INNOPOLIS UNIVERSITY

Machine Learning

Prof. Adil Khan

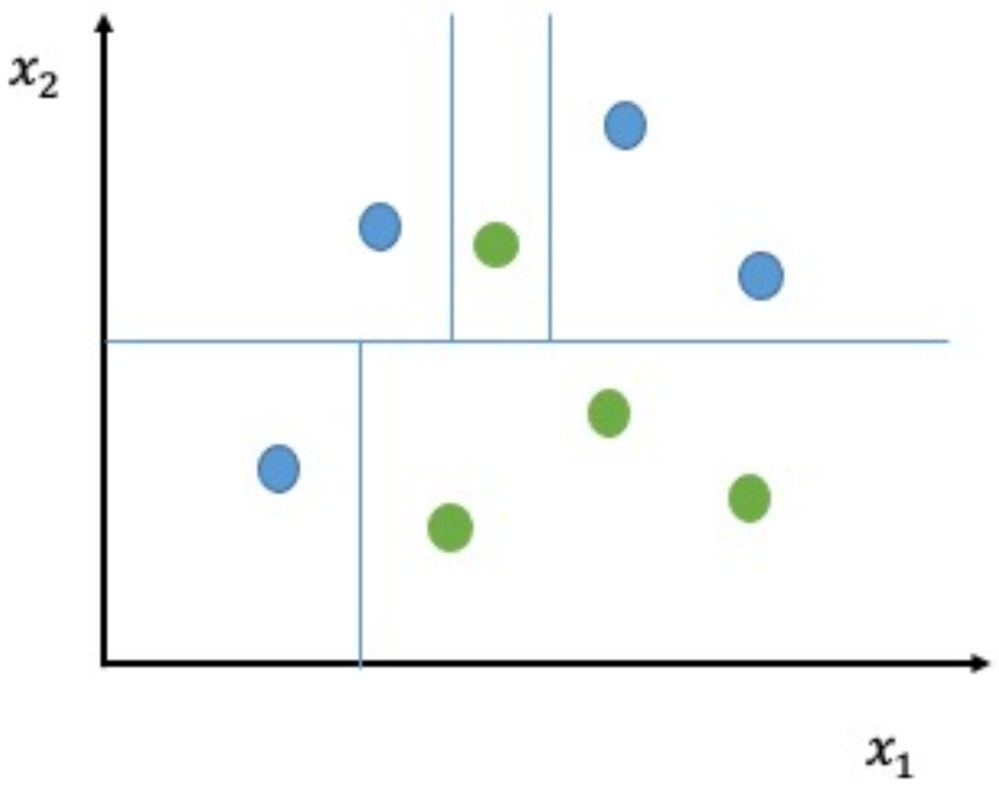
Today's Objectives

1. A quick recap of what we learned last week
2. Why do DT's overfit?
3. How can we avoid overfitting in DTs?
   - Early stopping
   - Pruning
4. What is Ensemble Learning? Why is it motivated?
   - Bagging
   - Boosting
5. RandomForest: an example of Bagging
6. Adaboost: an example of Boosting

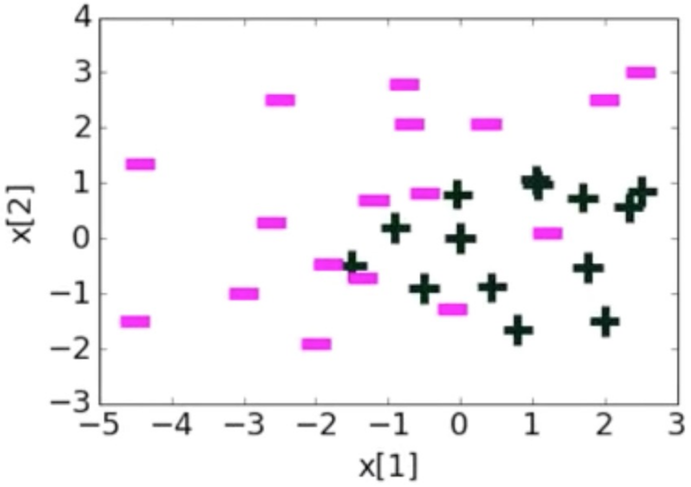
Recap

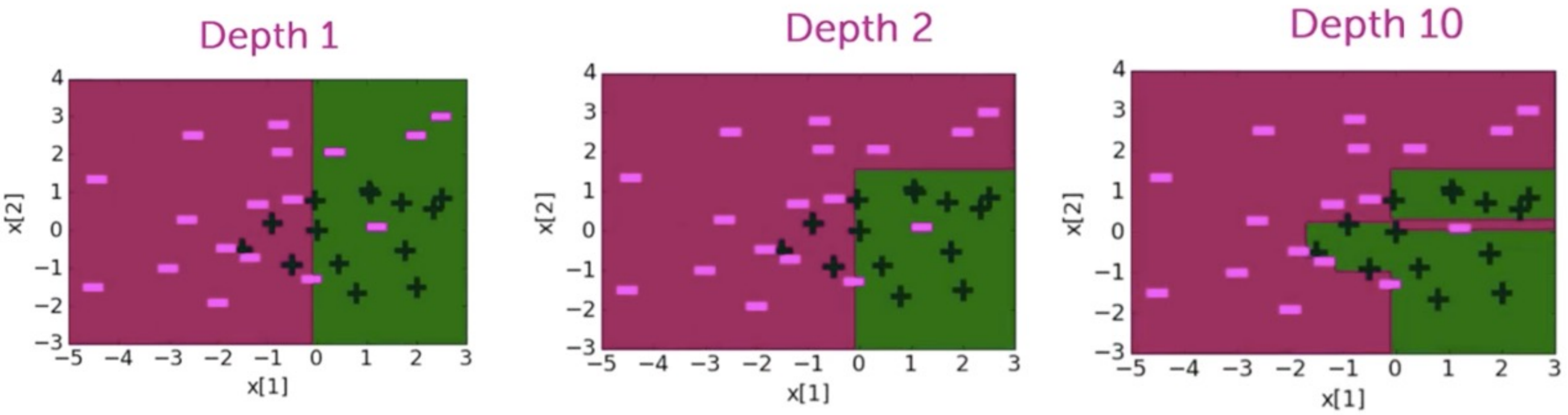
1. Machine Learning (Task, Expereince, Performance)
2. Predictors and Respone (Functional Relationship Estimation)
3. Regression and Classification
4. Linear Models
5. Non-linear Models
6. Representation Matters
7. Learning Representations

Recall: DT Class Boundaries



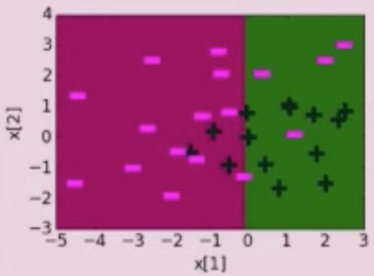
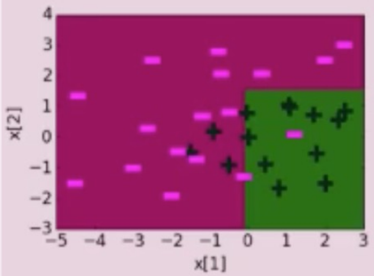
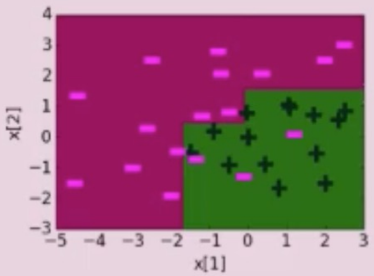
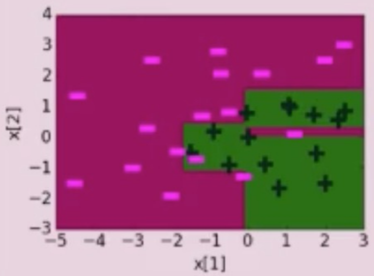
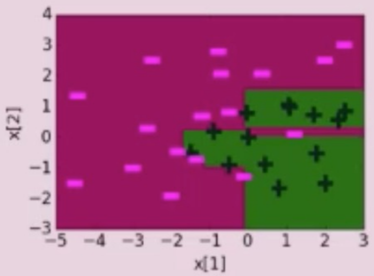
DT Boundaries at Different Depths





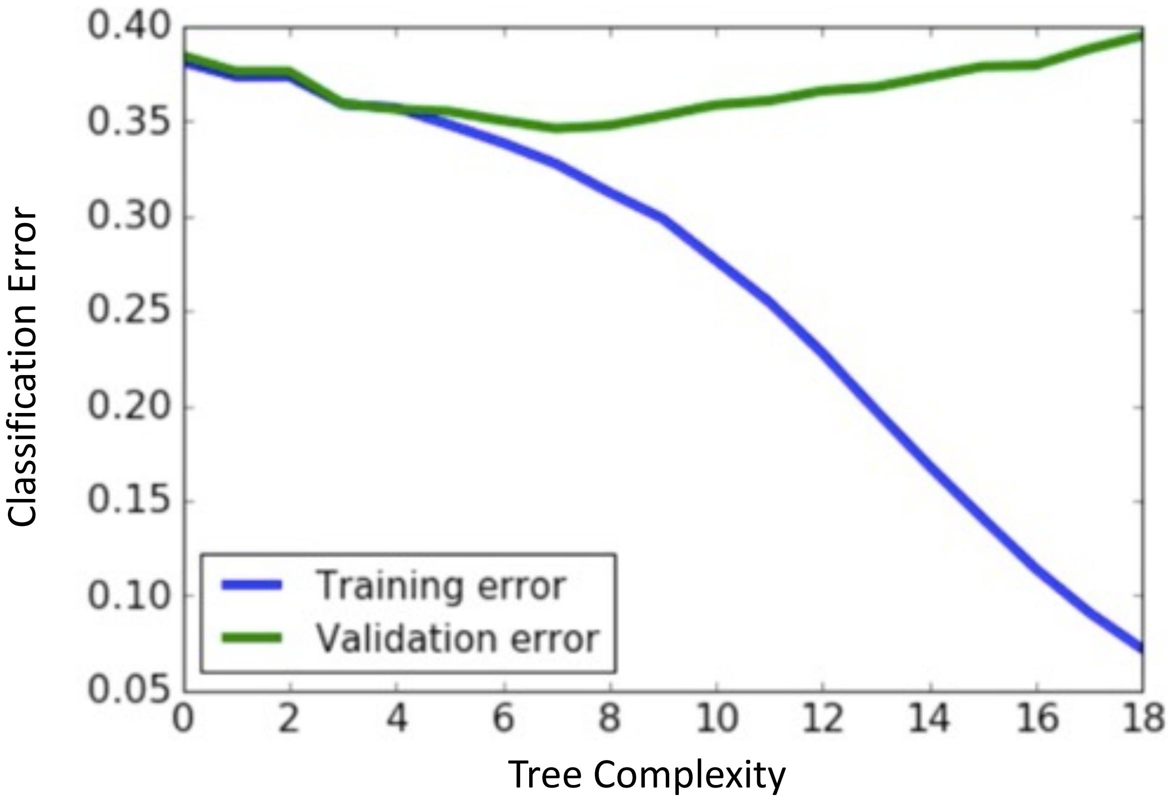
In General

Training error reduces with depth

Tree depth	depth = 1	depth = 2	depth = 3	depth = 5	depth = 10
Training error	0.22	0.13	0.10	0.03	0.00
Decision boundary					

Standard DT’s have no “learning Bias”

DT Overfitting



What Makes a Good DT?

