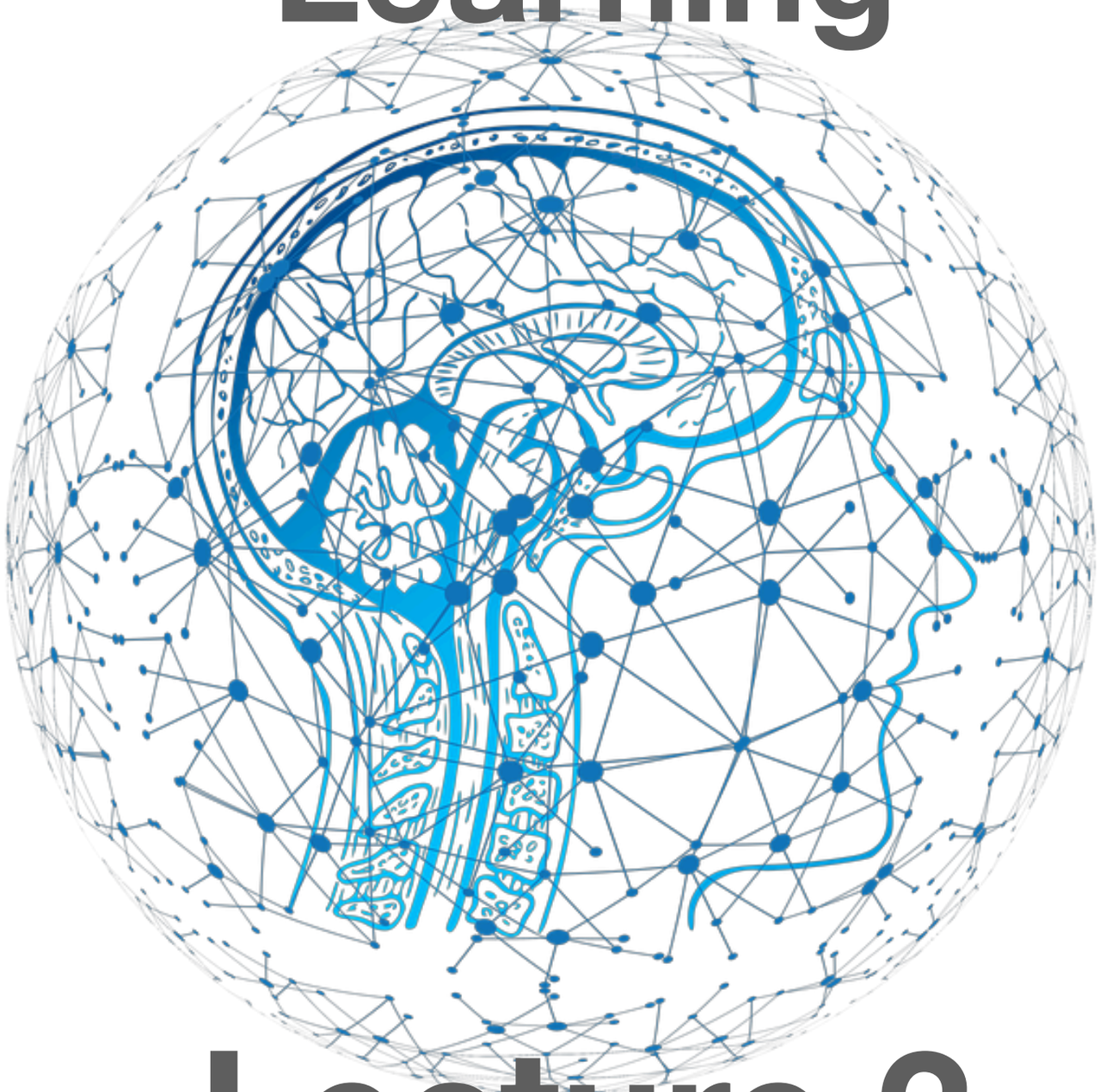
Course 395: Introduction to Machine Learning



Lecture 2

Dr Antoine Cully

Course 395:
Introduction to Machine Learning

Lab sessions every week, starting next week.

- Week 2: Intro and Instance Based Learning (*A. Cully*)
- **Week 3 (today!): Decision Trees & CBC Intro (*A. Cully*)**
- Week 4: Evaluating Hypotheses (*A. Cully*)
- Week 5: Artificial Neural Networks I (*N. Cingillioglu*)
- Week 6: Artificial Neural Networks II (*N. Cingillioglu*)
- Week 7: Unsupervised Learning and Density Estimation (*A. Cully*)
- Week 8: Genetic Algorithms (*A. Cully*)

Lecture Overview?

- Introduction of the CBC
    - Group forming
    - Presentation of the assignments.
- Decision Trees
    - Intuitions and motivations
    - Information Entropy
    - Information Gain
    - Algorithm
    - Example

Computer-Based Courseworks (CBC)

Goals:

- Implement basic machine learning techniques (in Python).
- Learn how to solve problems with machine learning.
- Evaluate your solutions.
- Work in a group.

Group Forming

- Students will be divided in **groups of 4**

- Simply fill the online form **https://tinyurl.com/y9u8kegx** with the following information:

	First Name	Last Name	Student login	Email	Course	CID
Student 1						
Student 2						
Student 3						
Student 4						

- Submit the form by **noon Wednesday January 23th**

- You will be assigned a TH who will email you informing you about your group number and confirming the members of your group by the end of the week

Group Forming

- 4 members per group, this is a hard limit!
  - In other words, groups of 5, 6, 7, etc are not allowed!!

- **You can use the Piazza tool to form groups**!

- If you cannot form a team with 4 members then just fill the form with your information and we will assign you to a team.
  - Case 1 member: We will add you to a group of 2-3 members
  - Case 2 members: 1-2 more members will be added to your group -
  - Case 3 members: 1 member may be added to your group

- Once the groups are formed you cannot change groups

Tutorial Helpers

Nuri Cingillioglu
Daniel Pace
Pedro Martínez Mediano
Akis Kefalas
Arnaud Tournier
Eftychia Fotiadou
Florimond Houssiau
Jeremy Tan
Jingqing Zhang
Kritaphat Songsri-In
Eugene Valassakis
Piyawat Lertvittayakumjorn

A Tutorial Helper (TH) will be assigned to each group.

- In the labs you can ask questions to any TH

Communication

- **Mainly Piazza!**

- Or via e-mail with: machinelearning395@gmail.com
  (**you have to** put your group ID in the title of your e-mail)

Organisation

- Each group must hand in a report of ~4-5 pages (excluding figures) per assignment, including discussion on implementation and answers to questions posed in the manual.
- ONE report per group

- Each group must hand in the code they implemented for each assignment.

- Hand in the code and the reports via CATE.
- This means that you have to create your group in CATE as well!

Assignment hand in

Hand in via CATE

- One group leader per group who submits the report/code and invites the other group members.
- Each and every group member individually has to confirm that s(he) is part of that particular group, for each and every assignment submission (under the pre-determined group leader) before each assignment submission deadline.

Late Submissions

- Up to 24h: Mark is capped to pass (50/40 for MSc/UG students)

- >24h: Mark = 0.

- It’s OK if one member confirms late that he/she belongs to the group.

- If the leader is late then the above penalties apply!!

Report + Code marking

- The THs will mark your report and provide feedback

- The THs will test the implemented algorithms using a separate test set (not available to the students).

- We will inform you about the performance of your algorithm on our test set.

- We just want to check that your code runs and your classifier has been properly trained, i.e., it can generalise to unknown data

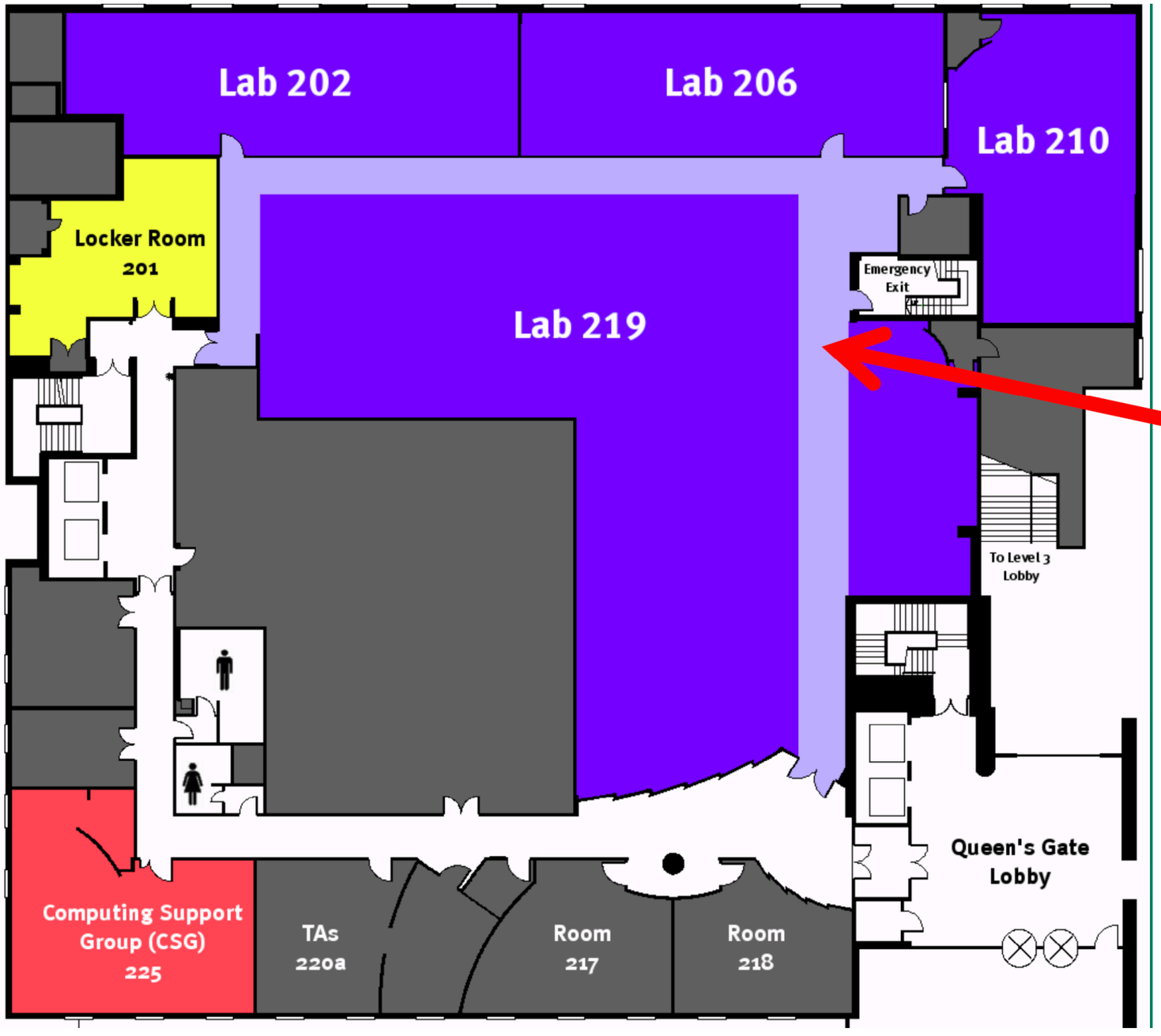
Lab Schedule

Assisted Labs (THs present to answer questions), starting on January 23th

- Every Wednesday 10:00-12:00, labs 202, 206, 219

- Every Thursday 16:00 – 18:00, labs 202, 206, 219

Labs



THs will be here

Deadlines

- Assignment 1: February 11th (Monday) – 7pm
- Assignment 2: March 1st (Friday) – 7pm

Grading

- Every group member is expected to have sufficient contribution to the implementation of every assignment.

- Plagiarism is not allowed!
  Involved groups will be instantly eliminated.

Assignment Grading

Report Content	*Code*	*Report Quality*
75%	*15%*	*10%*

*Group_grade = 0.75*report_content + 0.15*code + 0.1*report_quality*

- *Code Mark:* Depends on the performance on the hidden Test set

e.g. for Decision Trees = performance on hidden Test set + 15

- Make sure your code runs. If not you lose 30% of the code mark.

Assignment Grading

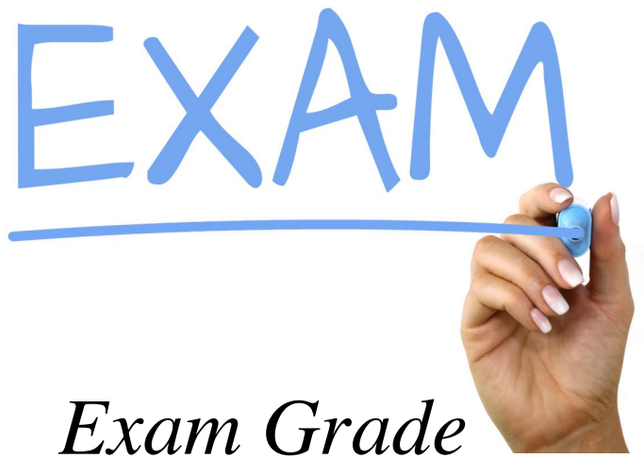
Grade1

Grade2

$$CBC\_grade = 0.4*Grade1 + 0.6*Grade2$$

Machine Learning Grade

CBC Grade
33%

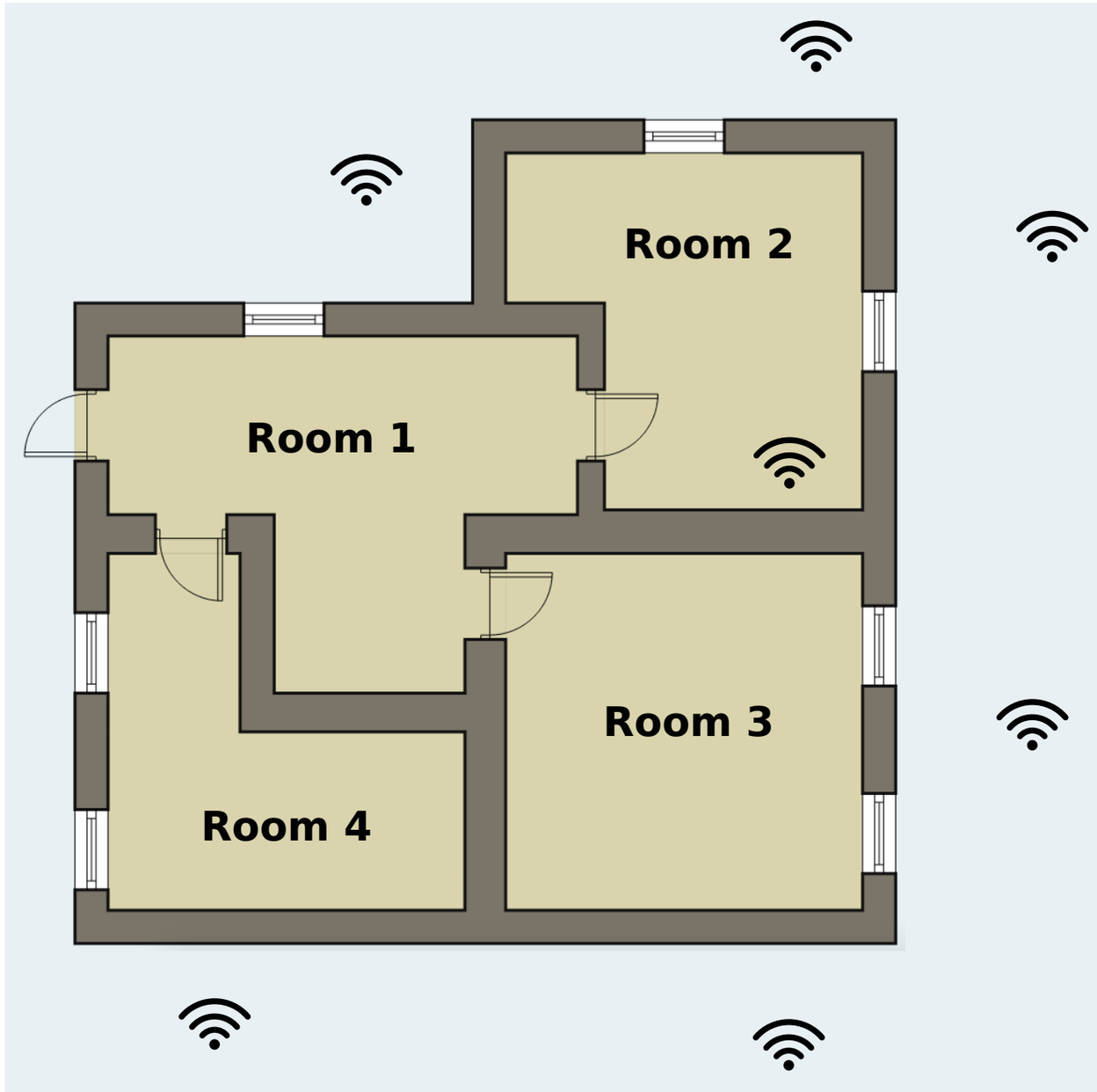


Exam Grade
67%

CBC accounts for 30% of the final grade for the Machine Learning Course.
In other words,
*final grade = 0.67*exam_grade + 0.33*CBC_grade*

