sis Café
DAN vs HSMA eat off
Rule 1. Take it in turns to take 1-3 sweets from the bag (HSMA starts)
Whoever is left with the chilli must take a bite of it
What we want to achieve

Module 7 : Machine Learning

Session 7A : Introduction to AI and Machine Learning

Dr Daniel Chalk

"Do HSMAs dream of electric sheep?"

HSMA 4

HSMA 4 Everyone

NIHR | Applied Research Collaboration South West Peninsula

@peninsula_ARC #HSMA4

What is AI?

AI (Artificial Intelligence) refers to the simulation of intelligence (or aspects of intelligence) by machines. Such simulated intelligence may include :

- learning (e.g. to make decisions, such as to classify)
- reasoning (e.g. applying logical rules to come to a conclusion)
- having coherent conversations
- perception (e.g. recognising images, speech or other sounds)

In the HSMA course, we mostly focus on the first of these : learning. Specifically we look at learning to simulate human decision making (this module), and learning of natural language patterns to extract information from written text (Natural Language Processing module).

Machine Learning

Machine Learning (ML) is a field of AI which is focused on developing *algorithms* that enable machines to *learn* by *experience* and by being shown *data*.

Often the purpose of developing ML algorithms is to make a *prediction / classification* given some data. This may be to provide decision support to human decision making by trying to learn inherent and possibly hidden patterns in data.

Machines should rarely replace human decision making, but should be seen as a Decision Support Tool.

Unless you’re building Skynet :)

Pros and Cons

Machine Learning allows us to :

- automate tasks undertaken by humans that would consume time and / or money resources
- try to better understand some of our own decision making, or hidden patterns in data
- allow for data to be used to generate insights

But you also need to be aware that :

- machines can and will get things wrong
- there may be ethical considerations, particularly in terms of getting things wrong (see next session)
- you may end up with more questions than answers

Features

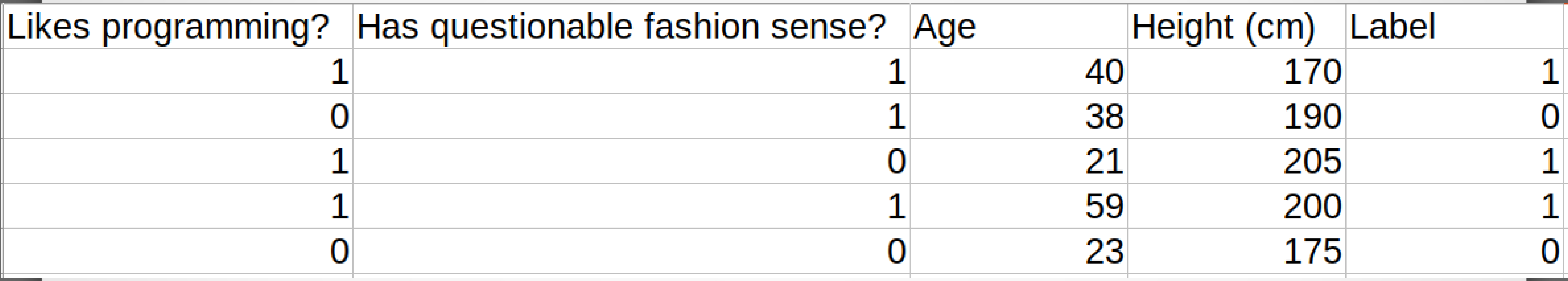
When we say that we want to teach a machine to make a prediction or classification, we mean that we want it to predict an *output* (or *label*) given a certain set of *inputs*. In ML terminology, inputs are referred to as *features.*

Examples :

- Predicting whether a patient has a certain condition or can receive a treatment based on aspects of their health, demographics, time since onset etc
- Predicting the likelihood of someone reoffending based on features such as the nature of their original crime, whether they served a custodial sentence etc
- Classifying whether the image presented is an image of a flower
- Classifying whether the text presented is positive or negative in tone.

Feature Weighting

Classifying whether someone is a PenCHORDian



Likes programming?	Has questionable fashion sense?	Age	Height (cm)	Label
1	1	40	170	1
0	1	38	190	0
1	0	21	205	1
1	1	59	200	1
0	0	23	175	0

In Machine Learning, features become *weighted* by the algorithm being used, so that their relative contribution to making a prediction of the label can be determined. This allows us to determine which features are more or less important for making a prediction.

Looking at the above, which features do you think are more important for predicting if someone is a PenCHORDian?

Types of Learning

There are three main types of learning in a Machine Learning algorithm :

Supervised Learning algorithms are ones in which the machine is shown examples of features and their outputs (labels) (a *labelled dataset*), and improves itself by getting better at getting from a given set of input values to the correct output.* Supervised Learning problems tend to either be *classification* problems (predict a discrete “class”) or *regression* problems (predict a continuous value).