- Not too small: need to handle important but possibly subtle distinctions in data
- Not too big:
  - Computational efficiency (avoid redundant, suprious attributes)
  - Avoid overfitting
  - Interpretability
- “Oscam’s Razor”: find the simplest hypothesis that fits the observations
- Thus, we favor simple trees with good performance

What do we mean by “Simple is Better”?

- When two trees have the same classification error on validation set, choose the one that is simpler

Tree Complexity	Training Error	Validation Error
Low	0.23	0.24
Moderate	0.12	0.15
Complex	0.7	0.15
Super Complex	0.0	0.18

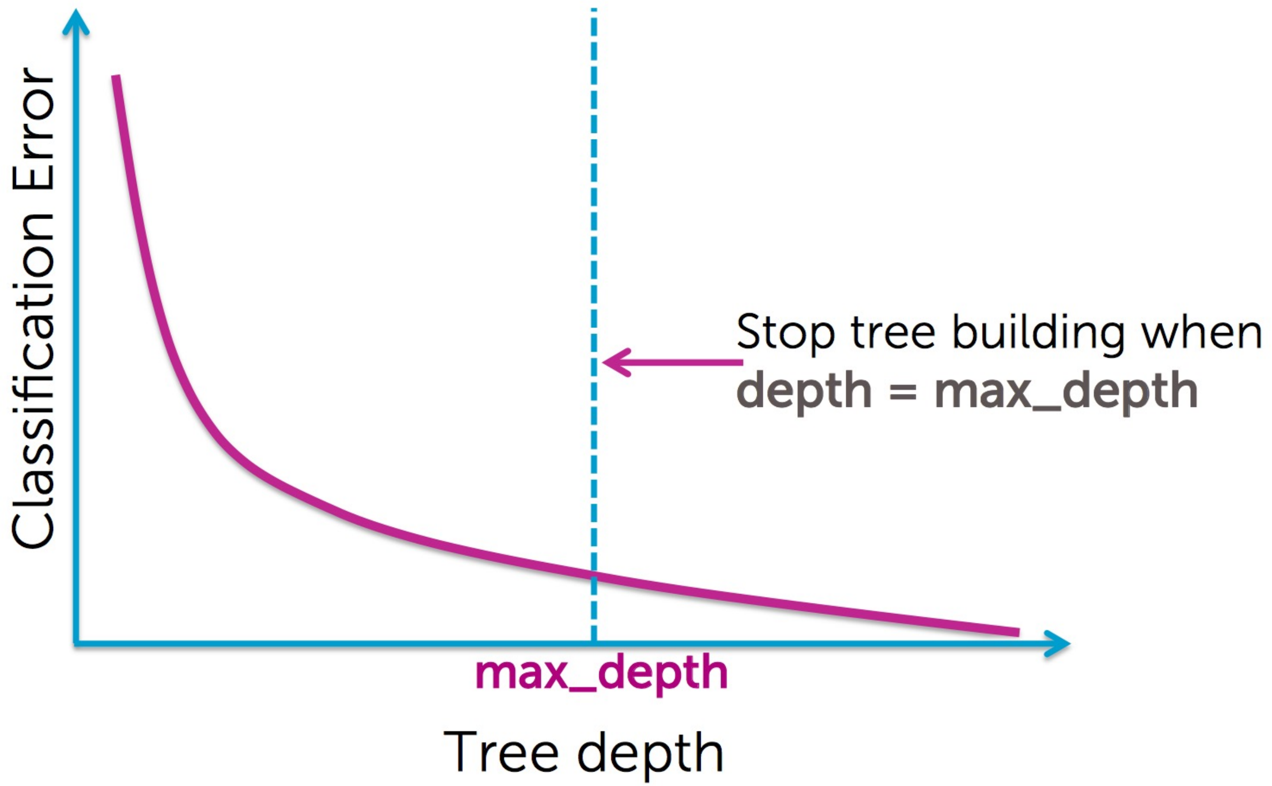
How to avoid Overfitting in DTs?

1. Early Stopping: Stop learning before the tree becomes too complex

2. Pruning: Simplify tree after learning algorithm terminates

Early Stopping: Criteria 1

- **Limit the depth:** stop splitting after max_depth is reached

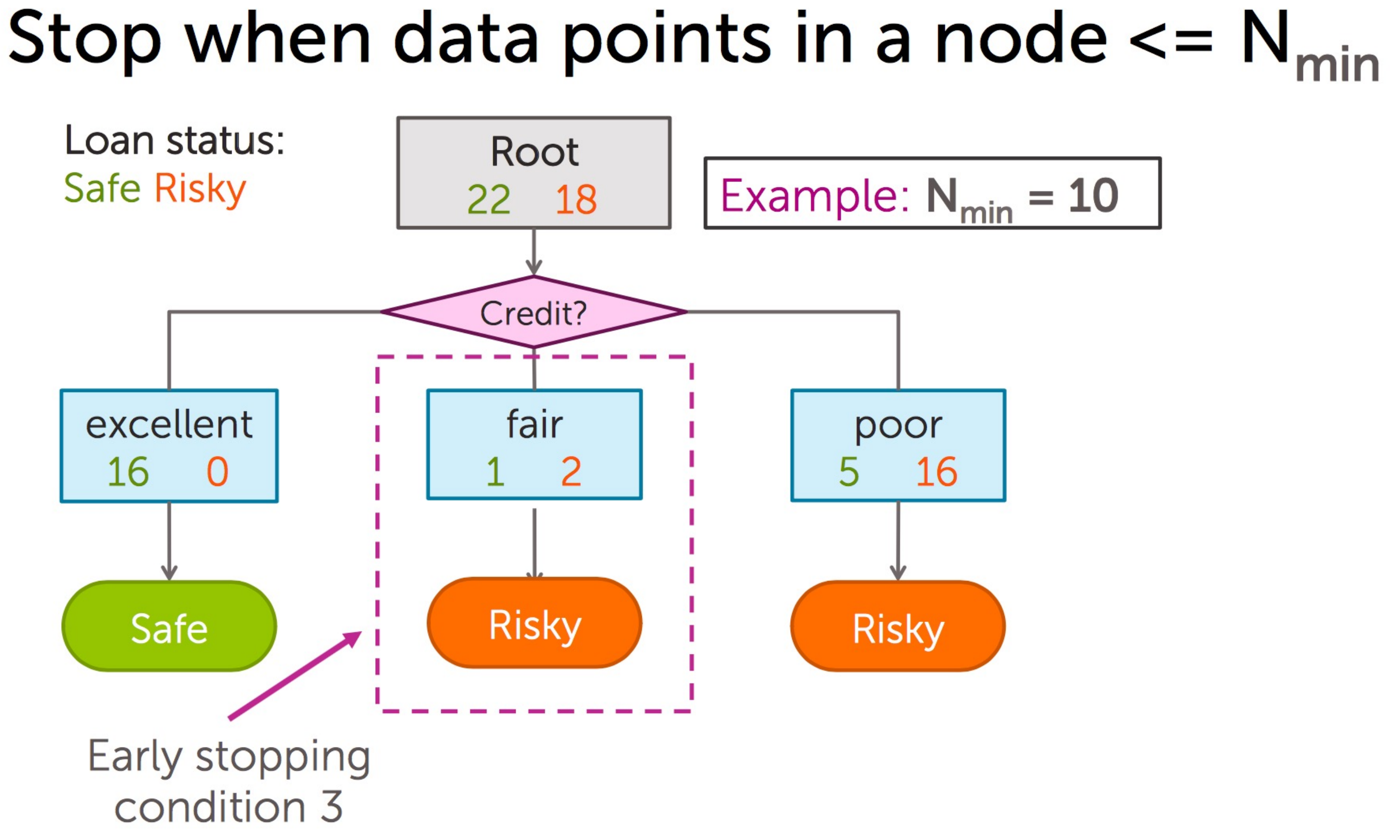


Early Stopping: Criteria 2

- Use a **threshold for decrease ε** in prediction error with a split
  - Stop if the error does not decrease more than ε

Early Stopping: Criteria 3

- Use a threshold for the number of data points at a node for it to be eligible for further split



