Data Science

Lecture 4: Structured Data Processing

(Decision Tree and Random Forest)

UNIVERSITY OF AMSTERDAM

Lecturer: Yen-Chia Hsu

Date: Feb 2023

This lecture will introduce the Decision Tree and Random Forest models.

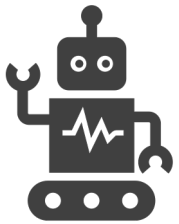
Recall that in the structured data processing module, we used air quality sensor data in the past to predict the presence of bad odors in the future.

O_3: 26 ppb	CO: 127 ppb
H_2S: 0 ppb	$PM_{2.5}$: 9 μg/m^3
Wind: 17 deg	…

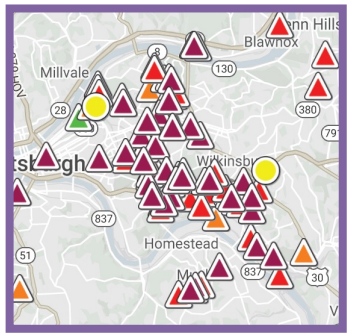
Observation 1

O_3: 1 ppb	CO: 1038 ppb
H_2S: 9 ppb	$PM_{2.5}$: 23 μg/m^3
Wind: 213 deg	…

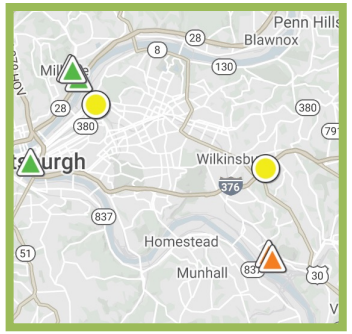
Observation 2



Machine Learning



Has Event



No Event

Structured data generally means the type of data that has standardized formats and well-defined structures (i.e., those that can be stored in a relational database).

- Samples of Citizen-Contributed Smell Reports

EpochTime	feelings_symptoms	smell_description	smell_value	zipcode
…	…	…	…	…
1478353854	Headache, sinus, seeping into house even though it is as shut and sealed as possible. Air purifiers are unable to handle it thoroughly.	Industrial, acrid, strong	4	15206
1478354971		Industrial	4	15218
…	…	…	…	…

These are technically not structured.

- Samples of Air Quality Sensor Measurements

EpochTime	3.feed_28.H2S_PPM	3.feed_28.SO2_PPM	3.feed_28.SIGTHETA_DEG	3.feed_28.SONICWD_DEG	3.feed_28.SONICWS_MPH
…	…	…	…	…	…
1478046600	0,019	0,020	14,0	215,0	3,2
1478050200	0,130	0,033	13,4	199,0	3,4
…	…	…	…	…	…

Mathematically speaking, we want to estimate a function f that can map feature X to label y such that the prediction $f(X)$ is close to y as much as possible.

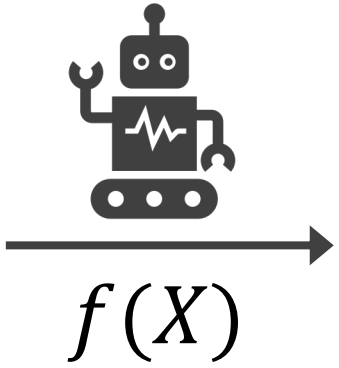
O_3	CO	…	…	Wind
26 ppb	127 ppb	…	…	17 deg

O_3
26 ppb
10 ppb
⋮
⋮
21 ppb

observation: $x^{(1)}$

$$X = \begin{bmatrix} 1 & x_1^{(1)} & x_2^{(1)} & \cdots & \cdots & x_p^{(1)} \\ 1 & x_1^{(2)} & x_2^{(2)} & \cdots & \cdots & x_p^{(2)} \\ \vdots & \vdots & \vdots & \cdots & \cdots & \vdots \\ \vdots & \vdots & \vdots & \cdots & \cdots & \vdots \\ 1 & x_1^{(n)} & x_2^{(n)} & \cdots & \cdots & x_p^{(n)} \end{bmatrix}$$

feature: x_1



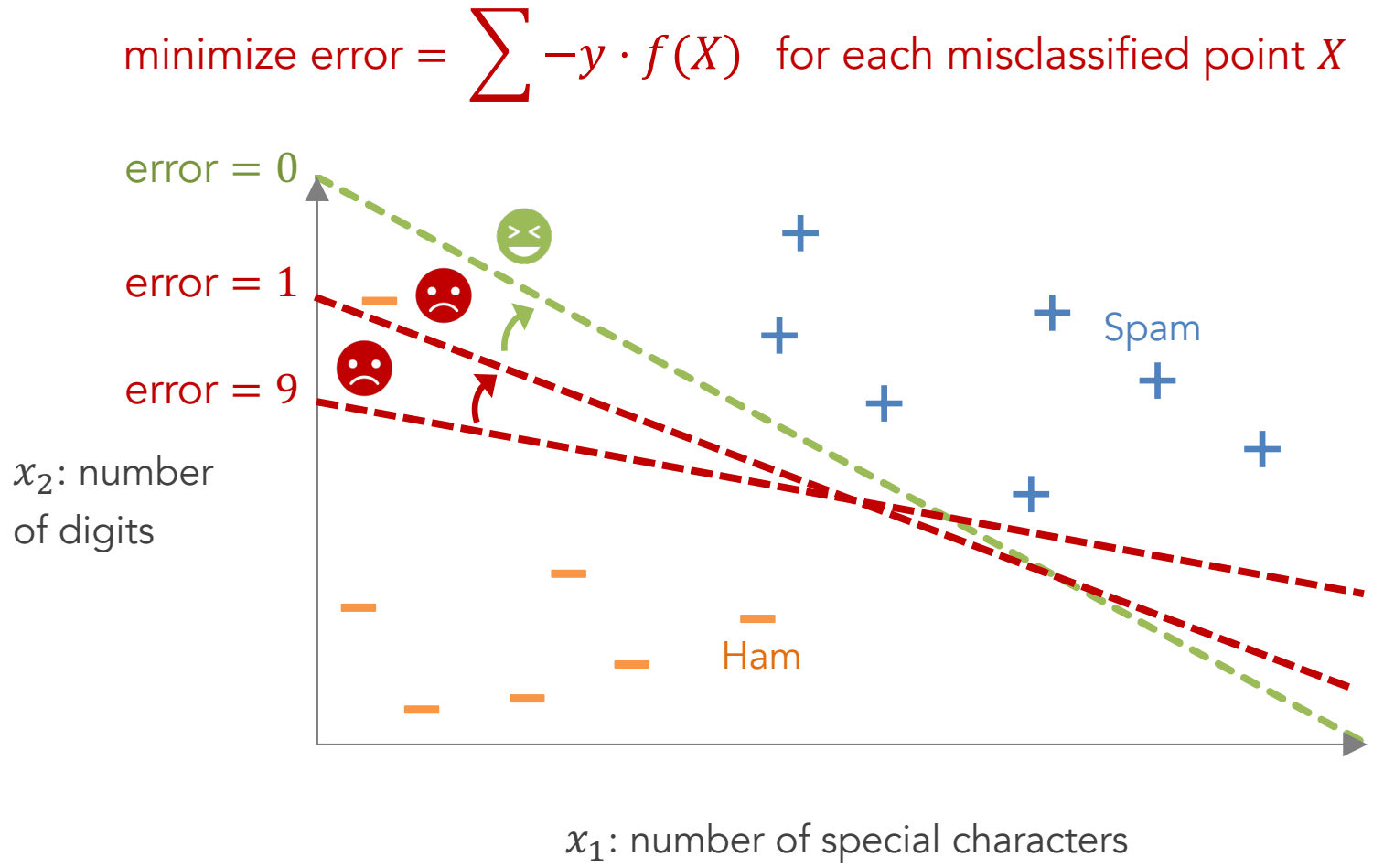
$f(X)$

$$\begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ \vdots \\ y^{(n)} \end{bmatrix} = y$$

label: y

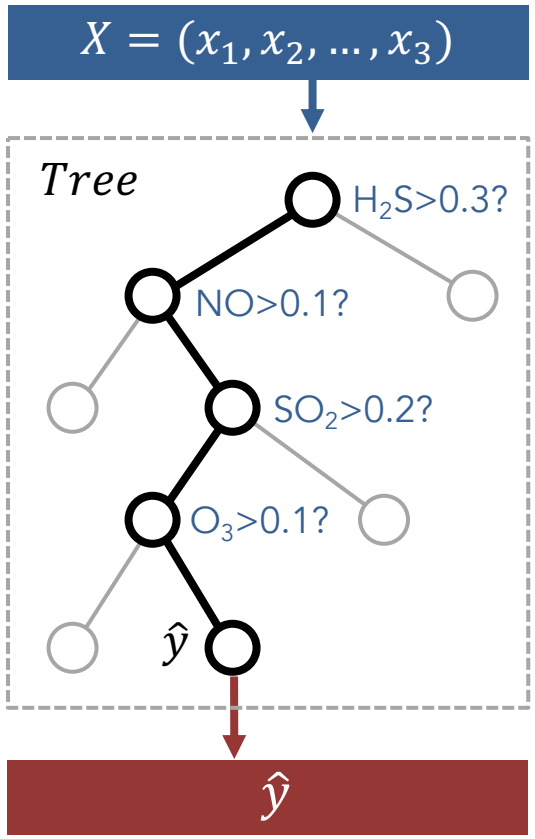
Has event?
no
yes
⋮
⋮
no

We have learned the concept of training a linear classifier using a general optimization algorithm to minimize an error metric to classify spam and ham (non-spam) messages.



Rosenblatt's Perceptron Learning Algorithm: section 4.5.1 in book https://hastie.su.domains/ElemStatLearn/

Not every model is trained using a general optimization algorithm. In this lecture, we will introduce Decision Tree, which has a different training mechanism.