Unsupervised Learning algorithms do not have access to any label data, and instead try to find inherent structures in the data (e.g. clusters of similar data, anomalies etc). This is useful when we don’t have data that has the “right answer”, often because we don’t know.

Reinforcement Learning algorithms learn by taking actions that are either rewarded or punished, and the machine gradually learns to make actions that lead to higher reward.

*There is also **Semi-Supervised Learning**, in which we have label data for some examples, but not all.

Classification vs Regression

Classification problems are ones in which we are trying to predict to which “class” an example belongs. Classifications may be *binary* (two classes – often True or False), or *multi-class* multiple different classes. There is also *multi-label classification*, where we are trying to predict more than one label / class for each example.

Examples :
- does this patient have condition x or not? (Binary classification)
- which condition does this patient have? (Multi-Class classification)
- what are the multiple conditions that this patient has? (Multi-Label classification)

Regression problems are ones in which we are trying to predict a continuous value as the output for an example. Often, the outputs represent a probability of something occurring or being true.

Examples :
- what is the probability this patient can safely receive this treatment?
- what is the probability that this patient will be imminently admitted to hospital?

Exercise 1

In your groups, spend the next 15 minutes discussing potential machine learning applications for each of the three main types :

- Supervised Learning
- Unsupervised Learning
- Reinforcement Learning

When you return, I’ll ask a few groups to share their ideas.

