Imperial College London

Introduction to Machine Learning
CO395

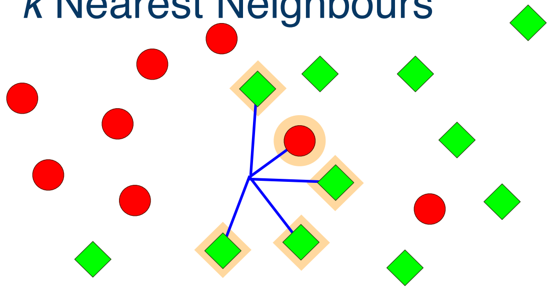
Lecture 2

Antoine Cully & Marek Rei & Josiah Wang

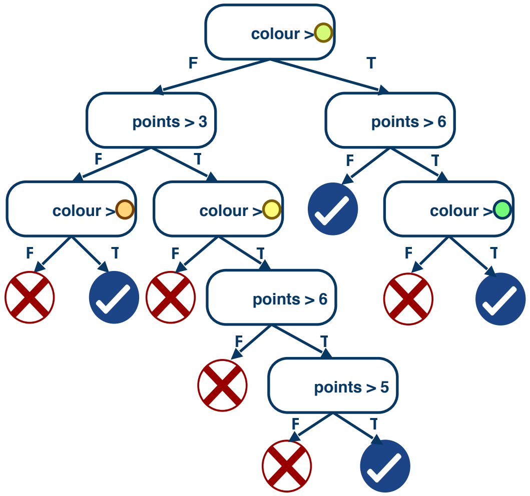
Course plan

	Lecture	Lecturer
Week 2	Introduction to ML	*Josiah*
Week 3	Instance-based Learning + Decision Trees	*Antoine*
Week 4	Machine Learning Evaluation	*Marek*
Week 5	Artificial Neural Networks I	*Marek*
Week 6	Artificial Neural Networks II	*Marek*
Week 7	Unsupervised Learning	*Antoine*
Week 8	Genetic Algorithms	*Antoine*

k Nearest Neighbours

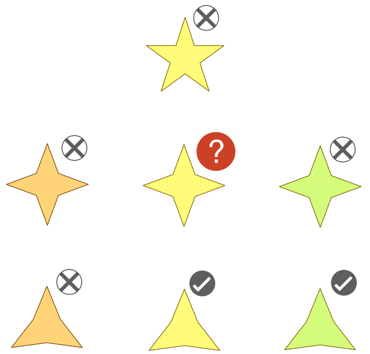


Decision Trees



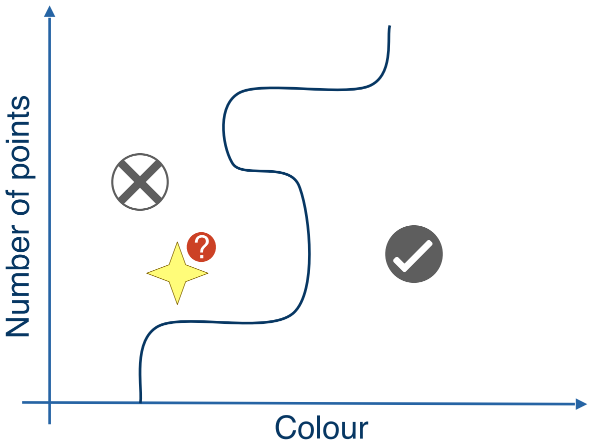
Classification: Lazy vs. Eager Learning

Lazy Learner
Stores the training examples and postpones generalising beyond these data until an explicit request is made at test time.



k Nearest Neighbours

Eager Learner
Constructs a general, explicit description of the target function based on the provided training examples.



Decision Trees

Today's lecture

- Classification with Instance-based Learning
    - *k* Nearest Neighbours (*k*-NN) classifier
    - Distance weighted *k*-NN
    - *k*-NN regression (quick intro)
- Classification with Decision Trees
    - Intuitions & Motivations
    - Information Entropy/Information Gain
    - Algorithm
    - Worked Example
    - Random forests & regression trees (quick intro)

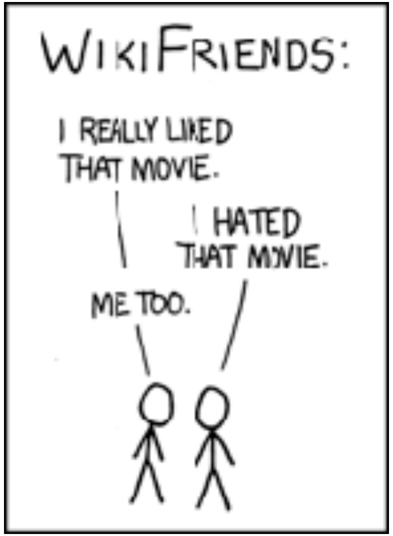
Today's lecture

- 6 videos:
  - This short intro
  - Classification with Instance-based Learning
  - Classification with Decision Trees
  - How to select the 'optimal' split rule?
  - Worked example for constructing decision tree
  - Summary and other considerations with decision tree

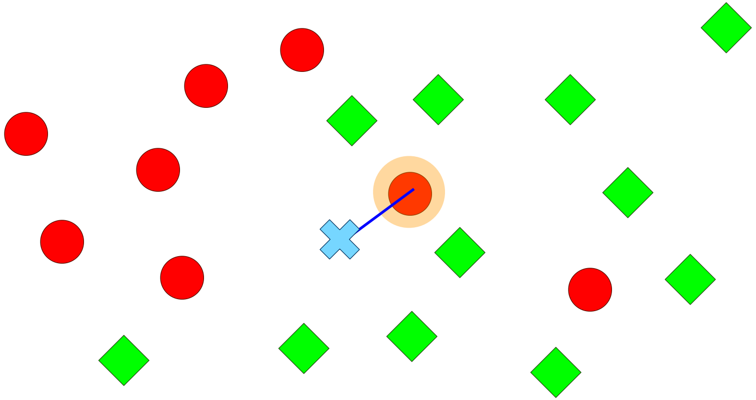
To be continued…
(Classification with Instance-based Learning)

Classification with Instance-based Learning (k-Nearest Neighbours)

Instance-based Learning



It's crazy how much my gut opinion of a movie/song is swayed by what other people say, regardless of how I felt coming out of the theater.



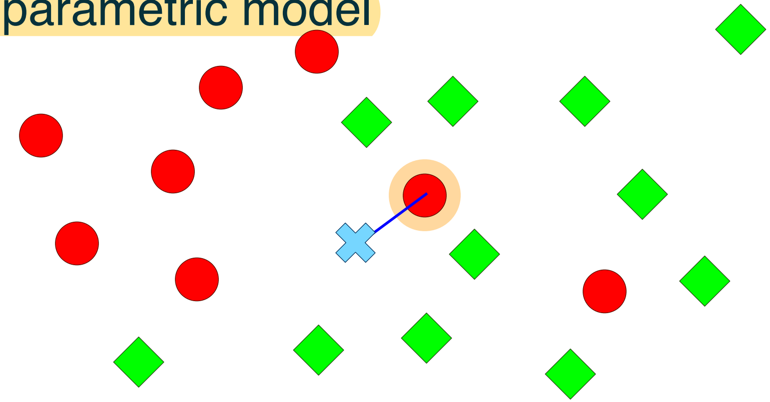
Instance-based Learning: instead of performing explicit generalization, compares new problem instances with instances seen in training, which have been stored in memory.

https://xkcd.com/185/
https://www.explainxkcd.com/wiki/index.php/185:_Wikifriends

Instance-based Learning

- Nearest Neighbour classifier
  - Classify a test instance to the class label of the nearest training instance (according to some distance metric)
    - Non-parametric model

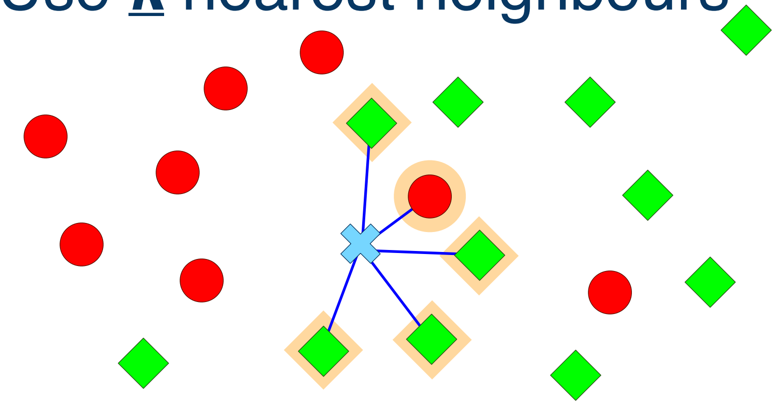
Non-parametric models assume that the data distribution cannot be defined in terms of such a finite set of parameters.



- Question: What is the problem with using just one nearest neighbour?

k-Nearest Neighbours (k-NN) classifier

- One nearest neighbour
  - Sensitive to noise
  - Overfit training data
- Solution: Use ***k*** nearest neighbours



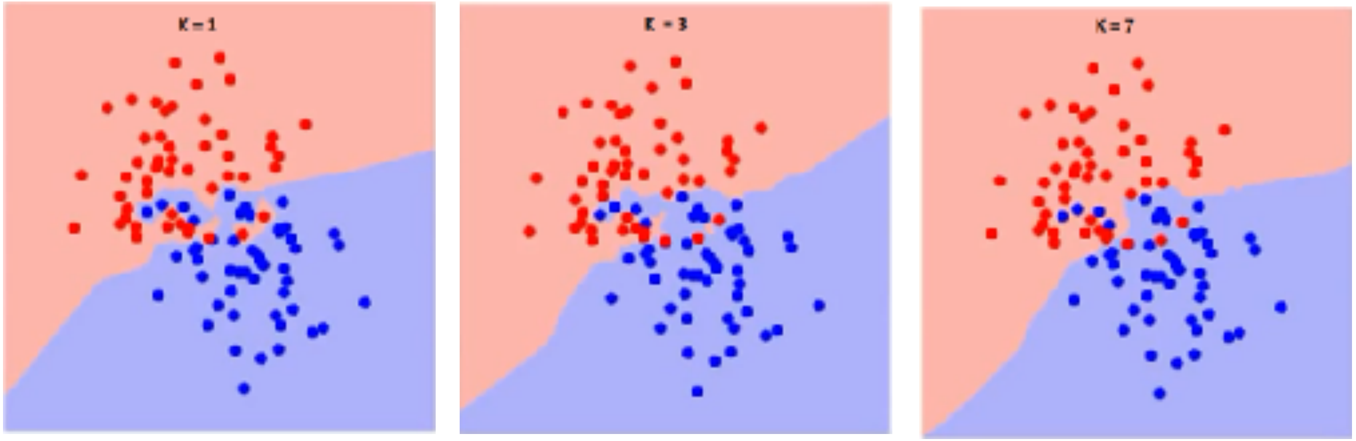
- *k* is usually an odd number for binary classification (why?)

Ensure the final vote will always have a decision

k-Nearest Neighbours (k-NN) classifier

- Increasing *k* will make the classifier:
  - have a smoother decision boundary (higher bias)
  - less sensitive to training data (lower variance)
- How to choose *k*? Use a validation dataset (Lecture 3)

Smaller k: Overfitting
Larger k: Underfitting

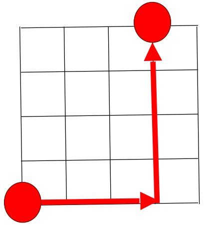


Figures from: https://www.analyticsvidhya.com/blog/2018/03/introduction-k-neighbours-algorithm-clustering/

k-NN classifier: Distance Metrics

Manhattan distance (L^1-norm)

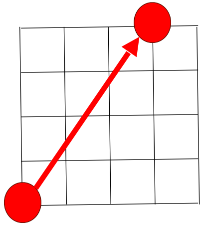
$$d(x^{(i)}, x^{(q)}) = \sum_{k=1}^{K} |x_k^{(i)} - x_k^{(q)}|$$



Only for continuous variable

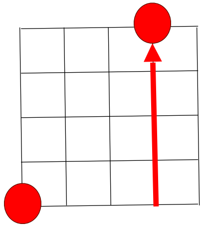
Euclidean distance (L^2-norm)

$$d(x^{(i)}, x^{(q)}) = \sqrt{\sum_{k=1}^{K} (x_k^{(i)} - x_k^{(q)})^2}$$



Chebyshev distance (L^∞-norm)

$$d(x^{(i)}, x^{(q)}) = \max_{k=1}^{K} |x_k^{(i)} - x_k^{(q)}|$$

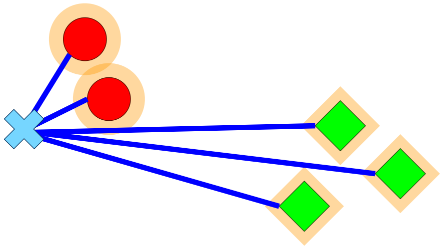


Absolute value from all features
For continuous variable

Others: Mahalanobis distance, Hamming distance, etc.

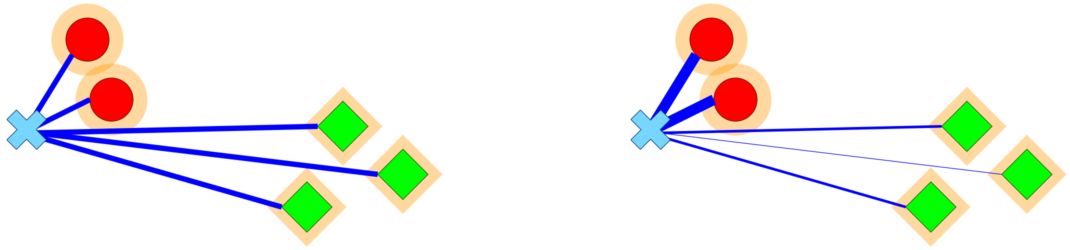
Distance weighted *k*-NN

- Should we really trust neighbours who are further away more than those close by?