Summary: Early Stopping

1. Limit tree depth
   - Stop splitting after a certain depth

2. Prediction performance
   - Do not consider a split unless it provides a significant boost in prediction performance

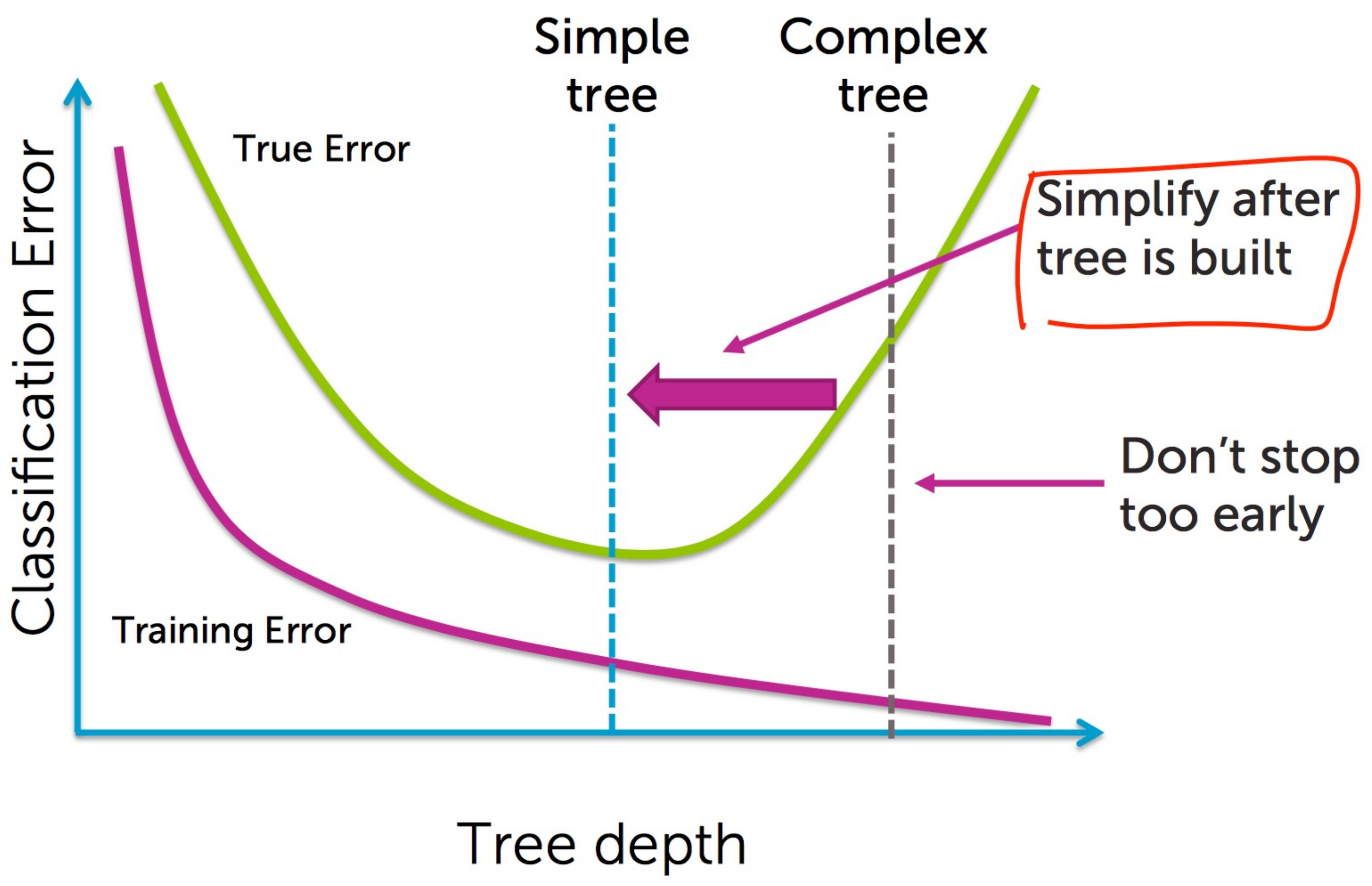
3. Minimum node size
   - Do not split a node unless it contains a significant number of data points

Pruning

- Grow a large tree and then prune back some nodes

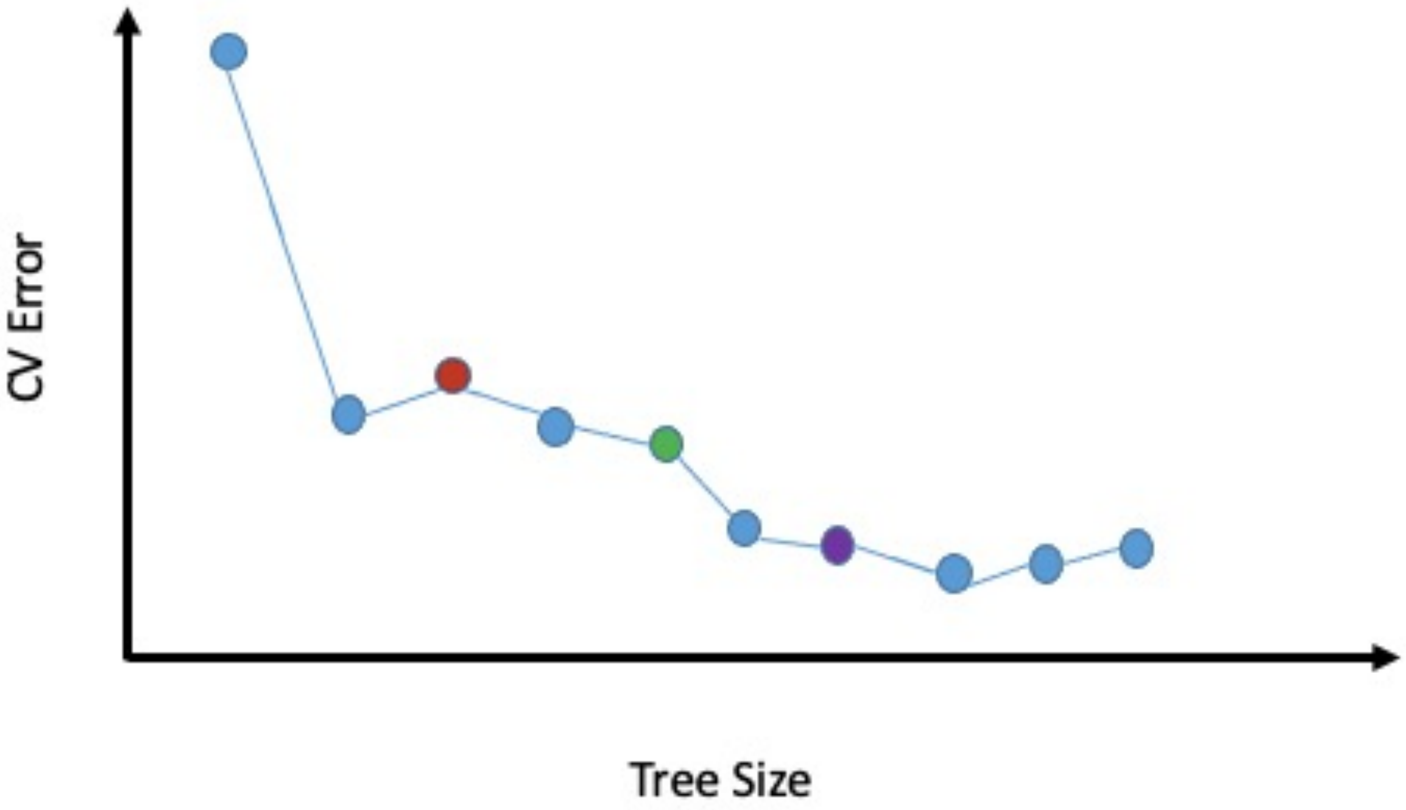
- In other words,
  - We learn a big, complex tree
  - Then we use pruning to reduce the size of the tree by removing parts of the tree

Pruning (2)

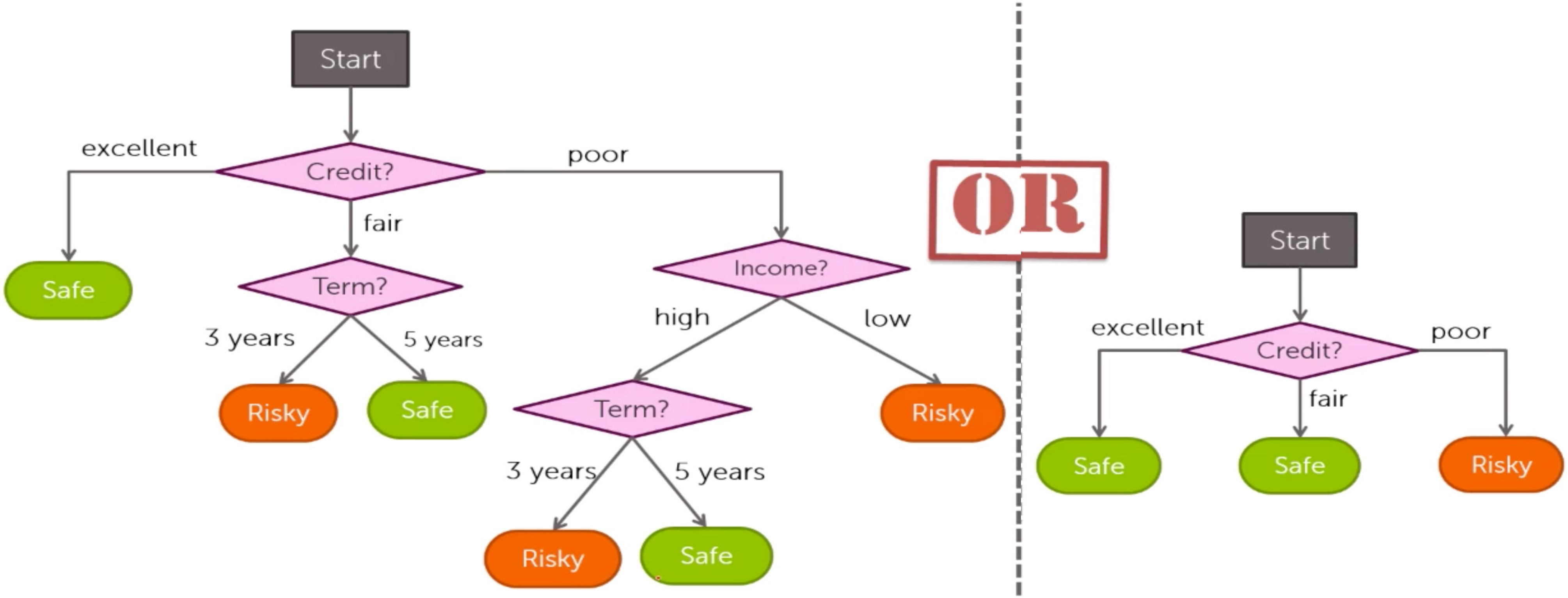


Simple tree
Complex tree
True Error
Simplify after tree is built
Don't stop too early
Classification Error
Training Error
Tree depth