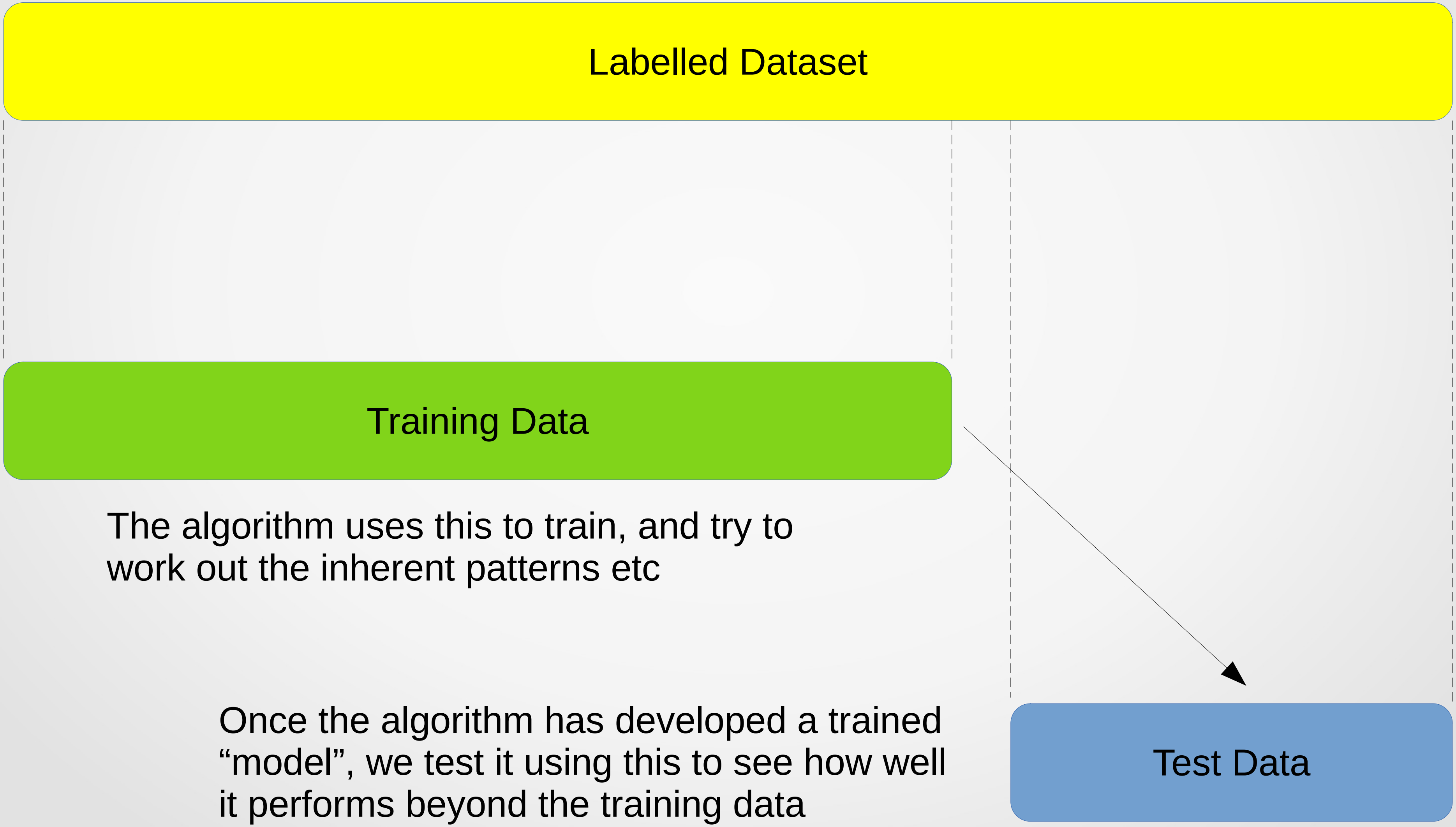
Train / Test Splits

When we’re trying to teach a machine to make a prediction using *Supervised Learning*, we need to present it with a set of data from which it will learn. This is known as the *Training Set*.

The Training Set contains examples of combinations of feature values, and the associated labels (ie – the “answer”).

But once the algorithm has “learned”, we also need a way of checking how well it’s learned. For that, we need to provide it with a set of data (complete with feature values and labels) **that it has not yet seen**, and against which it can check to see how well its learning translates beyond the data from which it has learned (*generalisability).* This is known as the *Test Set.*

Train / Test Splits



Labelled Dataset

Training Data

The algorithm uses this to train, and try to work out the inherent patterns etc

Once the algorithm has developed a trained “model”, we test it using this to see how well it performs beyond the training data

Test Data

Train / Validation / Test Splits

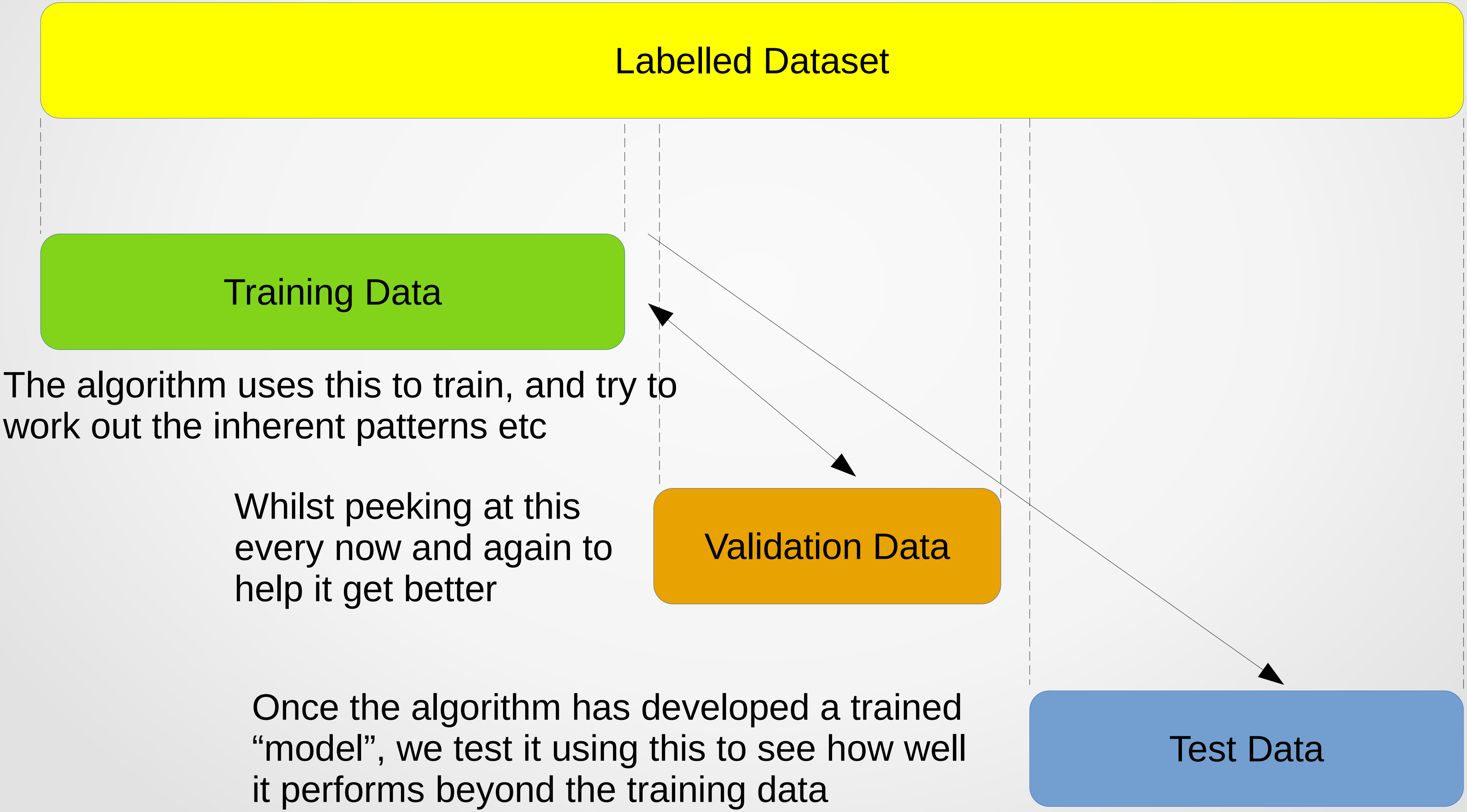
Later in the course, you will also hear about *Validation Sets*, and splitting your data into train / validation / test splits.

