https://ttpoll.com/p/237202

COSC 325: Introduction to Machine Learning

Dr. Hector Santos-Villalobos

THE UNIVERSITY OF TENNESSEE KNOXVILLE

Lecture 13: Ensemble Methods

THE UNIVERSITY OF
TENNESSEE
KNOXVILLE

Class Announcements

Homework/Quizzes:

No homework or quiz this week.

Course Project:

Teaming issues. Please contact me.

Lectures:

No attendance record today due to the Engineering Expo

Exams:

Exam #1: This Thursday, 10/03

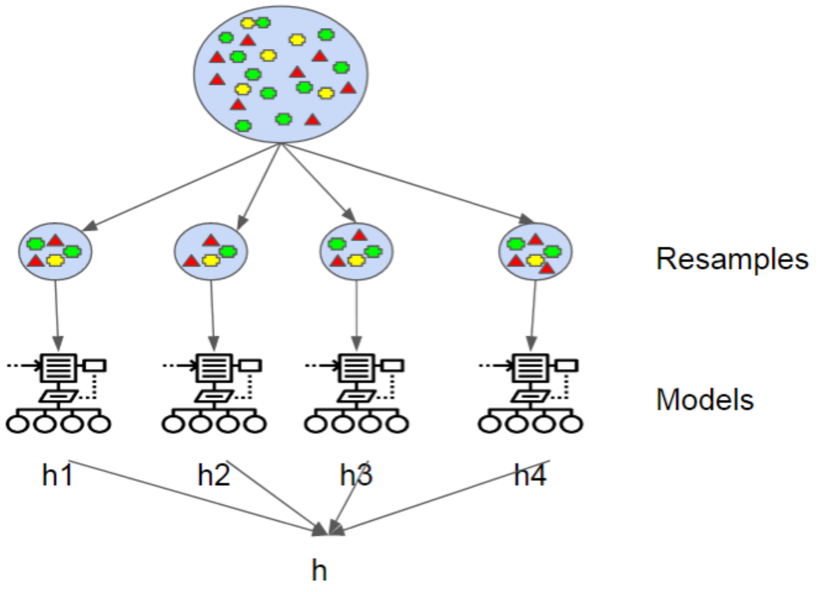
- Online
- Window 11 am to 1 pm
- 75 mins
- ***SDS accommodations set in Canvas***.

Review

- Decision Trees
  - Impurity
    - Entropy, Gini
  - Information Gain
    - Equivalent to Mutual Information
  - Shortcomings
    - Costly diagonal boundaries
    - Overfitting
    - Costly management of overfitting
  - Strengths
    - High capacity
    - Explainable
    - Simplicity and efficiency

Today's Topics

Ensemble Methods



Pop Quiz

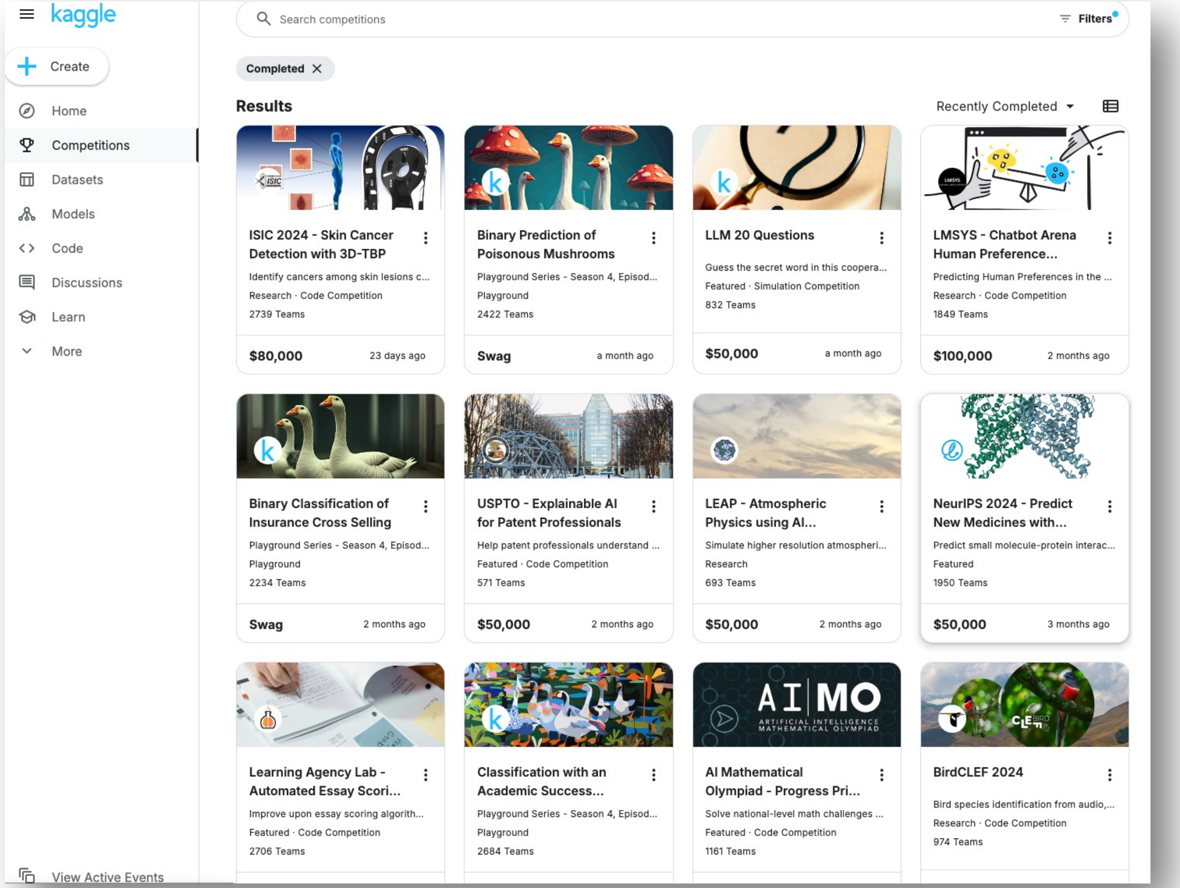
How many Lego blocks are in the jar?

Wisdom of the Crowd

- The collective opinion of a diverse independent group of individuals rather than a single expert
- ***One explanation:*** There is noise associated with each individual judgment, and taking the average over many responses will go some way toward canceling the effect of this noise

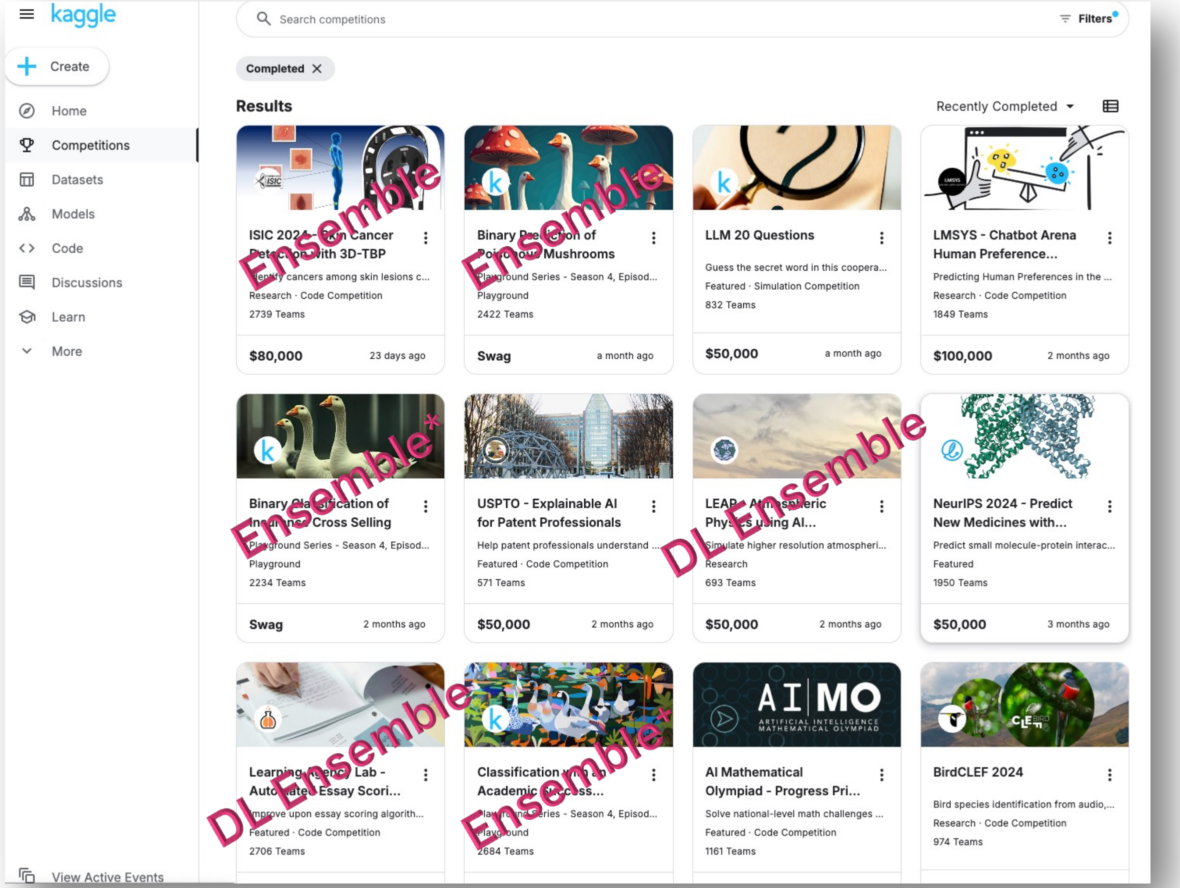
THE UNIVERSITY OF TENNESSEE KNOXVILLE

Ensemble Methods Motivation



- Joint model predictions tend to have lower bias and variance
- Ensemble techniques are the most widely used ML methods following DL techniques
  - More computationally efficient than DL techniques

Ensemble Methods Motivation



- Joint model predictions tend to have lower bias and variance
- Ensemble techniques are the most widely used ML methods following DL techniques
  - More computationally efficient than DL techniques

Example Project:
https://www.kaggle.com/competitions/isic-2024-challenge/overview

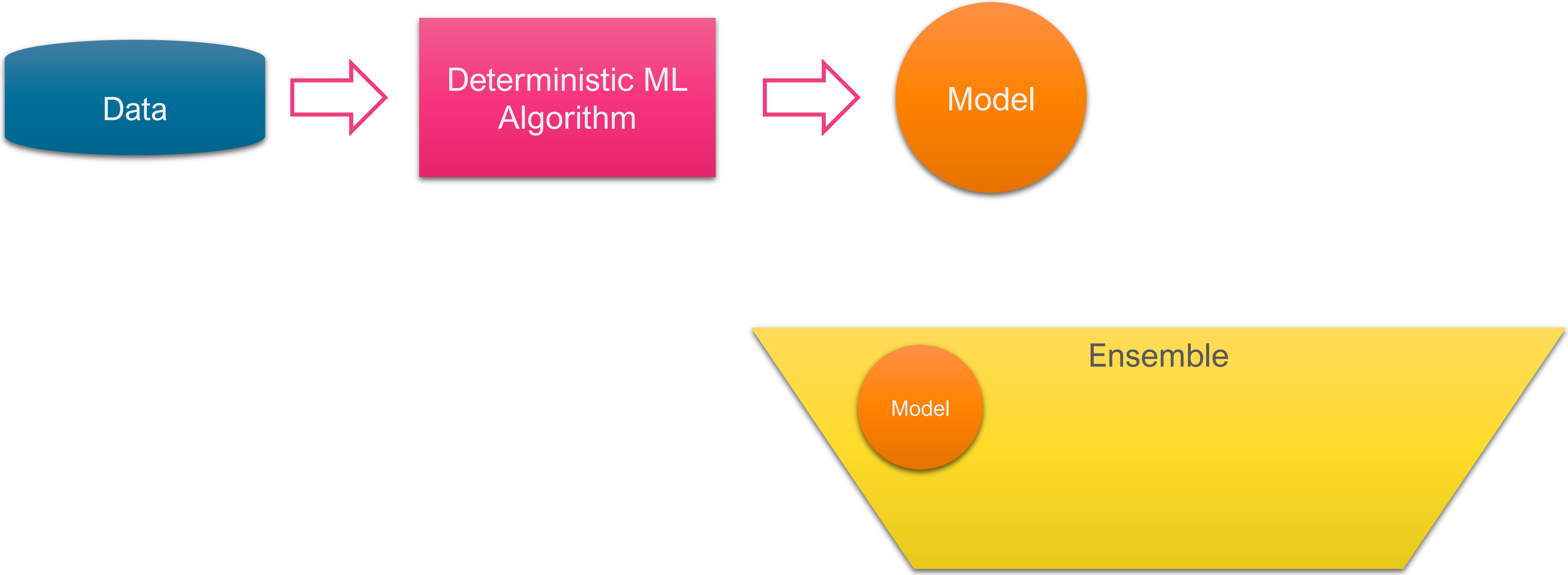
Ensemble methods still rock!