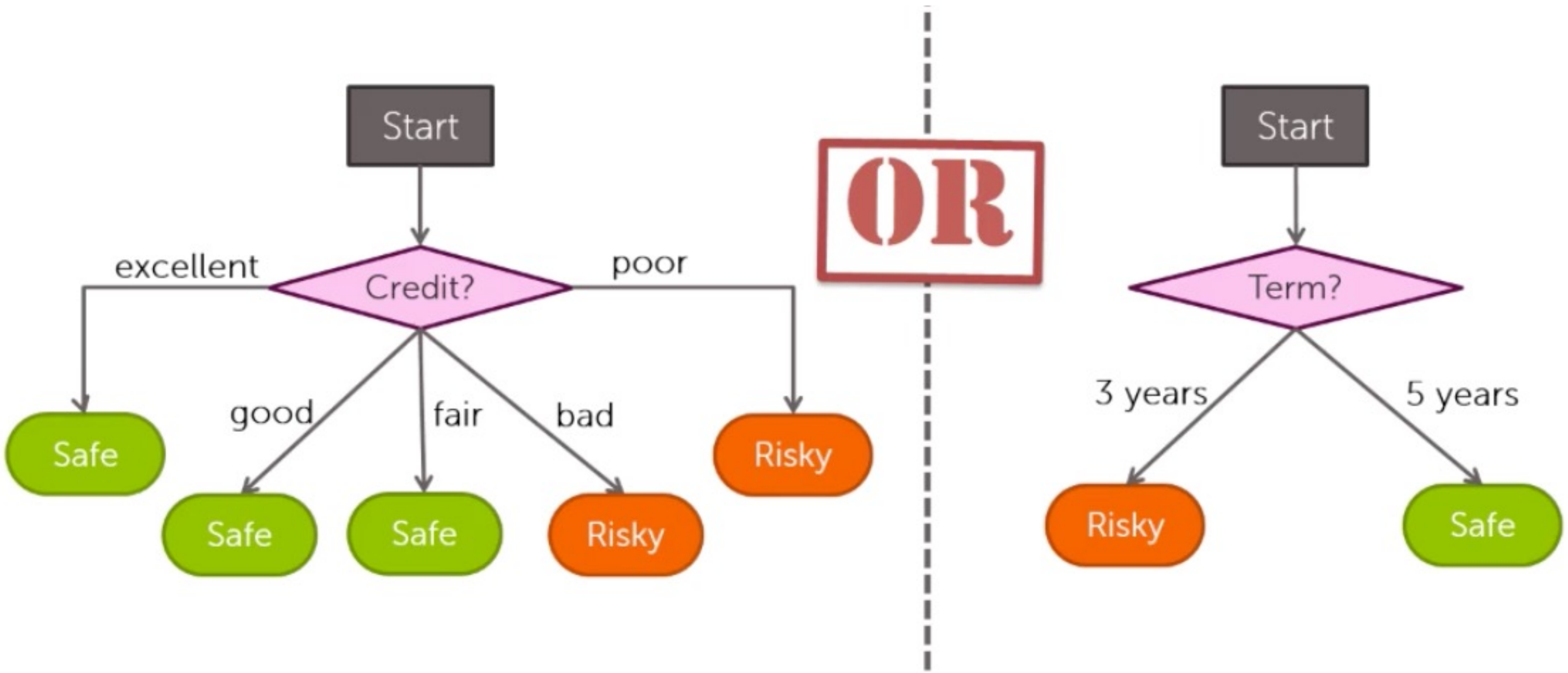
Why prune?



Which of These Trees is Simpler?

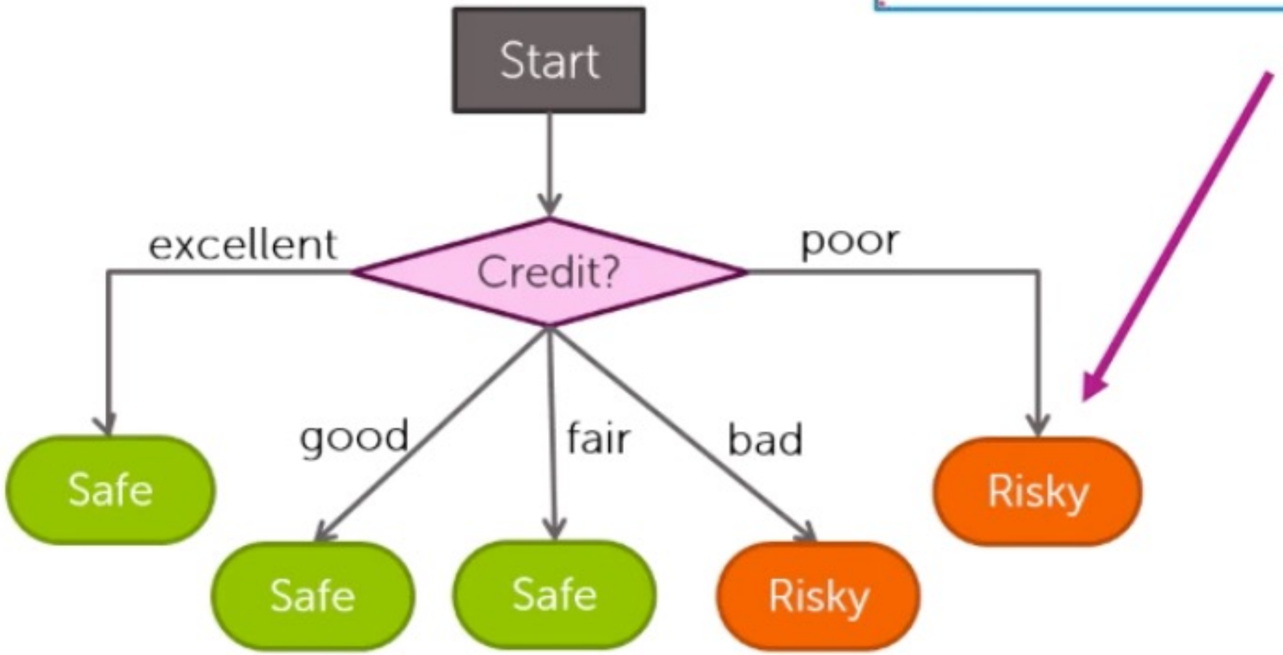


Which of These Trees is Simpler? (2)



Thus, Our Measure of Complexity is

$L(T)$ = # of leaf nodes



New Learning Objective

Total Cost = Measure of Fit + Measure of Complexity

$$C(T) = Error(T) + \lambda L(T)$$

$Error(T)$ is prediction error (large means bad fit to the data)

$L(T)$ is Number of Leaves (large means likely to overfit)

DT Pruning Algorithm

- Let T be the final tree
- Start at the bottom of T and traverse up, apply *prune_split* at each decision node M

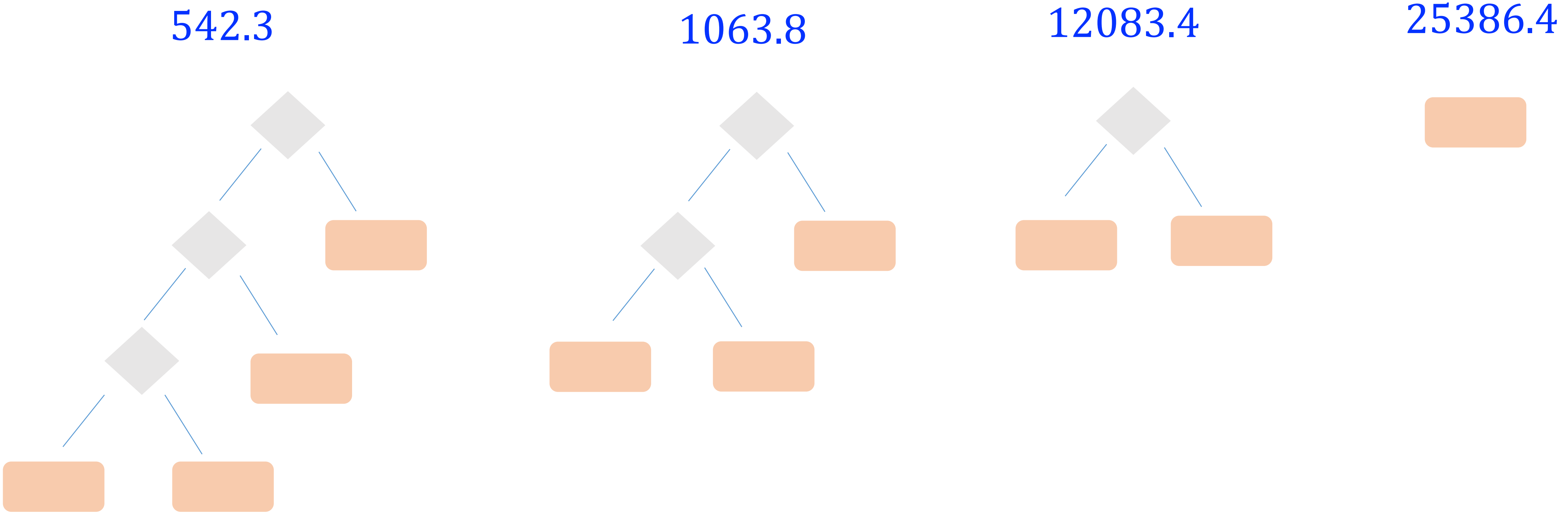
prune_split

- Prune_split (T, M)

1. Compute total cost $C(T)$
2. Let T_{small} be the tree after pruning T at M
3. Compute $C(T_{small})$
4. If $C(T_{small}) < C(T)$, prune T to T_{small}

