UNSW
SYDNEY

Ensemble Learning

COMP3411:
Artificial Intelligence

- ✓Different Subfields of AI Algorithms.
- ✓Decision Trees.
- ✓**Ensemble Learning:**
  - **Bagged Trees.**
  - Random Forests.
  - Boosting Trees.

Ensemble Learning, Bagged Trees:

- **Bagging** (short for *bootstrap aggregation*) was originally proposed by Leo Breiman.
- Bagging is a general ensemble method that can be applied to any regression or classification model and relies on bootstrapping to create multiple training datasets.
  - A bootstrap sample is a random sample of the data taken with replacement. Once a data point is selected for the sample, it remains available for subsequent selections. Typically, a bootstrap sample is **the same size** as the original dataset.
  - As a result, some data points may appear multiple times in a sample, while others may not be selected at all.
  - Data points that are not included in a bootstrap sample are commonly referred to as **out-of-bag (OOB)** samples.

Ensemble Learning, Bagged Trees:

- **Bagging** (short for *bootstrap aggregation*) was originally proposed by Leo Breiman.
- Bagging is a general ensemble method that can be applied to any regression or classification model and relies on bootstrapping to create multiple training datasets.

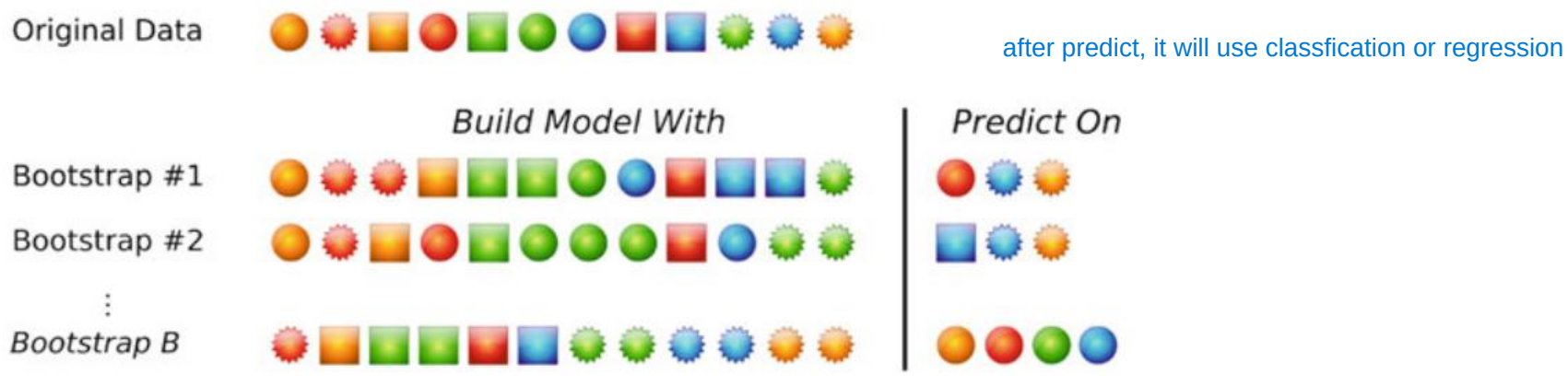


Fig. 3: A schematic of bootstrap resampling. Each subset is the same size as the original and can contain multiple instances of the same data point. Samples not selected by the bootstrap are predicted and used to estimate model performance.

Ensemble Learning, Bagged Trees:

- **Bagging** (short for *bootstrap aggregation*) was originally proposed by Leo Breiman.
- Bagging is a general ensemble method that can be applied to any regression or classification model and relies on bootstrapping to create multiple training datasets.
- The method is fairly simple in structure and consists of the steps in Algorithm 1.

```
1 for i = 1 to m do
2 |   Generate a bootstrap sample of the original data
3 |   Train an unpruned tree model on this sample
4 end
```

Algorithm 1: Bagging algorithm with *m* decision trees in ensemble.

Ensemble Learning, Bagged Trees:

- **Bagging** (short for *bootstrap aggregation*) was originally proposed by Leo Breiman.
- Bagging is a general ensemble method that can be applied to any regression or classification model and relies on bootstrapping to create multiple training datasets.

- The method is fairly simple in structure and consists of the steps in Algorithm 1.

- Each model in the ensemble generates a prediction for a new sample, and the final prediction is made using **the majority vote rule** (for classification) or by averaging the predictions (for regression).

Ensemble Learning, Bagged Trees:

Advantages of Bagging:

- Bagging effectively **reduces the variance** of predictions through its aggregation process.
- When predictions for a sample are averaged (e.g., across trees A to F), the **average prediction has lower variance** than the variance of the individual predictions.
- Individual trees may overfit or be strongly influenced by specific samples in their bootstrap sets, but their errors typically vary in different directions.
- By aggregating many such predictions, these errors **partially cancel out**, resulting in a more stable and less variable overall prediction.

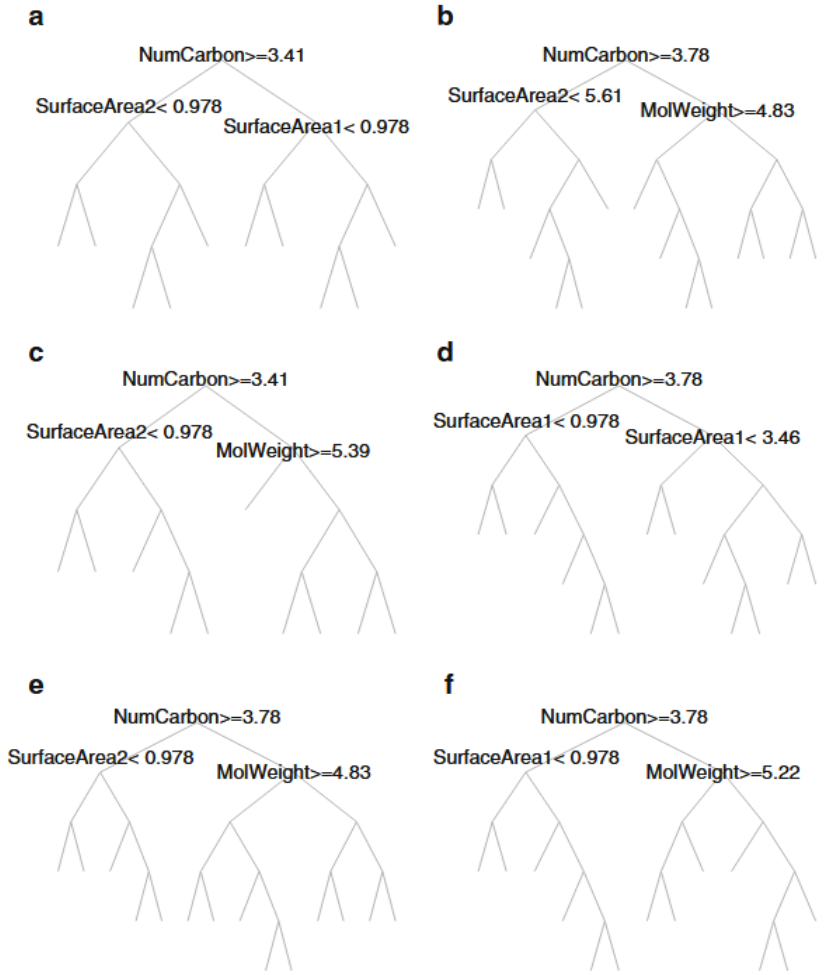


Fig. 4: Example of trees from bagging. Trees vary in structure, and hence the predictions will vary from tree to tree.

