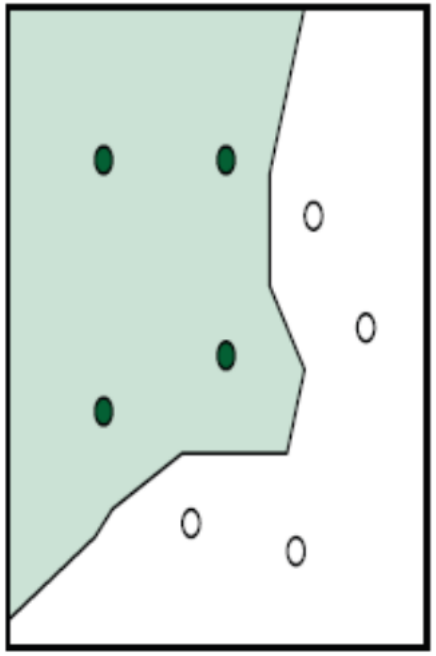
Orient BlackSwan

Models Based on Decision Trees

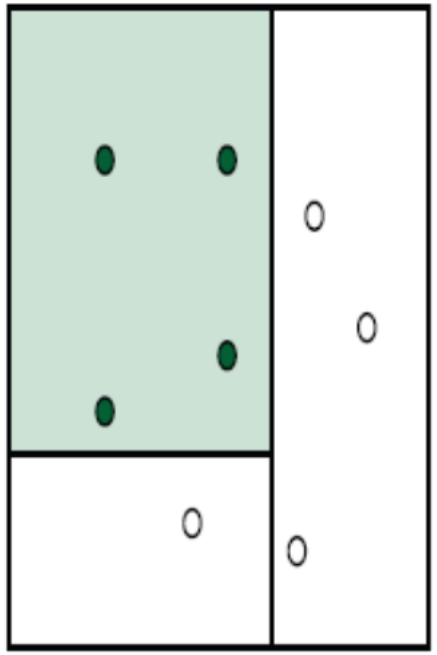
CHAPTER 3: MACHINE LEARNING – THEORY & PRACTICE

Universities Press

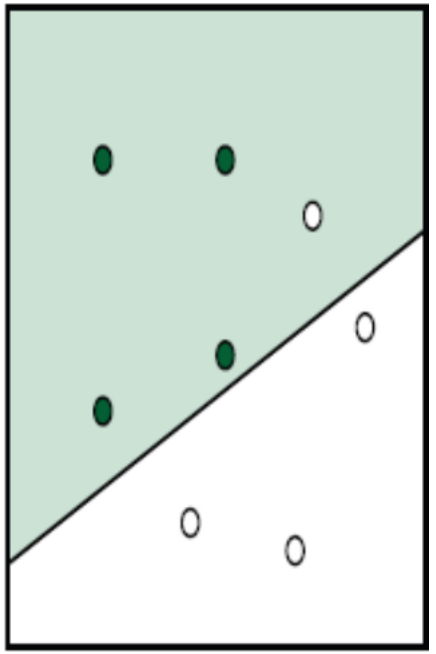
Linear Discriminant Functions



KNNC

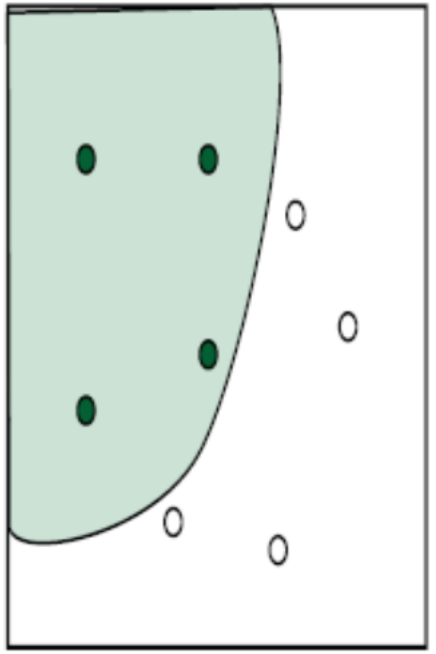


Decision Tree



Linear

$g(X) = W^t X + b$

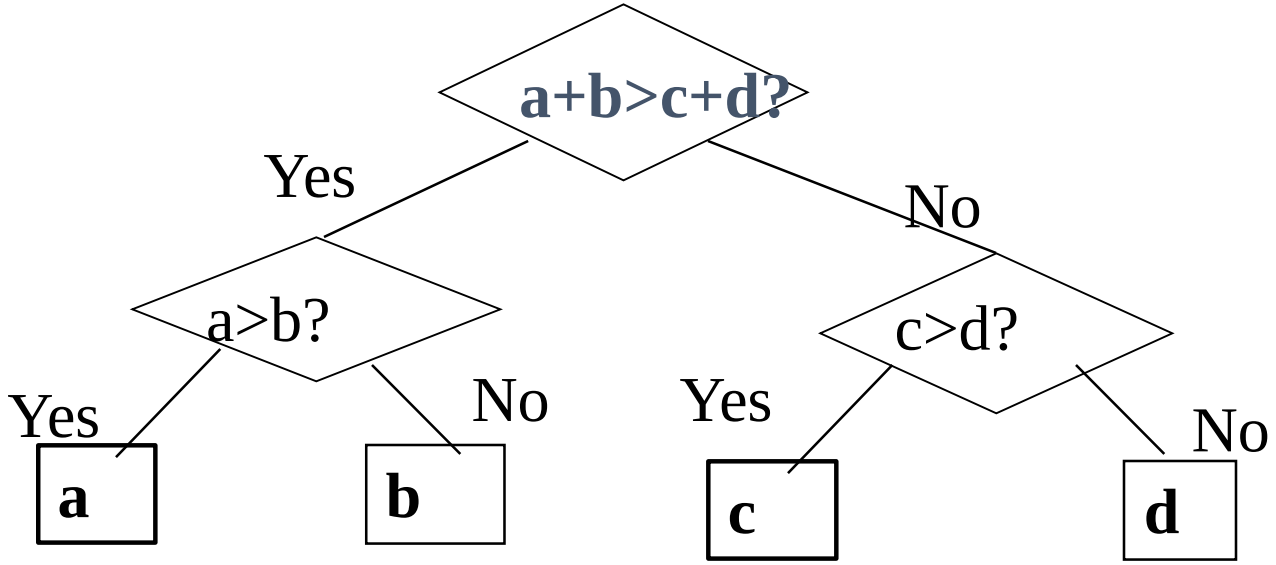


Non-Linear
Neural Networks

Andrew Moore's Tutorial on SVMs

Decision Trees

- Simplest and Easy to understand Abstraction
- Four coins *a,b,c,* and *d:* find the heavier coin



- A tree where each internal node is a decision node.
- It requires two weighings to decide.
- Leaf nodes are associated with outcome.
- Each path from the root to leaf is a rule.

Learning Rules

A learner, inducer, induction algorithm

A method or algorithm used to generalize a pattern from a set of examples.

Name	Balance	Age	Default
Mike	23,000	30	yes
Mary	51,100	40	yes
Bill	68,000	55	no
Jim	74,000	46	no
Dave	23,000	47	yes
Anne	1,00,000	49	no

Learner:

→ Induces a pattern from examples

↓

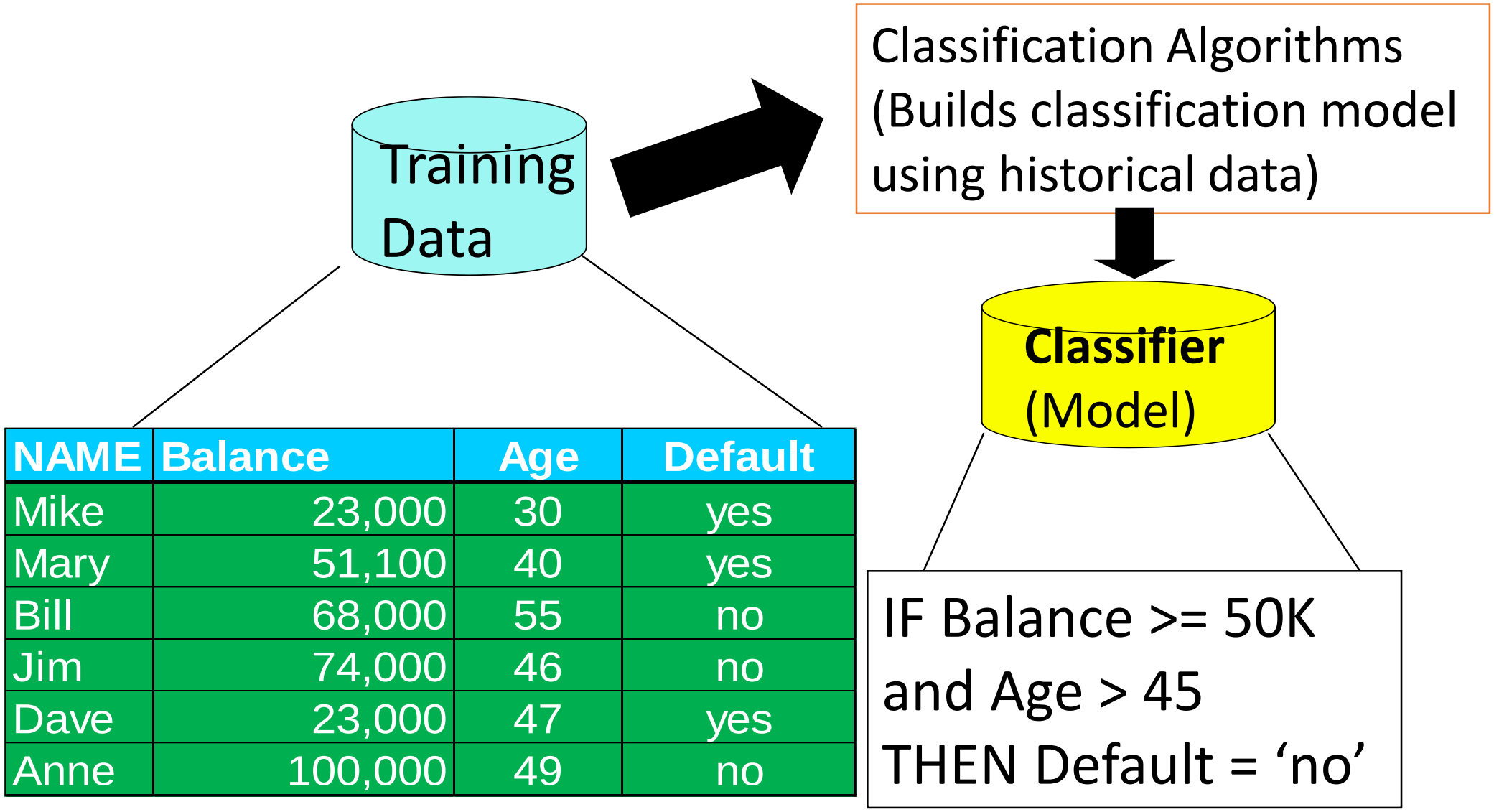
Pattern:

IF **Balance** >= 50K and **Age** > 45

Then **Default** = 'no'

Else **Default** = 'yes'

Classification Example



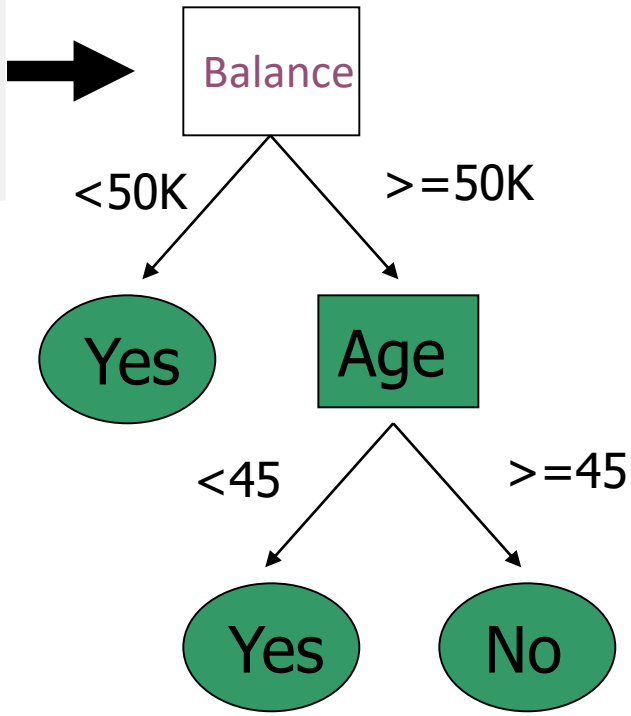
NAME	Balance	Age	Default
Mike	23,000	30	yes
Mary	51,100	40	yes
Bill	68,000	55	no
Jim	74,000	46	no
Dave	23,000	47	yes
Anne	100,000	49	no

Classification: Decision Trees

- Learn a series {IF (condition) Then (class)} rules.

  Example: Credit risk management

New Applicant:
(Mark, Balance=88K,Age= 40)

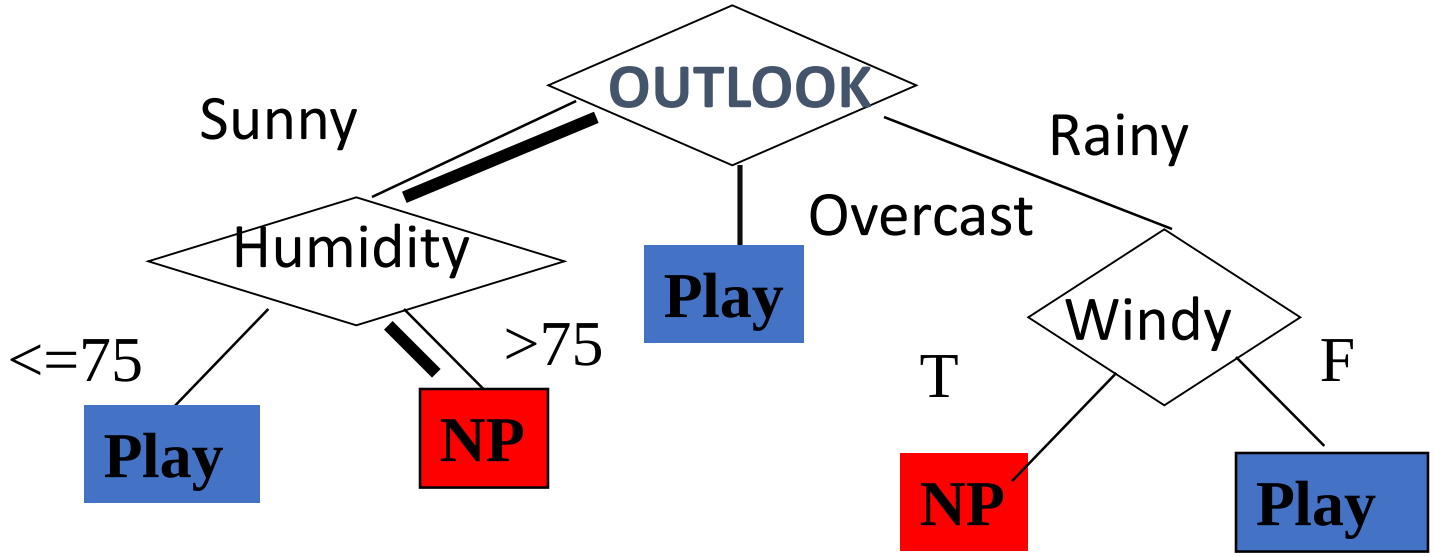


- DTs are very easy to understand
- Good for descriptive modeling too

Example Data Set

OUTLOOK	TEMP(F)	HUMIDTY	WINDY	CLASS
Sunny	79	90	Windy	No Play
Sunny	56	70	Nonwindy	Play
Sunny	60	90	Windy	Noplay
Sunny	79	75	Windy	Play
Overcast	88	88	Nonwindy	Play
Overcast	63	75	Windy	Play
Overcast	88	95	Nonwindy	Play
Rainy	78	60	Nonwindy	Play
Rainy	66	70	Nonwindy	Play
Rainy	68	60	Windy	Noplay

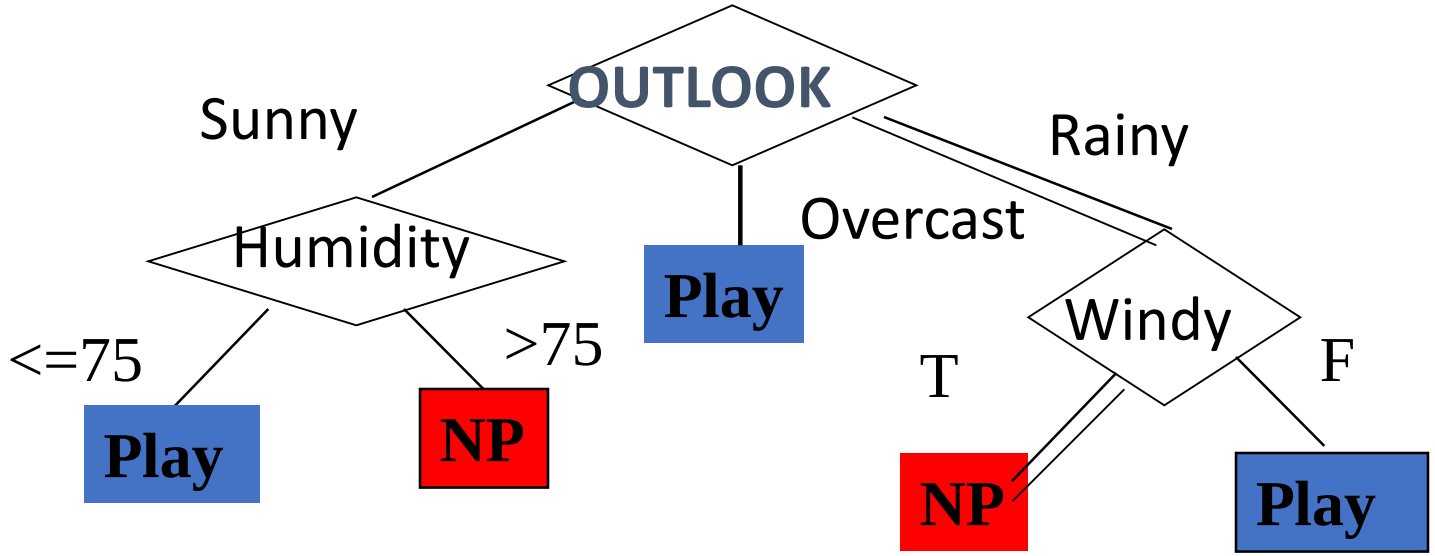
Example Decision Tree



- **Class labels** are associated **with leaf nodes**
- **Root to** leaf represents a rule:

If (outlook = sunny) and (humidity > 75) then noplay

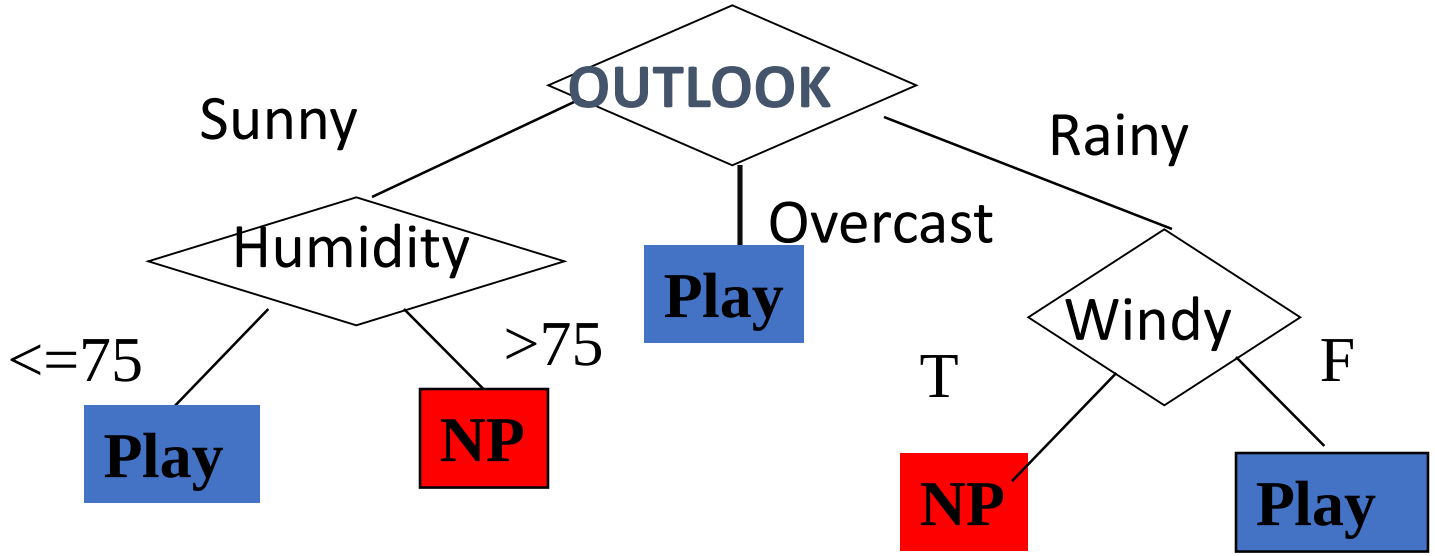
Example Decision Tree



- **Classification involves making decisions at the nodes and moving down the appropriate branch till a leaf**

If (outlook = rainy), (temp = 70), (humidity = 65) and (Windy = True) then Class = ? ---Noplay (NP)

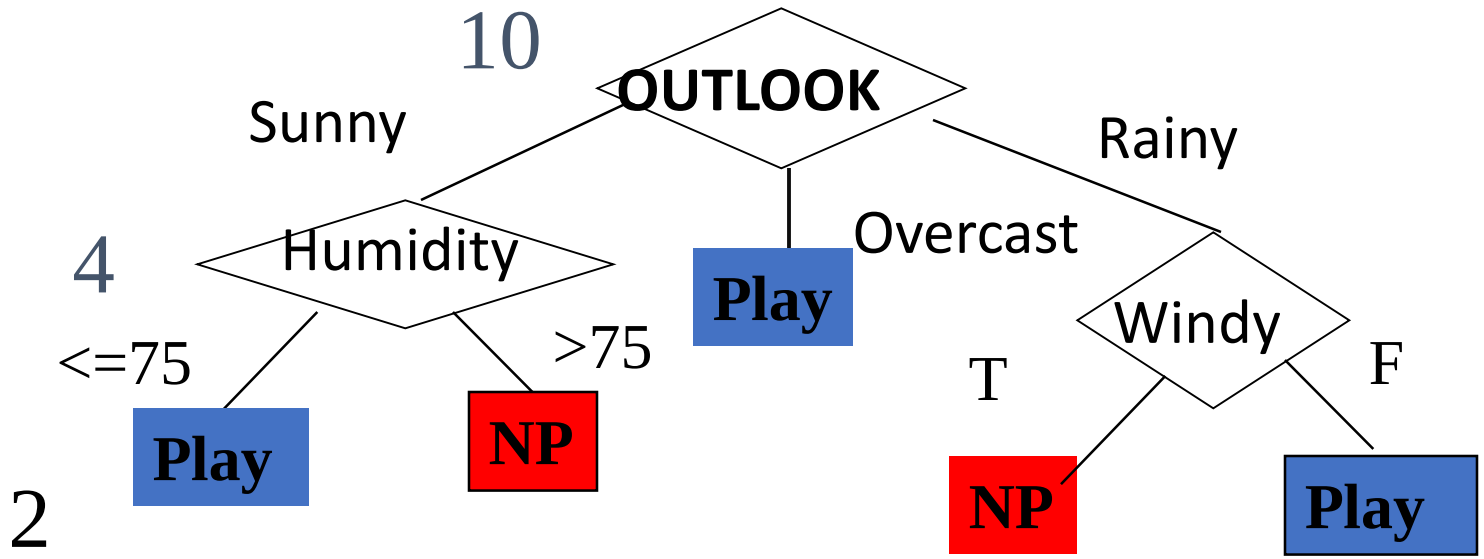
Example Decision Tree



- **Irrelevant features are eliminated : For example Temp**
- It can deal with both Numerical and Categorical

Features: Windy, Outlook – Categorical Temp and Humidity - Numerical

Example Decision Tree



- **The tree could be binary or nonbinary**
- The rules are simple and easy to understand
- A set of patterns is associated with each node

Construction of Decision Trees

There are different ways to construct trees from data.
We will concentrate on the top-down, greedy search approach:

Basic idea:

1. Choose the best attribute a* to place at the root of the tree.

2. Separate training set D into subsets {D1, D2, .., Dk} where each subset Di contains examples having the same value for a*

3. Recursively apply the algorithm on each new subset until examples have the same class or there are few of them.

Splitting Functions

- Type of Test:
  - Axis-parallel test
  - Test based on linear combination of features
  - Test based on nonlinear combination of features

- Which attribute is the best to split the data?

Entropy associated with a random variable X is defined as

$H(X) = - \Sigma\ pi \log pi$, where the log is in base 2.

Splitting Based on Entropy

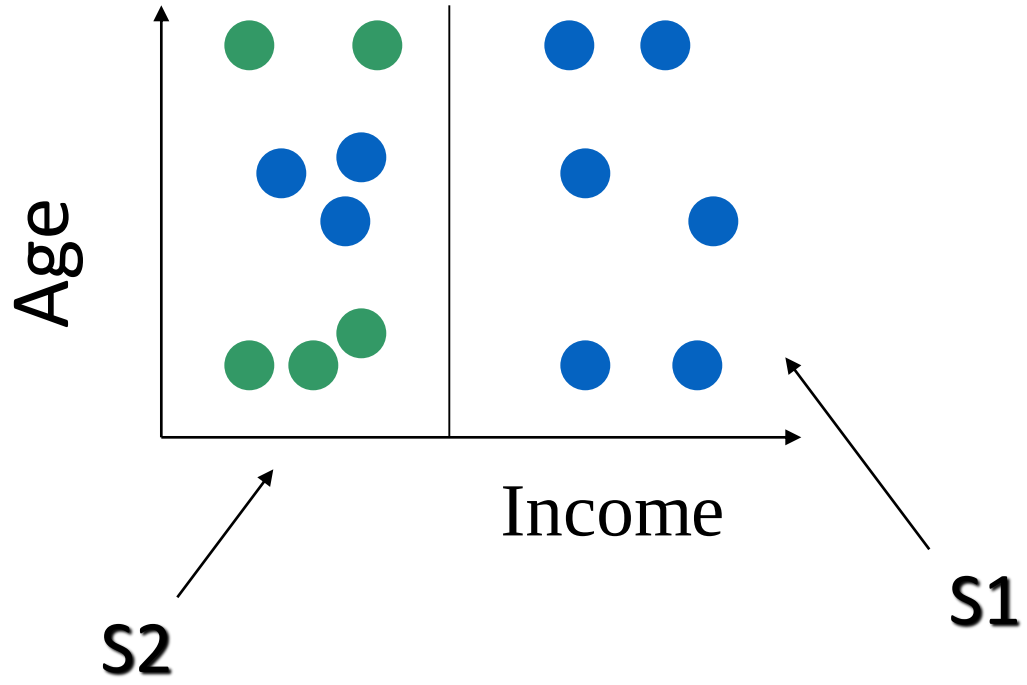
Income divides the sample in to:

S1 = { 6+, 0-}

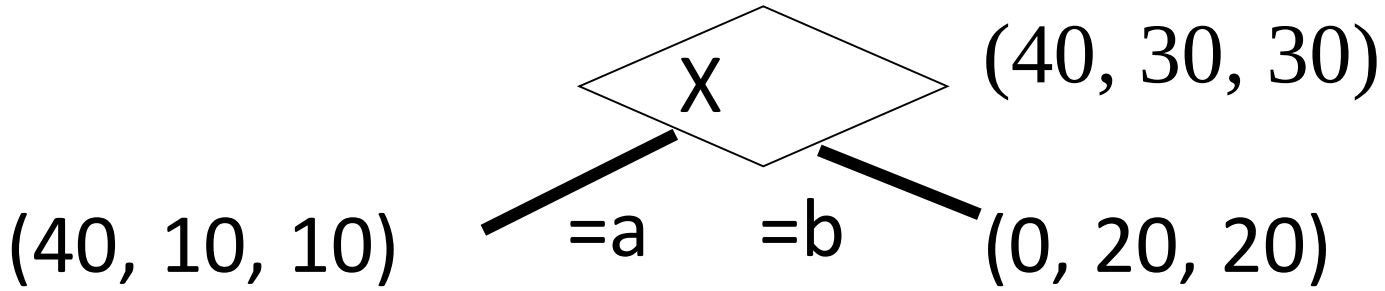
S2 = { 3+, 5-}

H(S1) = 0

H(S2) = -(3/8)log(3/8)
-(5/8)log(5/8)

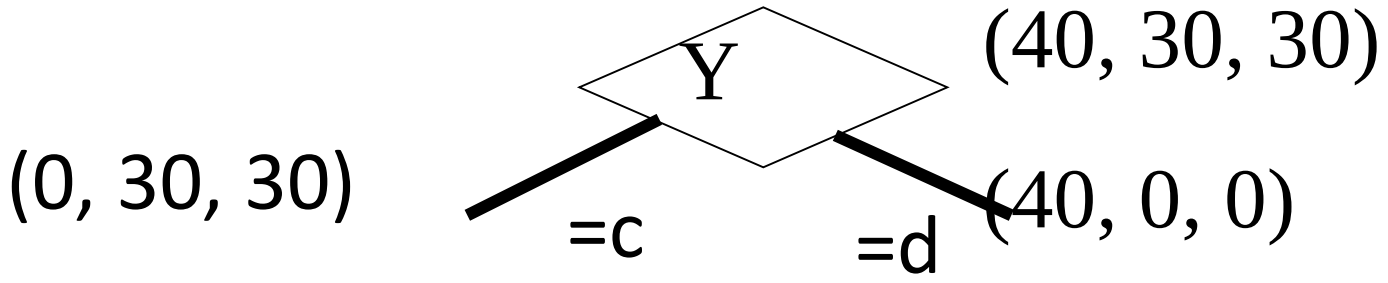


Which Attribute to Choose?