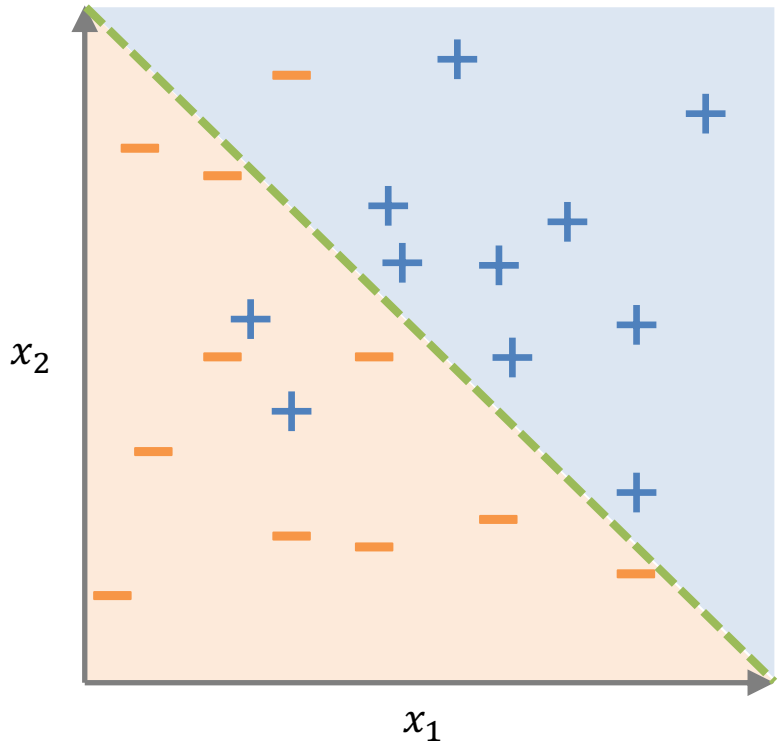


PM, SO_2, CO, NO, NO_2, O_3, H_2S, and wind information

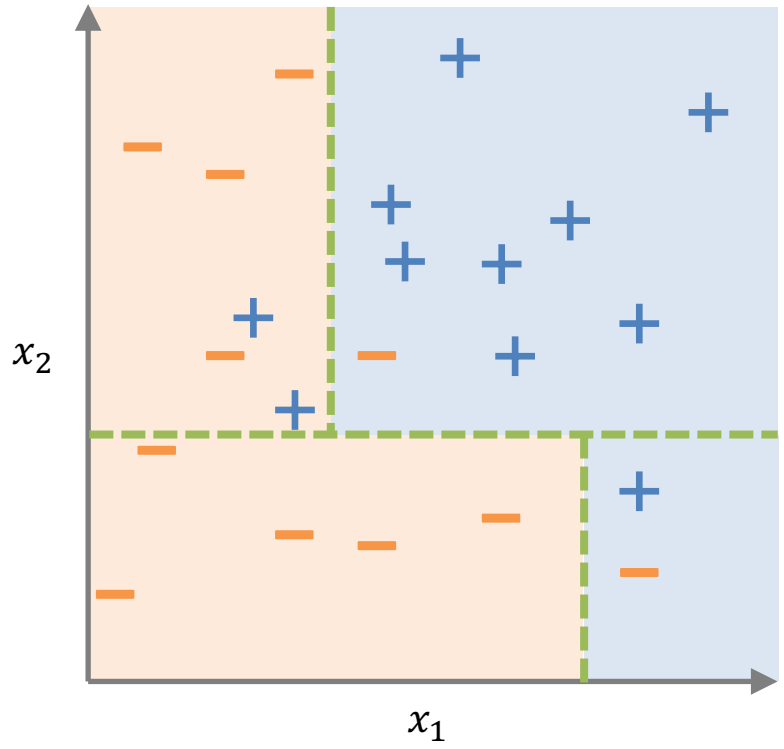
Prediction of bad smell (yes/no)

Unlike the linear classifier (which has a linear decision boundary), Decision Tree has a non-linear decision boundary that iteratively partitions the feature space.

Linear Decision Boundary

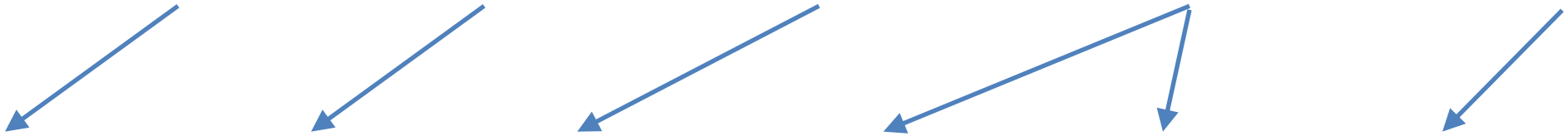


Non-linear Decision Boundary



For simplicity, assume that all features are binary. We can use domain knowledge to determine if H_2S can be detected by human nose (e.g., larger than 0.03 ppm).

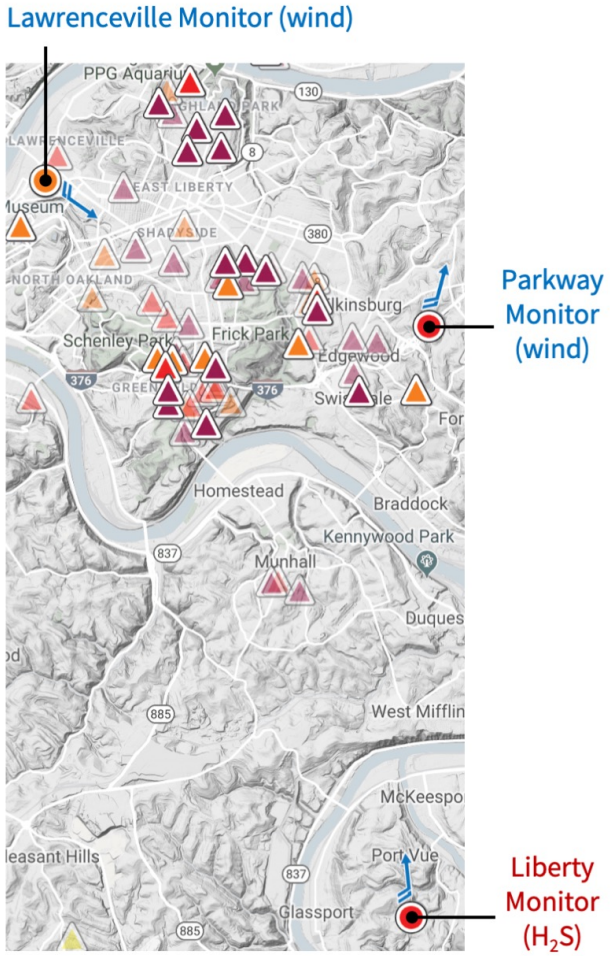
EpochTime	3.feed_28.H2S_PPM	3.feed_28.SO2_PPM	3.feed_28.SIGTHETA_DEG	3.feed_28.SONICWD_DEG	3.feed_28.SONICWS_MPH
…	…	…	…	…	…
1478046600	0,019	0,020	14,0	215,0	3,2
1478050200	0,130	0,033	13,4	199,0	3,4
…	…	…	…	…	…



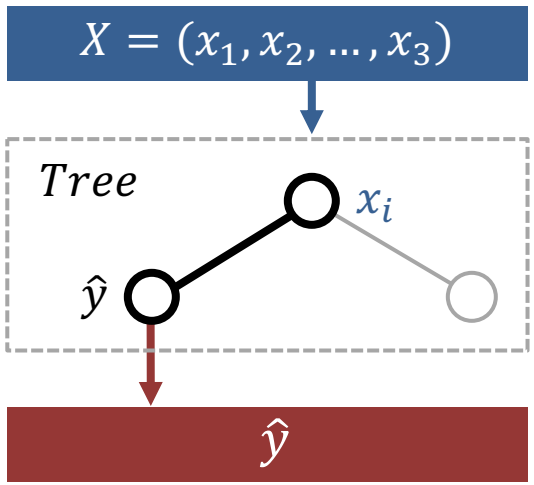
Is H_2S noticeable (sensor 28)?	Is SO_2 noticeable (sensor 28)?	Wind is turbulent (sensor 28)?	Is south wind (sensor 28?)	Is east wind (sensor 28)?	Is wind calm (sensor 28)?	…
…	…	…	…		…	…
No	No	No	Yes	No	Yes	…
Yes	No	No	Yes	No	Yes	…
…	…	…	…		…	…

In order to guess whether the smell will be bad in the future ("yes" or "no"), suppose we are only allowed to ask binary questions in a sequence.

- Q: Is the hourly averaged H_2S reading (at the Liberty monitor) noticeable two hours ago by human nose?
- A: YES
- Q: Is the hourly averaged wind direction from south (at the Parkway monitor) one hour ago?
- A: YES
- Q: Is the hourly averaged wind direction from east (at the Lawrenceville monitor) one hour ago?
- A: NO
- We predict that bad smell will happen within 8 hours.



If we could only ask one question, which question would we ask? We want to use the most useful feature that can give us the most information to help us guess.

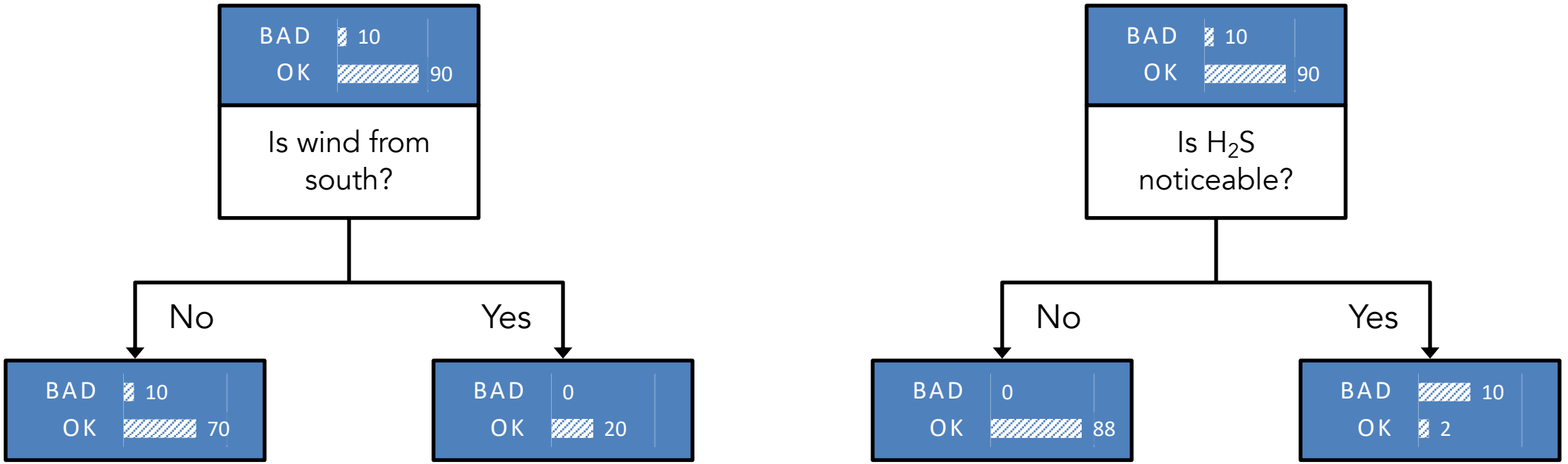


PM, SO_2, CO, NO, NO_2, O_3, H_2S, and wind information

Prediction of bad smell (yes/no)

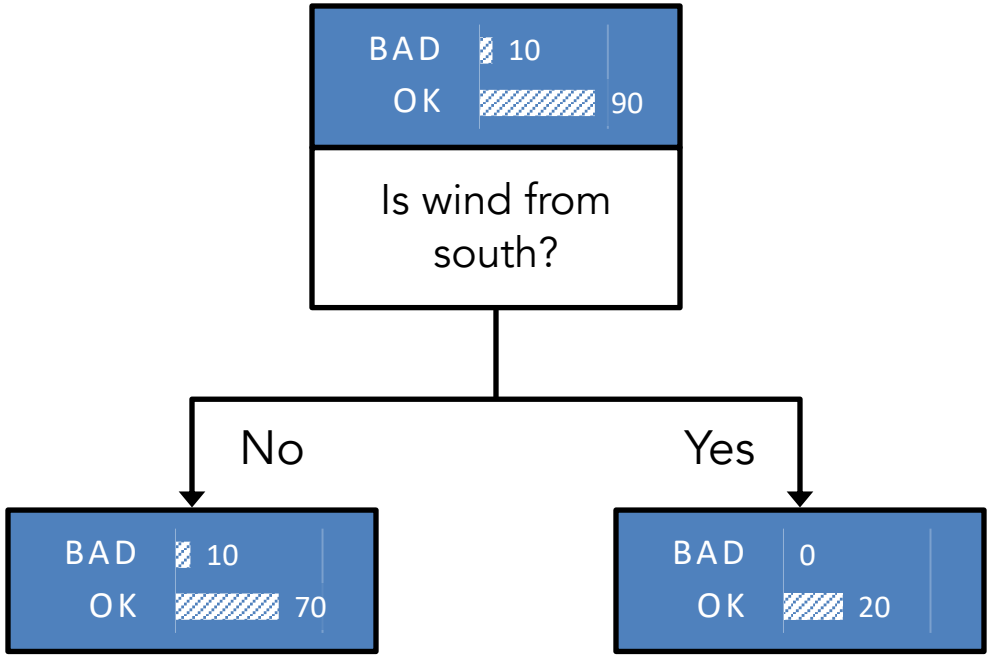
- Is PM noticeable?
- Is SO_2 noticeable?
- Is wind strong?
- Is wind turbulent?
- Is south wind?
- Is morning time?
- …

Exercise: Suppose we want to compare two features when predicting bad smell, as shown below. Which one is better? And why?