Ensemble Methods

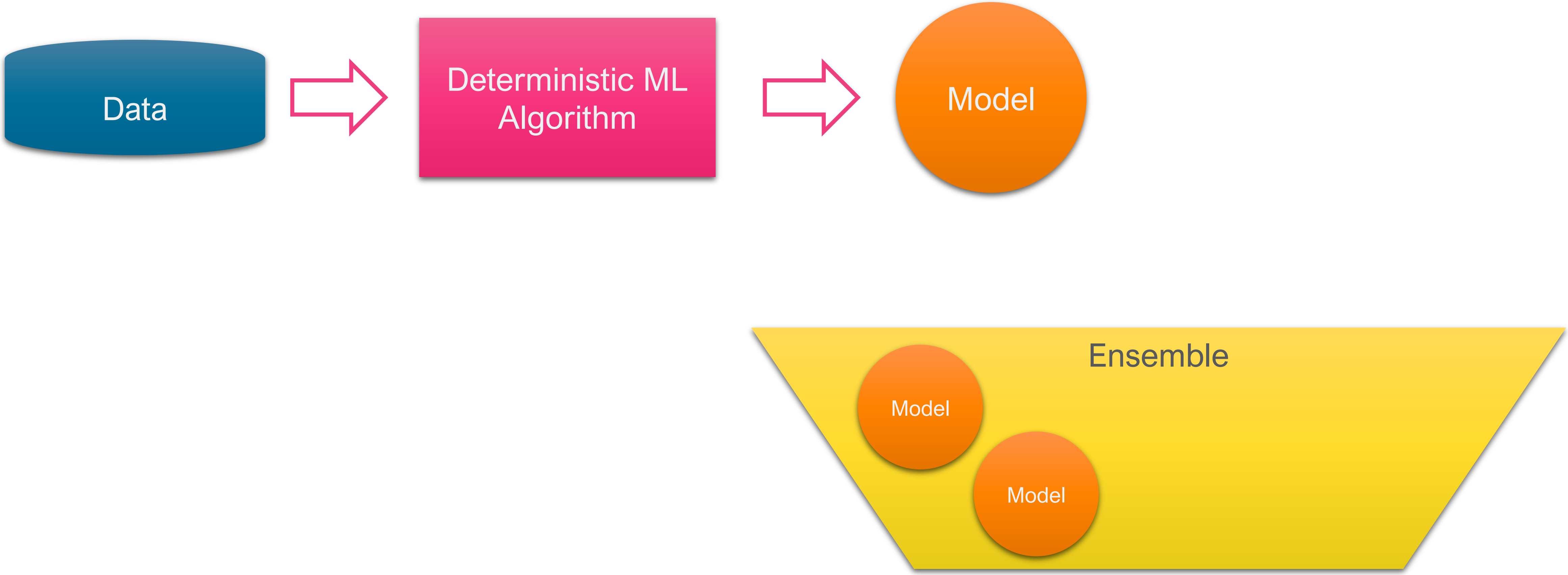
- Ensemble methods are learning models that combine the opinions of multiple learners
- When you're doing this, the individual learners can often end up being a lot simpler and still get really good performance
- Ensembles are also inherently parallel, which makes them more efficient at training and test time if you have access to multiple processors.

Ensembles

Ensembles



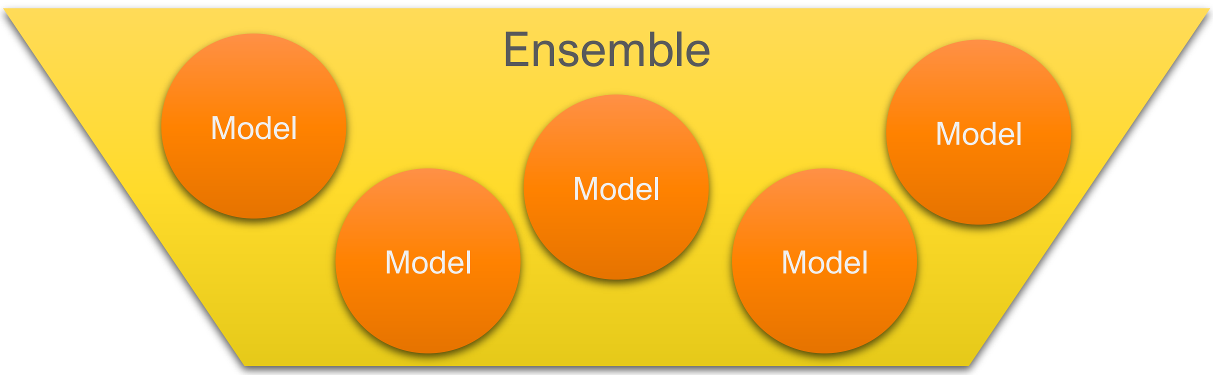
Ensembles



 Slide Credit: Dr. Schumman

Ensembles

Are these models different?



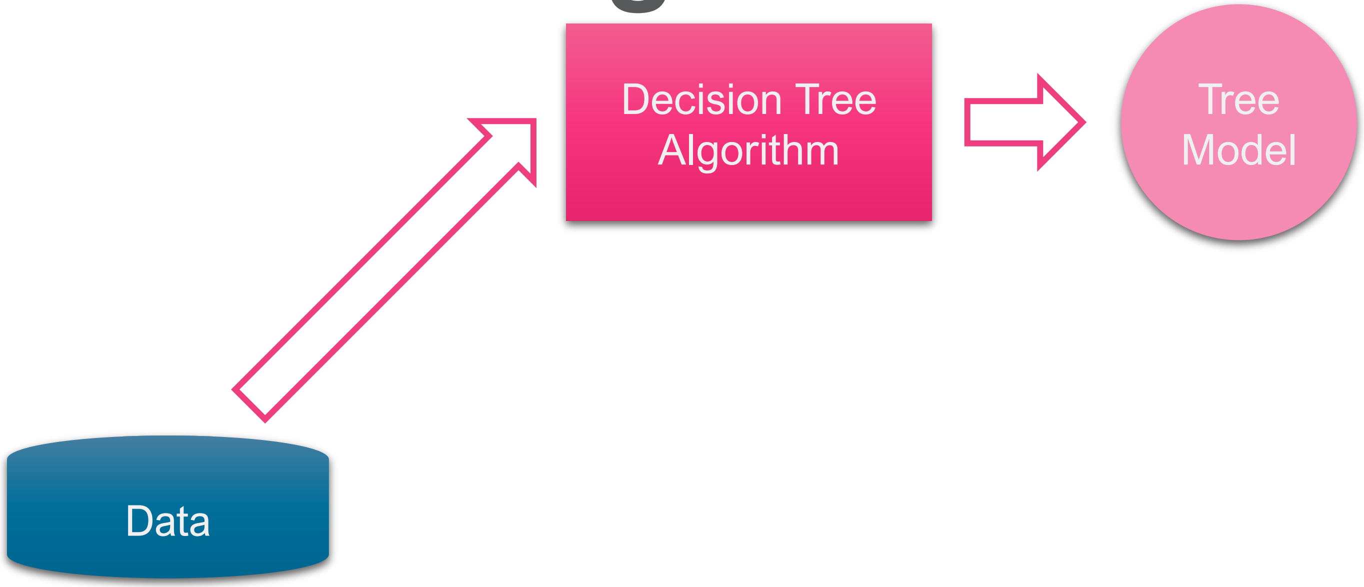
Moving to Ensembles

- If your learning algorithm is deterministic, then the algorithm will produce ***the same model given the same dataset***
- To really get the most out of your learning algorithm, you need diversity/variability!
- To do this, you can either change the ***learning algorithm*** or change the ***data set***

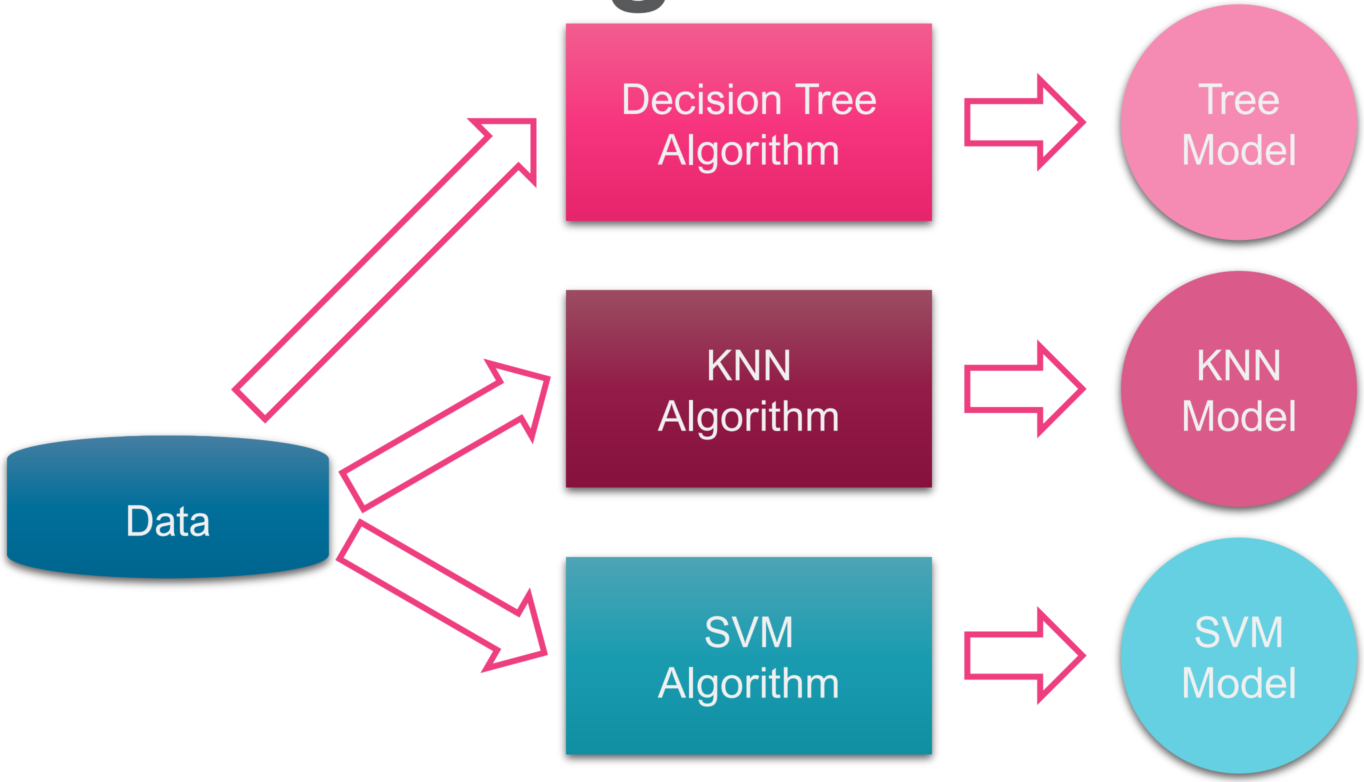
Different Models

- Let's do a binary classification problem: {Upgrade Car, Keep Car}
- Instead of learning a single classifier, you might learn a bunch of different classifiers:
  - Decision tree
  - Logistic Regression
  - KNN
  - SVM
  - Multiple neural networks with different architectures