Assignment 1:
Decision trees for localisation

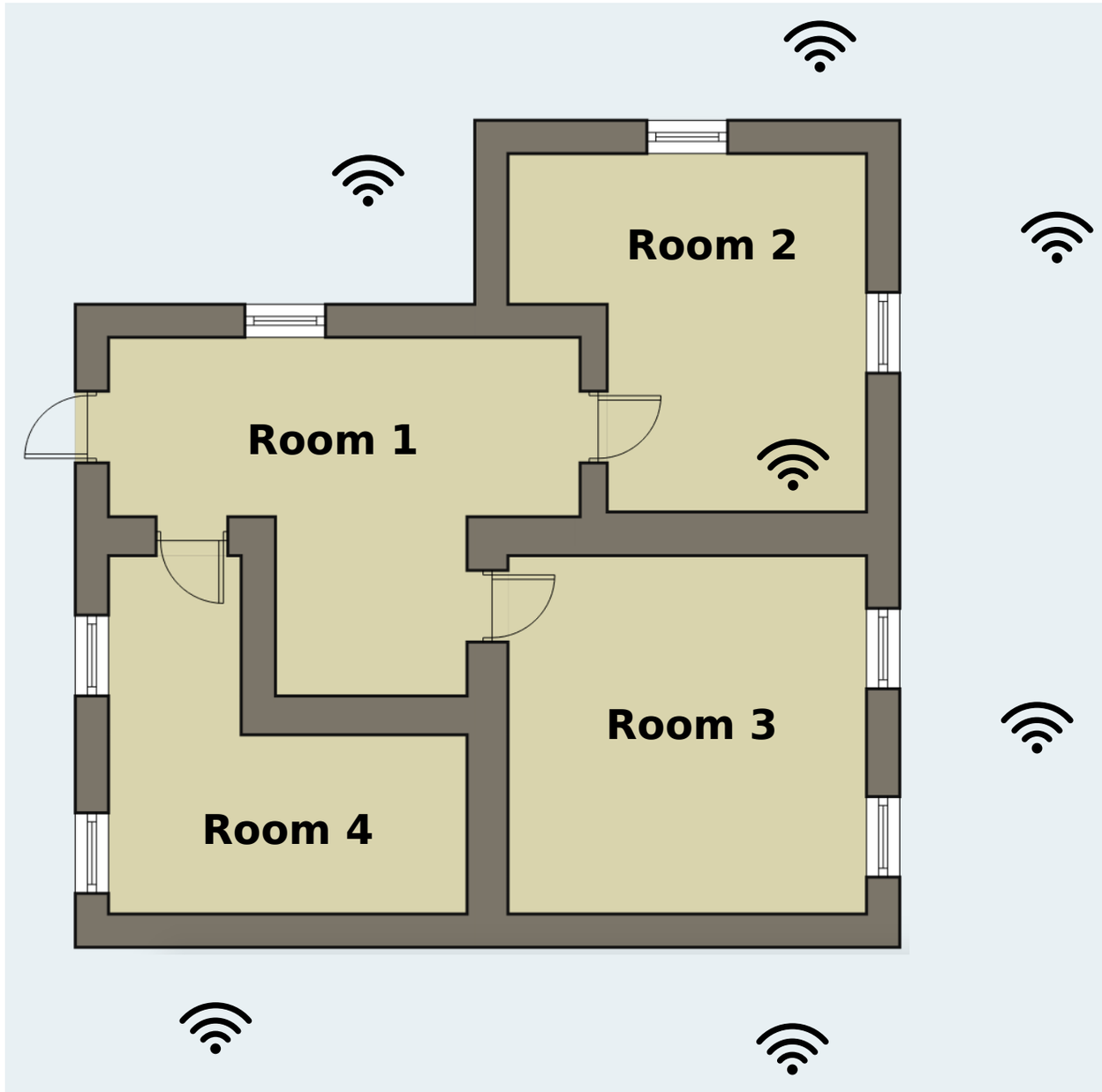
Question: Can you use the wifi signal strength to localise users in their flat?



Assignment 1:
Decision trees for localisation

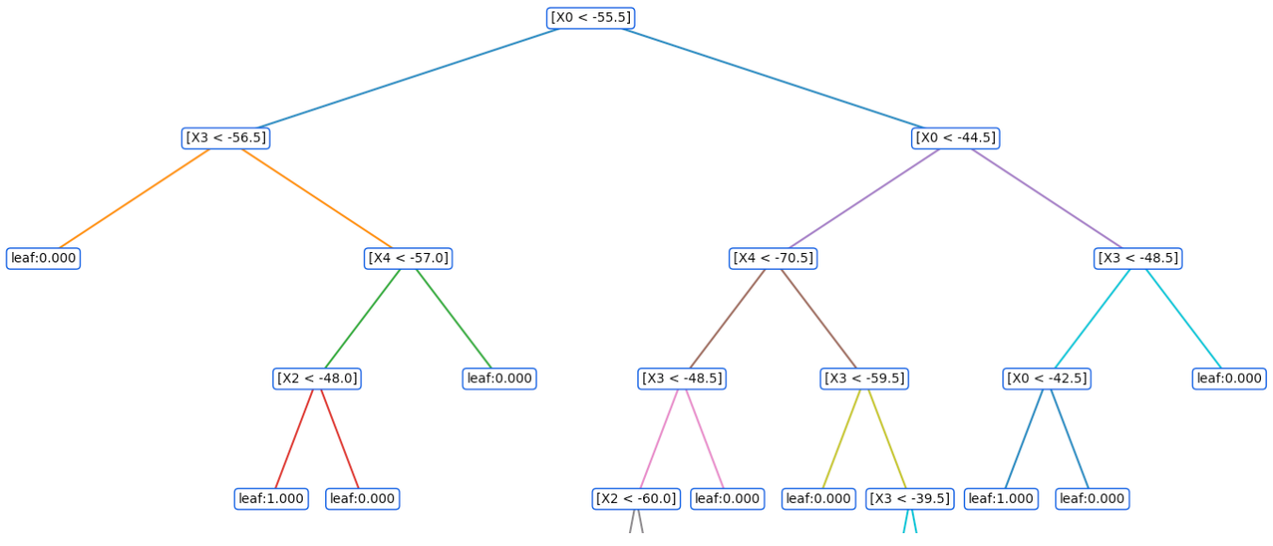
You will be given a dataset of this form:

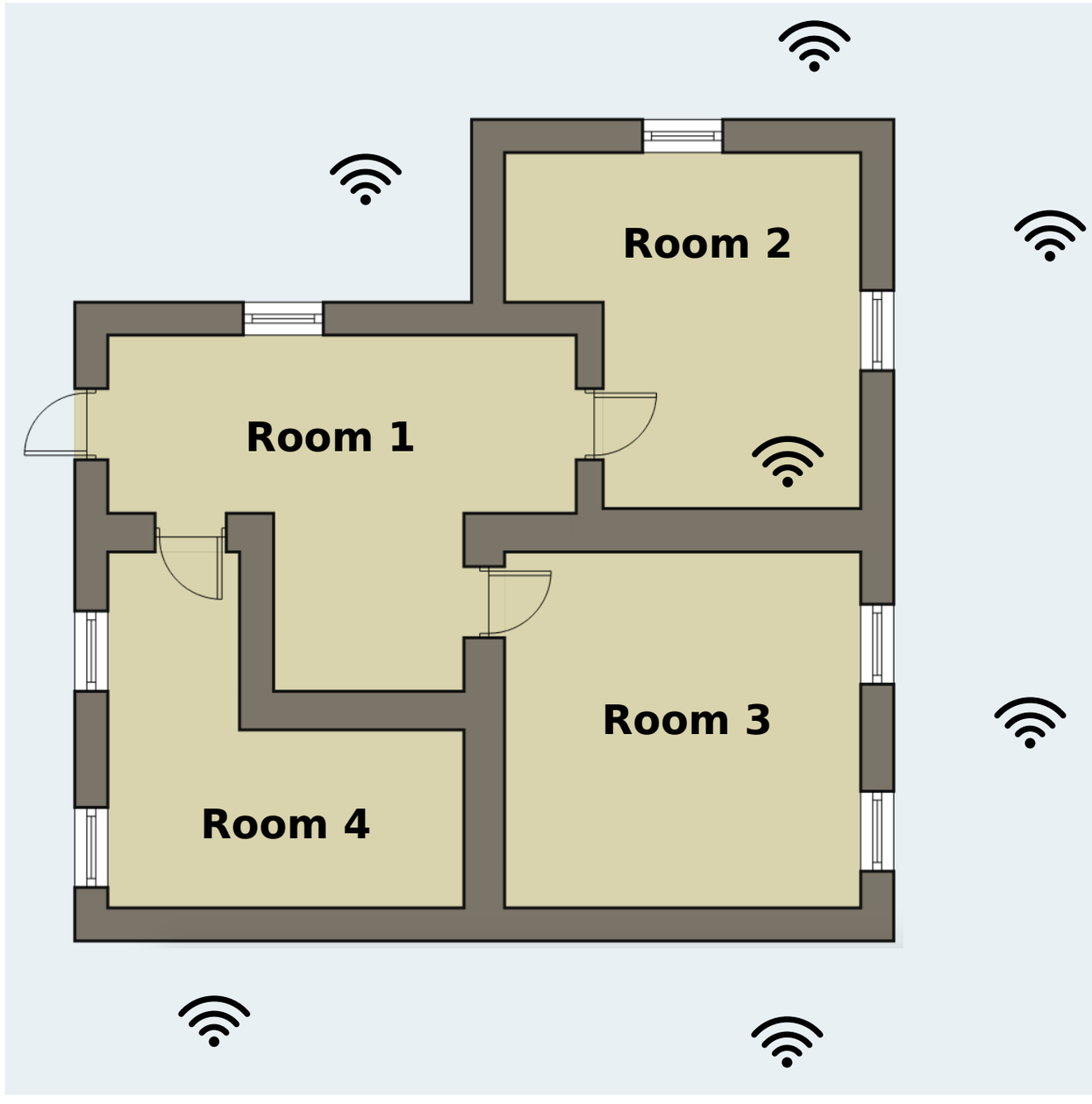
WIFI 1	WIFI 2	WIFI 3	WIFI 4	WIFI 5	WIFI 6	WIFI 7	Label
-68	-57	-61	-65	-71	-85	-85	1
-63	-60	-60	-67	-76	-85	-84	1
-61	-60	-68	-62	-77	-90	-80	2
-63	-65	-60	-63	-77	-81	-87	2
-64	-55	-63	-66	-76	-88	-83	3
-65	-61	-65	-67	-69	-87	-84	4
-61	-63	-58	-66	-74	-87	-82	1
-65	-60	-59	-63	-76	-86	-82	3
-62	-60	-66	-68	-80	-86	-91	3
-67	-61	-62	-67	-77	-83	-91	2
-65	-59	-61	-67	-72	-86	-81	4
-63	-57	-61	-65	-73	-84	-84	4



Assignment 1:
Decision trees for localisation

The goal is to train a decision tree to predict the label, given the 7 wifi signal strengths.

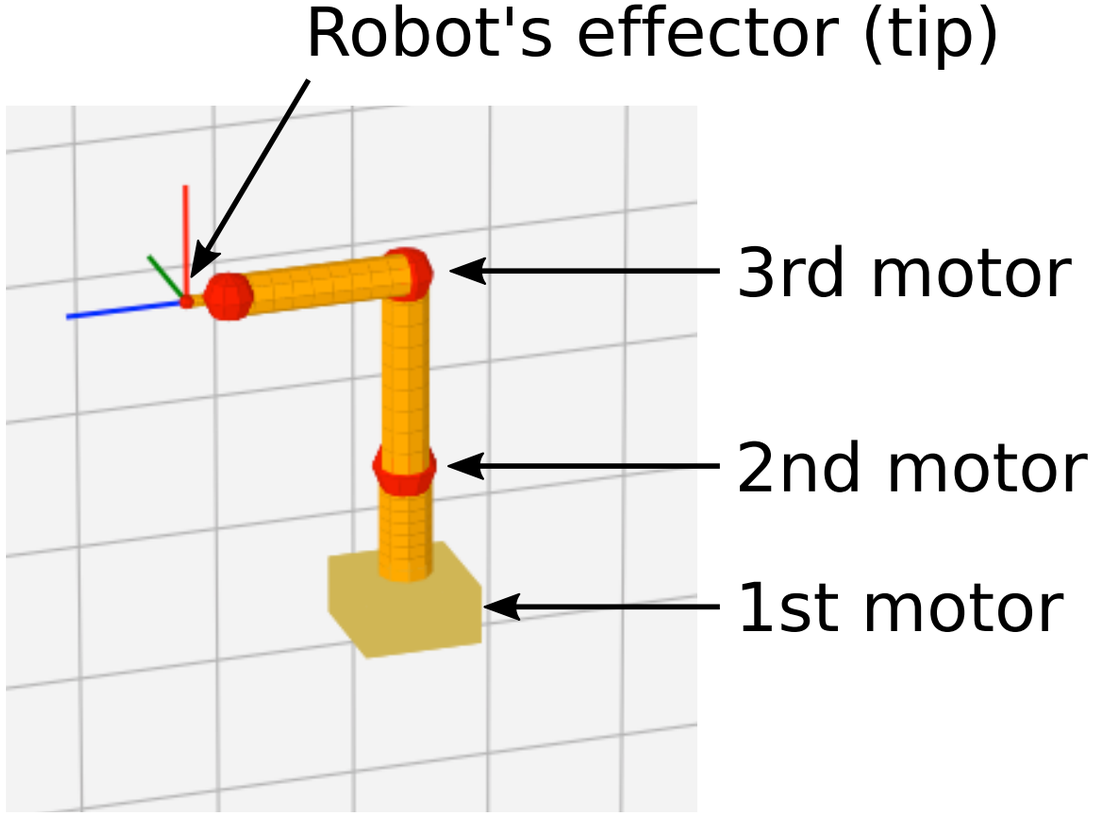




Assignment 2:
Neural Networks

Question: Can you use a neural network to predict the position of the robot's end effector, given the positions of the 3 motors?