Hint: Think about intuitions and how to mathematically formulate the intuitions.

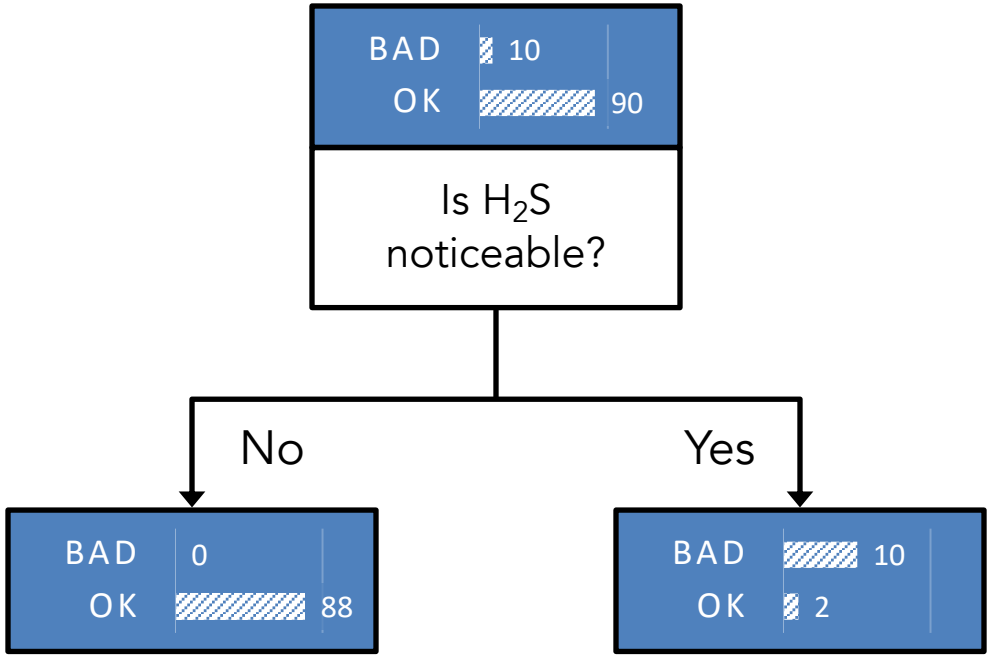
How can we quantify which feature gives the most information? We can use the misclassification error for the best guess that we can make after getting the answer.



Guess "OK"
10 errors

Guess "OK"
0 errors

Total error $= 10/100 = 0.1$

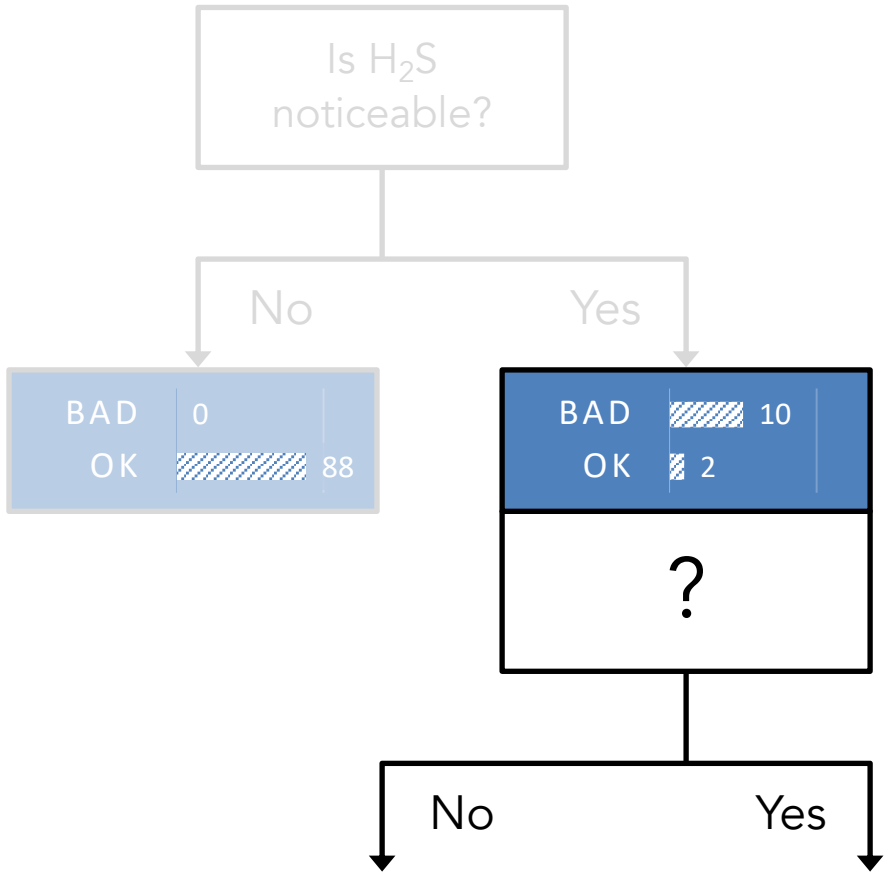


Guess "OK"
0 error

Guess "BAD"
2 error

Total error $= 2/100 = 0.02$

We can use the same strategy to iteratively select the best feature for each tree node.



Algorithm 1 DecisionTreeTrain(*data*, *remaining features*)

```
guess ← most frequent answer in data                    // default answer for this data
if the labels in data are unambiguous then
    return Leaf(guess)                                  // base case: no need to split further
else if remaining features is empty then
    return Leaf(guess)                                  // base case: cannot split further
else                                                    // we need to query more features
    for all f ∈ remaining features do
        NO ← the subset of data on which f=no
        YES ← the subset of data on which f=yes
        score[f] ← # of majority vote answers in NO
                   + # of majority vote answers in YES
                                                        // the accuracy we would get if we only queried on f
    end for
    f ← the feature with maximal score(f)
    NO ← the subset of data on which f=no
    YES ← the subset of data on which f=yes
    left ← DecisionTreeTrain(NO, remaining features \ {f})
    right ← DecisionTreeTrain(YES, remaining features \ {f})
    return Node(f, left, right)
end if
```

Algorithm source -- http://ciml.info/dl/v0_99/ciml-v0_99-ch01.pdf

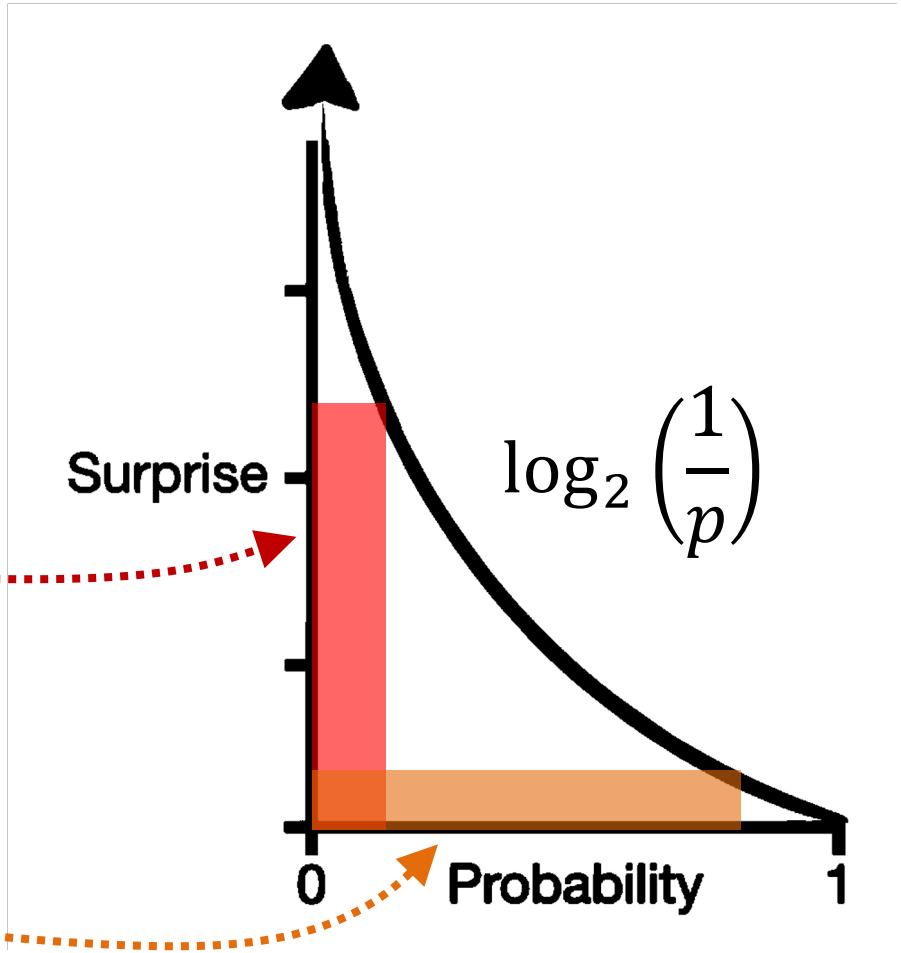
There are also other methods to quantify information, such as entropy.

Suppose we have a coin: one side has label “BAD”, and another side has label “OK”.

The entropy H intuitively means the averaged surprise when we flip this coin.

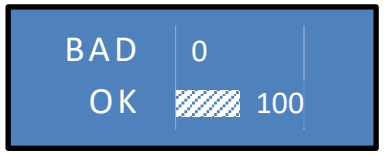
$$H = \sum \text{Probability} \cdot \text{Surprise}$$

$$= p_{\text{BAD}} \cdot \log_2\left(\frac{1}{p_{\text{BAD}}}\right) + p_{\text{OK}} \cdot \log_2\left(\frac{1}{p_{\text{OK}}}\right)$$

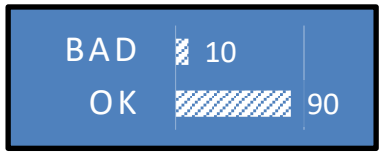


More information about entropy -- https://statquest.org/entropy-for-data-science-clearly-explained/

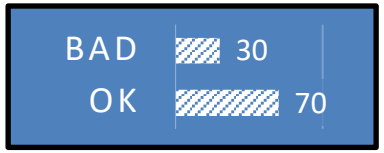
We want a small entropy. Entropy is zero when the coin always gives one side. Entropy reaches the maximum when the coin is fair, meaning two sides have equal probability.



$$H = 0 + 1 \cdot \log_2\left(\frac{1}{1}\right) = 0$$



$$H = 0.1 \cdot \log_2\left(\frac{1}{0.1}\right) + 0.9 \cdot \log_2\left(\frac{1}{0.9}\right) = 0.47$$



$$H = 0.3 \cdot \log_2\left(\frac{1}{0.3}\right) + 0.7 \cdot \log_2\left(\frac{1}{0.7}\right) = 0.88$$

$$H = 0.5 \cdot \log_2\left(\frac{1}{0.5}\right) + 0.5 \cdot \log_2\left(\frac{1}{0.5}\right) = 1$$