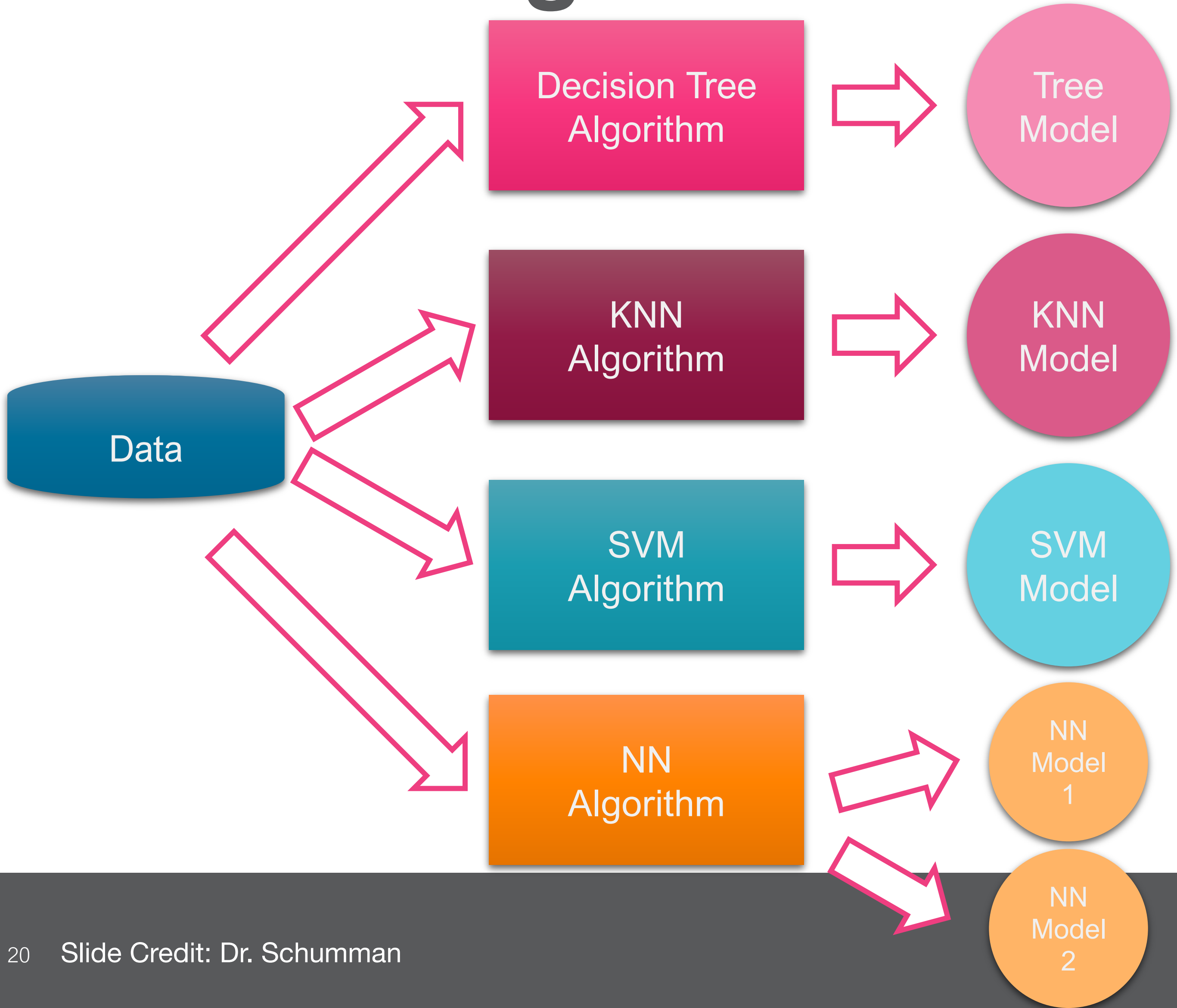
Generating Ensemble



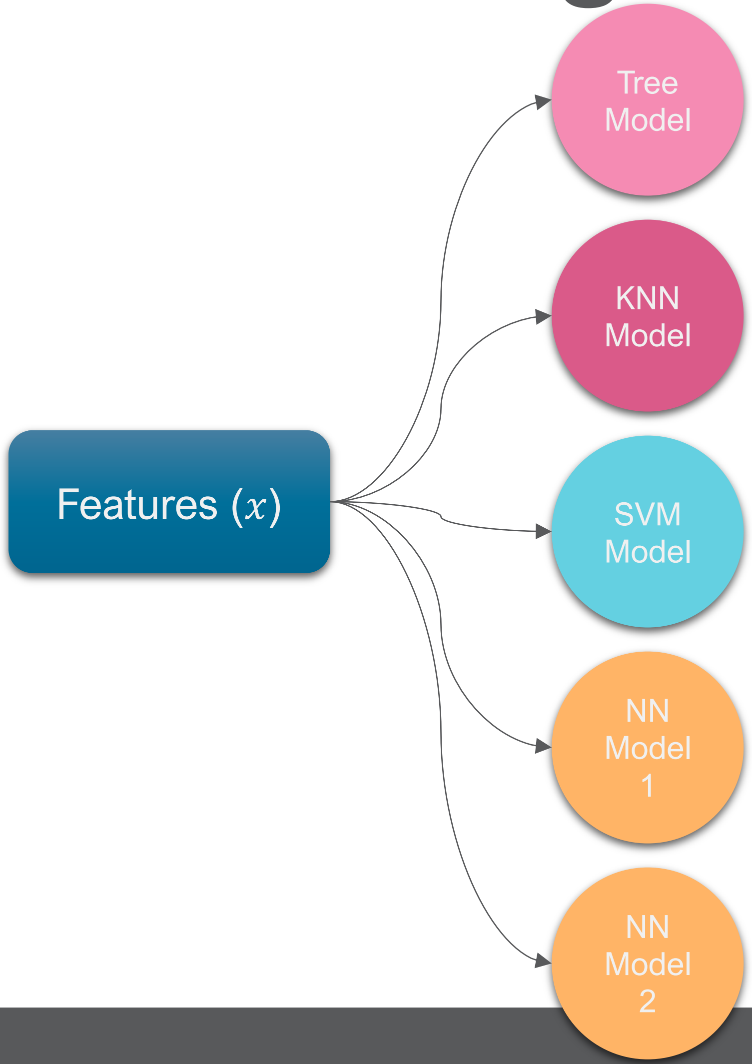
Generating Ensemble



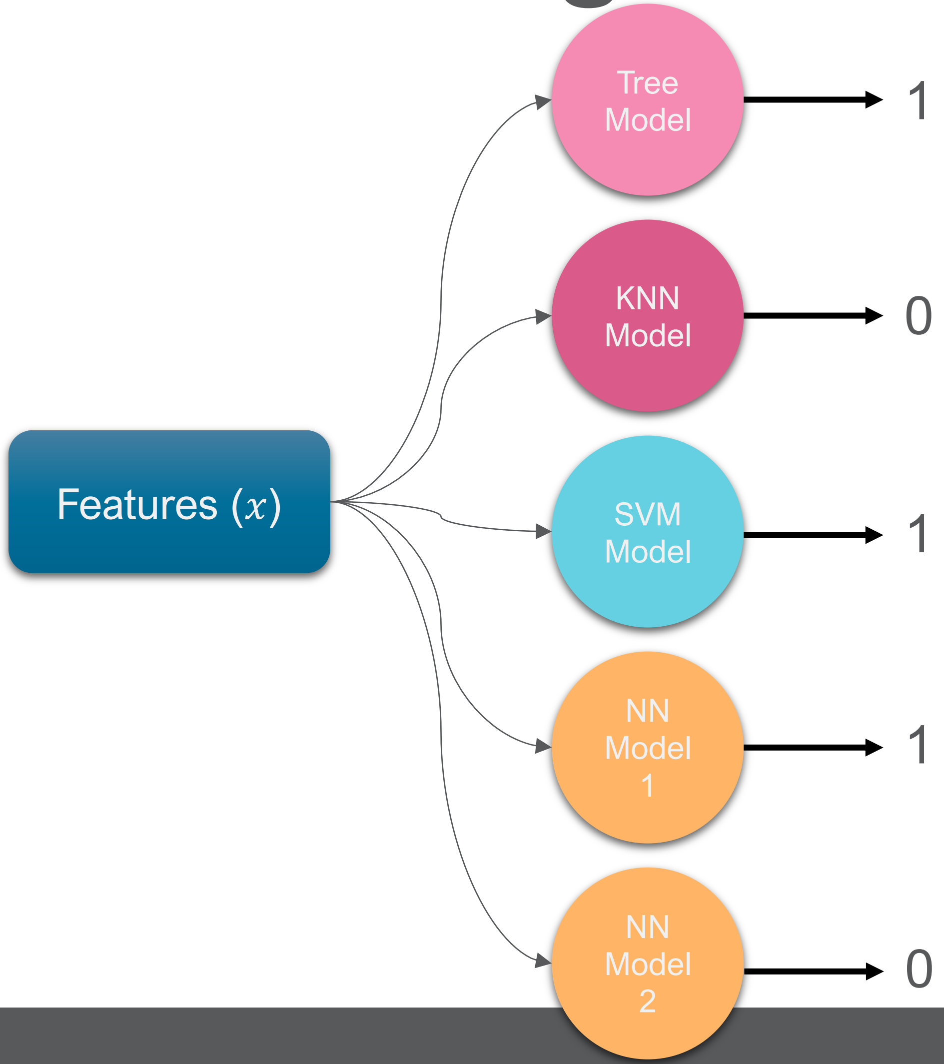
Generating Ensemble



Predicting with the Ensemble

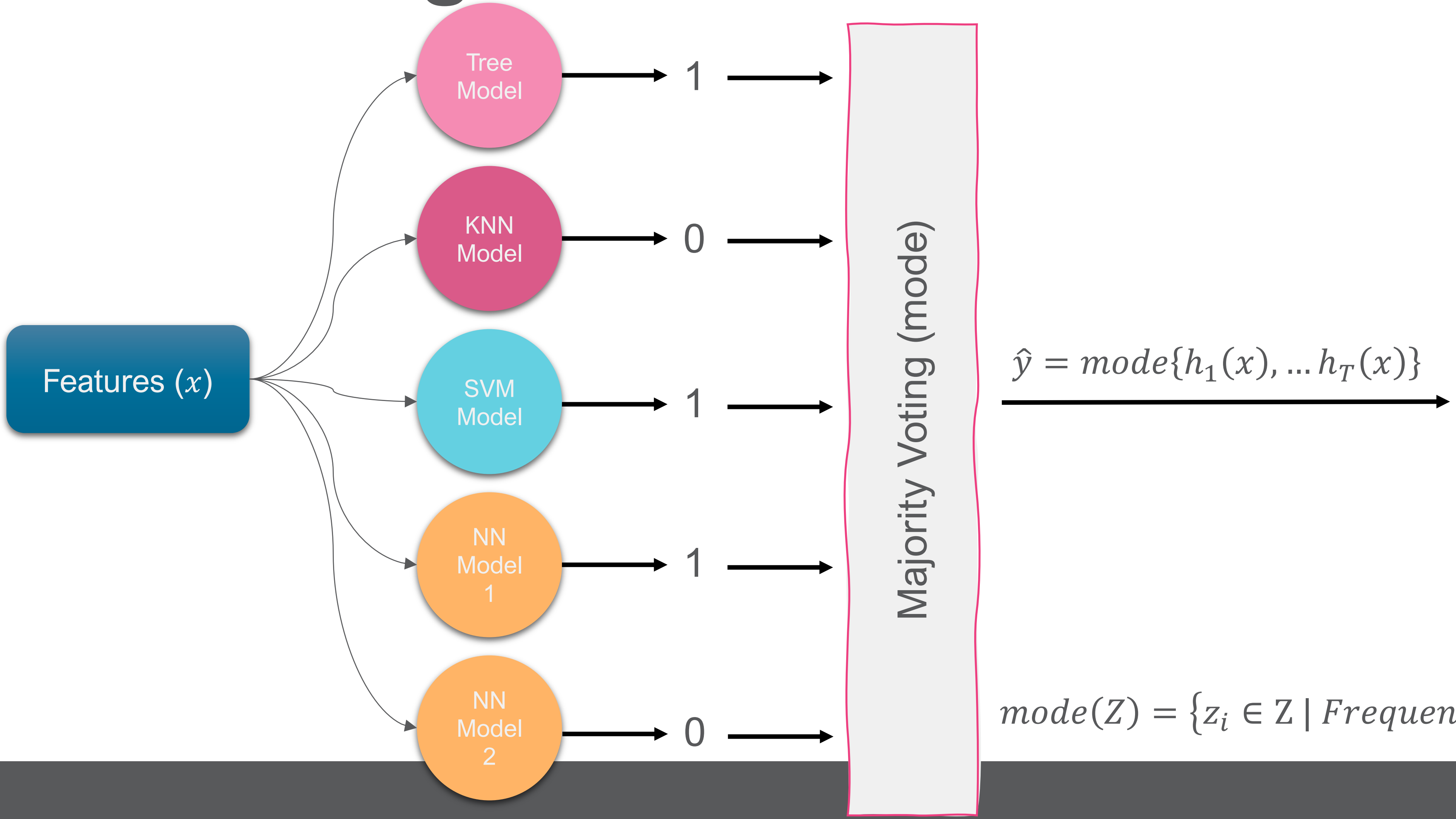


Predicting with the Ensemble



What could be the output of the ensemble?