Ensemble Learning, Bagged Trees:

Advantages of Bagging (cont.):

- **No separate test set required:** Bagging allows us to estimate model performance without needing a separate dataset, saving both time and data, as all training data can still be used for building the model.
- Each tree is trained on a **bootstrap sample**, which typically contains about 63% of the data.
- The remaining ~37% of the data, known as **out-of-bag (OOB) samples**, act as “unseen test data.”
- Testing each tree on its OOB samples provides an **unbiased estimate of performance**.

Ensemble Learning, Bagged Trees:

Example of Bagging:

- Training data: N observations
- Set of features= (Credit, Term, Income)
- $\text{Error}_{\text{total}} = \frac{\sum E_i}{N}$, where E_i=1 if the prediction is incorrect, 0 otherwise.

Credit	Term	Income	y
excellent	3 yrs	high	safe
fair	5 yrs	low	risky
fair	3 yrs	high	safe
poor	5 yrs	high	risky
excellent	3 yrs	low	risky
fair	5 yrs	low	safe
poor	3 yrs	high	risky
poor	5 yrs	low	safe
fair	3 yrs	high	safe

Ensemble Learning, Bagged Trees:

Example of Bagging (Cont.): Step 1: Bootstrapping with m=3 (number of trees).

Credit	Term	Income	y
excellent	3 yrs	high	safe
fair	5 yrs	low	risky
fair	3 yrs	high	safe
poor	5 yrs	high	risky
excellent	3 yrs	low	risky
fair	5 yrs	low	safe
excellent	3 yrs	low	risky
fair	5 yrs	low	safe
fair	5 yrs	low	risky

B1

Credit	Term	Income	y
poor	5 yrs	high	risky
Poor	3 yrs	high	risky
poor	5 yrs	low	safe
Poor	5 yrs	high	risky
excellent	3 yrs	low	risky
fair	5 yrs	low	safe
poor	3 yrs	high	risky
poor	5 yrs	low	safe
fair	3 yrs	high	safe

B2

Credit	Term	Income	y
excellent	3 yrs	high	safe
fair	5 yrs	low	risky
fair	3 yrs	high	safe
poor	5 yrs	low	safe
fair	3 yrs	high	safe
fair	5 yrs	low	safe
poor	3 yrs	high	risky
poor	5 yrs	low	safe
fair	3 yrs	high	safe

B3

Credit	Term	Income	y
poor	3 yrs	high	risky
poor	5 yrs	low	safe
fair	3 yrs	high	safe

OOB1

not used in the sample(never picked)

Credit	Term	Income	y
excellent	3 yrs	high	Safe
fair	5 yrs	low	risky
fair	3 yrs	high	safe

OOB2

Credit	Term	Income	y
excellent	3 yrs	low	risky
poor	5 yrs	high	risky
fair	5 yrs	low	safe

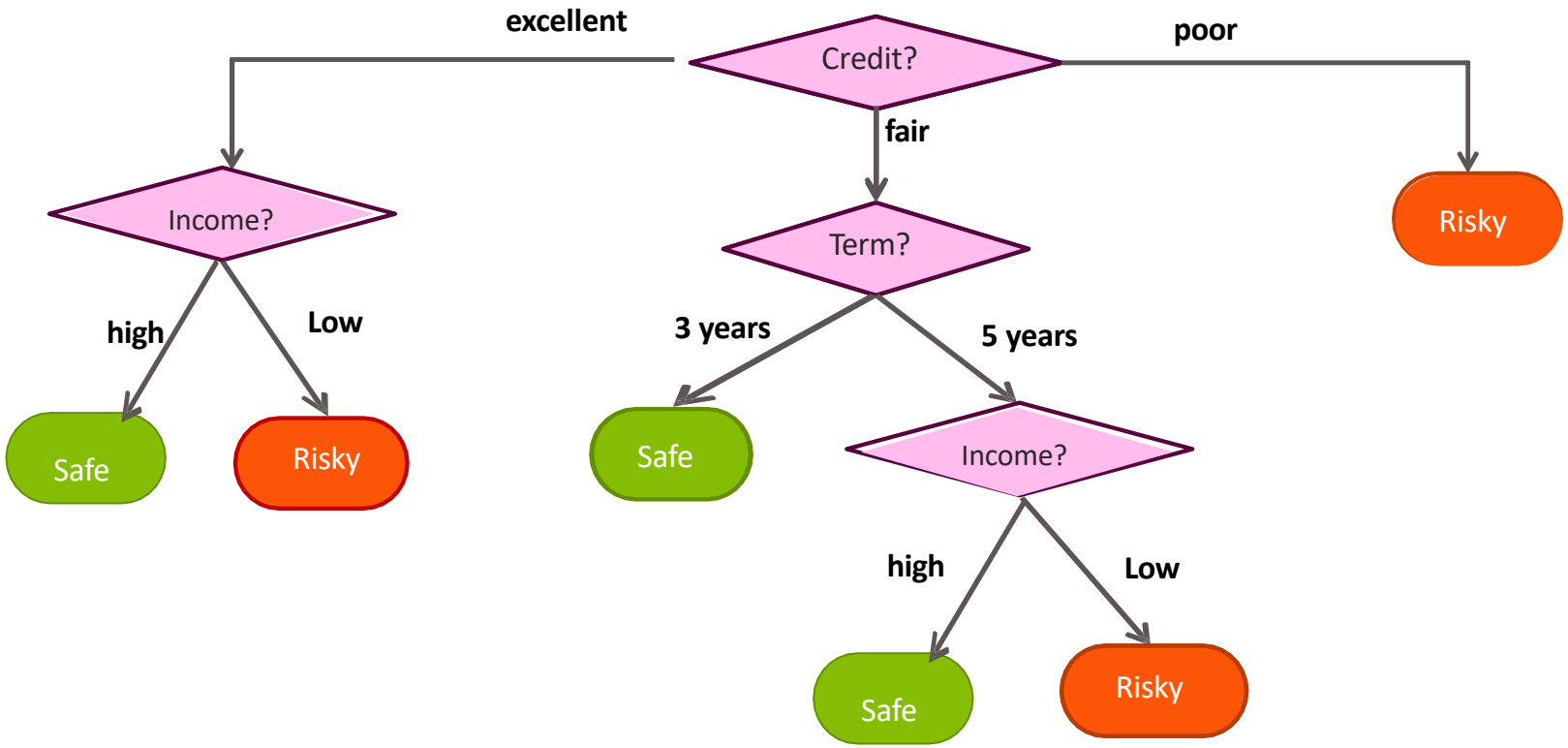
OOB3

Ensemble Learning, Bagged Trees:

Example of Bagging (Cont.): Step 2: Parallel model training.