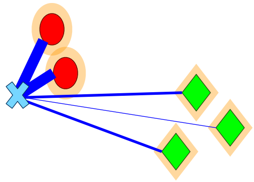


Distance weighted *k*-NN

- Should we really trust neighbours who are further away more than those close by?
- Refine *k*-NN by assigning a weight $w^{(i)}$ to each neighbour based on how close they are to the test query instance (closer -> higher weight)
- Sum the weights per class in neighbourhood, assign to class with largest sum



Distance weighted *k*-NN



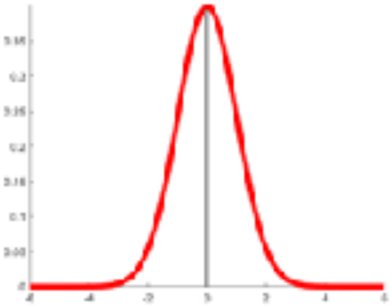
- Any measure favouring the votes of nearby neighbours will work

Inverse of distance

$$w^{(i)} = \frac{1}{d(x^{(i)}, x^{(q)})}$$

Gaussian distribution

$$w^{(i)} = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{d(x^{(i)}, x^{(q)})^2}{2}\right)$$



Distance weighted *k*-NN: Remarks

- The value of *k* is of minor importance in distance weighted *k-NN*. Distant examples will have small weights and won't greatly affect classification
- If *k=N* (size of training set): global method. Otherwise, it's a local method
- Robust to noisy training data: Classification is based on a weighted combination of all *k* nearest neighbours, effectively smoothing out the impact of isolated noise

The importance of K is reduced

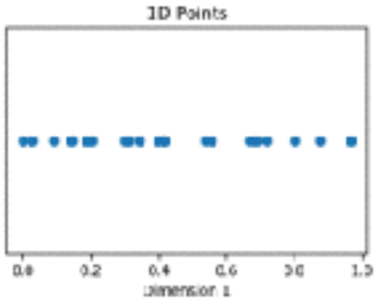
Ignore noises (i.e., that are very much further away)

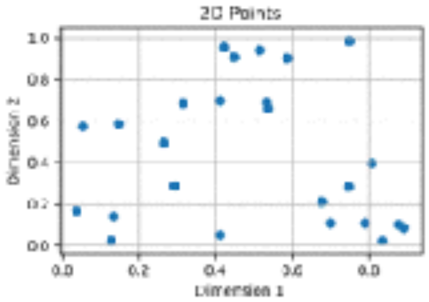
k-NN classifier: Discussion

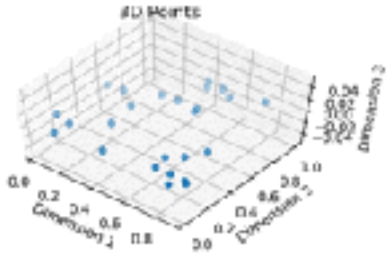
- *k*-NN is simple yet powerful
- … but might be slow for large datasets
- Speed up search: *k*-d trees, locality-sensitive hash, etc.

k-NN classifier: Discussion

- The curse of dimensionality (revisited)
  - *k*-NN relies on distance metrics, which may not work well if using all features in high dimensional spaces
  - If many features are irrelevant, instances belonging to the same class may be far from each other
  - Solution: Weight each feature differently, or perform feature selection/extraction

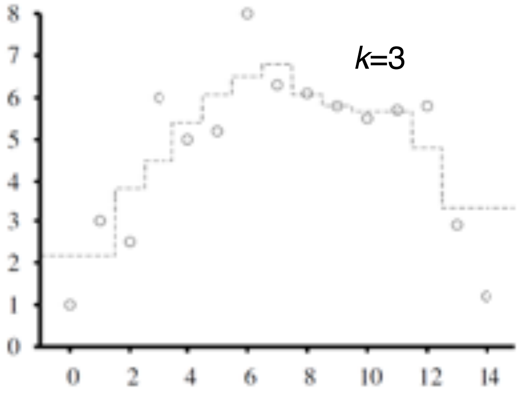




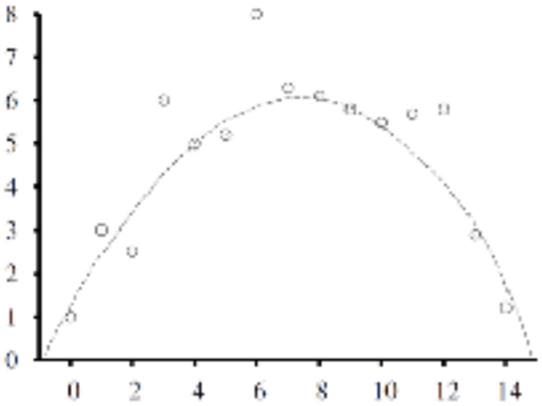


k-nearest neighbours for regression

- *k*-NN regression
  - Compute the **mean value** across *k* nearest neighbours



- Locally weighted regression
  - Distance-weighted *k*-NN for regression
  - Compute the weighted mean value across *k* nearest neighbours

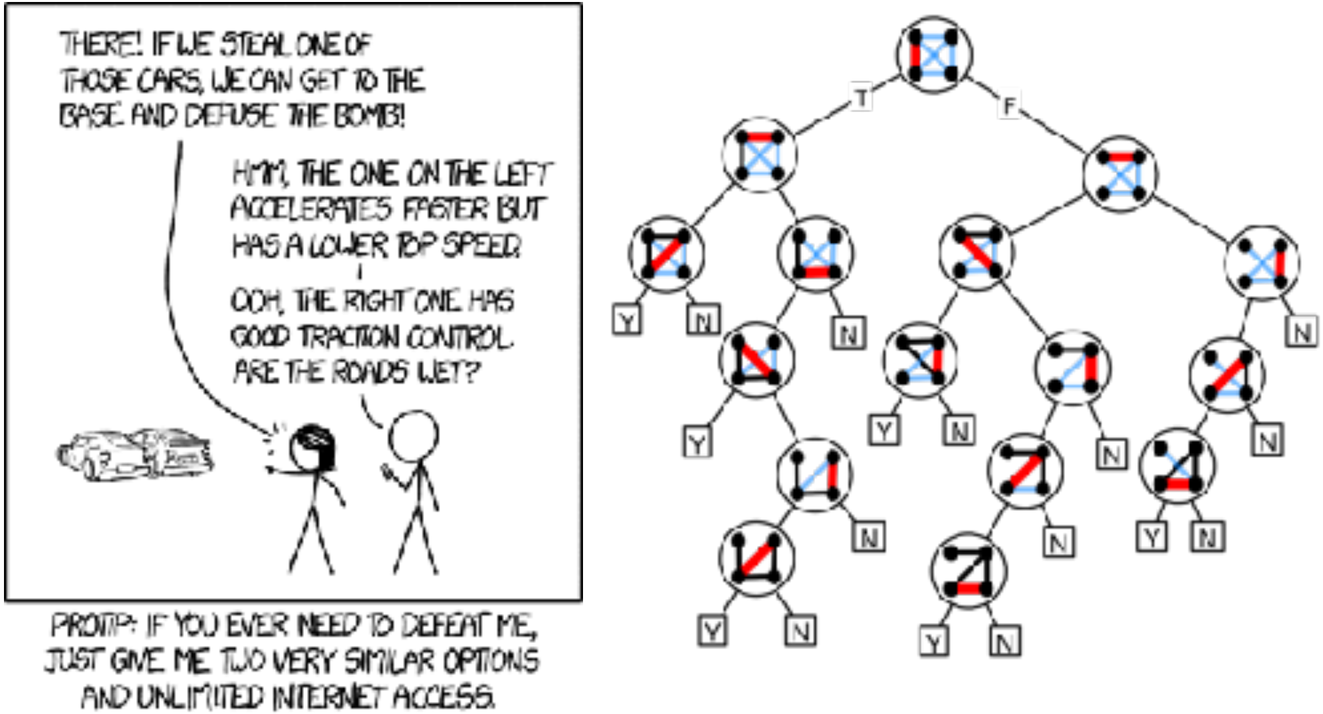


Points that are far away will have less impact to the mean value. So it is smoothed out in this case.

To be continued…
(Classification with Decision Trees)

Classification with Decision Trees

Decision Trees

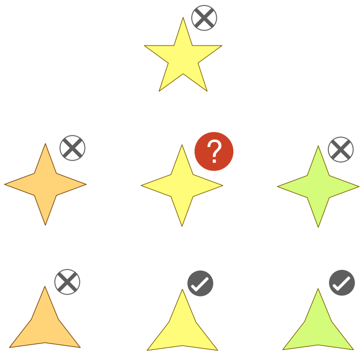


https://xkcd.com/1801/
https://www.explainxkcd.com/wiki/index.php/1801:_Decision_Paralysis

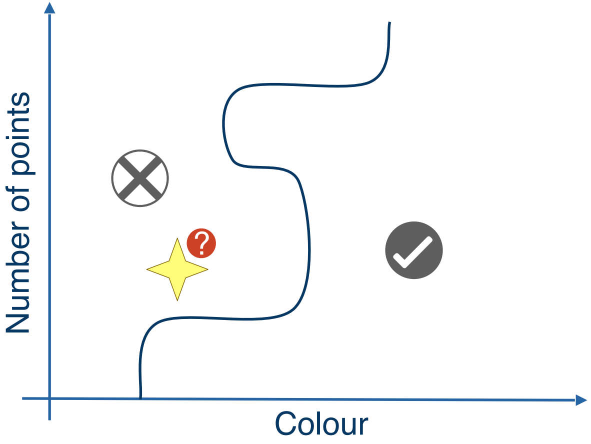
Classification: Lazy vs. Eager Learning

Lazy Learner
Stores the training examples and postpones generalising beyond these data until an explicit request is made at test time.

Eager Learner
Constructs a general, explicit description of the target function based on the provided training examples.

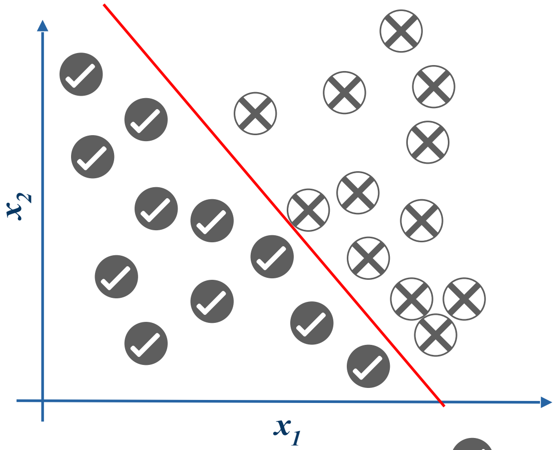


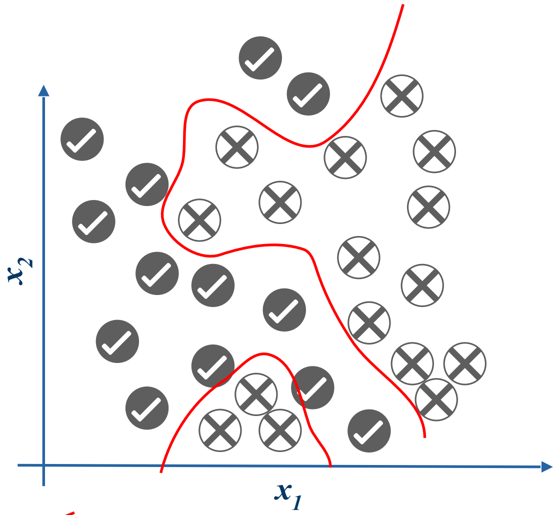
k Nearest Neighbours

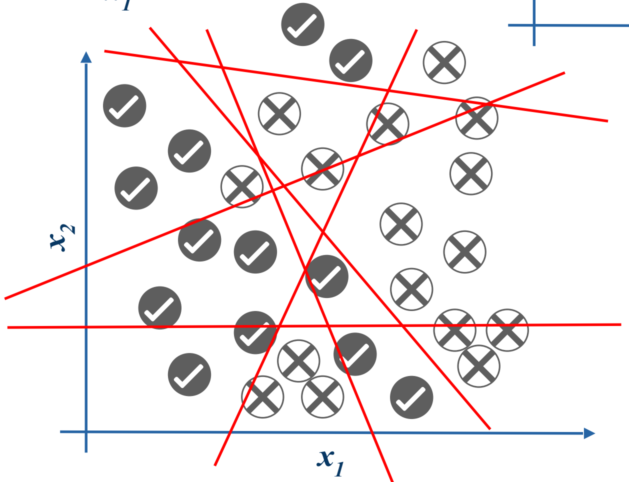


Decision Trees

Classification: Linear vs. Non-linear

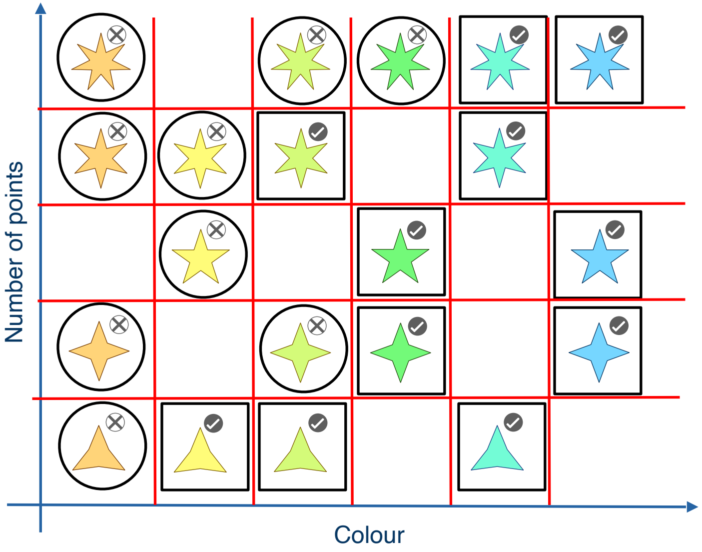




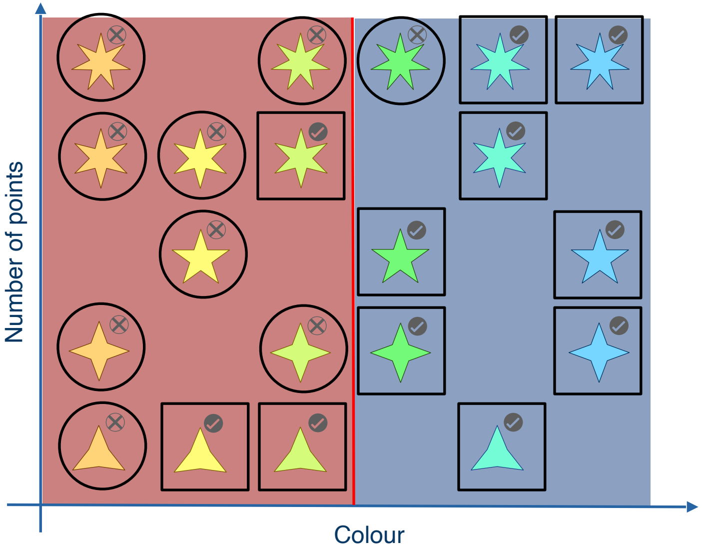


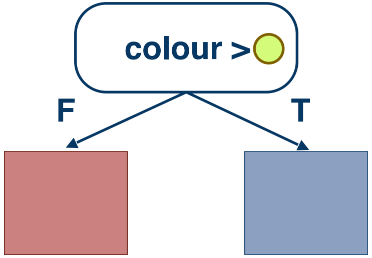
Mixture of multiple linear constraints

How to best divide the dataset?