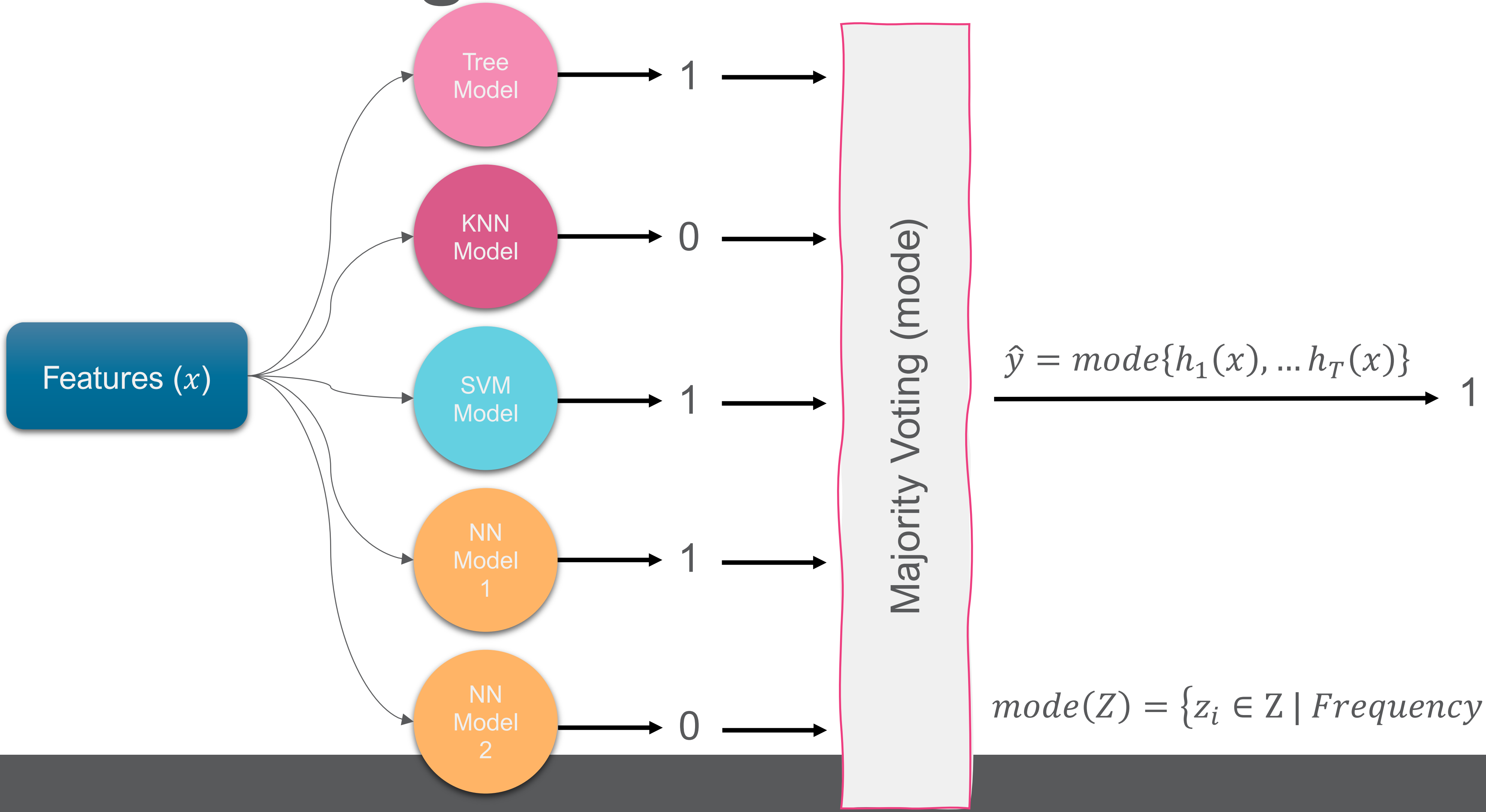
Predicting with the Ensemble



$$\hat{y} = mode\{h_1(x), \dots h_T(x)\}$$

$$mode(Z) = \{z_i \in Z \mid Frequency(z_i) \geq Frequency(z_j), \forall j\}$$

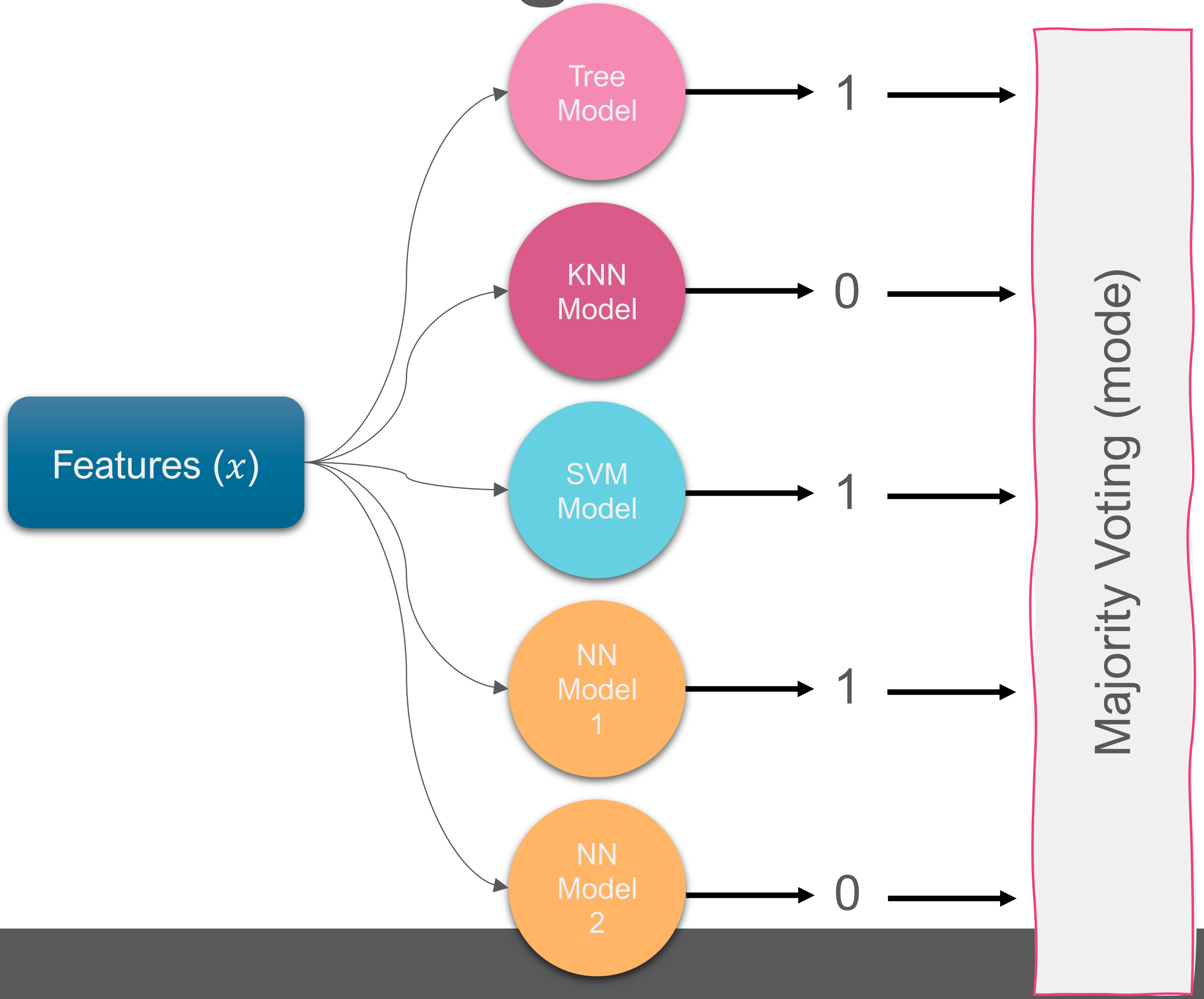
Predicting with the Ensemble



$$\hat{y} = mode\{h_1(x), \ldots h_T(x)\}$$

$$mode(Z) = \{z_i \in Z \mid Frequency(z_i) \geq Frequency(z_j), \forall j\}$$

Predicting with the Ensemble



For $\hat{y} \in \{-1,1\}$, and $h_t(x) \in \{-1,1\}$:

$$\hat{y} = H(x) = sign(\sum_{t=1}^{T} h_t(x))$$

$$\hat{y} = mode\{h_1(x), \dots h_T(x)\}$$

→ 1

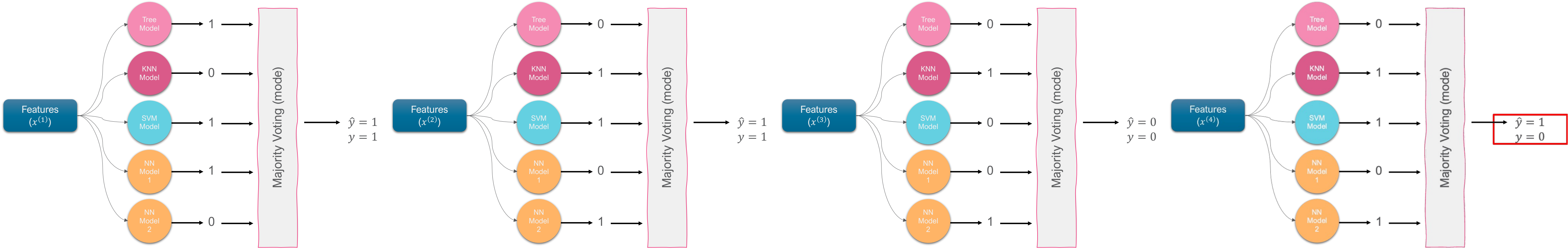
$$mode(Z) = \{z_i \in Z \mid Frequency(z_i) \geq Frequency(z_j), \forall j\}$$

Advantages of Ensembles

- Main advantage: ***Unlikely*** that all classifiers will make the ***same mistake***

Advantages of Ensembles

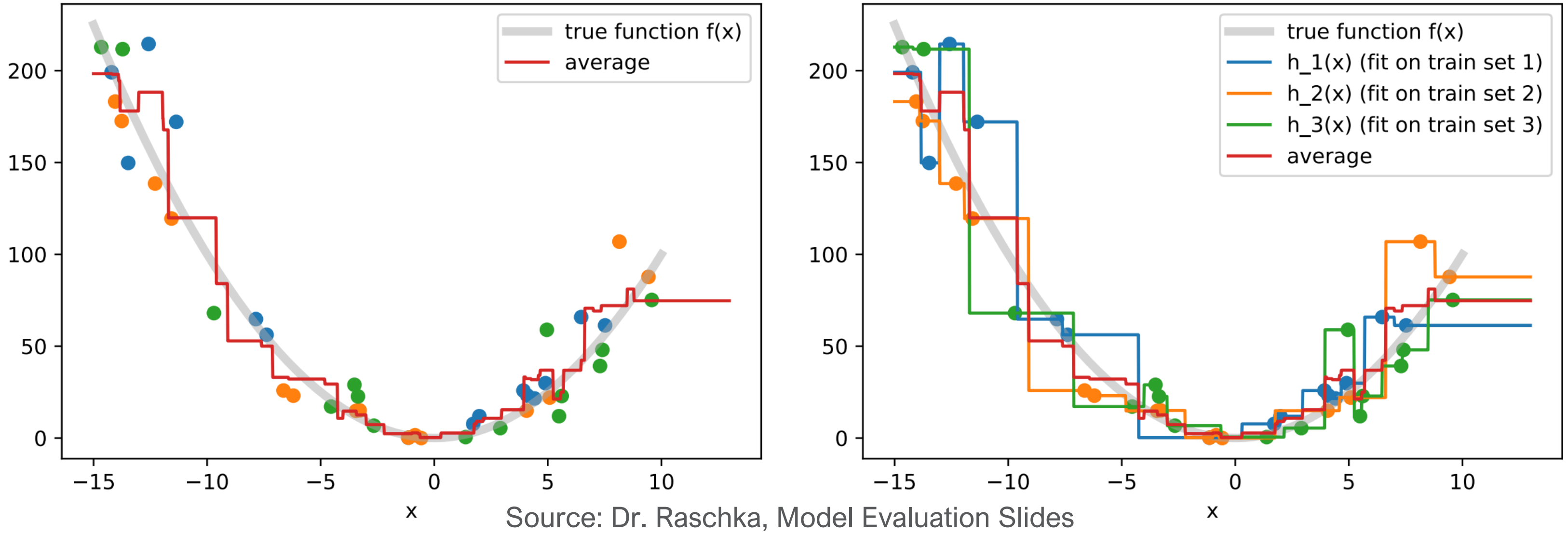
- Main advantage: ***Unlikely*** that all classifiers will make the ***same mistake***



- As long as a minority of classifiers makes every error, you will achieve optimal classification!

Advantages of Ensembles

- Main advantage: ***Unlikely*** that all classifiers will make the ***same mistake***
- Useful when individual models are high variance



Source: Dr. Raschka, Model Evaluation Slides

Pop Quiz

What types of errors are ensemble methods most effective at addressing? Select all that apply.

A. Variance

B. Bias

C. Irreducible Error

Ensembles and Types of Error

- Unfortunately, the ***inductive biases*** of different algorithms are highly correlated.
  - Example 1: Assume non-linear or linear relationship between y and x.
  - Example 2: Assume small changes in feature values leads to small changes in the model output.
  - Example 3: Application of regularization, which leads to lower capacity models.
- Irreducible errors are still present
  - Examples: missing features, noise in measurements, etc.

Voting Alternatives

- Classification
  - Majority voting (mode)

Binary: $\{1,1,0,1,0,1\} \rightarrow 1$

Categorical: $\{A, B, A, C, C, A\} \rightarrow A$

Voting Alternatives

- Classification
  - Majority voting (mode)

Binary: $\{1,1,0,1,0,1\} \rightarrow 1$

Categorical: $\{A, B, A, C, C, A\} \rightarrow A$

- Regression
  - Mean, median

Regression: $\{5.43, 5.44, 5.38\}$

Voting Alternatives

- Classification
  - Majority voting (mode)

Binary: $\{1,1,0,1,0,1\} \rightarrow 1$

Categorical: $\{A, B, A, C, C, A\} \rightarrow A$

- Regression
  - Mean, median

Regression: $\{5.43,5.44,5.38\} \xRightarrow{mean} 5.42$

$\xRightarrow{median} 5.43$

Voting Alternatives

- Classification
  - Majority voting (mode)

Binary: $\{1,1,0,1,0,1\} \rightarrow 1$

Categorical: $\{A, B, A, C, C, A\} \rightarrow A$

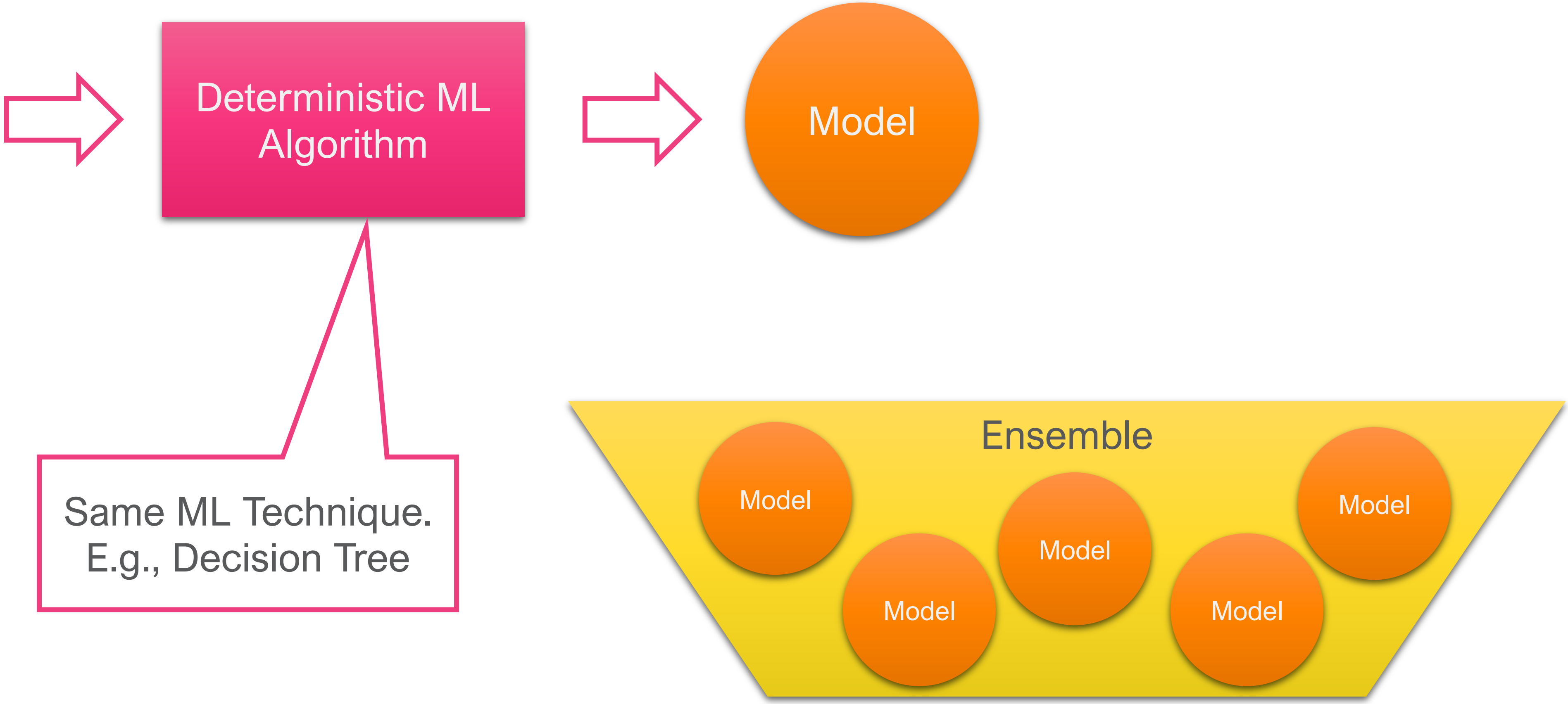
- Regression
  - Mean, median

Regression: $\{5.43,5.44,5.38\} \xRightarrow{mean} 5.42$

$\xRightarrow{median} 5.43$

- Ranking
  - Rank Aggregation (Similar to how College football teams are ranked)
  - Relevance scores, then take the average relevance score for ranking.