Real ABB IRB 120



Simulated ABB IRB 120

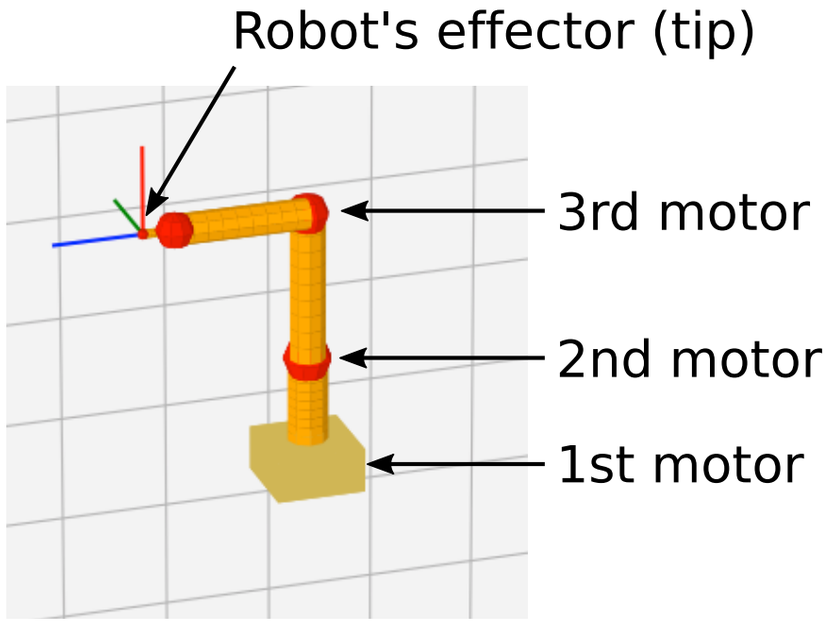
Assignment 2: Neural Networks

The goal is to train two neural networks:

One to predict the position of the robot's end effector (regression task)

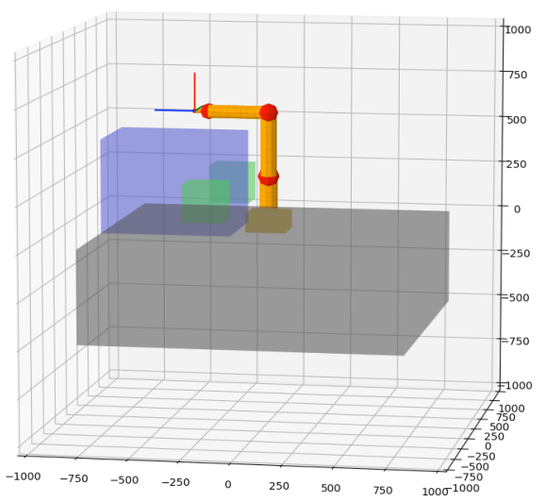
One to predict the zone that the end effector will reach (classification task)

Real ABB IRB 120

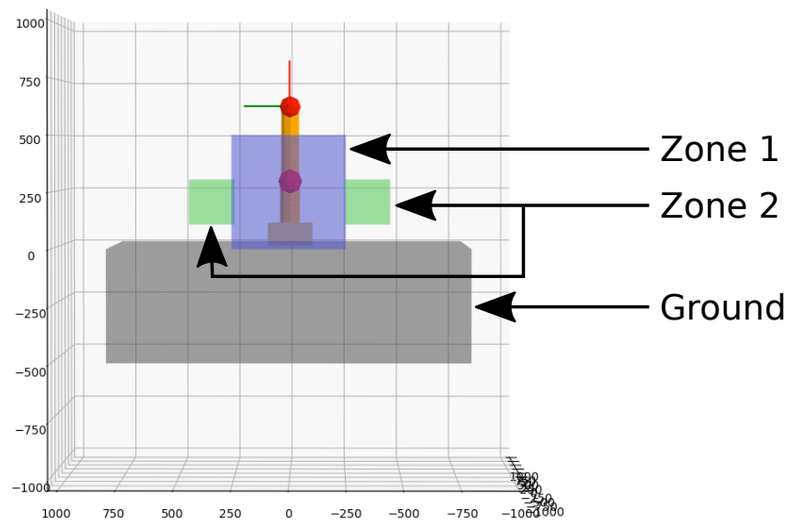


Simulated ABB IRB 120

Side view



Front view



Tips

- Work together!!! Divide the work, but you should all know how you have implemented your algorithms, why you took some specific decision, etc...

- Make sure your code runs!!

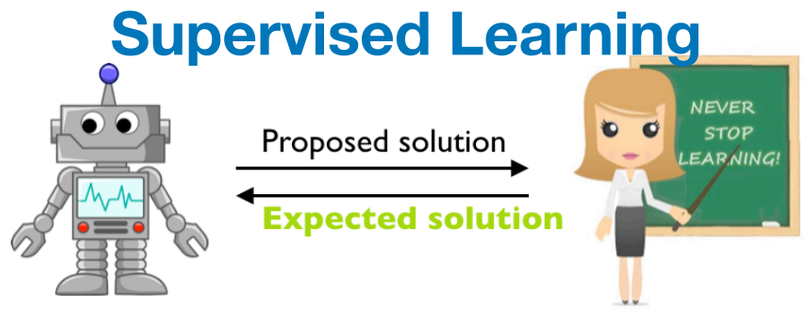
- **Make sure you use the new version of the manual**

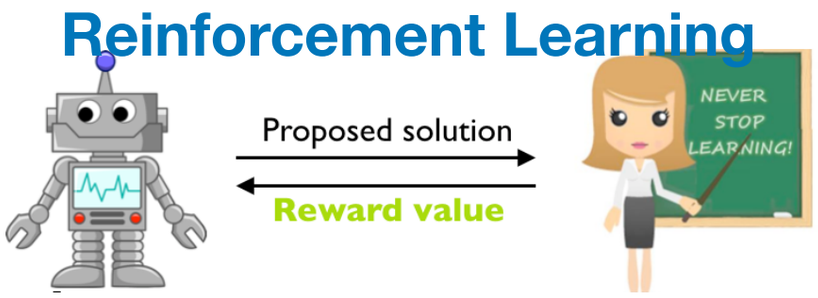
- If you have any complaints (marking, THs etc) contact me.

- Start early to finish early.

Sum-up from last week

Machine Learning approaches



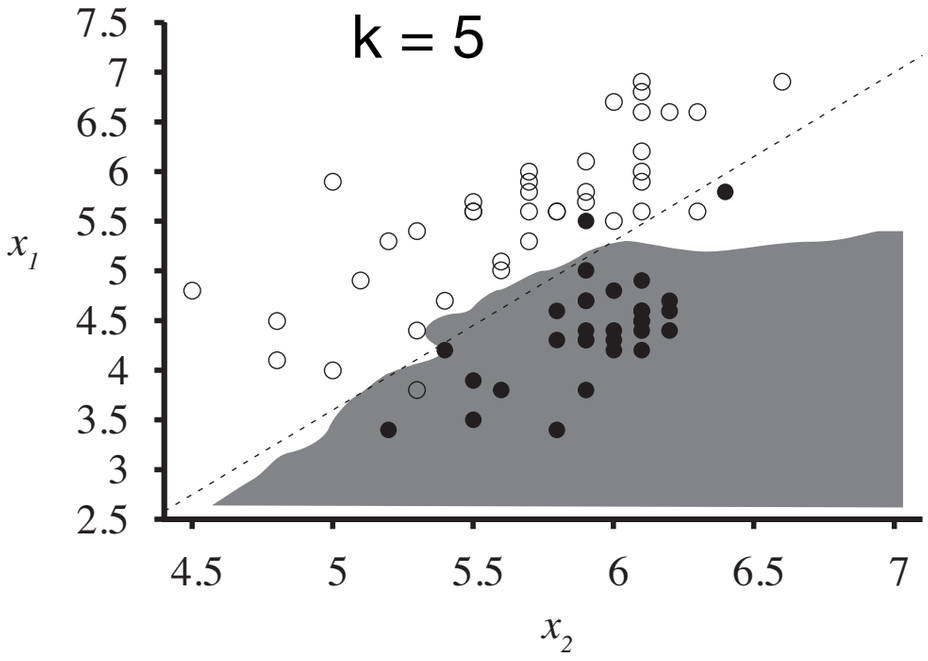


Machine Learning problems

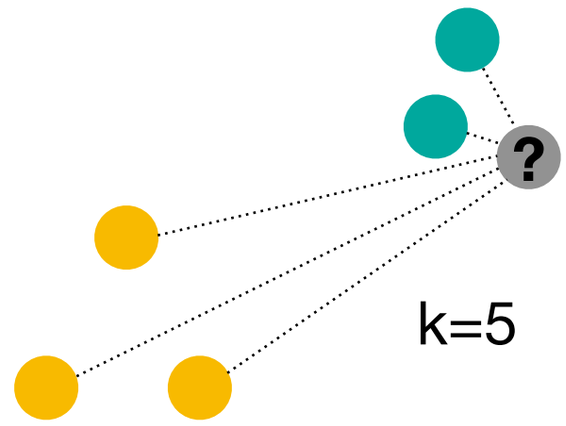
- Classification
- Regression
- Clustering
- Dimensionality reduction
- Density Estimation
- Policy Search

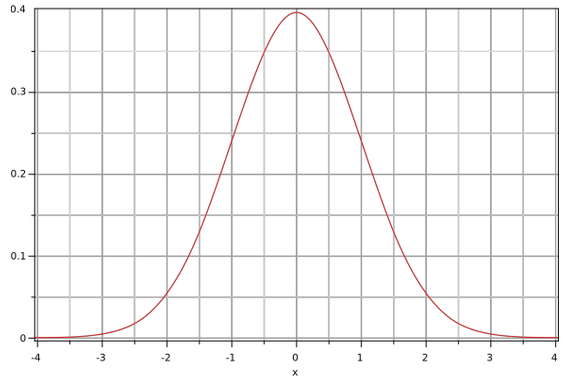
Sum-up from last week

k-nearest neighbours

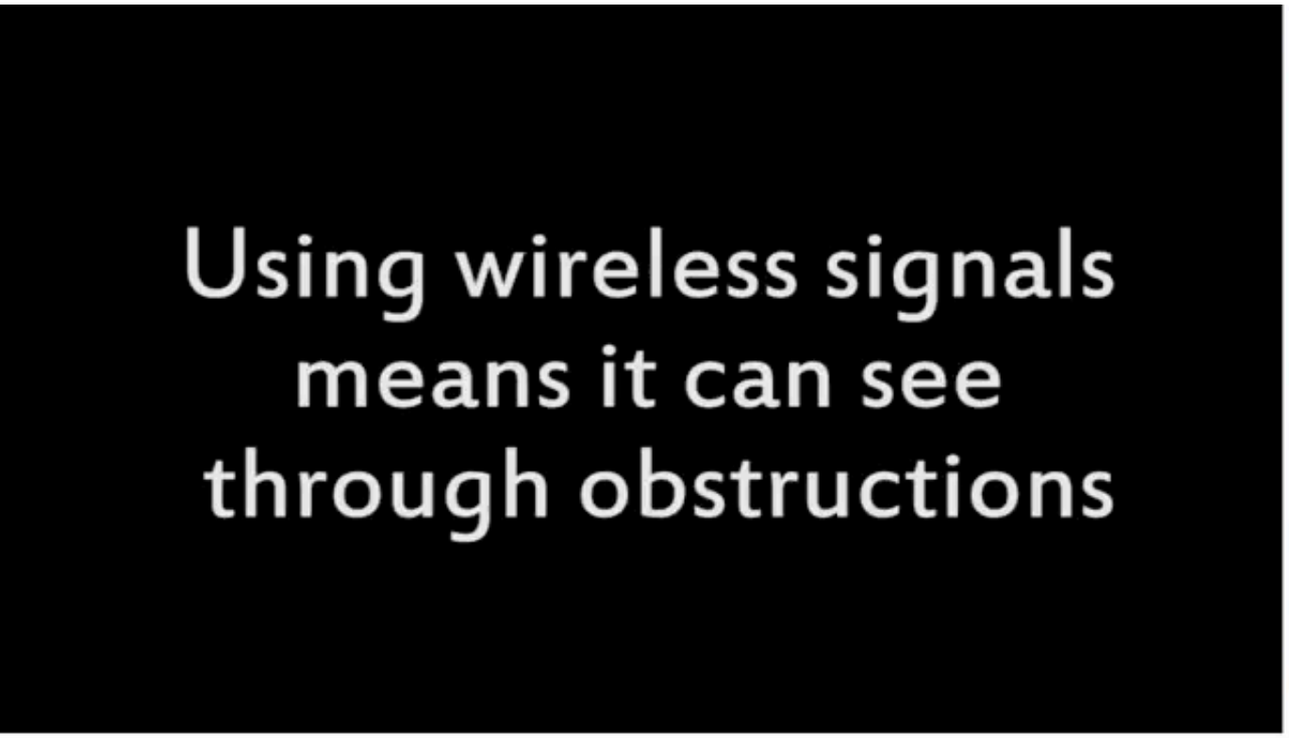


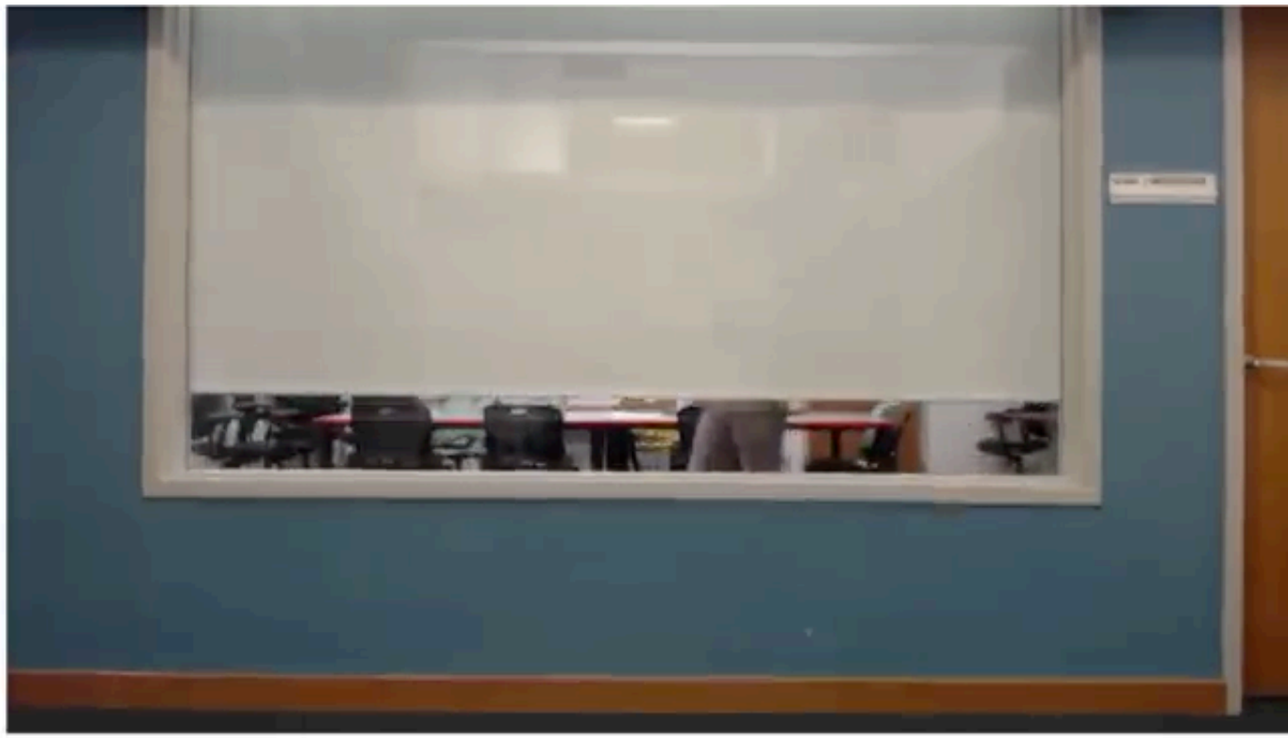
Distance weighted *k*-nearest neighbours