- The entropy impurity at the split is
  -0.4log 0.4 – 0.3log0.3 – 0.3log0.3 = 1.38
- The entropy of the left branch is
  -0.66log0.66 – 0.17log0.17 – 0.17log0.17 = 1.25
- The entropy of the right branch is
  -0.5log0.5 – 0.5log0.5 = 1.0
- The drop in impurity is therefore
  1.38 – 0.6*1.25 – 0.4*1.0 = 0.23

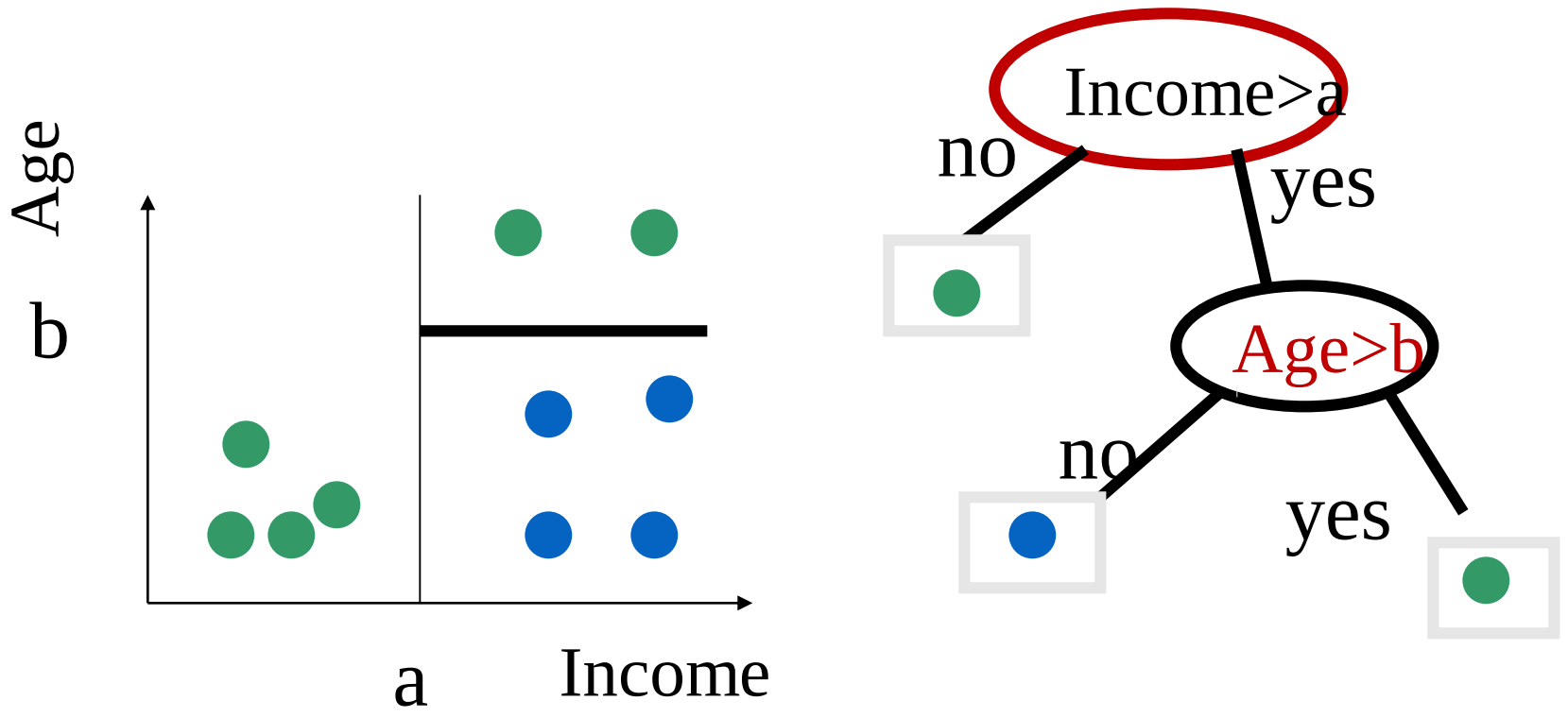
Which Attribute to Choose?



- The entropy impurity at the split is
  -0.4log 0.4 – 0.3log0.3 – 0.3log0.3 = 1.38
- The entropy of the left branch is
  -0.5log0.5 – 0.5log0.5 = 1.0
- The entropy of the right branch is 0.0

- The drop in impurity is therefore
  1.38 – 0.6*1.0 – 0.4*0.0 = 0.78

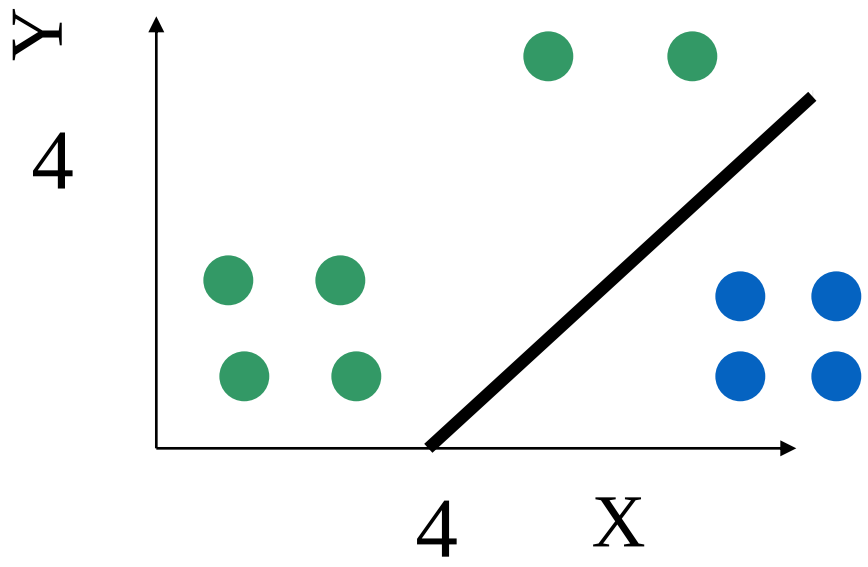
Axis Parallel Split

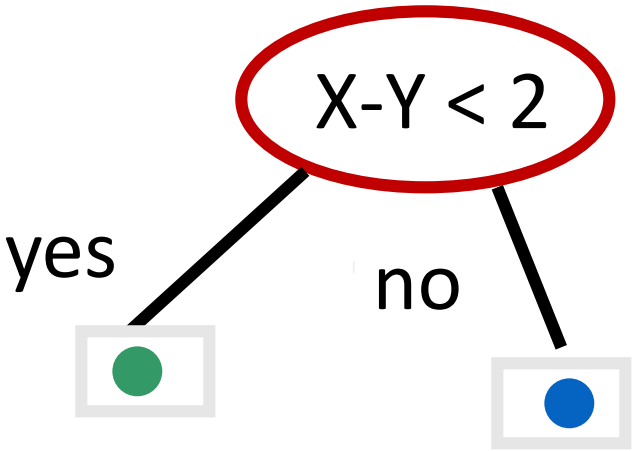


Oblique Split

(1,1), (2,1), (1,2), (2,2), (6,7), (7,7)

(6,1), (7,1), (6,2), (7,2)





Summary

•Pros

+ Reasonable training time

+ Fast application

+ Easy to interpret

+ Easy to implement

+ Can handle both numerical and categorical features

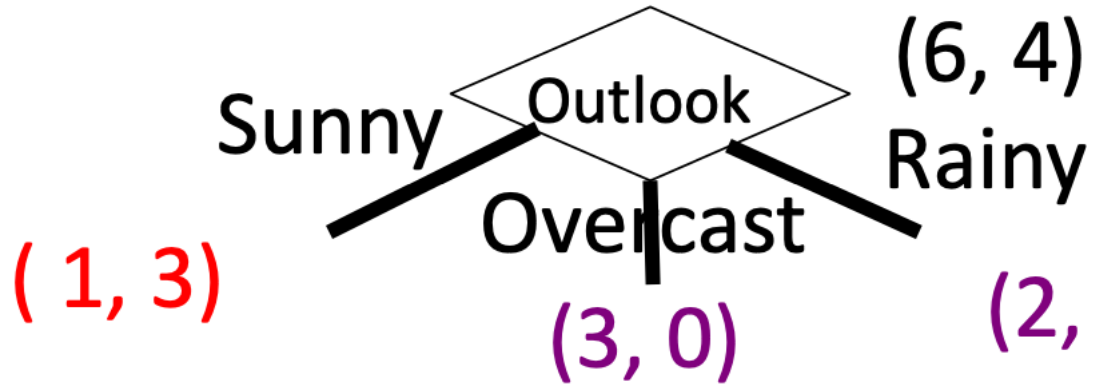
•Cons

- Cannot handle complicated relationship between features

- simple decision boundaries

- problems with lots of missing data

Calculating Information Gain for OUTLOOK



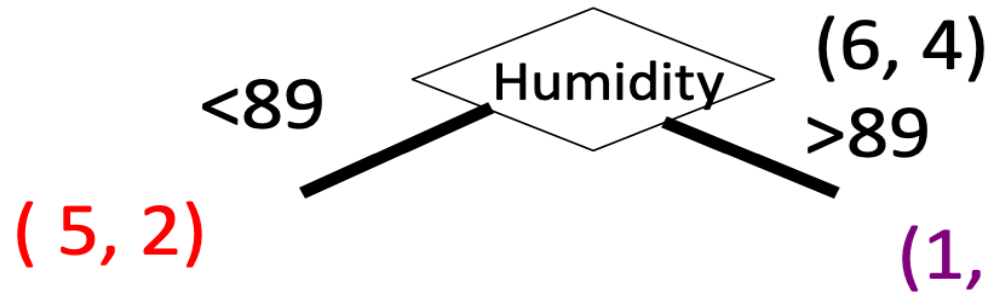
- Sunny : 4
- Overcast:3
- Rainy: 3

- Entropy at the split is -0.4 log 0.4 – 0.6 log 0.6 = 0.29

- Entropy at the left child is -0.25 log 0.25 – 0.75 log 0.75 = 0.244
- Entropy at the right child is 0.278; for the Middle node it is 0
- The drop in impurity is therefore
  - 0.29 - 0.4*0.244 - 0.3*0.0 - 0.3(0.278) = 0.11

- This is also called Information Gain

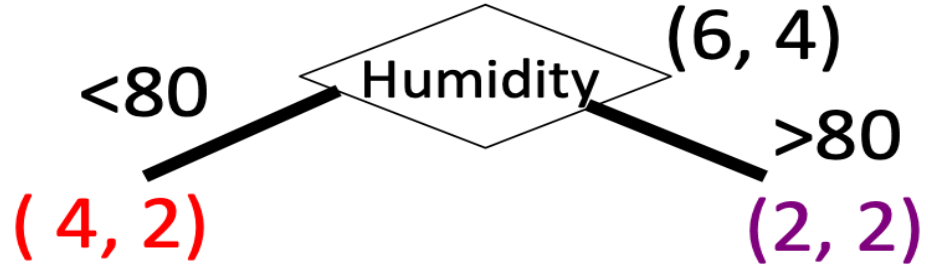
Calculating Information Gain for HUMIDITY



60- 2 (1,1)
70- 2 (1,1)
75- 2 (2,0)
88- 1 (1,0)

90- 2 (0,2)
95- 1 (1,0)

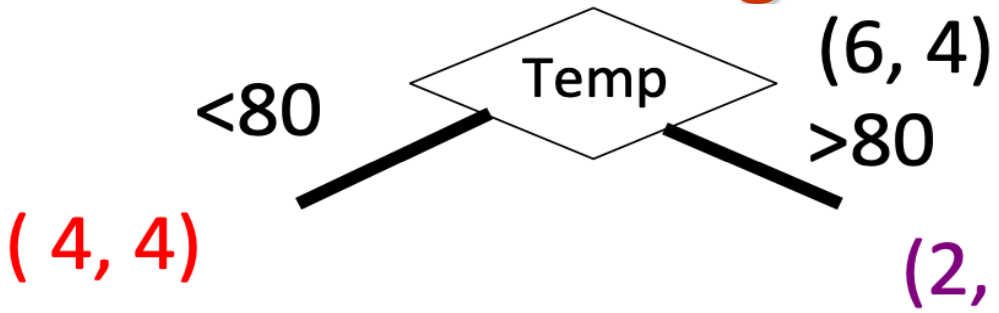
- Entropy at the split is -0.4log 0.4 – 0.6log0.6 =0.29
- The drop in impurity is 0.29- 0.7[-0.7 log 0.7-0.3 log 0.3] - 0.3[-0.33 log 0.33 -0.66 log 0.66]= 0.02 (Information Gain)



60- 2 (1,1)
70- 2 (1,1)
75- 2 (2,0)
88- 1 (1,0)
90- 2 (0,2)
95- 1 (1,0)

- Entropy at the split is -0.4log 0.4 – 0.6log0.6 =0.29
- The drop in impurity is 0.29- 0.6[-0.33 log 0.33 -0.66 log 0.66] - 0.4[-0.5 log 0.5 – 0.5 log 0.5]= 0.003 (Information Gain)

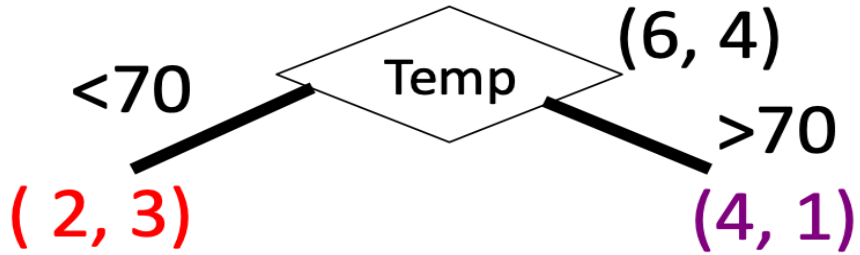
Calculating Information Gain for Temp



- 56- 1 (0,1)
- 60- 1 (0,1)
- 63- 1 (1,0)
- 66- 1 (1,0)
- 68- 1 (0,1)
- 78- 1 (1,0)
- 79- 2 (1,1)

- 88- 2 (2,0)

- Entropy at the split is -0.4log 0.4 – 0.6log0.6 =0.29
- The drop in impurity is 0.29- 0.8[-0.5 log 0.5-0.5 log 0.5] – 0.0 = 0.05 (Information Gain)

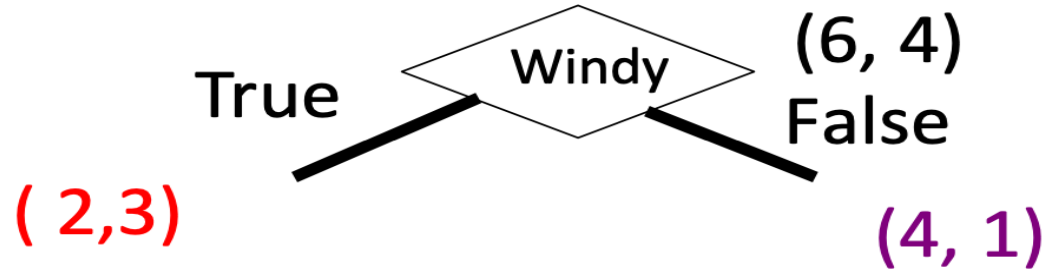


- 56- 1 (0,1)
- 60- 1 (0,1)
- 63- 1 (1,0)
- 66- 1 (1,0)
- 68- 1 (0,1)

- 78- 1 (1,0)
- 79- 2 (1,1)
- 88- 2 (2,0)

- Entropy at the split is -0.4log 0.4 – 0.6log0.6 =0.29
- The drop in impurity is 0.29- 0.5[-0.6 log 0.6 -0.4 log 0.4] -0.5[-0.8 log 0.8 – 0.2 log 0.2]= 0.035 (Information Gain)

Calculating Information Gain for Windy

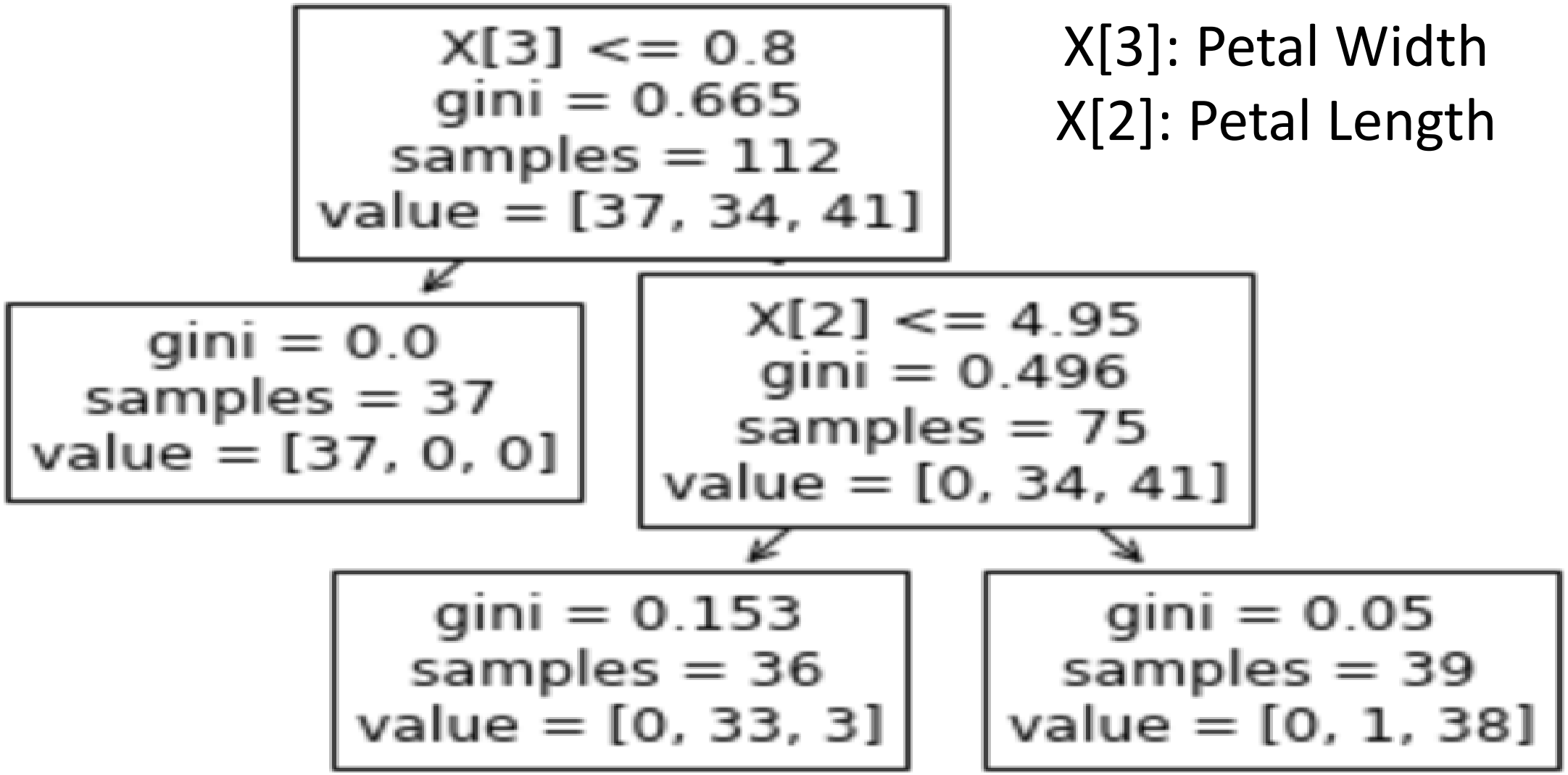


- Entropy at the split is -0.4log 0.4 – 0.6log0.6 =0.29
- The drop in impurity is 0.29- 0.5[-0.6 log 0.6 -0.4 log 0.4] -0.5[-0.8 log 0.8 – 0.2 log 0.2]= 0.035 (Information Gain)

- So, Outlook is the best feature in terms of the largest value of Information Gain, that is 0.11.

- That is why Outlook was used at the root node of the Decision Tree shown earlier against this data.

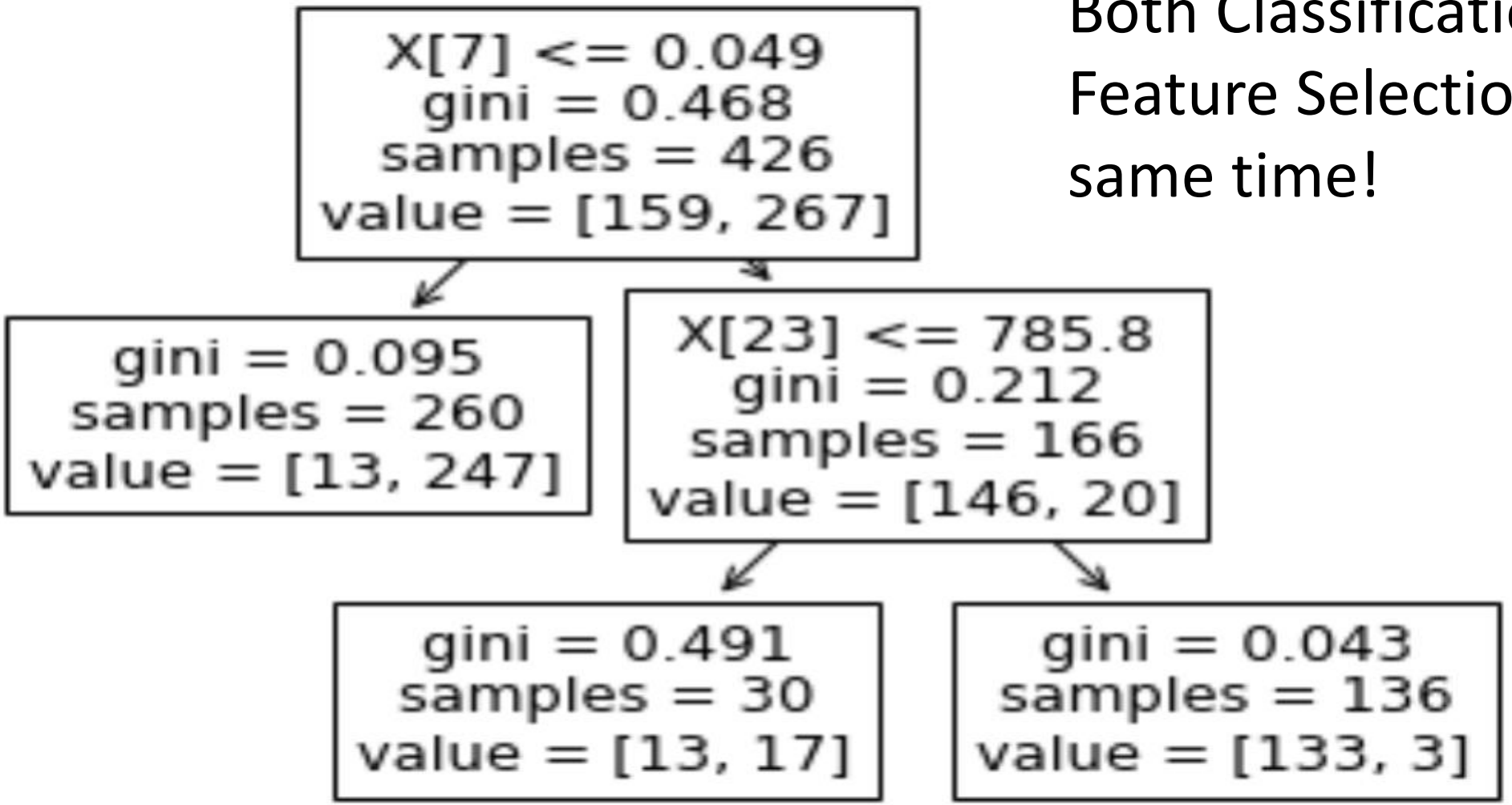
Decision-Tree: Iris Data



X[3]: Petal Width

X[2]: Petal Length

Decision-Tree: Breast Cancer Data

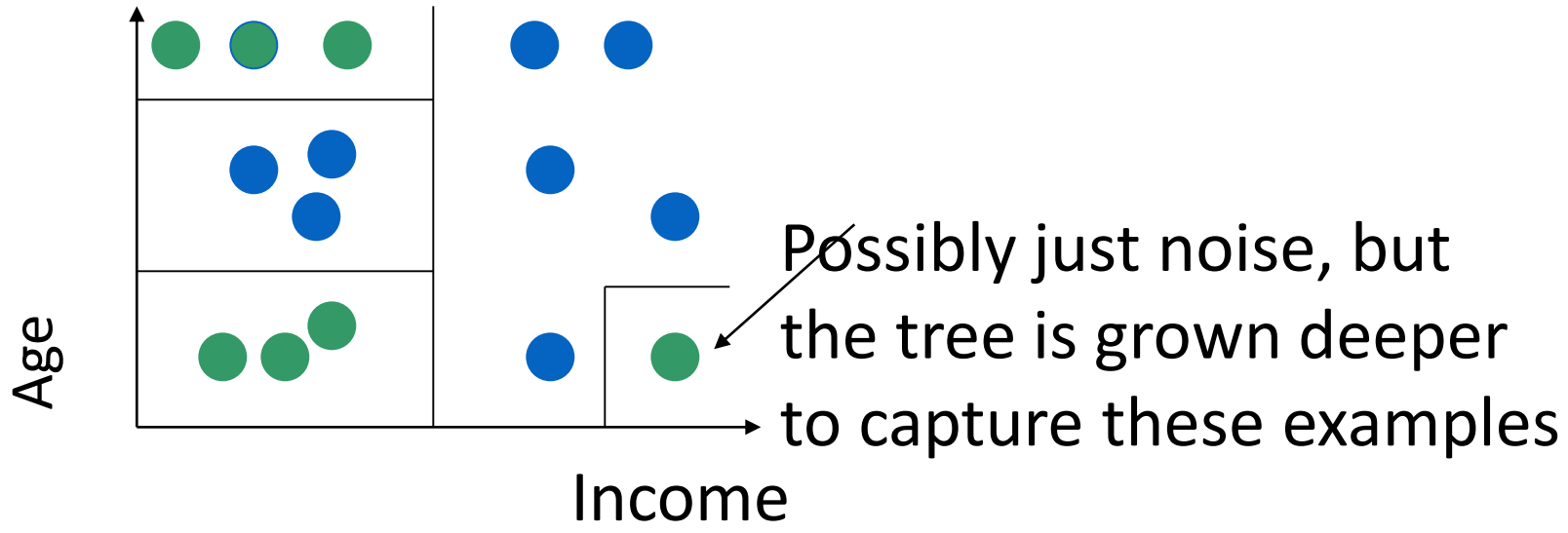


Both Classification and Feature Selection at the same time!