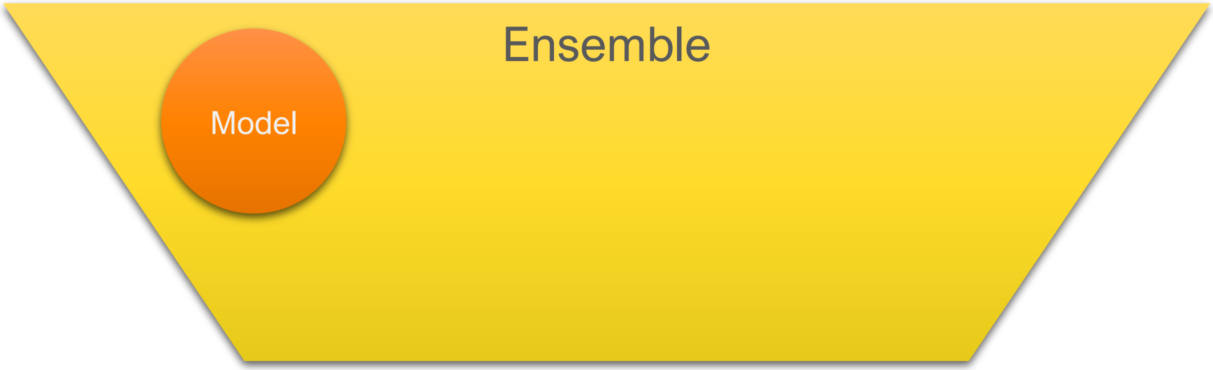
Other Ensembles Strategies



$$Data = \bigcup_{i=1}^{n} Data_i$$

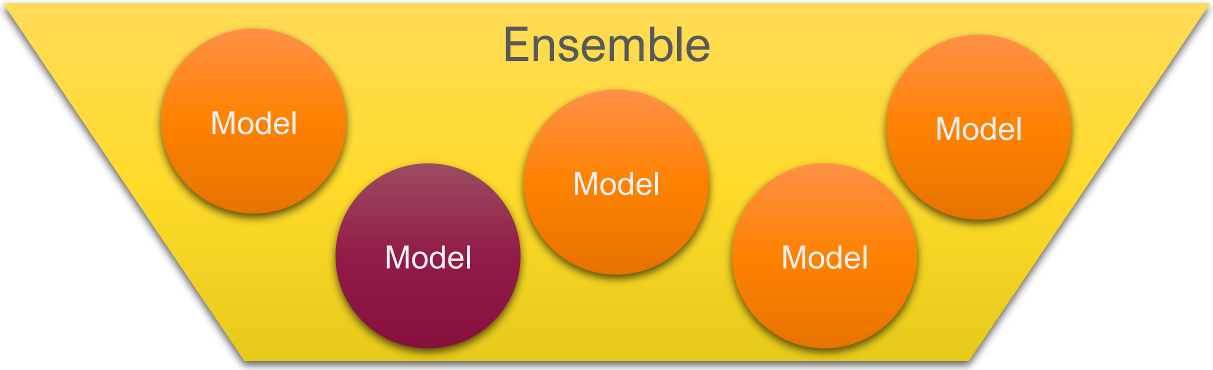
Other Ensembles Strategies

$$Data = \bigcup_{i=1}^{n} Data_i$$



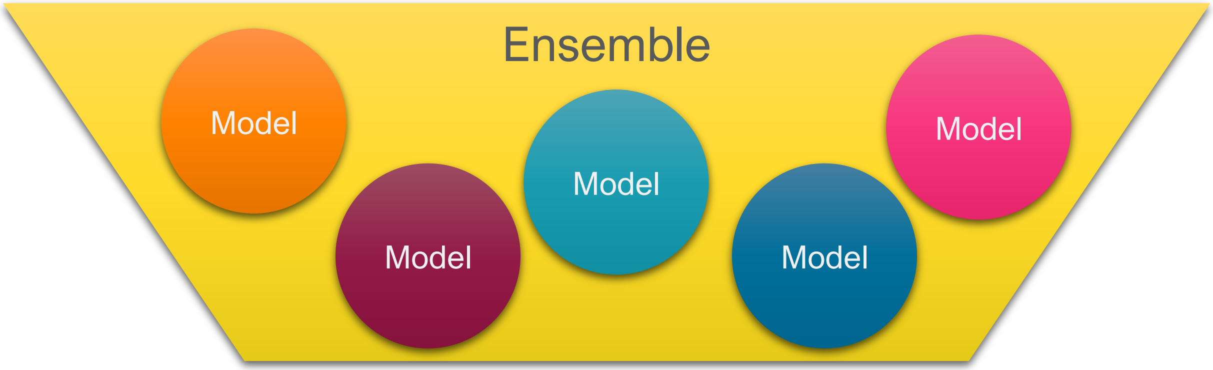
Other Ensembles Strategies

$$Data = \bigcup_{i=1}^{n} Data_i$$



Other Ensembles Strategies

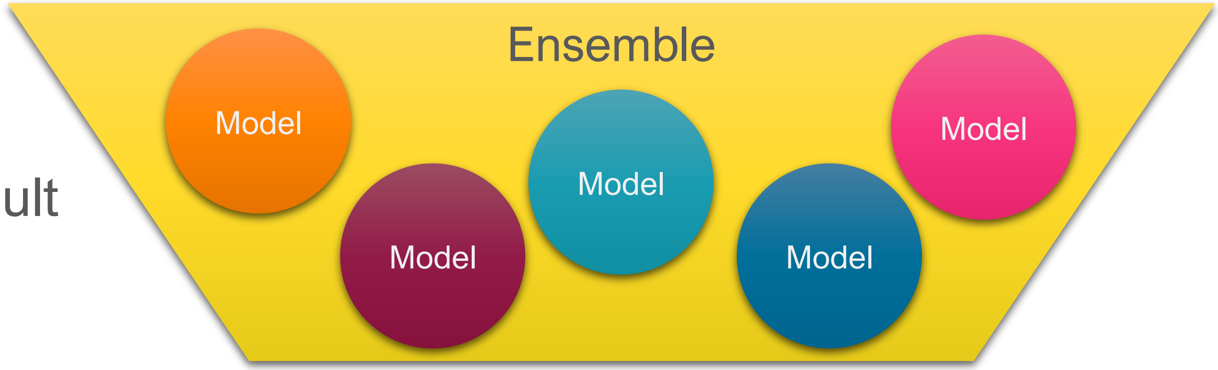
$$Data = \bigcup_{i=1}^{n} Data_i$$



What could be a problem with this approach?

Other Ensembles Strategies

$$Data = \bigcup_{i=1}^{n} Data_i$$

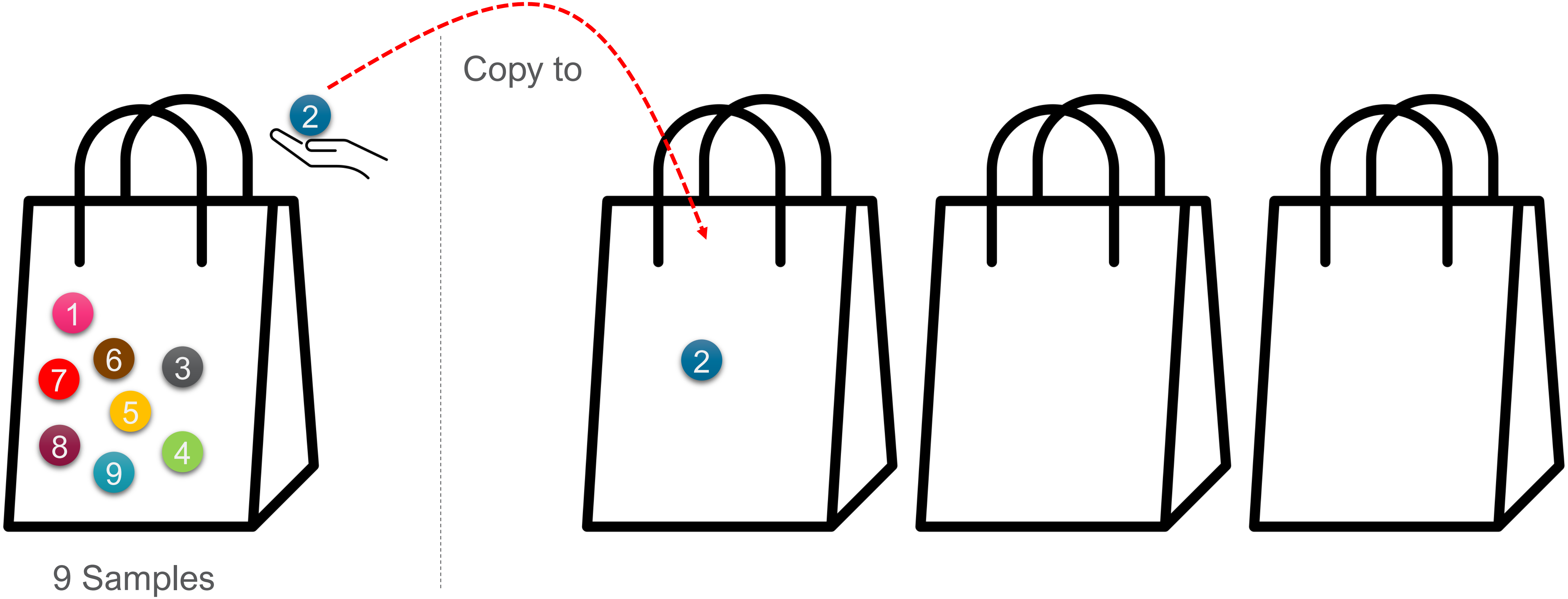


Splitting data into multiple smaller datasets may result in high variance models with poor ensemble performance.

