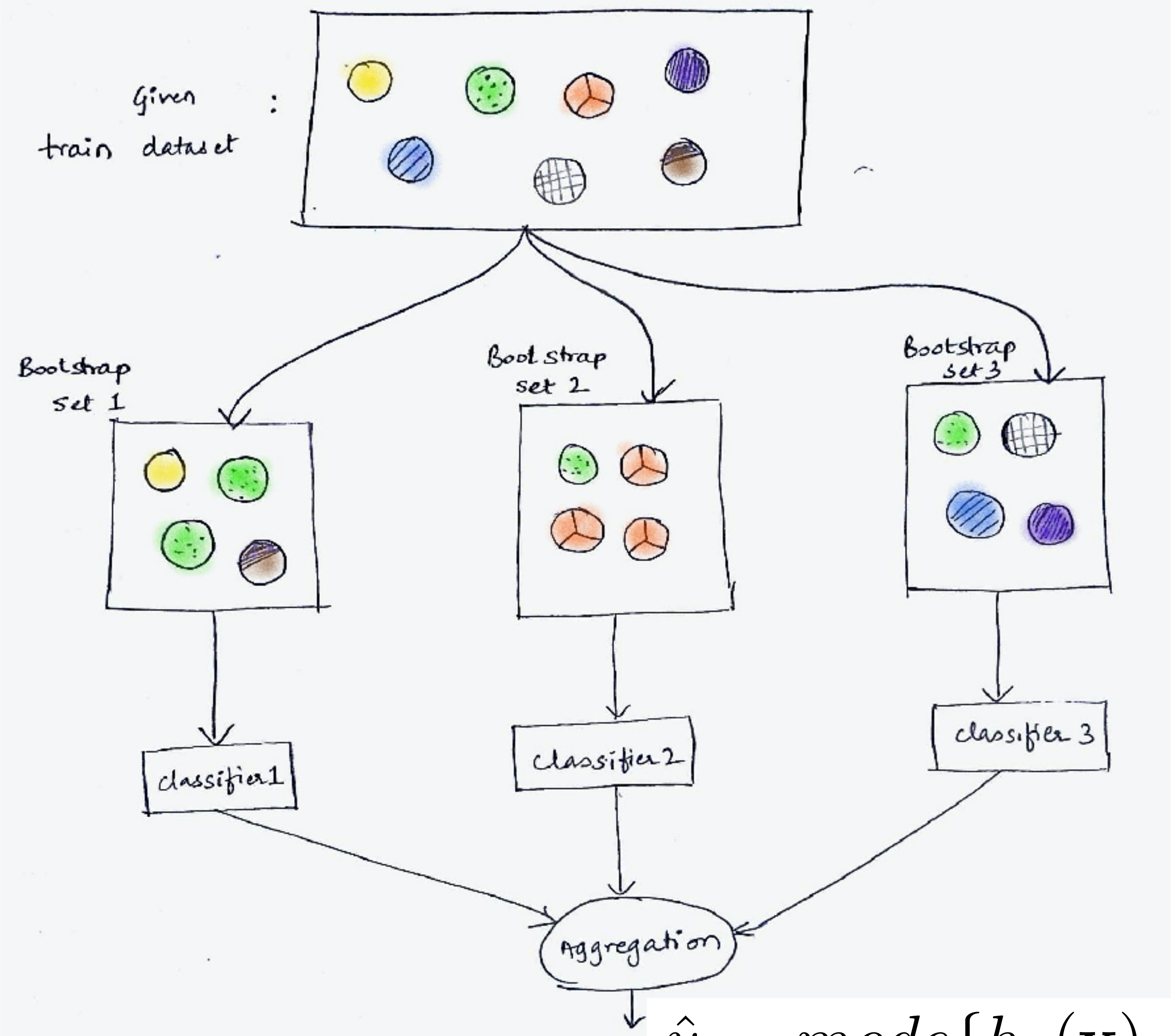
(Bootstrap Aggregating)