- X = X[:,[7,23]]: ACC = 0.930069930069301 (KNNC with K= 11)
- X = X[:,[3,7,23]]: ACC = 0.958041958041958

Depth of the Tree: Overfitting

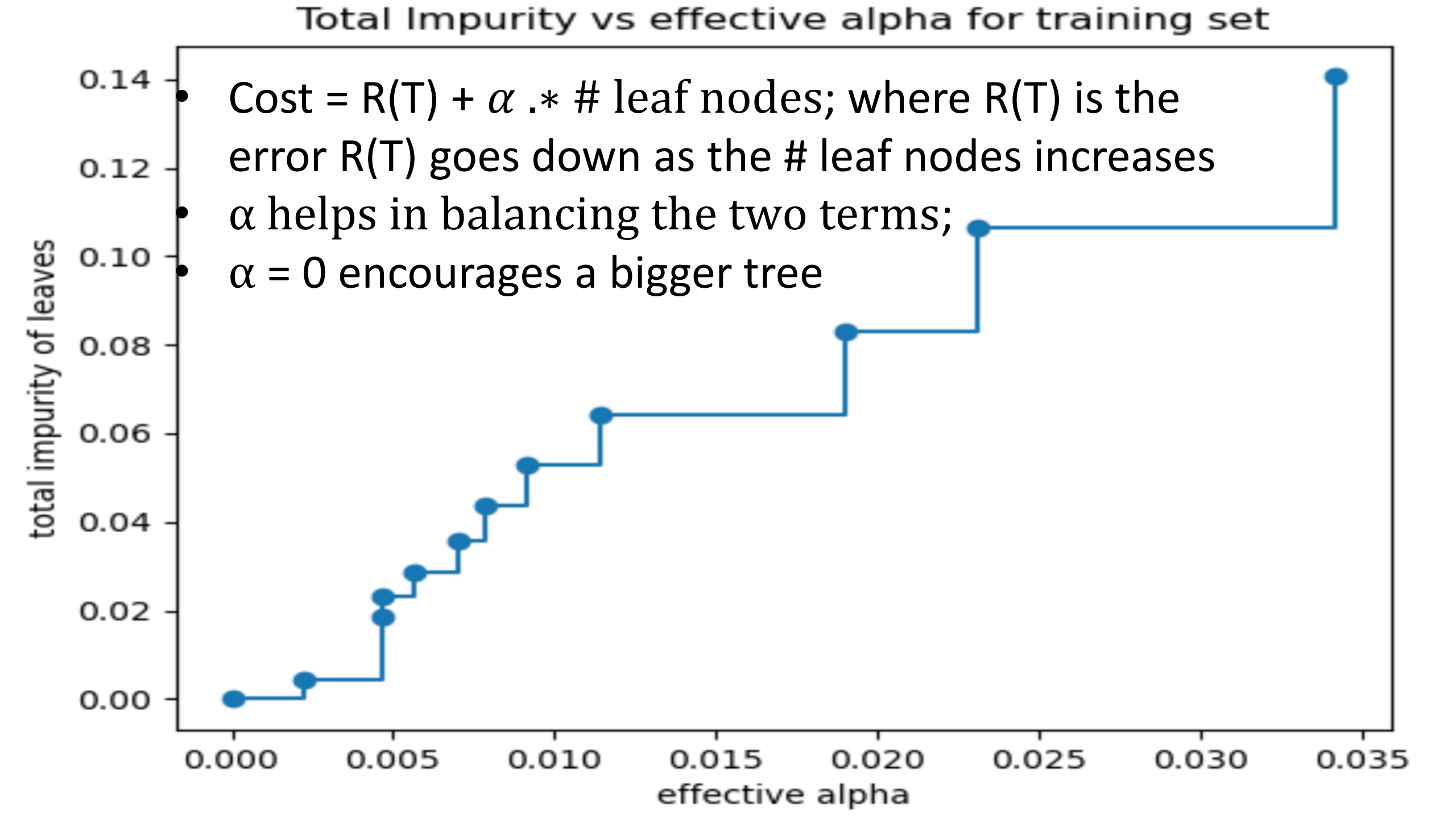
A tree *overfits* the data if we let it grow deep enough so that it begins to capture "aberrations" in the data that harm the predictive power on unseen examples:



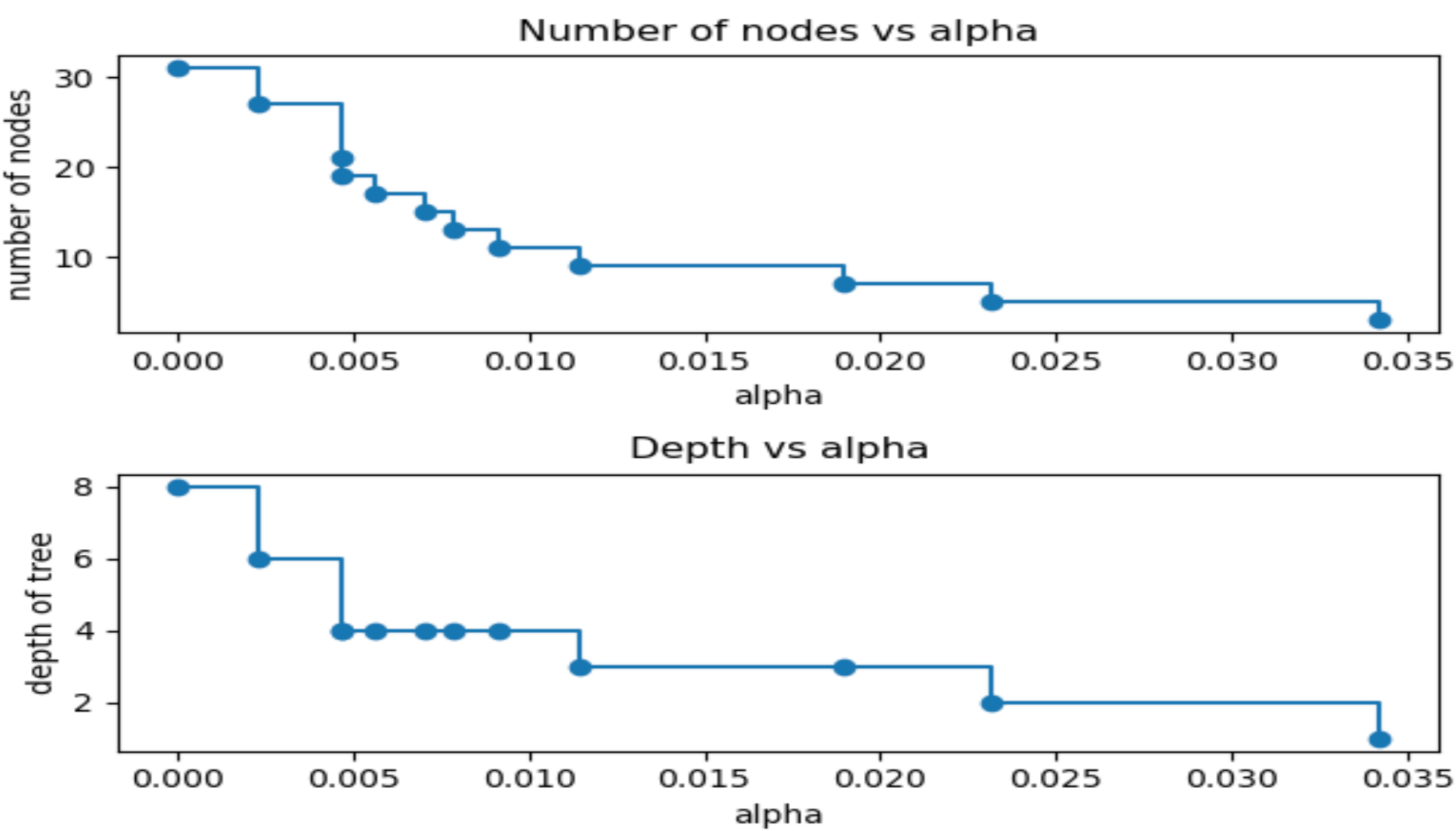
When to Stop Splitting?

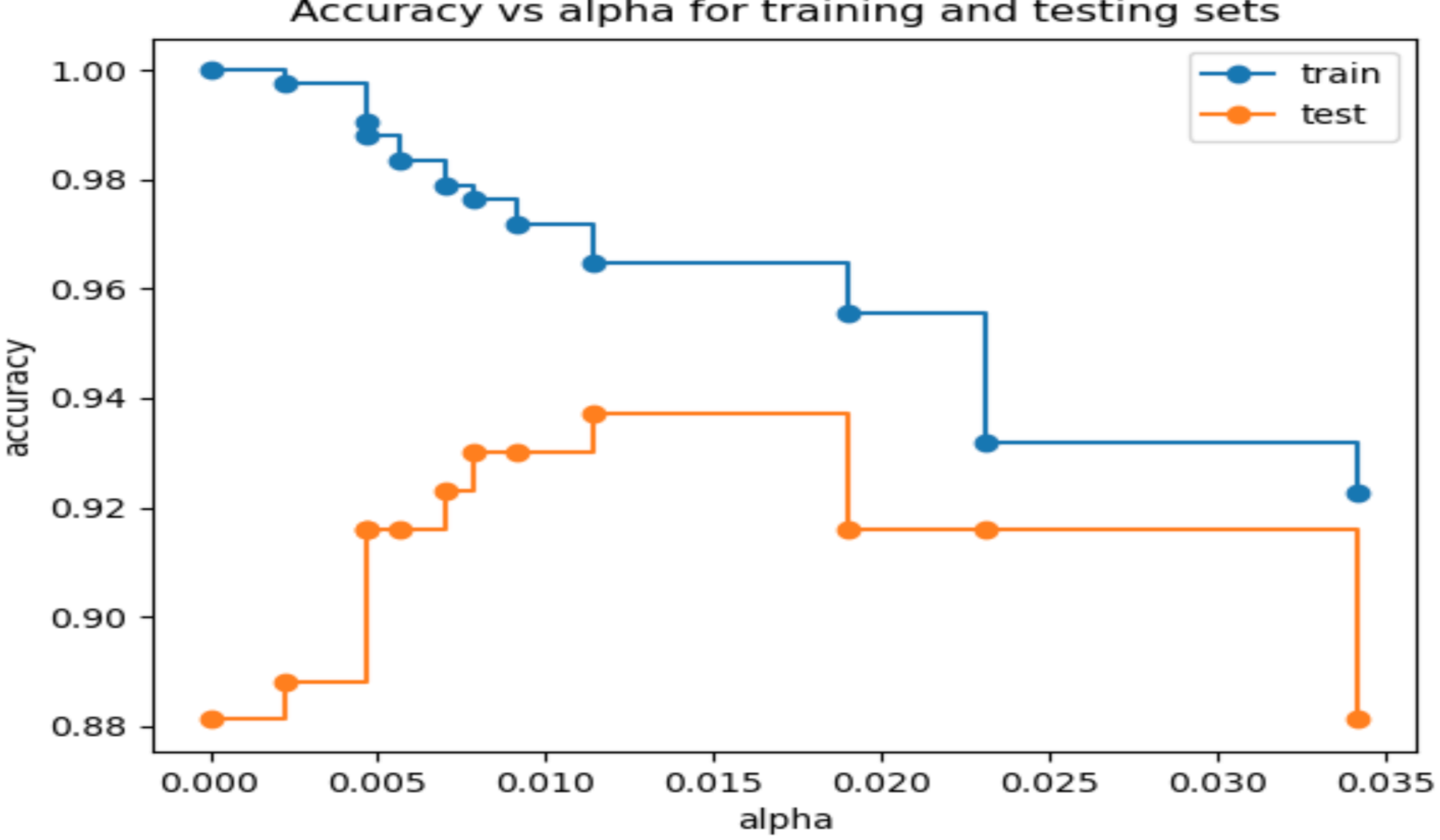
- Stop **when reduction in impurity is small**. Stop splitting when all the nodes are **pure or almost pure**.
- Training sample threshold *k:* **stop cutting when the number of samples at a node is $\leq$ k**. (alternatively as percentage of training sample size).
- Terminate the splitting process using a **global criterion function.**
- *A* **possible criterion function** is of the form:

α * **Size of the tree + sum of the impurities of the leaf nodes.**



- Cost = R(T) + α .$*$ # leaf nodes; where R(T) is the error R(T) goes down as the # leaf nodes increases
- α helps in balancing the two terms;
- α = 0 encourages a bigger tree


Number of nodes vs alpha
number of nodes
30
20
10
0.000
0.005
0.010
0.015
0.020
0.025
0.030
0.035
alpha
Depth vs alpha
depth of tree
8
6
4
2
0.000
0.005
0.010
0.015
0.020
0.025
0.030
0.035
alpha


Accuracy vs alpha for training and testing sets
train
test
accuracy
alpha
1.00
0.98
0.96
0.94
0.92
0.90
0.88
0.000
0.005
0.010
0.015
0.020
0.025
0.030
0.035

Overfitting the Data

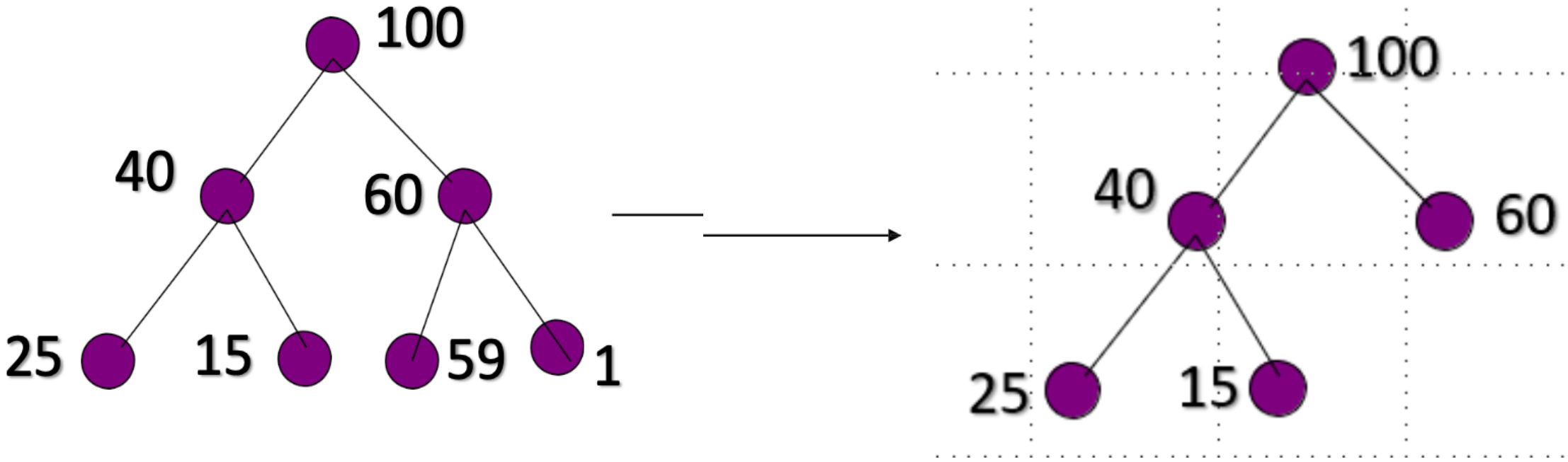
Overfitting: The tree can grow in size. For example, having a **unique path** in the tree for every training pattern .

Solutions:

1. **Grow the tree** until the algorithm stops even if overfitting problem shows up. **Then prune** the tree as a post-processing step.

2. **Stop growing the tree** as soon as the **size goes beyond some specified limit.**

Decision Tree Pruning



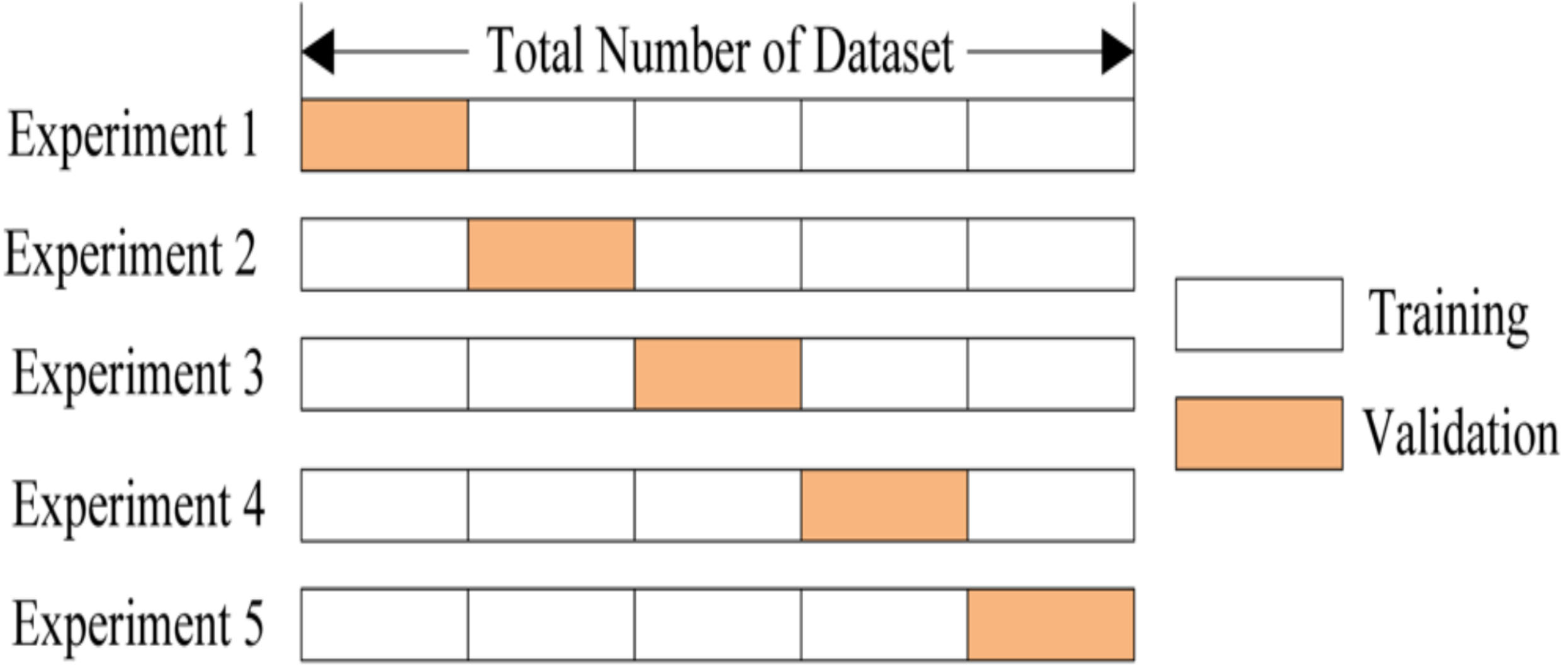
1) Grow the tree to learn the training data

2) Prune tree to avoid overfitting the data

When to Stop Splitting?

Use Cross- Validation: A part of the data is kept aside for validation.

Stop splitting when the best results are obtained on the validation data.



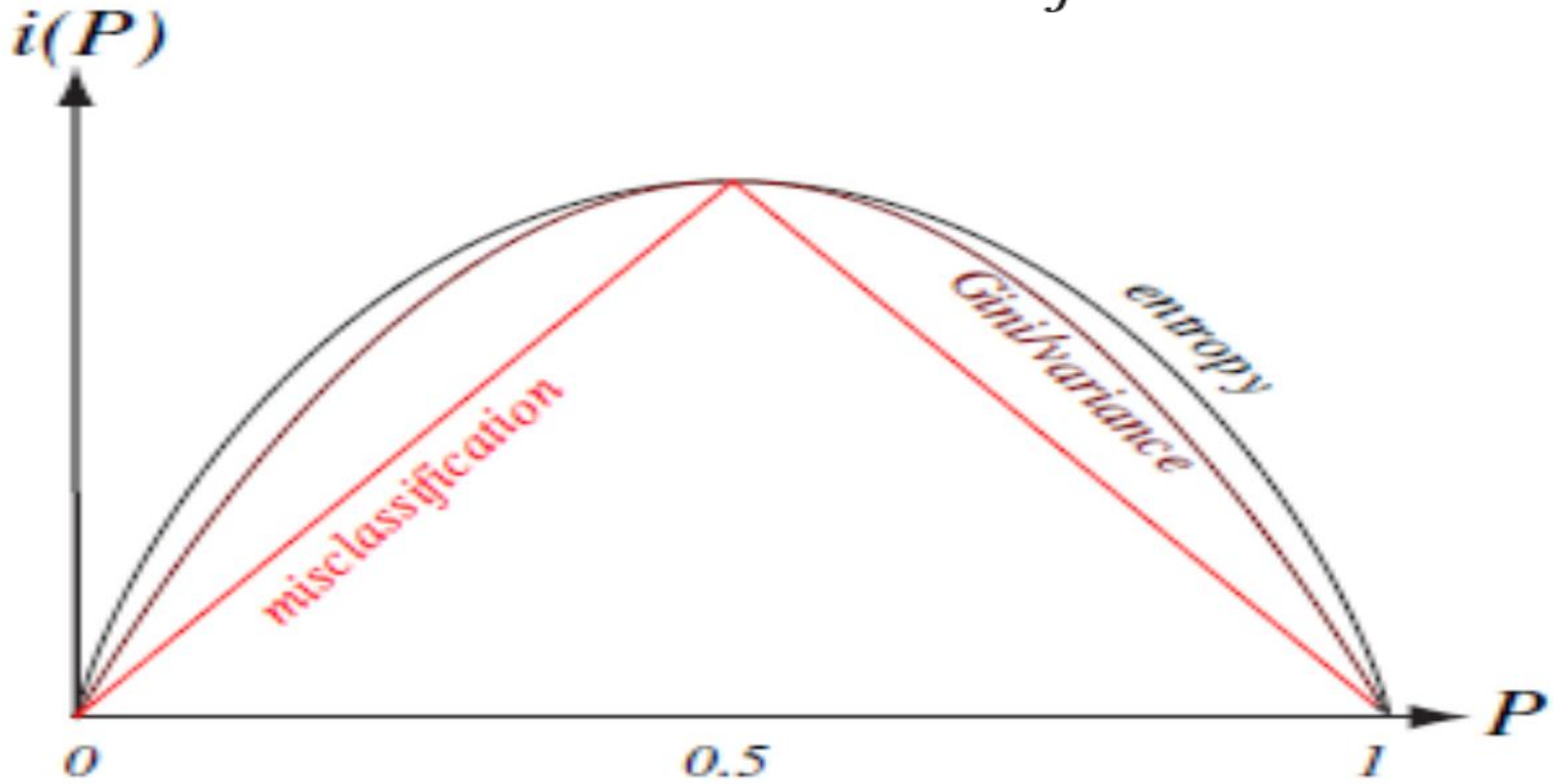
Impurity Functions

- Entropy
- Gini Index
- Misclassification

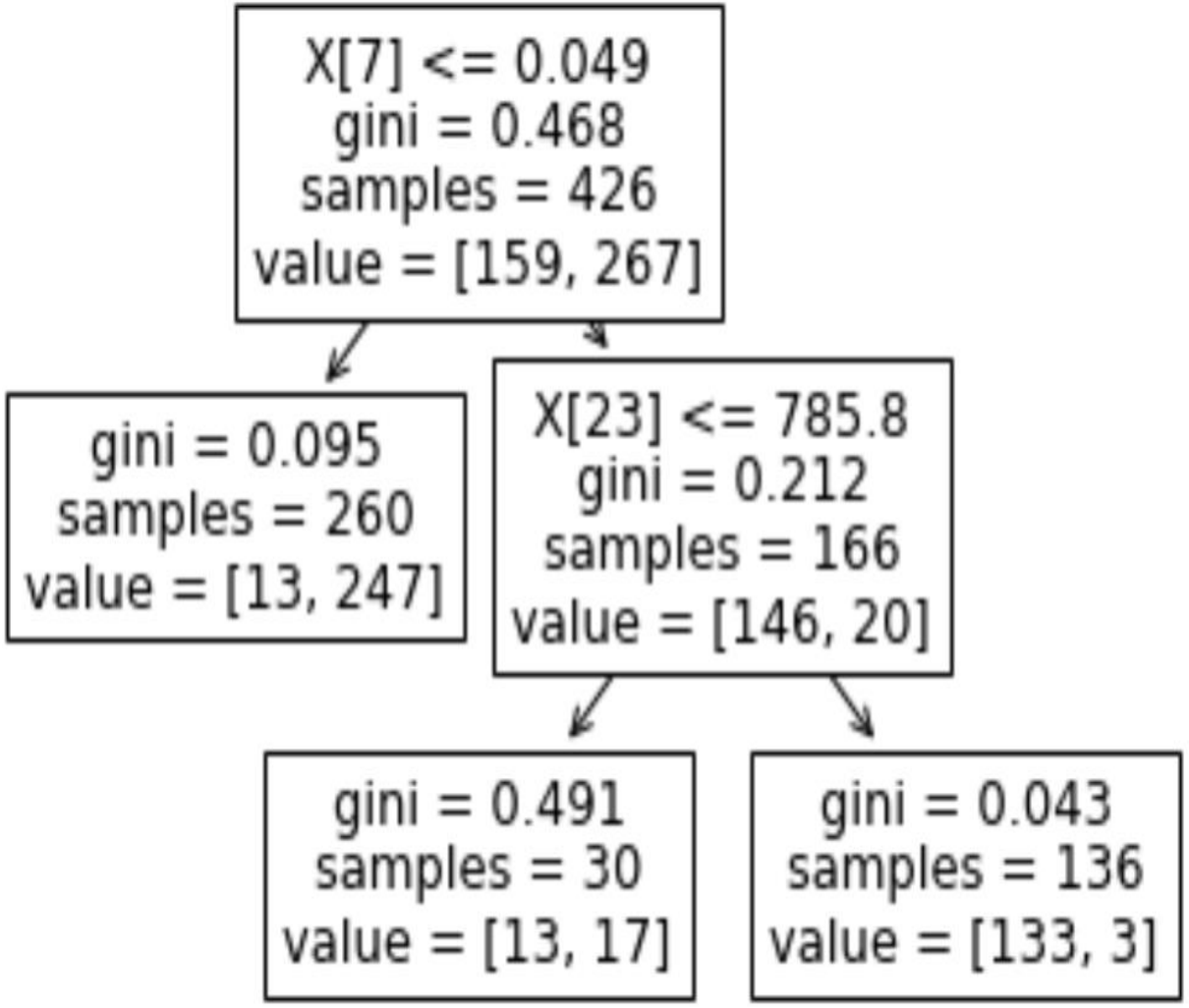
$$i(t) = -\sum_{j=1}^{c} P_j \log P_j$$

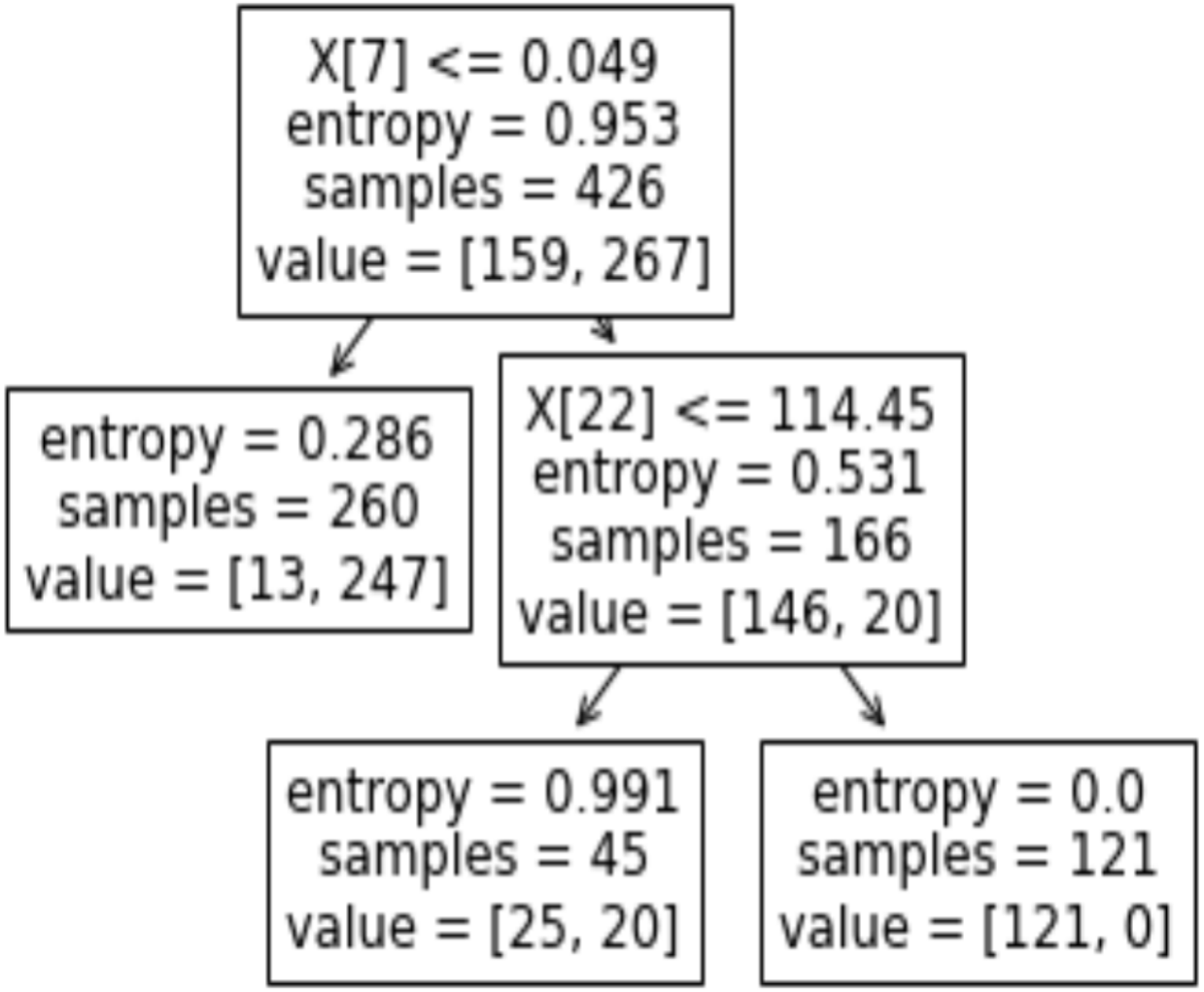
$$i(t) = 1 - \sum_{j=1}^{c} P_j^2$$

$$i(t) = 1 - \max_j P(\omega_j)$$



Impurity Functions: Gini Index and Entropy





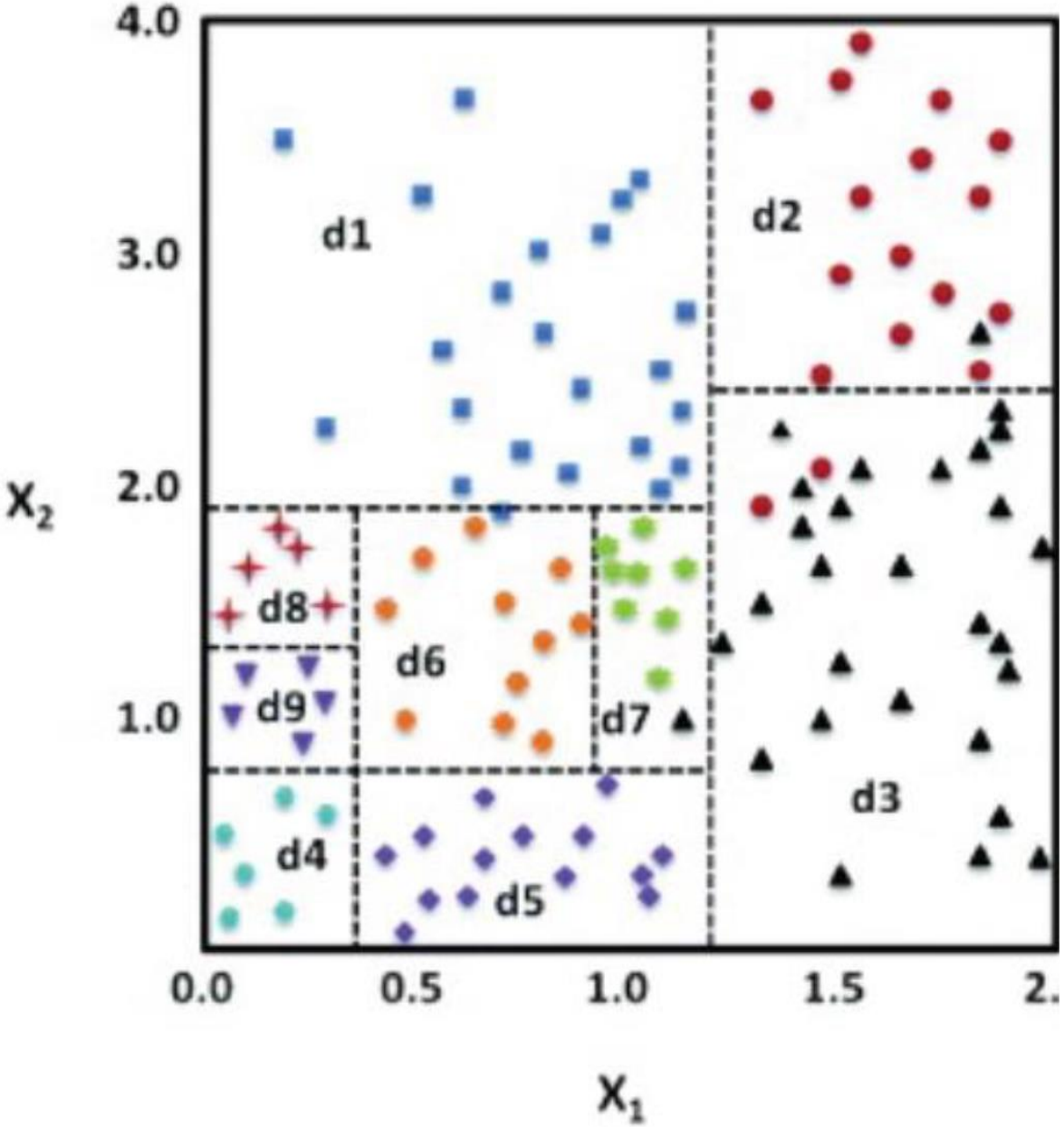
- X = X[:,[7,23]]: ACC = 0.9301 (KNNC)
- X = X[:,[3,7,23]]: ACC = 0.9580 (K=11)