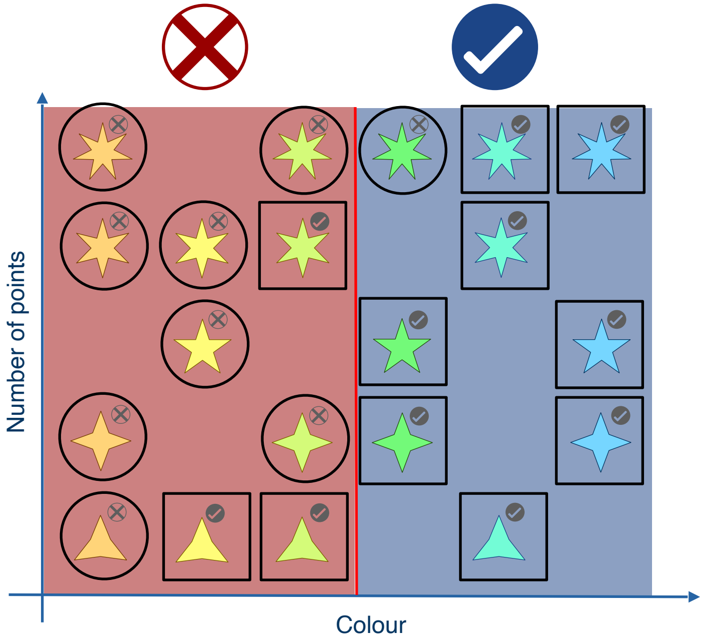


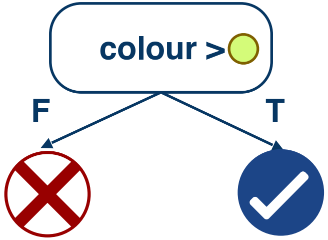
Choose this division.



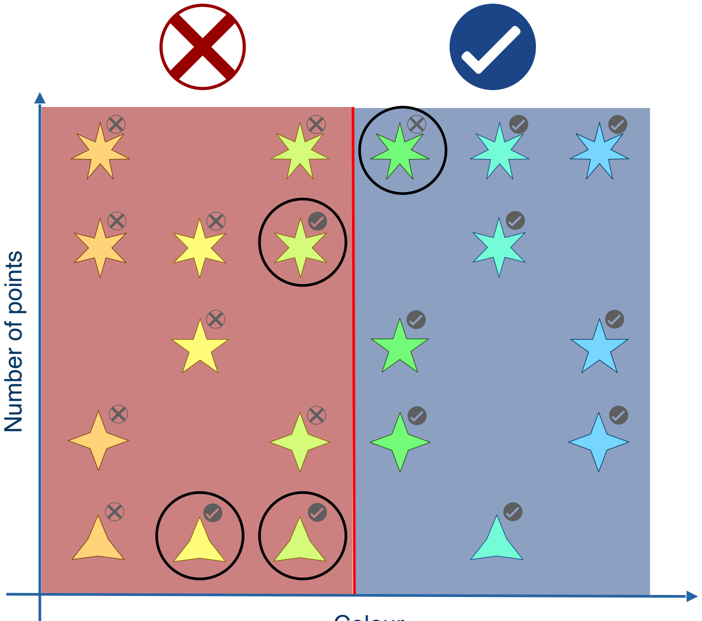


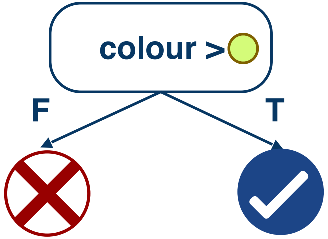
A linear classifier.



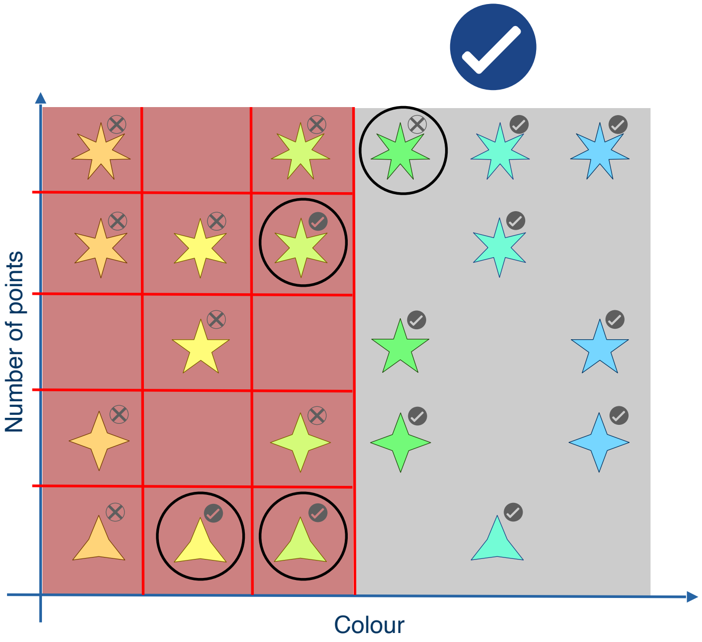


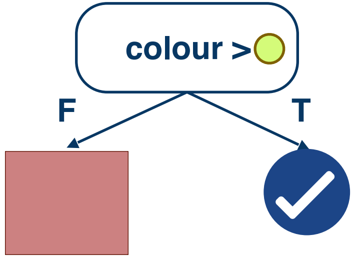
Still some mistakes.



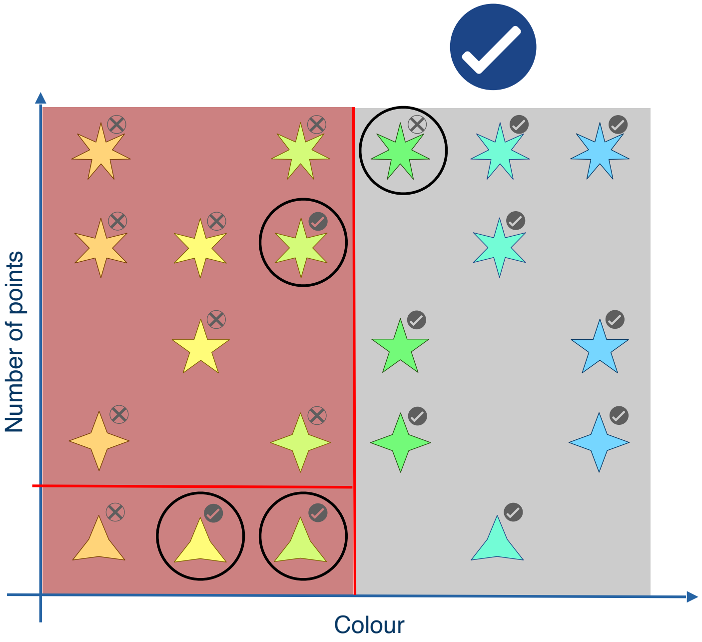


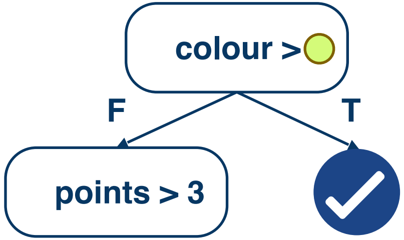
Maybe divide again on left partition?



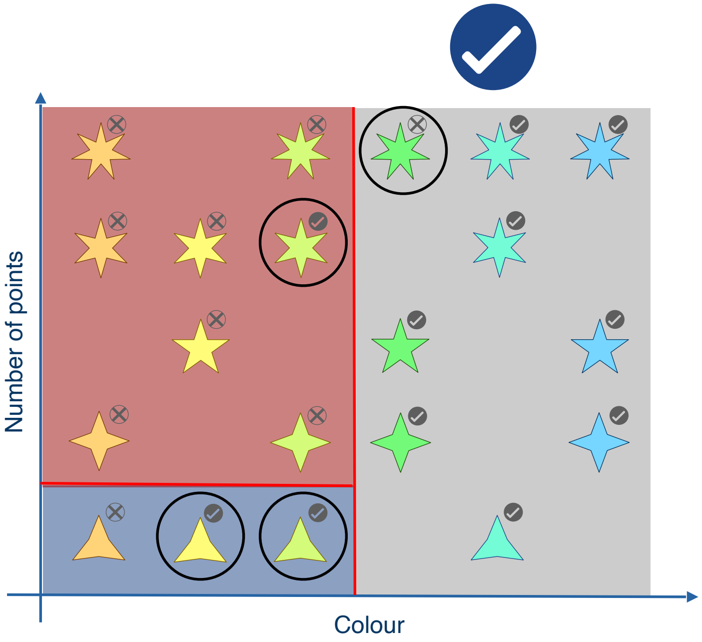


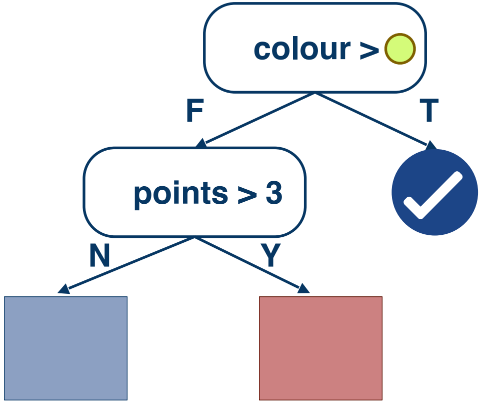
Maybe divide again on left partition?



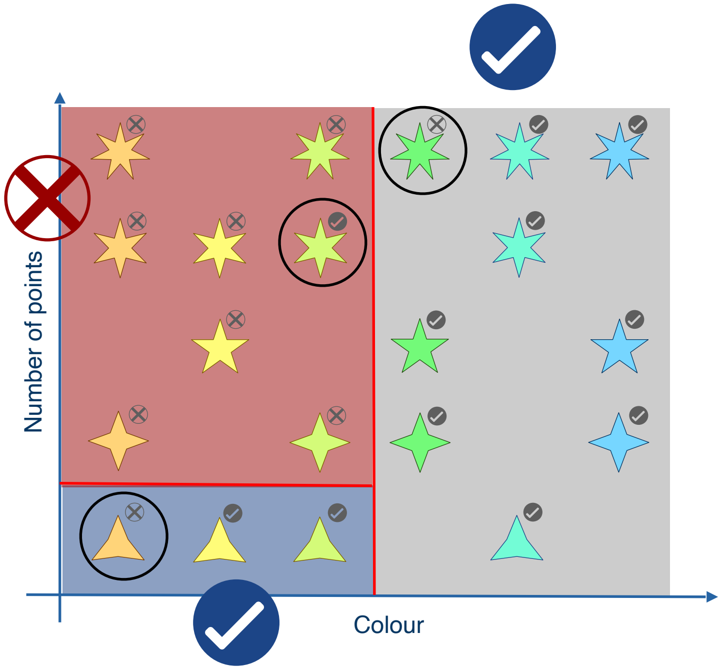


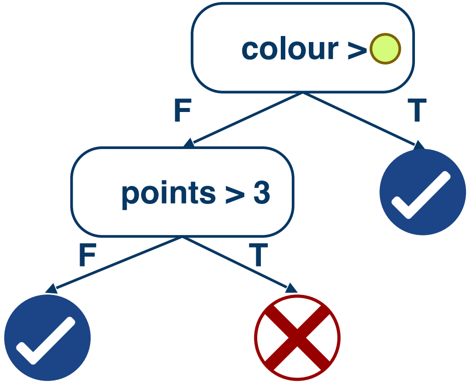
Partition the left partition.



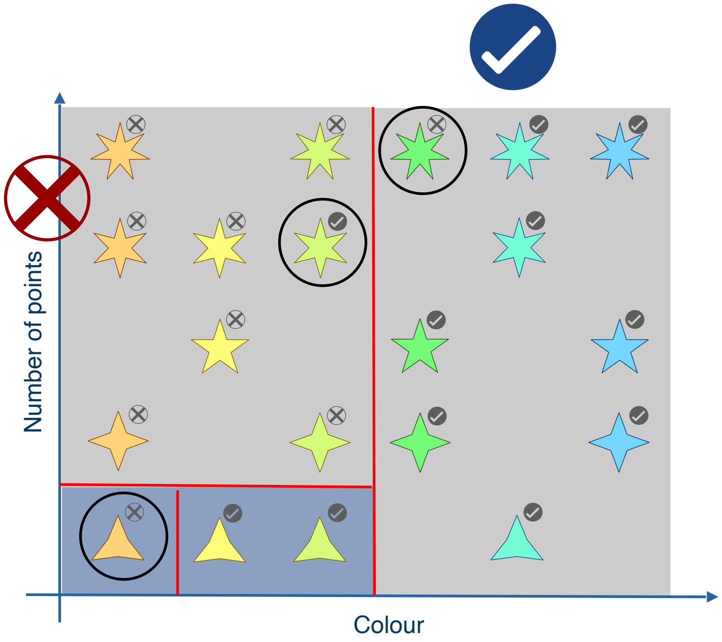


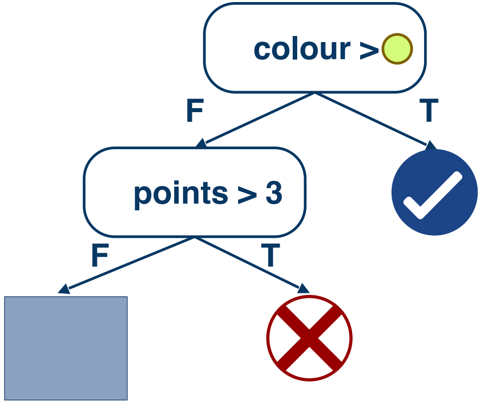
A more fine-grained tree.



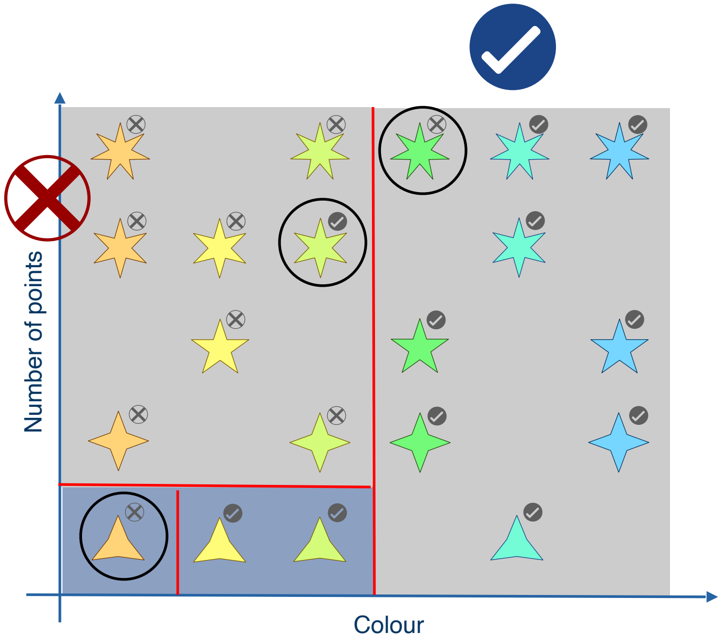


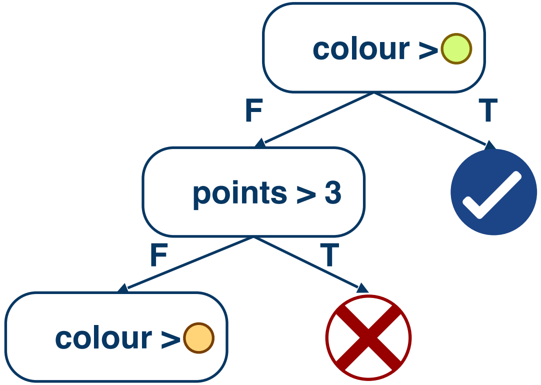
Partition further?