Example

$$C(T) = Error(T) + \lambda L(T)$$

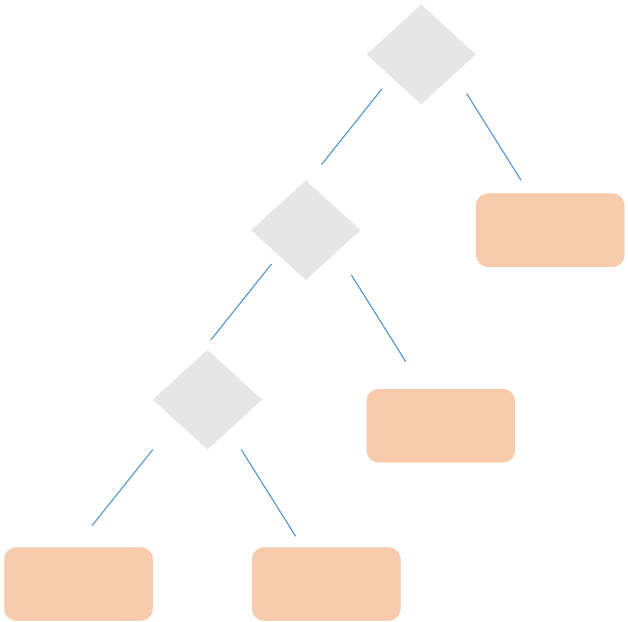


Example

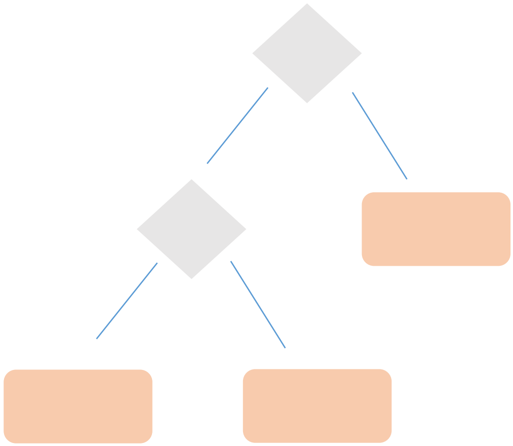
$$C(T) = Error(T) + \lambda L(T)$$

λ=10000

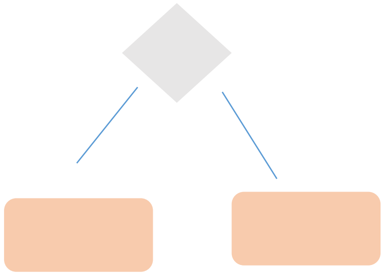
$542.3 + 10000 \times 4$



$1063.8 + 10000 \times 3$



$12083.4 + 10000 \times 2$



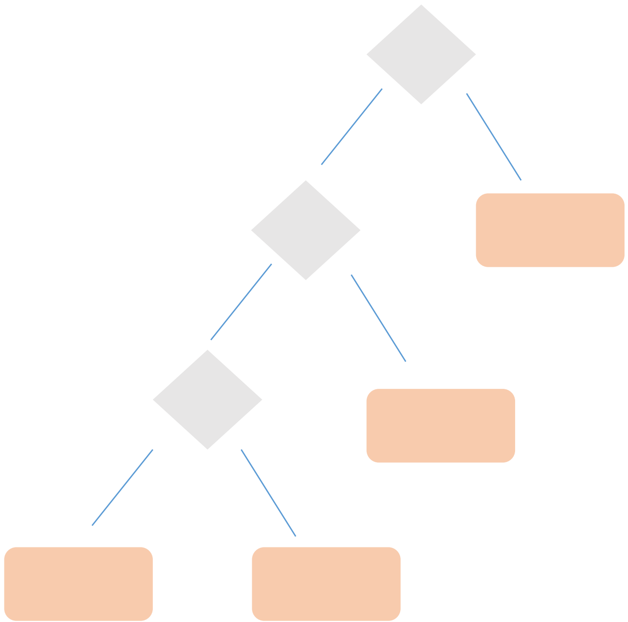
$25386.4 + 10000 \times 1$

Example

$$C(T) = Error(T) + \lambda L(T)$$

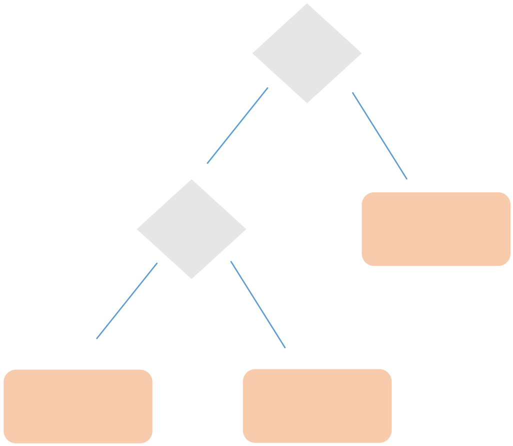
λ=10000

$542.3 + 10000 \times 4$



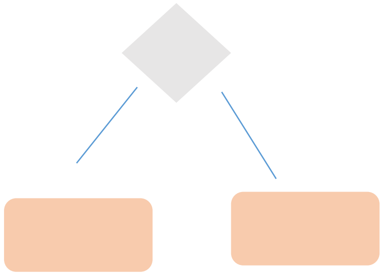
$C(T) = 40542.3$

$1063.8 + 10000 \times 3$



$C(T) = 31063.8$

$12083.4 + 10000 \times 2$



$C(T) = 32083.4$

$25386.4 + 10000 \times 1$

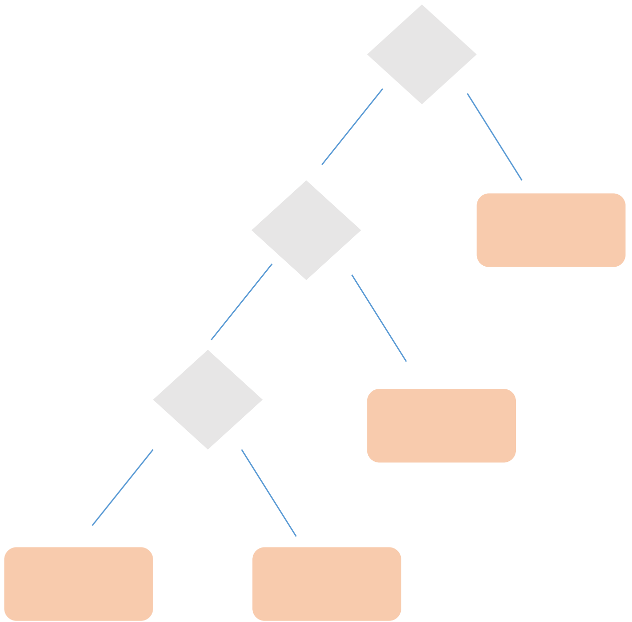
$C(T) = 35386{,}4$

Example

$$C(T) = Error(T) + \lambda L(T)$$

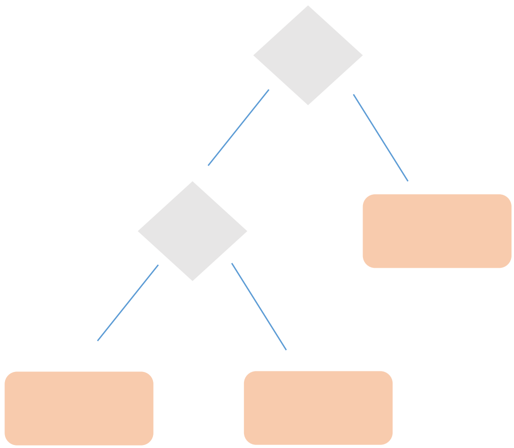
λ=10000

$$542.3 + 10000\times4$$



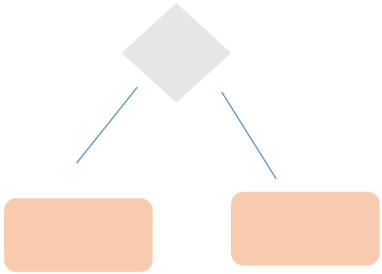
$$C(T) = 40542.3$$

$$1063.8 + 10000\times3$$



$$\boldsymbol{C(T) = 31063.8}$$

$$12083.4 + 10000\times2$$



$$C(T) = 32083.4$$

$$25386.4 + 10000\times1$$

$$C(T) = 35386,4$$

Ensemble Learning

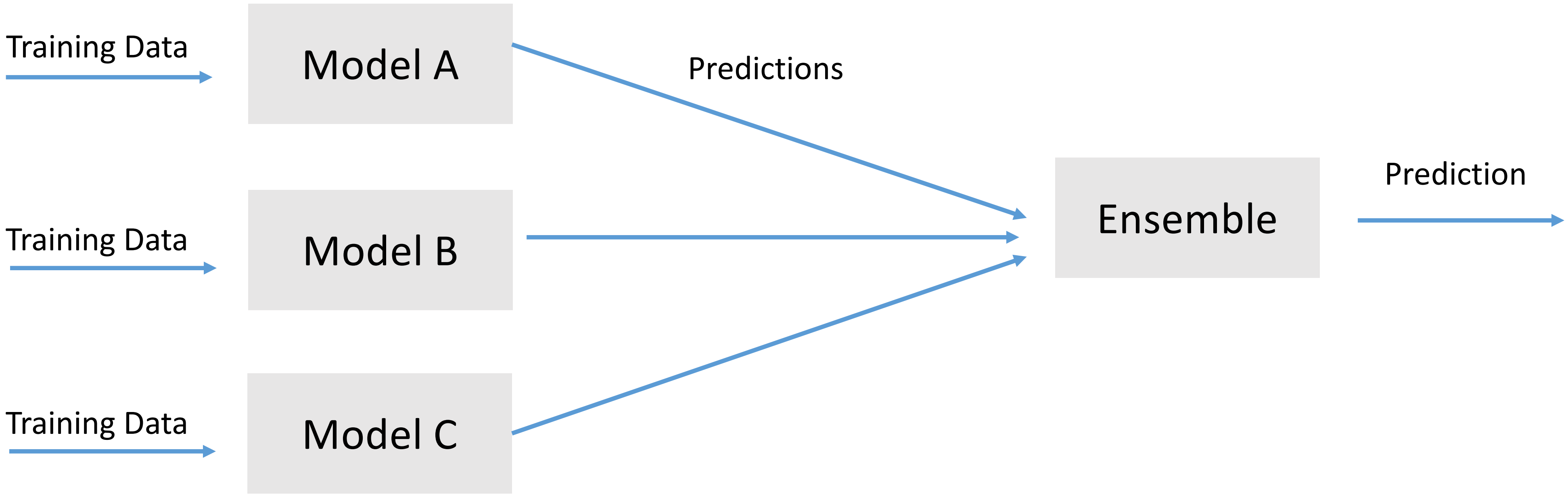
Recall: Bias and Variance

- A complex model could exhibit high variance
- A simple model could exhibit high bias

We can solve each case with ensemble learning.
Let’s first see what is ensemble learning.

Ensemble Learning

- Meta-learning algorithms that combine several ML models into one predictive model



Ensemble Model in General

- Goal:
  - Predict output y
    - Either +1 or -1
  - From input **x**
- Learn ensemble model:
  - Classifiers: $f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_T(\mathbf{x})$
  - Coefficients: $\hat{w}_1, \hat{w}_2, \ldots, \hat{w}_T$
- Prediction:

$$\hat{y} = sign\left(\sum_{t=1}^{T} \hat{\mathbf{w}}_t f_t(\mathbf{x})\right)$$

Ensemble Model in General (2)

- Goal:
  - Predict output y
    - Either +1 or -1
  - From input **x**
- Learn ensemble model:
  - Classifiers: $f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_T(\mathbf{x})$
  - Coefficients: $\hat{w}_1, \hat{w}_2, \ldots, \hat{w}_T$
- Prediction:

$$\hat{y} = sign\left(\sum_{t=1}^{T} \hat{\mathbf{w}}_t f_t(\mathbf{x})\right)$$

Ensemble Model in General (3)

- Goal:
  - Predict output y
    - Either +1 or -1
  - From input **x**
- Learn ensemble model:
  - Classifiers: $f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_T(\mathbf{x})$
  - Coefficients: $\hat{w}_1, \hat{w}_2, \ldots, \hat{w}_T$
- Prediction:

$$\boxed{\hat{y} = sign\left(\sum_{t=1}^{T} \hat{\mathbf{w}}_t f_t(\mathbf{x})\right)}$$

Bagging: Reducing Variance using An Ensemble of Classifiers from Bootstrap Samples