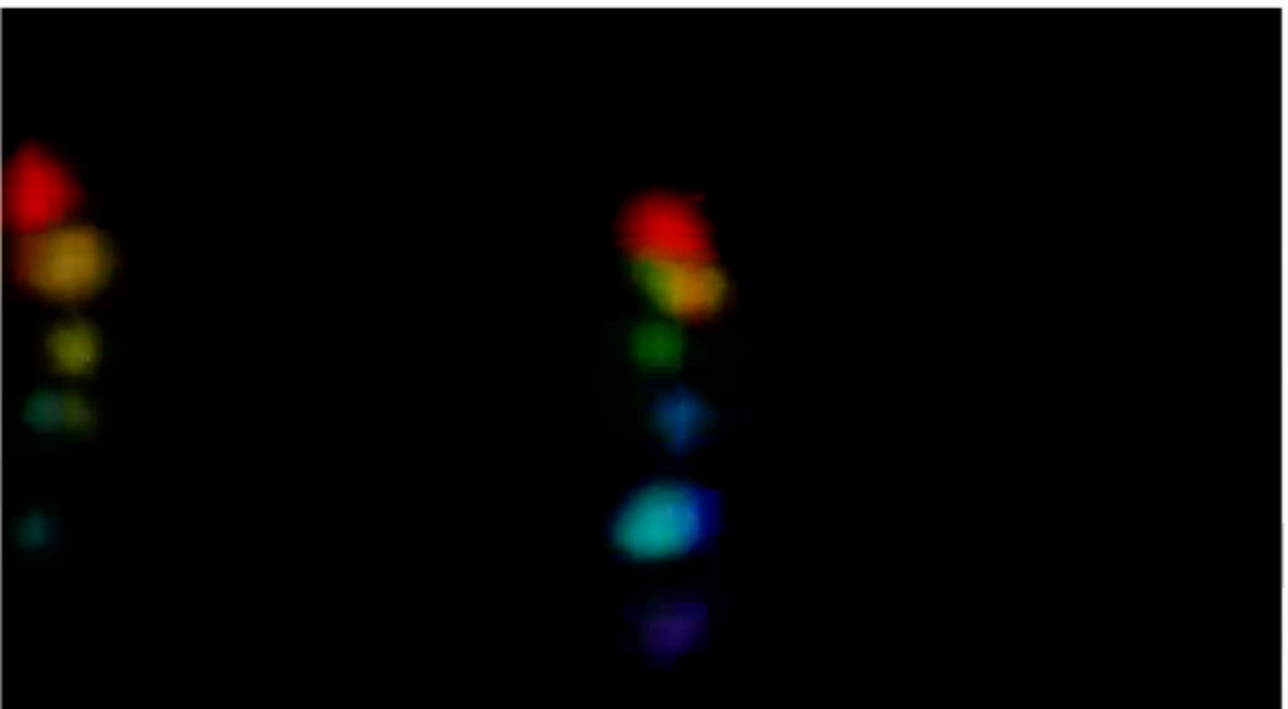


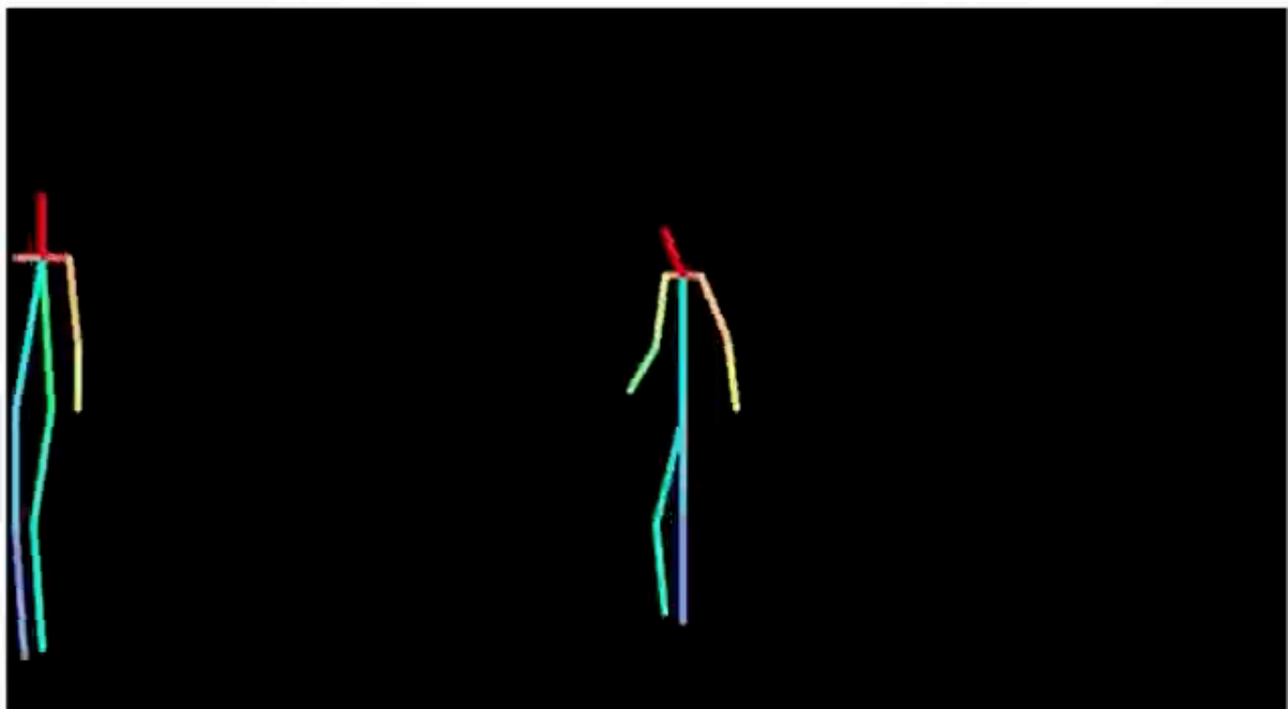


Some exercises.







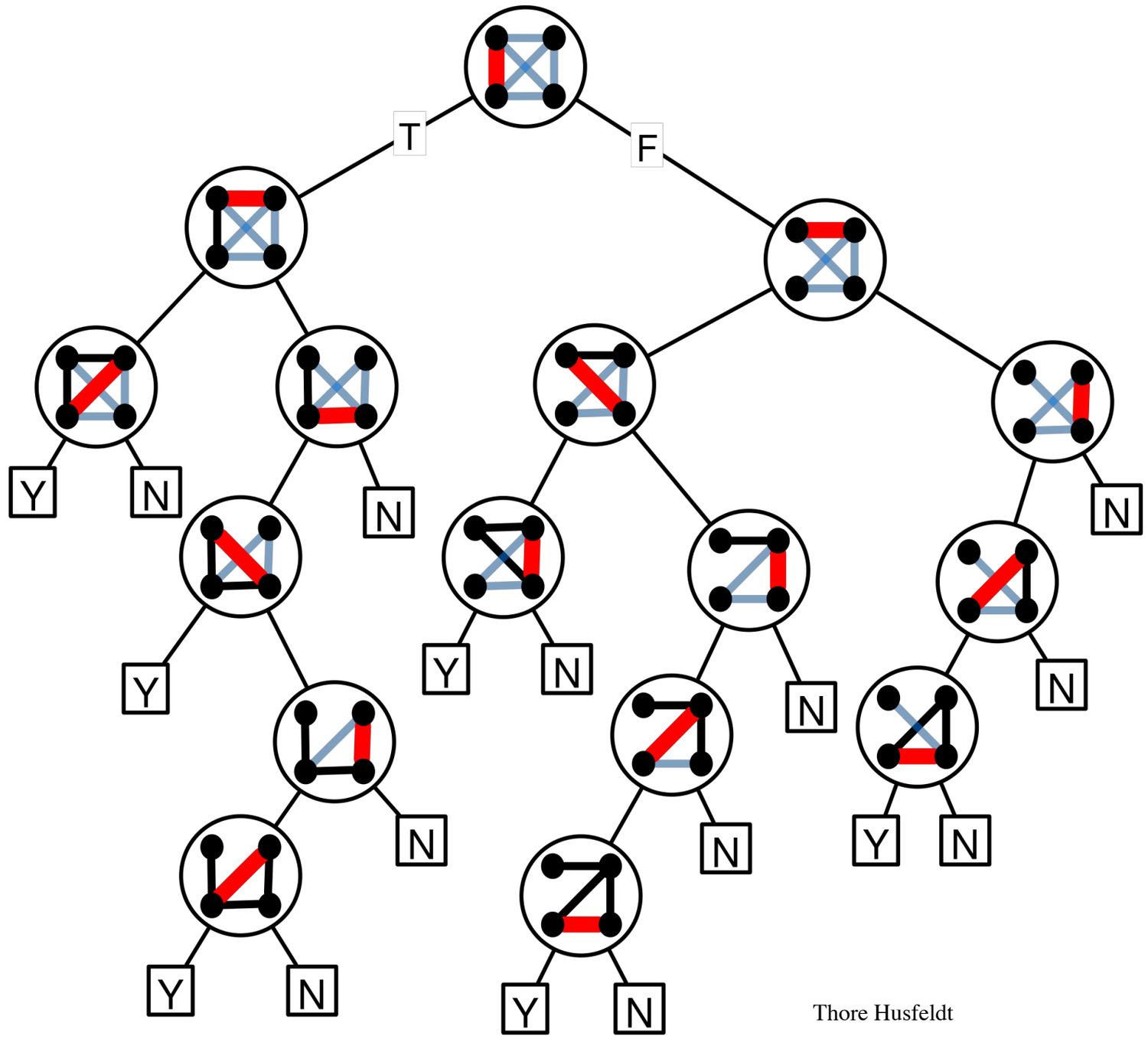


Some exercises.

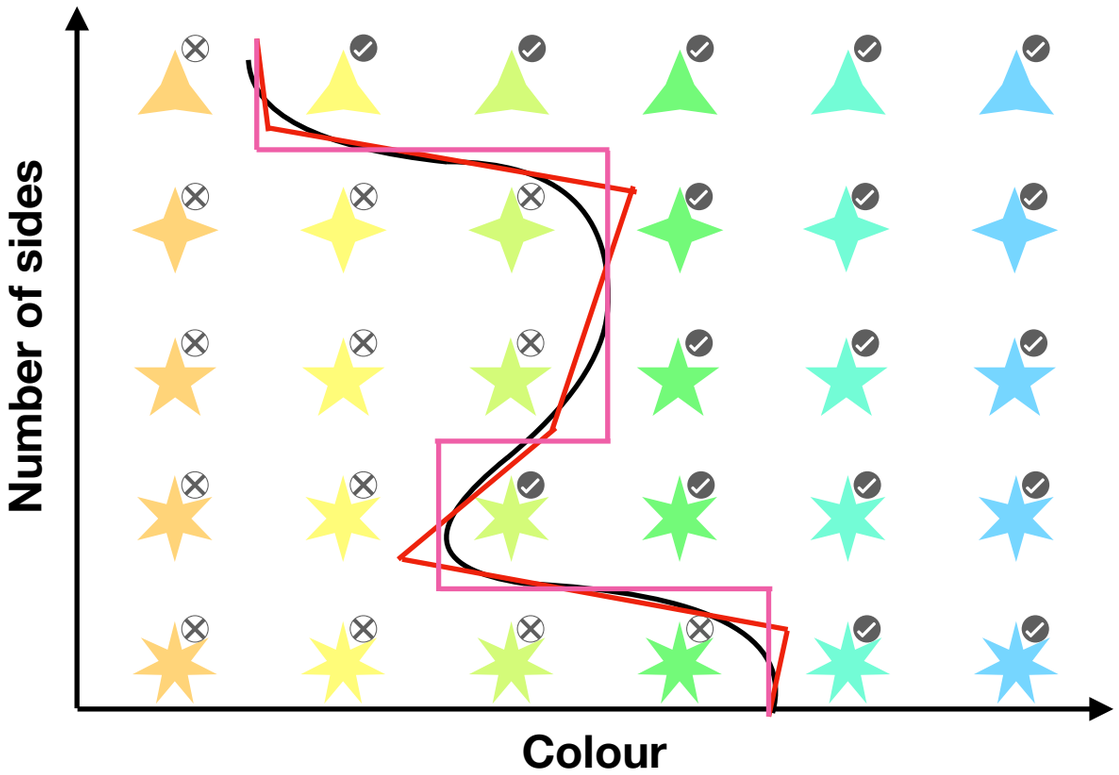
Anomalie detection

Some exercises.