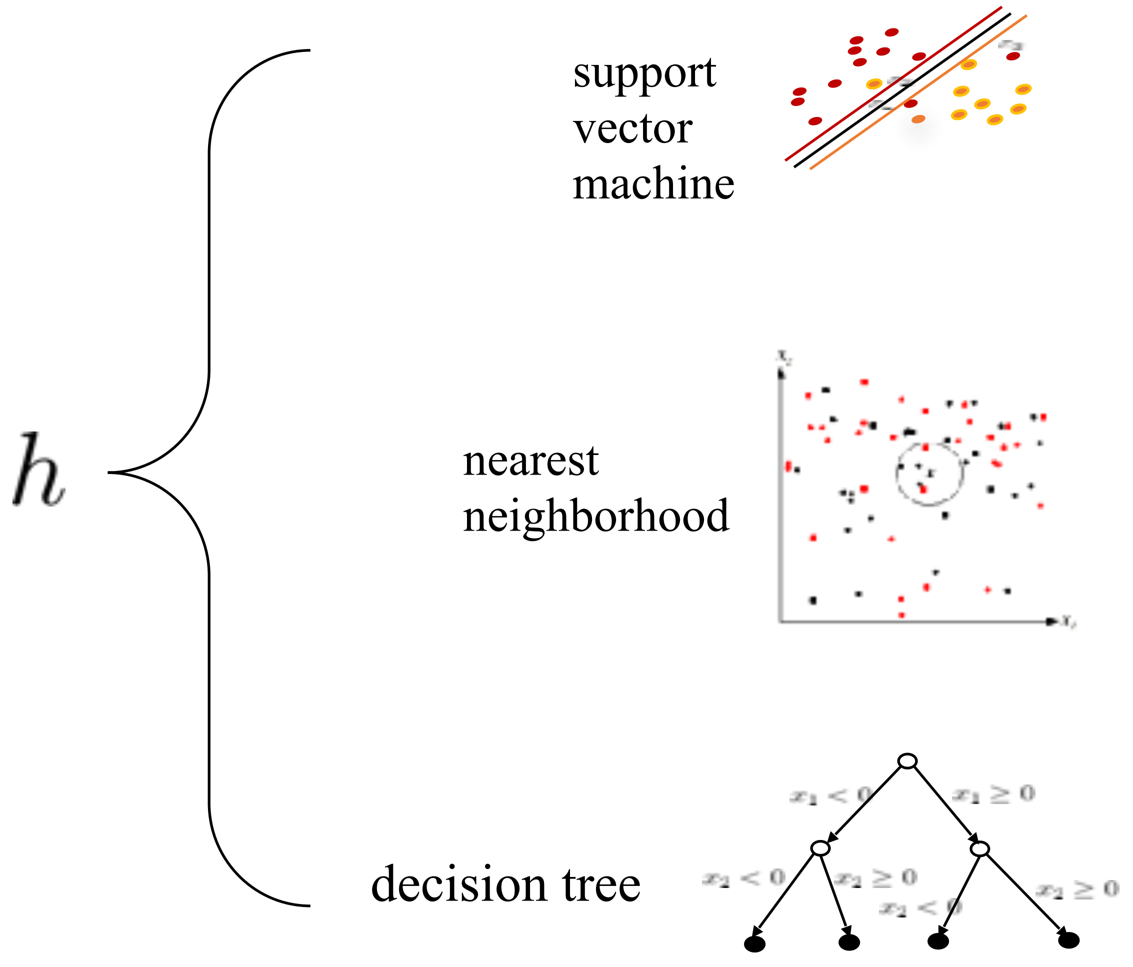
Algorithm 1 Bagging

```
Let n be the number of bootstrap samples

for i=1 to n do
    Draw bootstrap sample of size m, D_i
    Train base classifier h_i on D_i
ŷ = mode{h_1(x), ..., h_n(x)}
```

Bagging classifiers - our first ensemble!





$$\hat{y} = mode\{h_1(\mathbf{x}), ..., h_n(\mathbf{x})\}$$

Out Of Bag error - a built-in generalization estimate

Note the upward bias!

For each observation $(x_i, y_i) \in S$, construct a predictor by averaging only those $h(x_n)$ created from bootstrap samples missing (x_i, y_i)

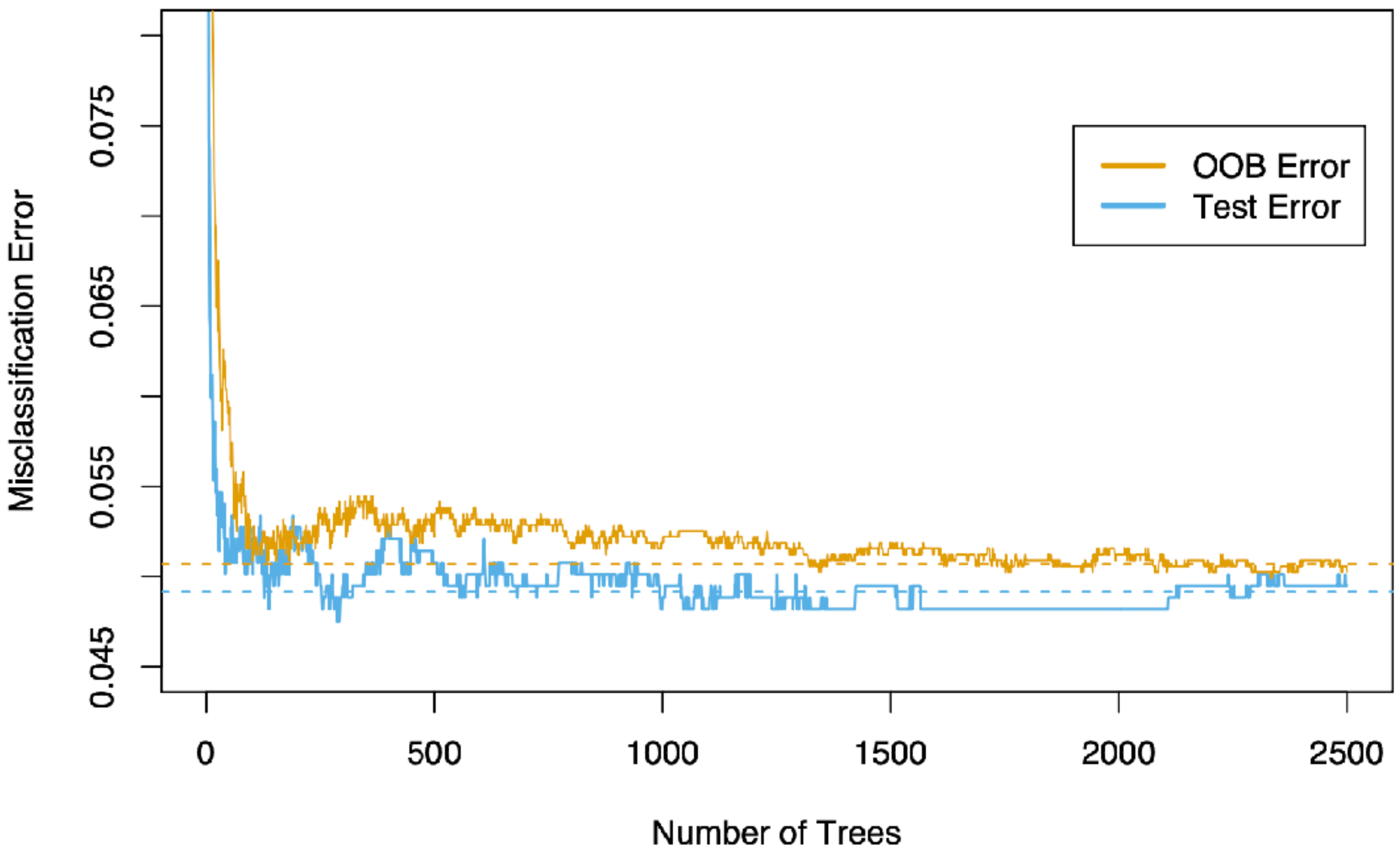


FIGURE 15.4. OOB *error computed on the* `spam` *training data, compared to the test error computed on the test set.*