Important

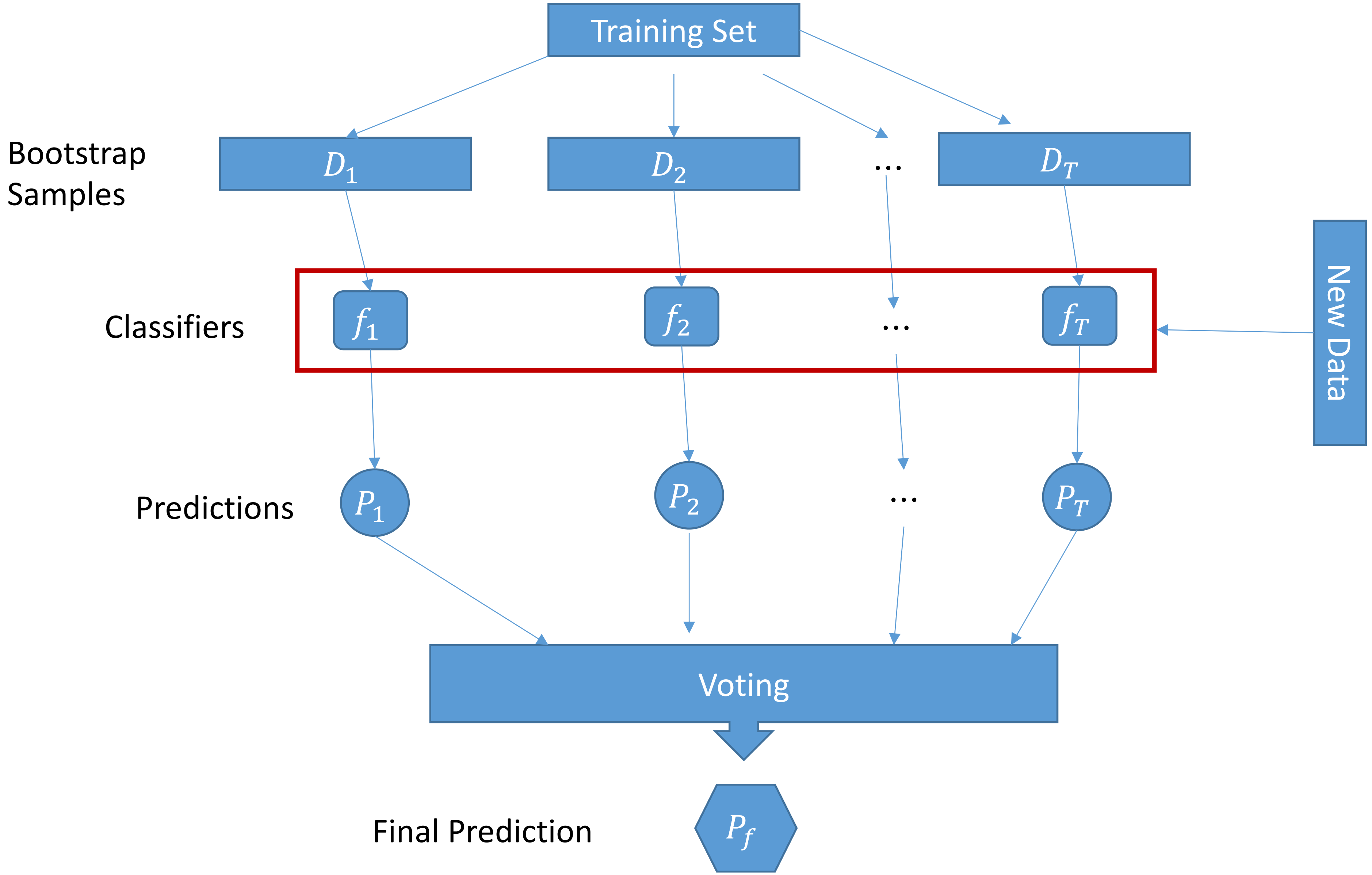
- In order for ensemble methods to be more accurate than any of its individual members, the base learners have to be as accurate as possible and as diverse as possible (Hansen & Salamon, 1990)

Aside: Bootstrapping

Training Data	Bootstrap 1	Bootstrap 2	…
1	2	7	…
2	2	3	…
3	1	2	…
4	3	1	…
5	7	1	…
6	2	7	…
7	4	7	…

Creating new datasets from the training data *with replacement*

Bagging

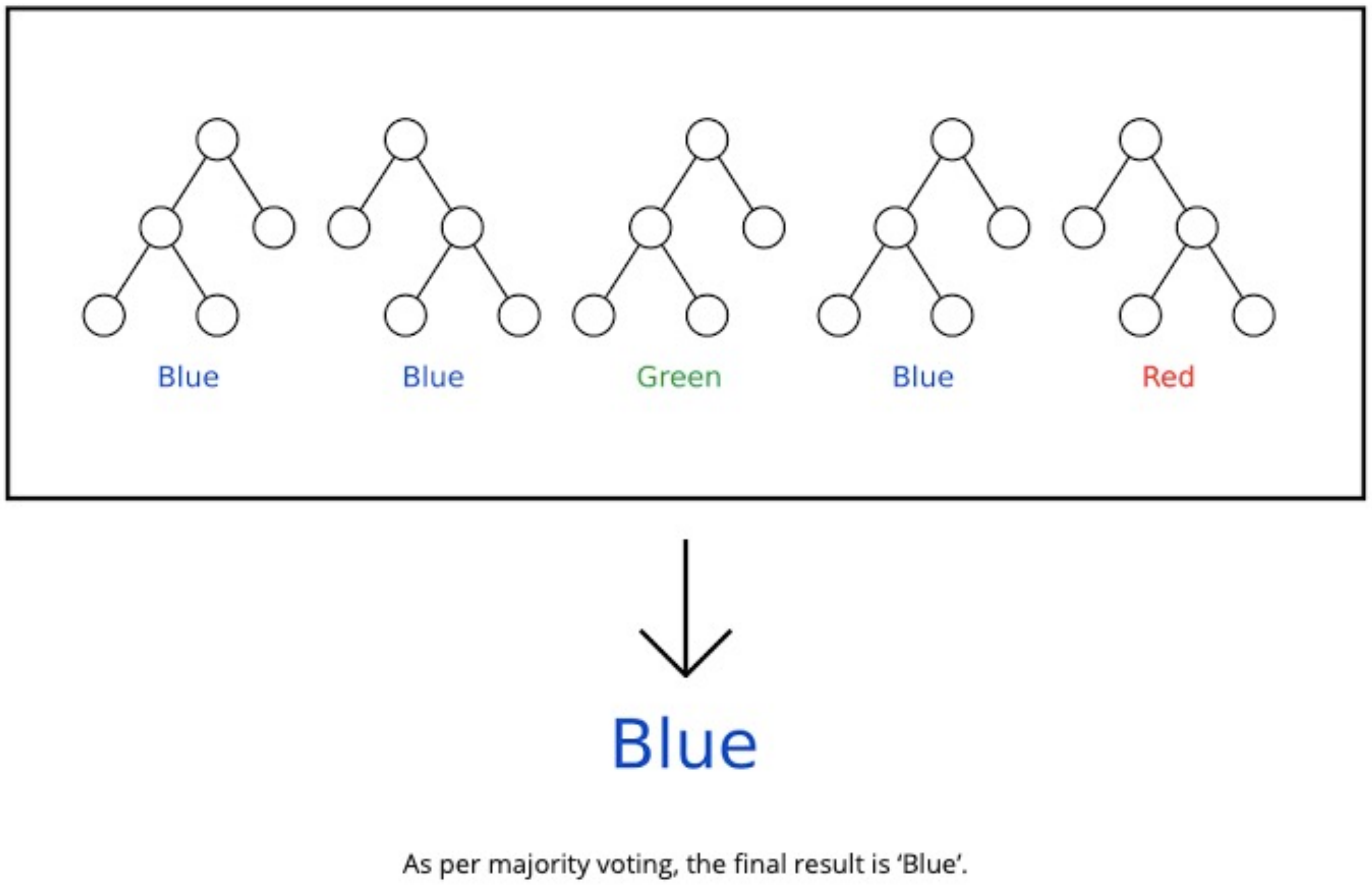


Training Set
Bootstrap Samples
D_1
D_2
…
D_T
Classifiers
f_1
f_2
…
f_T
New Data
Predictions
P_1
P_2
…
P_T
Voting
Final Prediction
P_f

Random Forests – Example of Bagging

1. Draw a random **bootstrap** sample
2. Grow a decision tree from the bootstrap sample. At each node:
   a) **Randomly** select **d** features without replacement ($d = \sqrt{n}$).
   b) Split the node using the feature that provides the best split according to the objective function, for instance, by maximizing the information gain.
3. Repeat the steps 1 to 2 **k** times.
4. Aggregate the prediction by each tree to assign the class label by **majority voting**

Making Prediction with a Tree Ensemble



https://www.kdnuggets.com/2020/01/random-forest-powerful-ensemble-learning-algorithm.html

Making Prediction with a Tree Ensemble (2)

the ensemble of trees $\{T_b\}_1^B$

To make a prediction at a new point x:

Regression: $\hat{f}_{\mathrm{rf}}^B(x) = \frac{1}{B}\sum_{b=1}^B T_b(x)$.

Classification: Let $\hat{C}_b(x)$ be the class prediction of the bth random-forest tree. Then $\hat{C}_{\mathrm{rf}}^B(x) = \textit{majority vote } \{\hat{C}_b(x)\}_1^B$.