Decision Trees

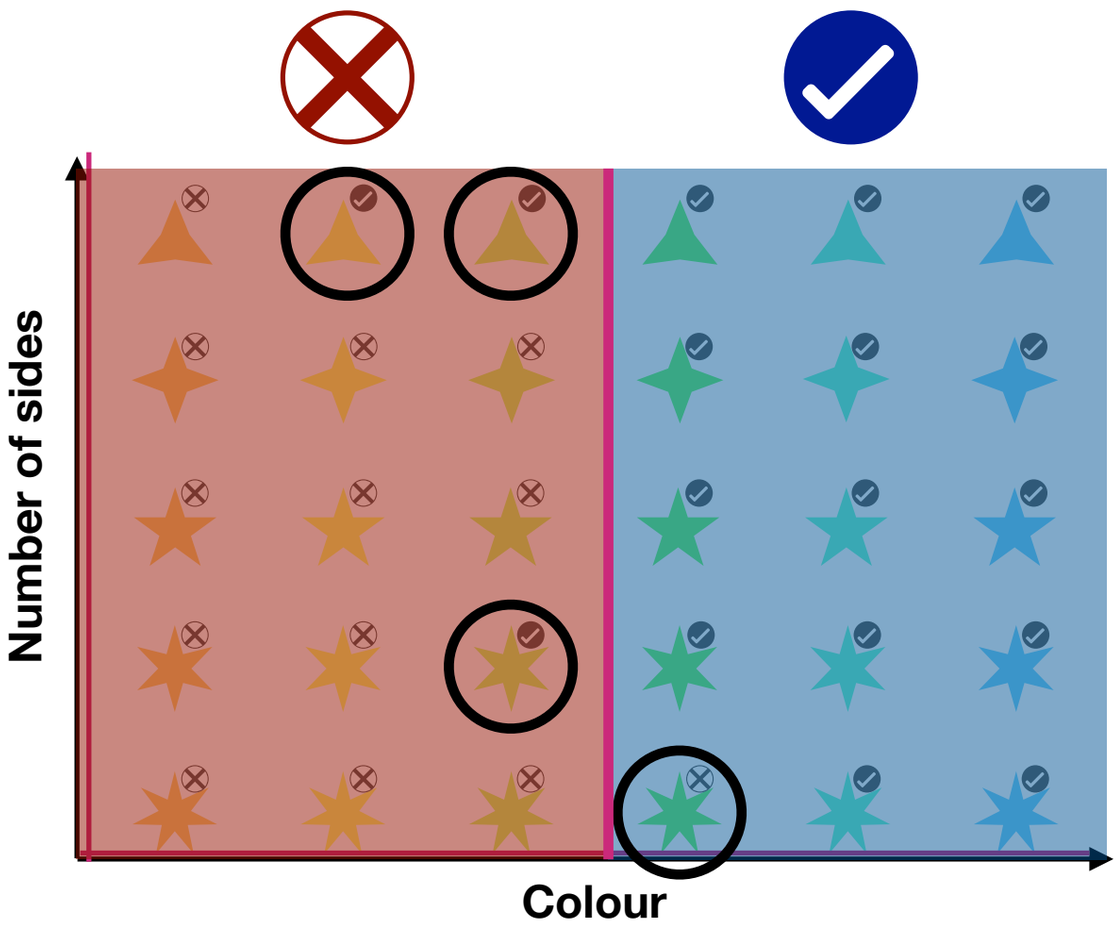


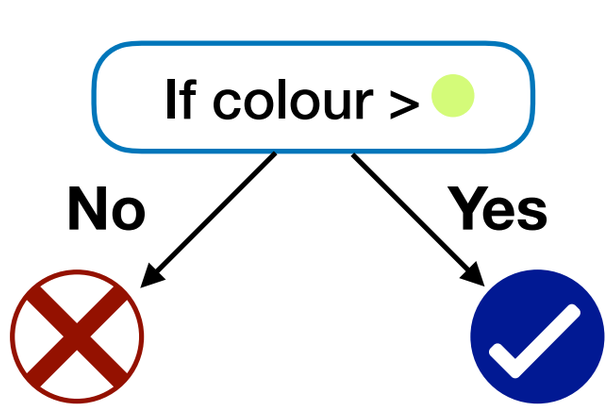
From last week…

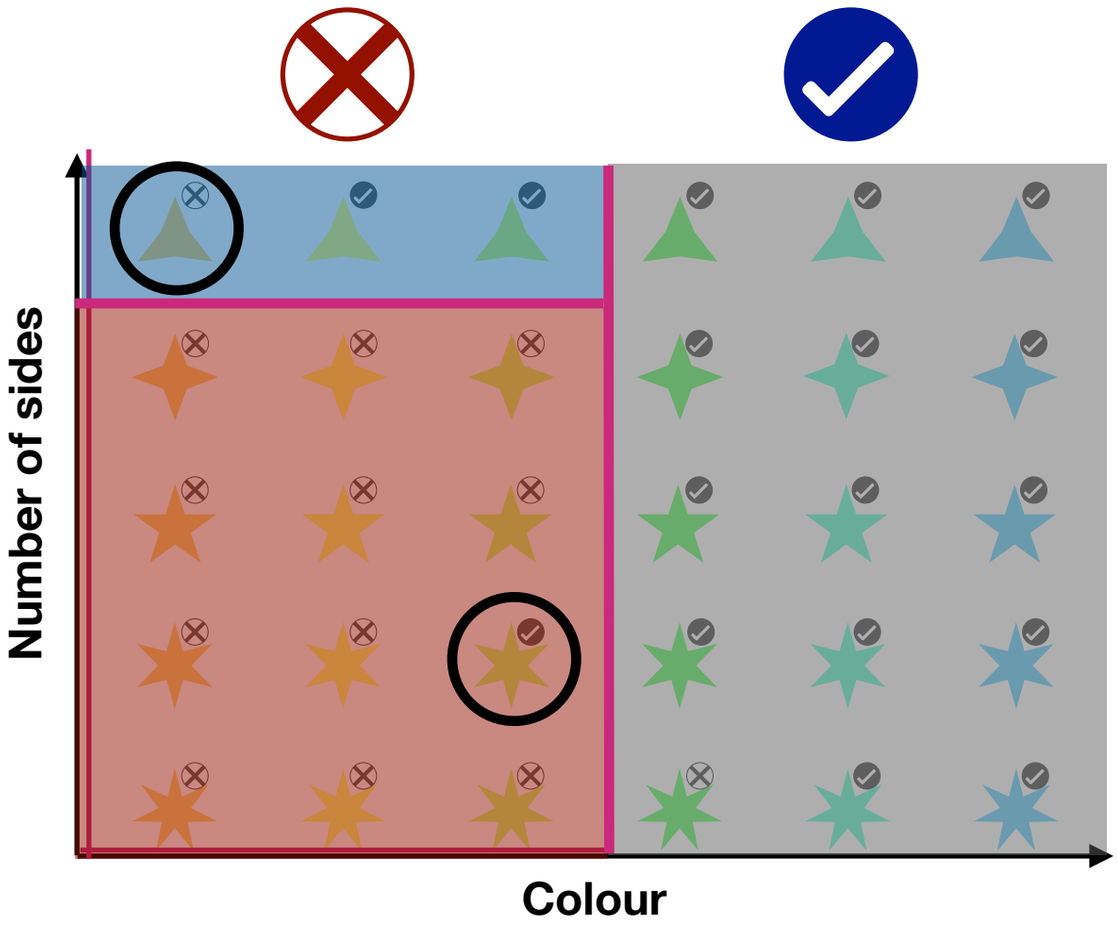


- We saw how Lazy learner (k-NN) can solve this.
- We also saw we can construct a global model of the solution (eager learner)
- In particular, we can consider linear model aligned with the features
- One approach to build such a model is to build a decision tree.

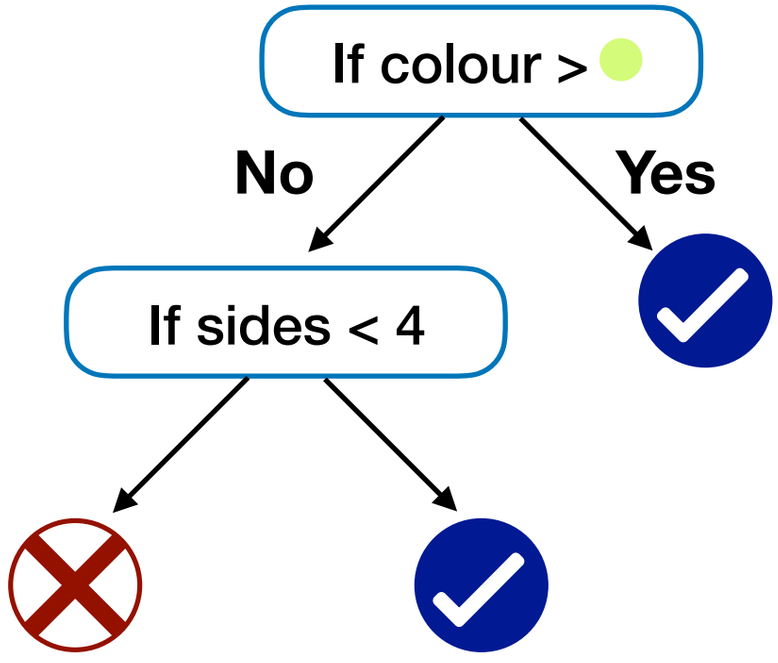
- We need to find the first split point that discriminates the most the dataset

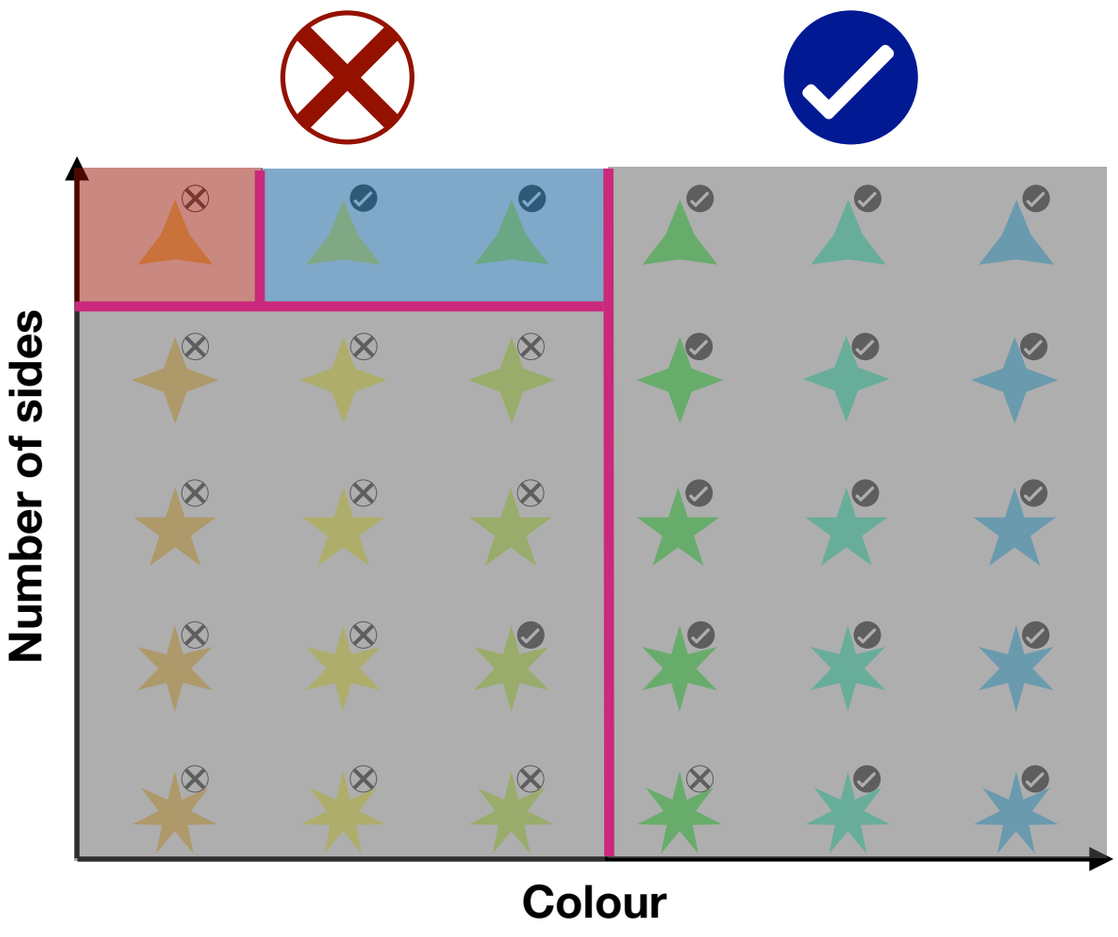




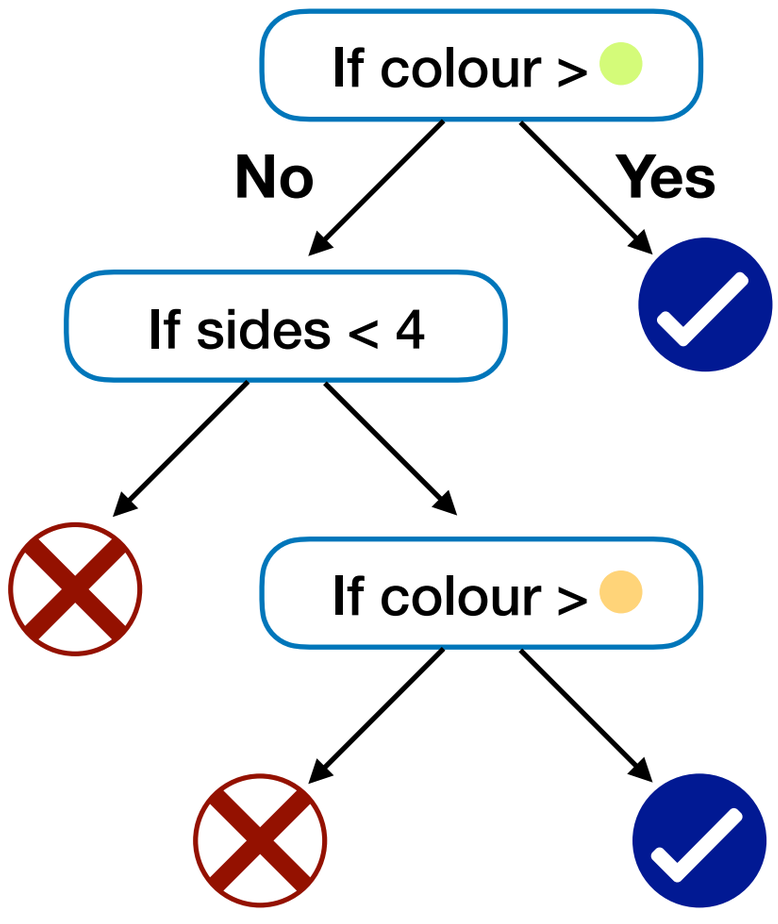


- We repeat the same approach on the two sets created by the split point.