Credit	Term	Income	y
excellent	3 yrs	high	safe
fair	5 yrs	low	risky
fair	3 yrs	high	safe
poor	5 yrs	high	risky
excellent	3 yrs	low	risky
fair	5 yrs	low	safe
excellent	3 yrs	low	risky
fair	5 yrs	low	safe
Fair	5 yrs	low	risky

B1



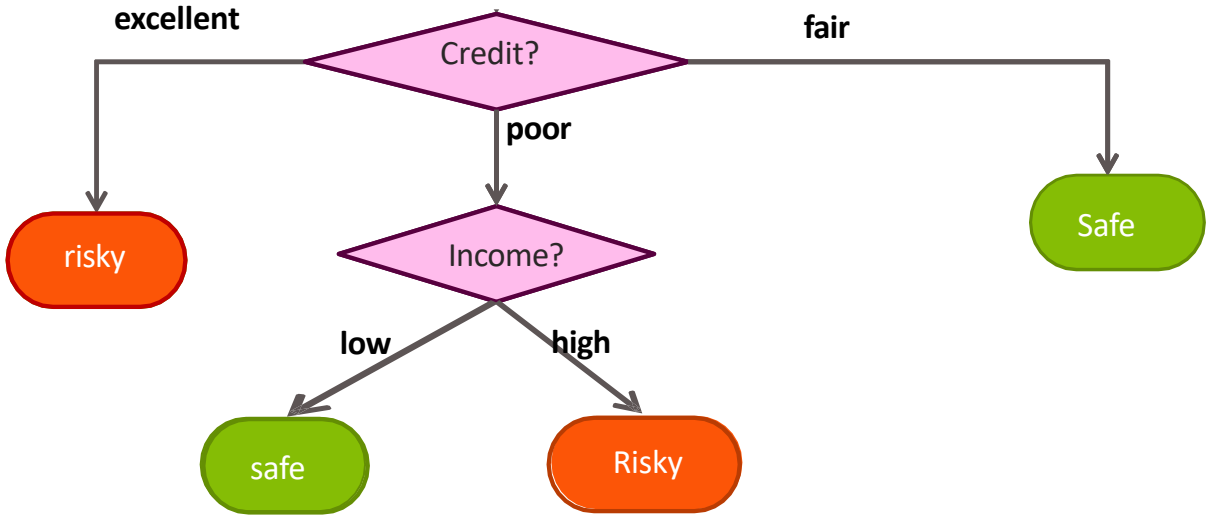
Ensemble Learning, Bagged Trees:

Example of Bagging (Cont.):

Step 2: Parallel model training.

Credit	Term	Income	y
poor	5 yrs	high	risky
Poor	3 yrs	high	risky
poor	5 yrs	low	safe
poor	5 yrs	high	risky
excellent	3 yrs	low	risky
fair	5 yrs	low	safe
poor	3 yrs	high	risky
poor	5 yrs	low	safe
fair	3 yrs	high	safe

B2

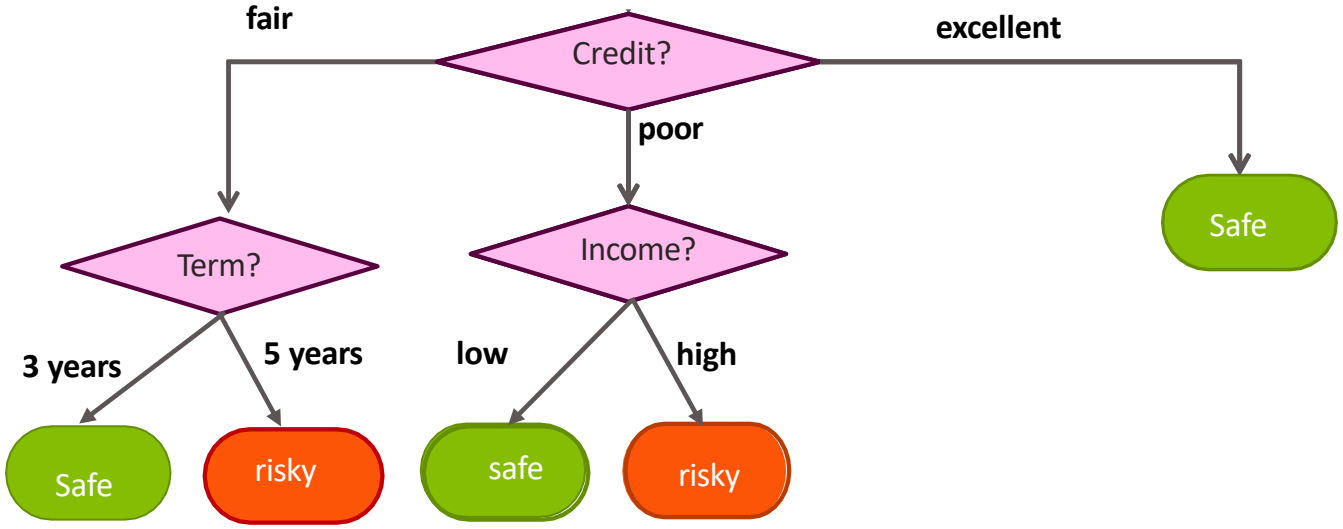


Ensemble Learning, Bagged Trees:

Example of Bagging (Cont.): Step 2: Parallel model training.

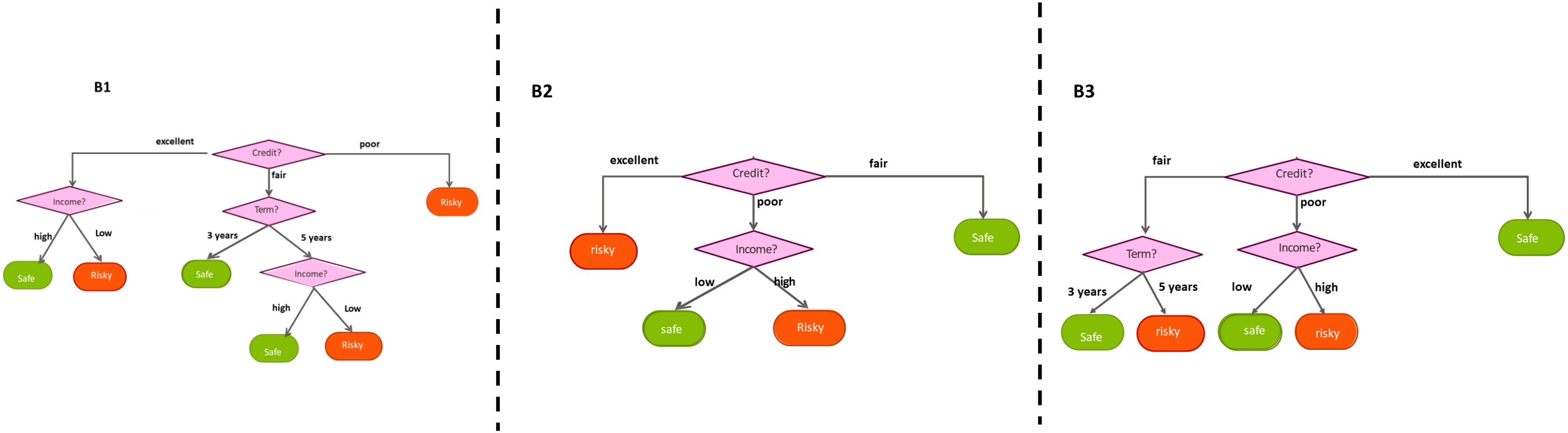
Credit	Term	Income	y
excellent	3 yrs	high	safe
fair	5 yrs	low	risky
fair	3 yrs	high	safe
poor	5 yrs	low	safe
fair	3 yrs	high	safe
fair	5 yrs	low	risky
poor	3 yrs	high	risky
poor	5 yrs	low	safe
fair	3 yrs	high	safe