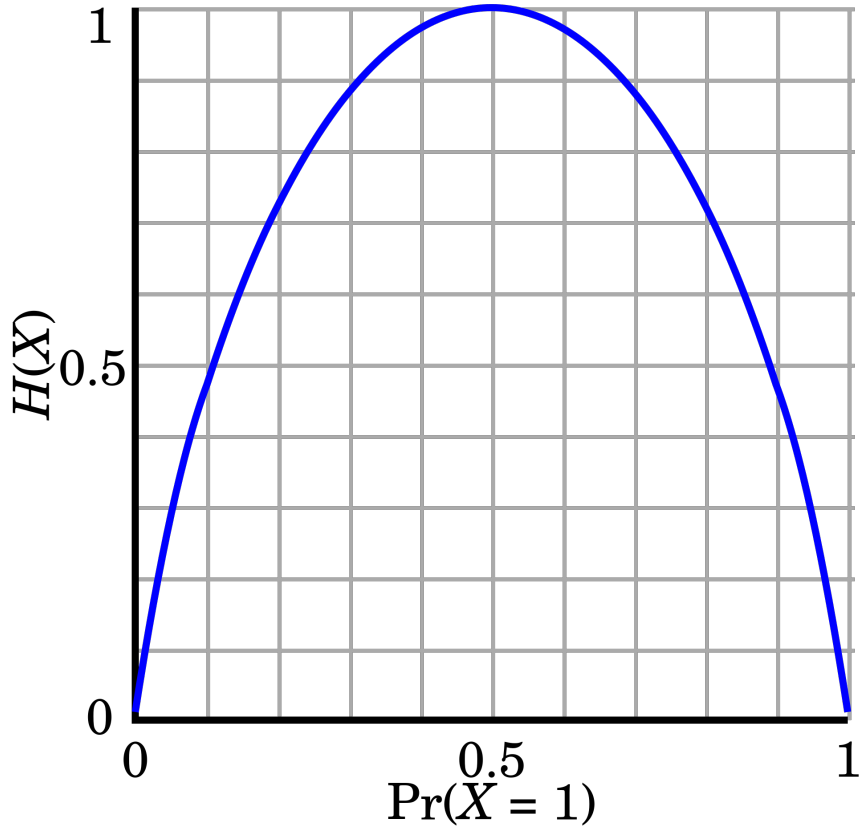
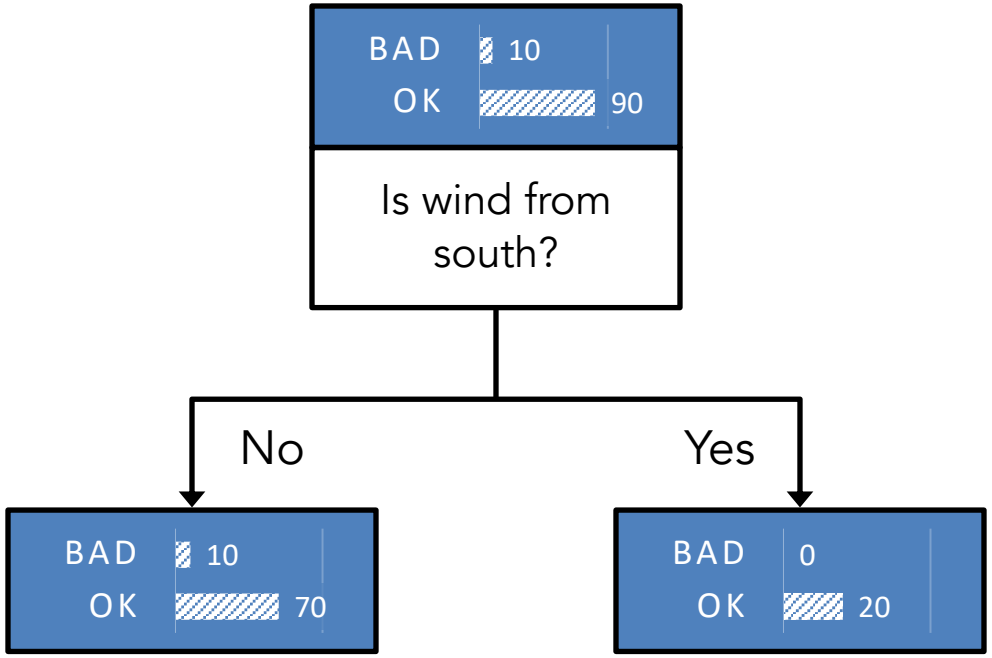


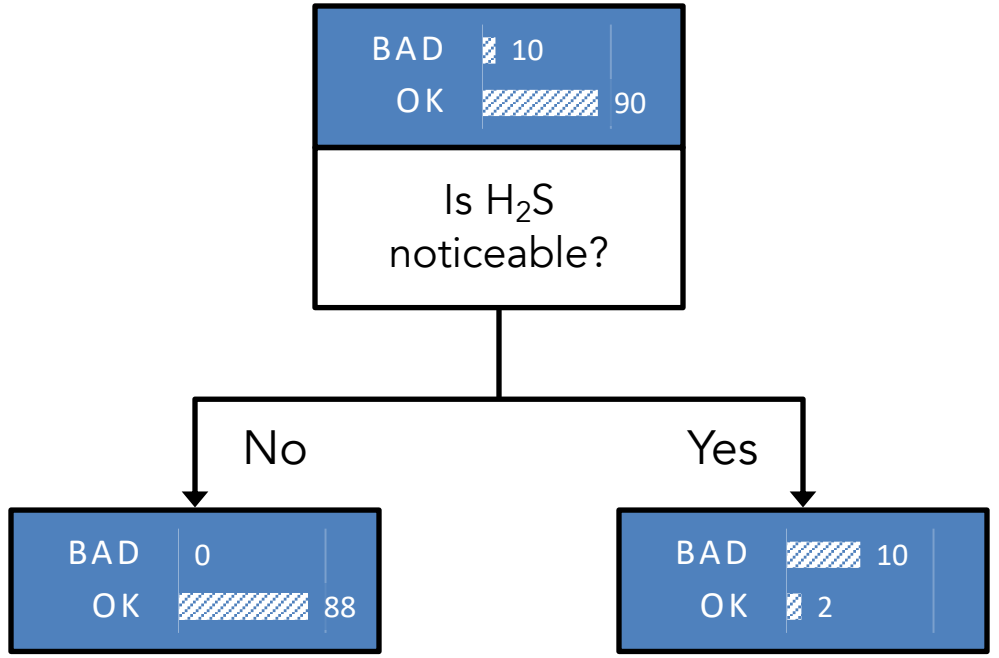
Image source -- https://en.wikipedia.org/wiki/Entropy_(information_theory)

When splitting the parent node, we can use the averaged entropy of the leaf nodes to measure and quantify the information that each feature gives.



Entropy = 0.54 Entropy = 0

Averaged entropy $= \frac{80}{100} * 0.54 + 0 = 0.43$

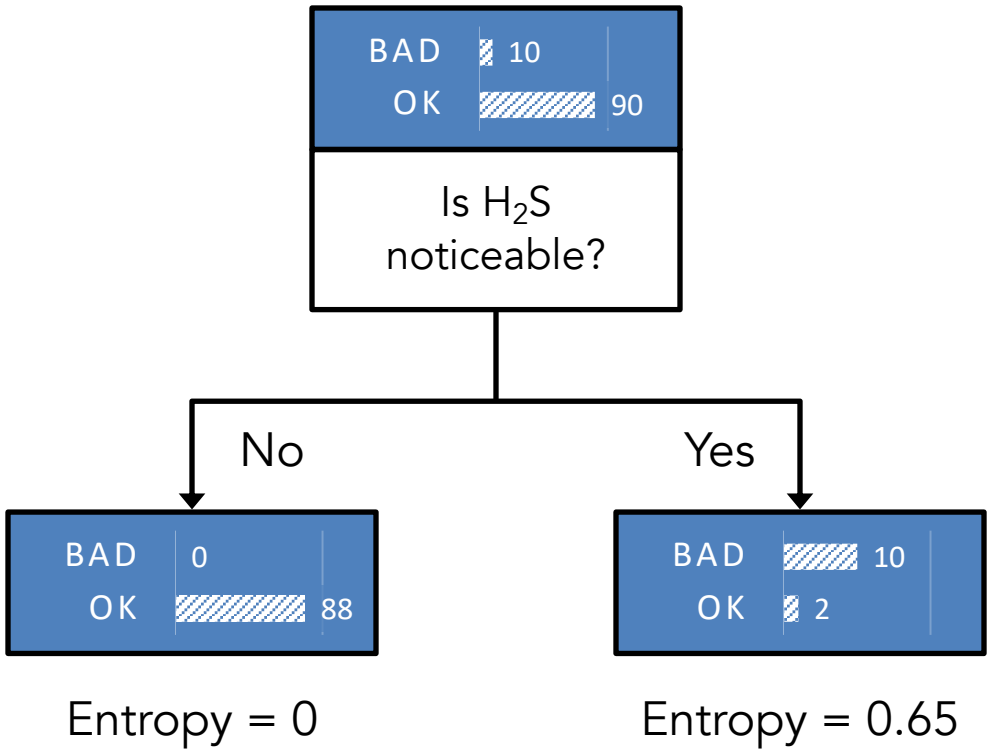


Entropy = 0 Entropy = 0.65

Averaged entropy $= 0 + \frac{12}{100} * 0.65 = 0.08$

We can also use information gain to measure the reduction in uncertainty after the split.

H_{parent} = entropy of the parent node = 0.47



Information Gain

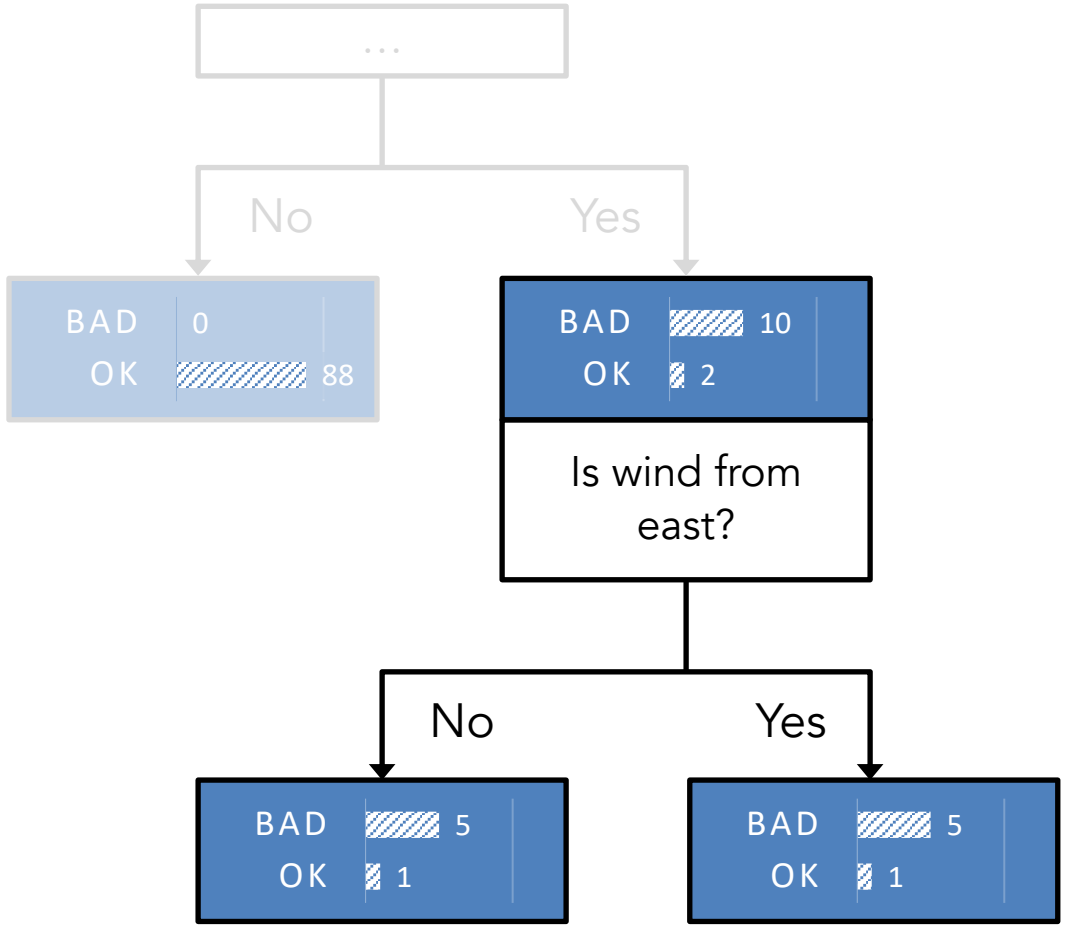
$= H_{parent} - H_{leaf}$

$= 0.47 - 0.08$

$= 0.39$

H_{leaf} = averaged entropy of leaves = 0.08

We can stop splitting when the information gain is too small for the best feature, which means splitting the node does not give a reasonable reduction of error (or uncertainty).