A Validation Set is a bit like a Test Set, but the algorithm has access to it during training so it can take a “peek” at how well it’s doing, and try to improve itself accordingly.

This is different to a Test Set, which is only seen by the algorithm once the algorithm has fully trained.

Validation Sets are commonly used when using Neural Networks for Machine Learning, which we’ll come onto later in the course.

Train / Validation / Test Splits



Labelled Dataset

Training Data

The algorithm uses this to train, and try to work out the inherent patterns etc

Whilst peeking at this every now and again to help it get better

Validation Data

Once the algorithm has developed a trained “model”, we test it using this to see how well it performs beyond the training data

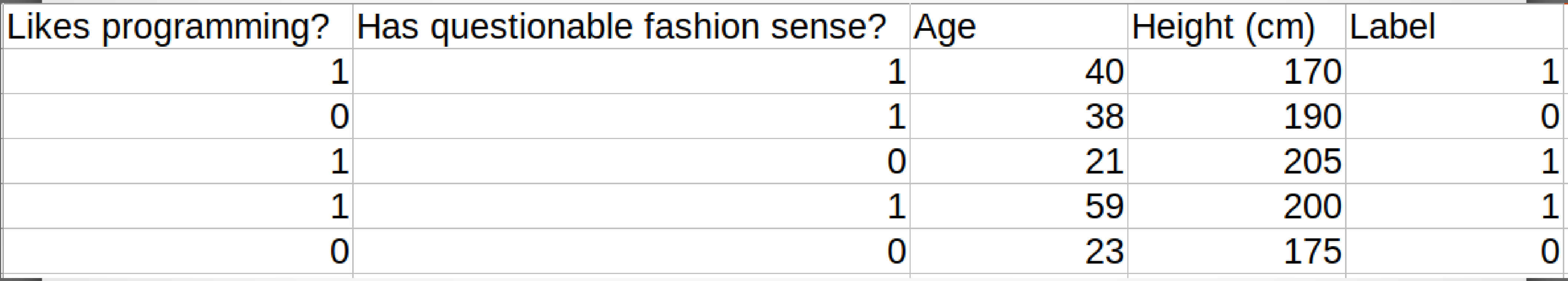
Test Data

Overfitting

When training a ML algorithm, we need to be careful about training too much, and running into a problem known as *Overfitting.* Let’s demonstrate using some examples.

Overfitting

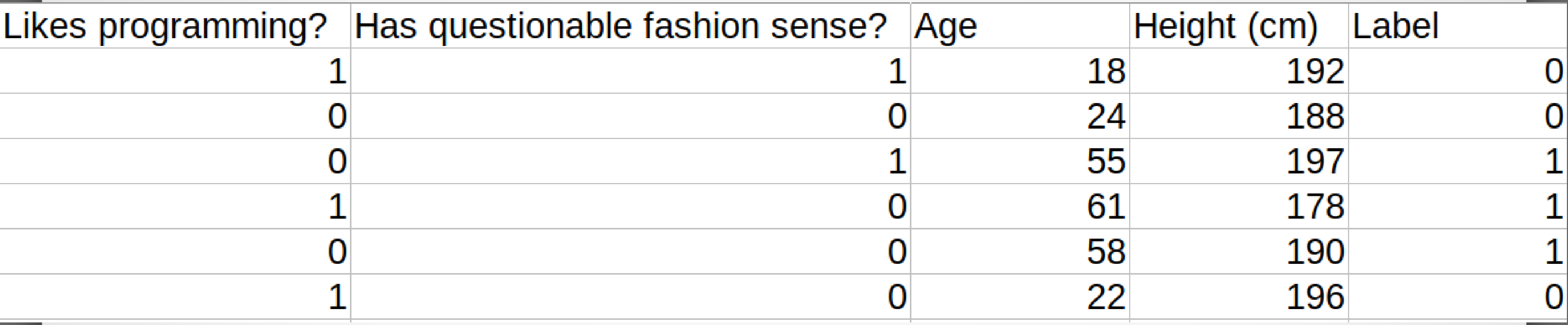
Study the following example from earlier again. Which would you say are the two most important features for determining if someone is a PenCHORDian?