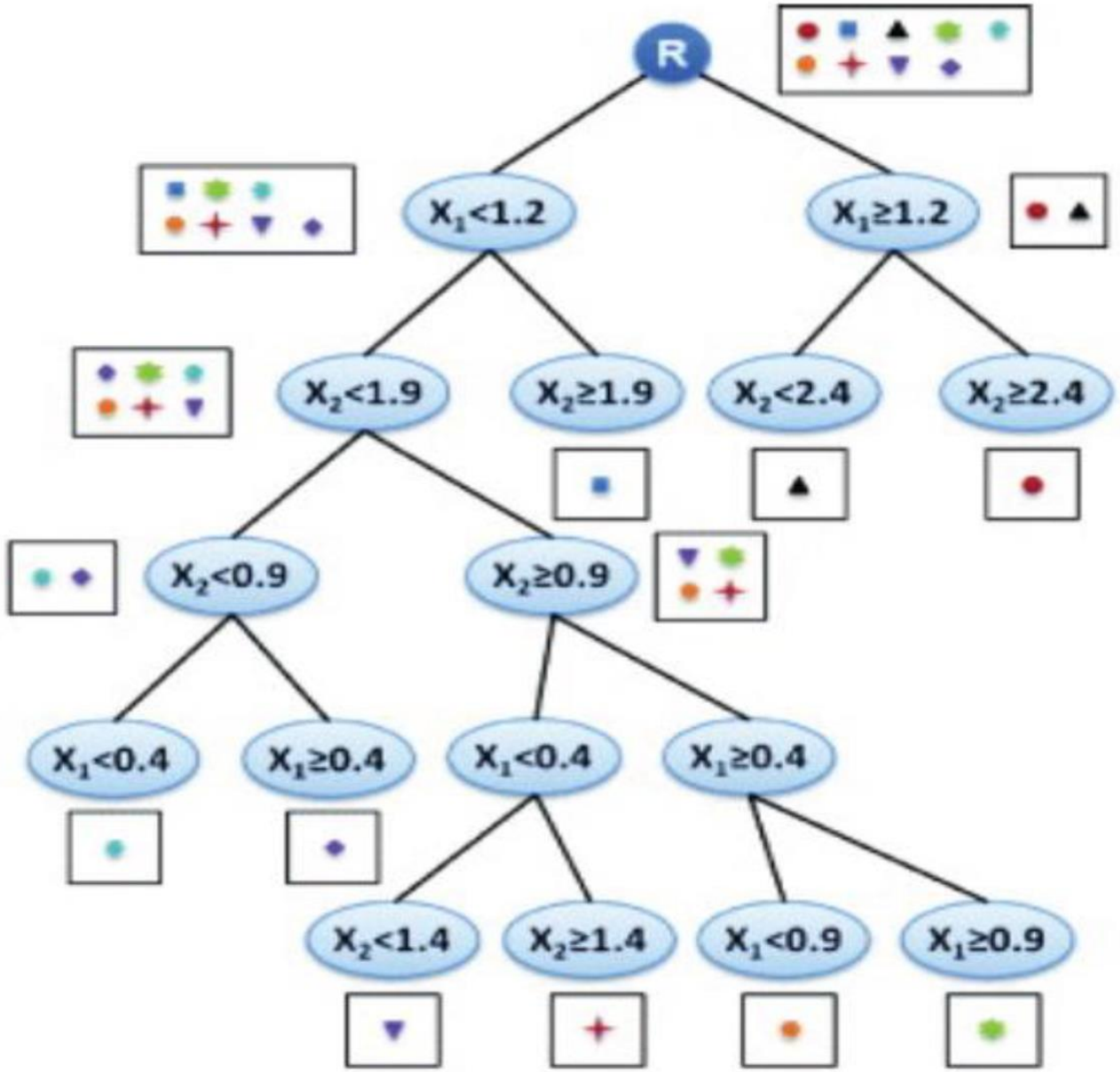
- X = X[:,[7,22]]: ACC = 0.9230 (KNNC)
- X = X[:,[1,7,22]]: ACC = 0.9650 (K=7)

Variance Impurity

- An impurity suitable for a 2-class problem is **i(n) = P(C_1) P(C_2)**
- It is 0 at a node when the node has patterns of **only C_1 or C_2**.
- This may be called **Variance Impurity.** In a 2-class case, we have P(C_2) = 1 – P(C_1).
- It cannot be a polynomial of degree 1 in P(C_1), of the form a.P(C_1) + b
- Because we need i(P(C_1)=0) = i(P(C_1) = 1) = 0 as the condition to be satisfied
- P(C_1) = 0 gives b=0 and P(C_1) = 1 gives a+b = 0→a = 0
- If we consider the form **a P(C_1)^2 + b P(C_1) + c** and the conditions, we get from **P(C_1) = 0, c = 0**, and from **P(C_1) = 1,** we get **a + b = 0 → a = -b**
- So, i(P(C_1)) = **a P(C_1)^2 + b P(C_1) + c = b P(C_1) (1- P(C_1)) = b P(C_1) P(C_2) → i ∝ P(C_1) P(C_2)**
- **E[i] = 1 P(C_1) + 0 P(C_2) = P(C_1)** if i is seen as a binary RV with 1 for C_1, 0 for C_2.
- We know that Var[X] = E[X^2] - ($E[X]$)^2 for a **RV X,** so
- **Var[i] = 1^2 P (C_1) + 0^2 P (C_2) - P (C_1)^2 = P(C_1) (1- P(C_1)) = P(C_1) P(C_2)**

Regions and Classes

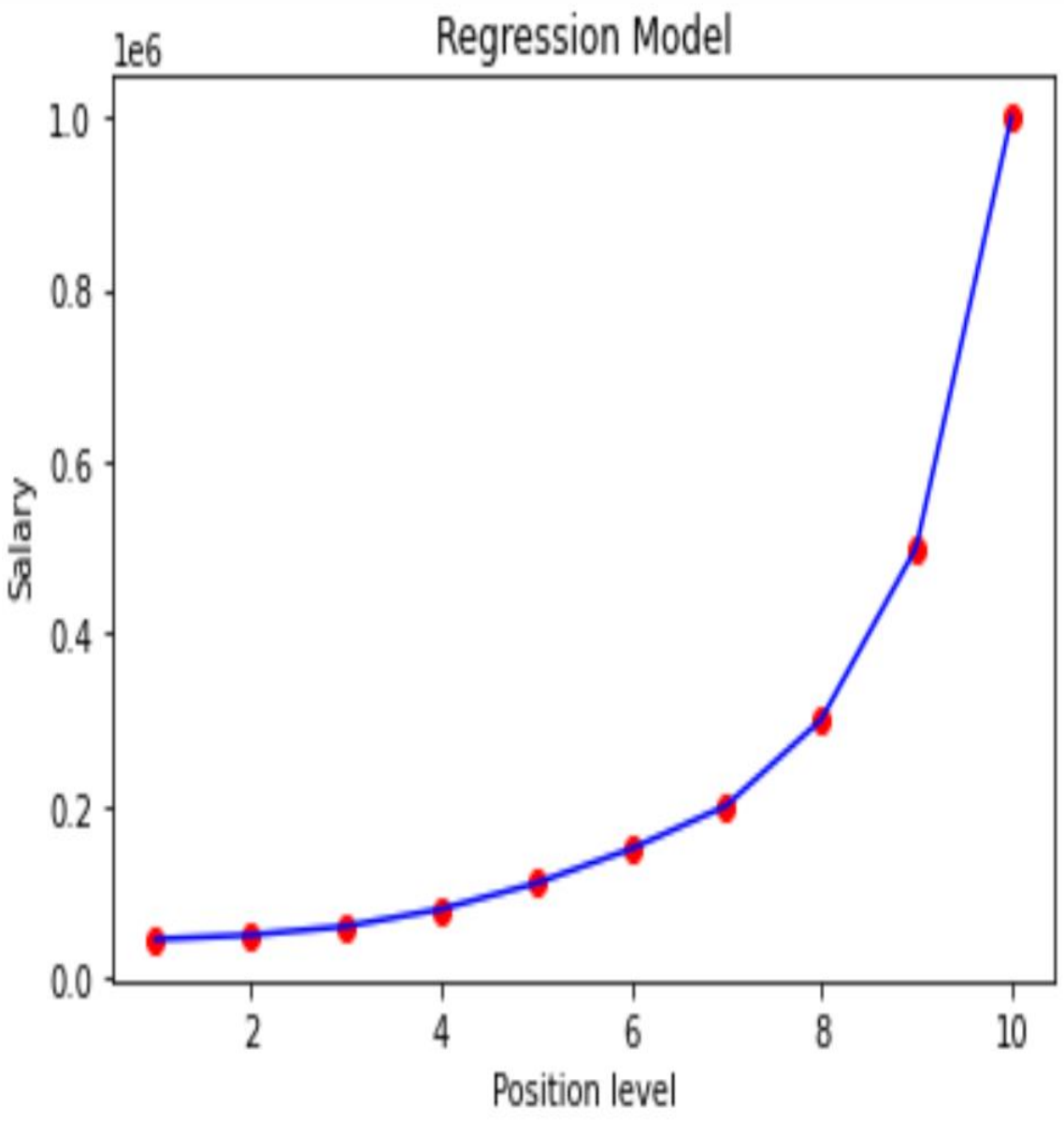


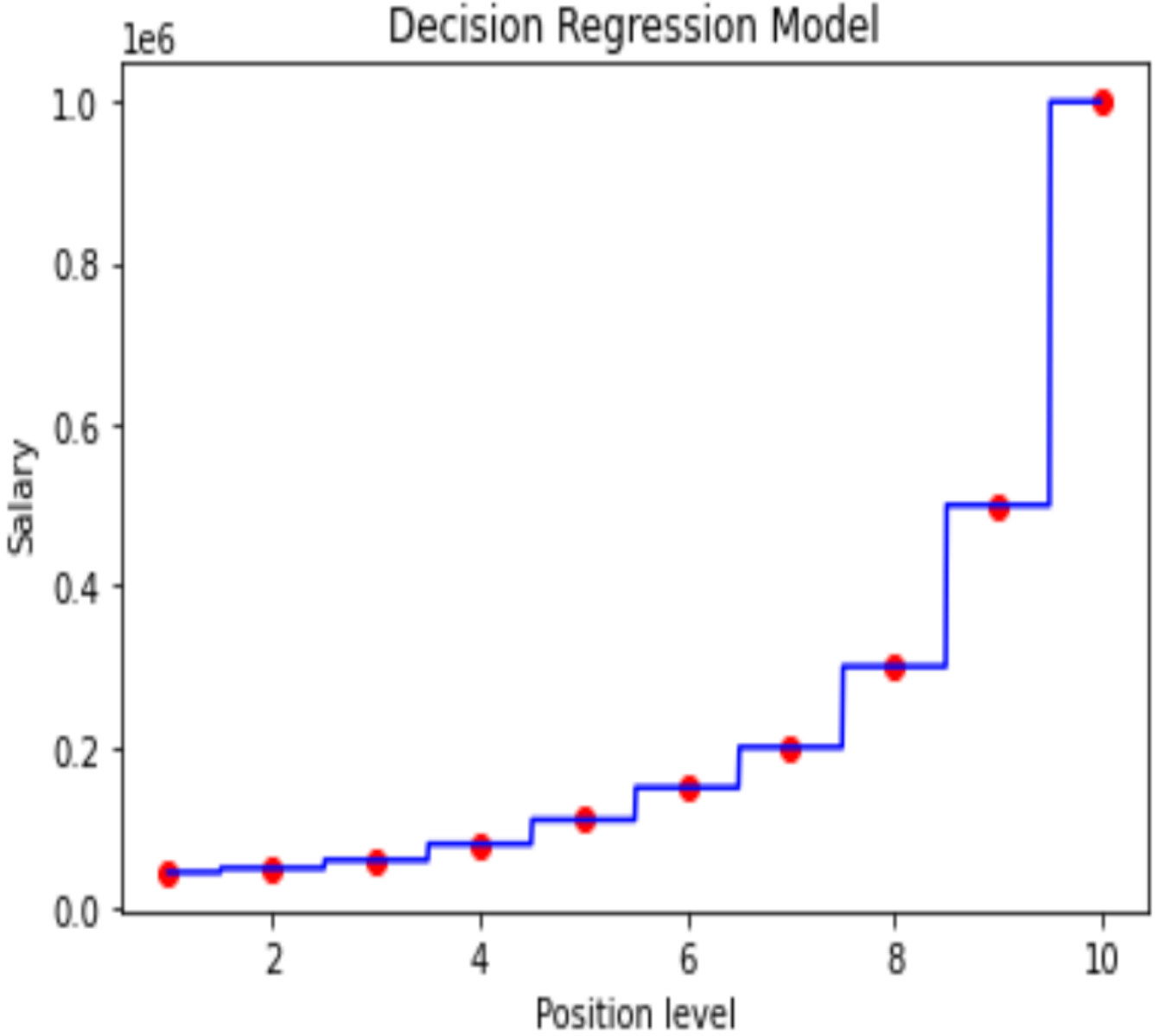


Regions and Function Values

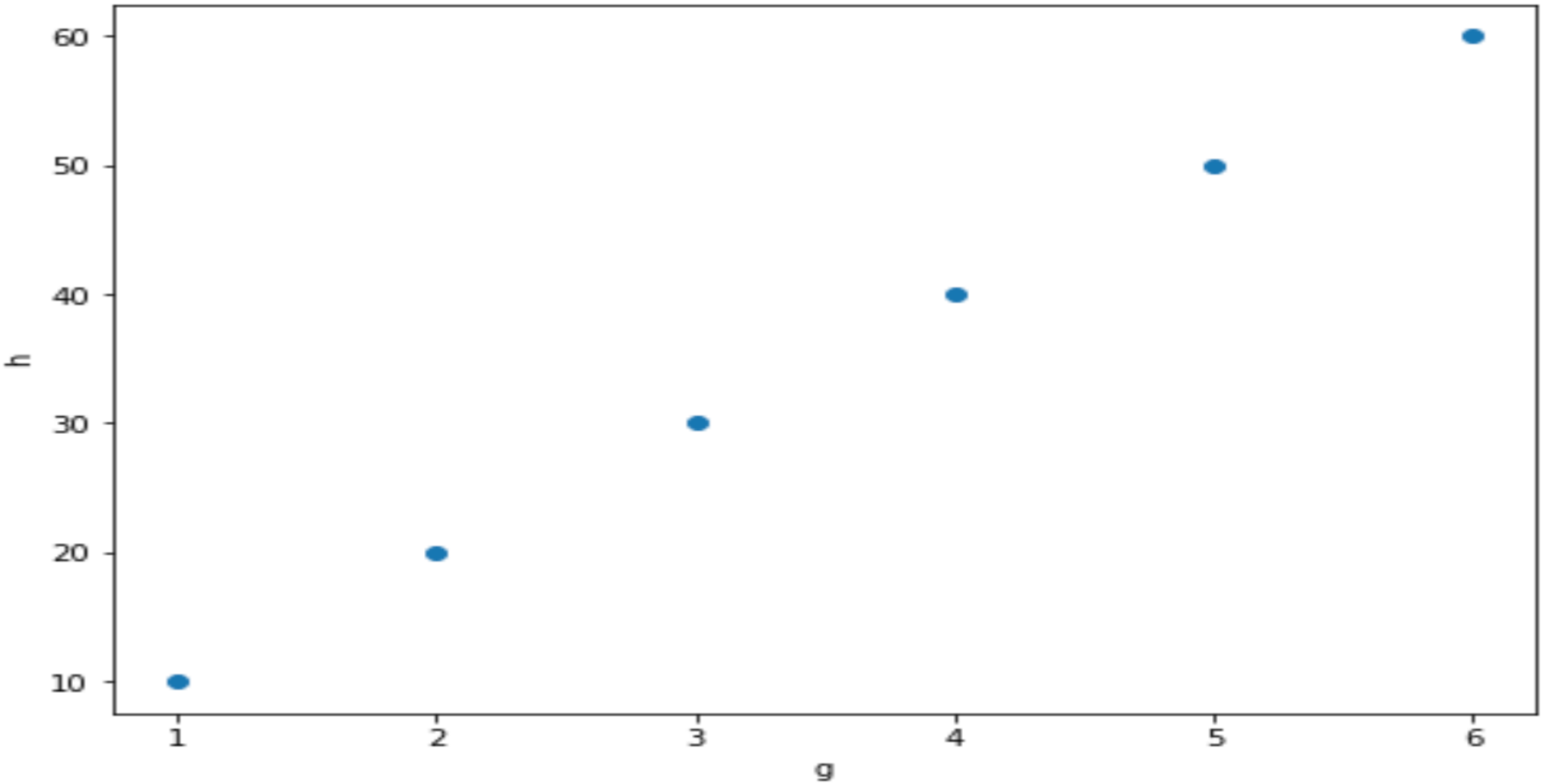
- Tree based methods for Regression involve **segmenting the space of points** into **piece-wise linear regions** using axis parallel splits.
- The splitting can be represented using a tree, where each **internal node abstracts a decision** and each **leaf node represents a simpler** region of similar patterns.
- The basic idea of these methods is to partition the space and **identify some representative vectors.**
- We make predictions in a regression problem by dividing the space of all the possible **values based on the variables** into Region_1, Region_2, …, Region_k corresponding to k leaf nodes.
- Then for every vector X that falls into a **particular region (say $\mathbf{Region_j}$)** we make the **same predicti**on.
- Used knn's earlier. Here a region is used.