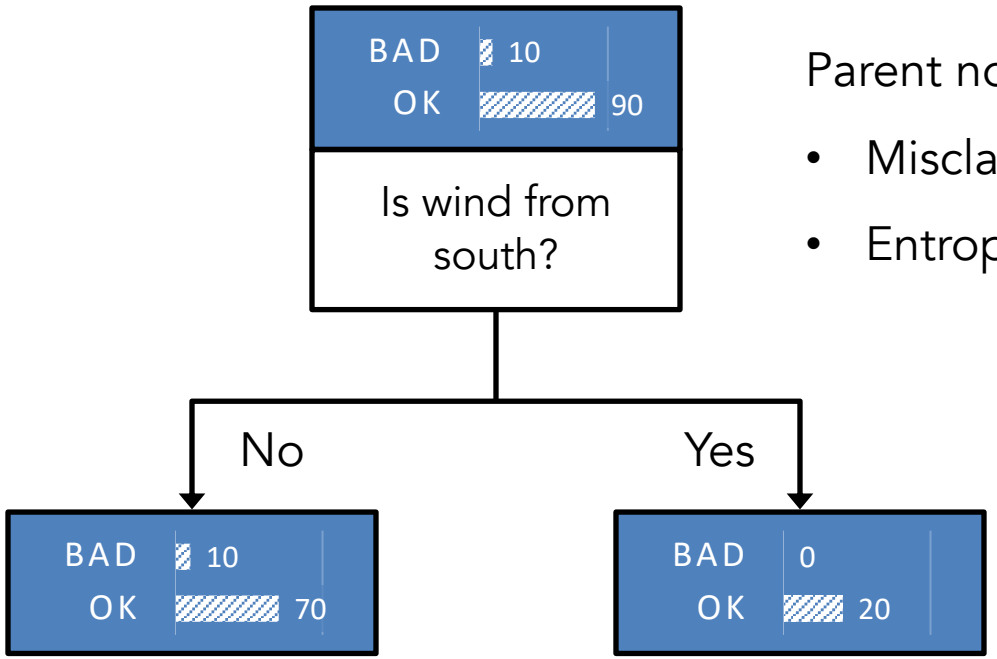
Parent node:

- Misclassification error = 0.17
- Entropy = 0.65

Leaf nodes:

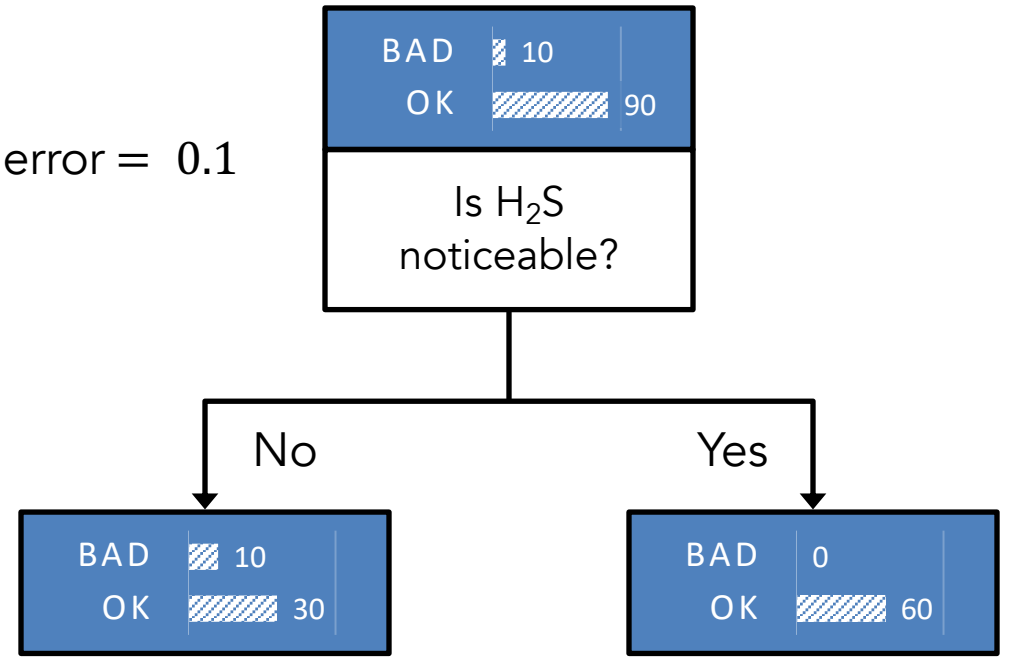
- Total misclassification error = 0.17
- Averaged entropy = 0.65

Exercise: Why using the misclassification error as the node splitting strategy is not a good idea when training a decision tree?



Parent node:

- Misclassification error = 0.1
- Entropy = 0.47



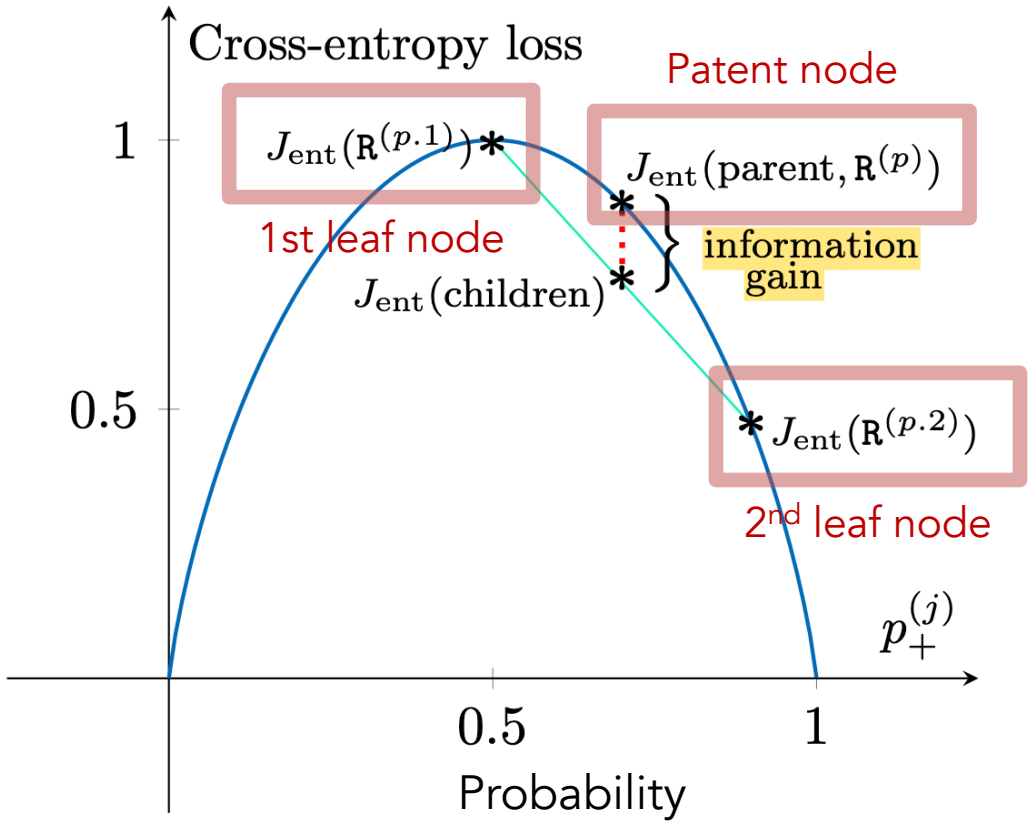
Leaf nodes:

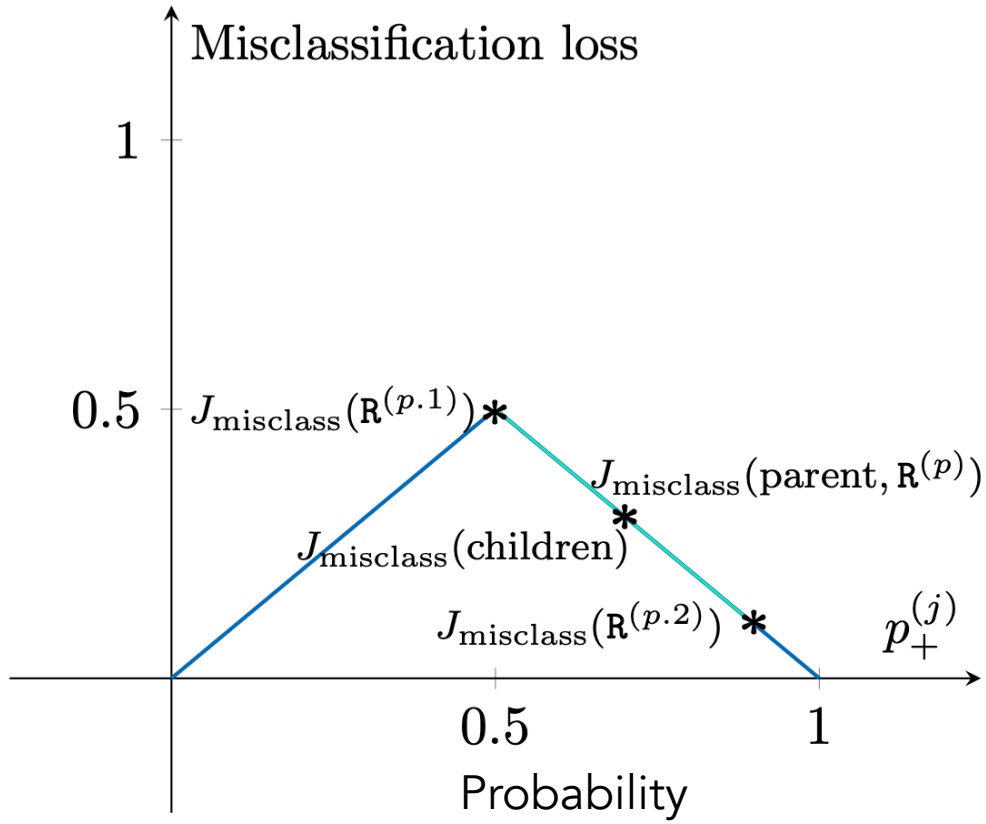
- Total misclassification error = 0.1
- Averaged entropy = 0.43

Leaf nodes:

- Total misclassification error = 0.1
- Averaged entropy = 0.32

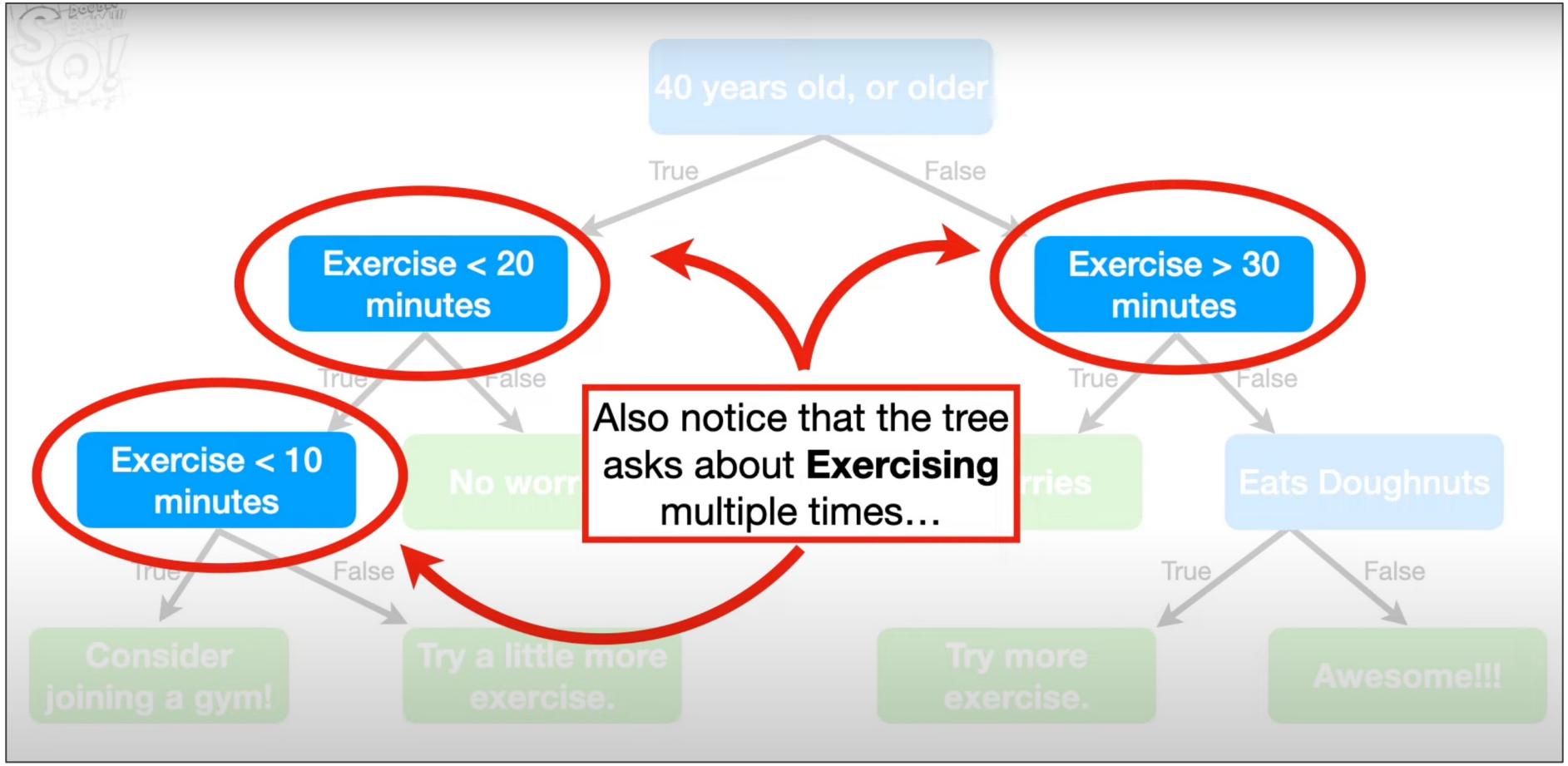
The misclassification loss (i.e., error) is not very sensitive to changes in probabilities and can lead to zero information gain, as shown in the bottom right graph.





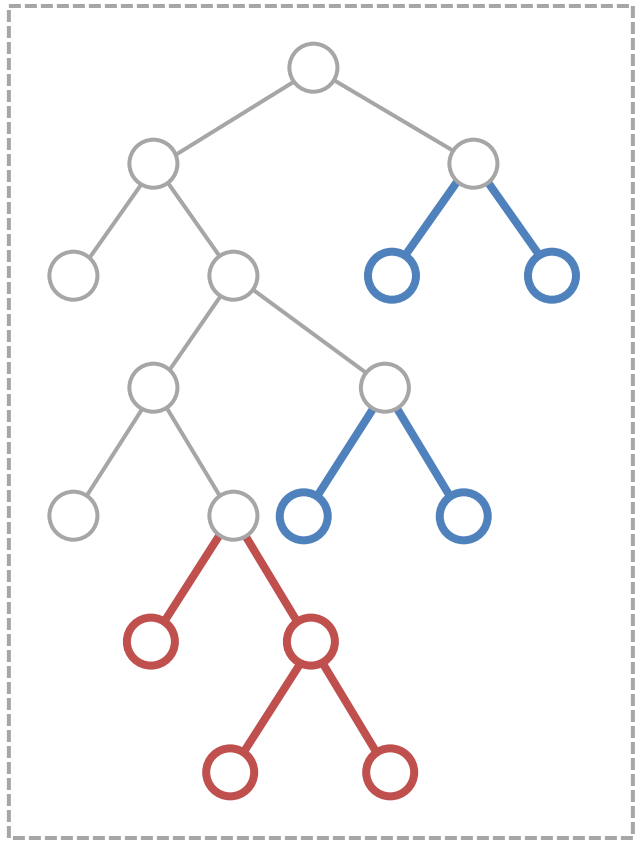
Algorithm source -- http://ciml.info/dl/v0_99/ciml-v0_99-ch01.pdf

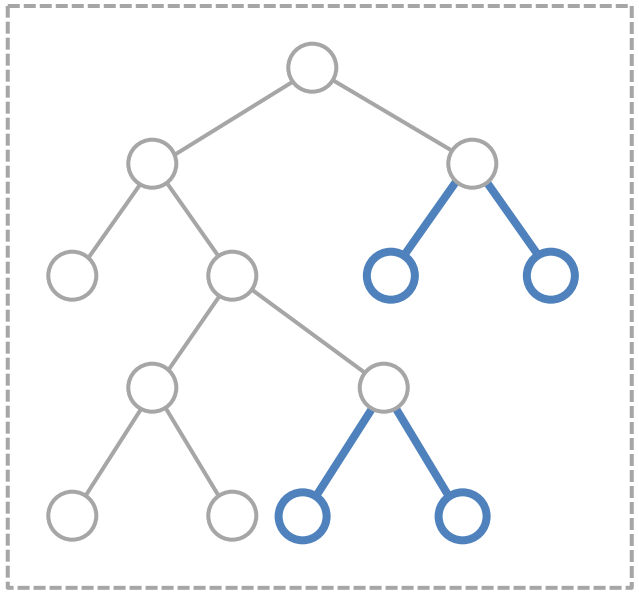
Decision Tree also works on continuous features but requires extra care. For example, one method is to bin the continuous values and treat each bin as a categorical feature.



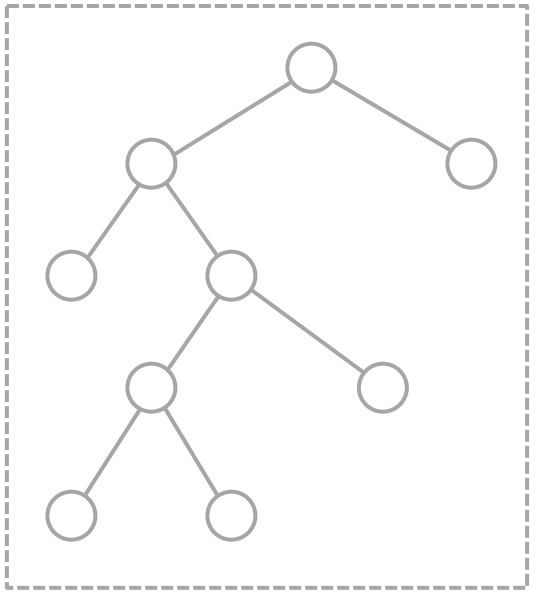
More information about Decision Tree -- https://statquest.org/decision-and-classification-trees-clearly-explained/

Decision Tree can overfit easily. To combat overfitting, we can stop splitting a node when it reaches the maximum tree depth or does not have a minimum sample size.



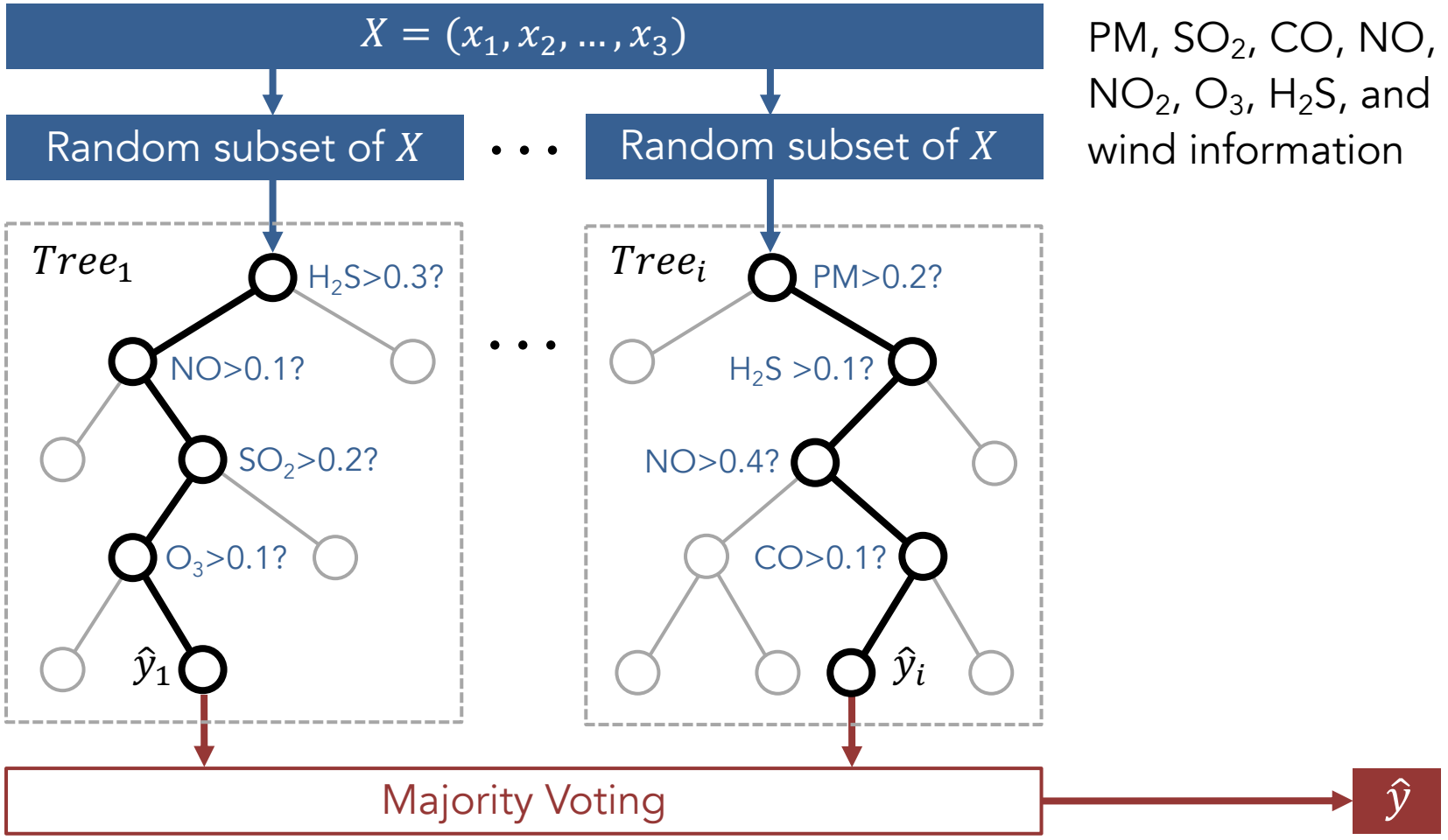


Limit max depth but allow a small sample size.



Limit max depth and restrict sample size.

We can also mitigate the overfitting problem of the Decision Tree by using the bagging technique, which is an ensemble of multiple trees, such as the Random Forest model.



PM, SO_2, CO, NO, NO_2, O_3, H_2S, and wind information

Prediction of bad smell (yes/no)

The bagging technique for the Random Forest model uses randomly selected features and bootstrapped samples (i.e., sampling with replacement).