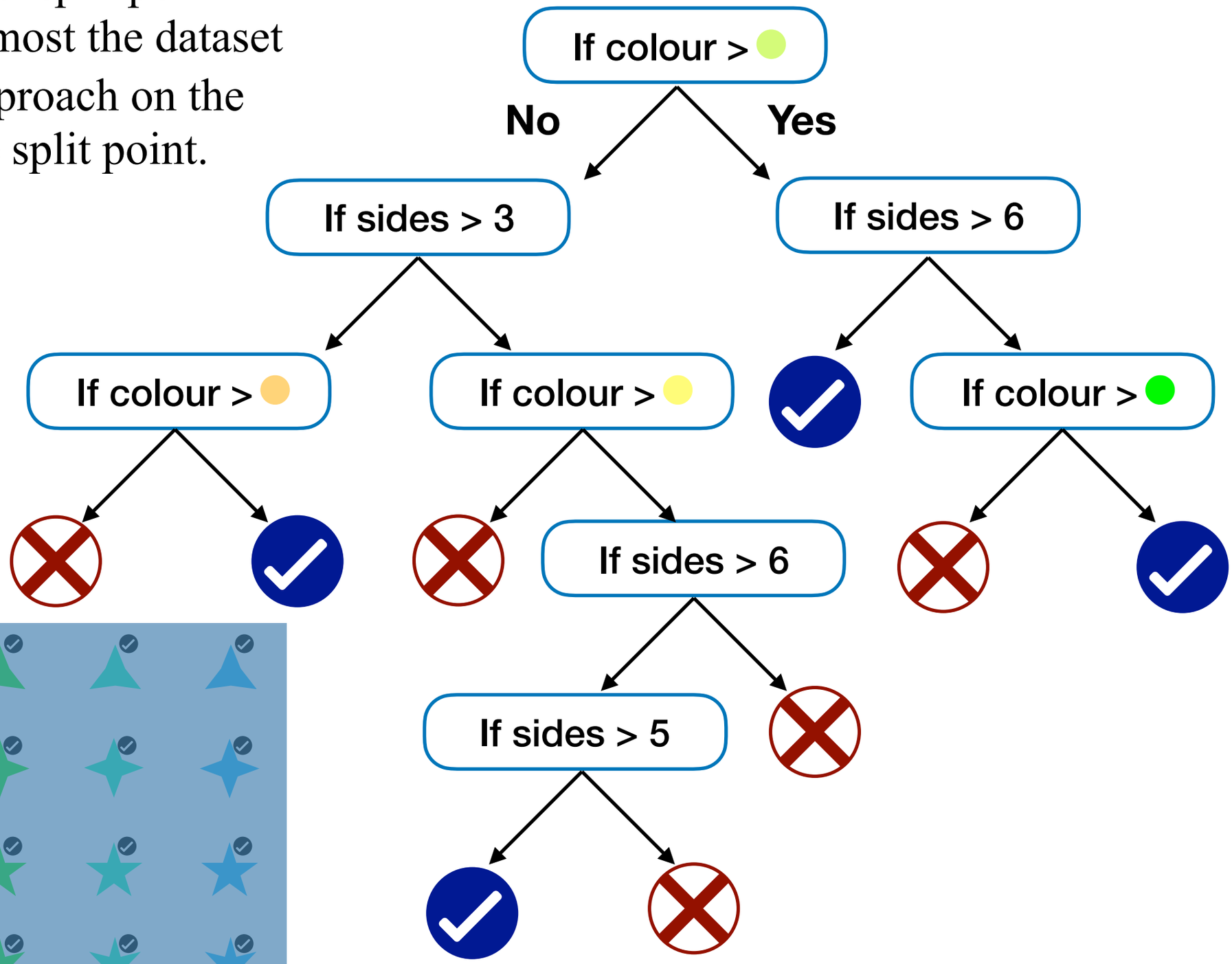


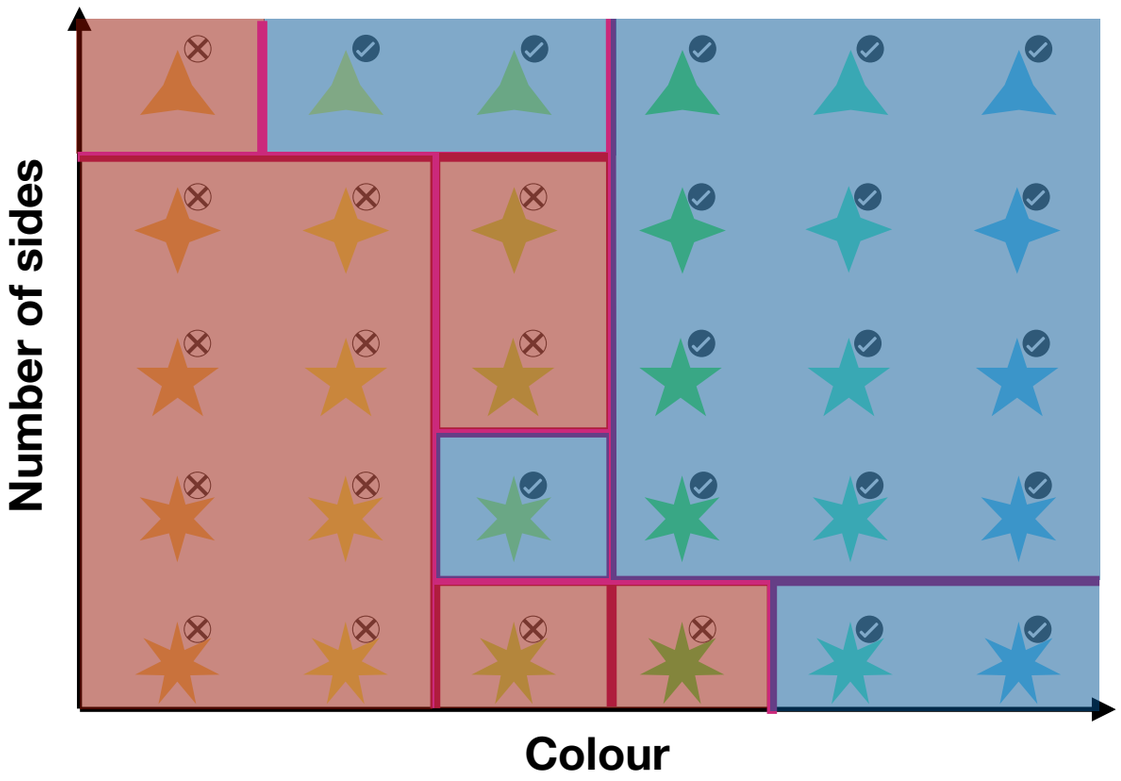


- We repeat the same approach on the two sets created by the split point.



- We need to find the first split point that discriminates the most the dataset
- We repeat the same approach on the two sets created by the split point.





Decision Trees: remarks

- Decision Tree learning is a method for approximating discrete classification functions by means of a tree-based representation
  A learned Decision Tree can be represented as a set of *if-then* rules

Algorithm: Decision Tree

1. Search for a split point (or attribute) using a statistical test of each attribute to determine how well it classifies the training examples when considered alone;
2. Split your dataset according to your split point (or attribute);
3. Repeat 1. and 2. on each of the created subsets.

- Decision Tree learning algorithms employ top-down greedy search through the space of possible solutions.
- ID3 algorithm is one of the most commonly used Decision Tree learning algorithms and it applies this general approach to learning the decision tree.

Selection of the best question

- Several statistical tests exist for this:

- **Information gain:** Quantifies the reduction of the information entropy.

Going further

- **GINI impurity:** Probability of a randomly chosen element from the set to be incorrectly labeled if randomly labeled according to the distribution of labels in the subset.

- **Variance reduction:** Mainly used for regression trees when the target variable is continuous

Information Entropy

Shannon (1948)

- Entropy is a measure of the uncertainty of a random variable.

- It can also be seen as the averaged quantity of information required to fully define a random state (or variable).

- Entropy is defined in combinatorial spaces, discrete probabilities and continuous probabilities.

$$H(V) = -\sum_{k} P(v_k)\log_2(P(v_k))$$

Information Entropy

When all the states are equally possible.

50% **50%**

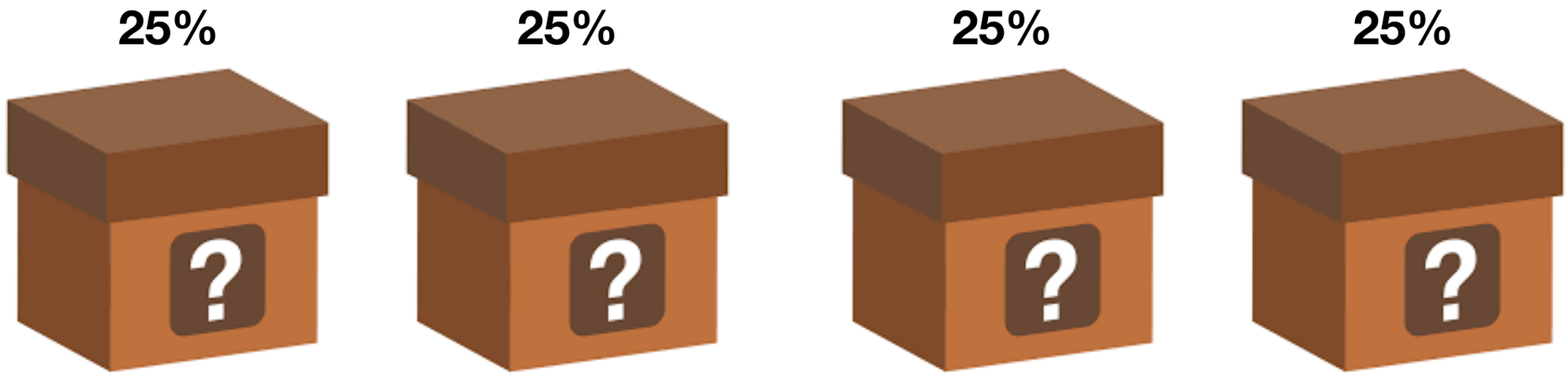
- I stored my keys in one of these two boxes

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

  - Just one bit: is it on the left? 0 or 1?

Information Entropy

When all the states are equally possible.



- With 2^n states, we need n bits.

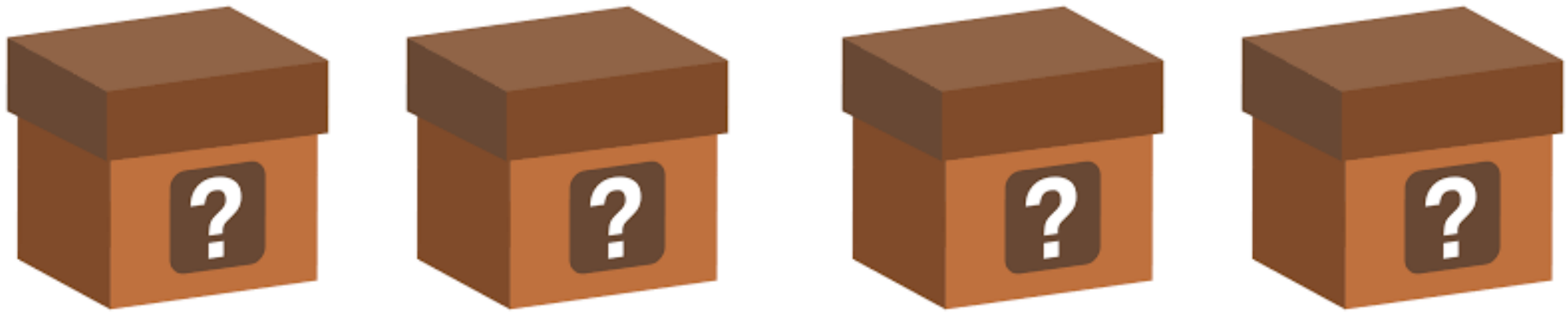
- Therefore, the quantity of information required to fully determine the state of a random variable is $H(V)=\log_2(N)$ where N is the number of possible states.

- From a probabilistic point of view: $P(b)=1/N$
  $H(V) = \log_2(1/P(b)) = - \log_2(P(b))$
  $H(V) = - \log_2(0.25) = 2$ bits

Information Entropy

When all the state are **not** equally possible.

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



- You are almost certain that it is in box 1, so telling you this does not provide a lot information (small entropy).

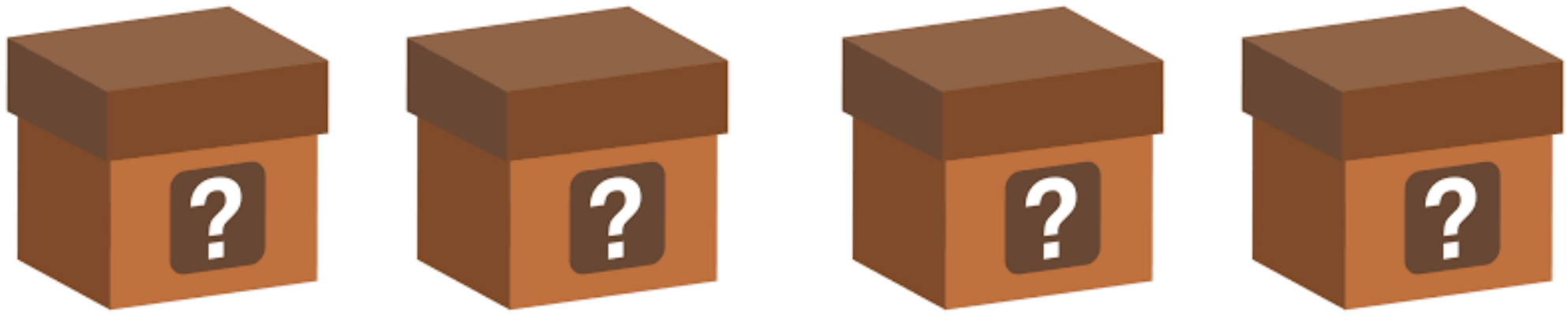
- However, telling you that it's in one of the other boxes represents a very important (or surprising) information.

Box1 = $-\log_2(0.97) = 0.0439$ bits
Box2 = $-\log_2(0.01) = 6.6439$ bits
Box3 = $-\log_2(0.01) = 6.6439$ bits
Box4 = $-\log_2(0.01) = 6.6439$ bits

Information Entropy

When all the state are **not** equally possible.

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



- However, 97% of the time, you will not require a lot of information.

- The Entropy is thus defined as the average quantity of information

$$H(V) = -\sum_{k} P(v_k)\log_2(P(v_k))$$

$H(V) = - 0.99*\log_2(0.97) - 0.99*\log_2(0.01) - 0.01*\log_2(0.01) - 0.01*\log_2(0.01) = 0.2419$ bits

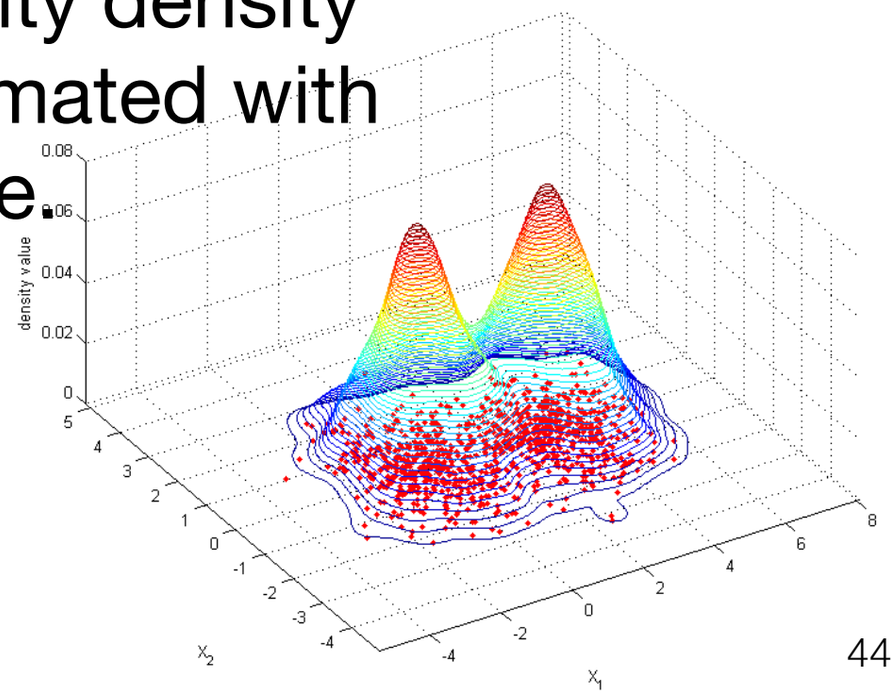
Continuous Entropy

- For a probability density function f(x), we can define the continuous entropy, as an analogy of Shanon's definition

$$H(V) = -\int_x f(x)\log_2(f(x))$$

- This analogy is imperfect (it can have negative values) but still often used in Deep Learning.