Regression Based on Decision Trees

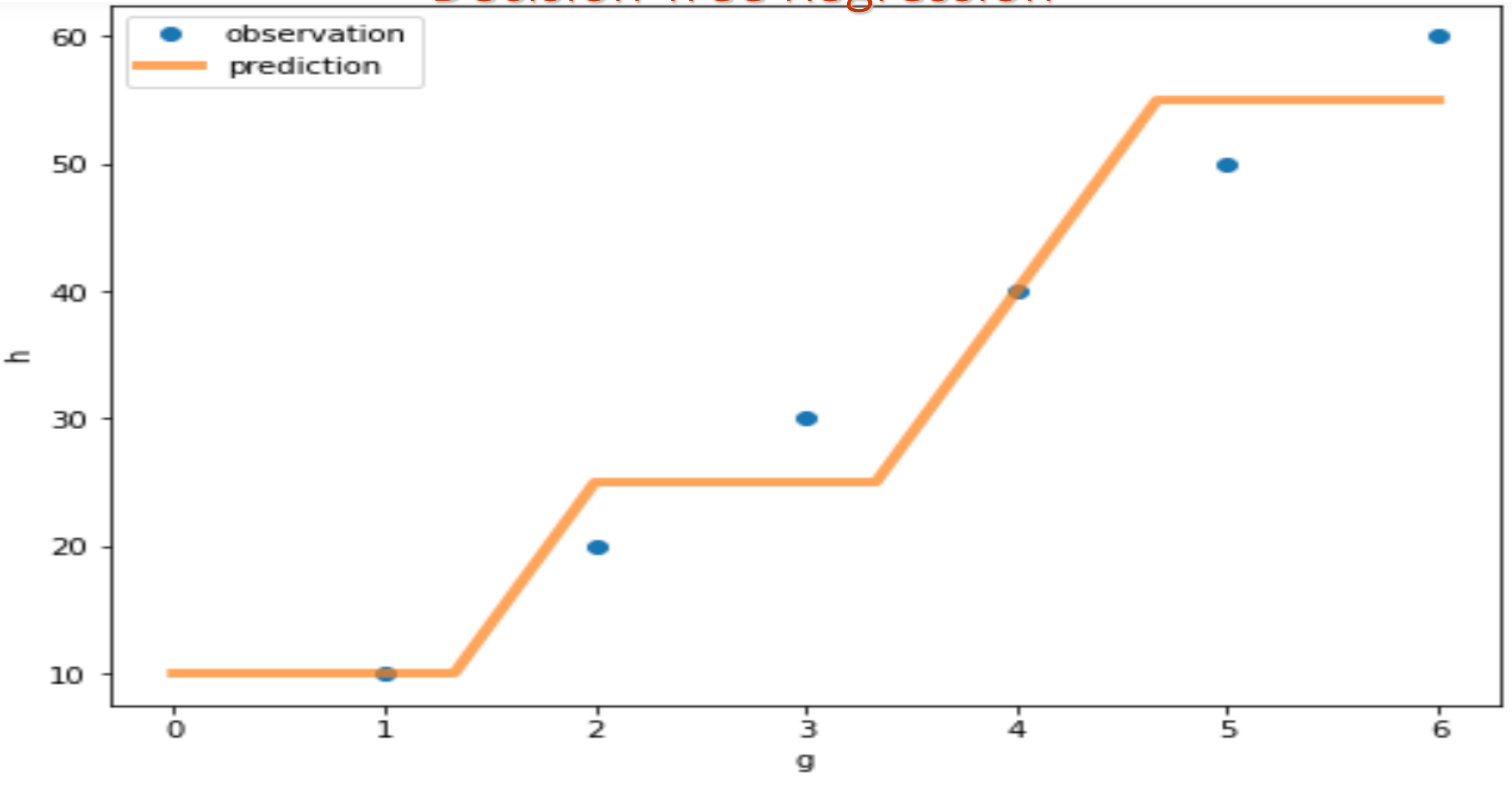




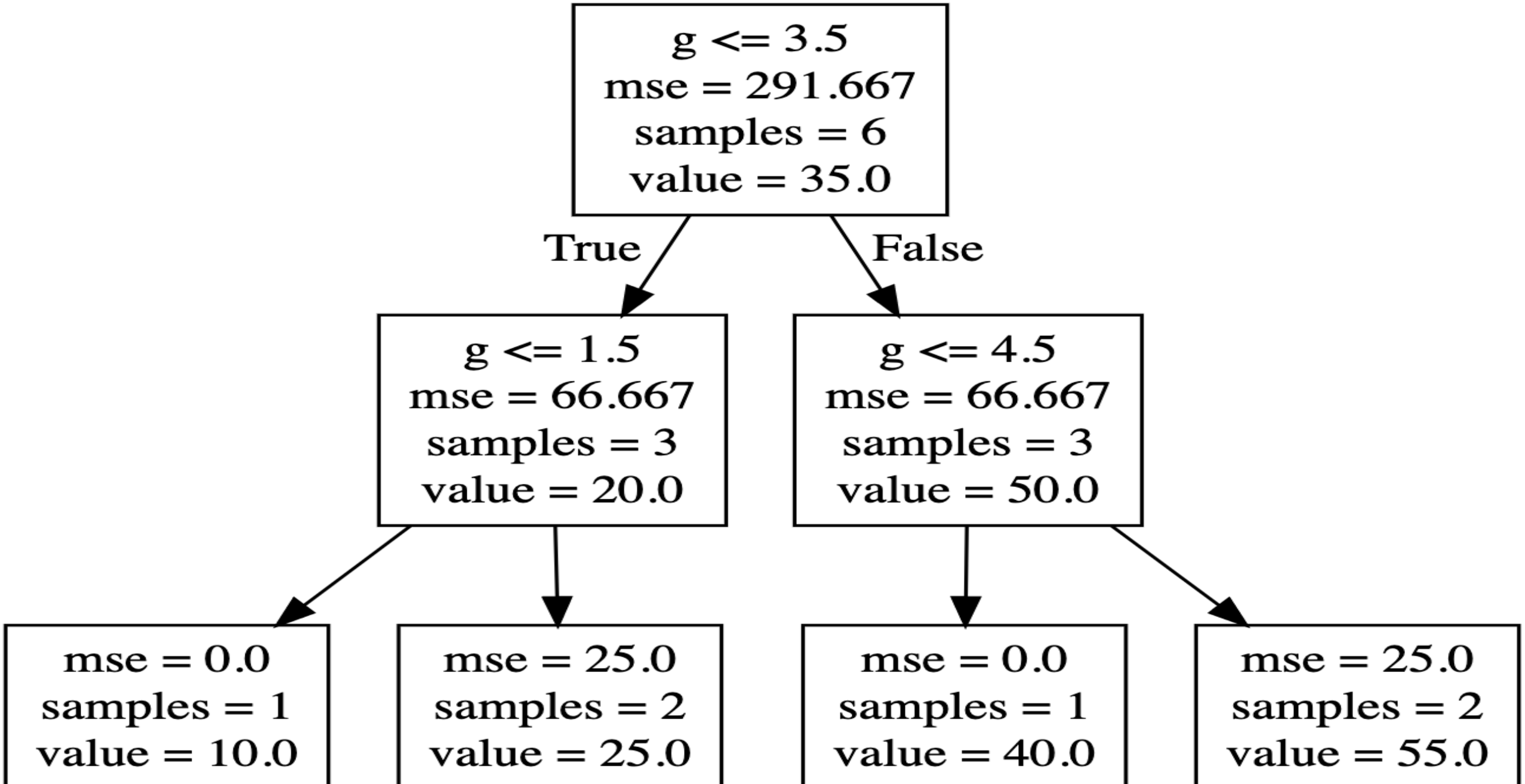
Data for Regression



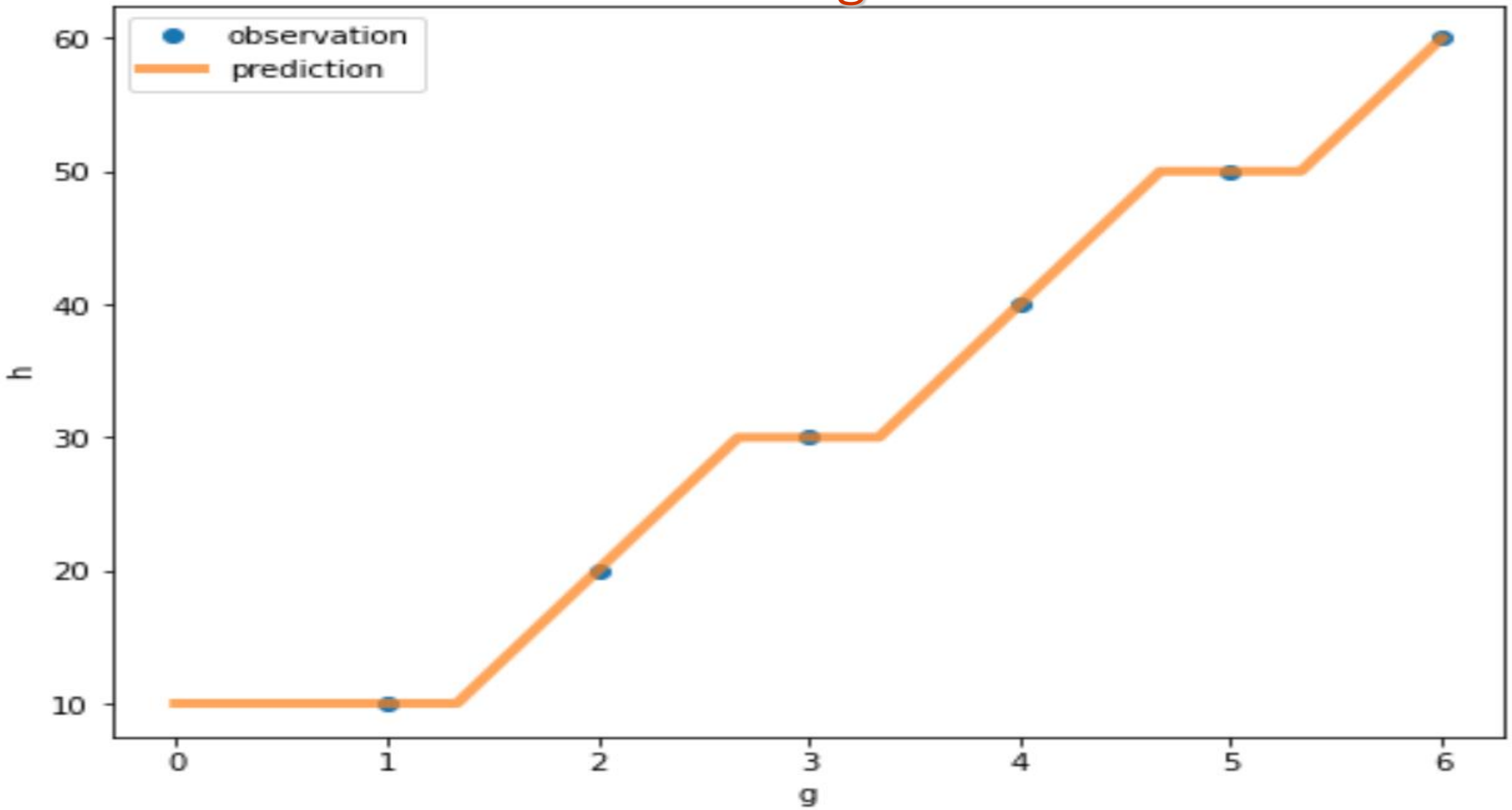
Decision Tree Regression



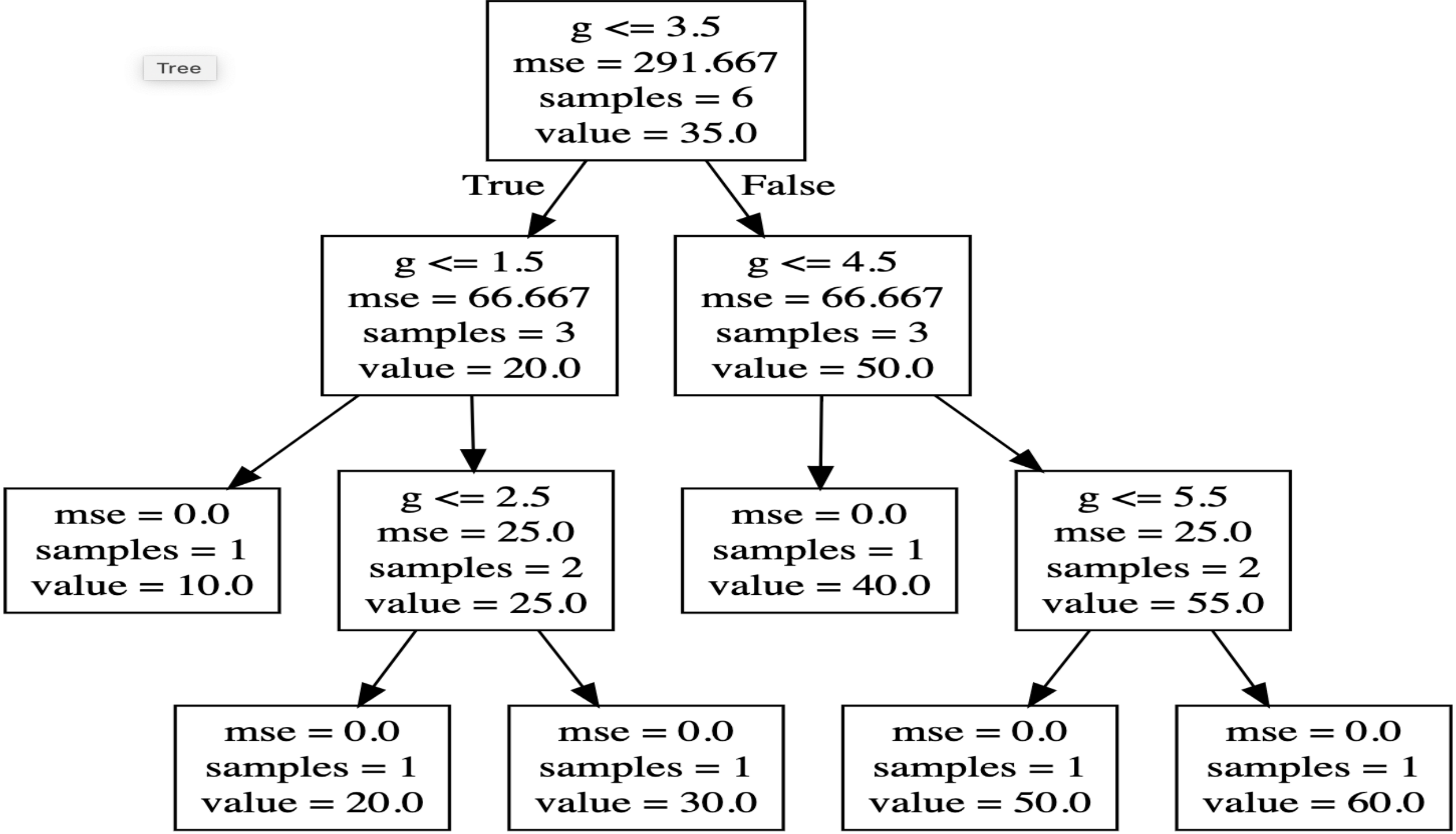
Decision Tree

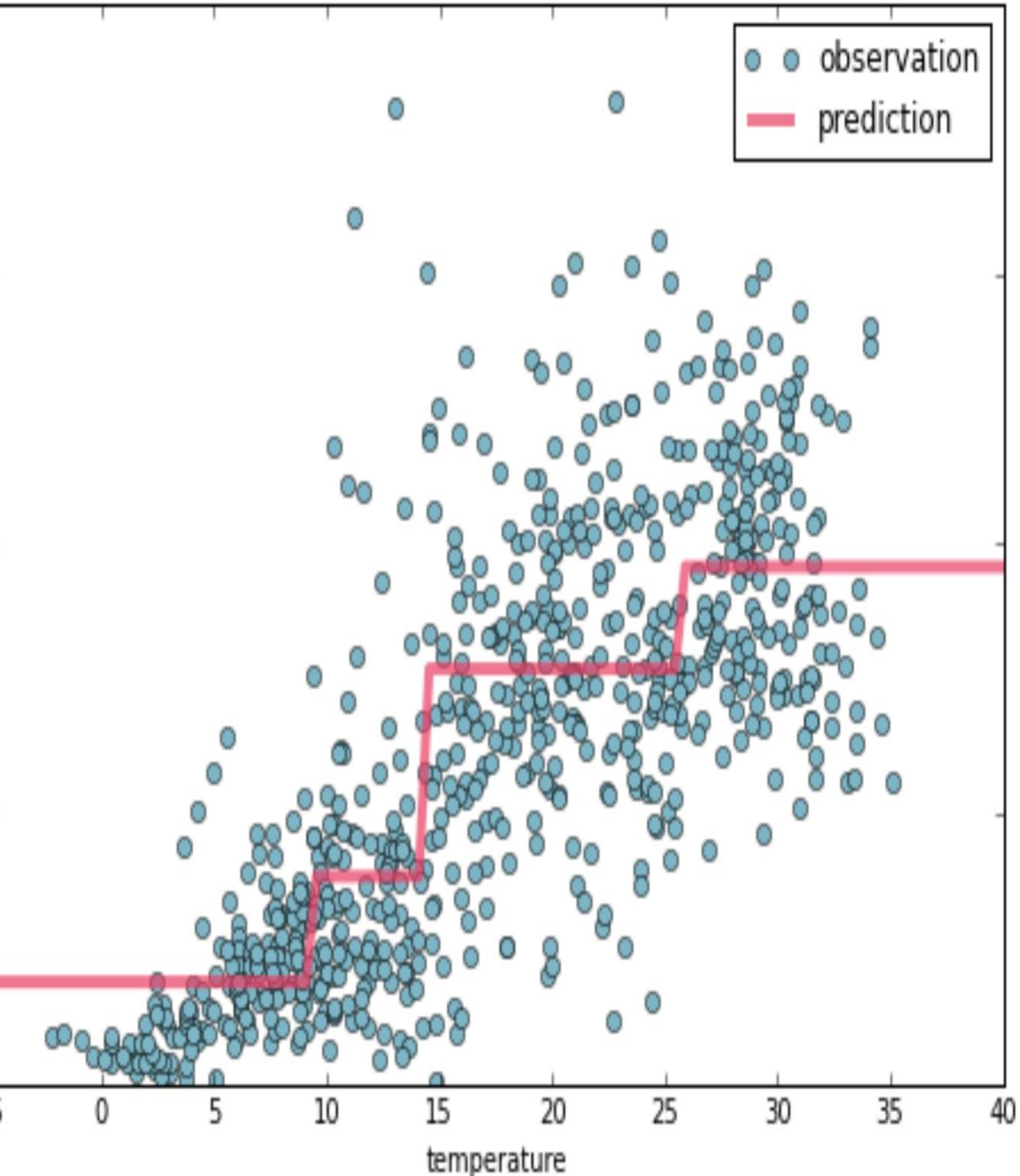


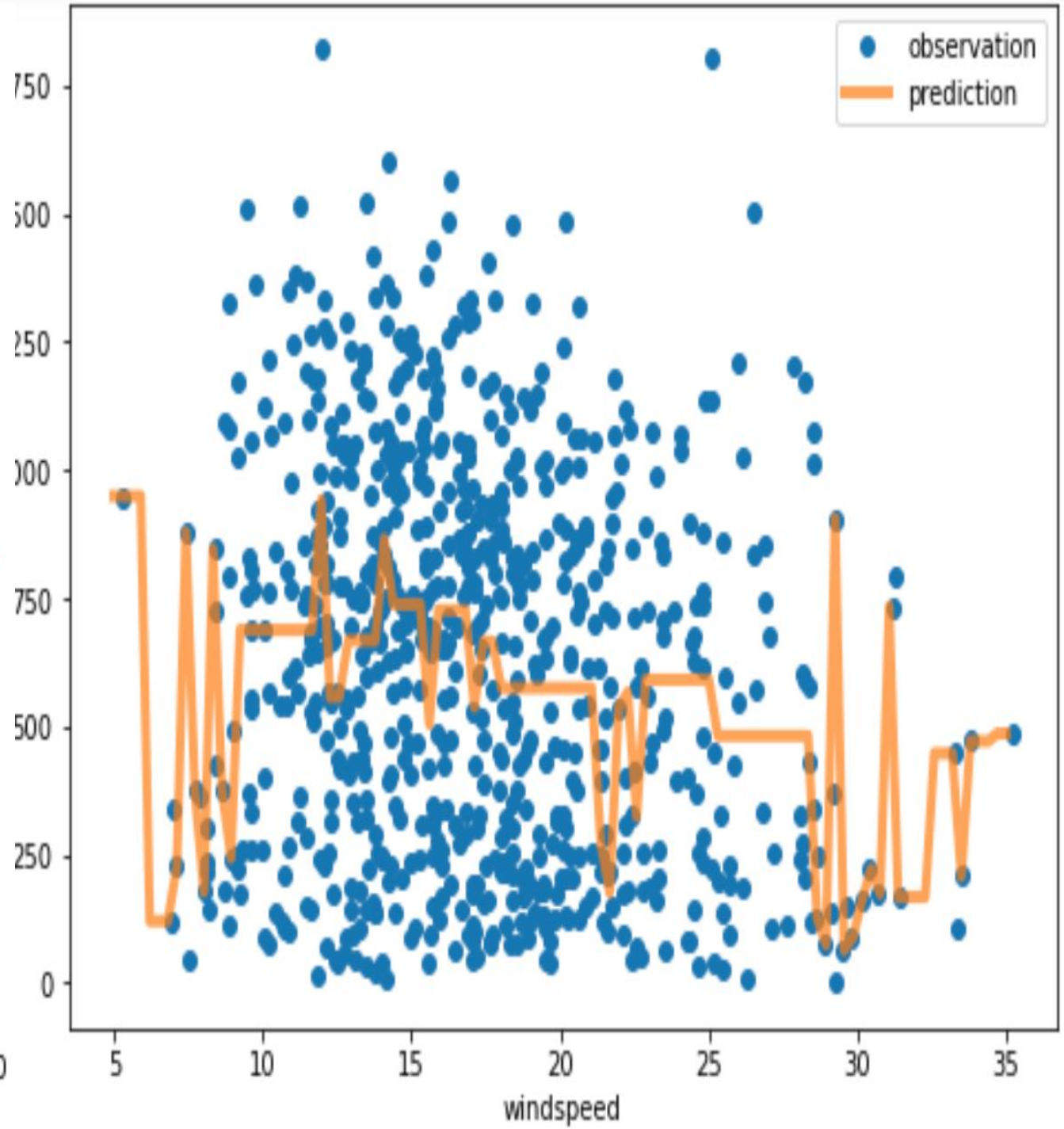
Decision Tree Regression



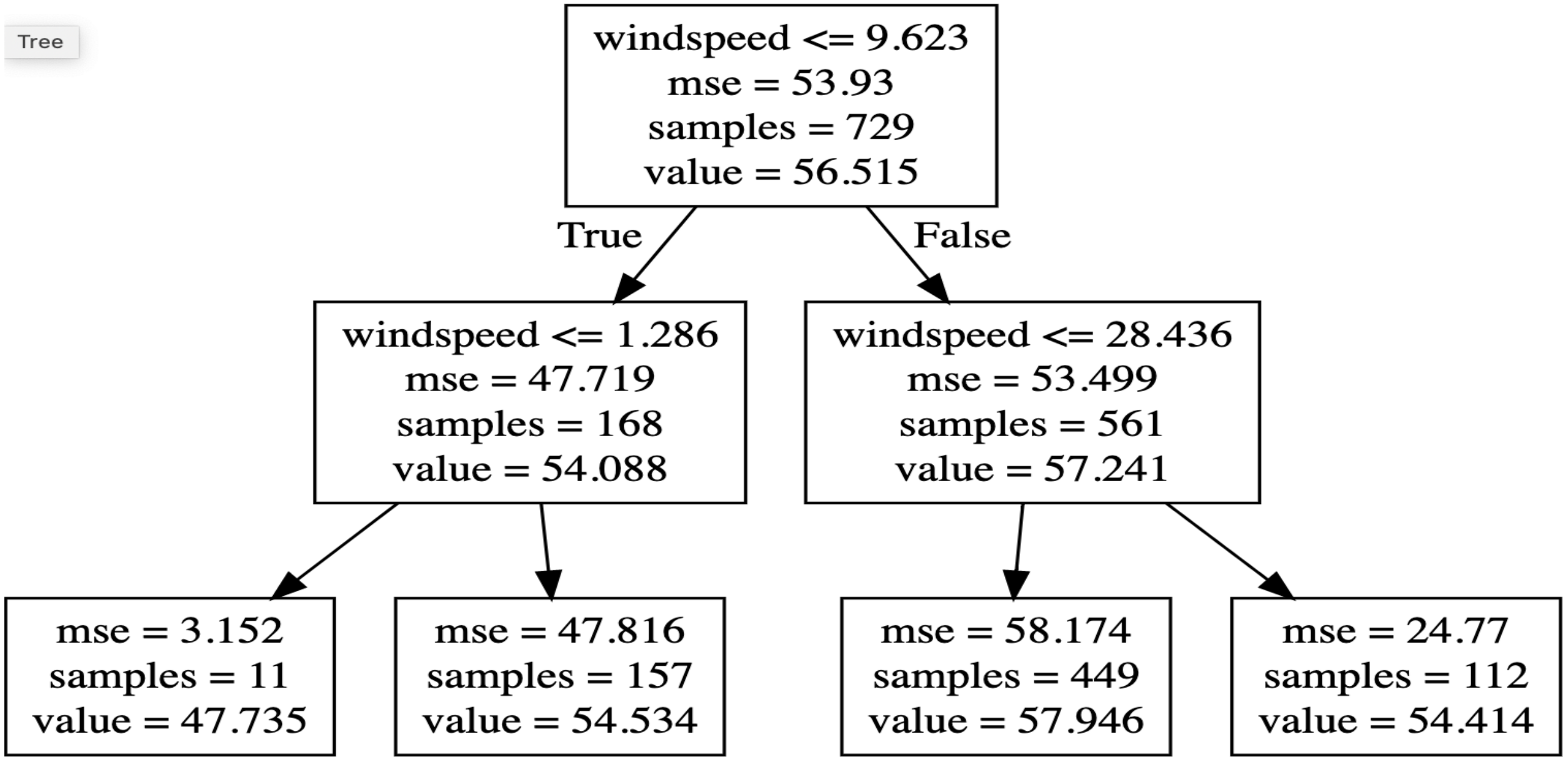
Tree


g <= 3.5
mse = 291.667
samples = 6
value = 35.0
True
False
g <= 1.5
mse = 66.667
samples = 3
value = 20.0
g <= 4.5
mse = 66.667
samples = 3
value = 50.0
mse = 0.0
samples = 1
value = 10.0
g <= 2.5
mse = 25.0
samples = 2
value = 25.0
mse = 0.0
samples = 1
value = 40.0
g <= 5.5
mse = 25.0
samples = 2
value = 55.0
mse = 0.0
samples = 1
value = 20.0
mse = 0.0
samples = 1
value = 30.0
mse = 0.0
samples = 1
value = 50.0
mse = 0.0
samples = 1
value = 60.0


observation
prediction
0
5
10
15
20
25
30
35
40
temperature


observation
prediction
750
500
250
000
750
500
250
0
5
10
15
20
25
30
35
windspeed

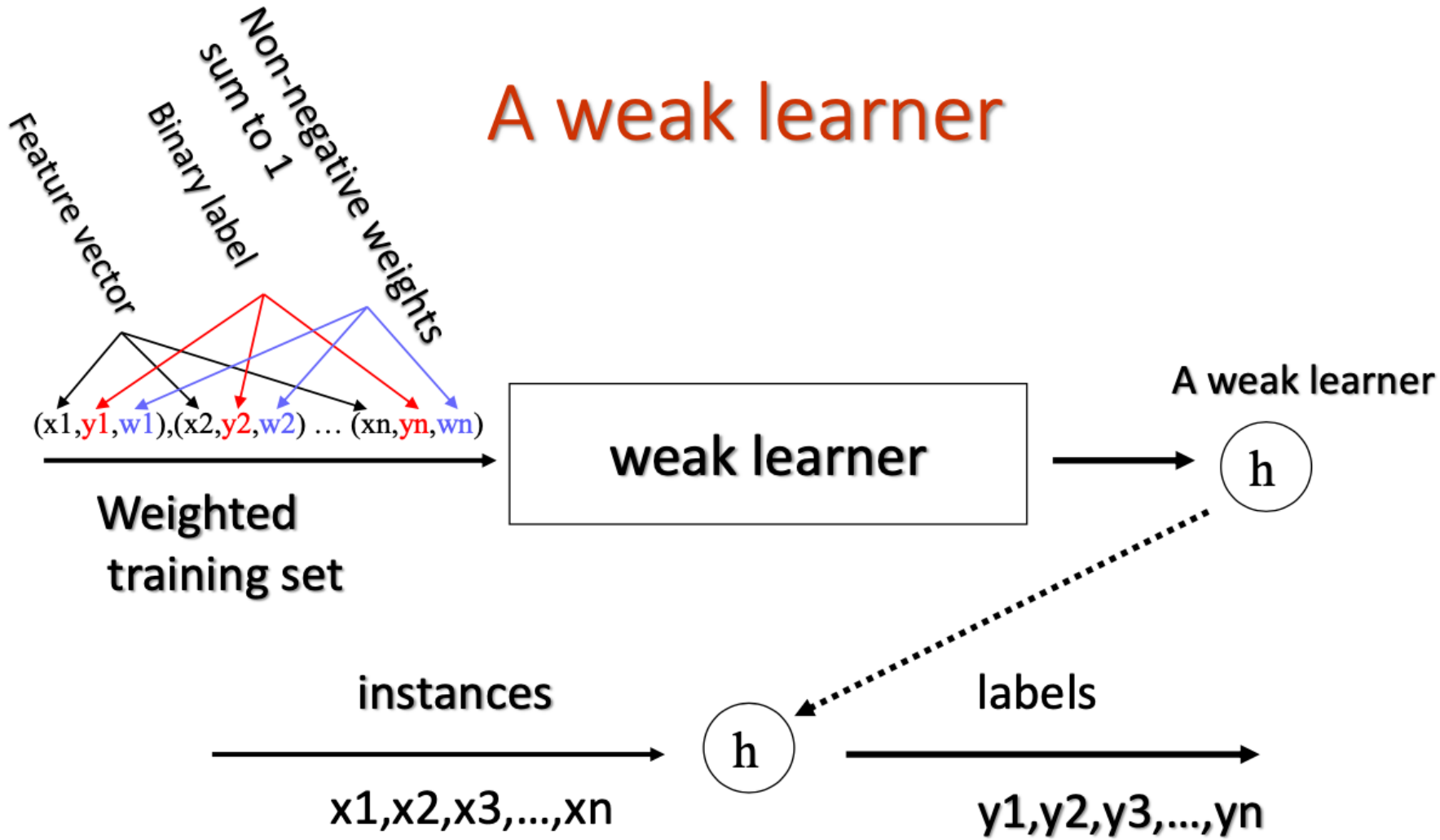
Tree


windspeed <= 9.623
mse = 53.93
samples = 729
value = 56.515
True
False
windspeed <= 1.286
mse = 47.719
samples = 168
value = 54.088
windspeed <= 28.436
mse = 53.499
samples = 561
value = 57.241
mse = 3.152
samples = 11
value = 47.735
mse = 47.816
samples = 157
value = 54.534
mse = 58.174
samples = 449
value = 57.946
mse = 24.77
samples = 112
value = 54.414

What is Boosting?

- A method for **improving classifier accuracy**
- Basic idea:
  - Perform iterative search to locate the **regions/examples that are more difficult to predict**.
  - Through each iteration **reward accurate predictions** on these regions
  - **Combine the rules** from different iterations.
  - Only requires that the underlying **learning algorithm be better than guessing.**

A weak learner



The **weak requirement**: $\frac{\sum_{i:y_i \neq \mathbf{y}_i} w_i}{\sum_{i=1}^{n} w_i} < \frac{1}{2} - \gamma$

Boosting: Weak Learners

- **Boosting** refers to **combining** multiple **weak learners** to get a **strong learner**.
- We can understand this definition by looking at some simple weak learner class.

- **1-level decision trees**: ones which classify examples on the basis of a single attribute
- A program, called 1R, that learns 1−rules from examples was compared to C4 on 16 datasets commonly used in ML research.
- Individually, these rules are not powerful enough to classify the dataset well. Therefore, these rules are called as **weak learners.**
- The main result of comparing 1R and C4 is insight into the trade-off between **simplicity and accuracy.**
- 1R's rules are only a little less accurate (3.1 percentage points) than C4's pruned decision trees on almost all of the datasets.
- **https://www.cs.cornell.edu/courses/cs478/2000SP/lectures/decision-stumps.pdf**

Decision Trees: Weak Learners

TABLE 5. 1Rw measured on the datasets.
C4 — as in Table 4.
1Rw — highest accuracy of the 1−rules produced when the whole dataset is used by 1R for both training and testing.

	Dataset							
	BC	CH	GL	G2	HD	HE	HO	HY
C4	72.0	99.2	63.2	74.3	73.6	81.2	83.6	99.1
1Rw	72.7	68.3	62.2	78.5	76.6	84.5	81.5	98.0

	Dataset							
	IR	LA	LY	MU	SE	SO	VO	V1
C4	93.8	77.2	77.5	100.0	97.7	97.5	95.6	89.4
1Rw	96.0	84.2	75.7	98.5	95.0	87.2	95.6	87.4

BC: breast-cancer/breast-cancer.data
CH: chess-end-games
GL: glass/glass.data.
G2: GL with classes 1 and 3 combined
HD: heart-disease
HE: hepatitis/hepatitis.data
HO: undocumented/taylor/horse-colic.data
HY: thyroid-disease/hypothyroid.data
IR: iris/iris.data
LA: labor-negotiations.
LY : lymphography/lymphography.data
MU: mushroom/agaricus-lepiota.data
SE: thyroid-disease/sick-euthyroid.data
SO: soybean/soybean-small.data
VO: voting-records/house-votes-84.data
V1: VO with some attribute deleted

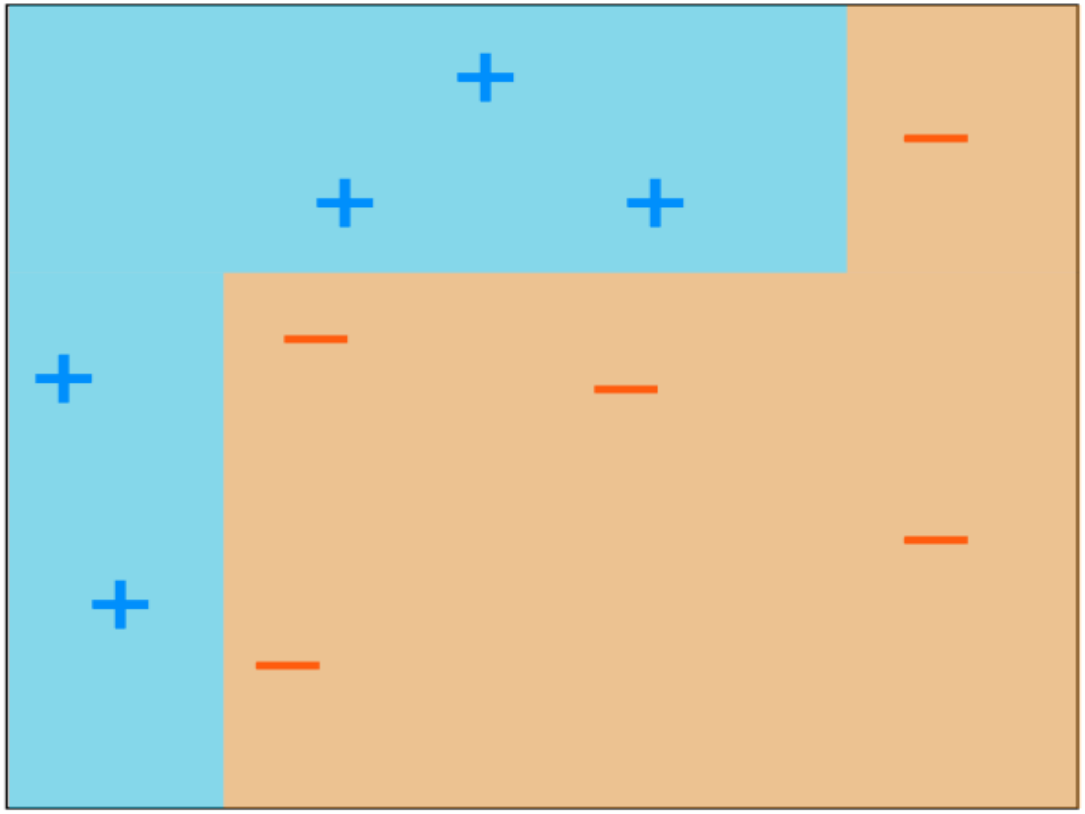
Robert Holte, Very Simple Classification Rules Perform Well on Most Commonly Used Datasets, machine learning, 1993.