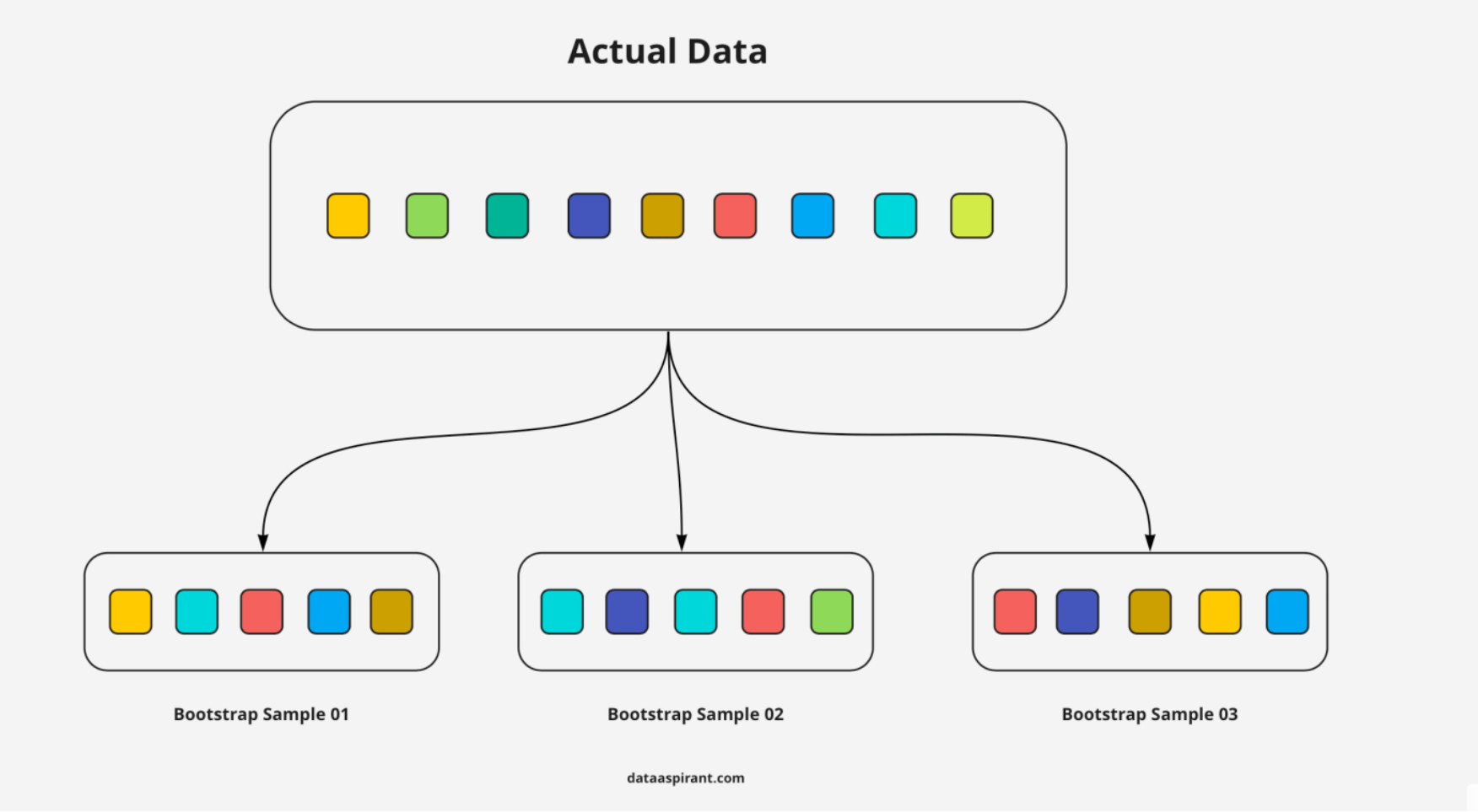
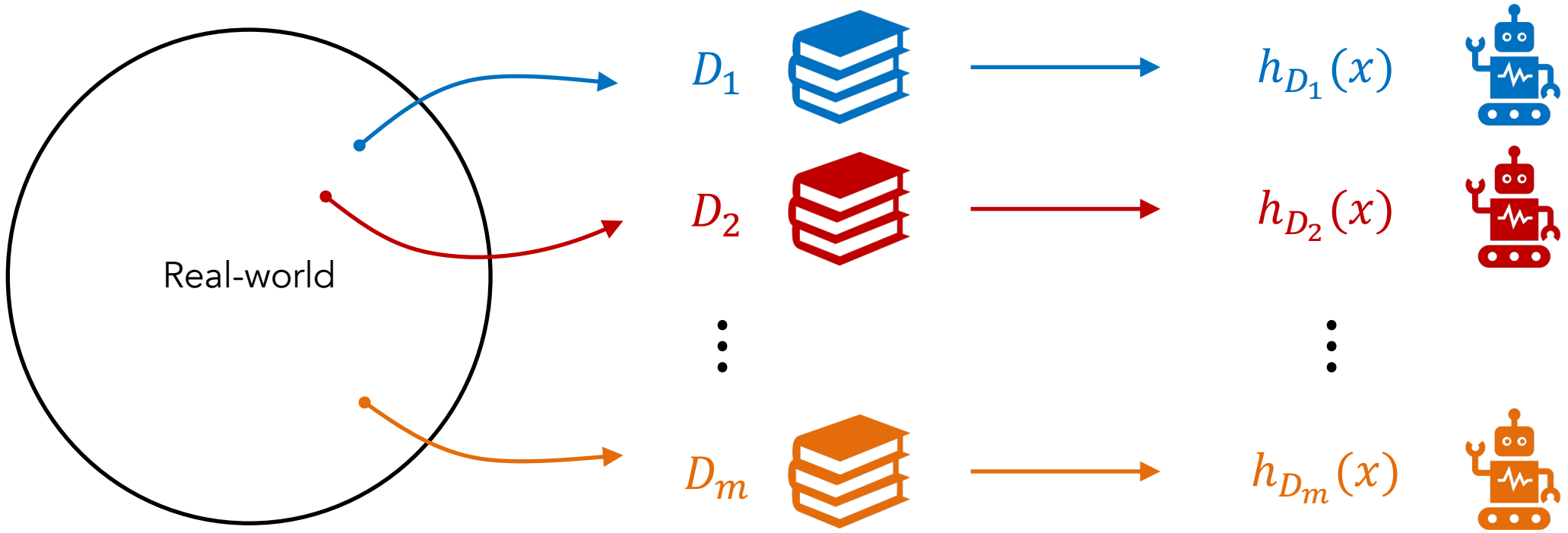


Figure source -- https://dataaspirant.com/ensemble-methods-bagging-vs-boosting-difference/

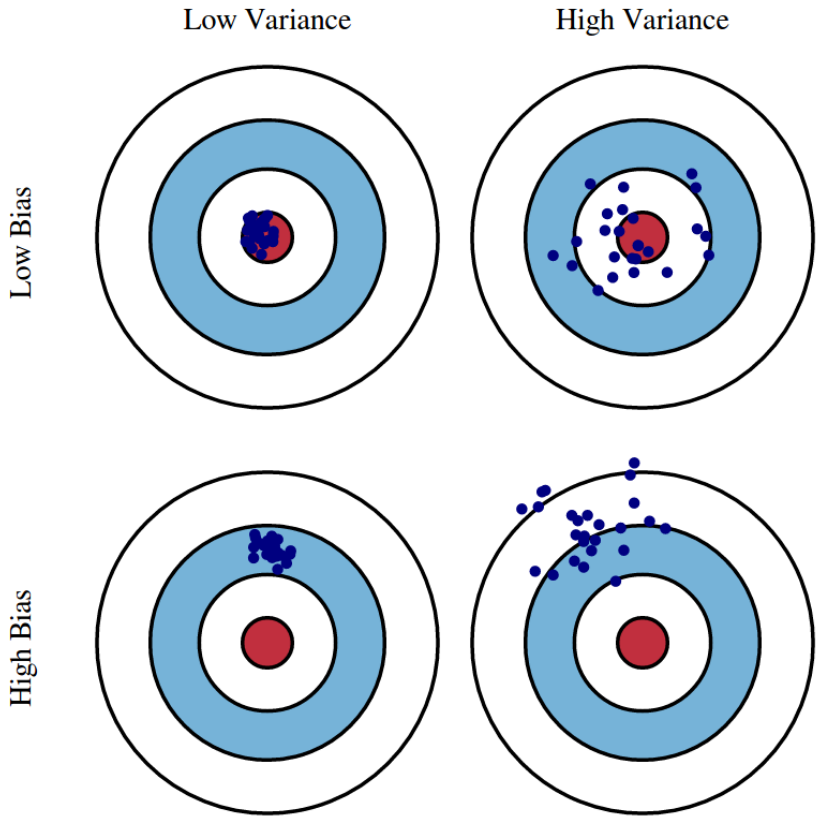
Why and how does the bagging technique work in dealing with overfitting?

Statistically speaking, the classifier that we trained is one of the all possible classifiers (i.e., drawn from a statistical distribution). We can sample many datasets D with pairs of features x and labels y. For all D, we can train a set of models $\{h_{D_1}(x), \ldots, h_{D_m}(x)\}$.



Errors of the model that we trained can be decomposed into bias, variance, and noise.

$$\underbrace{\mathbb{E}[(h_D(x) - y)^2]}_{\text{Error}} = \underbrace{\mathbb{E}[(h_D(x) - \bar{h}(x))^2]}_{\text{Variance}} + \underbrace{\mathbb{E}[(\bar{h}(x) - \bar{y}(x))^2]}_{\text{Bias}} + \underbrace{\mathbb{E}[(\bar{y}(x) - y(x))^2]}_{\text{Noise}}$$



$\mathbb{E}$: the expected value (i.e., average value)

D: the training dataset $\left(x^{(i)}, y^{(i)}\right)$ pairs, sampled from P

$h_D(x)$: the classification/regression model, trained on D

$\bar{h}(x)$: the average model (from all possible models)

y: the true labels that are used for testing a model

$\bar{y}\ (x)$: the average label value given feature x

Figure source -- http://scott.fortmann-roe.com/docs/BiasVariance.html

Overfitting usually comes from training a very complex model that has a high variance.