B3



Ensemble Learning, Bagged Trees:

Example of Bagging (Cont.): Step 2: Parallel model training, Summary



Ensemble Learning, Bagged Trees:

Example of Bagging (Cont.): Step 3: Aggregation of Predictions

Credit	Term	Income	y	B1_prediction	B2_prediction	B3_prediction	Total
excellent	3 yrs	high	safe	safe	risky	safe	**safe**
fair	5 yrs	low	risky	risky	safe	risky	**risky**
fair	3 yrs	high	safe	safe	safe	safe	**safe**
poor	5 yrs	high	risky	risky	risky	risky	**risky**
excellent	3 yrs	low	risky	risky	risky	safe	**risky**
fair	5 yrs	low	safe	risky	safe	risky	**risky**
poor	3 yrs	high	risky	risky	risky	risky	**risky**
poor	5 yrs	low	safe	risky	safe	safe	**safe**
fair	3 yrs	high	safe	safe	safe	safe	**safe**

Ensemble Learning, Bagged Trees:

Example of Bagging (Cont.):

Calculate the error of **bagged** tree on the given dataset:

$E_{total} = (E_1 + E_2 + E_3 + E_4 + E_5 + E_6 + E_7 + E_8 + E_9)/N$

$$E_{total} = \frac{0 + 0 + 0 + 0 + 0 + 1 + 0 + 0 + 0}{9}$$

$E_{total} = 1/9 = 0.111$

Conclusion:

$$E_{total}^{bagged\ tree} = 0.111$$

Credit	Term	Income	y	B1_prediction	B2_prediction	B3_prediction	Total
excellent	3 yrs	high	safe	safe	Risky	safe	**safe**
fair	5 yrs	low	risky	risky	Safe	risky	**risky**
fair	3 yrs	high	safe	safe	Safe	safe	**Safe**
poor	5 yrs	high	risky	risky	Risky	safe	**risky**
excellent	3 yrs	low	risky	risky	Risky	safe	**risky**
fair	5 yrs	low	safe	risky	Safe	risky	**risky**
poor	3 yrs	high	risky	risky	risky	risky	**risky**
poor	5 yrs	low	safe	risky	safe	safe	**safe**
fair	3 yrs	high	safe	safe	Safe	safe	**safe**

Ensemble Learning, Bagged Trees:

- In bagging, each tree is ultimately unique—no two trees are exactly the same.
- However, the trees are still somewhat **correlated** with each other.
- As a result, the variance reduction achieved by bagging can be improved further.
- From a statistical perspective, reducing correlation among predictors can be accomplished by introducing **additional randomness** into the tree construction process.

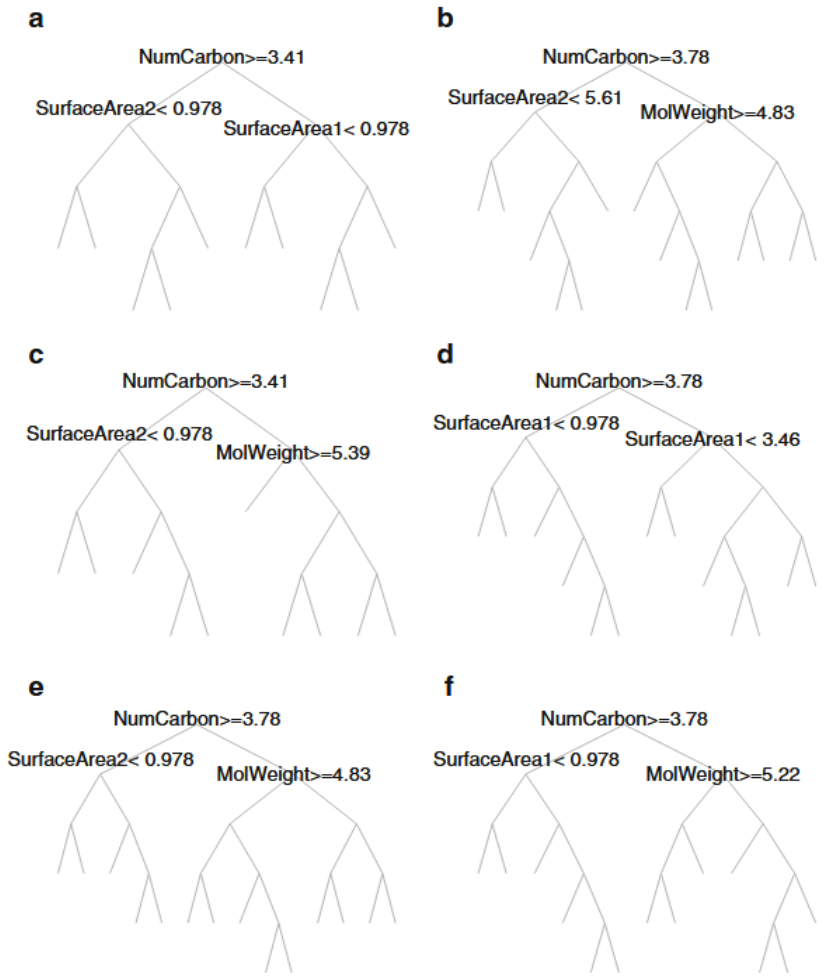


Fig. 4: Example of trees from bagging. Trees vary in structure, and hence the predictions will vary from tree to tree.

- ✓Different Subfields of AI Algorithms.
- ✓Decision Trees.
- ✓**Ensemble Learning:**
  - Bagged Trees.
  - **Random Forests.**
  - Boosting Trees.

Ensemble Learning, Random Forests:

- In 2000, Dietterich introduced the idea of random split selection, where trees are built using a random subset of the top *k* predictors (features) at each split.
- Around the same time, Breiman (2000) proposed adding noise to the response to perturb the tree structure, calling this approach Random Forests.

```
1  Select the number of models to build, m
2  for i = 1 to m do
3      Generate a bootstrap sample of the original data
4      Train a tree model on this sample
5      for each split do
6          Randomly select k (< P) of the original predictors
7          Select the best predictor among the k predictors and
           partition the data
8      end
9      Use typical tree model stopping criteria to determine when a
       tree is complete (but do not prune)
10 end
```

Algorithm 2: Basic Random Forests with *m* decision trees in ensemble with total *P* predictors.