Decision Stumps : UCI ML Datasets

TABLE 2. Datasets used in the experiments.
(blank entries represent 0's)

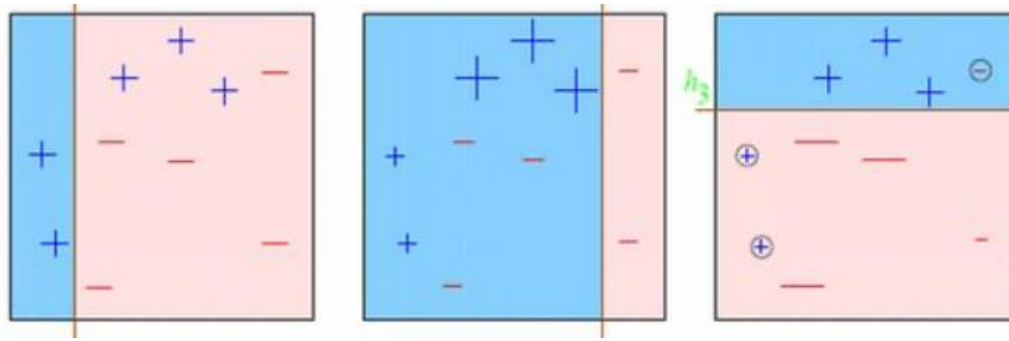
Dataset	Size	Baseline Accuracy	Missing Values	Attributes ... number of distinct values							
				cont	2	3	4	5	6	>6	TOTAL
BC	286	70.3	yes		3	2		1	1	2	9
CH	3196	52.2	no		35	1					36
GL (6)	214	35.5	no	9							9
G2	163	53.4	no	9							9
HD	303	54.5	yes	5	3	3	2				13
HE	155	79.4	yes	6	13						19
HO	368	63.0	yes	7	2	5	5	2	1		22
HY	3163	95.2	yes	7	18						25
IR (3)	150	33.3	no	4							4
LA	57	64.9	yes	8	3	5					16
LY (4)	141	56.7	no	2	9	2	5				18
MU	8124	51.8	yes		5	1	5	1	2	7	22
SE	3163	90.7	yes	7	18						25
SO (4)	47	36.2	no		13	3	4			1	35
VO	435	61.4	yes		16						16
V1	435	61.4	yes		15						15

Example Classification Problem



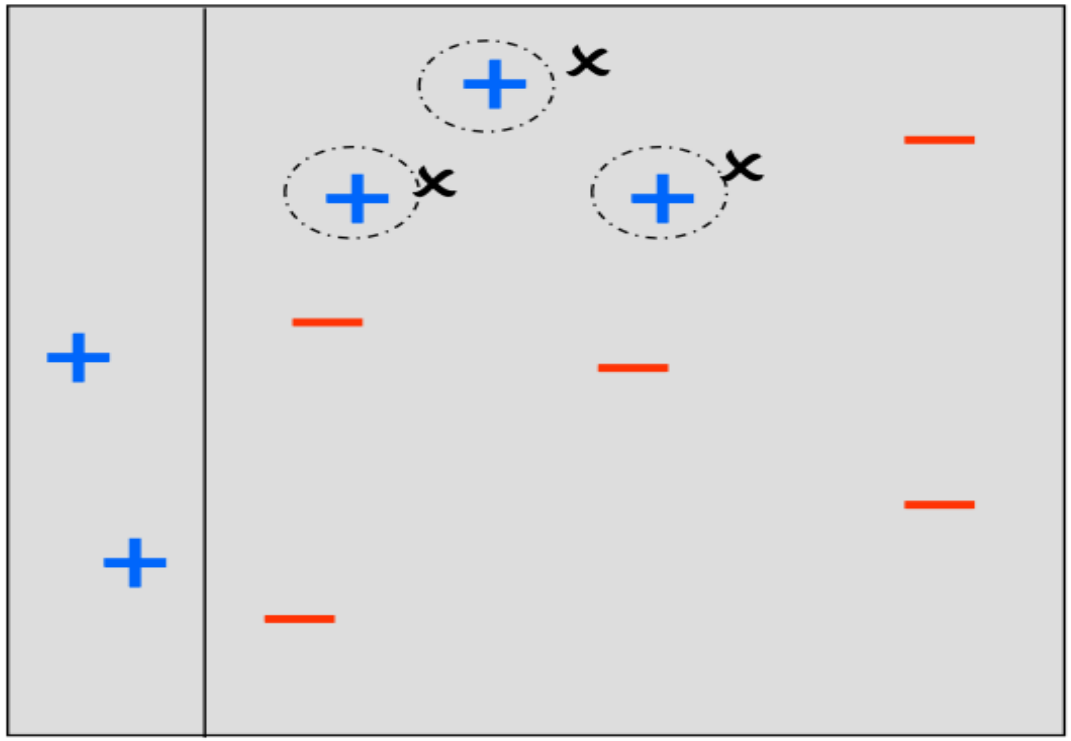
AdaBoost Blog: https://mccormickml.com/2013/12/13/adaboost-tutorial/

AdaBoost: The Overall Idea



- Consider the diagram below:
- Decision stump on feature1 is the **first weak learner.** (first box).
- We have **3 misclassified observations out of total 10** using this stump.
- We give **larger weights** to these 3 misclassified observations next.
- It becomes very important to classify these right. Hence, the **next decision stump moves towards the right side** edge in the second box.
- We repeat this process and combine the learners weighted appropriately.
- Which learner is **more important**?

Weak Learner - 1



$\varepsilon_1 = 0.300$

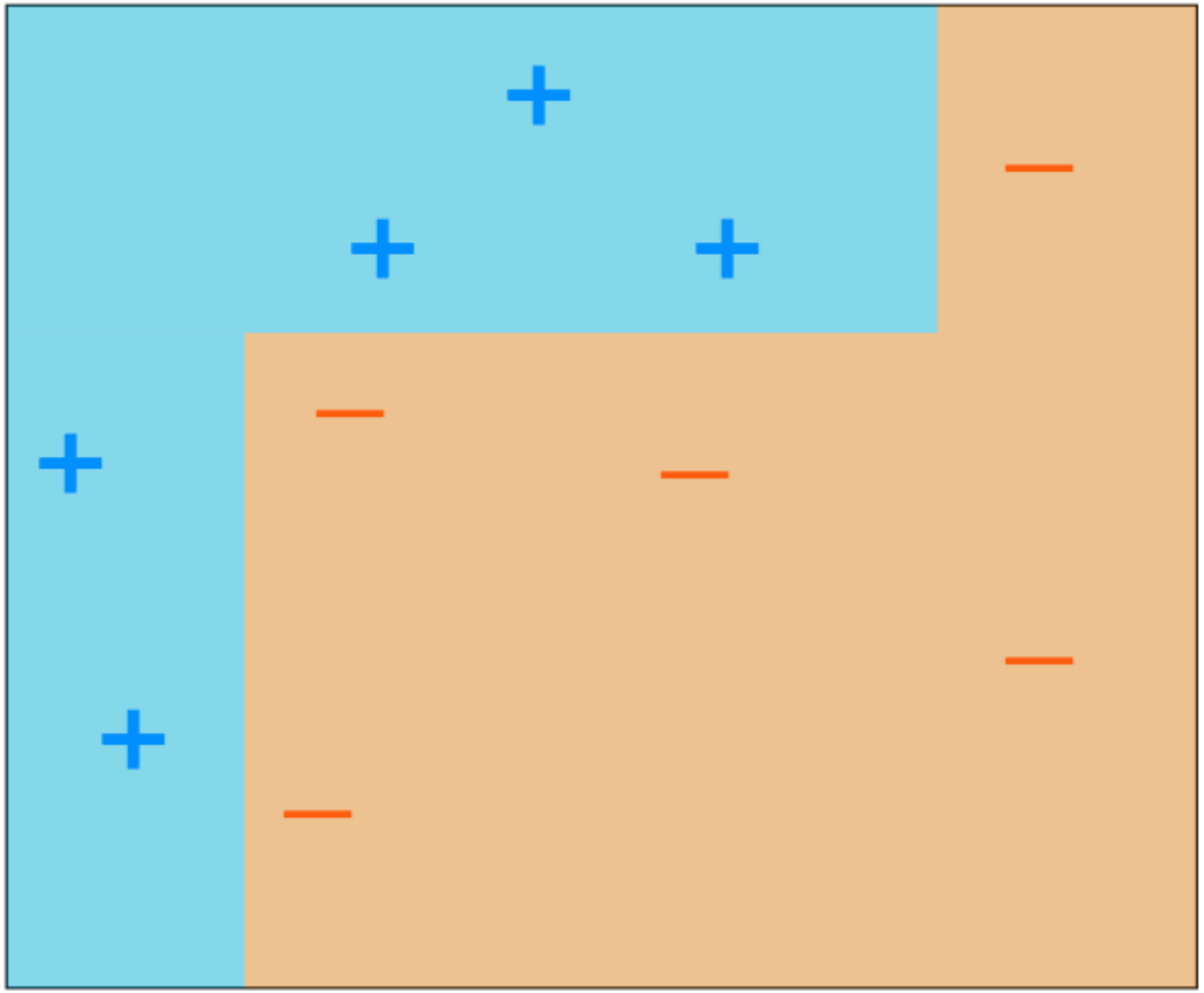
*h*1

What happens if the classifier says left region is –ve and right one is + ve?

The error is 0.7 (> 0.5). How do you correct?

Swap the Class Labels!

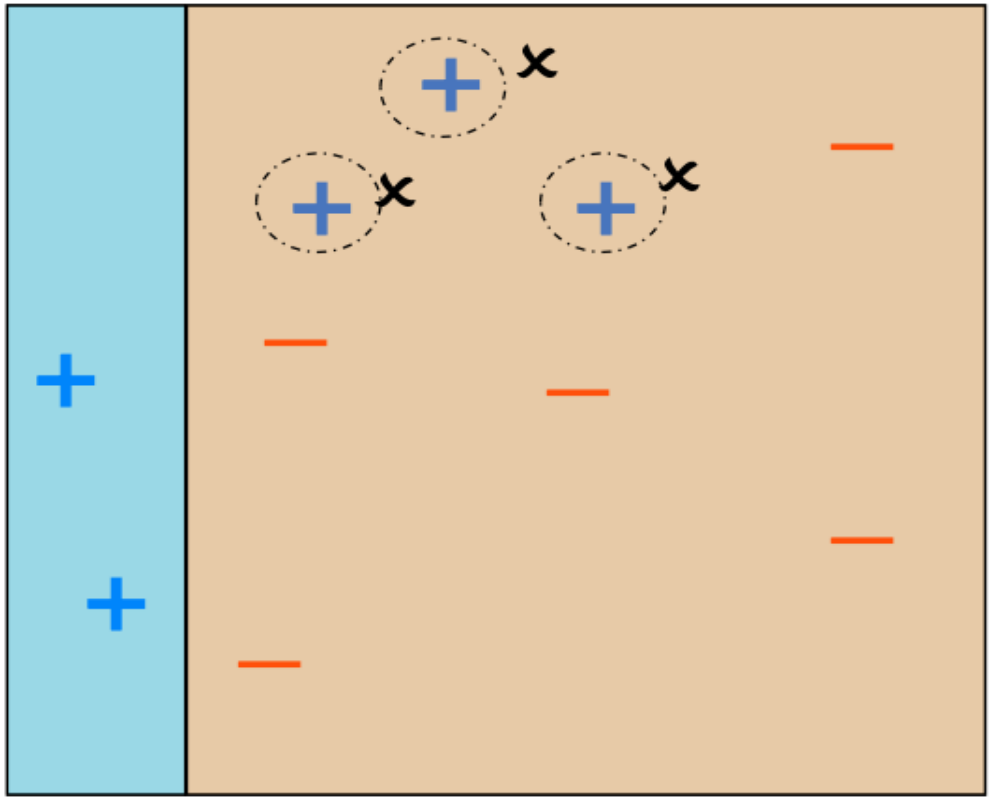
AdaBoost: Example 2-Class Data

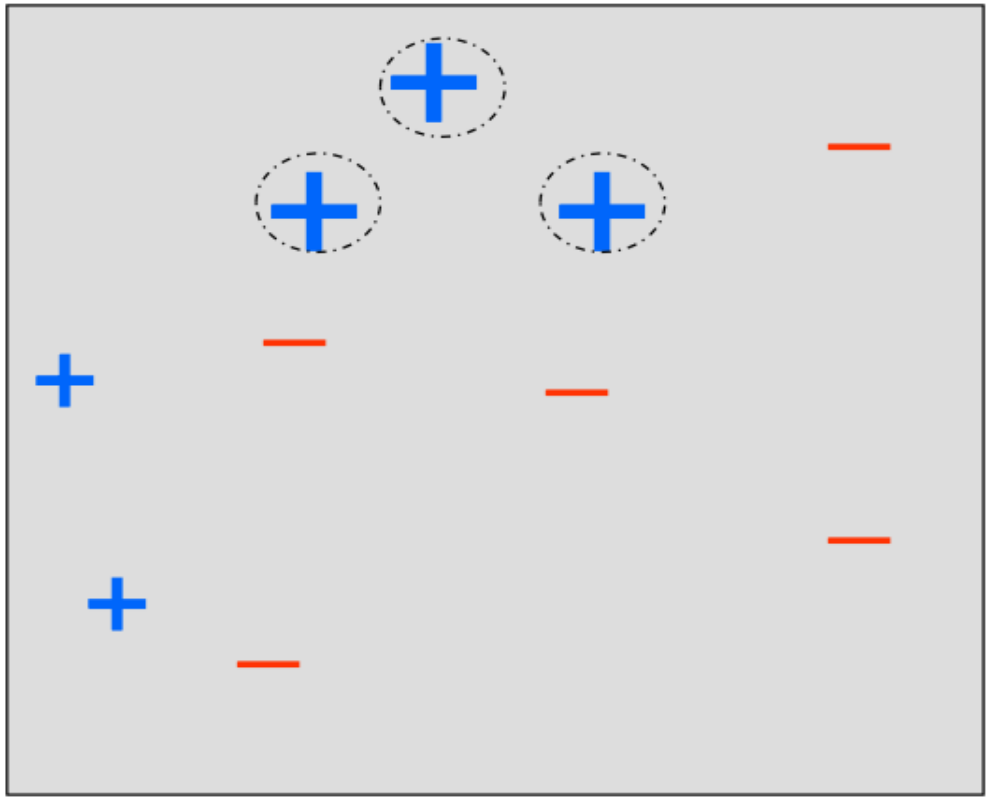


$$D_1\ (\text{i}) = \frac{1}{10}$$

All the patterns are given equal weight

AdaBoost: First Weak Learner





h_1

$\varepsilon_1 = 0.300$

D_2

$$\alpha_t = \frac{1}{2}\ln\left(\frac{1-\varepsilon_t}{\varepsilon_t}\right)$$

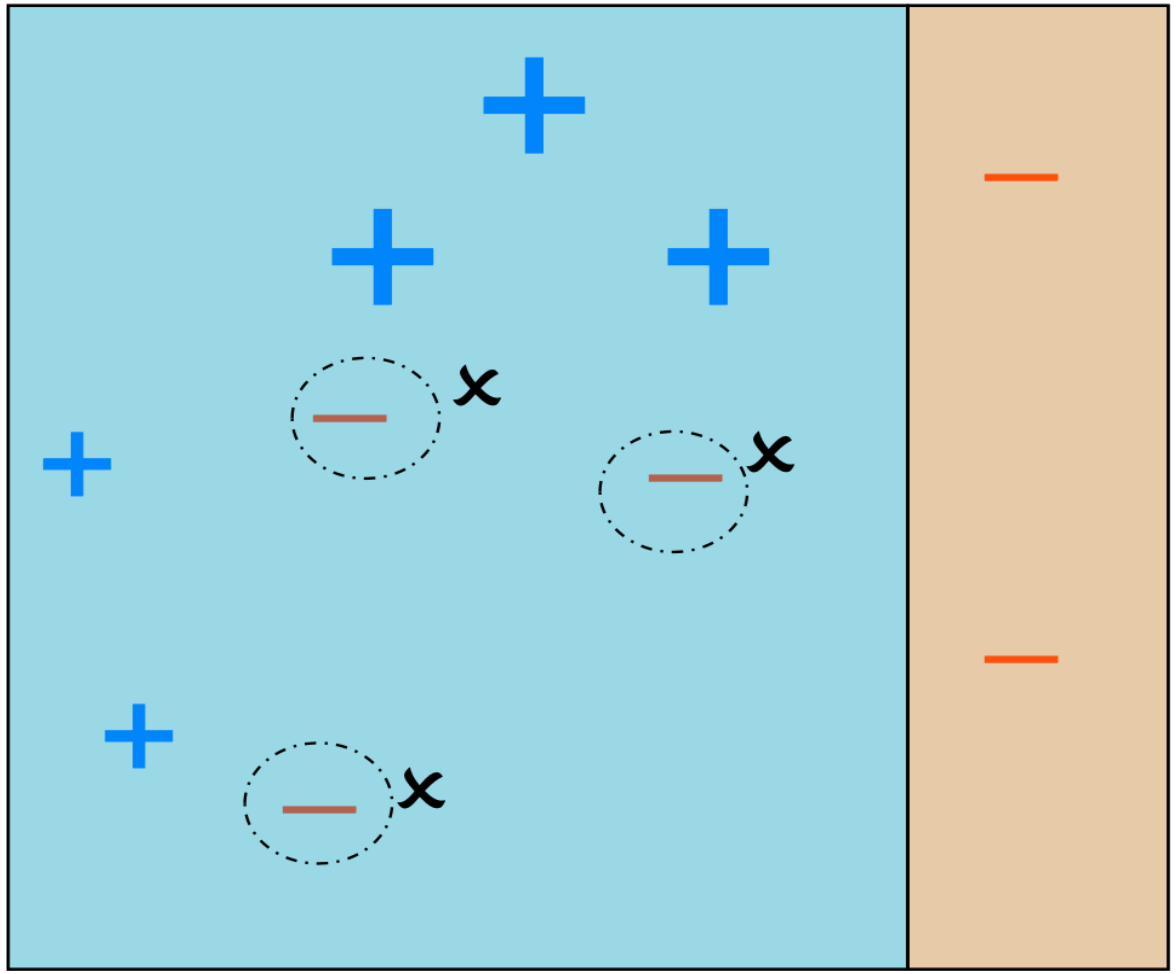
The weight Adapts. The bigger e_t becomes the smaller a_t becomes.

$\alpha_1 = 0.424$

$$D_{t+1}(i) = \frac{D_t(i)}{Z_t} \times \begin{cases} e^{-\alpha_t} & \text{if } h_t(x_i) = y_i \\ e^{\alpha_t} & \text{if } h_t(x_i) \neq y_i \end{cases}$$

Boost example if incorrectly predicted.

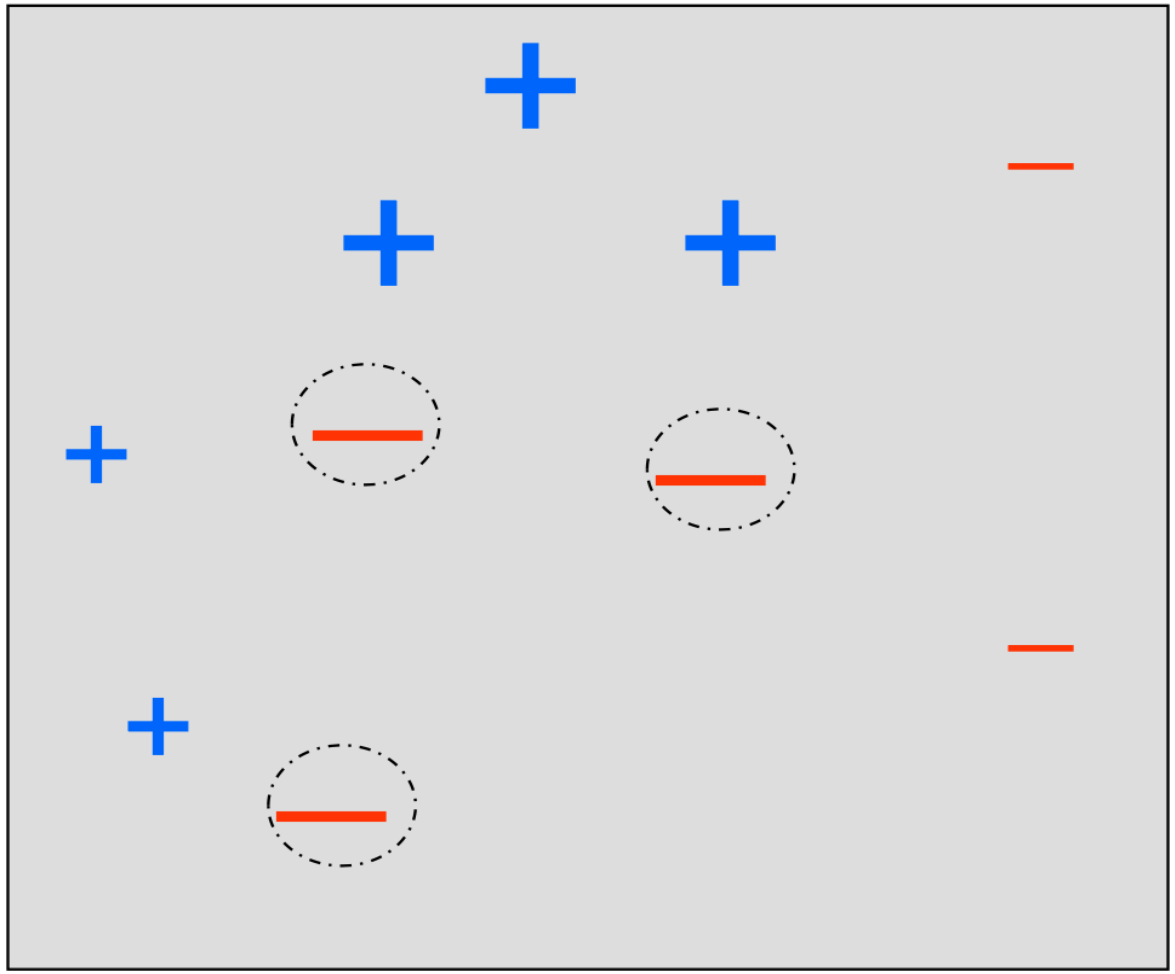
AdaBoost: Second Weak Learner



$\varepsilon_2 = 0.196$

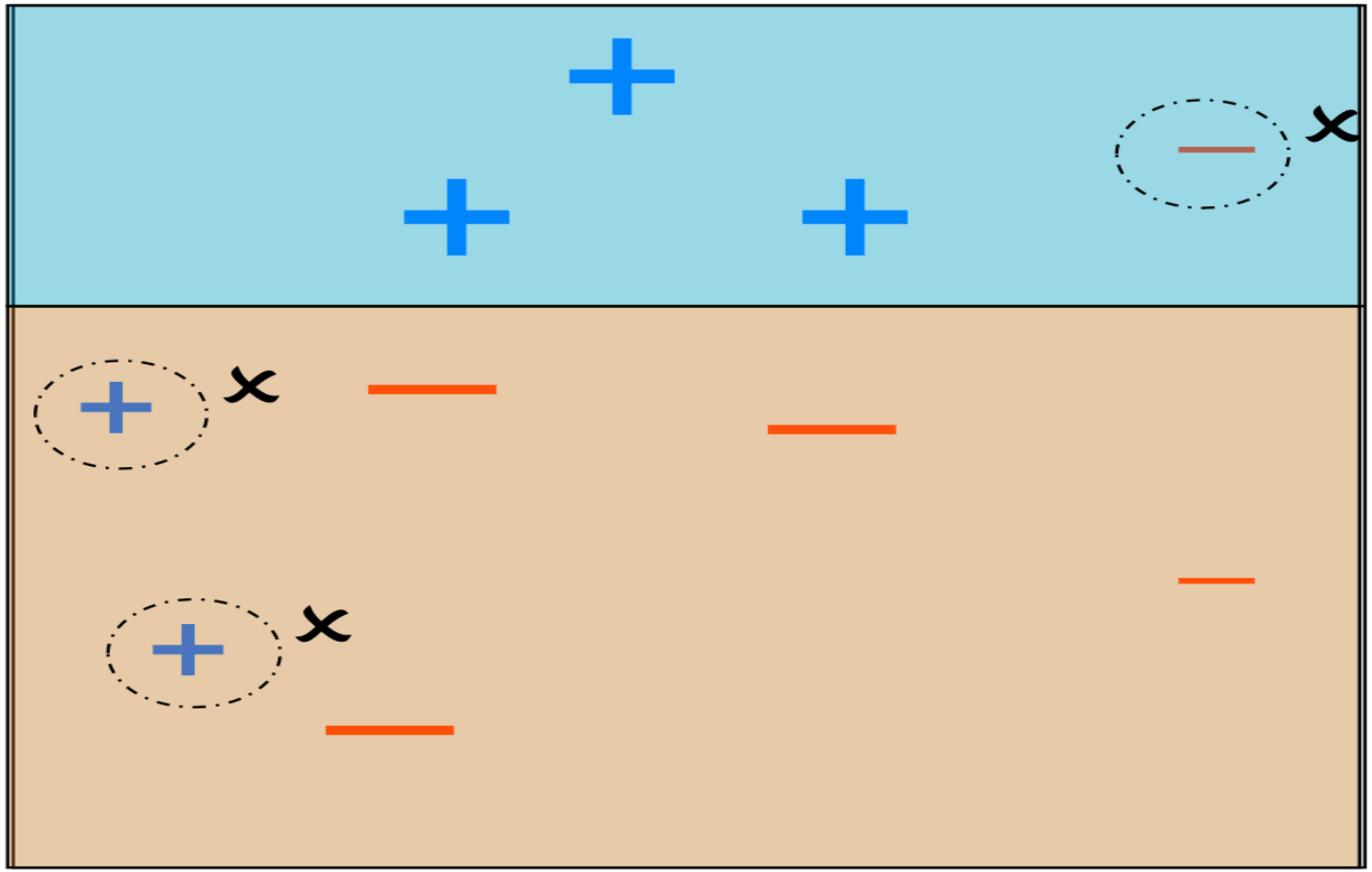
h_2

$\alpha_2 = 0.704$



D_2

AdaBoost: Third Weak Learner



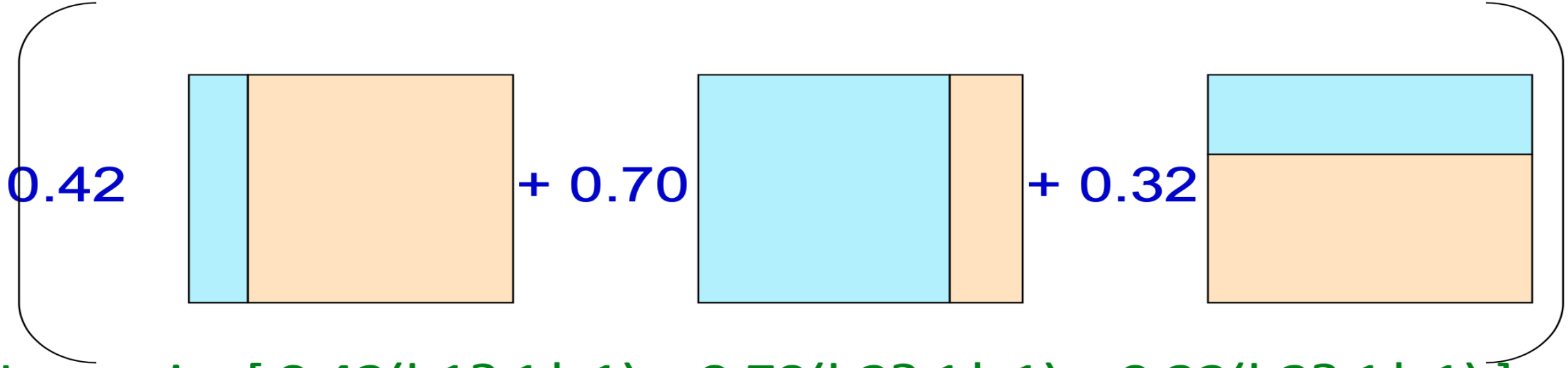
h_3

STOP

$\varepsilon_3 = 0.344$

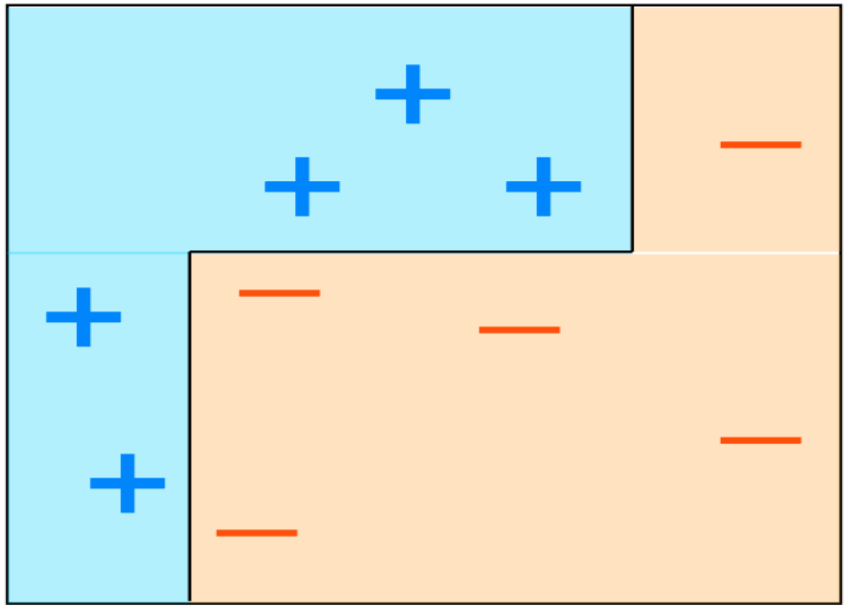
$\alpha_2 = 0.323$

AdaBoost: Final Classifier



$$H_{final} = \text{sign}[\ 0.42(h1?\ 1|{-1}) + 0.70(h2?\ 1|{-1}) + 0.32(h3?\ 1|{-1})\ ]$$