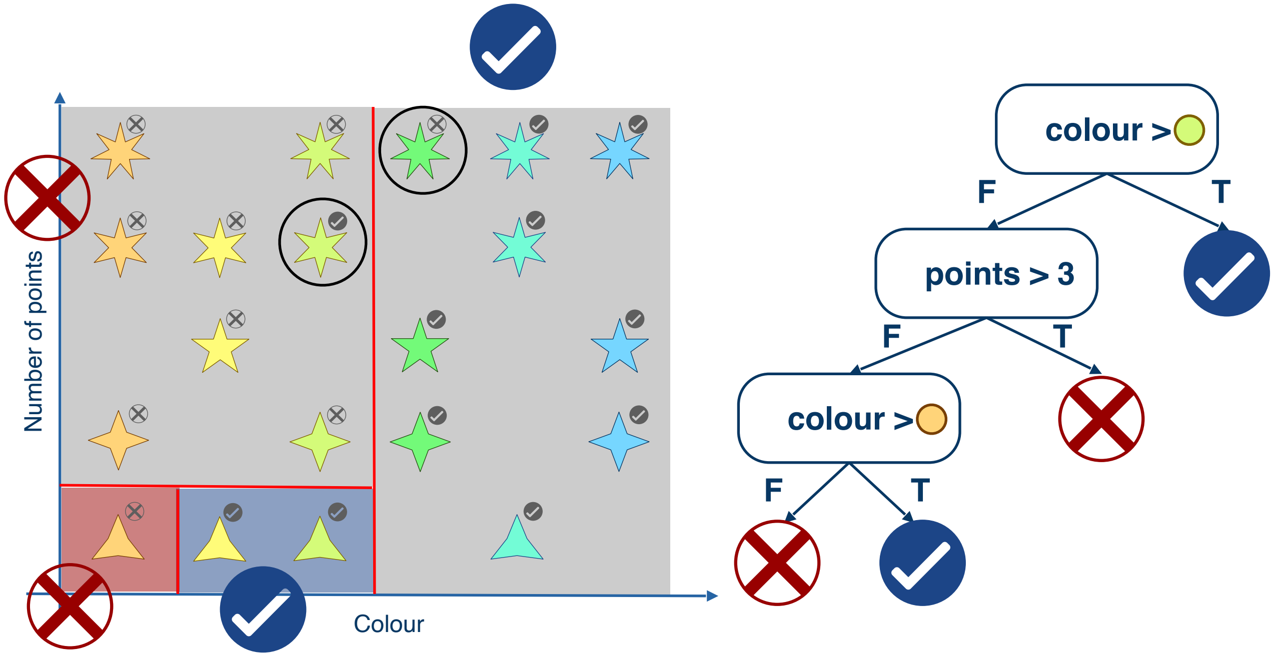


Partition further?

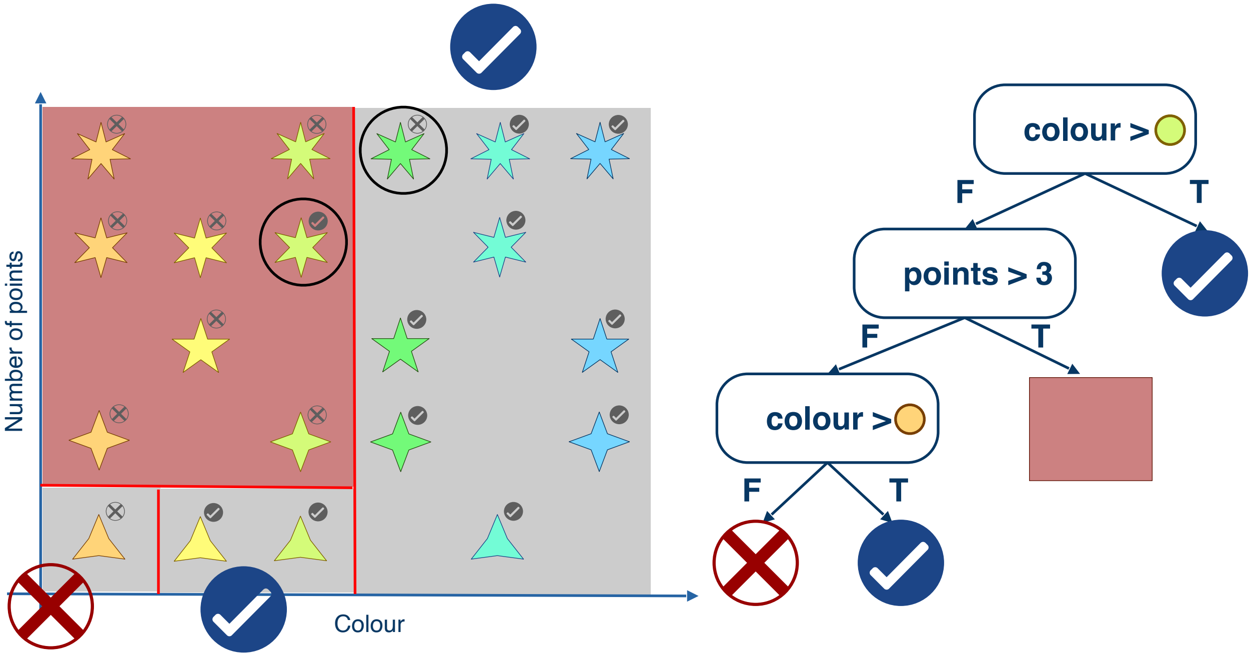




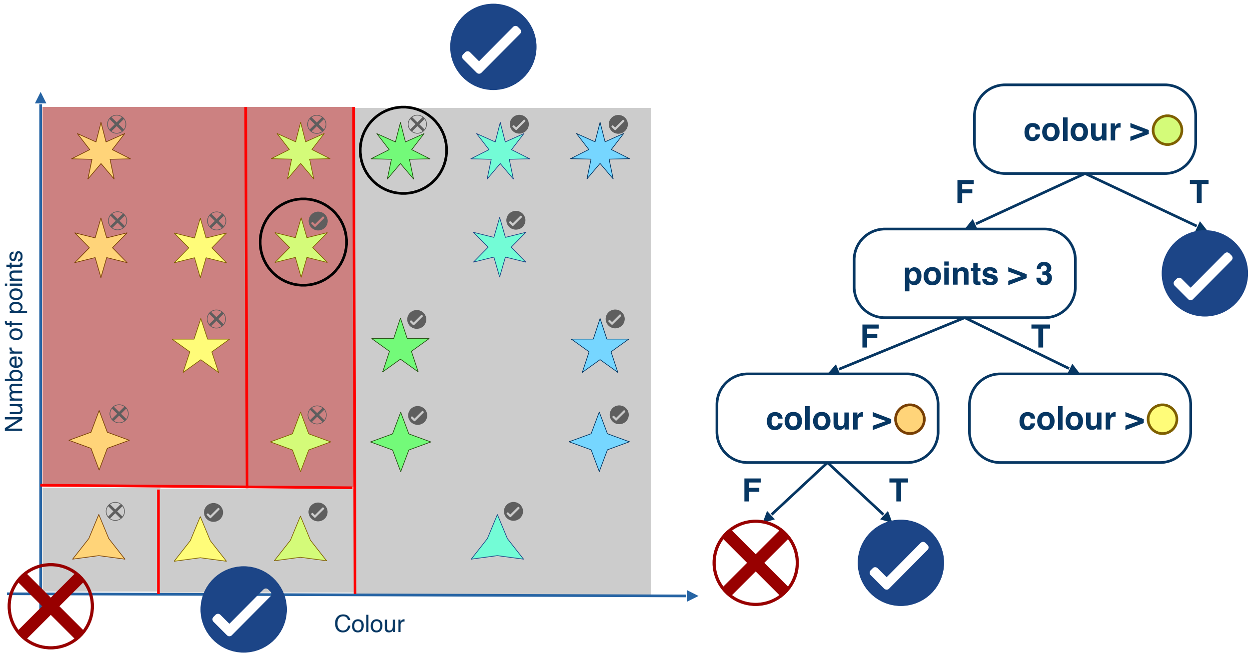
The last two partitions are now “pure”.



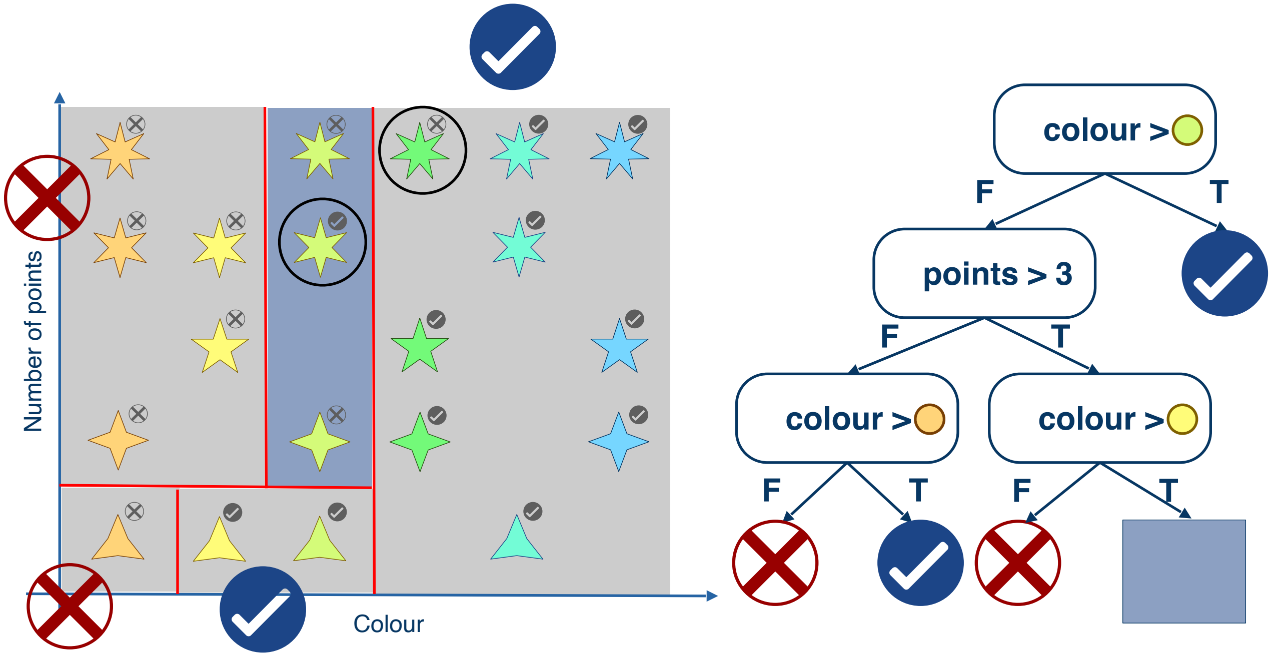
Repeat on other partitions.



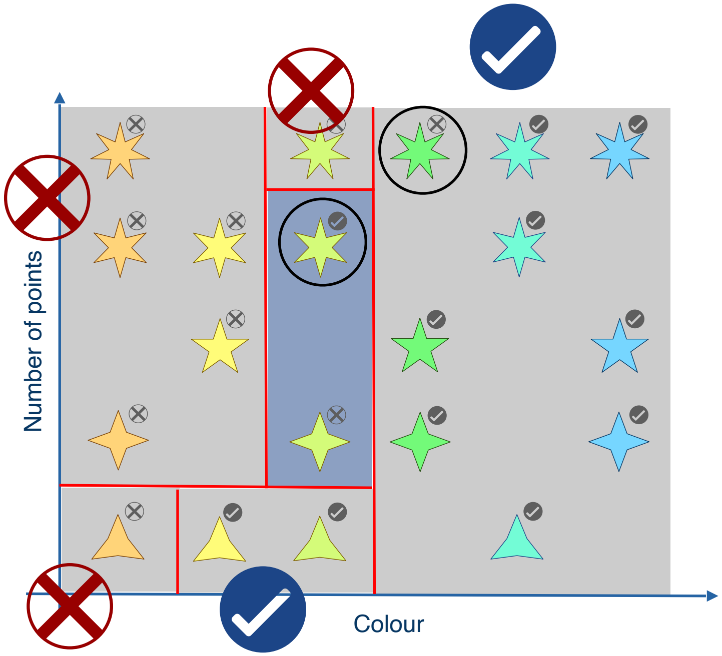
Repeat on other partitions.

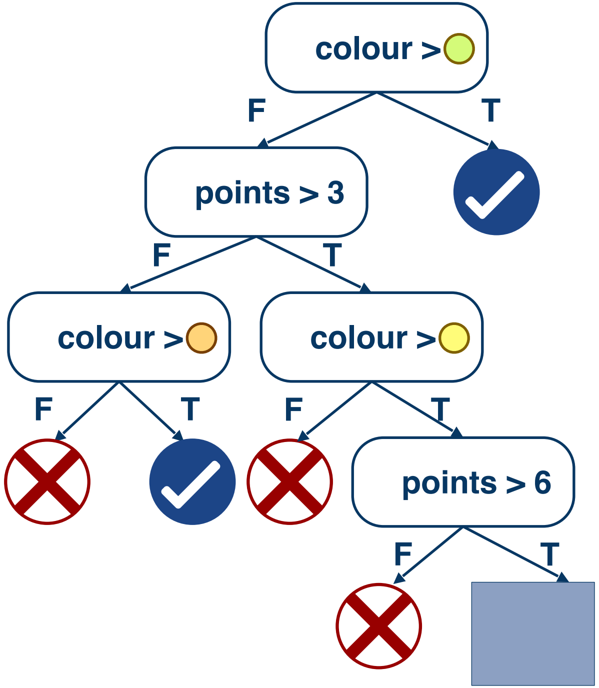


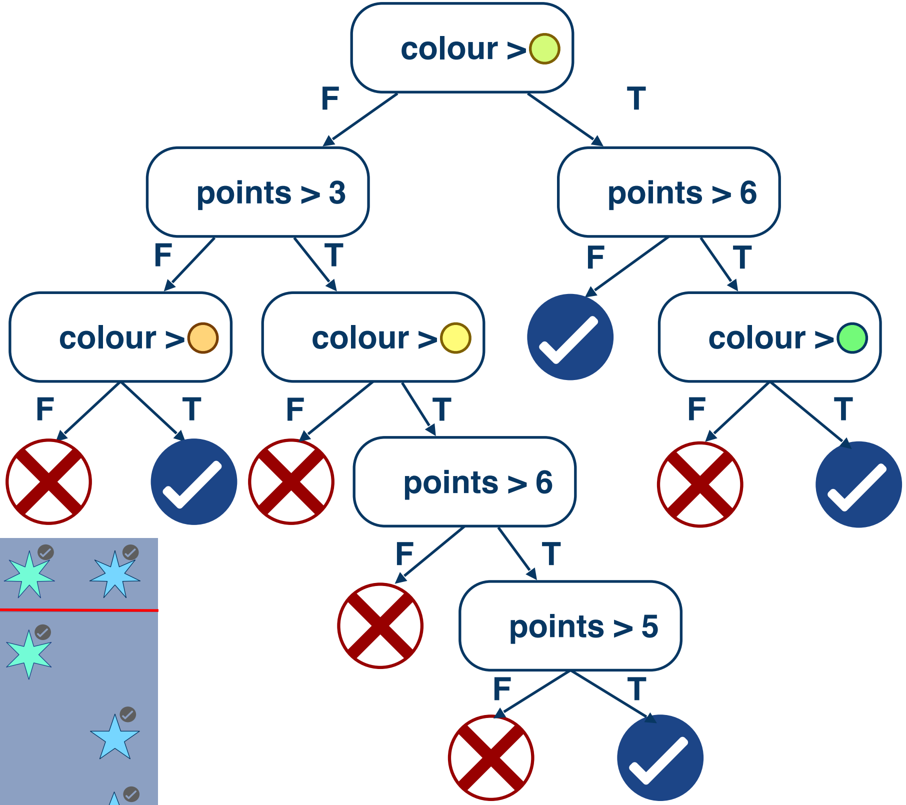
Repeat on other partitions.