Ensemble Learning, Random Forests:

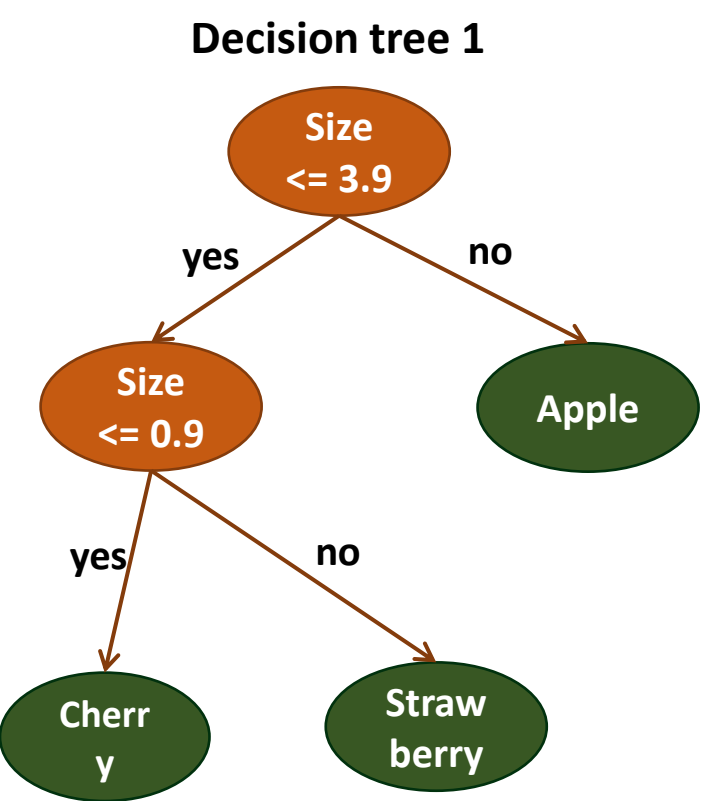
- Each model in the ensemble is then used to generate a prediction for a new sample, and the final prediction is made using **the majority vote rule** (for classification) or by averaging the predictions (for regression).
- At each split in a tree, the algorithm randomly selects **k predictors (features)**. Typically, k = $\sqrt{p}$, where p is the total number of features.
- The tree then chooses the best split only from those k features.
- This randomness reduces correlation between trees (the problem with bagged trees).

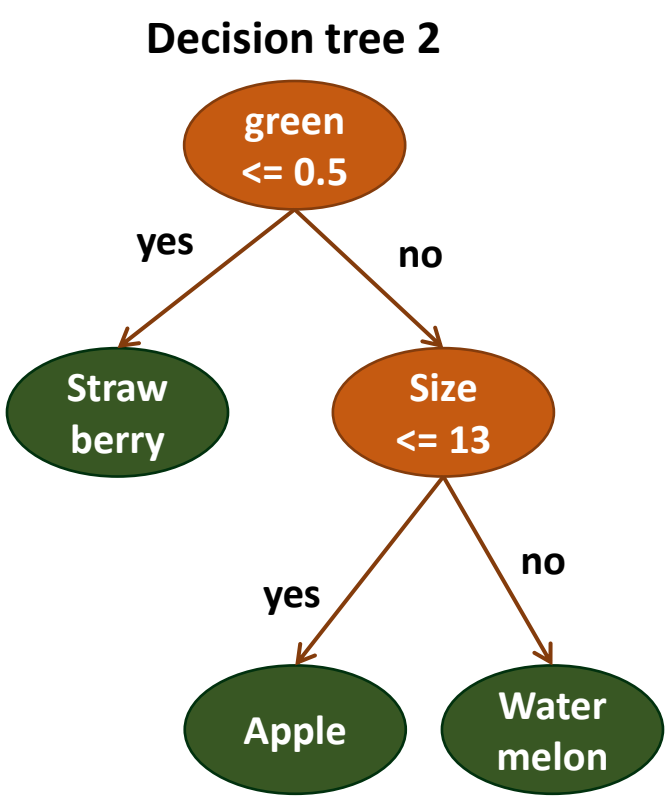
```
1  Select the number of models to build, m
2  for i = 1 to m do
3      Generate a bootstrap sample of the original data
4      Train a tree model on this sample
5      for each split do
6          Randomly select k (< P) of the original predictors
7          Select the best predictor among the k predictors and
           partition the data
8      end
9      Use typical tree model stopping criteria to determine when a
       tree is complete (but do not prune)
10 end
```

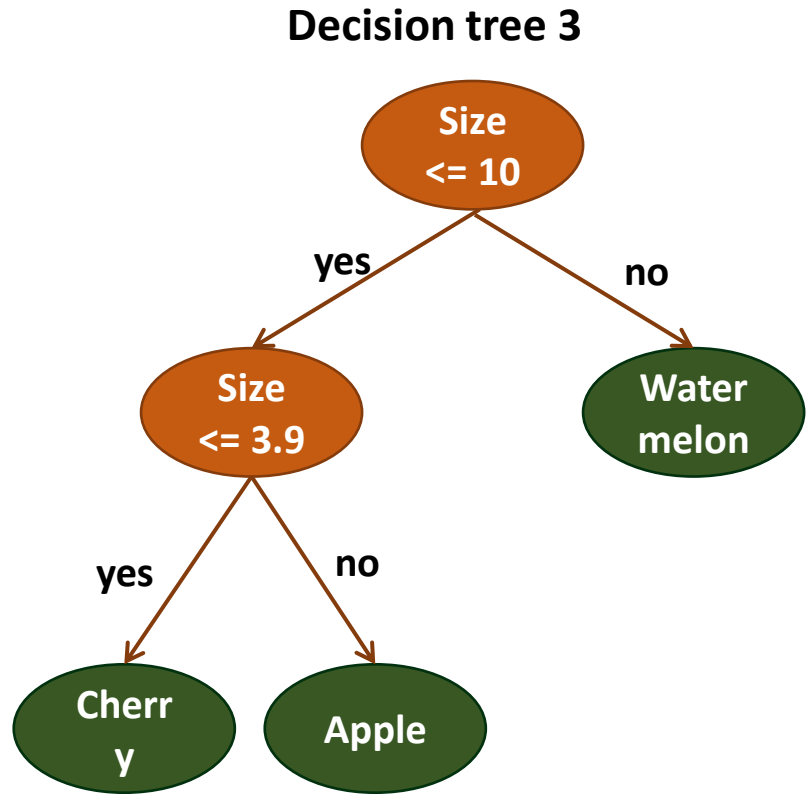
Algorithm 2: Basic Random Forests with *m* decision trees in ensemble with total *P* predictors.

Ensemble Learning, Random Forests:

- Compared to bagging, random forests is more computationally efficient on a tree-by-tree basis since the tree building process only needs to evaluate **a fraction of the original features** at each split, although **more trees** are usually required by random forests.







Ensemble Learning, Random Forests:

Choosing the number of trees (*m*) in Random Forests and Bagging:

- **Variance reduction:** Adding more trees decreases variance, making predictions more stable.
- **Constraints:** Training time, memory usage, and overfitting are the main limiting factors when increasing m.
- **Test error behavior:** The test error typically decreases monotonically as *m* increases—rapidly at first and then levelling off, becoming nearly constant after a sufficient number of trees.