Likes programming?	Has questionable fashion sense?	Age	Height (cm)	Label
1	1	40	170	1
0	1	38	190	0
1	0	21	205	1
1	1	59	200	1
0	0	23	175	0

Overfitting

Imagine that was our Training Set. Now imagine this is our Test Set.

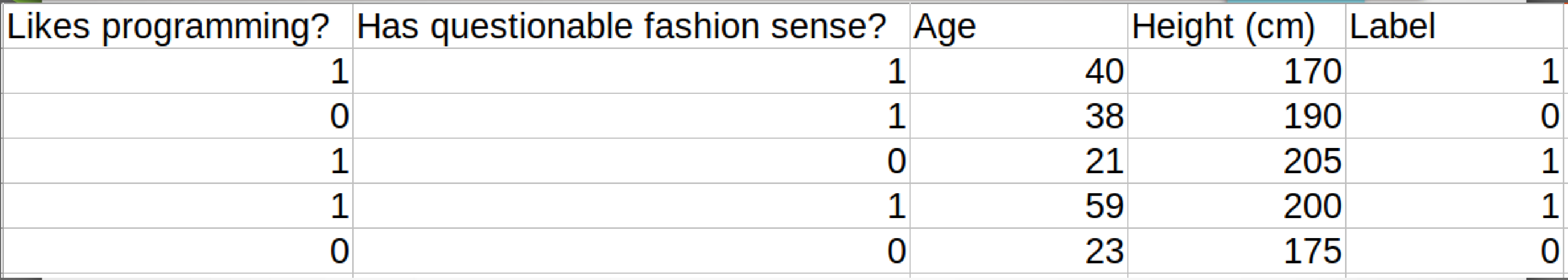


Likes programming?	Has questionable fashion sense?	Age	Height (cm)	Label
1	1	18	192	0
0	0	24	188	0
0	1	55	197	1
1	0	61	178	1
0	0	58	190	1
1	0	22	196	0

How well do you think your learning applies to the test set?

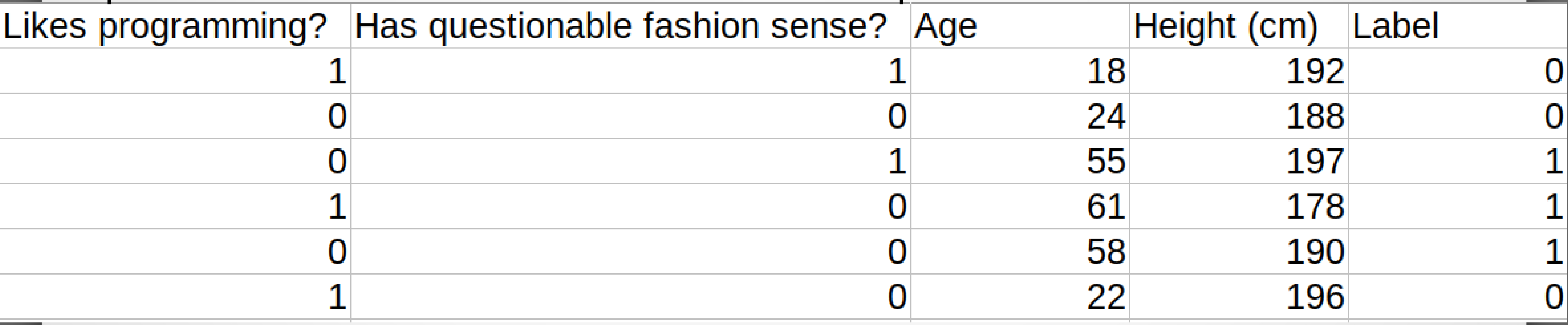
What’s happened?

Overfitting



Likes programming?	Has questionable fashion sense?	Age	Height (cm)	Label
1	1	40	170	1
0	1	38	190	0
1	0	21	205	1
1	1	59	200	1
0	0	23	175	0

In the Training Set (above), we concluded that “likes programming” and “questionable fashion sense” were the most important features.