Yoav Freund

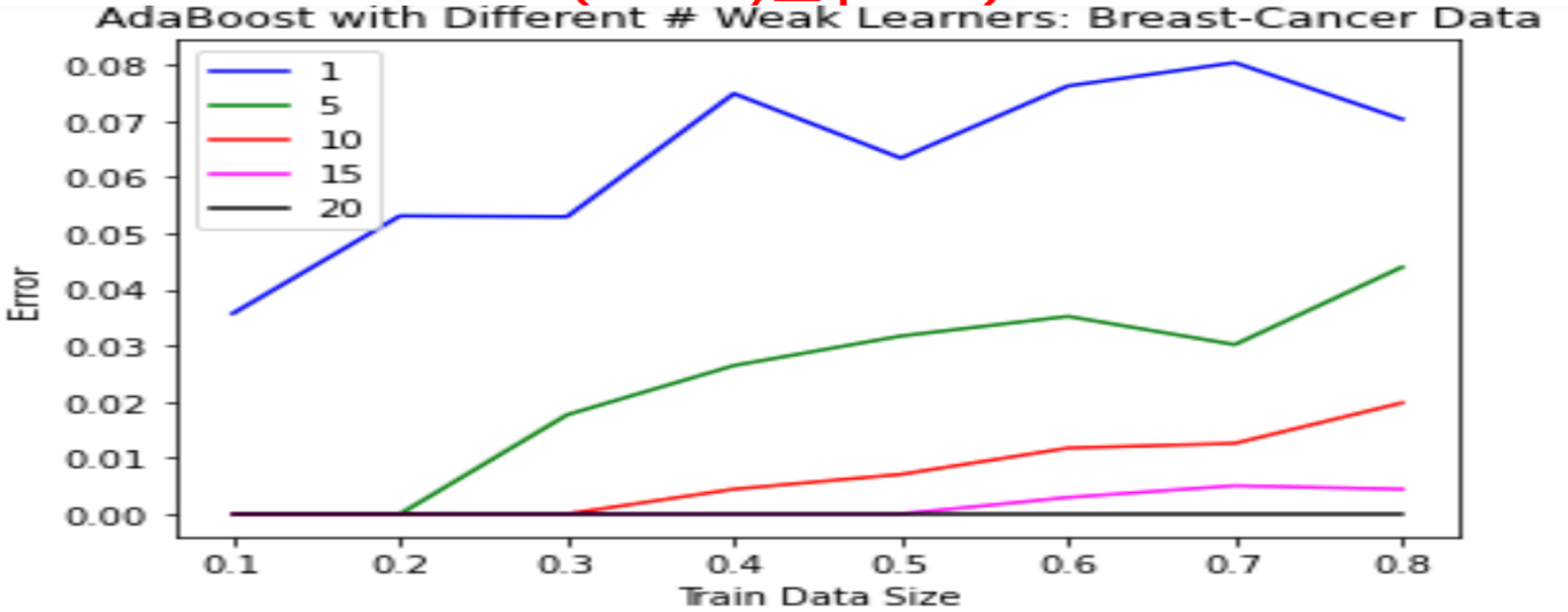


Robert Schapire

Training Error of AdaBoost

- If advantages of weak rules over random guessing are: $\Upsilon_1, \Upsilon_2, .., \Upsilon_T$ then in-sample error (training error) of final rule is at most

$$\exp\left(-2\sum_{t=1}^{T}\gamma_t^2\right)$$




AdaBoost with Different # Weak Learners: Breast-Cancer Data
Accuracy
1.00
0.98
0.96
0.94
0.92
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
Test Data Size
20
40
60
80
100
120

Regression

- Problem: Given $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$, the problem is to learn (y =) F(x) to minimize squared loss.
- You check this model and find the model to be good but makes errors.
- There are some mistakes: $F(x_1) = 0.8$, while $y_1 = 0.9$, and $F(x_2) = 1.4$ while $y_2 = 1.3$. How to improve this model?
- **Constraints on the learners:**
  - **The model to learn F cannot change or** no change **in any parameter of F**
  - You can add another model (regression tree) to learn h, so the new prediction will be F(x) + h(x).

- **A simple solution**:
- We would like to **improve the model** such that

$$F(x_1) + h(x_1) = y_1$$
$$F(x_2) + h(x_2) = y_2$$
$$\dots$$
$$F(x_n) + h(x_n) = y_n$$

- Or **equivalently**, we want

$$h(x_1) = y_1 - F(x_1)$$
$$h(x_2) = y_2 - F(x_2)$$
$$\dots$$
$$h(x_n) = y_n - F(x_n)$$

- Can any **regression tree** h achieve this goal?

Additive Models

- **Some regression tree** might be able to do this **approximately.**
- Just fit a regression tree h to data

$$(x_1, y_1 - F(x_1)), (x_2, y_2 - F(x_2)), \ldots, (x_n, y_n - F(x_n))$$

$y_i - F(x_i)$ are residuals. These are the parts that existing model F cannot do well.

- The role of h is to compensate the shortcoming of the existing F. If the new **model F + h is still not satisfactory**, we can **add another regression tree** …
- They improve the predictions on the training data, but will the procedure also **work on test data?**
- We are building a model, and the model can be **applied on test data** also.
- How is this related to **gradient descent**?

Gradient Boosting

- Gradient boosting: train **many models sequentially**.
- Each new model minimizes further the loss function (y = ax + b + e, e needs special attention as it is an error term) of the whole system using **gradient descent method**.
- The learning procedure consecutively fits new models to provide a more accurate estimate of the **response variable, y.**
- The principle idea behind this algorithm is to construct **new base learners** which can be **maximally correlated with negative gradient** of the loss function, associated with the whole ensemble.

Gradient Boosting

- Problem- set of variables x_1, x_2, and x_3 . We need to predict $y \in \mathbb{R}$
- Gradient Boost Algorithm

1. Start with mean to predict using all variables.
2. Calculate deviation of each observation from the mean.
3. Find the variable to split the deviations better and find the value for the split. This is assumed to be the latest.
4. Calculate errors of each observation from the mean in both sides of split.
5. Repeat steps 3 and 4 till the objective function minimizes.
6. The final predictor is a weighted mean of all the classifiers.

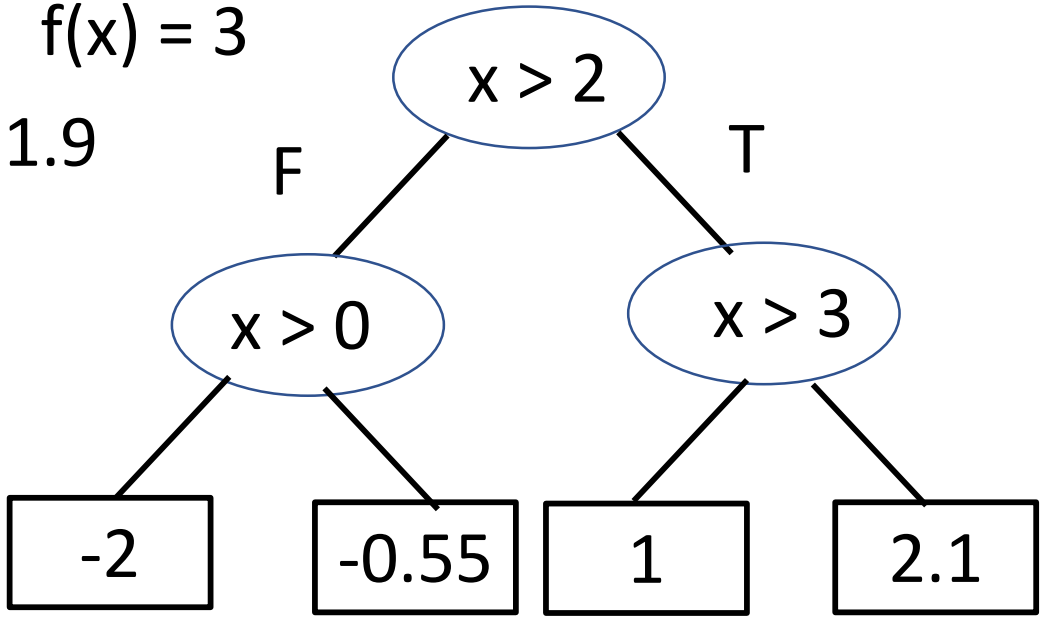
Gradient Boosting: Example

x	y
0	1
1	2
2	2.9
3	4
4	5.1

x	Y-f(x)-h(x)
0	0
1	-0.45
2	0.45
3	0
4	0

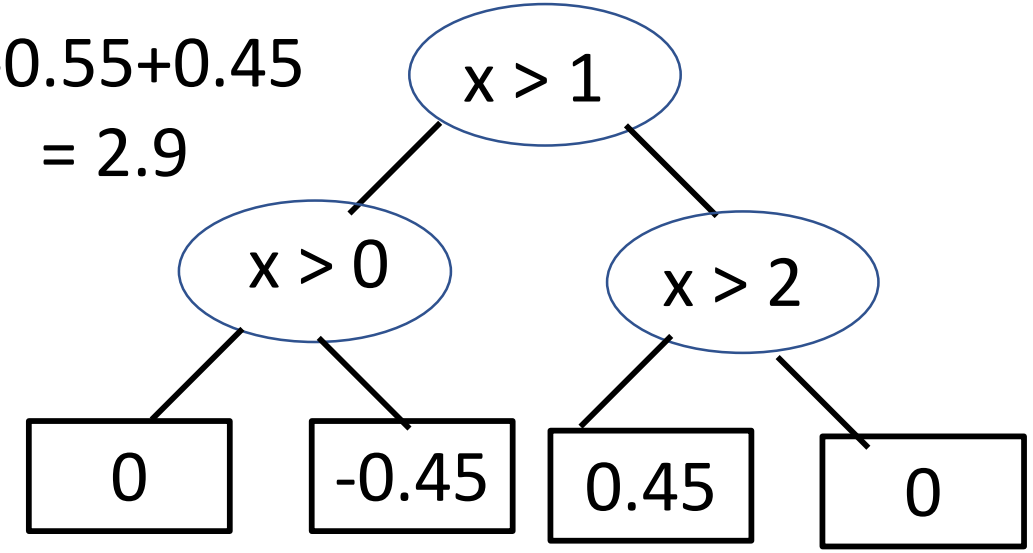
f(x) = 3

1.9



3 -0.55+0.45

= 2.9



x	Y-f(x)
0	-2
1	-1
2	-0.1
3	1
4	2.1

x	g(x)
0	0
1	-0.45
2	0.45
3	0
4	0

Gradient Descent

- Minimize a function by going in the negative direction of the gradient.

$$\theta_i := \theta_i - \rho \frac{\partial J}{\partial \theta_i}$$

- How is this related to gradient descent?

- Loss function $L(y, F(x)) = (y - F(x))^2 / 2$

- We want to minimize $J = \sum_i L(y_i, F(x_i))$ by adjusting $F(x_1)$, $F(x_2)$, …, $F(x_n)$

- Note that $F(x_1)$, $F(x_2)$, …, $F(x_n)$ are some numbers.

Gradient Descent

- We can view as follows: We can treat $F(x_i)$ as parameters and take derivatives

$$\frac{\partial J}{\partial F(x_i)} = \frac{\partial \sum_i L\left(y_i, F(x_i)\right)}{\partial F(x_i)} = \frac{\partial L\left(y_i, F(x_i)\right)}{\partial F(x_i)} = F(x_i) - y_i$$

- We can see that y_i - $F(x_i)$ as the negative of the gradient, that is

$$y_i - F(x_i) = - \frac{\partial J}{\partial F(x_i)}$$

- This means $F(x_i) := F(x_i) + h(x_i) = F(x_i) + y_i - F(x_i)$

- Hence $F(x_i) := F(x_i) - 1. \frac{\partial J}{\partial F(x_i)}$

- It is of the form

$$\theta_i := \theta_i - \rho \frac{\partial J}{\partial \theta_i}$$ where $\Theta_i = F(x_i)$ is the parameter

Gradient Boost

- For regression with squared loss,

residual $\Leftrightarrow$ negative gradient

h fits the residual $\Leftrightarrow$ h fits the negative gradient

Update F based on residual $\Leftrightarrow$ Update F based on negative gradient

- So, the model is updated using gradient descent!

- It is known that the concept of gradients is more general and useful than the concept of residuals.

- So, the name of gradient boost

Random Forest

- Decision Tree classifier is greedy and *computationally expensive* to deal with high dimensional data sets.
- *Random forest* (or random forests) is an ensemble classifier that consists of *many decision trees* and outputs the class that is the majority class label output by individual trees.
- If there are N data points and M features, then select S1, S2, …, Sk where |Si| = N, select with replacement, k >=100. Build k decision trees
- At each node of a decision tree, choose *m <<M features* and find the best feature.
- *Decide based on majority class label out of k outputs.*

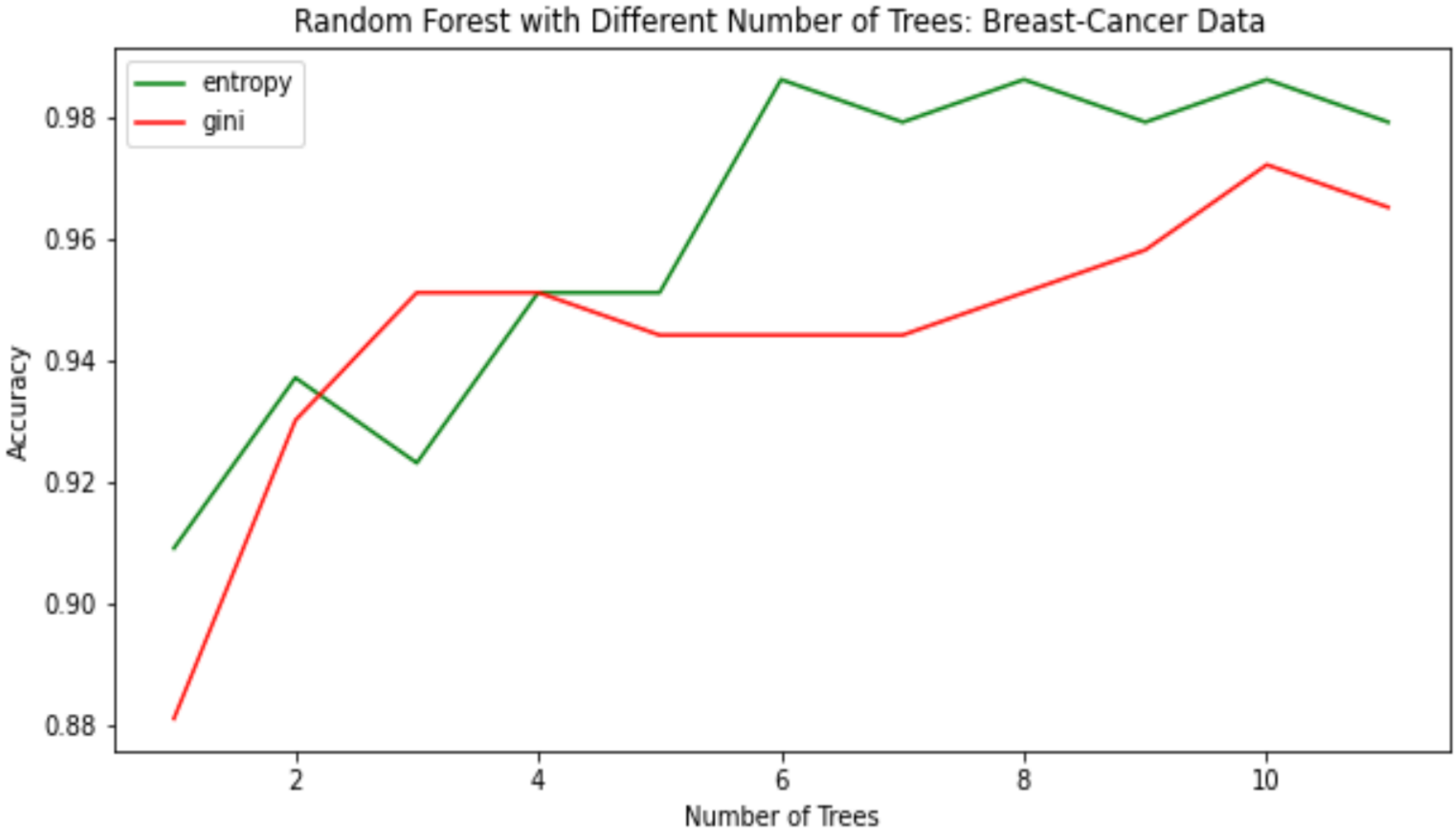
Random Forest: Strength and Correlation

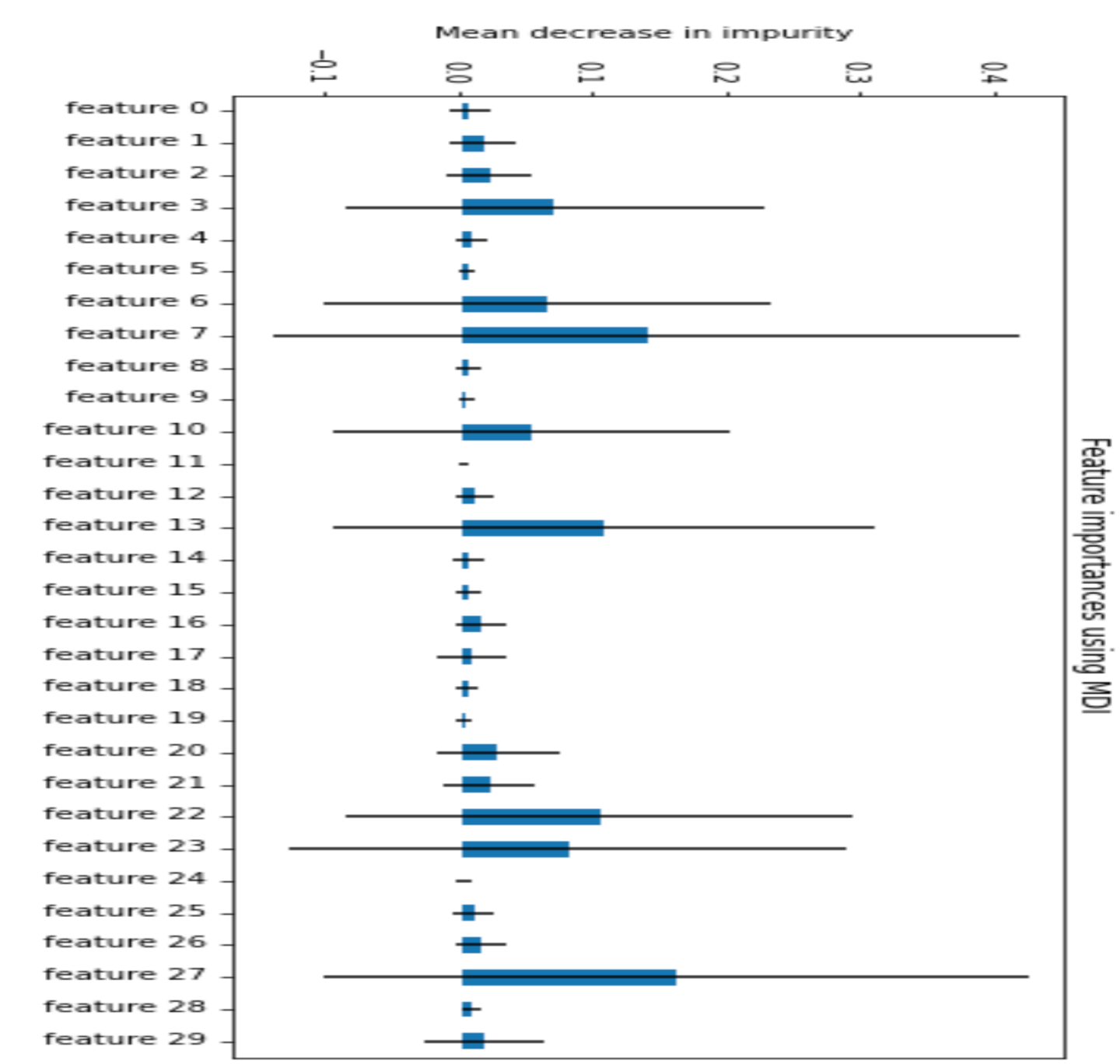
If s is the average accuracy of a classifier and $\bar{\rho}$ is the average correlation of the k decision trees, then the average probability of error PE* is given by

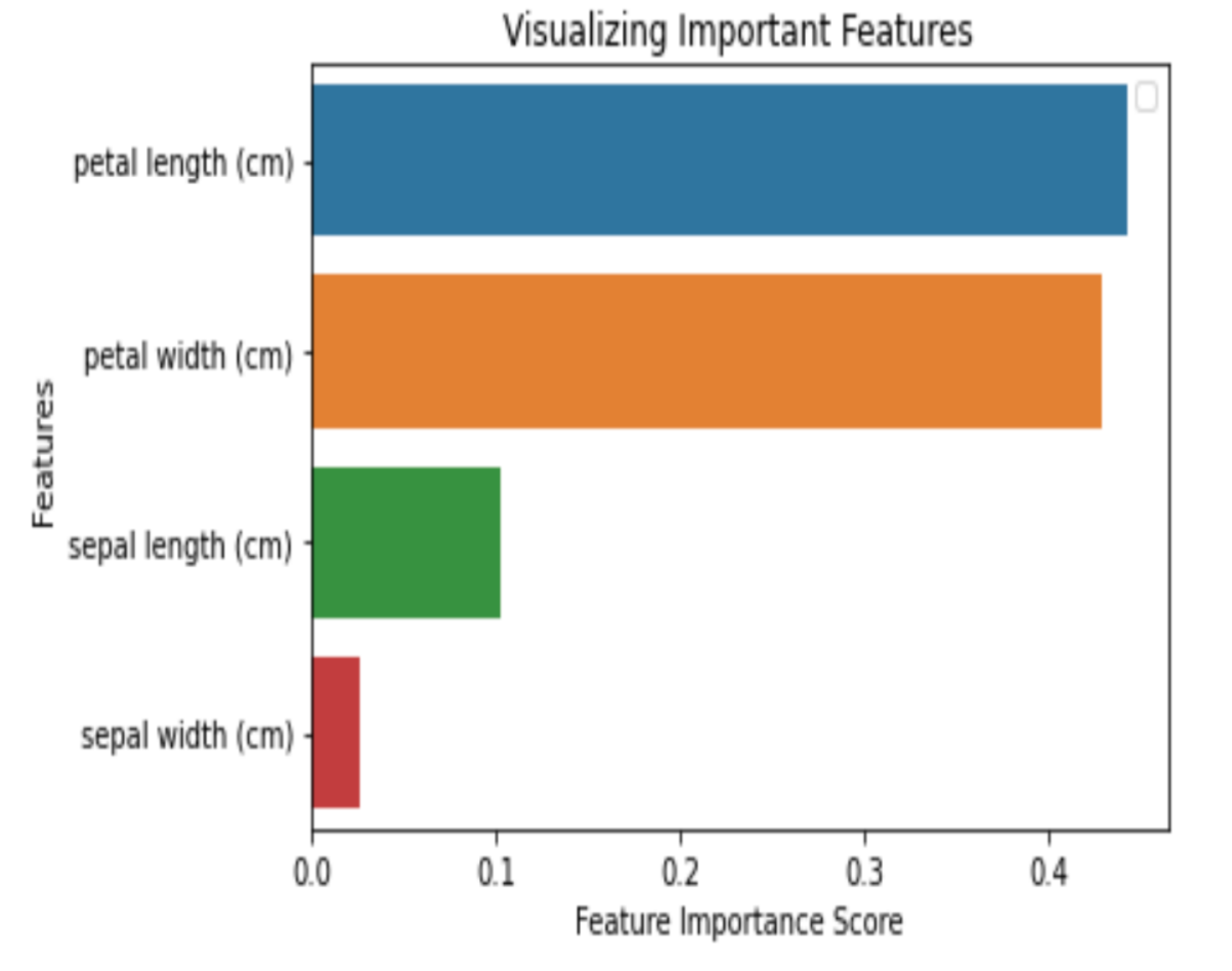
$$PE^* \leq \bar{\rho}\,(1- s^2)/s^2$$

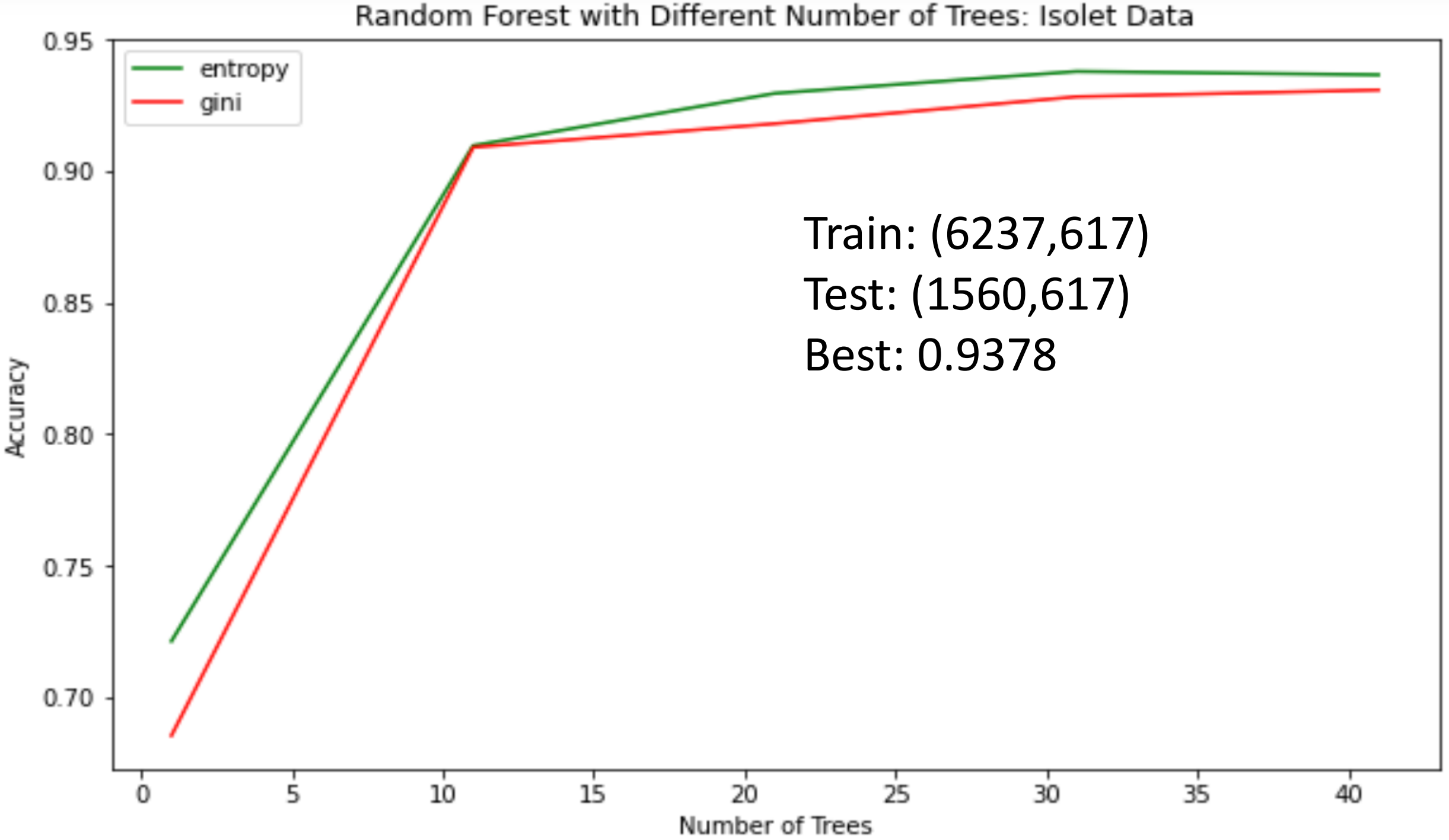
So, we require s to be close to 1 and $\bar{\rho}$ to be small.