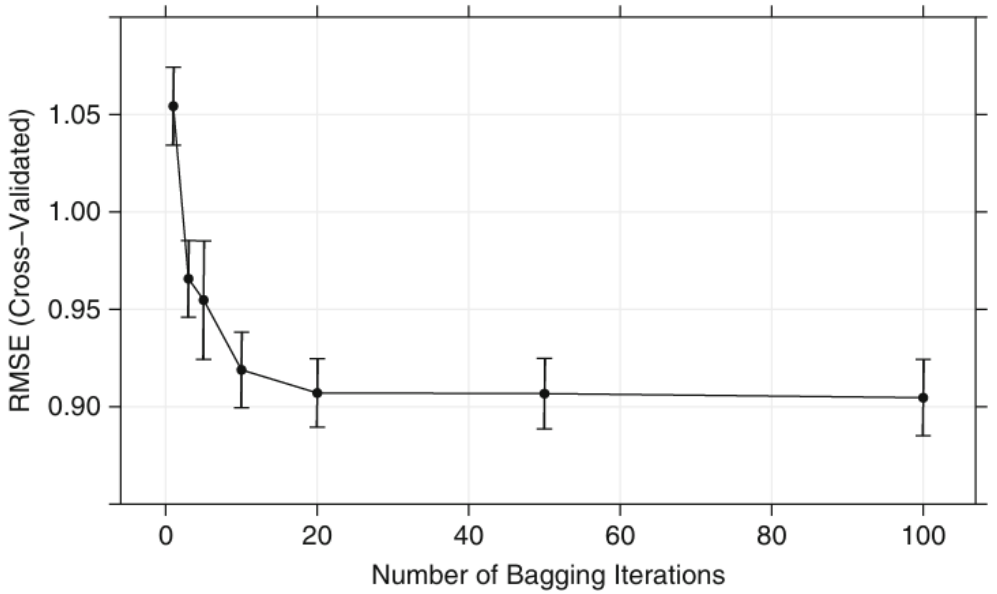


Fig. 5: Root mean square error (RMSE) as a function of number of iterations for the solubility data. Vertical lines indicate ± one-standard error of RMSE.

- ✓Different Subfields of AI Algorithms.
- ✓Decision Trees.
- ✓**Ensemble Learning:**
  - Bagged Trees.
  - Random Forests.
  - **Boosting Trees.**

Ensemble Learning, Boosting:

There are many types of boosting algorithms, but here we focus on one of the most influential: **AdaBoost**.

- At each iteration, AdaBoost selects the best classifier based on the current weighted data points.
- Data points that are misclassified in the k-th iteration receive higher weights in the (k+1)-st iteration.
- As a result, samples that are difficult to classify gradually receive increasingly larger weights.
- This process ensures that each iteration focuses on learning a different aspect of the data.
- Finally, the sequence of weighted classifiers is combined into an ensemble, producing a strong overall model.

Ensemble Learning, Boosting:

1. Initialize the observation weights $w_i=1/N$, i=1,2,…,N.
2. For m=1 to M:
   (a) Fit a classifier $G_m(x)$ to the training data using weights w_i.
   (b) Compute

$$err_m = \frac{\sum_1^N w_i \, I(y_i \,!= G_m(x_i))}{\sum_1^N w_i}$$

   (c) Compute the influence α_m

$$\alpha_m = \tfrac{1}{2} \ln \frac{1-err_m}{err_m}$$

   (d) Set $\boldsymbol{w_i^{new\_iteration} = w_i^{old\_iteration} \, e^{\pm \alpha}}$, i=1,2,…,N.
3. Output $G(x)=sign[\sum_{m=1}^{M} \alpha_m G_m(x)]$.

Algorithm 3: AdaBoost algorithm with *M* decision trees in ensemble for classification problems.

Ensemble Learning, Boosting:

- Boosting builds an ensemble of learners **sequentially**:
  - The first predictions contain some bias (errors).
  - Boosting focuses on the mistakes by assigning greater weight to misclassified predicted samples.
  - The next model is trained to correct those errors.
  - Each new model becoming “specialized” in fixing the weaknesses of the previous ones.
- By building trees **sequentially**—where each new tree corrects the errors of the previous ones—boosting **reduces bias** and improves predictive performance.

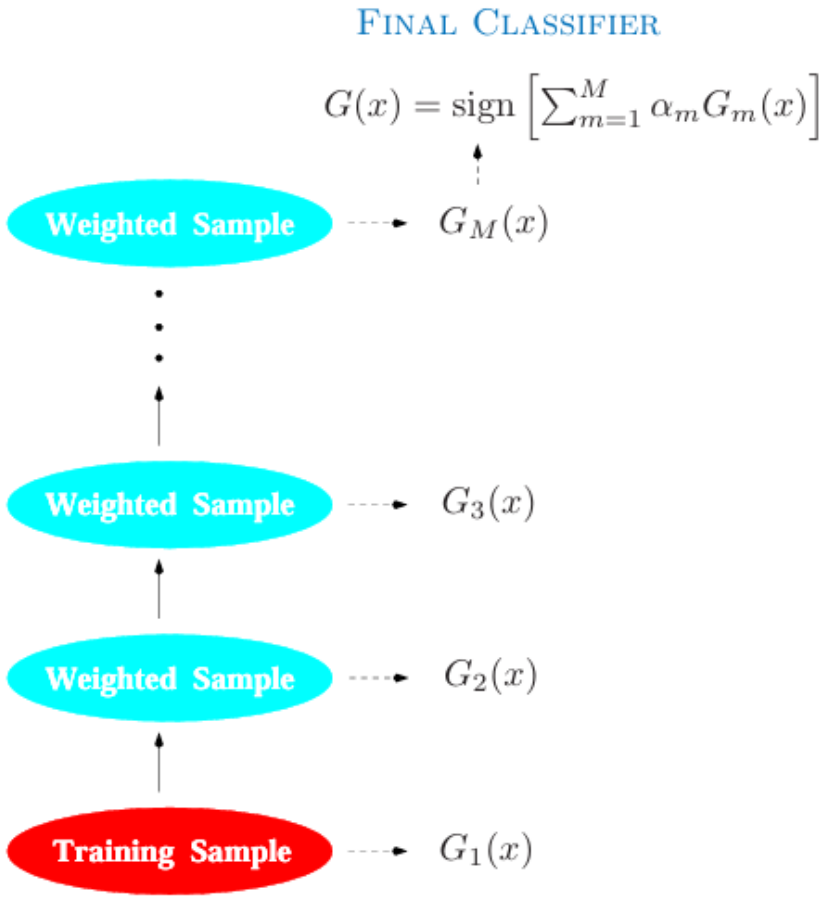


Fig. 6: Schematic of AdaBoost. Classifiers are sequentially trained on weighted versions of the dataset and then combined to produce a final prediction.

Ensemble Learning, Boosting:

Example of Boosting:

- Decide whether or not a person should go rock climbing based on their age, whether the person likes goat,s and height.

- Training data: N observations= 10
- M= 2 (two iterations will be explored)

- $\text{Error}_{\text{total}} = \frac{\sum E_i}{N}$, where E_i=1 if the prediction is incorrect, 0 otherwise.