Bagging

9 Samples

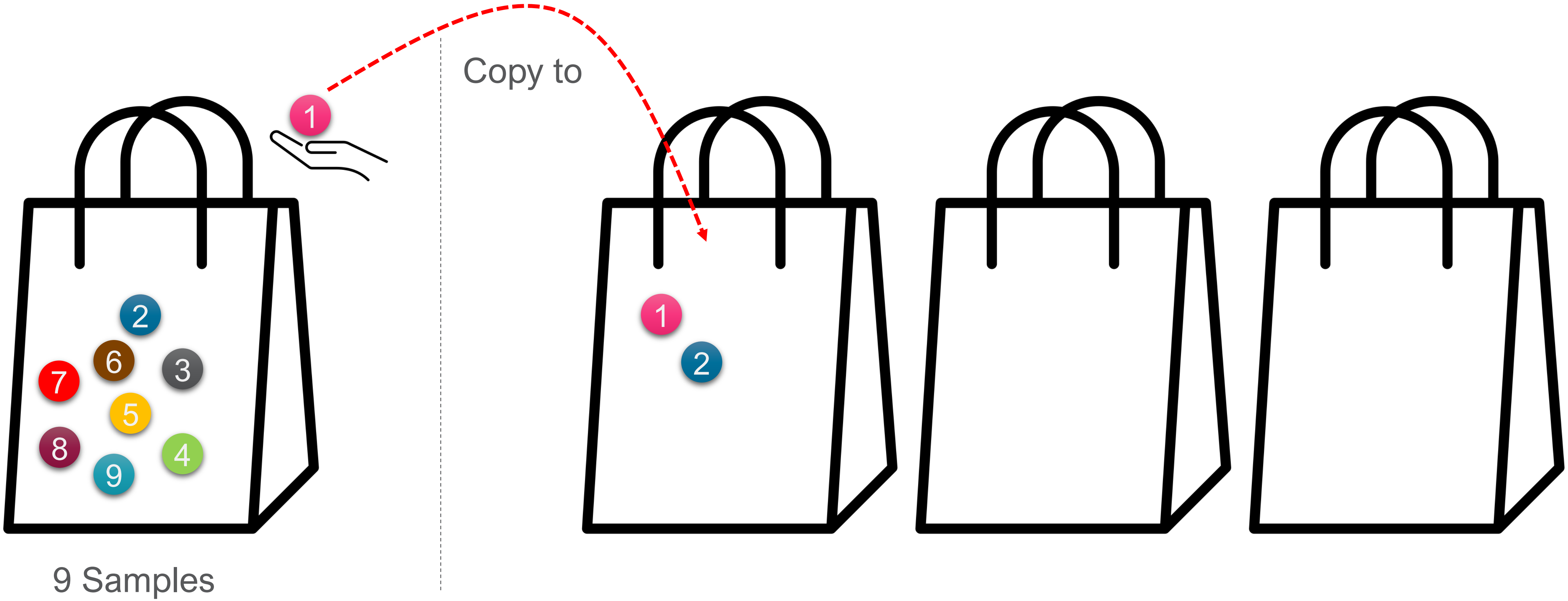
Bagging



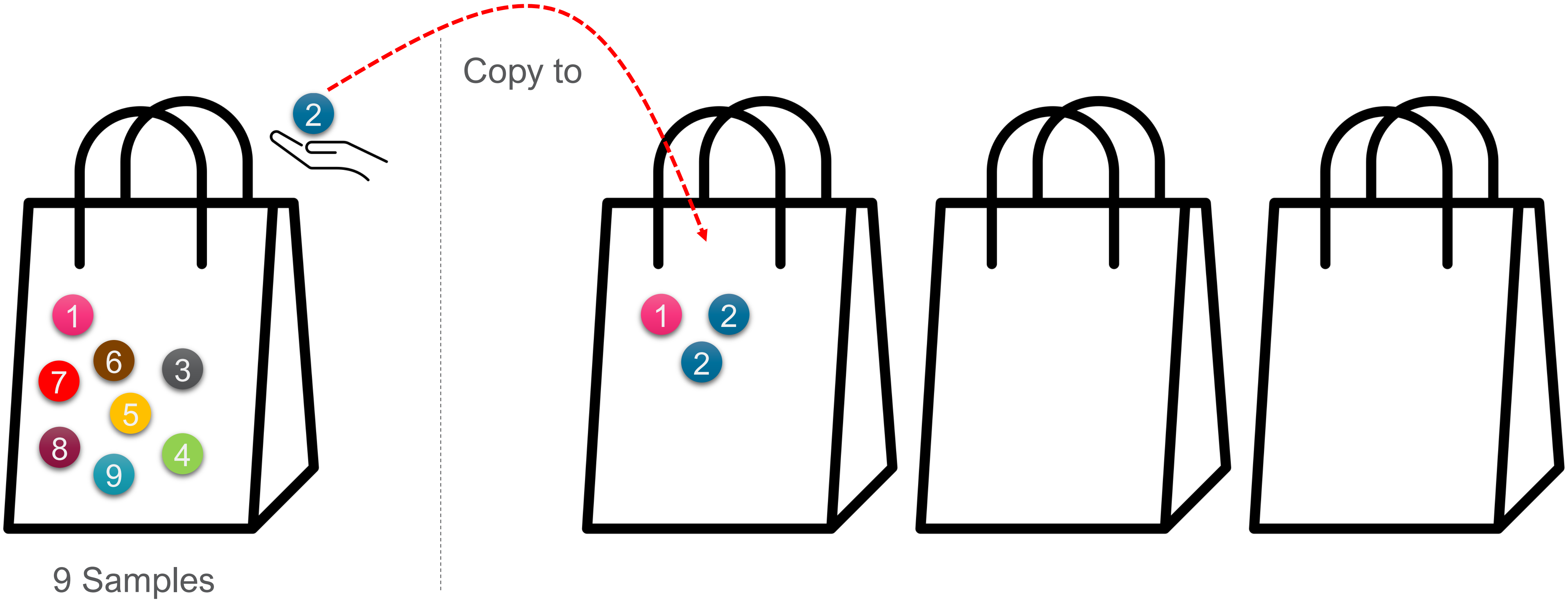
Bagging

9 Samples

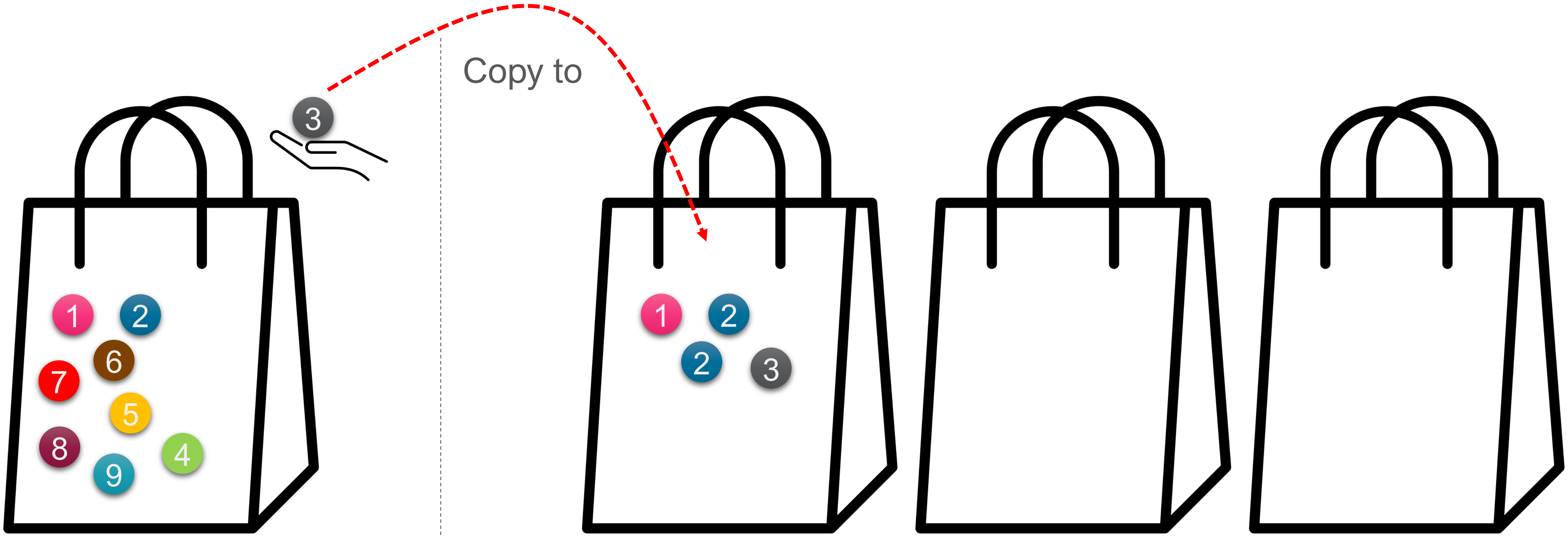
Bagging



Bagging



Bagging



9 Samples

Bagging

9 Samples

9 Samples

Bagging

Bootstrap Resampling

- Statistical technique
- Observation
  - The data set we're given D is drawn i.i.d. (independently and identically distributed) from an unknown distribution $f(X, y)$
  - If we draw a new dataset $\widetilde{D}$ by random sampling from D with ***replacement***, then $\widetilde{D}$ is also a sample from $f(X, y)$

Bagging Algorithm

- Given: dataset D with n samples
- We create M bootstrapped sets $\tilde{D}_1, \tilde{D}_2, \tilde{D}_3, \ldots, \tilde{D}_M$
- Each bootstrapped set contains n training examples drawn randomly from D with ***replacement***
- Then, we can train M classifiers $h_{\tilde{D}_1}, h_{\tilde{D}_2}, \ldots, h_{\tilde{D}_M}$
- After training the ensemble of M models, proceed with testing as before.

Bagging

Properties of Bagging

- The bootstrapped data sets will be similar but ***not too similar.***

- If n is large, the number of examples that are not present in any particular bootstrapped sample is relatively large.

Properties of Bagging

- The probability that the first training example is NOT selected is

$$P_{Sel} = \left(1 - \frac{1}{n}\right)$$

- The probability that it's not selected at all is

$$P_{All} = P_{Sel}^{n} = \left(1 - \frac{1}{n}\right)^{n}$$

- As $n \to \infty$, $P_{All} = \frac{1}{e} \approx 0.3679$

Properties of Bagging

- The probability that the first training example is NOT selected is

$$P_{Sel} = \left(1 - \frac{1}{n}\right)$$

- The probability that it's not selected at all is

$$P_{All} = P_{Sel}^{n} = \left(1 - \frac{1}{n}\right)^{n}$$

- As $n \to \infty$, $P_{All} = \frac{1}{e} \approx 0.367\overline{9}$

For $n = 1{,}000$:

$$P_{all} = \left(1 - \frac{1}{1000}\right)^{1000} = 0.36769\overline{5}$$

Properties of Bagging

- The probability that the first training example is NOT selected is

$$P_{Sel} = \left(1 - \frac{1}{n}\right)$$

- The probability that it's not selected at all is

$$P_{All} = P_{Sel}^{n} = \left(1 - \frac{1}{n}\right)^{n}$$

- As $n \to \infty$, $P_{All} = \frac{1}{e} \approx 0.3679$

On average only 63% of the original training examples will be represented in any given bootstrapped set.

Bagging and Overfitting

- Bagging tends to reduce variance, so it provides an alternative to regularization.
- Even if learned classifiers $h_{\tilde{D}_1}, h_{\tilde{D}_2}, \ldots, h_{\tilde{D}_M}$ are individually overfit, they're likely to overfit to different things.
- Through voting, we can overcome a significant portion of the overfitting.

Pop Quiz

True or False. When using the bagging approach to sample a dataset D (consisting of n samples) to form subsets D1 and D2, D will end up with zero samples, and the number of samples for D1 and D2 will be half the original number of samples in D (n/2).

A. True

B. False

Boosting

THE UNIVERSITY OF TENNESSEE KNOXVILLE

Strong and Weak Learning Algorithms

- A ***strong*** learning algorithm $\mathcal{L}$ given
  - A desired error rate ϵ,
  - A failure probability δ, and
  - Access to enough labeled examples from distribution $\mathcal{D}$
- Then, with high probability (at least $1 - \delta$), $\mathcal{L}$ learns a classifier h that has error at most ϵ.

Strong and Weak Learning Algorithms

Building a ***strong*** learning algorithm might be difficult!

Strong and Weak Learning Algorithms

We instead build a ***weak*** learning algorithm $\mathcal{W}$ that only has to achieve an error rate ***slightly better than random***.

Boosting

- Tries to turn a weak learning $\mathcal{W}$ algorithm into a strong learning algorithm $\mathcal{L}$ by an ensemble of multiple weak models.

- AdaBoost: Adaptive Boosting Algorithm
  - It doesn't require a large number of hyperparameters
  - It runs in polynomial time

AdaBoost: Intuition

- Suppose you're studying for an exam by using past exams
- You take the past exams and grade yourself

- *Which questions do you focus your studying on?*

AdaBoost: Intuition

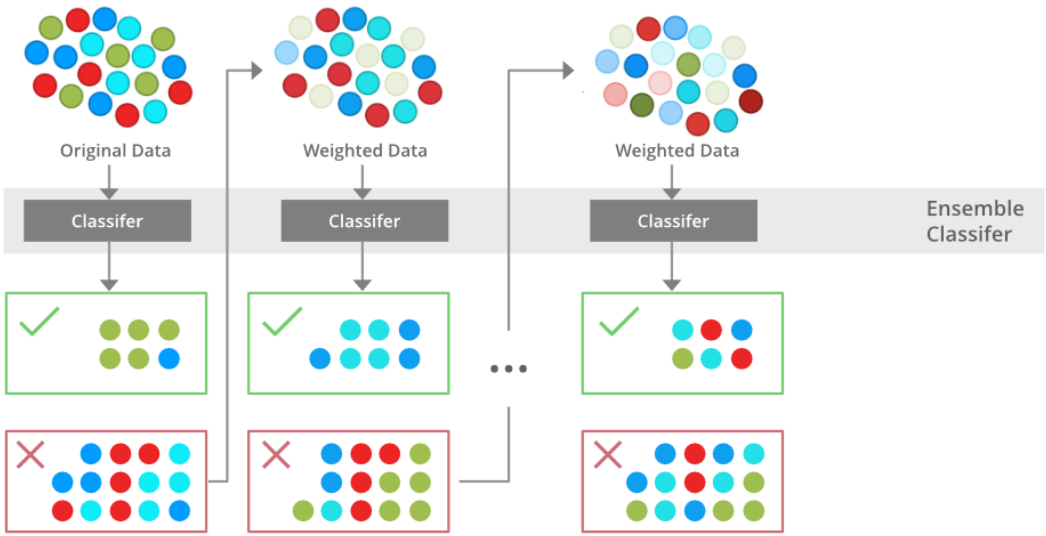
- Suppose you're studying for an exam by using past exams
- You take the past exams and grade yourself
- Most likely you…
  - ***Pay less attention*** to the questions that you got right
  - ***Study more*** the questions that you got wrong
  - You ***retake*** the exam and ***repeat*** this process until mastery

AdaBoost: Intuition

- Suppose you’re studying for an exam by using past exams
- You take the past exams and grade yourself

- ***Is it possible that you get stronger for your previously weak topics and weaker for your previously strong topics?***

AdaBoost Algorithm



Source: https://www.geeksforgeeks.org/implementing-the-adaboost-algorithm-from-scratch/

Algorithm 1 AdaBoost Algorithm

1: **Input:** Training dataset $(X, y) \in D$ with n samples and $y \in \{-1, 1\}$, The number of $\mathcal{W}$ models to train T.

2: **Initialize:** Weights $w_0 = \left[\frac{1}{n}, \frac{1}{n}, ..., \frac{1}{n}\right]^T$ with same shape as y.

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

4: Train a weak learner $\hat{y} = h_t(X, w_{t-1})$ using samples weights w_{t-1}.

5: Compute the weak learner's weighted error: $\varepsilon_t = (w_{t-1})^T I(y \neq \hat{y})$

6: Compute the weight of the weak learner: $\alpha_t = \frac{1}{2} \log\left((1 - \varepsilon_t) / \varepsilon_t\right)$

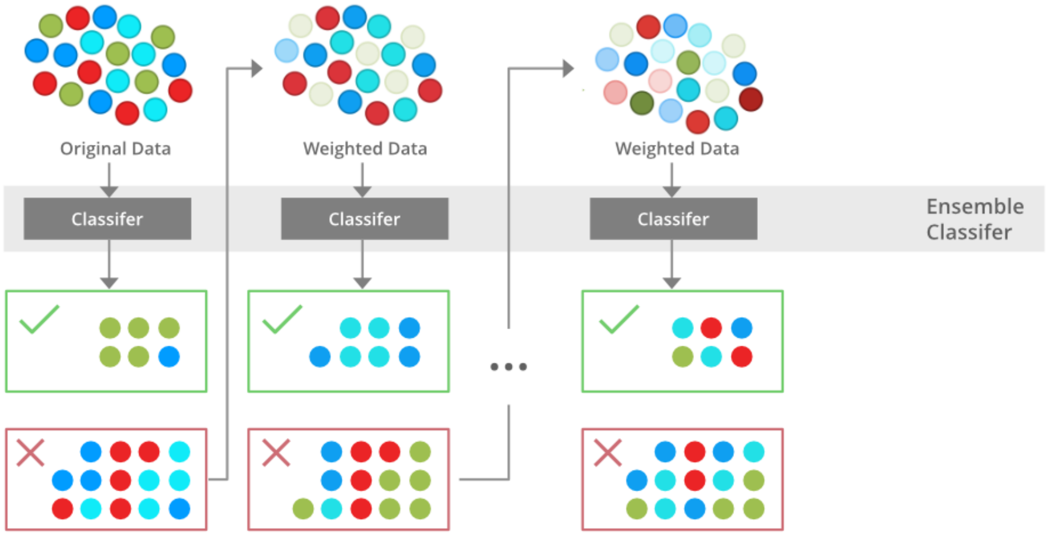
7: Update training sample weights: $w_t = w_{t-1} \circ \exp(-\alpha_t y \circ \hat{y})$

8: Normalize the weights: $w_t = w_t / \left(\sum_{i=1}^n w_t^{(i)}\right)$

9: **end for**

10: **Return:** $H(x) = \text{sign}\left(\sum_{t=1}^T \alpha_t h_t(x)\right)$

AdaBoost Algorithm



Source: https://www.geeksforgeeks.org/implementing-the-adaboost-algorithm-from-scratch/

Algorithm 1

For example, scaling the loss:

$$J(\theta, w) = \sum_{i=1}^{n} w^{(i)} \mathcal{L}(y^{(i)}, \hat{y}^{(i)})$$

1: **Input:** Tr … $\{-1, 1\}$, The number of …

2: **Initialize:** …

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

4: Train a weak learner $\hat{y} = h_t(X, w_{t-1})$ using samples weights w_{t-1}.