Hastie et al, Elements of Statistical Learning

OOB is pessimistic

- 0.632 estimate: because on average only 63.2% of the unique samples get into the bag

$$ACC_{.632} = \frac{1}{b} \sum_{i=1}^{b} 0.632\, ACC_{OOB_i} + 0.368\, ACC_{train_i}$$

- .632+ estimate: because the .632 estimate can be optimistic if the model tends to overfit

$$ACC_{.632+} = \frac{1}{b} \sum_{i=1}^{b} \left(\omega * ACC_{OOB_i} + (1 - \omega) * ACC_{\text{train}_i} \right)$$

$\omega = \frac{.632}{(1 - .368)R}$, $R = -\frac{ACC_{OOB_i} - ACC_{train_i}}{\gamma - (1 - ACC_{OOB_i})}$, where γ is a constant calculated empirically on the dataset (the no information rate, related to class priors)

Evaluating generalization via bootstrap sampling

Limited data technique

All the data

Build k bootstrap samples

1. Training set | Out of bag set
2. Training set | Out of bag set

⋮

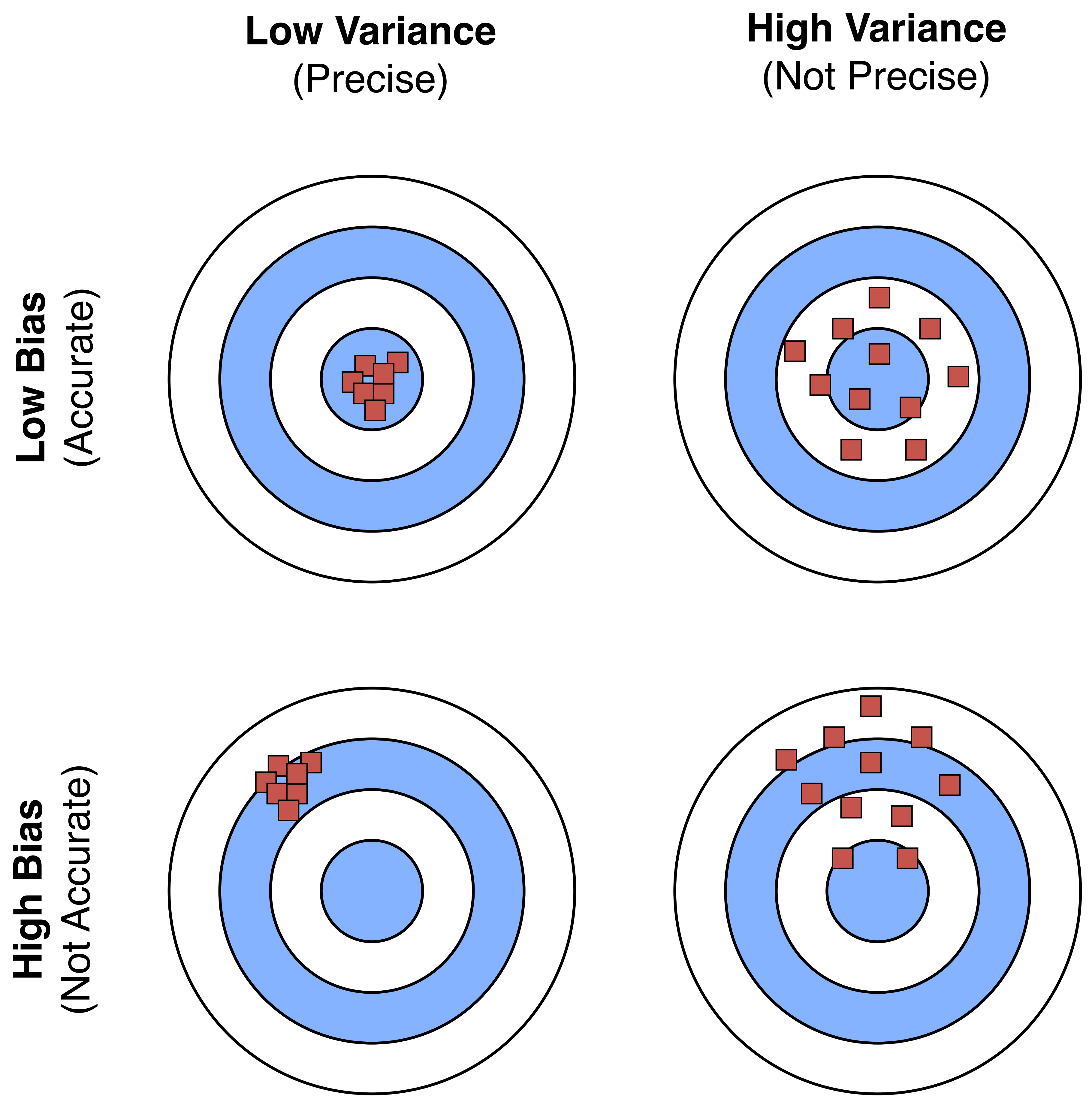
k. Training set | Out of bag set

Use the a corrected mean of the n performances, combining a known overestimate with a known underestimate

$$\bar{\epsilon}_{.632+} = 1/k \sum_{i\in[0,k]} \left(\omega * \epsilon_{\text{OOB}_i} + (1-\omega) * \epsilon_{\text{training}_i}\right)$$