#	Age	Likes height?	Likes goats?	Go climbing?
1	23	0	0	-1
2	31	1	1	1
3	35	1	0	1
4	35	0	0	-1
5	42	0	0	-1
6	43	1	1	1
7	45	0	1	-1
8	46	1	1	1
9	46	1	0	-1
10	51	1	0	-1

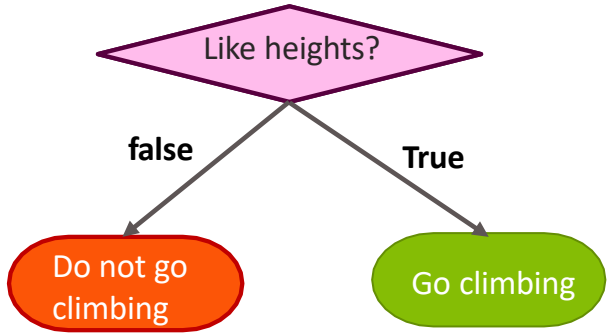
Ensemble Learning, Boosting:

Example of Boosting (cont.): Step 0: Assigning weights

#	Age	Likes height?	Likes goats?	Go climbing?	Weight
1	23	0	0	-1	0.1
2	31	1	1	1	0.1
3	35	1	0	1	0.1
4	35	0	0	-1	0.1
5	42	0	0	-1	0.1
6	43	1	1	1	0.1
7	45	0	1	-1	0.1
8	46	1	1	1	0.1
9	46	1	0	-1	0.1
10	51	1	0	-1	0.1

Ensemble Learning, Boosting:

Example of Boosting (cont.): Iteration 1, Step 1: Fit a classifier to the training data ($G_1(x)$).



#	Age	Likes height?	Likes goats?	Go climbing?
1	23	0	0	-1
2	31	1	1	1
3	35	1	0	1
4	35	0	0	-1
5	42	0	0	-1
6	43	1	1	1
7	45	0	1	-1
8	46	1	1	1
9	46	1	0	-1
10	51	1	0	-1

Ensemble Learning, Boosting:

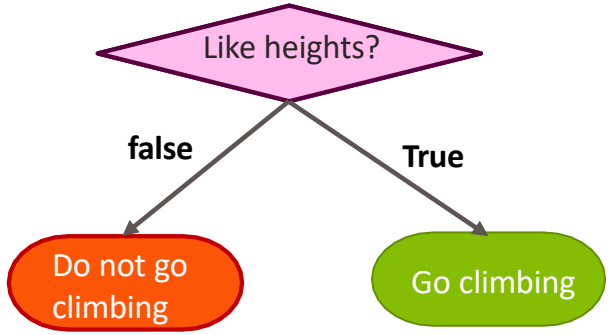
Example of Boosting (cont.): Iteration 1 (m=1), Step 2: Calculating err_m and α_m (influence)

$err_m = err_1 = 2/10$

$$\alpha_1 = \frac{1}{2} \ln \frac{1 - 2/10}{2/10}$$

$$\alpha_1 = \frac{1}{2} \ln \frac{8/10}{2/10}$$

$$\alpha_1 = \frac{1}{2} \ln 4 = 0.69$$



#	Age	Likes height?	Likes goats?	Go climbing?
1	23	0	0	-1
2	31	1	1	1
3	35	1	0	1
4	35	0	0	-1
5	42	0	0	-1
6	43	1	1	1
7	45	0	1	-1
8	46	1	1	1
9	46	1	0	-1
10	51	1	0	-1

Ensemble Learning, Boosting:

Example of Boosting (cont.): Iteration 1, Step 3: Calculating $w_i^{new\_iteration}$

$$\boldsymbol{w_i^{new\_iteration} = w_i^{old\_iteration}\, e^{\pm\alpha}}$$

For correctly classified samples (i ∈ $[1, 8]$) , we get:

$$w_i^{new\_iteration} = 0.1\, e^{-0.69} = 0.050$$

and for wrongly classified samples:

$$w_{10}^{new\_iteration} = w_9^{new\_iteration} = 0.1\, e^{+0.69} = 0.199$$

These weights still need to be **normalized**, so that their sum equals 1.

#	Age	Likes height?	Likes goats?	Go climbing?
1	23	0	0	-1
2	31	1	1	1
3	35	1	0	1
4	35	0	0	-1
5	42	0	0	-1
6	43	1	1	1
7	45	0	1	-1
8	46	1	1	1
9	46	1	0	-1
10	51	1	0	-1

Ensemble Learning, Boosting:

Example of Boosting (cont.): Iteration 1, Step 3: Calculating $w_i^{new\_iteration}$

$$w_i^{new\_iteration} = w_i^{old\_iteration}\, e^{\pm\alpha}$$

#	Age	Likes height?	Likes goats?	Go climbing?	Weight	Normalized Weight
1	23	0	0	-1	0.1	0.062
2	31	1	1	1	0.1	0.062
3	35	1	0	1	0.1	0.062
4	35	0	0	-1	0.1	0.062
5	42	0	0	-1	0.1	0.062
6	43	1	1	1	0.1	0.062
7	45	0	1	-1	0.1	0.062
8	46	1	1	1	0.1	0.062
9	46	1	0	-1	0.1	**0.25**
10	51	1	0	-1	0.1	**0.25**

Ensemble Learning, Boosting:

Example of Boosting (cont.): Iteration 1, Step 4: generating bins ($[0, w_1]$, $[w_1, w_1+w_2]$, …)

#	Age	Likes height?	Likes goats?	Go climbing?	Weight	Weight 2	bins
1	23	0	0	-1	0.1	0.062	[0, 0.062]
2	31	1	1	1	0.1	0.062	(0.62, 0.124]
3	35	1	0	1	0.1	0.062	(0.124, 0.186]
4	35	0	0	-1	0.1	0.062	(0.186, 0.248]
5	42	0	0	-1	0.1	0.062	(0.248, 0.31]
6	43	1	1	1	0.1	0.062	(0.31, 0.372]
7	45	0	1	-1	0.1	0.062	(0.372, 0.434]
8	46	1	1	1	0.1	0.062	(0.434, 0.5]
9	46	1	0	-1	0.1	**0.25**	**(0.5, 0.75]**
10	51	1	0	-1	0.1	**0.25**	**(0.75, 1]**

Ensemble Learning, Boosting:

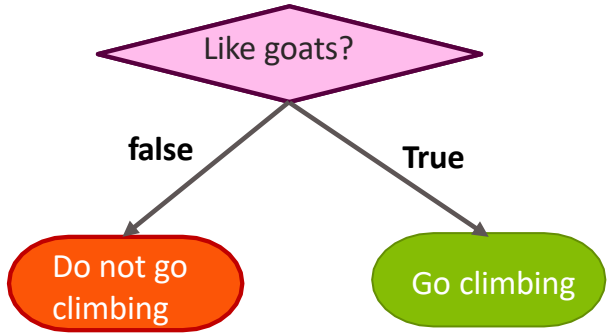
Example of Boosting (cont.): Iteration 1, Step 5: Random numbers

- Pick N random number between 0 and 1 (say [0.1, 0.15, 0.063, 0.55, 0.65, 0.05, 0.8, 0.7, 0.95, 0.97]).
    - Check in which bin the random number falls and pick the according data samples ([data#2, data#3, data#2, data#9, data#9, data#1, data#10, data#9, data#10, data#10]).
- Due to the higher weight of the misclassified example, this example has a larger bin, and the **probability of picking it is higher**.