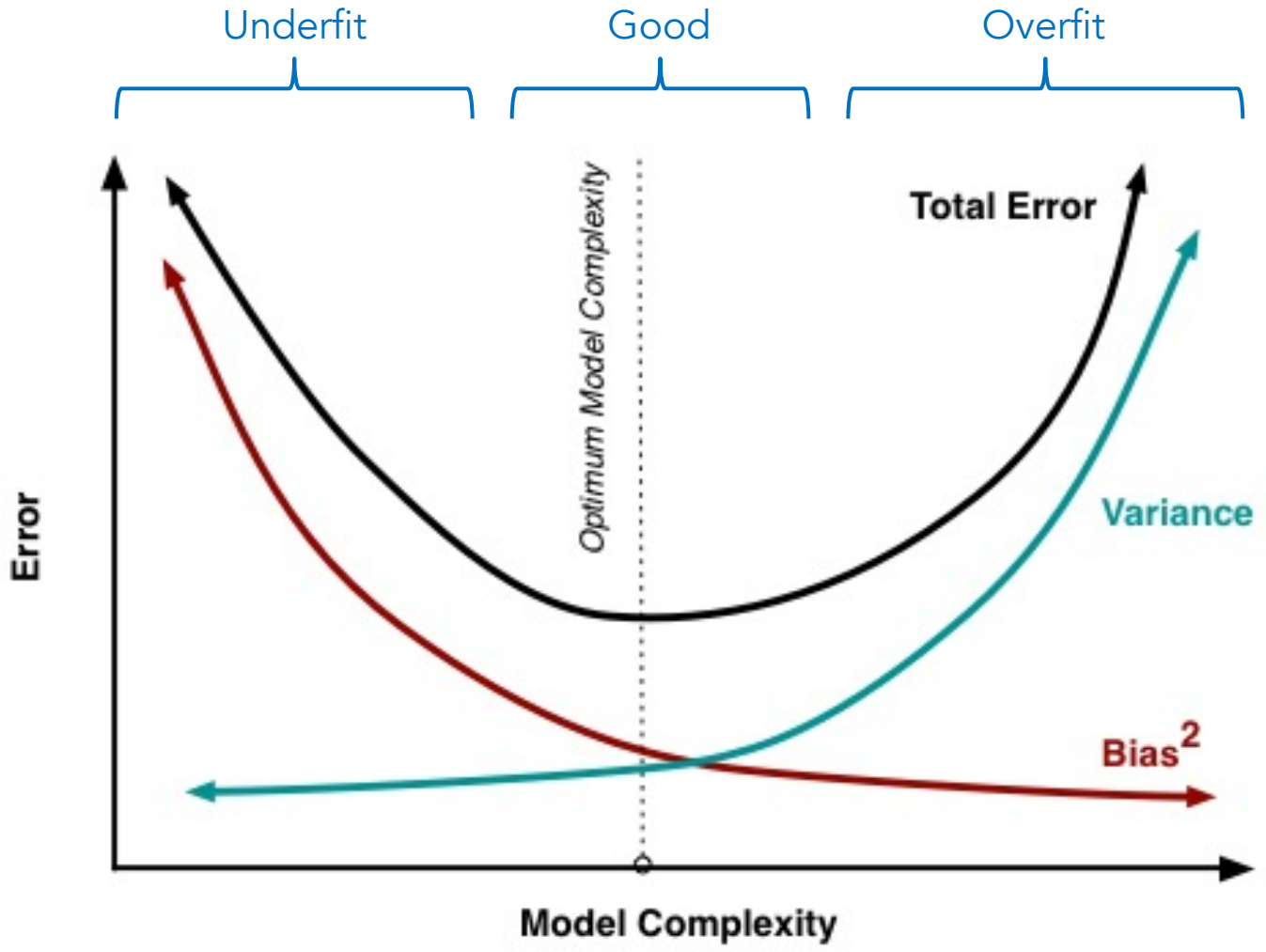


Figure source -- http://scott.fortmann-roe.com/docs/BiasVariance.html

We can use the weak law of large numbers to reduce the variance of a complex model.

$$\underbrace{\mathbb{E}[(h_D(x) - y)^2]}_{\text{Error}} = \underbrace{\mathbb{E}[(h_D(x) - \bar{h}(x))^2]}_{\text{Variance}} + \underbrace{\mathbb{E}[(\bar{h}(x) - \bar{y}(x))^2]}_{\text{Bias}} + \underbrace{\mathbb{E}[(\bar{y}(x) - y(x))^2]}_{\text{Noise}}$$

Our goal is to reduce the variance term: $\mathbb{E}[(h_D(x) - \bar{h}(x))^2]$.
For this, we want $h_D \to \bar{h}$.

The weak law of large numbers says (roughly) for i.i.d. random variables x_i with mean $\bar{x}$, we have,

$$\frac{1}{m}\sum_{i=1}^{m} x_i \to \bar{x} \text{ as } m \to \infty$$

Apply this to classifiers: Assume we have m training sets $D_1, D_2, \ldots, D_n$ drawn from P^n. Train a classifier on each one and average result:

$$\hat{h} = \frac{1}{m}\sum_{i=1}^{m} h_{D_i} \to \bar{h} \qquad as\ m \to \infty$$

We refer to such an average of multiple classifiers as an **ensemble** of classifiers.

More about bagging -- https://www.cs.cornell.edu/courses/cs4780/2018fa/lectures/lecturenote18.html

Bagging is one of the ensemble learning methods, where multiple weak classifiers are combined into a stronger classifier using various techniques.

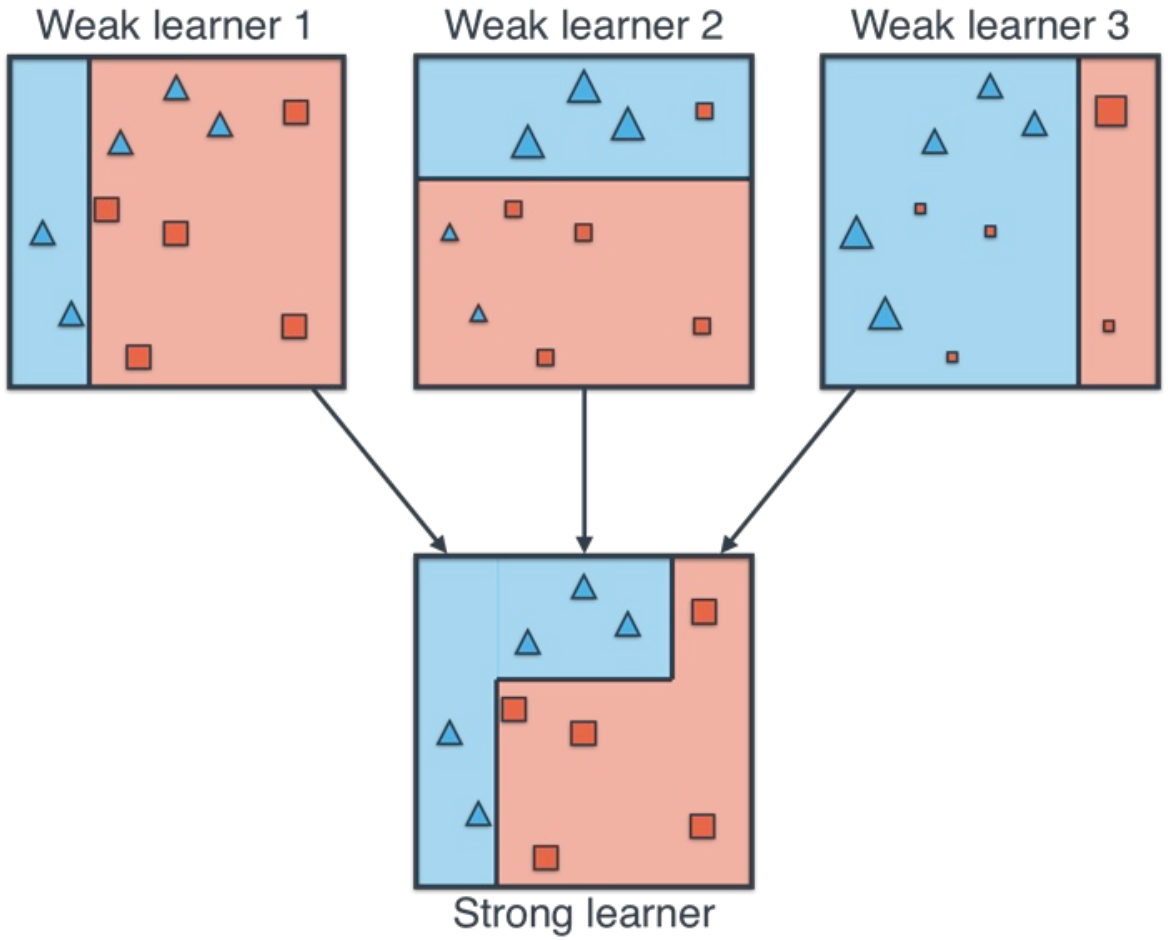
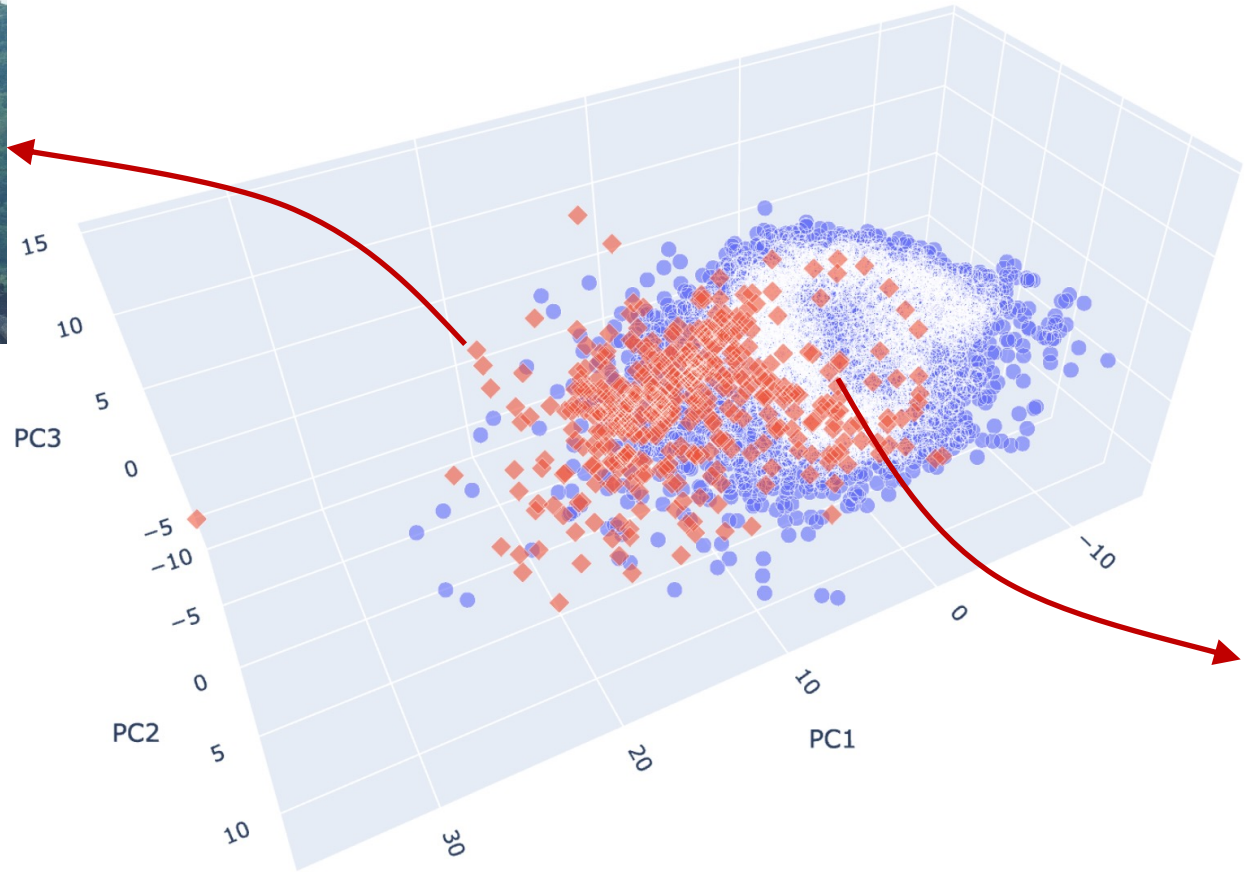


Figure source -- https://mathchi.medium.com/weak-learners-strong-learners-for-machine-learning-e73e32f86ebd

Not everything in the data is learnable, and we typically consider these as noise. For example, smell reports could come from sources that may not have obvious patterns.

Smell from pollution may have patterns.



BBQ outside may have no obvious patterns.

Source for the right photo -- https://unsplash.com/photos/YZQd7lCWAsQ

Take-Away Messages

- Decision Tree has a non-linear decision boundary.
- Decision Tree can overfit easily. You can limit the tree depth or sample size to mitigate the problem.
- Entropy and misclassification error can help us find the best feature to split a tree node.
- Entropy is the averaged surprise from flipping a coin with two classes (in the binary setting).
- We want to see the entropy as small as possible after splitting a tree node.
- Random Forest can be seen as a committee of Decision Tree models.
- The bagging technique can reduce the variance term in the error.
- We can combine weak models together into a stronger model.

Questions?