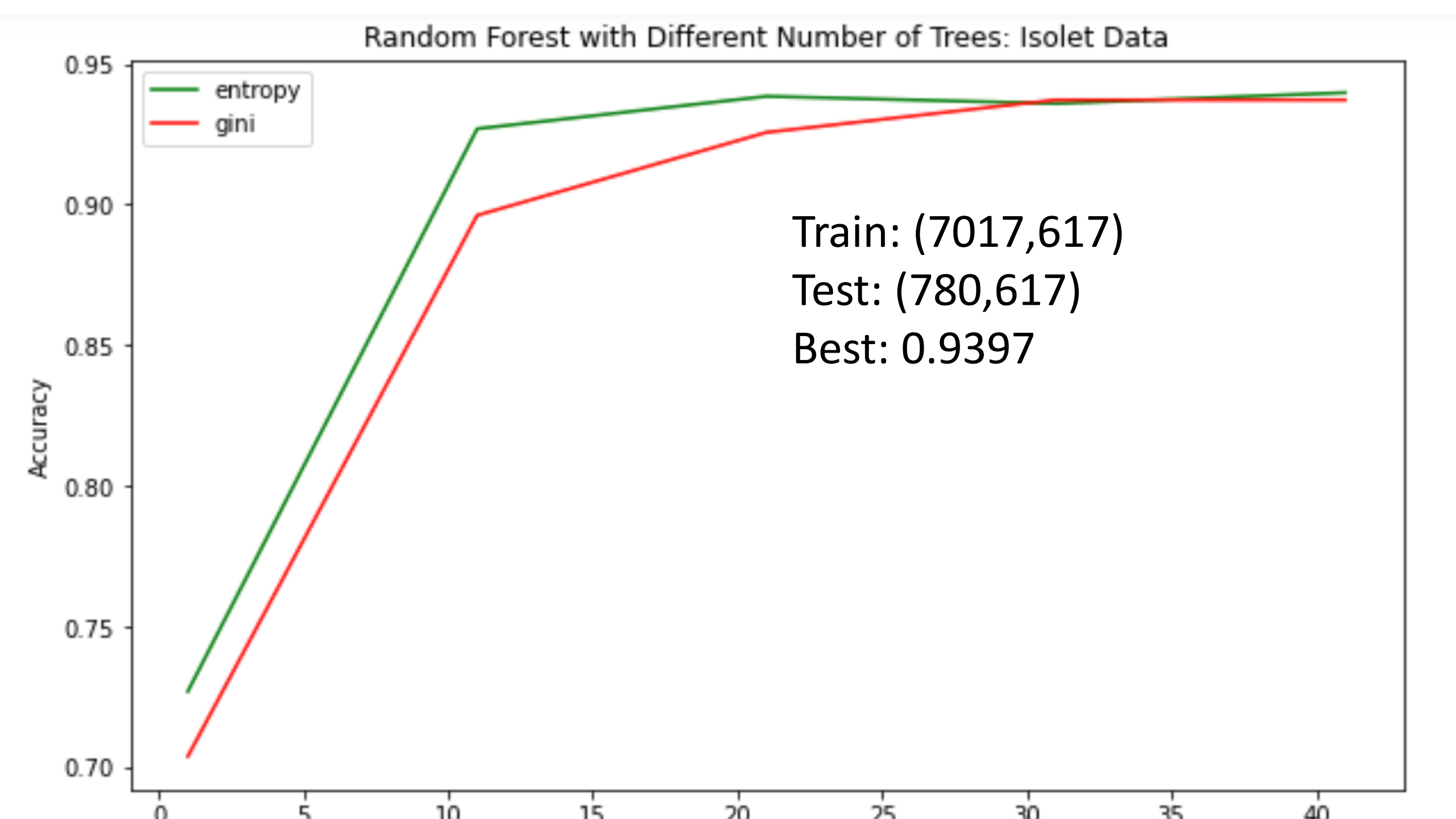
Leo Brieman, Random Forests, Machine Learning, Volume-45, 2001.


Random Forest with Different Number of Trees: Breast-Cancer Data
entropy
gini
0.98
0.96
0.94
0.92
0.90
0.88
Accuracy
2
4
6
8
10
Number of Trees


Mean decrease in impurity
−0.1
0.0
0.1
0.2
0.3
0.4
feature 0
feature 1
feature 2
feature 3
feature 4
feature 5
feature 6
feature 7
feature 8
feature 9
feature 10
feature 11
feature 12
feature 13
feature 14
feature 15
feature 16
feature 17
feature 18
feature 19
feature 20
feature 21
feature 22
feature 23
feature 24
feature 25
feature 26
feature 27
feature 28
feature 29
Feature importances using MDI


Visualizing Important Features
petal length (cm)
petal width (cm)
sepal length (cm)
sepal width (cm)
Features
0.0
0.1
0.2
0.3
0.4
Feature Importance Score


Random Forest with Different Number of Trees: Isolet Data
entropy
gini
Train: (6237,617)
Test: (1560,617)
Best: 0.9378
Accuracy
Number of Trees
0.95
0.90
0.85
0.80
0.75
0.70
0
5
10
15
20
25
30
35
40


Random Forest with Different Number of Trees: Isolet Data
entropy
gini
Train: (7017,617)
Test: (780,617)
Best: 0.9397
Accuracy
0.95
0.90
0.85
0.80
0.75
0.70
0
5
10
15
20
25
30
35
40

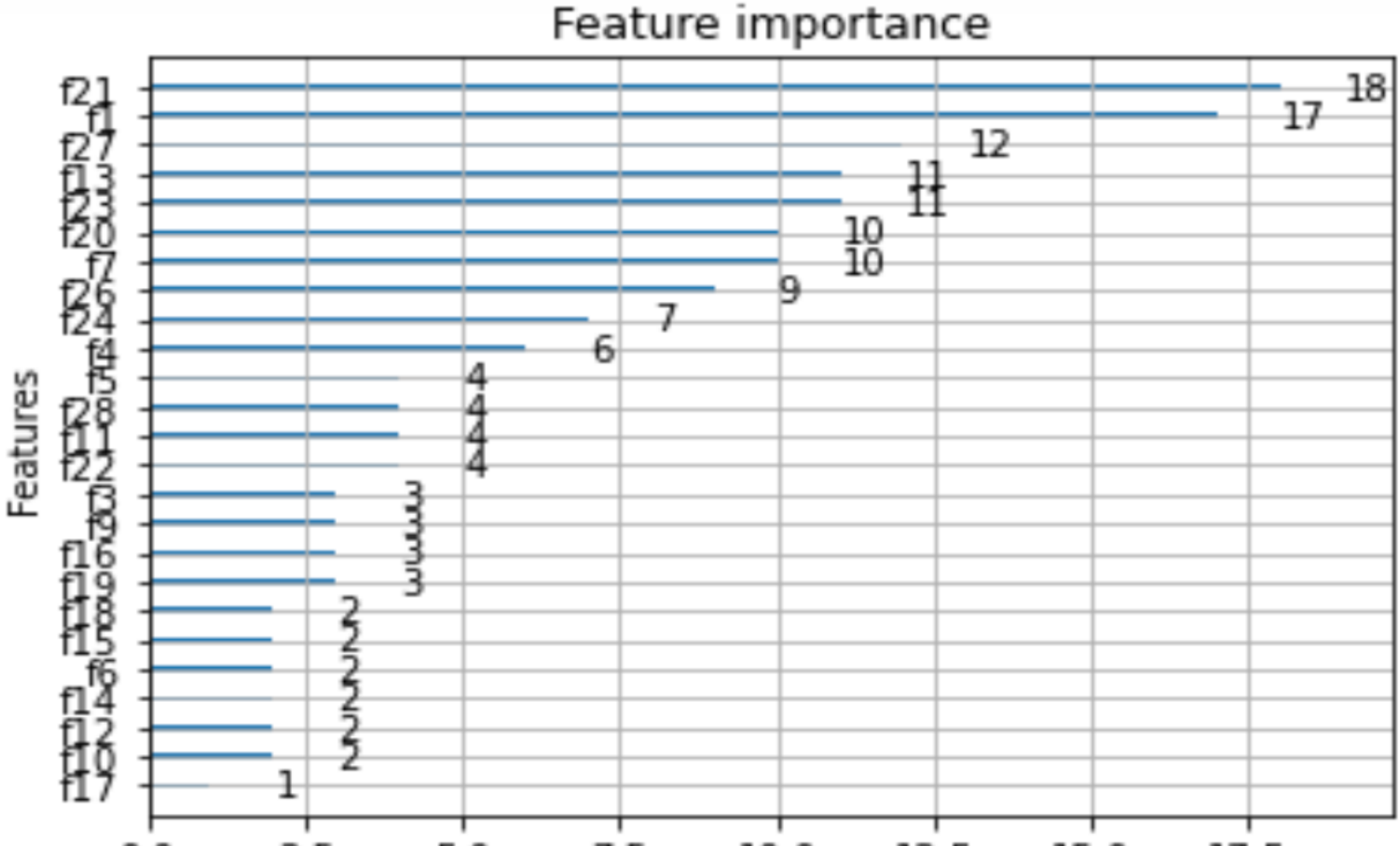
Random Forest: Practical Considerations

- Splits are chosen according to a purity measure:
  - E.g. squared error (regression), Gini index or entropy(classification)
- How to select number of trees or forest size?
  - Build trees until the error no longer decreases
- How to select M?
  - Try to recommend defaults, half of them and twice of them and pick the best.

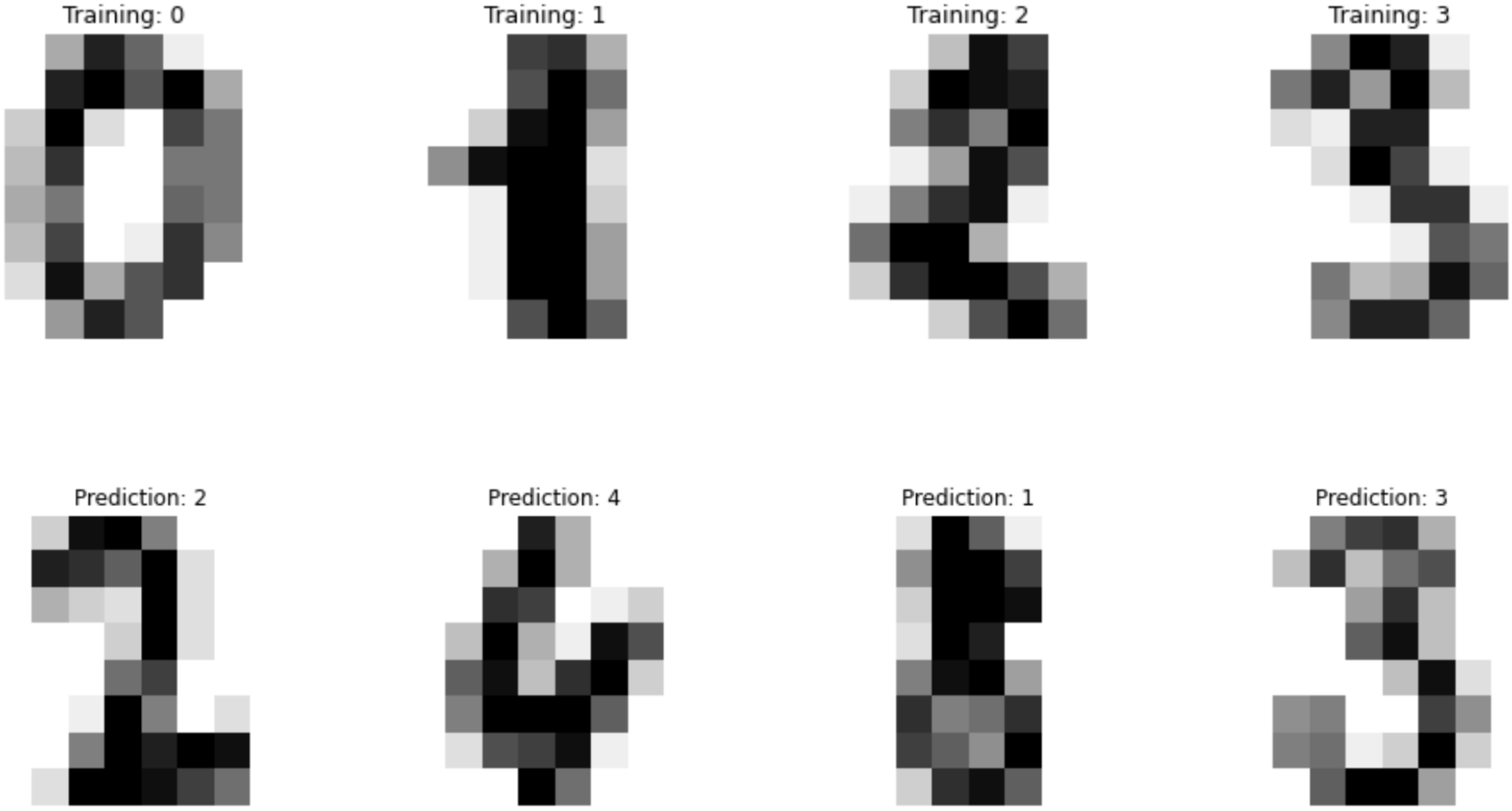
Features and Advantages

The advantages of random forest are:

- It is one of the most accurate learning algorithms available. For many data sets, it produces a highly accurate classifier.
- It runs efficiently on large databases.
- It can handle thousands of input variables without variable deletion.
- It gives estimates of what variables are important in the classification.
- It generates an internal unbiased estimate of the generalization error as the forest building progresses.
- It has an effective method for estimating missing data and maintains accuracy when a large proportion of the data are missing.


Feature importance
Features
f21
f1
f27
f13
f23
f20
f7
f6
f24
f4
f8
f22
f3
f16
f19
f15
f5
f18
f14
f12
f10
f17
18
17
12
11
11
10
10
9
7
6
4
4
4
4
3
3
3
2
2
2
2
2
2
1

Example Digit Patterns: XGBoost Classifier



Random Forest

- Decision Tree classifier is greedy and *computationally expensive* to deal with high dimensional data sets.
- *Random forest* (or random forests) is an ensemble classifier that consists of *many decision trees* and outputs the class that is the majority class label output by individual trees.
- If there are N data points and D features, then select S1, S2, …, Sk where |Si| = N, select with replacement, k >=100. Build k decision trees
- At each node of a decision tree, choose *d << D features* and find the best feature.
- *Decide based on majority class label out of k outputs.*

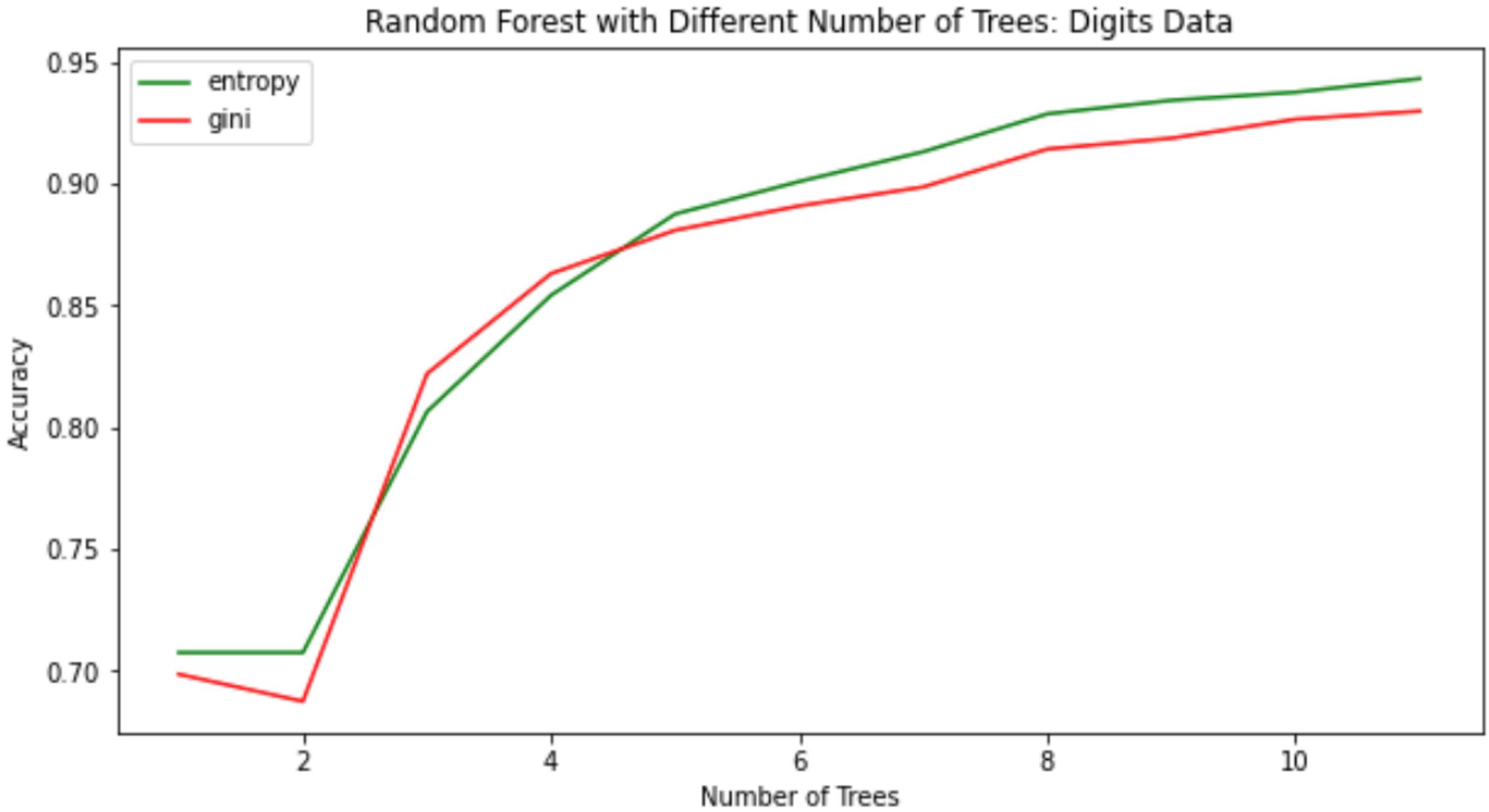
Random Forest: Strength and Correlation

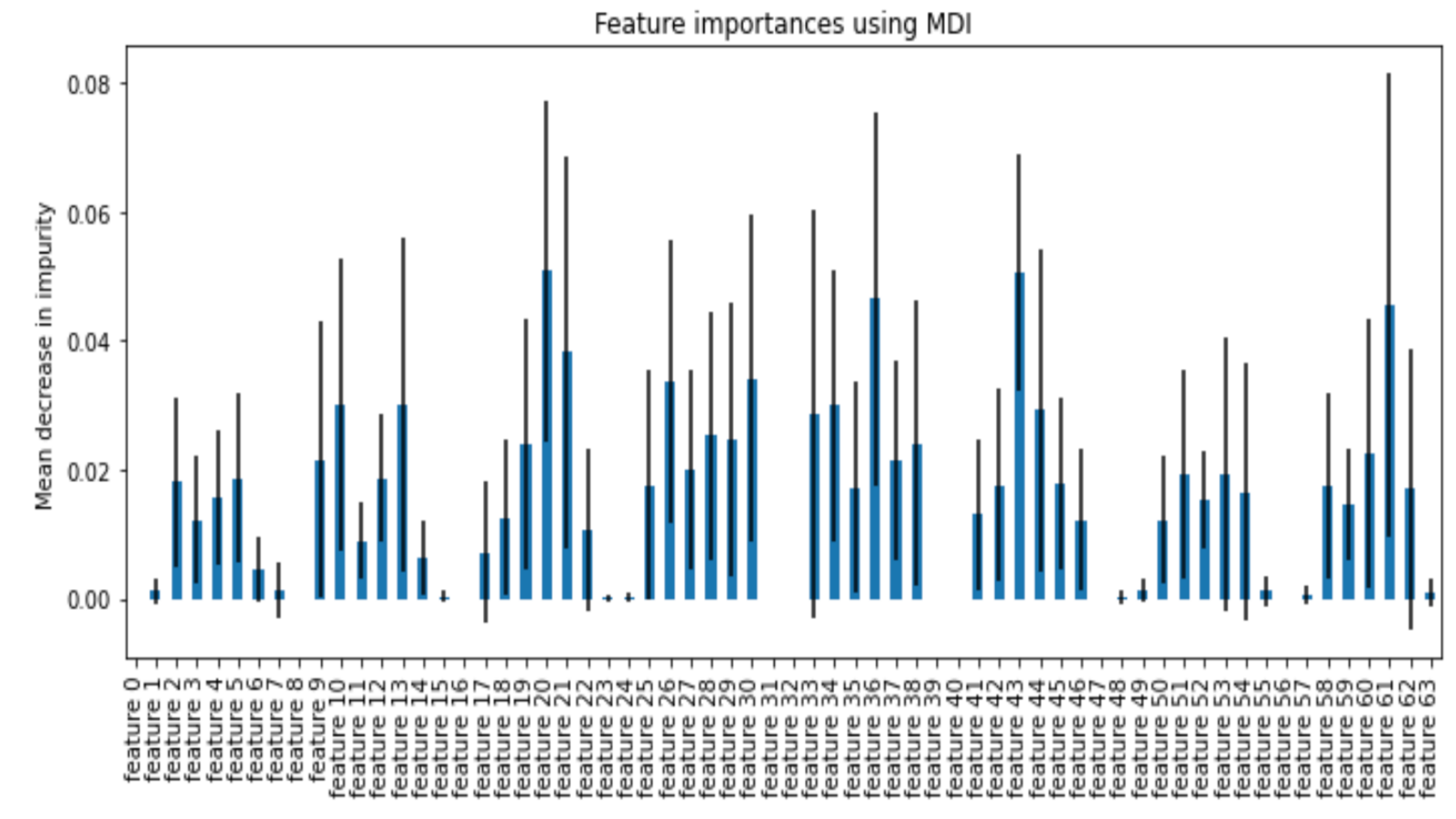
If s is the average accuracy of a classifier and $\bar{\rho}$ is the average correlation of the k decision trees, then the average probability of error PE* is given by

$$PE^* \leq \bar{\rho}\,(1- s^2)/s^2$$

So, we require s to be close to 1 and $\bar{\rho}$ to be small.

Leo Brieman, Random Forests, Machine Learning, Volume-45, 2001.


Random Forest with Different Number of Trees: Digits Data
entropy
gini
0.95
0.90
0.85
0.80
0.75
0.70
Accuracy
2
4
6
8
10
Number of Trees


Feature importances using MDI
Mean decrease in impurity
0.08
0.06
0.04
0.02
0.00
feature 0
feature 1
feature 2
feature 3
feature 4
feature 5
feature 6
feature 7
feature 8
feature 9
feature 10
feature 11
feature 12
feature 13
feature 14
feature 15
feature 16
feature 17
feature 18
feature 19
feature 20
feature 21
feature 22
feature 23
feature 24
feature 25
feature 26
feature 27
feature 28
feature 29
feature 30
feature 31
feature 32
feature 33
feature 34
feature 35
feature 36
feature 37
feature 38
feature 39
feature 40
feature 41
feature 42
feature 43
feature 44
feature 45
feature 46
feature 47
feature 48
feature 49
feature 50
feature 51
feature 52
feature 53
feature 54
feature 55
feature 56
feature 57
feature 58
feature 59
feature 60
feature 61
feature 62
feature 63

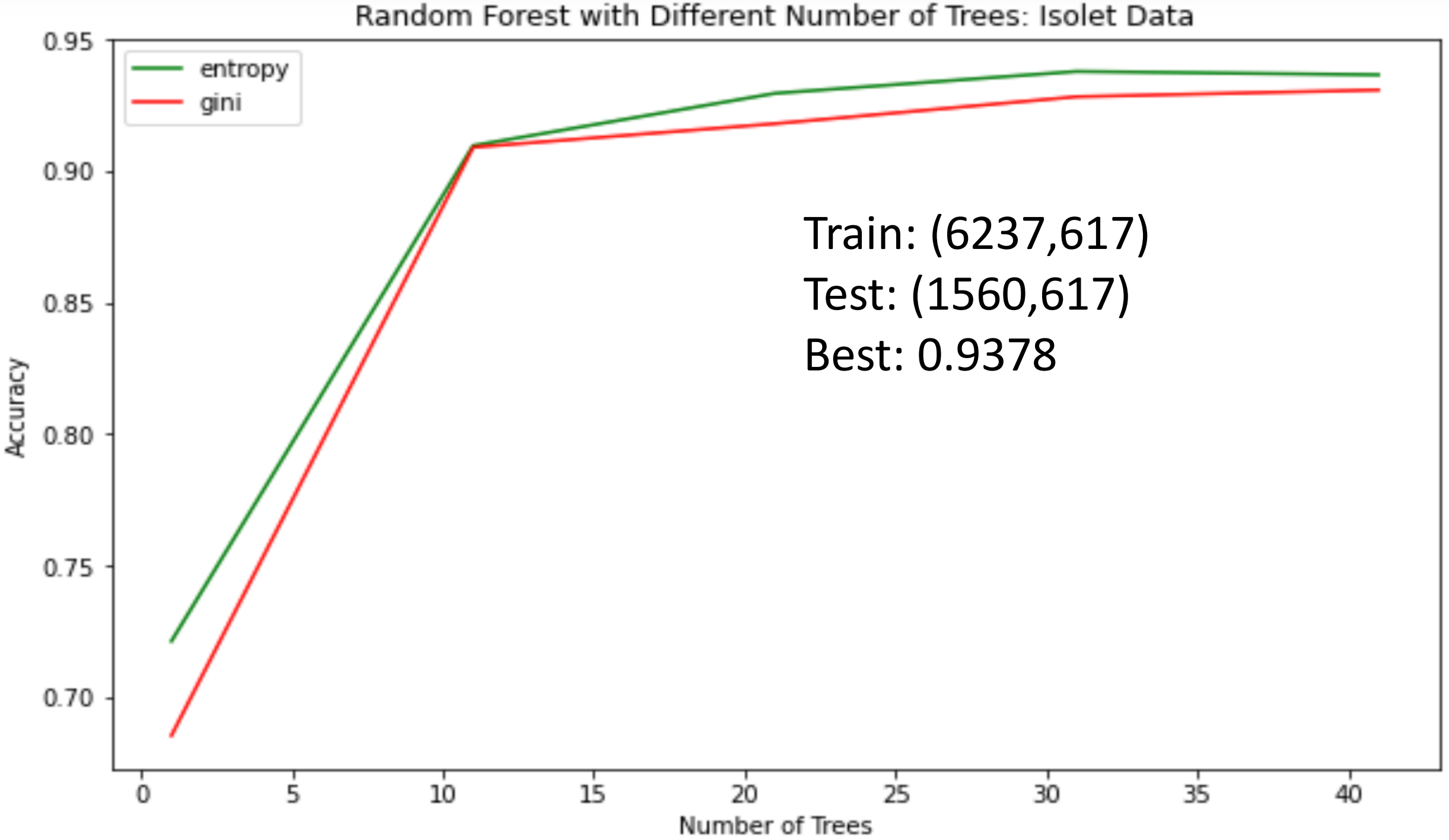
Confusion Matrices: RF and XGB Classifiers

```
[[92  0  0  0  0  0  0  0  0  0]
 [ 0 92  0  0  0  0  0  0  0  1]
 [ 0  0 86  0  0  0  0  0  4  0]
 [ 1  0  0 70  0  3  1  1  3  2]
 [ 1  0  0  0 86  0  1  2  0  0]
 [ 3  2  0  2  1 75  1  0  1  0]
 [ 1  0  0  1  1  0 79  0  0  0]
 [ 0  0  0  0  0  0  0 94  0  0]
 [ 1  6  0  3  1  0  1  3 75  0]
 [ 0  2  0  4  1  4  1  3  0 87]]
```

Random Forest Classifier

```
[[94  0  0  0  0  0  0  0  1  0]
 [ 0 86  0  1  0  1  0  0  0  1]
 [ 2  0 88  1  0  0  0  0  1  0]
 [ 0  1  0 86  0  2  0  2  2  1]
 [ 0  1  0  0 92  0  0  2  0  0]
 [ 0  0  0  0  0 86  0  0  0  0]
 [ 0  2  0  0  0  0 89  0  2  0]
 [ 0  1  0  0  0  0  0 87  0  1]
 [ 1  2  0  1  0  0  0  1 77  0]
 [ 0  4  0  0  0  0  0  1  1 78]]
```

XGBoost Classifier


Random Forest with Different Number of Trees: Isolet Data
entropy
gini
Train: (6237,617)
Test: (1560,617)
Best: 0.9378
Accuracy
Number of Trees
0.95
0.90
0.85
0.80
0.75
0.70
0
5
10
15
20
25
30
35
40

Random Forest: Practical Considerations

- Splits are chosen according to a purity measure:
  - E.g. squared error (regression), Gini index or entropy(classification)
- How to select number of trees or forest size?
  - Build trees until the error no longer decreases
- How to select d?
  - Try to recommend defaults, half of them and twice of them and pick the best.

Features and Advantages

The advantages of random forest are:

- It is one of the most accurate learning algorithms available. For many data sets, it produces a highly accurate classifier.
- It runs efficiently on large databases.
- It can handle thousands of input variables without variable deletion.
- It gives estimates of what variables are important in the classification.
- It generates an internal unbiased estimate of the generalization error as the forest building progresses.
- It has an effective method for estimating missing data and maintains accuracy when a large proportion of the data are missing.