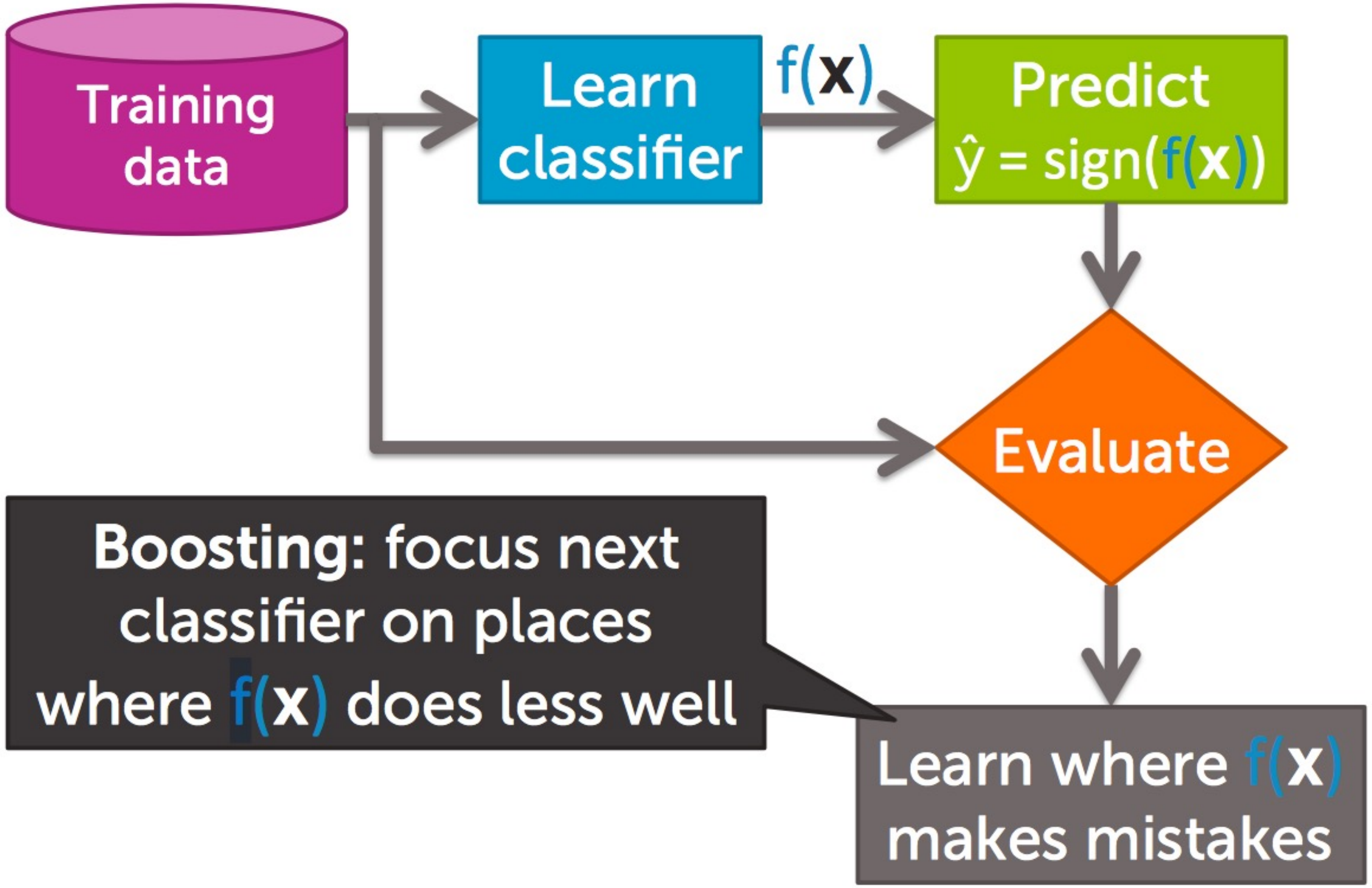
Boosting: Converting Weak Learners to Strong Learners through Ensemble Learning

Boosting vs. Bagging

- Works in a similar way as bagging
- Except:
  - Models are built sequentially: each model is built using information from previously built models.
  - Boosting does not involve bootstrap sampling; instead each tree is fit on a modified version of the original data set

Boosting (1)

Boosting: (2) Train Next Classifier by Focusing More on the Hard Points

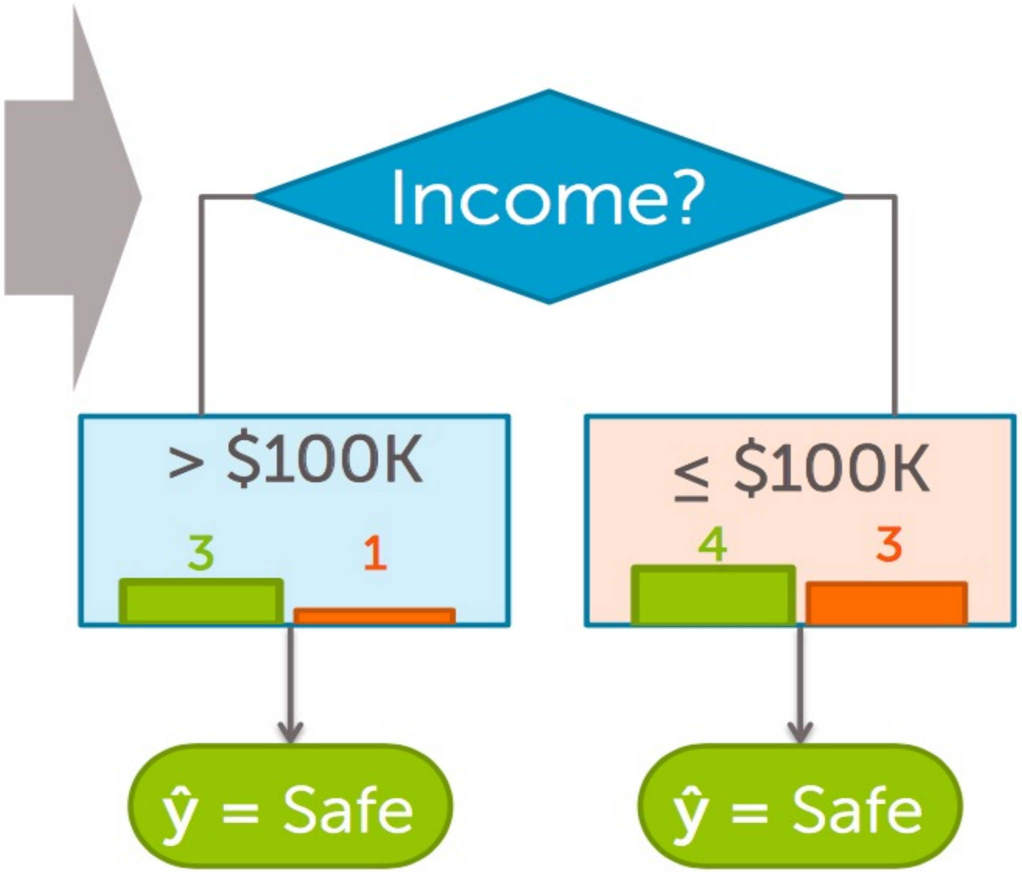


What does it mean to focus more?

- Weighted Dataset
  - Each $(\boldsymbol{x}_i, y_i)$ is weighted by α_i
  - More important point $\boldsymbol{x}_i$ = higher weight α_i

Example : Learning a Simple Decision Stump

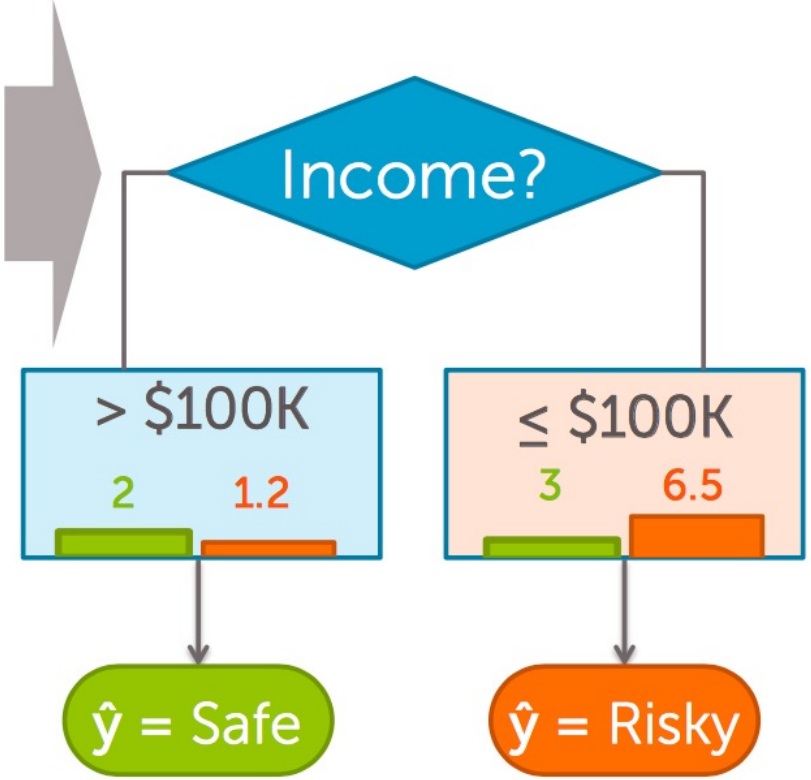
Credit	Income	y
A	$130K	Safe
B	$80K	Risky
C	$110K	Risky
A	$110K	Safe
A	$90K	Safe
B	$120K	Safe
C	$30K	Risky
C	$60K	Risky
B	$95K	Safe
A	$60K	Safe
A	$98K	Safe



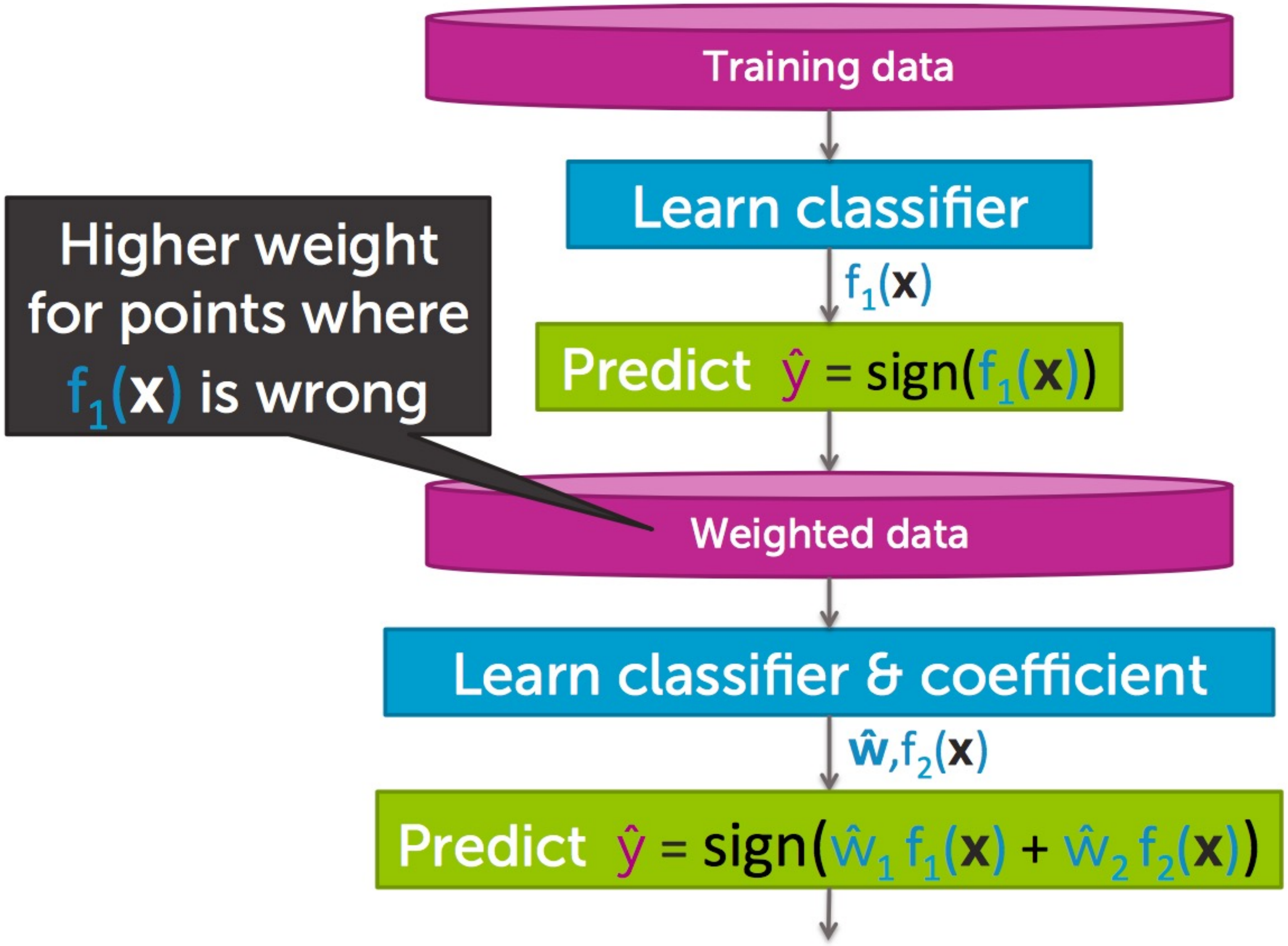
Example: Learning a Decision Stump on Weighted Data