$\omega = \frac{.632}{(1-.368)R}, R = -\frac{\epsilon_{\text{OOB}_i} - \epsilon_{\text{training}_i}}{\gamma - (1-\epsilon_{\text{OOB}_i})}$, where γ is the no information rate

Bias-Variance Intuition



a bagging model has a lower variance than the individual trees and is less prone to overfitting

7.3 The Bias–Variance Decomposition

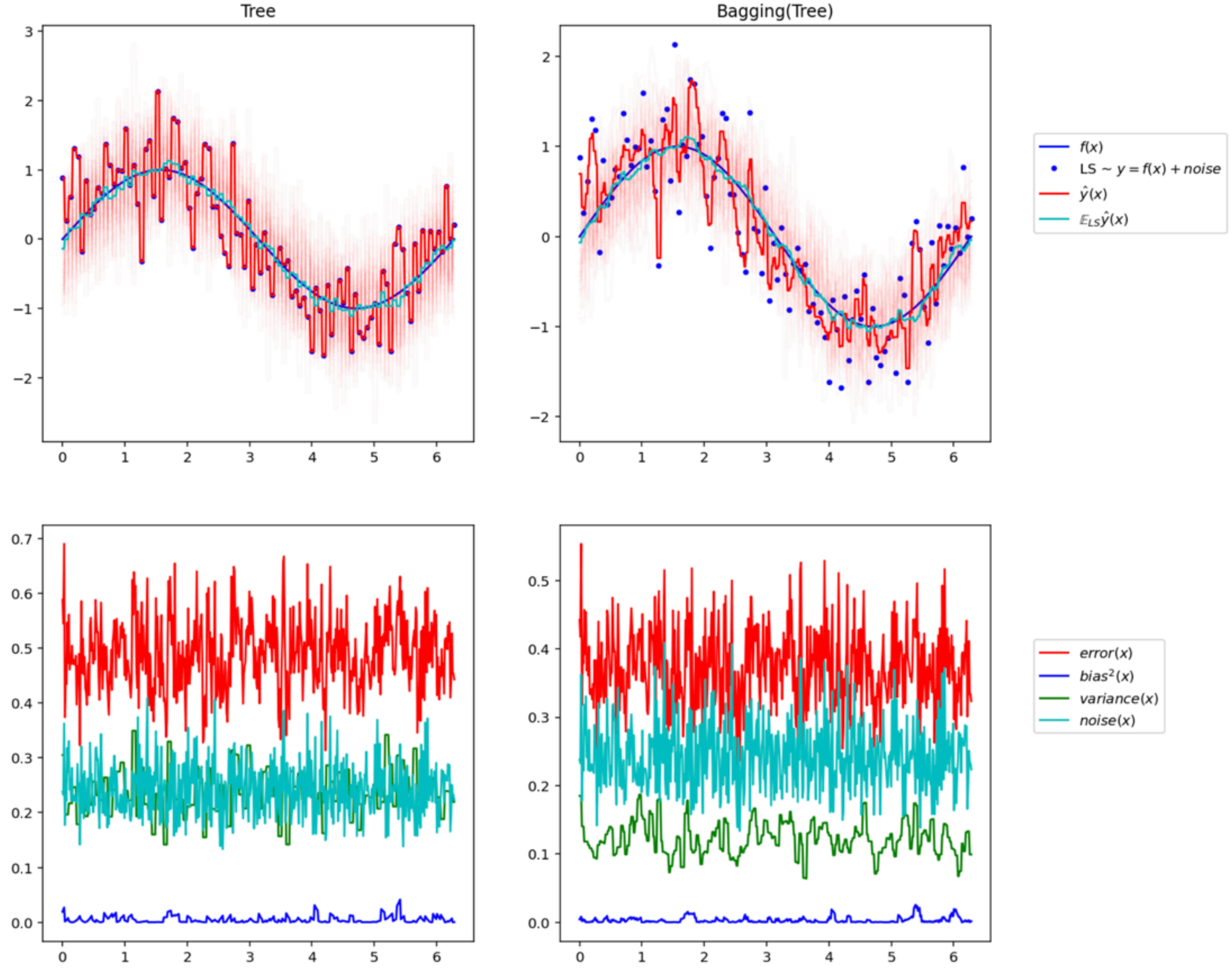
As in Chapter 2, if we assume that $Y = f(X) + \varepsilon$ where $\mathrm{E}(\varepsilon) = 0$ and $\mathrm{Var}(\varepsilon) = \sigma_\varepsilon^2$, we can derive an expression for the expected prediction error of a regression fit $\hat{f}(X)$ at an input point $X = x_0$, using squared-error loss:

$$\begin{aligned} \mathrm{Err}(x_0) &= E[(Y - \hat{f}(x_0))^2 | X = x_0] \\ &= \sigma_\varepsilon^2 + [\mathrm{E}\hat{f}(x_0) - f(x_0)]^2 + E[\hat{f}(x_0) - \mathrm{E}\hat{f}(x_0)]^2 \\ &= \sigma_\varepsilon^2 + \mathrm{Bias}^2(\hat{f}(x_0)) + \mathrm{Var}(\hat{f}(x_0)) \\ &= \text{Irreducible Error} + \mathrm{Bias}^2 + \text{Variance}. \end{aligned} \tag{7.9}$$

The first term is the variance of the target around its true mean $f(x_0)$, and cannot be avoided no matter how well we estimate $f(x_0)$, unless $\sigma_\varepsilon^2 = 0$. The second term is the squared bias, the amount by which the average of our estimate differs from the true mean; the last term is the variance; the expected squared deviation of $\hat{f}(x_0)$ around its mean. Typically the more complex we make the model $\hat{f}$, the lower the (squared) bias but the higher the variance.