- Usually in machine learning, the probability density function is unknown, but can be approximated with density estimation algorithms for instance.



Back to decision trees

- The most informative question maximises the information gain. Therefore, the difference between the initial entropy and the (weighted) average entropy of the produced sub-sets.

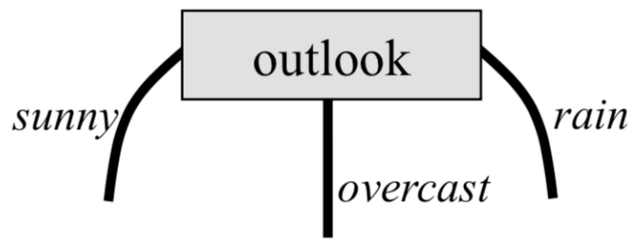
$$G(q) = H(\textbf{dataset}) - \left(\frac{|\textbf{subsetA}|}{|\textbf{dataset}|} H(\textbf{subsetA}) + \frac{|\textbf{subsetB}|}{|\textbf{dataset}|} H(\textbf{subsetB}) \right)$$

- $|\textbf{dataset}| = |\textbf{subsetA}| + |\textbf{subsetB}|$

Different types of inputs

- **Ordered values** (like real-values):
  find a split point
  For each feature: start by sorting the values of the attribute, and then consider only split points that are between two examples in sorted order that have different classifications.

- **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

Algorithm

Algorithm: Decision Tree

1. Search for a split point (or attribute) using a statistical test of each attribute to determine how well it classifies the training examples when considered alone;
2. Split your dataset according to your split point (or attribute);
3. Repeat 1. and 2. on each of the created subsets.

- The statistical test is often the Information Gain.
- This is the case with the ID3 algorithm.

Example

	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 (Mitchell's book, p. 59)

Example

	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

- It is a problem with symbolic values.
- We compute the information gain of each feature independently
  (outlook, temperature, humidity, wind).
- H(dataset)=9/14*$\log_2$(9/14) + 5/14*$\log_2$(5/14)
  =0.940
- H(Dsunny)=2/5log2(2/5)+3/5*log2(3/5)
  =0.971
- H(Dovercast)=4/4log2(2/4)+0/4*log2(0/4)
  =0
- H(Drain)=3/5log2(3/5)+2/5*log2(2/5)
  =0.971

Gain(outlook) = H(dataset) - 5/14*H(Dsunny)— 4/14*H(Dovercast) - 5/14*H(Drain)
= 0.940 – 0.357 · 0.971 – 0 – 0.357 · 0.971 = 0.246

Example

	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

Gain(outlook) = H(dataset) - 5/14*H(Dsunny)— 4/14*H(Dovercast) - 5/14*H(Drain)

$= 0.940 – 0.357 \cdot 0.971 – 0 – 0.357 \cdot 0.971 = 0.246$

Gain(temp.) = H(dataset) - 4/14*H(Dhot)— 6/14*H(Dmild) - 4/14*H(Dcool)

$= 0.940 – 0.286 \cdot 1 – 0.429 \cdot 0.918 – 0.286 \cdot 0.811 = 0.029$

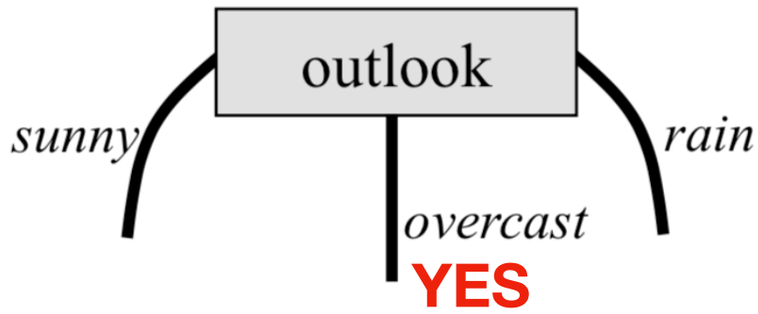
Gain(humidity) = H(dataset) - 7/14*H(Dhigh)— 7/14*H(Dnormal)

$= 0.940 – 7/14 \cdot 0.985 – 7/14 \cdot 0.591 = 0.151$

Gain(wind) = H(dataset) - 8/14*H(Dweak)— 6/14*H(Dstrong)

$= 0.940 – 0.571 \cdot 0.811 – 0.429 \cdot 1 = 0.048$

Example

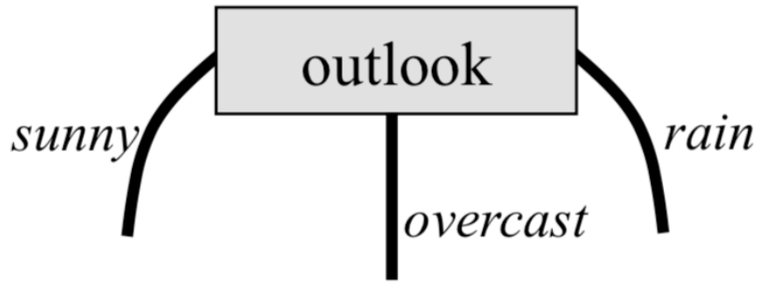


	PlayTennis(d)	*outlook*	*temperature*	*humidity*	*wind*
1	0	sunny	hot	high	weak
2	0	sunny	hot	high	strong
8	0	sunny	mild	high	weak
9	1	sunny	cool	normal	weak
11	1	sunny	mild	normal	strong
4	1	rain	mild	high	weak
5	1	rain	cool	normal	weak
6	0	rain	cool	normal	strong
10	1	rain	mild	normal	weak
14	0	rain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

D1

D2

Example



	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

D1

Gain(temp.) = H(D1) - 2/5*H(D1hot)— 2/5*H(D1mild) - 1/5*H(D1cool)
= 0.971 – 0.4 · 0 – 0.4 · 1 – 0.4 · 0 = 0.571

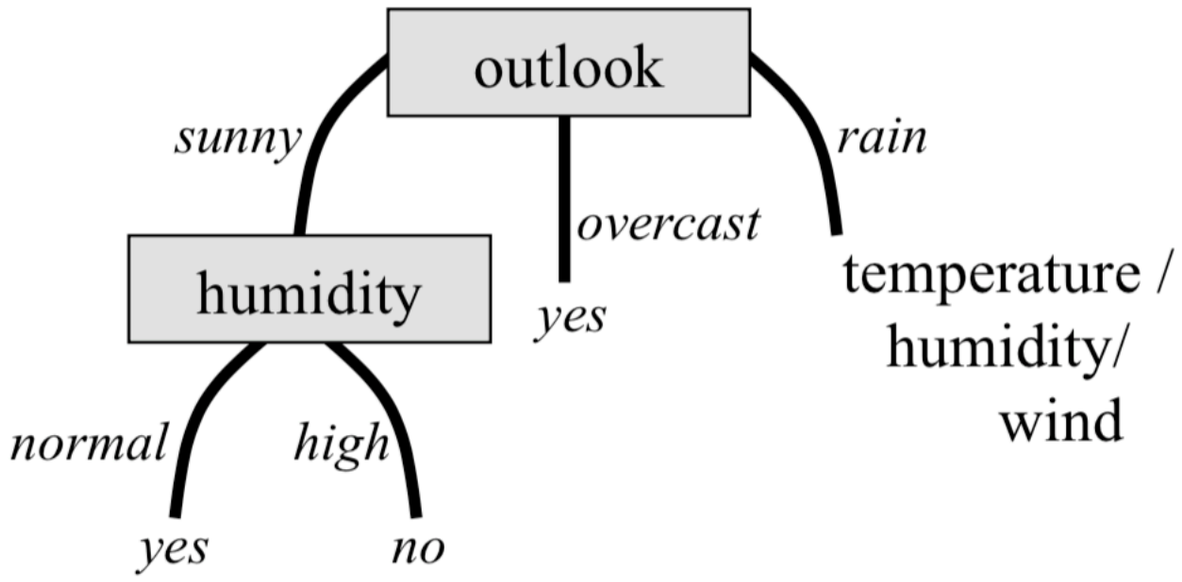
Gain(humidity) = H(D1) - 3/5*H(D1high)— 2/5*H(D1normal)
= 0.971 – 0.6 · 0 – 0.4 · 0 = 0.971

Gain(wind) = H(D1) - 3/5*H(D1weak)— 2/5*H(D1strong)
= 0.971 – 0.6 · 0.918 – 0.4 · 1 = 0.02

Example

	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

D1



Example

	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
…	…	…	…	…	…

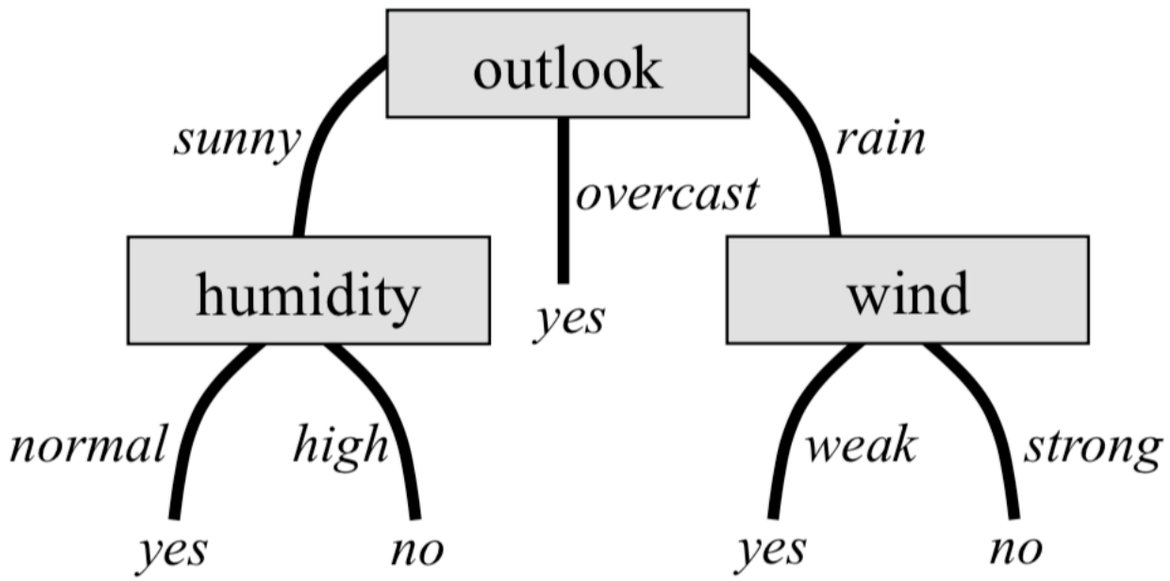
Gain(temp.) = H(D2) - 0/5*H(D2hot)— 3/5*H(D2mild) - 2/5*H(D2cool)
= 0.971 – 0 – 0.6 · 0.918 – 0.4 · 1 = 0.02

Gain(humidity) = H(D2) - 2/5*H(D2high)— 3/5*H(D2normal)
= 0.971 – 0.4 · 1 – 0.6 · 0.918 = 0.02

Gain(wind) = H(D2) - 3/5*H(D2weak)— 2/5*H(D2strong)
= 0.971 – 0.6 · 0 – 0.4 · 0 = 0.971

Example

	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
…	…	…	…	…	…



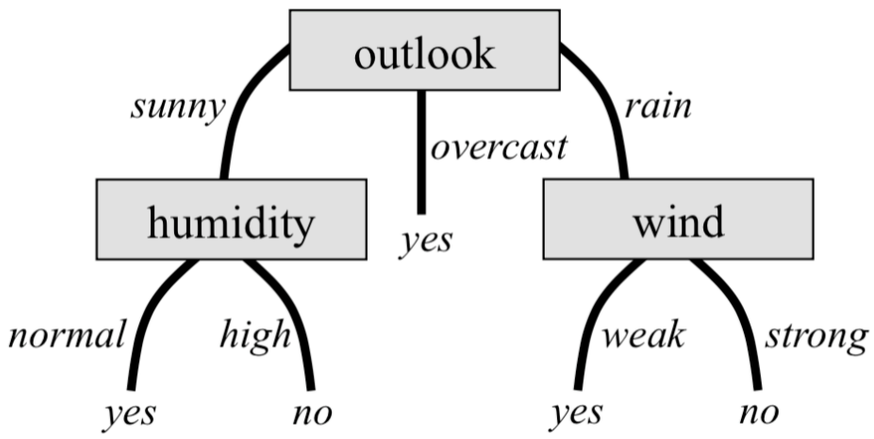
Sum-up

Algorithm: Decision Tree

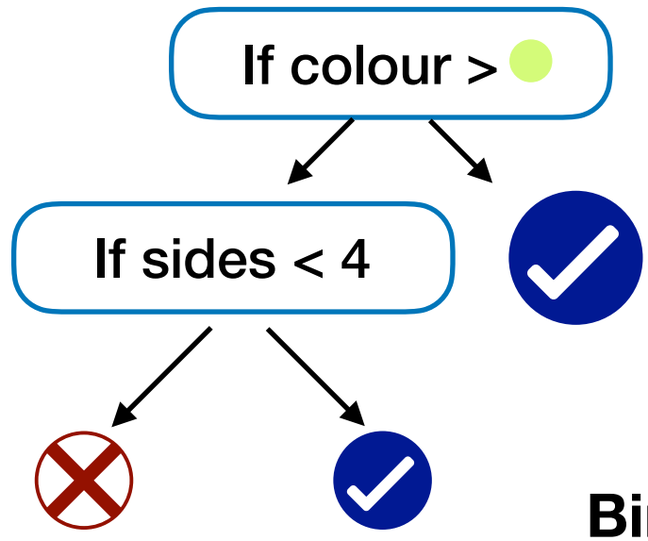
1. Search for a split point (or attribute) using a statistical test of each attribute to determine how well it classifies the training examples when considered alone;
2. Split your dataset according to your split point (or attribute);
3. Repeat 1. and 2. on each of the created subsets.