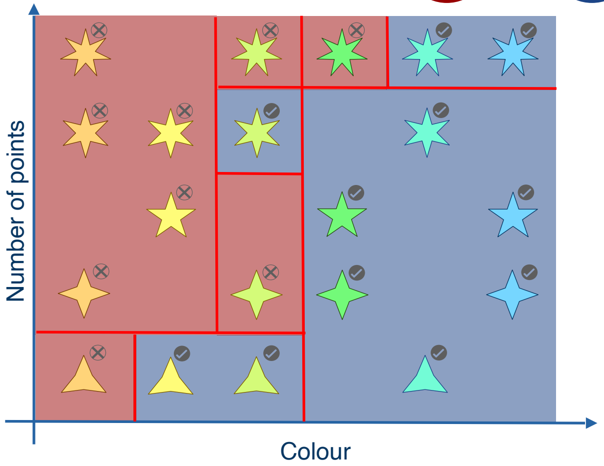


Repeat on other partitions.

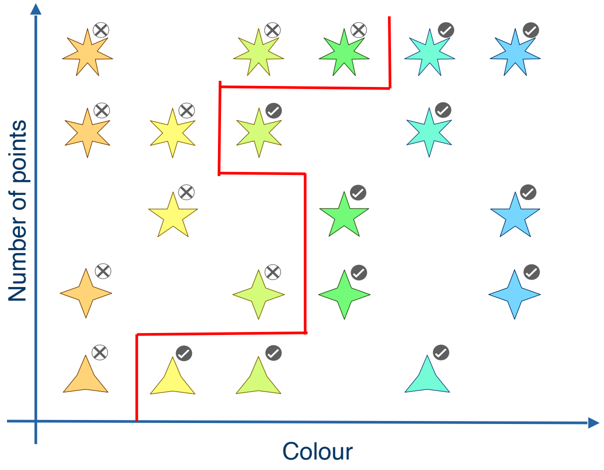




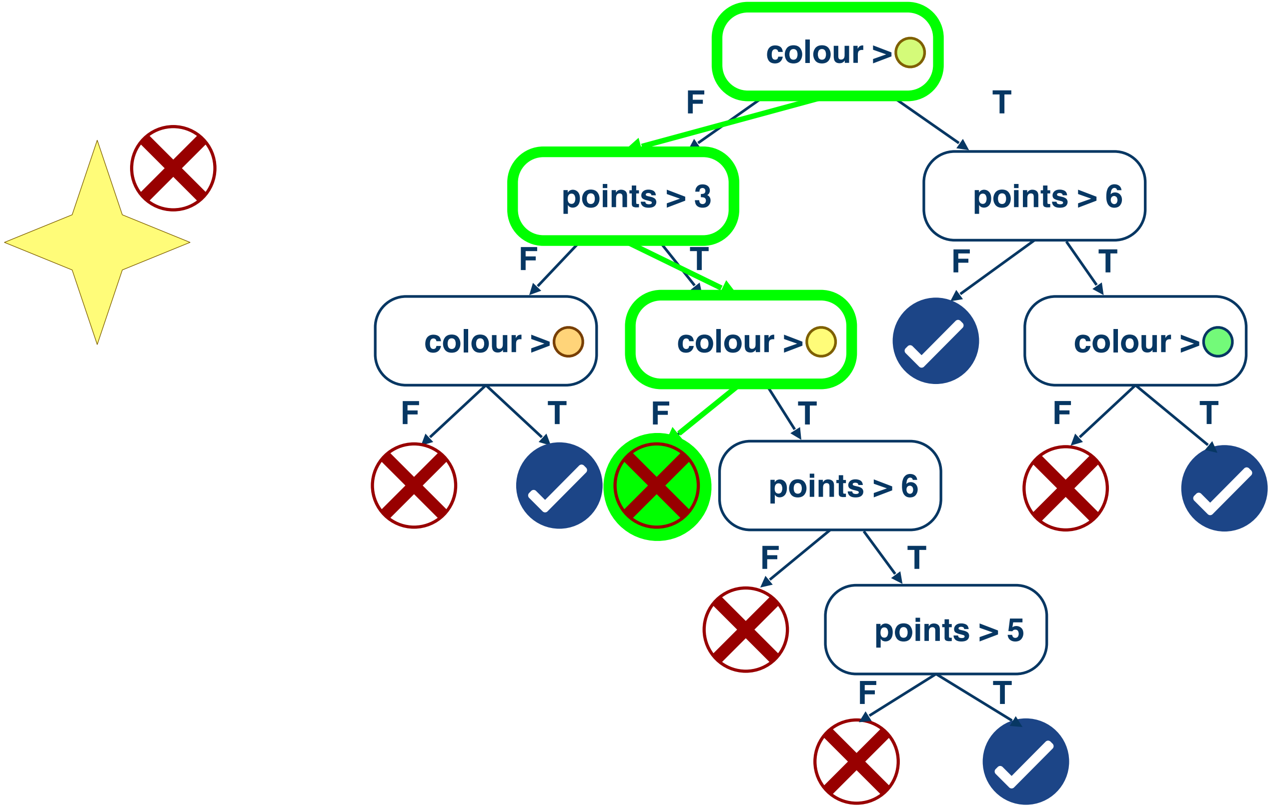

colour >
F
T
points > 3
points > 6
F
T
F
T
colour >
colour >
colour >
F
T
F
T
F
T
points > 6
F
T
points > 5
F
T


Number of points
Colour

Decision Boundary for decision trees



Perform prediction.



Decision Tree learning

- Decision Tree learning (or construction/induction) is a method for approximating discrete classification functions by means of a tree-based representation
- A decision tree can be represented as a set of if-then rules
- Decision Tree learning algorithms employ top-down greedy search through the space of possible solutions
- Algorithms: ID3, C4.5, CART

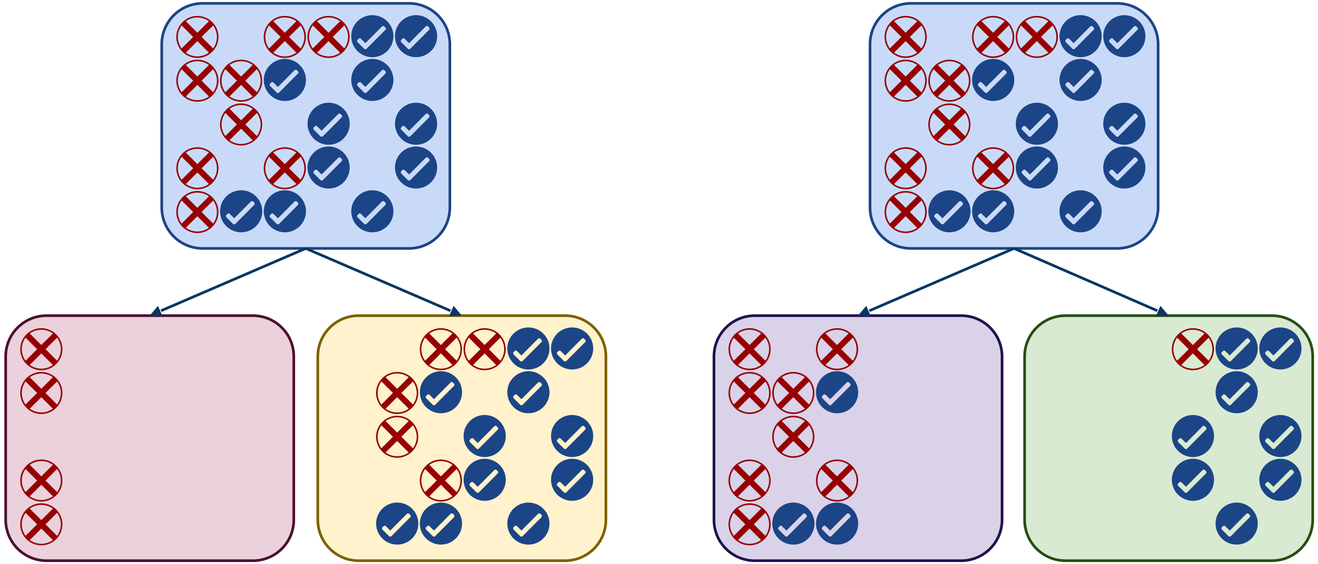
Decision Tree learning: General algorithm

1. Search for an ‘optimal’ splitting rule on training data
2. Split your dataset according to your chosen splitting rule
3. Repeat 1. and 2. on each new splitted subset

To be continued…

(How to select the ‘optimal’ split rule?)

How to select the ‘optimal’ split rule?



Intuitively:
Want partitioned datasets that are more ‘pure’ (as a whole) than the original dataset

Selecting the optimal splitting rule

- Several metrics exist:
  - **Information gain**: Used in ID3, C4.5
    - Quantifies the reduction of information entropy
  - **Gini impurity**: Used in CART
    - If I randomly pick a point, and randomly classify it to a label according to the class label distribution, what is the probability of me getting the label incorrect?
  - **Variance reduction**: Used in CART
    - Mainly used for regression trees where the target variable is continuous

Information Entropy

Information Entropy

Claude Shannon (1916-2001)
"Father of Information Theory"

- How to best to encode information a sender wants to transmit?
- Data compression

Information Entropy

- Entropy is a measure of the **uncertainty** of a random variable
- It can also be seen as the average/ expected amount of information required to fully define a random state (or variable)

Low entropy ⟷ High entropy

Low entropy	High entropy
Certain	*Uncertain*
Boring	*Surprise*
Predictable	*Unpredictable*

I stored my key in one of these two boxes...

50% 50%

0 1

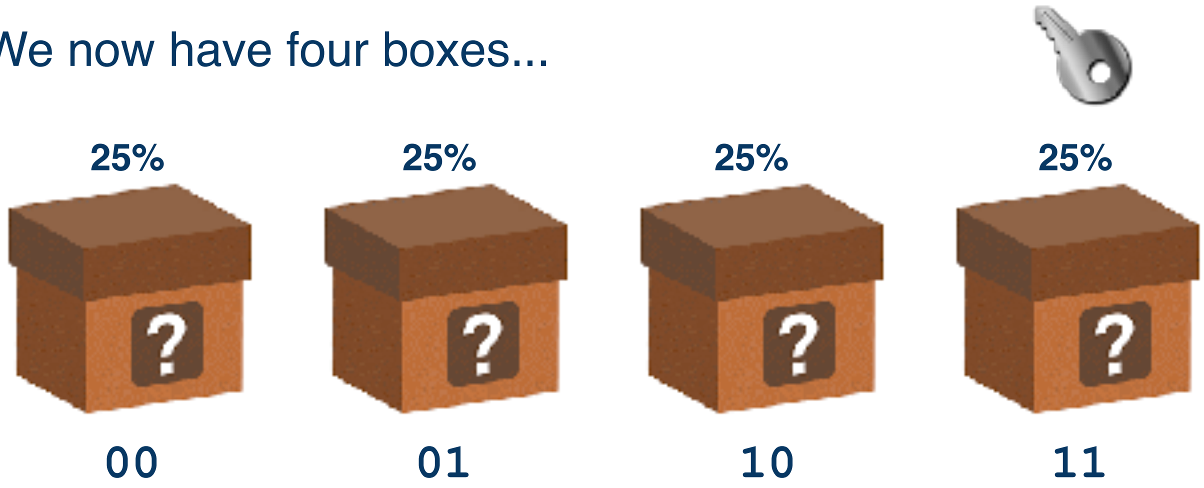
How much information do you need to be fully certain of the key’s location?

“Is it in the left box?”

One bit: **0** or **1**

2 states = 1 bit

We now have four boxes...



How many bits?

Two bits: `00`, `01`, `10`, `11`

2 states = 1 bit
4 states = 2 bits

We now have four boxes...

25% 25% 25% 25%

2^1 = 2 states = 1 bit

2^2 = 4 states = 2 bits

2^3 = 8 states = 3 bits

2^B = K states = B bits

We now have four boxes...

25% 25% 25% 25%

$2^B = K$ states

$B = log_2(K)$

the amount of information required to fully determine the state of a random variable is

$I(x) = log_2(K)$

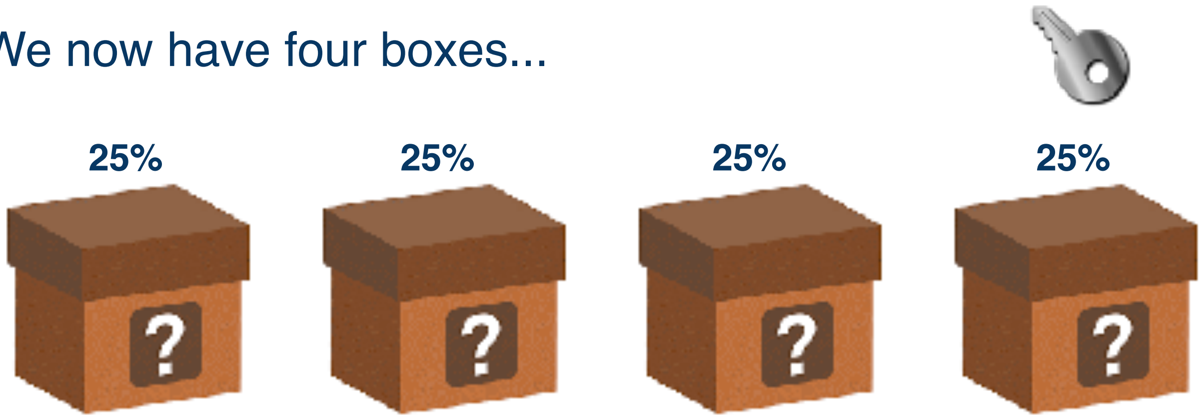
2^1 = 2 states = 1 bit

2^2 = 4 states = 2 bits

2^3 = 8 states = 3 bits

2^B = K states = B bits

We now have four boxes...



$$I(x) = \log_2(K)$$

Probabilistic point of view:

$P(x) = 1/K$ $\qquad$ $K = 1/P(x)$

$I(x) = \log_2(1/P(x)) = -\log_2(P(x))$

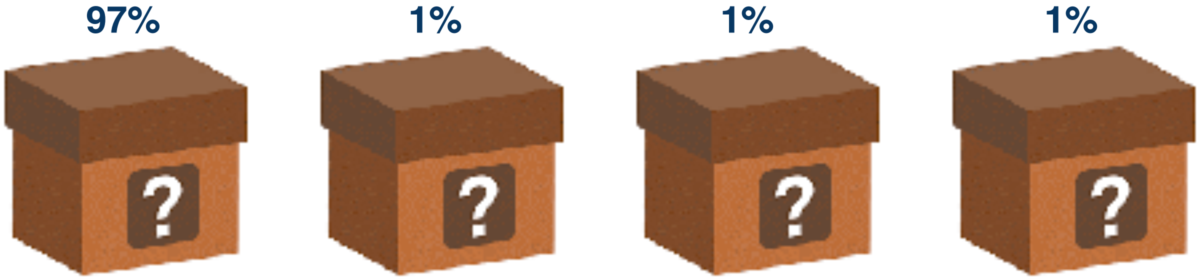
$$I(x=box_1) = I(x=box_2) = I(x=box_3) = I(x=box_4) = -\log_2(0.25)$$
$$= 2 \text{ bits}$$

What if you knew that I (almost) always store my key in box 1?