#	Age	Likes height?	Likes goats?	Go climbing?
2	31	1	1	1
3	35	1	0	1
2	31	1	1	1
9	46	1	0	-1
9	46	1	0	-1
1	23	0	0	-1
10	51	1	0	-1
9	46	1	0	-1
10	51	1	0	-1
10	51	1	0	-1

Ensemble Learning, Boosting:

Example of Boosting (cont.): Iteration 2, Step 1: Fit a classifier to the training data



#	Age	Likes height?	Likes goats?	Go climbing?
2	31	1	1	1
3	35	1	0	1
2	31	1	1	1
9	46	1	0	-1
9	46	1	0	-1
1	23	0	0	-1
10	51	1	0	-1
9	46	1	0	-1
10	51	1	0	-1
10	51	1	0	-1

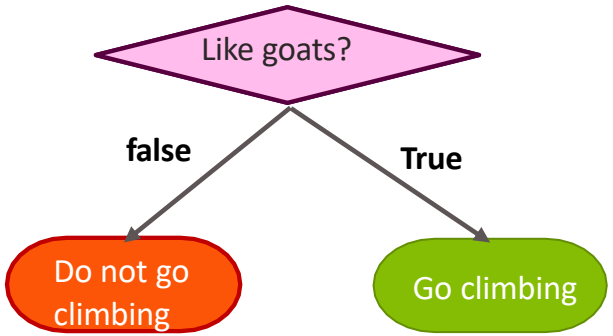
Ensemble Learning, Boosting:

Example of Boosting (cont.): Iteration 2, Step 2: Calculating err_m and α_m (influence)

$$err_2 = \frac{0.062*1 + 0 + 0 + 0 + 0 + 0 + 0 + 0 + 0 + 0}{1} = 0.062$$

$$\alpha_2 = \frac{1}{2} \ln \frac{1 - 0.062}{0.062}$$

$$\alpha_2 = \frac{1}{2} \ln 15.13 = 1.35$$



#	Age	Likes height?	Likes goats?	Go climbing?
2	31	1	1	1
3	35	1	0	1
2	31	1	1	1
9	46	1	0	-1
9	46	1	0	-1
1	23	0	0	-1
10	51	1	0	-1
9	46	1	0	-1
10	51	1	0	-1
10	51	1	0	-1

Ensemble Learning, Boosting:

Example of Boosting (cont.): Iteration 2, Step 3: Calculating $w_i^{new\_iteration}$

$$\boldsymbol{w_i^{new\_iteration} = w_i^{old\_iteration}\, e^{\pm\alpha}}$$

For correctly classified samples, we get:

$w_i^{new\_iteration} = 0.026e^{-1.35}$ = 0.0067
$w_9^{new\_iteration} = 0.25e^{-1.35}$ = 0.064

and for wrongly classified samples:

$w_3^{new\_iteration} = 0.026e^{+1.35}$ = 0.1

These weights still need to be normalized, so that their sum equals 1.

#	Age	Likes height?	Likes goats?	Go climbing?
2	31	1	1	1
3	35	1	0	1
2	31	1	1	1
9	46	1	0	-1
9	46	1	0	-1
1	23	0	0	-1
10	51	1	0	-1
9	46	1	0	-1
10	51	1	0	-1
10	51	1	0	-1

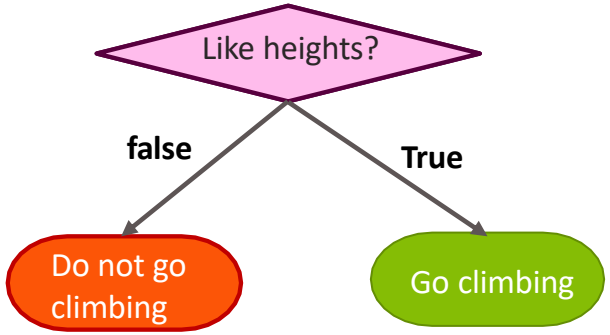
Ensemble Learning, Boosting:

Example of Boosting (cont.):

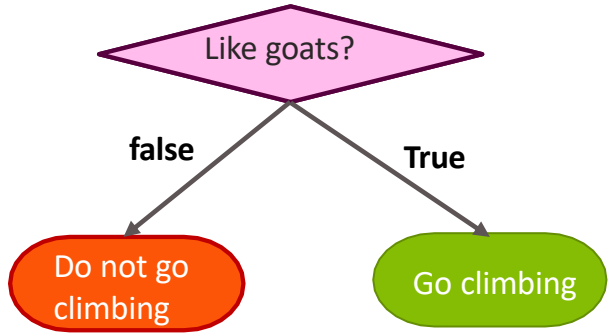
We reached M=2 and constructed two individual trees and their calculated values α.

Let's consider one of the samples in the dataset: [Age: 46, height: 1, goats: 0, climbing: 0]

Tree 1
α= 0.69



Tree 2
α= 1.35



Output= G(x)= sign[$\sum_{m=1}^{M} \alpha_m G_m(x)$]=

sign[(0.69*1 + 1.35*(-1)]= sign(-0.66)= 0. So, don't go rock climbing.

Ensemble Learning, Summary:

Aspect	Bagging	Random Forests	Boosting
Training	Parallel	Parallel	Sequential
Sampling	Bootstrapping	Bootstrapping + feature randomness	Weighted sampling (focus on errors)
Bias	Doesn’t reduce much	Doesn’t reduce much	Reduces bias significantly
Variance	Reduces variance	Strong variance reduction	Reduces variance + bias

Ensemble Learning, Summary:

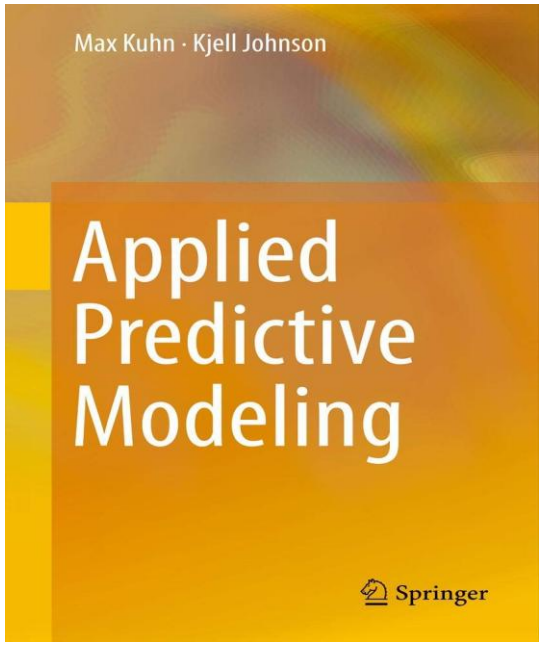
- Trees are sets of *if–then* rules used for classification or prediction.
  - They can be built using **entropy** (or other criteria) but typically suffer from high variance.

- Bagging mitigates **high variance** by training multiple trees on bootstrap resamples and aggregating their predictions.

- Random Forests improve **variance reduction** further by adding **randomness** to training (e.g., selecting random subsets of features at each split).
  - Trees are trained **in parallel** and weighted equally, which reduces variance well but does not significantly **reduce bias**.

- Boosting reduces both **bias and variance** by training trees **sequentially**, where each new tree focuses on correcting the mistakes of the previous ones, and weighting trees according to their performance.

References:



Chapter 8: Regression Trees and Rule-based Models

Chapter 8: Decision Trees