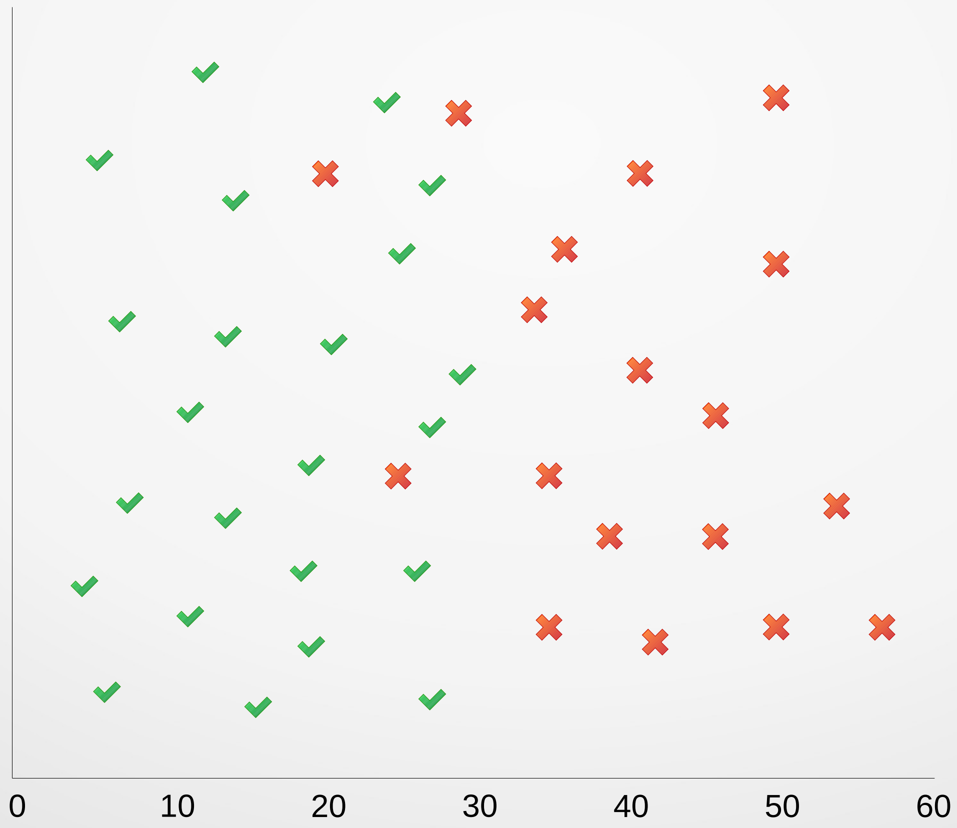


Likes programming?	Has questionable fashion sense?	Age	Height (cm)	Label
1	1	18	192	0
0	0	24	188	0
0	1	55	197	1
1	0	61	178	1
0	0	58	190	1
1	0	22	196	0

In the Test Set (above), we see that it seems the importance of these features was unique to the training set. When we look at the wider data, we see that actually “age” plays a much more important role. “Likes programming” and “questionable fashion sense” seem to generate as many 0s as 1s for our labels. In other words, our training has become too specific to the nuances of the training set – an overfit (*fitting* is the usual term used for “training” in ML. So *overfitting* = trained too much). Overfitted models lack generalisability, and are therefore typically useless.

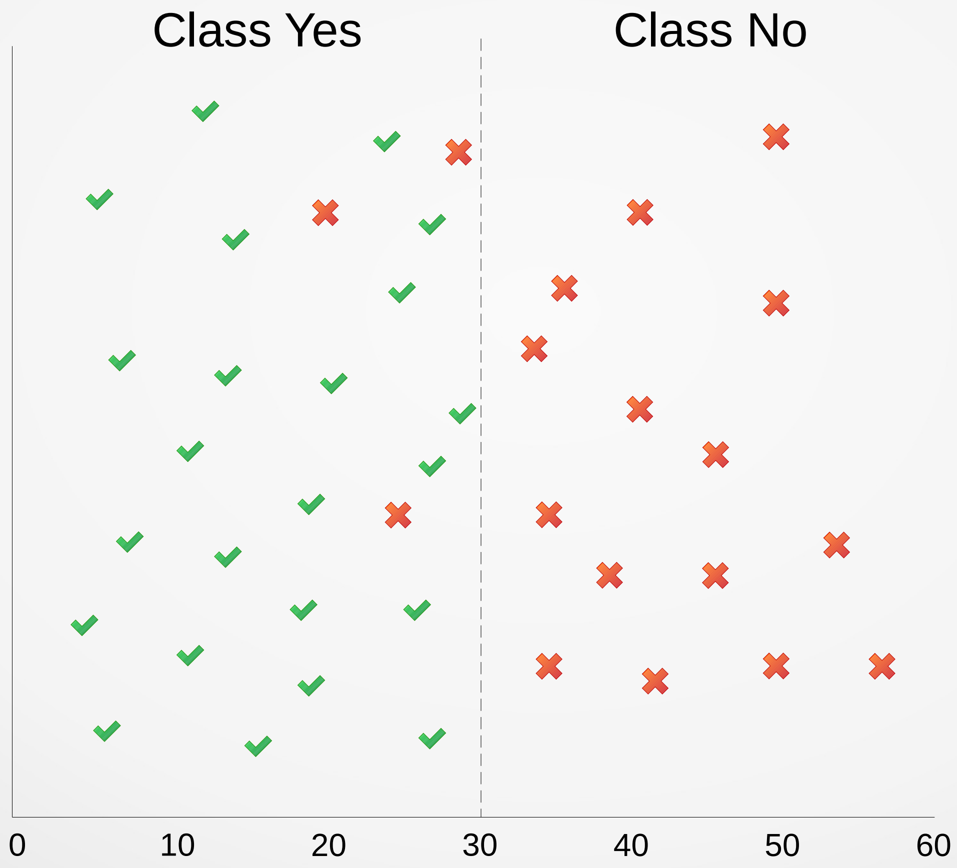
Overfitting

Let’s consider another example. Study the following plot. Where would you say the true dividing line is between the two classes?