97% **1%** **1%** **1%**

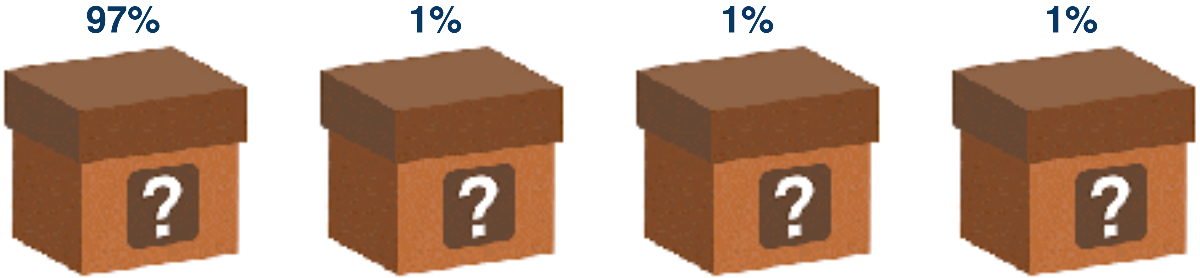
- You can almost be certain that it is in box 1! Telling you this does not give you a lot of new information (*low entropy*)...
- ...but telling you that it's in one of the other boxes represents a very important (or surprising) information (*high entropy*)!

What if you knew that I (almost) always store my key in box 1?



$$I(x=box_1) = -\log_2(0.97) = \boxed{0.0439\ bits}$$
$$I(x=box_2) = -\log_2(0.01) = 6.6439\ bits$$
$$I(x=box_3) = -\log_2(0.01) = 6.6439\ bits$$
$$I(x=box_4) = -\log_2(0.01) = 6.6439\ bits$$

How much information ***on average***?



- 97% of the time, you will not need much information
- Entropy is defined as the ***average*** amount of information:

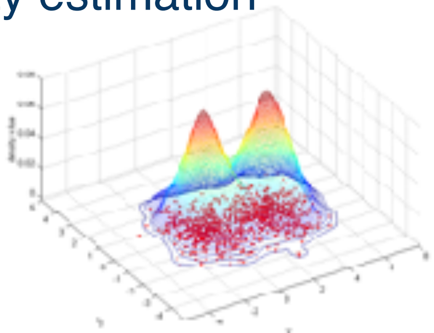
$$H(X) = -\sum_k^K P(x_k)\, log_2(P(x_k))$$

In most cases(i.e., 97% time), only small amount of information is needed to determine the exact location of the box

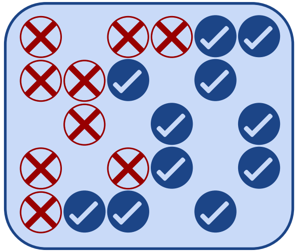
$$H(X) = -0.97 \times log_2(0.97) - 0.01 \times log_2(0.01) - 0.01 \times log_2(0.01) - 0.01 \times log_2(0.01) = 0.2419\, bits$$

Continuous Entropy

- For a probability density function *f(x)*, we can define the continuous entropy, as an analogy of Shanon's definition: $H(X) = -\int_x f(x)\, log_2(f(x))$
  - This analogy is imperfect (it can have negative values) but is still often used in Deep Learning.
  - The probability density function is often unknown, but can be approximated with density estimation algorithms for instance.



How to use ENTROPY to select the ‘optimal’ split rule?

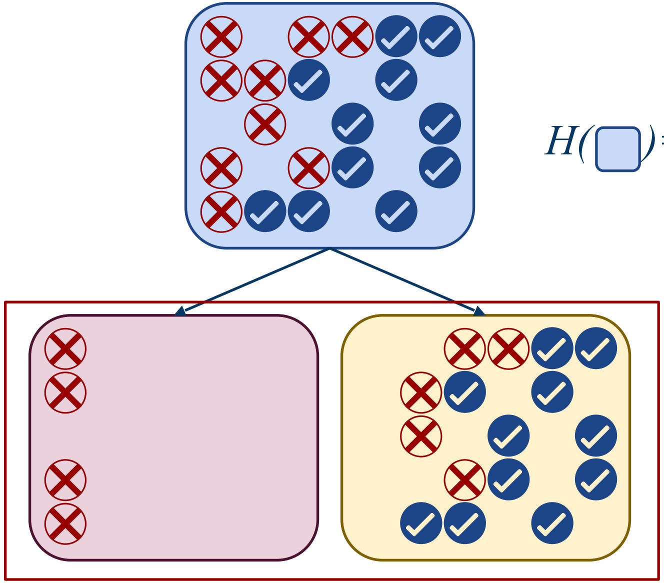


$$P(\checkmark) = 11/20$$

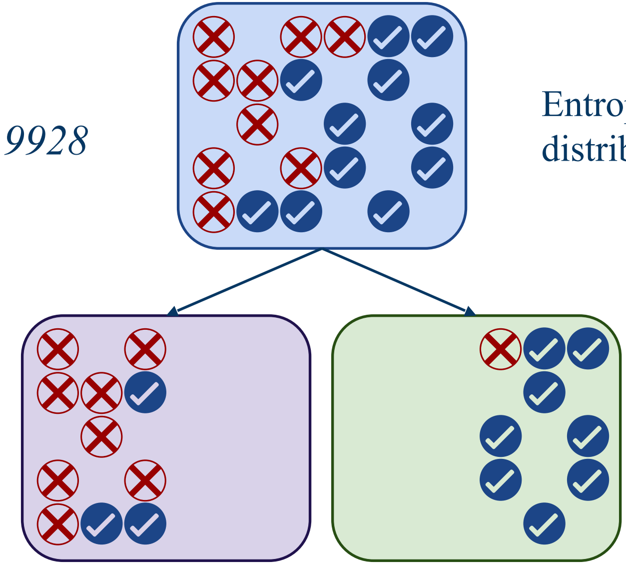
$$P(\times) = 9/20$$

$$H(\square) = -\ 11/20 * \log_2(11/20) - 9/20 * \log_2(9/20)$$

$$H(\square) = 0.9928$$



$H(\square)=0.9928$



Entropy = 1 means almost 50/50 distribution of data. No certainty at all.

$H(\square)=0$ $\quad H(\square)=0.8960$ $\quad H(\square)=0.8454$ $\quad H(\square)=0.5033$

$$H(\{\square,\square\})=4/20 * 0 + 16/20 * 0.8960 = 0.7168$$

$$H(\{\square,\square\})=11/20 * 0.8454 + 9/20 * 0.5033 = 0.6915$$

$$H(\square) - H(\{\square,\square\})=\mathbf{0.2760}$$

$$H(\square) - H(\{\square,\square\})=\mathbf{0.3013}$$

Information Gain = Amount of certainty gained from the parent state to the child state

Information Gain

Information Gain

- Information Gain is the difference between the initial entropy and the (weighted) average entropy of the produced subsets.

$$IG(dataset, subsets) = H(dataset) - \sum_{S \in subsets} \frac{|S|}{|dataset|} H(S)$$

$$|dataset| = \sum_{S \in subsets} |S|$$

- For a binary tree,

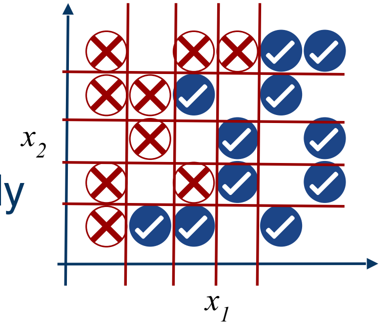
$$IG(dataset, subsets) = H(dataset) - \left(\frac{|S_{left}|}{|dataset|} H(S_{left}) + \frac{|S_{right}|}{|dataset|} H(S_{right}) \right)$$

$$|dataset| = |S_{left}| + |S_{right}|$$

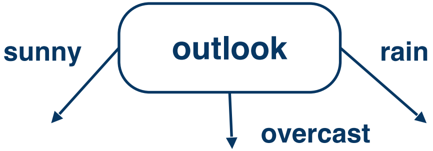
- Select split rule that maximises $IG(dataset, subsets)$

Different types of inputs

- **Ordered values** (e.g. real values)
  - attribute and split point (weight < 60)
  - for each feature, sort its values, and consider only split points that are between two examples with different class labels



- **Categorical/symbolic values**
  - search for the most informative feature and then create as many branches as there are different values for this feature



	PlayTennis(d)	*outlook*	*temperature*	*humidity*
1	0	sunny	hot	high
2	0	sunny	hot	high
...	...	...	...	...
13	1	overcast	hot	normal
14	0	rain	mild	high

To be continued…
(Worked example for constructing decision tree)

Worked example for constructing decision tree (categorical attributes)

	PlayTennis(d)	*outlook*	*temperature*	*humidity*	*wind*
1	0	sunny	hot	high	weak
2	0	sunny	hot	high	strong
3	1	overcast	hot	high	weak
4	1	rain	mild	high	weak
5	1	rain	cool	normal	weak
6	0	rain	cool	normal	strong
7	1	overcast	cool	normal	strong
8	0	sunny	mild	high	weak
9	1	sunny	cool	normal	weak
10	1	rain	mild	normal	weak
11	1	sunny	mild	normal	strong
12	1	overcast	mild	high	strong
13	1	overcast	hot	normal	weak
14	0	rain	mild	high	strong

Play Tennis (Tom Mitchell's book, p.59)

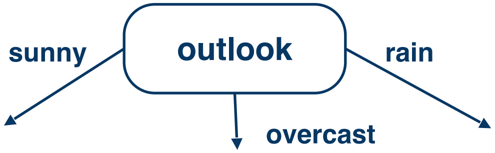
Which attribute to select as split rule?

	PlayTennis(d)	*outlook*
1	0	sunny
2	0	sunny
3	1	overcast
4	1	rain
5	1	rain
6	0	rain
7	1	overcast
8	0	sunny
9	1	sunny
10	1	rain
11	1	sunny
12	1	overcast
13	1	overcast
14	0	rain

- Which attribute to select as split rule?
- First need to compute:
  - *IG(D, outlook)*
  - *IG(D, temperature)*
  - *IG(D, humidity)*
  - *IG(D, wind)*
- … select the attribute with the highest Information Gain.

D

	PlayTennis(d)	outlook
1	0	sunny
2	0	sunny
3	1	overcast
4	1	rain
5	1	rain
6	0	rain
7	1	overcast
8	0	sunny
9	1	sunny
10	1	rain
11	1	sunny
12	1	overcast
13	1	overcast
14	0	rain



$$H(D) = -\frac{9}{14} \times log_2(\frac{9}{14}) - \frac{5}{14} \times log_2(\frac{5}{14}) = 0.940$$

Calculate D {0, 1} distribution

$$H(D_{sunny}) = -\frac{2}{5} \times log_2(\frac{2}{5}) - \frac{3}{5} \times log_2(\frac{3}{5}) = 0.971$$

$$H(D_{overcast}) = -\frac{4}{4} \times log_2(\frac{4}{4}) - \frac{0}{4} \times log_2(\frac{0}{4}) = 0$$

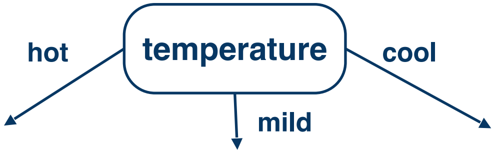
$$H(D_{rain}) = -\frac{3}{5} \times log_2(\frac{3}{5}) - \frac{2}{5} \times log_2(\frac{2}{5}) = 0.971$$

Calculate {sunny, 1} and {sunny, 0} distribution

$$\begin{aligned} & IG(D, outlook) \\ & = H(D) - (\tfrac{5}{14} H(D_{sunny}) + \tfrac{4}{14} H(D_{overcast}) + \tfrac{5}{14} H(D_{rain})) \\ & = 0.940 - (\tfrac{5}{14} \times 0.971 + \tfrac{4}{14} \times 0 + \tfrac{5}{14} \times 0.971) \\ & = 0.940 - 0.694 \\ & = 0.246 \end{aligned}$$

D

	PlayTennis(d)	*temperature*
1	0	hot
2	0	hot
3	1	hot
4	1	mild
5	1	cool
6	0	cool
7	1	cool
8	0	mild
9	1	cool
10	1	mild
11	1	mild
12	1	mild
13	1	hot
14	0	mild



$$H(D) = -\frac{9}{14} \times log_2(\frac{9}{14}) - \frac{5}{14} \times log_2(\frac{5}{14}) = 0.940$$

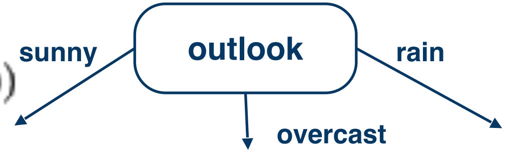
$$H(D_{hot}) = -\frac{2}{4} \times log_2(\frac{2}{4}) - \frac{2}{4} \times log_2(\frac{2}{4}) = 1$$

$$H(D_{mild}) = -\frac{4}{6} \times log_2(\frac{4}{6}) - \frac{2}{6} \times log_2(\frac{2}{6}) = 0.918$$

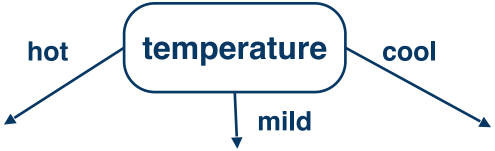
$$H(D_{cool}) = -\frac{3}{4} \times log_2(\frac{3}{4}) - \frac{1}{4} \times log_2(\frac{1}{4}) = 0.811$$

$$\begin{aligned} &IG(D, temp.) \\ &= H(D) - (\tfrac{4}{14} H(D_{hot}) + \tfrac{6}{14} H(D_{mild}) + \tfrac{4}{14} H(D_{cool})) \\ &= 0.940 - (\tfrac{4}{14} \times 1 + \tfrac{6}{14} \times 0.918 + \tfrac{4}{14} \times 0.811) \\ &= 0.940 - 0.911 \\ &= 0.029 \end{aligned}$$

$$IG(D, outlook)$$
$$= H(D) - \left(\tfrac{5}{14} H(D_{sunny}) + \tfrac{4}{14} H(D_{overcast}) + \tfrac{5}{14} H(D_{rain})\right)$$
$$= 0.940 - \left(\tfrac{5}{14} \times 0.971 + \tfrac{4}{14} \times 0 + \tfrac{5}{14} \times 0.971\right)$$
$$= \boxed{0.246}$$