5: Compute the weak learner's weighted error: $\varepsilon_t = (w_{t-1})^T I(y \neq \hat{y})$

6: Compute the weight of the weak learner: $\alpha_t = \frac{1}{2} \log\left((1 - \varepsilon_t)/\varepsilon_t\right)$

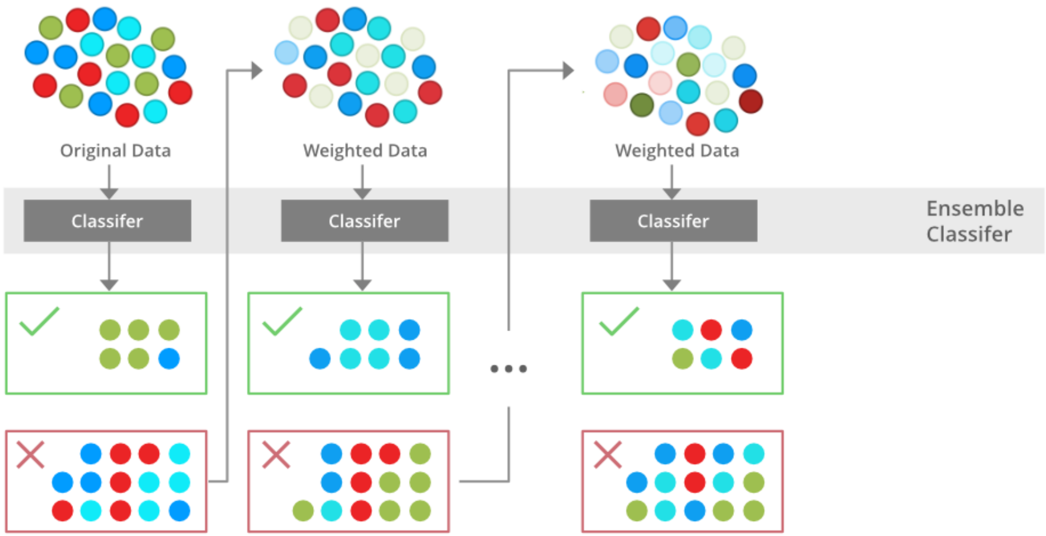
7: Update training sample weights: $w_t = w_{t-1} \circ \exp(-\alpha_t y \circ \hat{y})$

8: Normalize the weights: $w_t = w_t / \left(\sum_{i=1}^{n} w_t^{(i)}\right)$

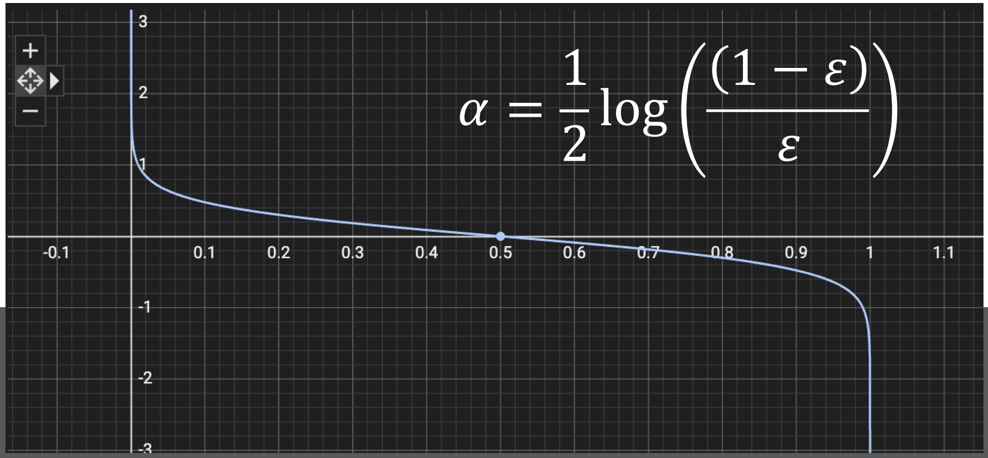
9: **end for**

10: **Return:** $H(x) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$

AdaBoost Algorithm



Source: https://www.geeksforgeeks.org/implementing-the-adaboost-algorithm-from-scratch/



Algorithm 1 AdaBoost Algorithm

1: **Input:** Training dataset $(X, y) \in D$ with n samples and $y \in \{-1, 1\}$, The number of $\mathcal{W}$ models to train T.

2: **Initialize:** Weights $w_0 = [\frac{1}{n}, \frac{1}{n}, \ldots, \frac{1}{n}]^T$ wit…

3: **for** $t = 1$ to T …

4: Train a we… …) using …mples weights w_{t-1}.

5: C… …s weig…d error: $\varepsilon_t = (w_{t-1})^T I(y \neq \hat{y})$

6: C… …weak learner: $\alpha_t = \frac{1}{2}\log\left((1-\varepsilon_t)/\varepsilon_t\right)$

7: Update training sample weights: $w_t = w_{t-1} \circ \exp(-\alpha_t y \circ \hat{y})$

8: Normalize the weights: $w_t = w_t / \left(\sum_{i=1}^{n} w_t^{(i)}\right)$

9: **end for**

10: **Return:** $H(x) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$

$\varepsilon_t \in [0,1]$

α typically $[-1,1]$

$\alpha > 0$ for $\varepsilon_t < 0.5$

$y \circ \hat{y} \in \{1, -1\}$

AdaBoost Algorithm

- When h_t is a weak learner
  - $\varepsilon_t < 0.5 \Rightarrow \alpha > 0$
  - Then, when $y = \hat{y}$, $w_t < w_{t-1}$
  - Otherwise, $w_t \geq w_{t-1}$

Algorithm 1 AdaBoost Algorithm

1: **Input:** Training dataset $(X, y) \in D$ with n samples and $y \in \{-1, 1\}$, The number of $\mathcal{W}$ models to train T.

2: **Initialize:** Weights $w_0 = [\frac{1}{\ }, \frac{1}{\ }, \ldots, \frac{1}{\ }]^T$ wit

3: **for** $t = 1$ to T

4: Train a we ...) using ... mples weights w_{t-1}.

5: C ... s weig ... d error: $\varepsilon_t = (w_{t-1})^T I(y \neq \hat{y})$

6: C ... weak learner: $\alpha_t = \frac{1}{2} \log\left((1 - \varepsilon_t) / \varepsilon_t\right)$

7: Update training sample weights: $w_t = w_{t-1} \circ \exp(-\alpha_t y \circ \hat{y})$

8: Normalize the weights: $w_t = w_t / \left(\sum_{i=1}^{n} w_t^{(i)}\right)$

9: **end for**

10: **Return:** $H(x) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$

$\varepsilon_t \in [0,1]$

α typically $[-1,1]$

$\alpha > 0$ for $\varepsilon_t < 0.5$

$y \circ \hat{y} \in \{1, -1\}$

AdaBoost Algorithm

- When h_t is a weak learner
  - $\varepsilon_t < 0.5 \Rightarrow \alpha > 0$
  - Then, when $y = \hat{y}$, $w_t < w_{t-1}$
  - Otherwise, $w_t \geq w_{t-1}$
- When h_t is not a weak learner; ***we don't trust the model (flip decisions)***
  - $\varepsilon_t \geq 0.5 \Rightarrow \alpha \leq 0$
  - Then, when $y = \hat{y}$, $w_t \geq w_{t-1}$
  - Otherwise, $w_t < w_{t-1}$

Algorithm 1 AdaBoost Algorithm

1: **Input:** Training dataset $(X, y) \in D$ with n samples and $y \in \{-1, 1\}$, The number of $\mathcal{W}$ models to train T.

2: **Initialize:** Weights $w_0 = [\frac{1}{n}, \frac{1}{n}, \ldots, \frac{1}{n}]^T$ wit

3: **for** $t = 1$ to T

4: Train a we ...) using ... mples weights w_{t-1}.

5: C ... s weig ... d error: $\varepsilon_t = (w_{t-1})^T I(y \neq \hat{y})$

6: C ... weak learner: $\alpha_t = \frac{1}{2} \log\left((1 - \varepsilon_t) / \varepsilon_t\right)$

7: Update training sample weights: $w_t = w_{t-1} \circ \exp(-\alpha_t y \circ \hat{y})$

8: Normalize the weights: $w_t = w_t / \left(\sum_{i=1}^{n} w_t^{(i)}\right)$

9: **end for**

10: **Return:** $H(x) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$

$\varepsilon_t \in [0,1]$

α typically $[-1,1]$

$\alpha > 0$ for $\varepsilon_t < 0.5$

$y \circ \hat{y} \in \{1, -1\}$

AdaBoost Algorithm

- When h_t is a weak learner
  - $\varepsilon_t < 0.5 \Rightarrow \alpha > 0$
  - Then, when $y = \hat{y}$, $w_t < w_{t-1}$
  - Otherwise, $w_t \geq w_{t-1}$
- When h_t is not a weak learner; ***we don't trust the model (flip decisions)***
  - $\varepsilon_t \geq 0.5 \Rightarrow \alpha \leq 0$
  - Then, when $y = \hat{y}$, $w_t \geq w_{t-1}$
  - Otherwise, $w_t < w_{t-1}$

Algorithm 1 AdaBoost Algorithm

Input: Training dataset $(X, y) \in D$ with n samples and $y \in \{-1, 1\}$, The number of $\mathcal{W}$ models to train T.

Initialize: Weights $w_0 = [\frac{1}{n}, \frac{1}{n}, \ldots \frac{1}{n}]^T$ wit

for $t = 1$ to T

Train a we ...) using ... mples weights w_{t-1}.

C ... s weig... error: $\varepsilon_t = (w_{t-1})^T I(y \neq \hat{y})$

C ... weak learner: $\alpha_t = \frac{1}{2} \log\left((1 - \varepsilon_t) / \varepsilon_t\right)$

Update training sample weights: $w_t = w_{t-1} \circ \exp(-\alpha_t y \circ \hat{y})$

Normalize the weights: $w_t = w_t / \left(\sum_{i=1}^{n} w_t^{(i)}\right)$

end for

Return: $H(x) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$

$\varepsilon_t \in [0,1]$

α typically $[-1,1]$

$\alpha > 0$ for $\varepsilon_t < 0.5$

$y \circ \hat{y} \in \{1, -1\}$

Same Logic in the Ensemble

AdaBoost Example

- Assume $h_1(x) = c$, where c is a constant.
- If the total weight for the positive examples exceeds that of the negative examples,
  - then, $h_1(x) = +1 \; \forall x$
  - Otherwise, $h_1(x) = -1 \; \forall x$
- Suppose that in our original training set, there are 80 positive and 20 negative samples.
- The original cumulative weight distribution is
  - Positive samples $80 * \frac{1}{100} = 0.8$
  - Negative samples $20 * \frac{1}{100} = 0.2$

AdaBoost Example (Continue)

- Then, $h_1(x) = +1 \ \forall x$
  - $\varepsilon_1 = 0.2, \quad \alpha_1 = \frac{1}{2}\log(4)$
- Weights for each ***correctly*** predicted sample is $\frac{1}{100}\exp\left(-\frac{1}{2}\log 4\right) = \frac{1}{200}$
- Weights for each ***incorrectly*** predicted sample is $\frac{1}{100}\exp\left(\frac{1}{2}\log 4\right) = \frac{1}{50}$

Algorithm 1 AdaBoost Algorithm

Input: Training dataset $(X, y) \in D$ with n samples and $y \in \{-1, 1\}$, The number of $\mathcal{W}$ models to train T.
Initialize: Weights $w_0 = \left[\frac{1}{n}, \frac{1}{n}, ..., \frac{1}{n}\right]^T$ with same shape as y.
for $t = 1$ to T **do**
  Train a weak learner $\hat{y} = h_t(X, w_{t-1})$ using samples weights w_{t-1}.
  Compute the weak learner's weighted error: $\varepsilon_t = (w_{t-1})^T I(y \neq \hat{y})$
  Compute the weight of the weak learner: $\alpha_t = \frac{1}{2}\log\left((1 - \varepsilon_t)/\varepsilon_t\right)$
  Update training sample weights: $w_t = w_{t-1} \circ \exp(-\alpha_t y \circ \hat{y})$
  Normalize the weights: $w_t = w_t / \left(\sum_{i=1}^{n} w_t^{(i)}\right)$
end for
Return: $H(x) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$

AdaBoost Example (Continue)

- The cumulative weight sum is $80 * \frac{1}{200} + 20 * \frac{1}{50} = \frac{4}{5}$
- Weights after normalization
  - Positive samples: $\frac{1}{160}$
  - Negative samples: $\frac{1}{40}$
- The new cumulative weight distribution is
  - Positive samples $80 * \frac{1}{160} = \frac{1}{2}$
  - Negative samples $20 * \frac{1}{40} = \frac{1}{2}$

Algorithm 1 AdaBoost Algorithm

1: **Input:** Training dataset $(X, y) \in D$ with n samples and $y \in \{-1, 1\}$, The number of $\mathcal{W}$ models to train T.
2: **Initialize:** Weights $w_0 = \left[\frac{1}{n}, \frac{1}{n}, ..., \frac{1}{n}\right]^T$ with same shape as y.
3: **for** $t = 1$ to T **do**
4: Train a weak learner $\hat{y} = h_t(X, w_{t-1})$ using samples weights w_{t-1}.
5: Compute the weak learner's weighted error: $\varepsilon_t = (w_{t-1})^T I(y \neq \hat{y})$
6: Compute the weight of the weak learner: $\alpha_t = \frac{1}{2} \log\left((1 - \varepsilon_t)/\varepsilon_t\right)$
7: Update training sample weights: $w_t = w_{t-1} \circ \exp(-\alpha_t y \circ \hat{y})$
8: Normalize the weights: $w_t = w_t / \left(\sum_{i=1}^{n} w_t^{(i)}\right)$
9: **end for**
10: **Return:** $H(x) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$

AdaBoost Example (Continue)

- Data is now evenly weighted.
- The next weak learner $h_2(x)$ needs to do something more interesting than a constant output to achieve an error rate below 0.5.