Increase weight **α** of harder/ misclassified points

Credit	Income	y	Weight α
A	$130K	Safe	0.5
B	$80K	Risky	1.5
C	$110K	Risky	1.2
A	$110K	Safe	0.8
A	$90K	Safe	0.6
B	$120K	Safe	0.7
C	$30K	Risky	3
C	$60K	Risky	2
B	$95K	Safe	0.8
A	$60K	Safe	0.7
A	$98K	Safe	0.9



Boosting



Training data
Learn classifier
$f_1(\mathbf{x})$
Predict $\hat{y} = \text{sign}(f_1(\mathbf{x}))$
Higher weight for points where $f_1(\mathbf{x})$ is wrong
Weighted data
Learn classifier & coefficient
$\hat{w}, f_2(\mathbf{x})$
Predict $\hat{y} = \text{sign}(\hat{w}_1 f_1(\mathbf{x}) + \hat{w}_2 f_2(\mathbf{x}))$

AdaBoost (Example of Boosting)

1. Start with the same weights for all points: $\alpha_i = \frac{1}{m}$
2. For each $t = 1, \cdots, T$
   - Learn $f_t(x)$ with data weights α_i
   - Compute coefficient $\hat{w}_t$
   - Recompute weights α_i

Weight of the model

New weights of the data points

- Final model predicts as:

$$\hat{y} = sign\left(\sum_{t=1}^{T} \hat{w}_t \, f_t(x)\right)$$

AdaBoost: Computing $\hat{w}_t$ of f_t

Yes → Large $\hat{w}_t$

Is f_t good?

No → Small $\hat{w}_t$

- f_t is good → f_t has low training error

$$\hat{w}_t = \frac{1}{2} ln\left(\frac{1 - weighted\ error(f_t)}{weighted\ error(f_t)}\right)$$

Weighted Prediction Error

- Total weight of the mistakes:

$$= \sum_{i=1}^{m} \alpha_i I(\hat{y}_i \neq y_i)$$

- Total weight of all points:

$$= \sum_{i=1}^{m} \alpha_i$$

- Weighted error measures fraction of weight of mistakes:

$$= \frac{Total\ weight\ of\ the\ mistakes}{Total\ weight\ of\ all\ points}$$

AdaBoost: Computing $\hat{w}_t$ of f_t

$$\hat{w}_t = \frac{1}{2} ln\left(\frac{1 - weighted\ error(f_t)}{weighted\ error(f_t)}\right)$$

Weighted error on training data	$\frac{1 - weighted\ error(f_t)}{weighted\ error(f_t)}$	$\hat{w}_t$
0.01		
0.5		
0.99		

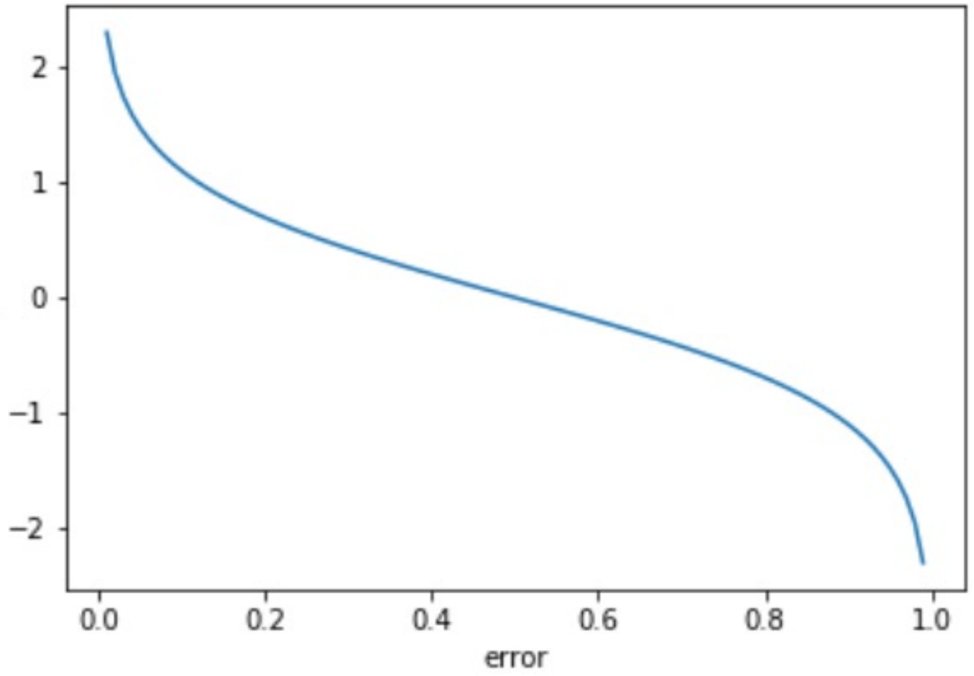
AdaBoost: Computing $\hat{w}_t$ of f_t

$$\hat{w}_t = \frac{1}{2} ln\left(\frac{1 - weighted\ error(f_t)}{weighted\ error(f_t)}\right)$$

Weighted error on training data	$\frac{1 - weighted\ error(f_t)}{weighted\ error(f_t)}$	$\hat{w}_t$
0.01	99	2.297
0.5	1	0
0.99	0.01	-2.3

AdaBoost: Computing $\hat{w}_t$ of f_t

$$\hat{w}_t = \frac{1}{2} ln\left(\frac{1 - weighted\ error(f_t)}{weighted\ error(f_t)}\right)$$



AdaBoost

1. Start with the same weights for all points: $\alpha_i = \frac{1}{m}$
2. For each $t = 1, \cdots, T$
   - Learn $f_t(x)$ with data weights α_i
   - Compute coefficient $\hat{w}_t$
   - Recompute weights α_i

$$\boxed{\hat{w}_t = \frac{1}{2} ln\left(\frac{1 - weighted\ error(f_t)}{weighted\ error(f_t)}\right)}$$

- Final model predicts as:

$$\hat{y} = sign\left(\sum_{t=1}^{T} \hat{w}_t\, f_t(x)\right)$$

AdaBoost: Updating α_i

Did f_t get x_i right?

- Yes → Decrease α_i
- No → Increase α_i

$$\alpha_i \leftarrow \begin{cases} \alpha_i e^{-\hat{w}_t}, & \text{if } f_t(\mathbf{x}_i)=y_i \\ \alpha_i e^{\hat{w}_t}, & \text{if } f_t(\mathbf{x}_i)\neq y_i \end{cases}$$

AdaBoost: Updating α_i

$$\alpha_i \leftarrow \begin{cases} \alpha_i e^{-\hat{w}_t}, & \text{if } f_t(\mathbf{x}_i)=y_i \\ \alpha_i e^{\hat{w}_t}, & \text{if } f_t(\mathbf{x}_i)\neq y_i \end{cases}$$

Predicted Label	$\hat{w}_t$	$e^{-\hat{w}_t}$ OR $e^{\hat{w}_t}$	Result
Correct	2.3	0.1	?
Correct	0	1	?
Mistake	2.3	9.98	?
Mistake	0	1	?

Increase, Decrease, or Keep the Same

AdaBoost

1. Start with the same weights for all points: $\alpha_i = \frac{1}{m}$
2. For each $t = 1, \cdots, T$
   - Learn $f_t(x)$ with data weights α_i
   - Compute coefficient $\hat{w}_t$
   - Recompute weights α_i

$$\hat{w}_t = \frac{1}{2} ln\left(\frac{1 - weighted\ error(f_t)}{weighted\ error(f_t)}\right)$$

$$\alpha_i \leftarrow \begin{cases} \alpha_i e^{-\hat{w}_t}, & \text{if } f_t(\mathbf{x}_i) = y_i \\ \alpha_i e^{\hat{w}_t}, & \text{if } f_t(\mathbf{x}_i) \neq y_i \end{cases}$$

- Final model predicts as:

$$\hat{y} = sign\left(\sum_{t=1}^{T} \hat{w}_t f_t(x)\right)$$

Important: Normalizing Weights

If $\boldsymbol{x}_i$ is often mistaken, α_i could get very large

If $\boldsymbol{x}_i$ is often correct, α_i could get very small

Can cause numerical instability

Thus we normalize weights after every iteration

$$\alpha_i \leftarrow \frac{\alpha_i}{\sum_{j=1}^{N} \alpha_j}$$

AdaBoost

1. Start with the same weights for all points: $\alpha_i = \frac{1}{m}$
2. For each $t = 1, \cdots, T$
   - Learn $f_t(x)$ with data weights α_i
   - Compute coefficient $\hat{w}_t$
   - Recompute weights α_i
   - Normalize α_i

- Final model predicts as:

$$\hat{y} = sign\left(\sum_{t=1}^{T} \hat{w}_t \, f_t(x)\right)$$

$$\boxed{\hat{w}_t = \frac{1}{2} ln\left(\frac{1 - weighted\ error(f_t)}{weighted\ error(f_t)}\right)}$$

$$\alpha_i \leftarrow \begin{cases} \alpha_i e^{-\hat{w}_t}, & \text{if } f_t(\mathbf{x}_i) = y_i \\ \alpha_i e^{\hat{w}_t}, & \text{if } f_t(\mathbf{x}_i) \neq y_i \end{cases}$$

$$\boxed{\alpha_i \leftarrow \frac{\alpha_i}{\sum_{j=1}^{N} \alpha_j}}$$

Self-Study

- What is the effect of of:

  - Increasing the number of classifiers in *bagging*

  vs.

  - Increasing the number of classifiers in *boosting*

- Why does Bagging reduce variance?

Summary

1. Why do DT’s overfit?
2. How can we avoid overfitting in DTs?
   - Early stopping
   - Pruning
3. What is Ensemble Learning? Why is it motivated?
   - Bagging
   - Boosting
4. RandomForest: an example of Bagging
5. Adaboost: an example of Boosting