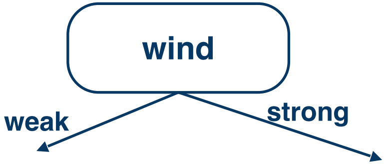


$$IG(D, temperature)$$
$$= H(D) - \left(\tfrac{4}{14} H(D_{hot}) + \tfrac{6}{14} H(D_{mild}) + \tfrac{4}{14} H(D_{cool})\right)$$
$$= 0.940 - \left(\tfrac{4}{14} \times 1 + \tfrac{6}{14} \times 0.918 + \tfrac{4}{14} \times 0.811\right)$$
$$= 0.029$$

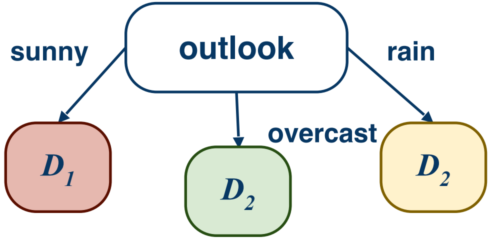


$$IG(D, humidity)$$
$$= H(D) - \left(\tfrac{7}{14} H(D_{high}) + \tfrac{7}{14} H(D_{normal})\right)$$
$$= 0.940 - \left(\tfrac{7}{14} \times 0.985 + \tfrac{7}{14} \times 0.591\right)$$
$$= 0.151$$

$$IG(D, wind)$$
$$= H(D) - \left(\tfrac{8}{14} H(D_{weak}) + \tfrac{6}{14} H(D_{strong})\right)$$
$$= 0.940 - \left(\tfrac{8}{14} \times 0.811 + \tfrac{6}{14} \times 1\right)$$
$$= 0.048$$



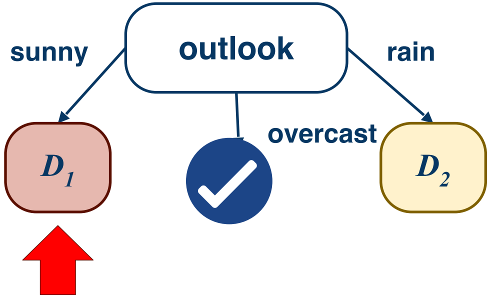
		PlayTennis(d)	outlook	temperature	humidity	wind
	1	0	sunny	hot	high	weak
	2	0	sunny	hot	high	strong
D_1	8	0	sunny	mild	high	weak
	9	1	sunny	cool	normal	weak
	11	1	sunny	mild	normal	strong
	3	1	overcast	hot	high	weak
D_2	7	1	overcast	cool	normal	strong
	12	1	overcast	mild	high	strong
	13	1	overcast	hot	normal	weak
	4	1	rain	mild	high	weak
	5	1	rain	cool	normal	weak
D_3	6	0	rain	cool	normal	strong
	10	1	rain	mild	normal	weak
	14	0	rain	mild	high	strong



	PlayTennis(d)	outlook	temperature	humidity	wind
1	0	sunny	hot	high	weak
2	0	sunny	hot	high	strong
8	0	sunny	mild	high	weak
9	1	sunny	cool	normal	weak
11	1	sunny	mild	normal	strong
3	1	overcast	hot	high	weak
7	1	overcast	cool	normal	strong
12	1	overcast	mild	high	strong
13	1	overcast	hot	normal	weak
4	1	rain	mild	high	weak
5	1	rain	cool	normal	weak
6	0	rain	cool	normal	strong
10	1	rain	mild	normal	weak
14	0	rain	mild	high	strong

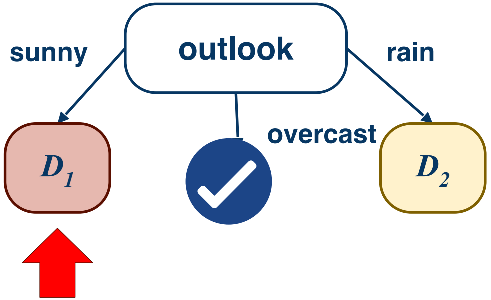
D_1

D_3



D_1

	PlayTennis(d)	outlook	temperature	humidity	wind
1	0	sunny	hot	high	weak
2	0	sunny	hot	high	strong
8	0	sunny	mild	high	weak
9	1	sunny	cool	normal	weak
11	1	sunny	mild	normal	strong



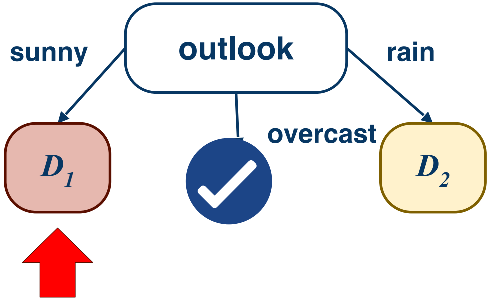
$$H(D_1) = -\frac{2}{5} \times log_2(\frac{2}{5}) - \frac{3}{5} \times log_2(\frac{3}{5}) = 0.971$$

$$IG(D_1, temperature)$$
$$= H(D_1) - (\frac{2}{5}H(D_{1,hot}) + \frac{2}{5}H(D_{1,mild}) + \frac{1}{5}H(D_{1,cool}))$$
$$= 0.971 - (\frac{2}{5} \times 0 + \frac{2}{5} \times 1 + \frac{1}{5} \times 0)$$
$$= 0.571$$

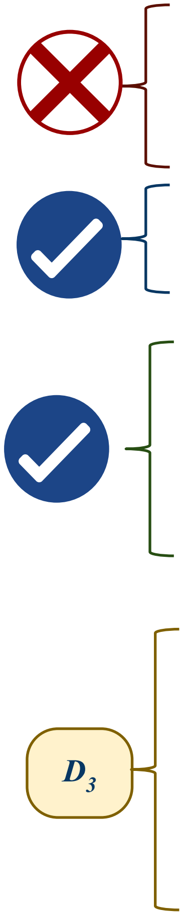
$$IG(D_1, humidity)$$
$$= H(D_1) - (\frac{3}{5}H(D_{1,high}) + \frac{2}{5}H(D_{1,normal}))$$
$$= 0.971 - (\frac{3}{5} \times 0 + \frac{2}{5} \times 0)$$
$$= \boxed{0.971}$$

$$IG(D_1, wind)$$
$$= H(D_1) - (\frac{3}{5}H(D_{1,weak}) + \frac{2}{5}H(D_{1,strong}))$$
$$= 0.971 - (\frac{3}{5} \times 0.918 + \frac{2}{5} \times 1)$$
$$= 0.020$$

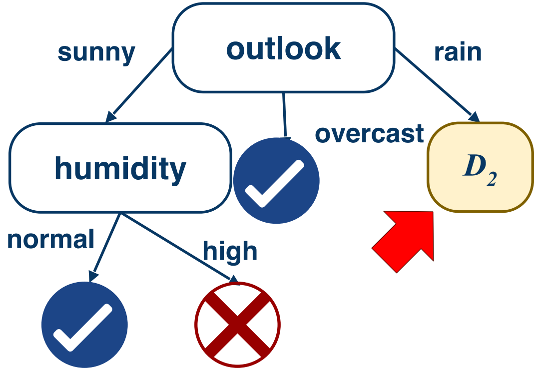
	PlayTennis(d)	outlook	temperature	humidity	wind
1	0	sunny	hot	high	weak
2	0	sunny	hot	high	strong
8	0	sunny	mild	high	weak
9	1	sunny	cool	normal	weak
11	1	sunny	mild	normal	strong
3	1	overcast	hot	high	weak
7	1	overcast	cool	normal	strong
12	1	overcast	mild	high	strong
13	1	overcast	hot	normal	weak
4	1	rain	mild	high	weak
5	1	rain	cool	normal	weak
6	0	rain	cool	normal	strong
10	1	rain	mild	normal	weak
14	0	rain	mild	high	strong

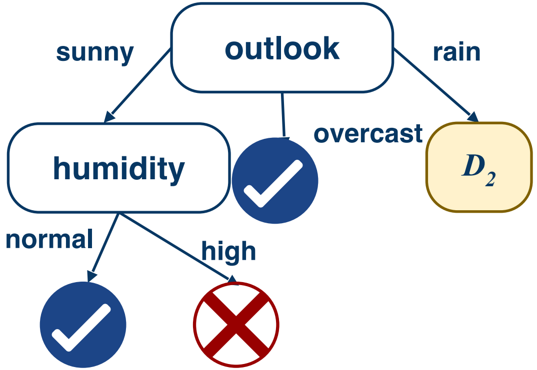


$$IG(D_1, humidity) = 0.971$$



	PlayTennis(d)	outlook	temperature	humidity	wind
1	0	sunny	hot	high	weak
2	0	sunny	hot	high	strong
8	0	sunny	mild	high	weak
9	1	sunny	cool	normal	weak
11	1	sunny	mild	normal	strong
3	1	overcast	hot	high	weak
7	1	overcast	cool	normal	strong
12	1	overcast	mild	high	strong
13	1	overcast	hot	normal	weak
4	1	rain	mild	high	weak
5	1	rain	cool	normal	weak
6	0	rain	cool	normal	strong
10	1	rain	mild	normal	weak
14	0	rain	mild	high	strong





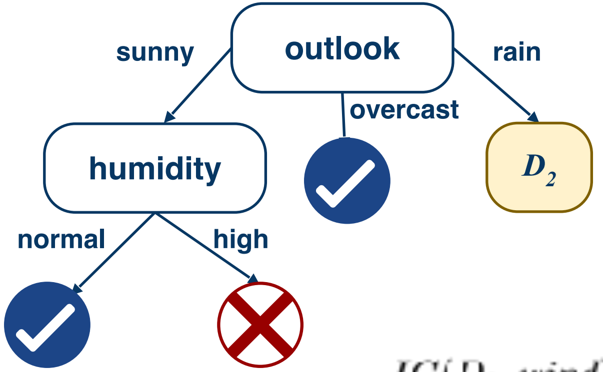
D_2	*PlayTennis(d)*	*outlook*	*temperature*	*humidity*	*wind*
4	1	rain	mild	high	weak
5	1	rain	cool	normal	weak
6	0	rain	cool	normal	strong
10	1	rain	mild	normal	weak
14	0	rain	mild	high	strong

$$H(D_2) = -\frac{3}{5} \times log_2(\frac{3}{5}) - \frac{2}{5} \times log_2(\frac{2}{5}) = 0.971$$

$$\begin{aligned} &IG(D_2, temperature) \\ &= H(D_2) - (\frac{0}{5}H(D_{2,hot}) + \frac{3}{5}H(D_{2,mild}) + \frac{2}{5}H(D_{2,cool})) \\ &= 0.971 - (0 + \frac{3}{5} \times 0.918 + \frac{2}{5} \times 1) \\ &= 0.020 \end{aligned}$$

$$\begin{aligned} &IG(D_2, humidity) \\ &= H(D_2) - (\frac{2}{5}H(D_{2,high}) + \frac{3}{5}H(D_{2,normal})) \\ &= 0.971 - (\frac{2}{5} \times 1 + \frac{3}{5} \times 0.918) \\ &= 0.020 \end{aligned}$$

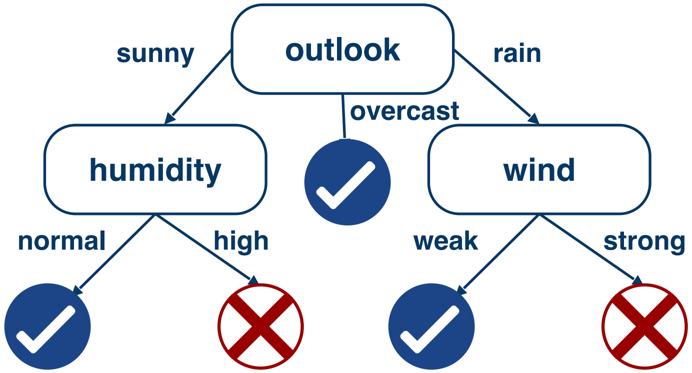
$$\begin{aligned} &IG(D_2, wind) \\ &= H(D_2) - (\frac{3}{5}H(D_{2,weak}) + \frac{2}{5}H(D_{2,strong})) \\ &= 0.971 - (\frac{3}{5} \times 0 + \frac{2}{5} \times 0) \\ &= \boxed{0.971} \end{aligned}$$



$$
\begin{aligned}
&IG(D_2, wind) \\
&= H(D_2) - \left(\tfrac{3}{5} H(D_{2,weak}) + \tfrac{2}{5} H(D_{2,strong})\right) \\
&= 0.971 - \left(\tfrac{3}{5} \times 0 + \tfrac{2}{5} \times 0\right) \\
&= \boxed{0.971}
\end{aligned}
$$

D_2

	PlayTennis(d)	outlook	temperature	humidity	wind
4	1	rain	mild	high	weak
5	1	rain	cool	normal	weak
6	0	rain	cool	normal	strong
10	1	rain	mild	normal	weak
14	0	rain	mild	high	strong



	PlayTennis(d)	*outlook*	*temperature*	*humidity*	*wind*
4	1	rain	mild	high	weak
5	1	rain	cool	normal	weak
10	1	rain	mild	normal	weak
6	0	rain	cool	normal	strong
14	0	rain	mild	high	strong

To be continued…
(Summary and other considerations with decision tree)

Summary and other considerations with decision tree

Summary...