Hastie et al, Elements of Statistical Learning

```
n_train  100  noise  0.5  n_bags  50
Tree: 0.4896 (error) = 0.0051 (bias^2)  + 0.2377 (var) + 0.2424 (noise)
Bagging(Tree): 0.3726 (error) = 0.0031 (bias^2)  + 0.1226 (var) + 0.2424 (noise)
```



Code for this example based off of https://scikit-learn.org/stable/auto_examples/ensemble/plot_bias_variance.html

Random Forests
= bagging to make a bunch of trees
+ each node selects from a random subset of features

Random Feature Subset for each Tree or Node?

Tin Kam Ho used the "**random subspace method,**" where each tree got a random subset of features.

"Our method relies on an autonomous, pseudo-random procedure to select a small number of dimensions from a given feature space …"

- Ho, Tin Kam. "The random subspace method for constructing decision forests." IEEE transactions on pattern analysis and machine intelligence 20.8 (1998): 832-844.

"Trademark" random forest:

"… random forest with random features is formed by selecting at random, at each node, a small group of input variables to split on."

- **Breiman**, Leo. "Random Forests" Machine learning 45.1 (2001): 5-32.

Random Feature Subset for each Tree or Node?

Tin Kam Ho used the “random subspace method,” where each tree got a random subset of features.

“Our method relies on an autonomous, pseudo-random procedure to select a small number of dimensions from a given feature space …”

- Ho, Tin Kam. “The random subspace method for constructing decision forests.” IEEE transactions on pattern analysis and machine intelligence 20.8 (1998): 832-844.

“Trademark” random forest:

“… random forest with random feature … andom, at each node, a small group of input variab…

- Breiman, Leo. “Random Forests” Ma… 2.

num features $= \log_2 m + 1$

where m is the number of input features

In contrast to the original publication
[Breiman, “Random Forests”, Machine Learning, 45(1), 5-32, 2001]
the scikit-learn implementation combines classifiers by averaging their probabilistic prediction, instead of letting each classifier vote for a single class.

"Soft Voting"

Features of Random Forests

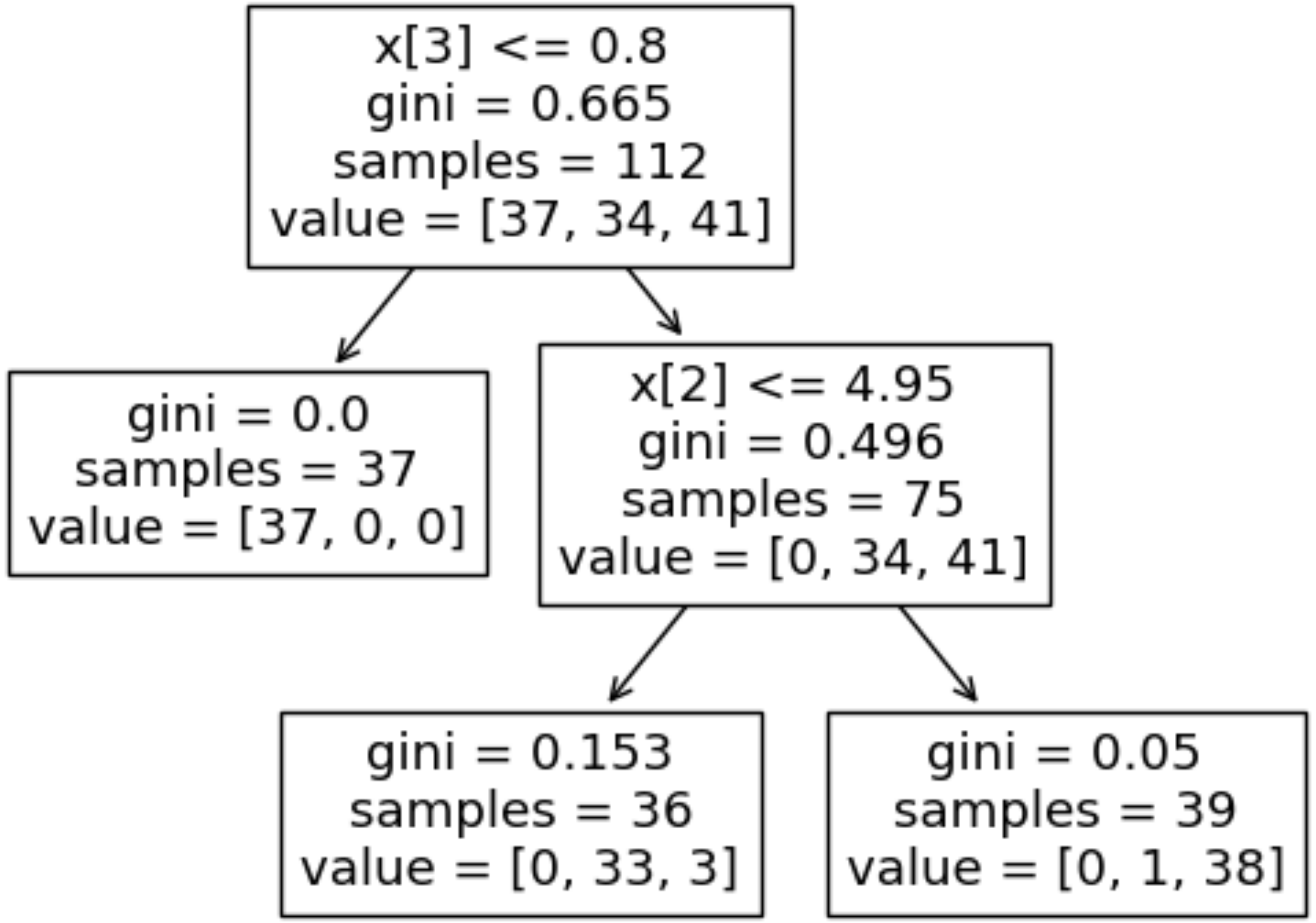
- GOOD ENOUGH performance and FAST to train on large datasets.
- It can handle thousands of input variables without needing other forms of feature selection (does FS itself…).
- It gives estimates of what features are important in the classification.
- It generates an internal unbiased estimate of the generalization error as the forest building progresses (OOB).
- It has an effective method for estimating missing data and maintains accuracy when a large proportion of the data are missing.