- Different type of data: Symbolic or real-valued
- Information gain:

$$G(q) = H(\textbf{dataset}) - \left(\frac{|\textbf{subsetA}|}{|\textbf{dataset}|} H(\textbf{subsetA}) + \frac{|\textbf{subsetB}|}{|\textbf{dataset}|} H(\textbf{subsetB}) \right)$$



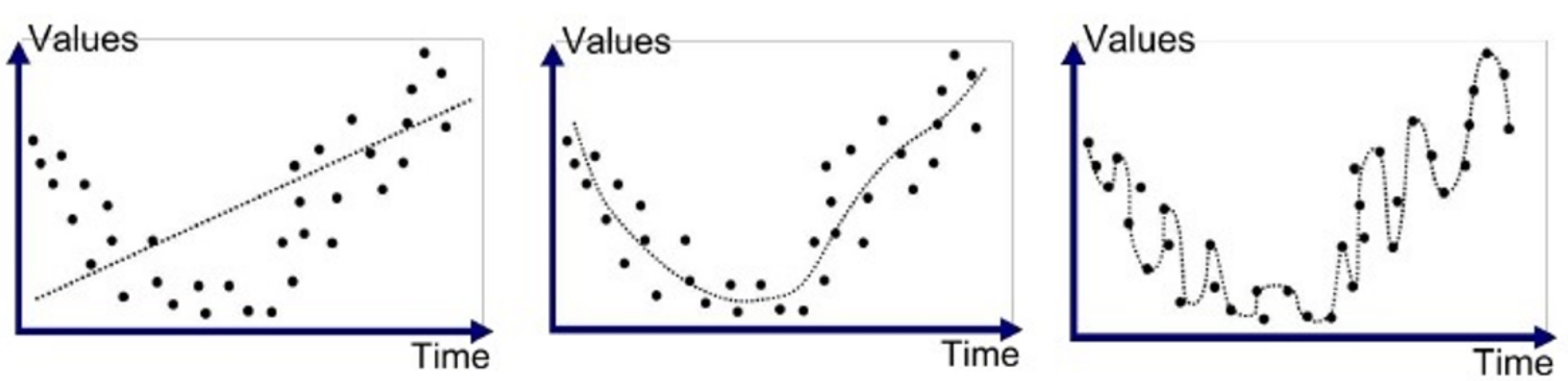
Non-binary tree.



Binary tree.

Decision Tree: Remarks

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



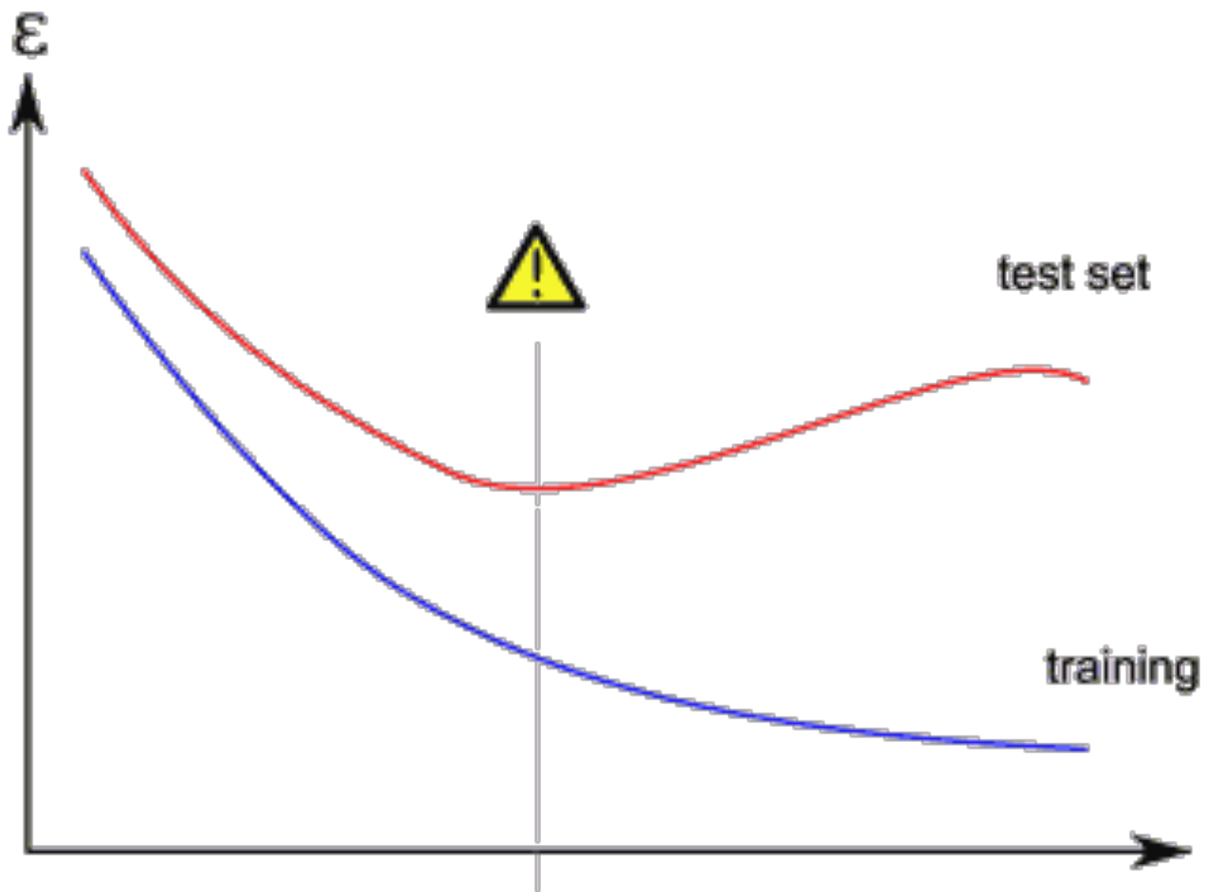
Underfitted | Good Fit/Robust | Overfitted

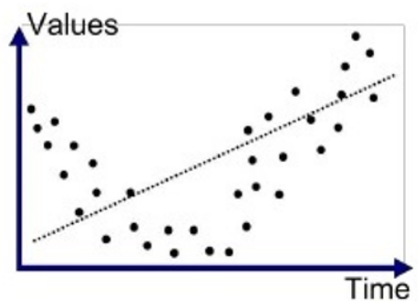
Overfitting

How do deal with overfitting?

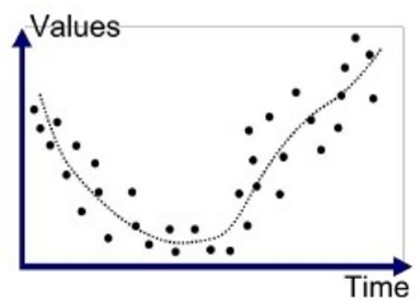
Split your dataset into two parts: **training** dataset and a **validation** dataset.

You can use the training dataset to train your model and the validation one to stop the training when the performance on the validation test degrades.

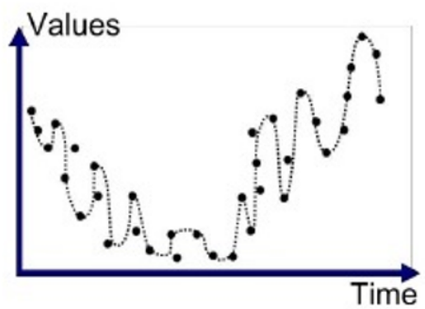




Underfitted



Good Fit/Robust



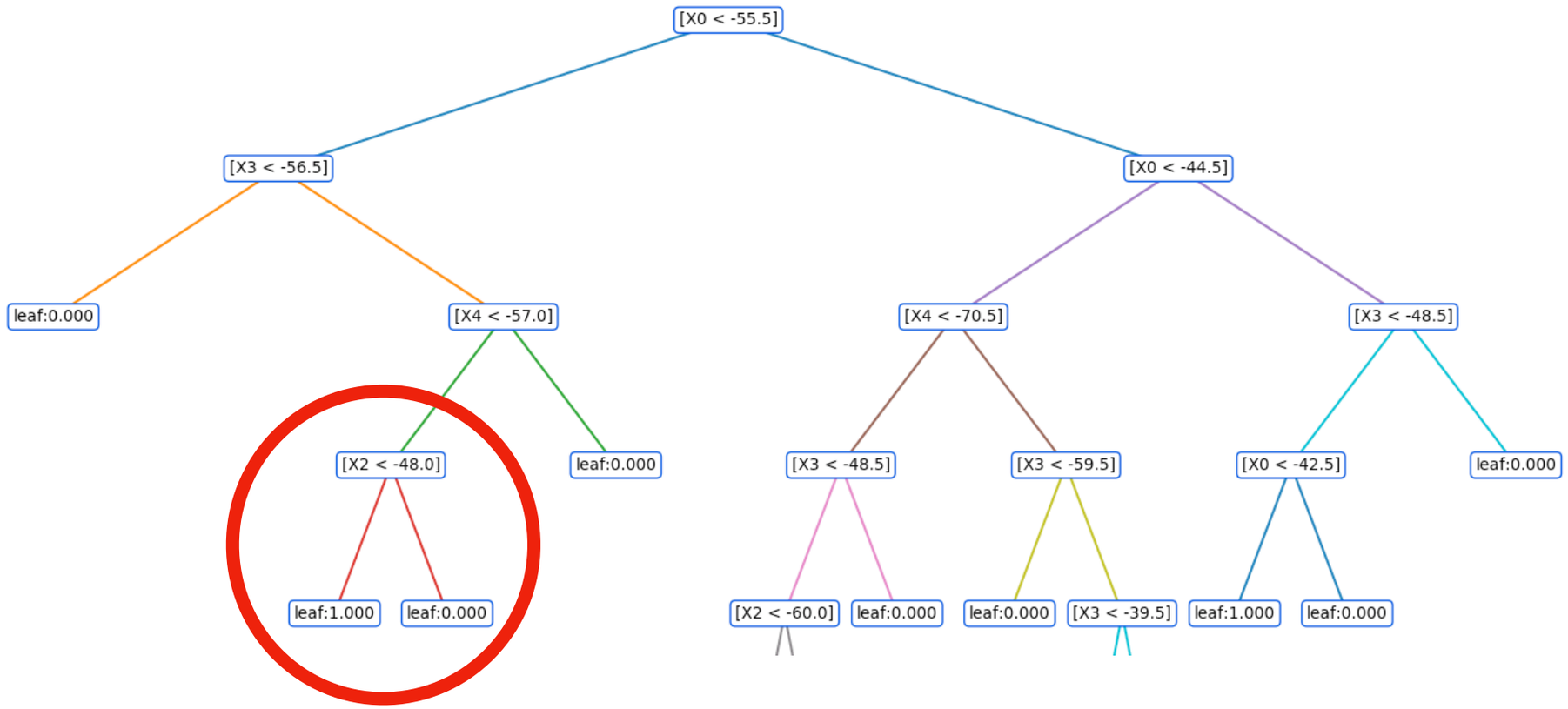
Overfitted

Overfitting

- Common approach with decision trees: **Pruning**.

- Go through all the nodes that are only connected to leaves and check if the accuracy on the validation dataset would increase if this node is turned into a leaf.

- You need to do this recursively as when you turn nodes into leaves, you might create new nodes that are connected to two leaves.

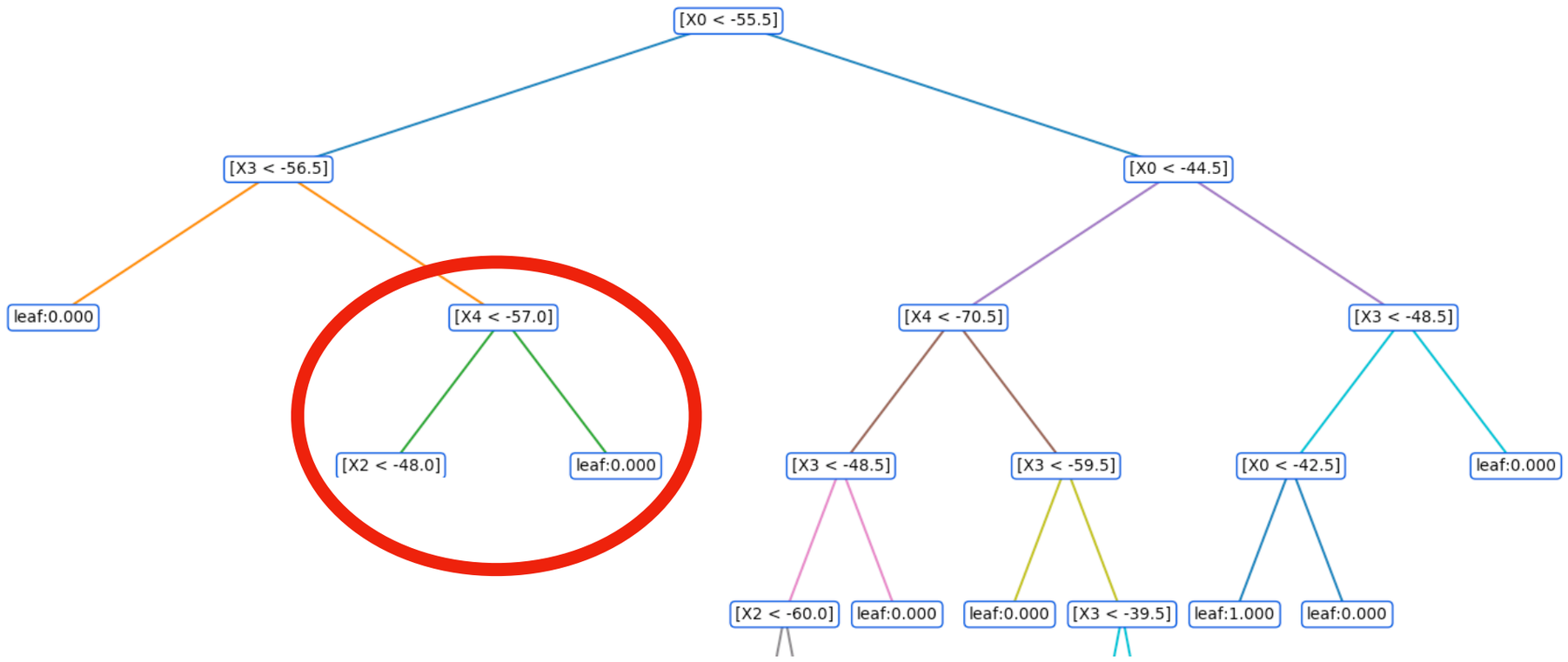


Overfitting

- Common approach with decision trees: **Pruning**.

- Go through all the nodes that are only connected to leaves and check if the accuracy on the validation dataset would increase if this node is turned into a leaf.

- You need to do this recursively as when you turn nodes into leaves, you might create new nodes that are connected to two leaves.



Regression

- Like k-NN, decision trees can be used for classification and regression (but classification is way more common).

- We call it a regression tree in this case.

- Instead of a single class in each leaf, there is a linear function of some subset of the numerical features.

- It requires that the learning algorithm must decide when to stop splitting and begin applying linear regression over the features.

Group Forming

- Students will be divided in **groups of 4**

- Simply fill the online form **https://tinyurl.com/y9u8kegx** with the following information:

	First Name	Last Name	Student login	Email	Course	CID
Student 1						
Student 2						
Student 3						
Student 4						

- Fill the form by **noon Wednesday January 23th**

- You will be assigned a TH who will email you informing you about your group number and confirming the members of your group by the end of the week