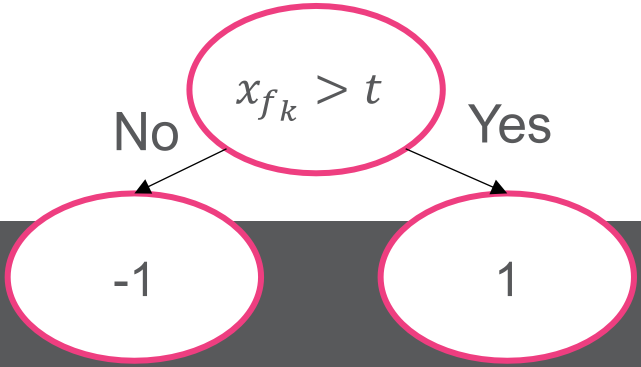
Algorithm 1 AdaBoost Algorithm

Input: Training dataset $(X, y) \in D$ with n samples and $y \in \{-1, 1\}$, The number of $\mathcal{W}$ models to train T.
Initialize: Weights $w_0 = \left[\frac{1}{n}, \frac{1}{n}, ..., \frac{1}{n}\right]^T$ with same shape as y.
for $t = 1$ to T **do**
    Train a weak learner $\hat{y} = h_t(X, w_{t-1})$ using samples weights w_{t-1}.
    Compute the weak learner's weighted error: $\varepsilon_t = (w_{t-1})^T \, I(y \neq \hat{y})$
    Compute the weight of the weak learner: $\alpha_t = \frac{1}{2} \log\left((1 - \varepsilon_t) / \varepsilon_t\right)$
    Update training sample weights: $w_t = w_{t-1} \circ \exp(-\alpha_t y \circ \hat{y})$
    Normalize the weights: $w_t = w_t / \left(\sum_{i=1}^{n} w_t^{(i)}\right)$
end for
Return: $H(x) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$

Decision Stumps

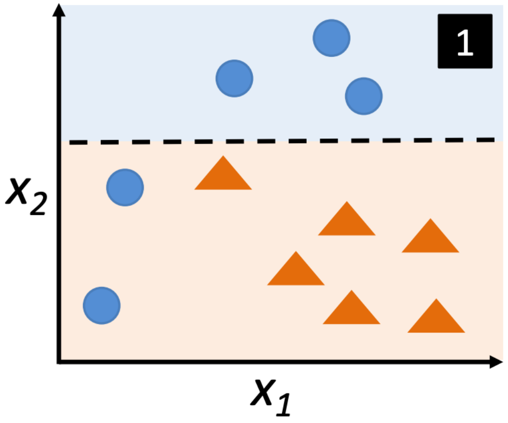
- Shallow decision trees are a popular weak learner
- A very popular weak learner is a ***decision stump***
  - This is a decision tree that can only ask one question
- A decision stump is pretty useless on its own, but very effective when combined with boosting

$$h(x) = s\left(I\left(x_{f_k} > t\right)\right), \text{where } s \in \{-1,1\}, x_{f_k} \text{ is feature } k$$

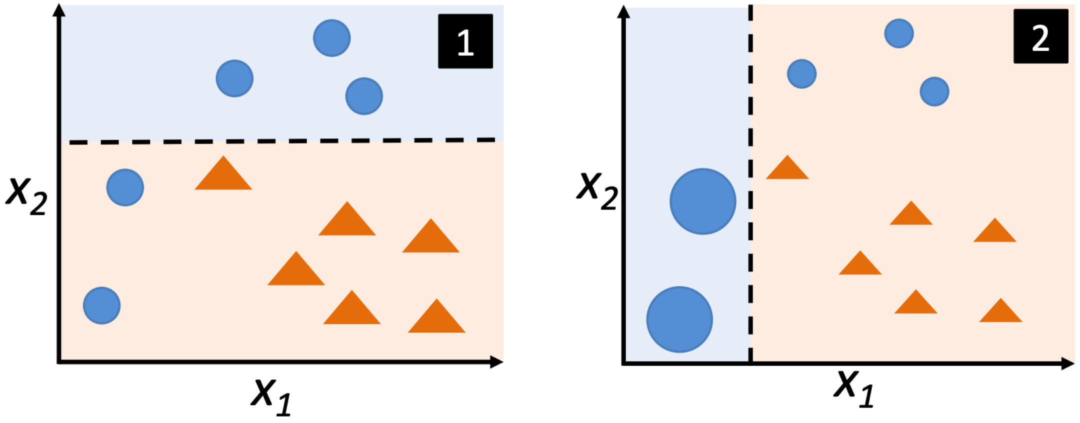


AdaBoost with Decision Stumps

Uniform weight
across samples



AdaBoost with Decision Stumps



Weight the Entropy Impurity scores.

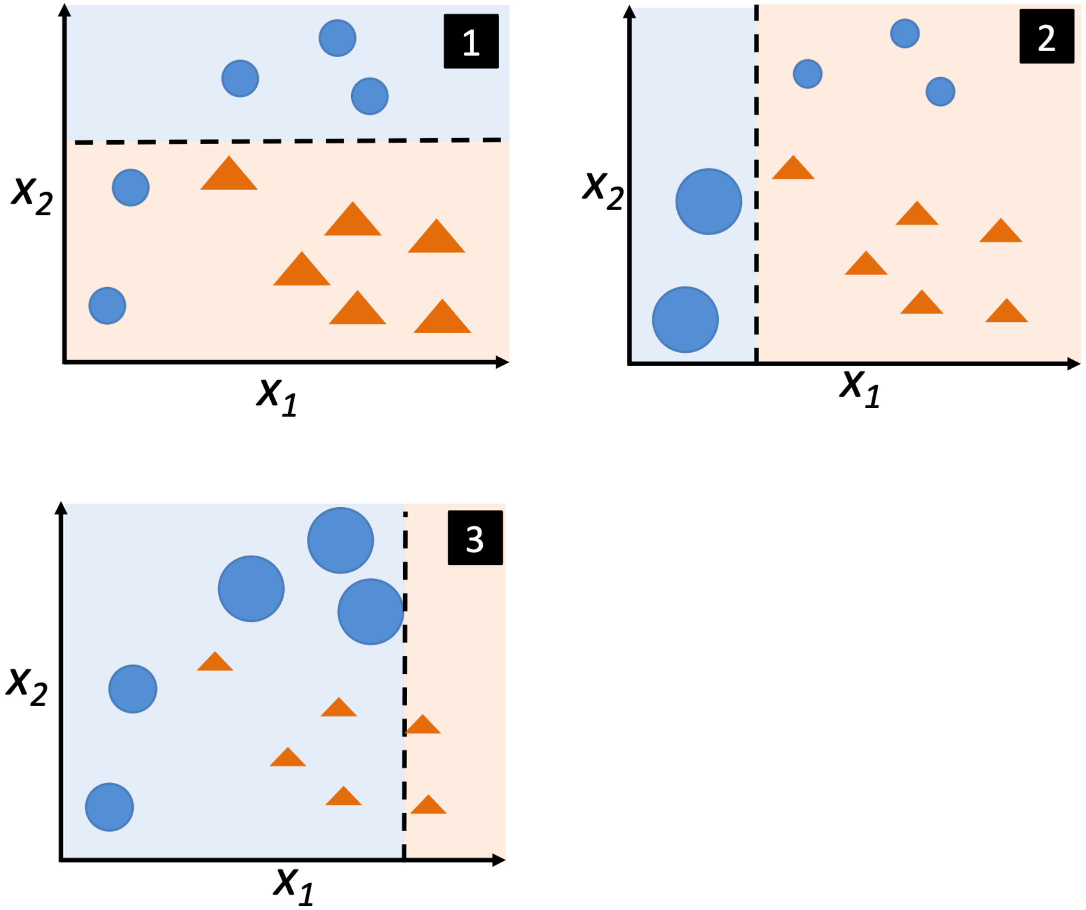
$$I_H(D,t) = -\sum_{i=1}^{c} p(D=i|t) \log_2\big(p(D=i|t)\big)$$

Weighted classification error

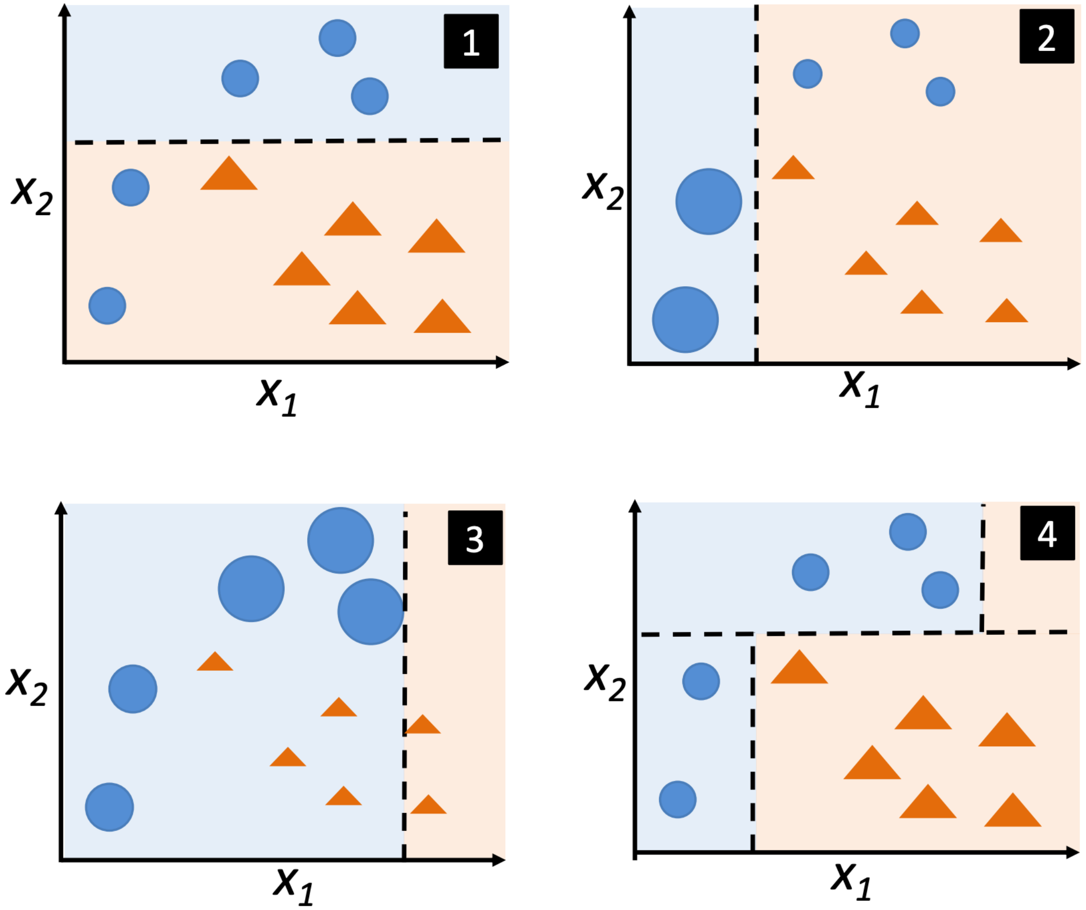
$$\varepsilon_t = \left(\sum_{i\in D} w^{(i)}\, I\big(y^{(i)} \neq \hat{y}^{(i)}\big)\right) / \sum_{i\in D} w^{(i)}$$

 Slide Credit: Dr. Raschka

AdaBoost with Decision Stumps



AdaBoost with Decision Stumps



 Slide Credit: Dr. Raschka

Random Forests

THE UNIVERSITY OF TENNESSEE KNOXVILLE

Random Ensembles

- If we wanted to use an ensemble of decision trees, it might be expensive to train each tree individually.
- It's super fast for decision stumps but prohibitively expensive for deep trees.
- The expensive part is picking the tree structure.
- If we had a tree structure already, it would be cheap to fill in the leaf nodes (the predictions of the tree) using the training data.

https://ttpoll.com/p/237202

Random Forests

- In random forests, we use trees with fixed structures and random features.
- Parallel generation of T ***full binary*** trees.
- Features randomly assigned to internal nodes (typically with replacement)
- Leaves filled by training data D_t

Algorithm 1 Random Forest Algorithm

Input: Training data $(X, y) \in D$, number of trees T, number of features to sample m
Initialize: Set of decision trees $F = \{\}$
for $t = 1$ to T **do**
    Draw a bootstrap sample D_t from the training data D
    Randomly select m features from the total feature set
    Train a decision tree h_t on D_t, using only the selected m features
    Add the trained tree h_t to the forest F
end for
Output: Ensemble of trees $F = \{h_1, h_2, \ldots, h_T\}$

Prediction for new input x:
For classification:

$$\hat{y} = \text{majority\_vote}\left(\{h_t(x) : t = 1, \ldots, T\}\right)$$

For regression:

$$\hat{y} = \frac{1}{T}\sum_{t=1}^{T} h_t(x)$$

THE UNIVERSITY OF TENNESSEE KNOXVILLE

Random Forests

- They work remarkably well
- Best practice to remove irrelevant features
- Why do they work?
  - Some nodes will make random choices.
  - While others will make good choices.
  - It is more likely for most nodes to be weak learners due to training data
  - Performance increases with the number of trees.

Pop Quiz

True or False. Random Forests fill a pre-determined tree structure's internal nodes with randomly selected features.

A. True

B. False

Review

- Ensemble of techniques
- Ensemble of datasets
  - Different datasets -> Bagging
- They Reduce variance
- Perform well as long as only a few of the models make the same mistakes
- Boosting
  - Ensemble of weak learners
  - Easier to design
  - Computational efficient
- Random Forests

Next Lecture

- Model Evaluation

Helper Slides

https://ttpoll.com/p/237202