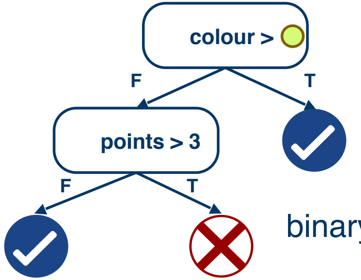
Algorithm: Decision Tree induction

1. Search for an ‘optimal’ splitting rule on training data
2. Split your dataset according to your chosen splitting rule
3. Repeat 1. and 2. on each new splitted subset

Information Gain

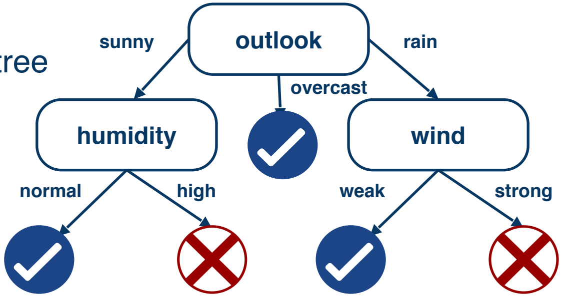
$$IG(dataset, subsets) = H(dataset) - \sum_{S \in subsets} \frac{|S|}{|dataset|} H(S)$$

Categorical vs. Real-valued attributes



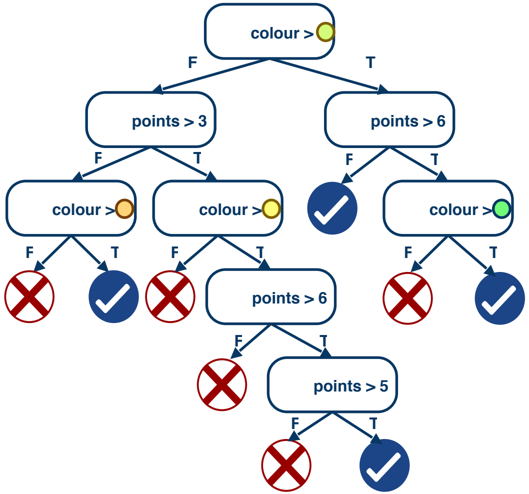
binary tree

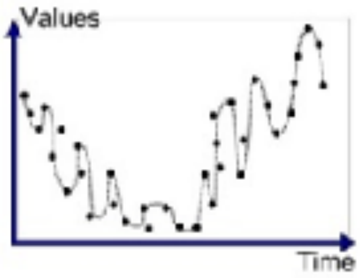
multiway tree



Decision Trees: Overfitting

Like many ML algorithms, decision trees can **overfit**



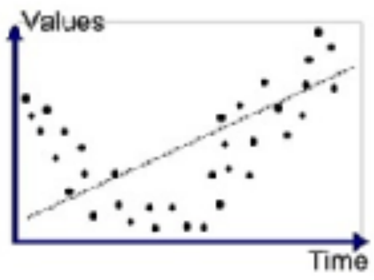


Overfitted

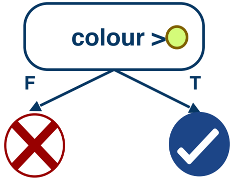
High variance

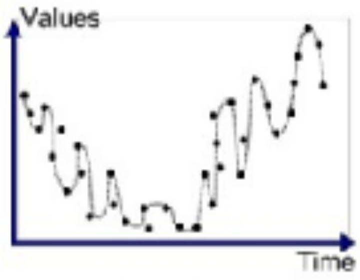
Decision Trees: Overfitting

Like many ML algorithms, decision trees can **overfit**



Underfitted





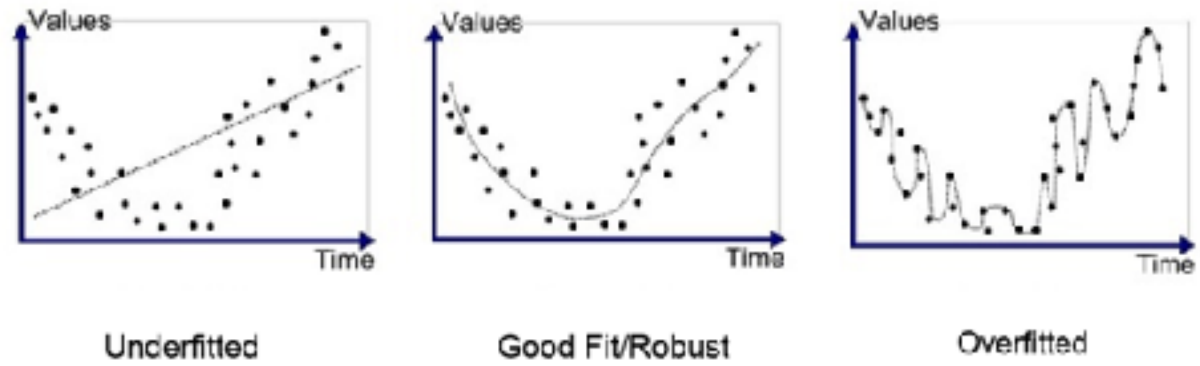
Overfitted

High variance

High bias

Decision Trees: Overfitting

Like many ML algorithms, decision trees can **overfit**



Low variance

High bias

Bias/variance trade-off

High variance

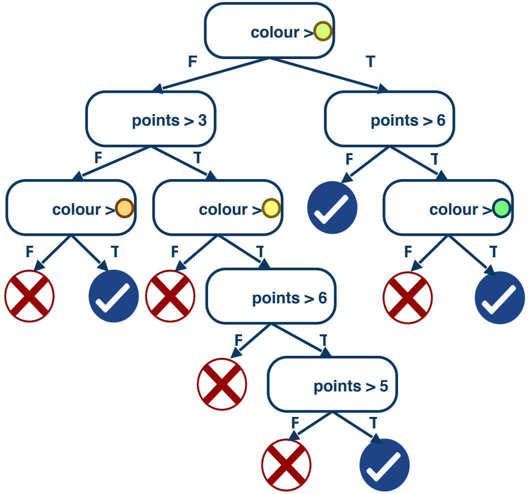
Low bias

Greater k for nearest neighbor
Greater depth for decision tree

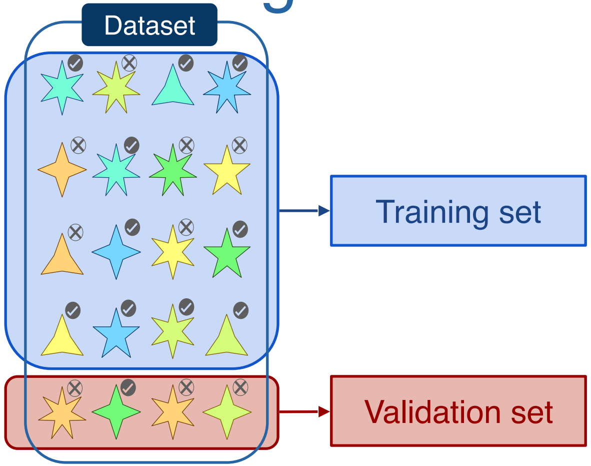
***Ockham's razor**: Simpler hypotheses are generally better than the complex ones*

Decision Trees: Dealing with overfitting

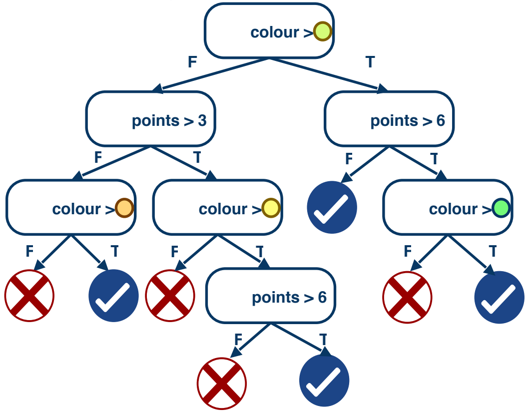
- More on overfitting next week!
- For decision trees:
    - Early stopping
        - max depth, min examples ...
    - Pruning



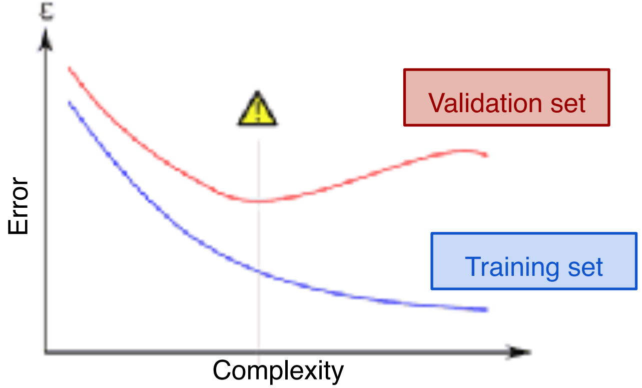
Decision Trees: Dealing with overfitting



- More on overfitting next week!
- For decision trees:
  - Early stopping
    - max depth, min examples ...
  - Pruning

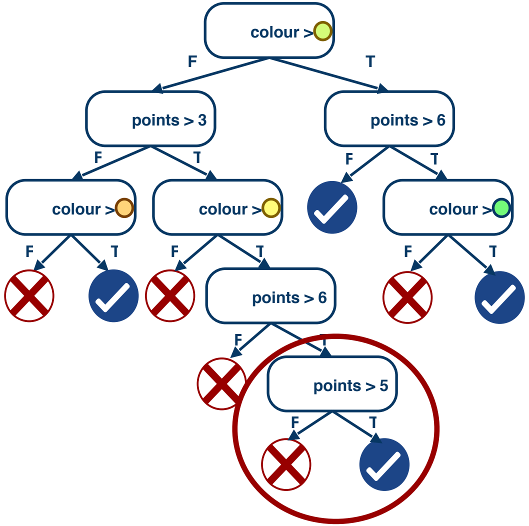


Pruned better than unpruned?



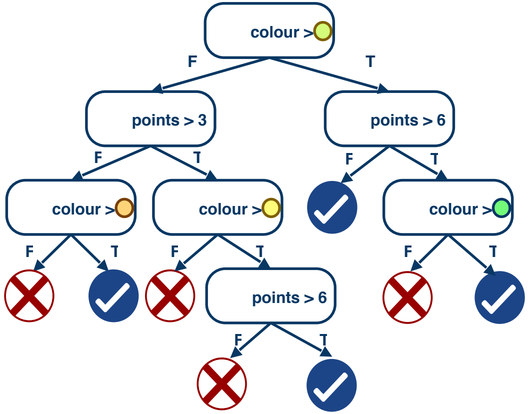
Decision Trees: Pruning

1. Go through each internal node that are connected only to leaf nodes.



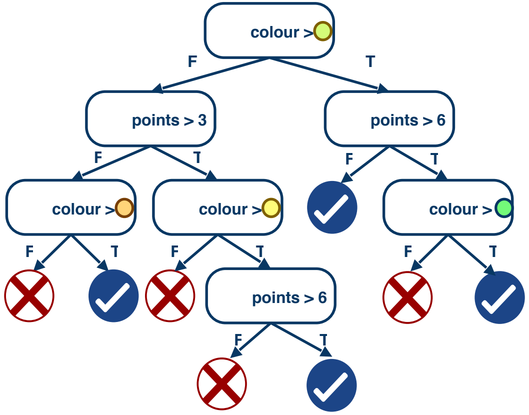
Decision Trees: Pruning

1. Go through each internal node that are connected only to leaf nodes.
2. Turn each into a leaf node (with majority class label)



Decision Trees: Pruning

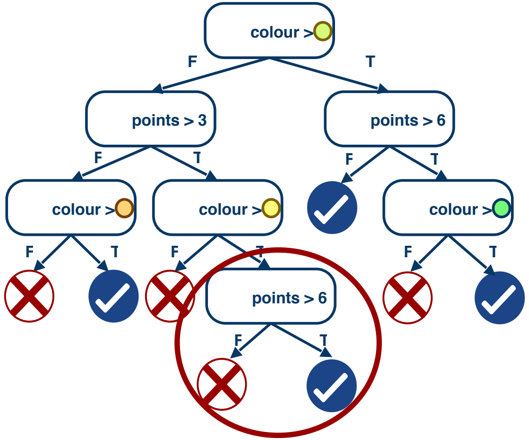
1. Go through each internal node that are connected only to leaf nodes.
2. Turn each into a leaf node (with majority class label)
3. Evaluate pruned tree on validation set. Prune if accuracy higher than unpruned.



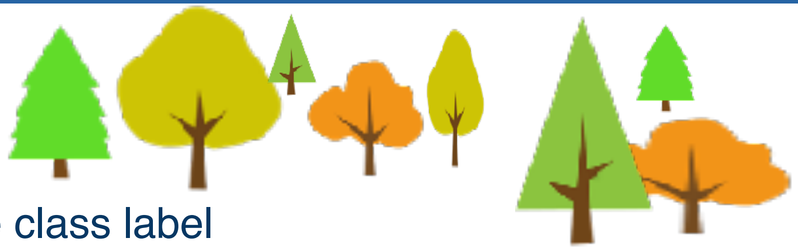
Validation accuracy pruned > validation accuracy old ?

Decision Trees: Pruning

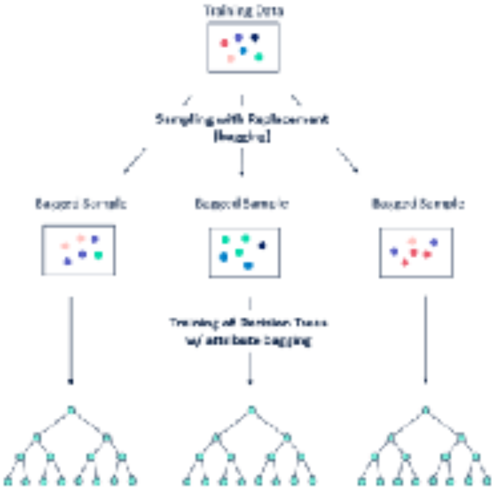
1. Go through each internal node that are connected only to leaf nodes.
2. Turn each into a leaf node (with majority class label).
3. Evaluate pruned tree on validation set. Prune if accuracy higher than unpruned.
4. Repeat until all such nodes have been tested.

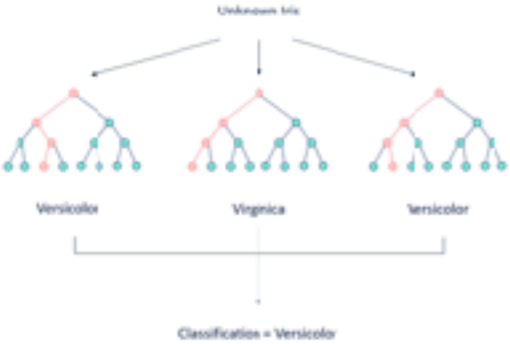


Random Forests



- Many decision trees voting on the class label
- Each tree generated with random sample of training set (bagging) and random subset of features





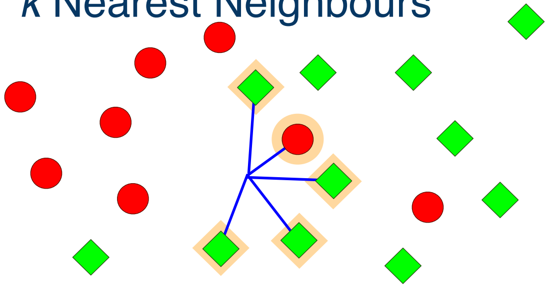
For those interested to learn more about random forests:

- https://community.alteryx.com/t5/Alteryx-Designer-Knowledge-Base/Seeing-the-Forest-for-the-Trees-An-Introduction-to-Random-Forest/ta-p/158062 (above images are taken from here)
- https://victorzhou.com/blog/intro-to-random-forests/

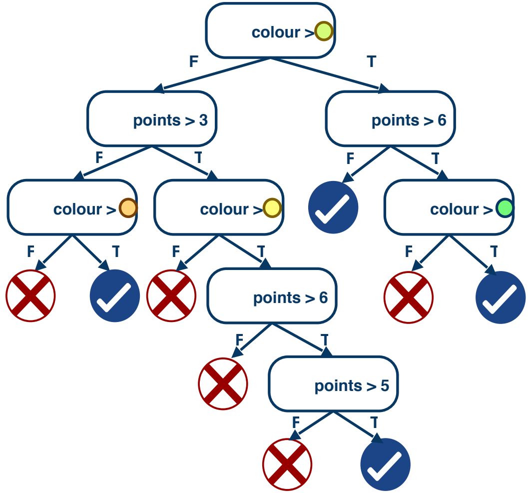
Regression Trees

- Decision trees can also be used for regression (***regression trees***)
- Instead of a class label, each leaf node now predicts a real-valued number
  - Use training examples at leaf node to estimate the output value (e.g. value that minimise min squared error) or learn some linear function of some subset of the numerical features
- Use a different metric for splitting
  - Variance reduction

k Nearest Neighbours



Decision Trees



See you next week!