RF regression

- It has methods for balancing error in class population unbalanced data sets.

Weighting of classes

Feature importance in a single decision tree



Random Forest Feature Importance

"Method A" (this is used in scikit-learn)

Usually measured as

- impurity decrease (Gini, Entropy) for a given node/feature decision
- weighted by number of examples at that node
- averaged over all trees
- then normalize so that sum of feature importances sum to 1

- (Unfair for variables with many vs few values)

Random Forest Feature Importance

Method B: Permutation Importance

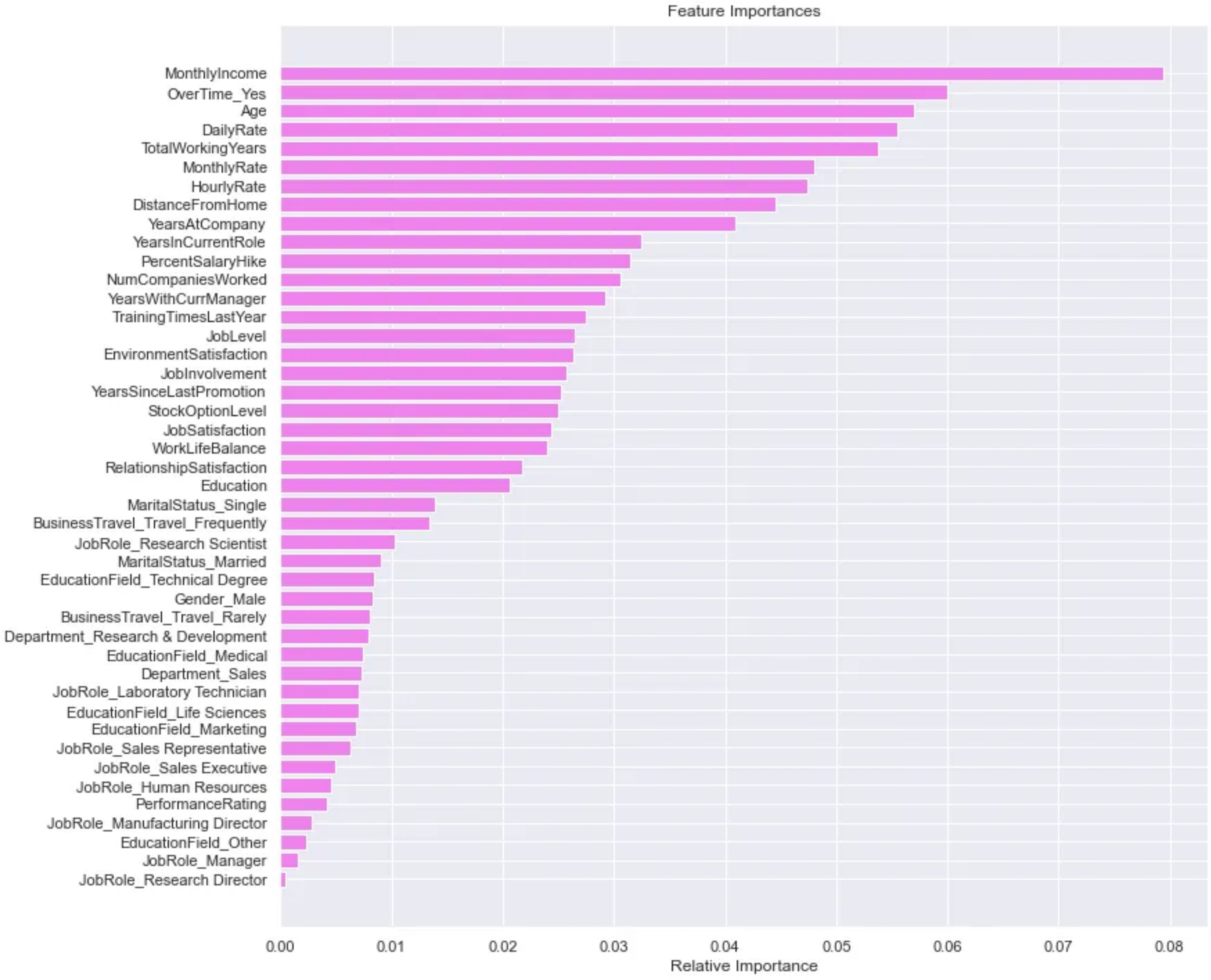
Out-of-bag accuracy:

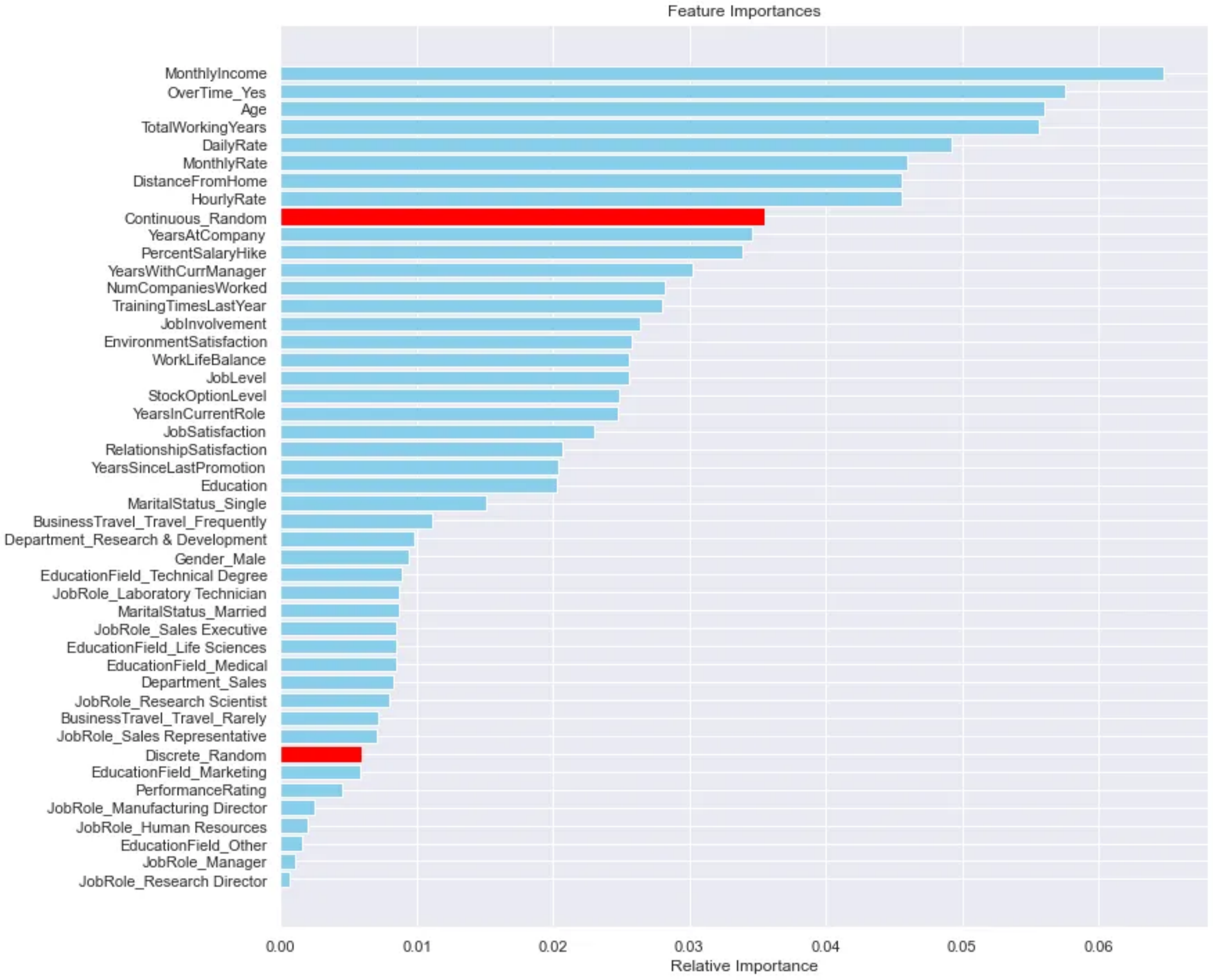
- During training, for each tree, make prediction for OOB sample (~1/3 of the training data)
- Based on those predictions where example *i* was OOB, compute label via majority vote among the trees that did not use example *i* during model fitting
- The proportion over all examples where the prediction (by majority vote) is correct is the OOB accuracy estimate

Out-of-bag feature importance via permutation:

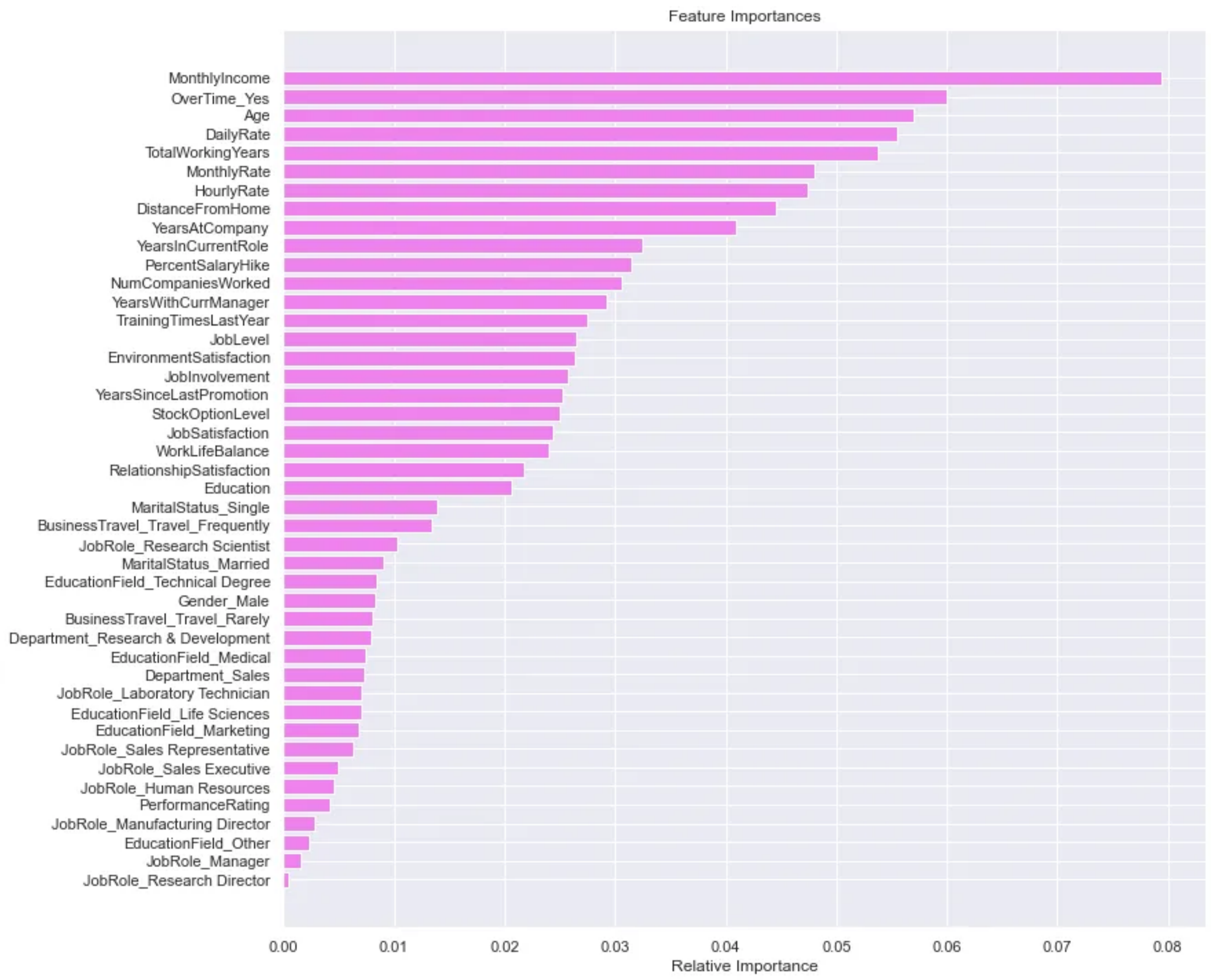
(we will also cover a generalized version with a hold out set later)

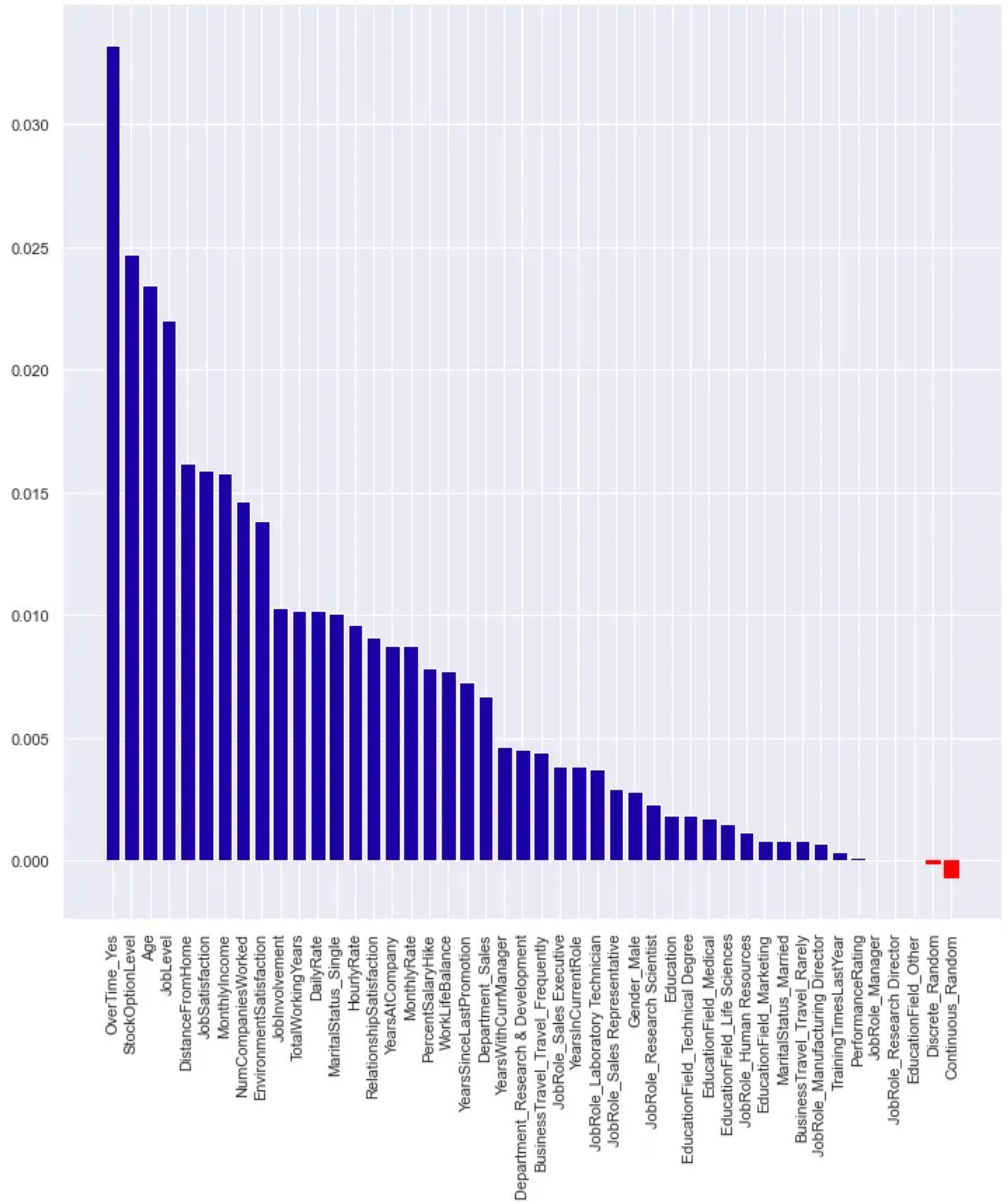
- Count votes for correct class
- Given feature *i*, permute this feature in OOB examples of a tree
- Compute the number of correct votes after permutation from the number of votes before permutation for given tree
- Repeat for all trees in the random forest and average the importance
- Repeat for other features





https://medium.com/@ali.soleymani.co/stop-using-random-forest-feature-importances-take-this-intuitive-approach-instead-4335205b933f





https://medium.com/@ali.soleymani.co/stop-using-random-forest-feature-impo