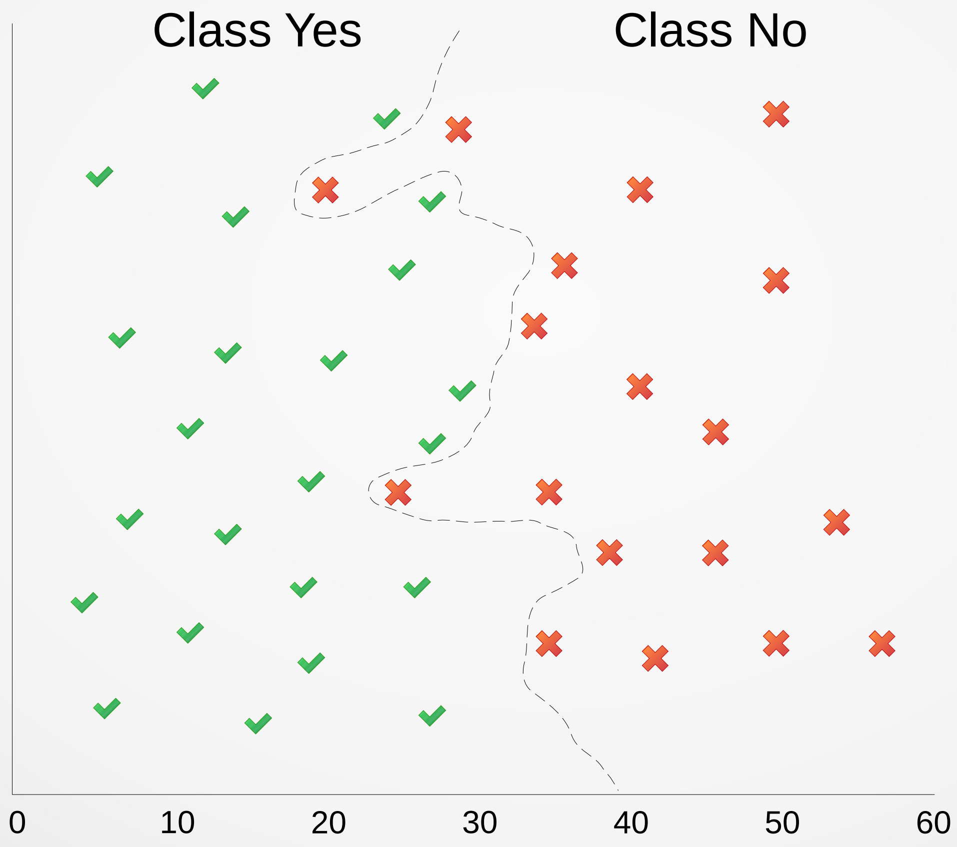
Overfitting

A model that has been trained (fitted) well



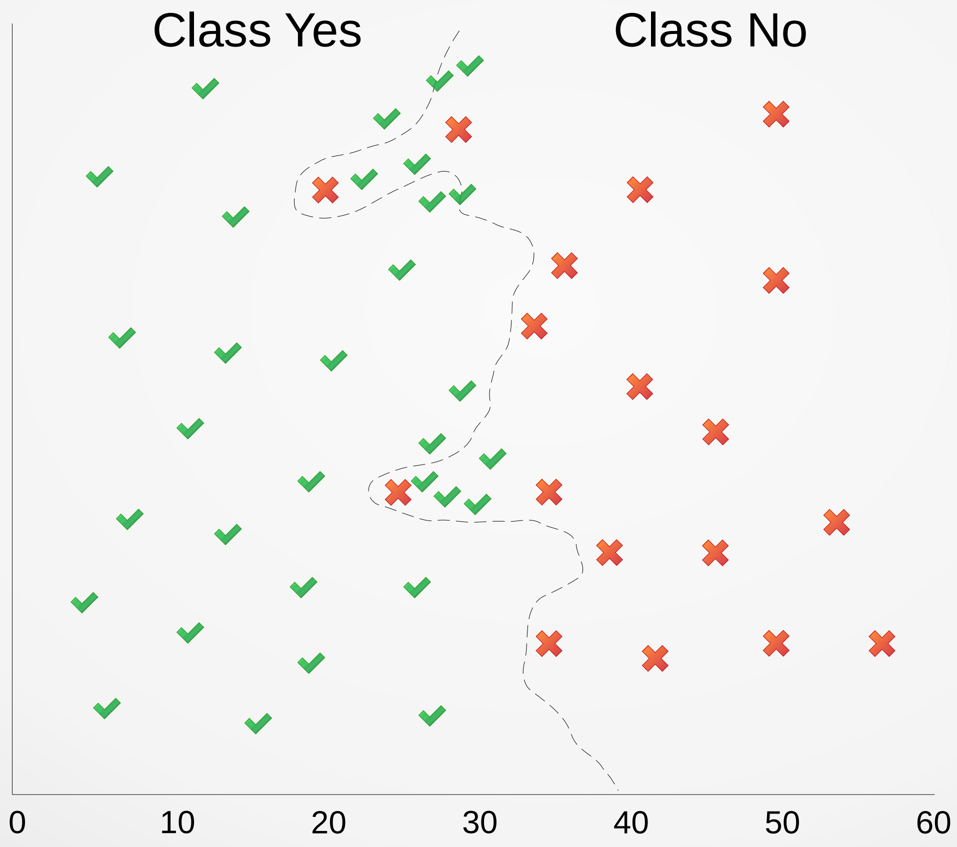
Overfitting

A model that has been overfitted.



It’s got everything correctly classified in the training set, but let’s now add some other points from the wider data...

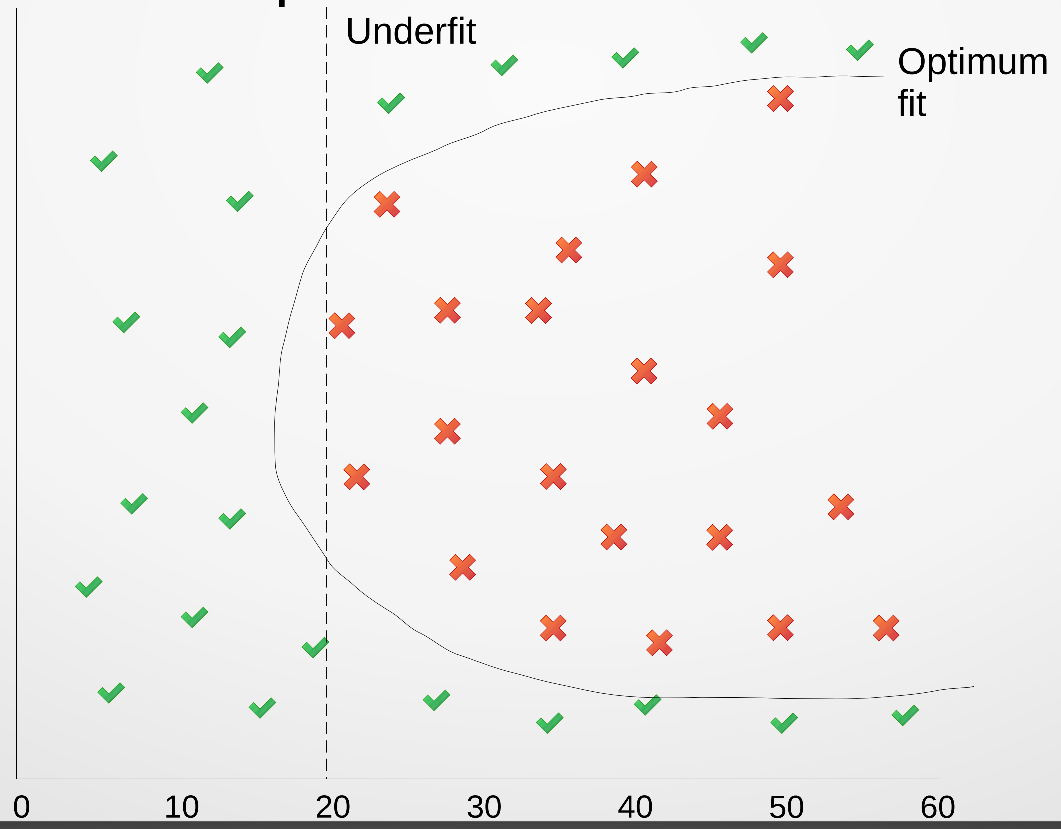
Overfitting



We've learned a pattern that's too specific to the training data, and which isn't generalisable. It's overfitted.

Underfitting

Underfitting can also be a problem. This is the opposite problem – where you haven’t trained enough (or the model is otherwise too simple), and your model is rubbish on both the training set and the test set. Fortunately, because it’s rubbish on the training set too, that’s usually easier to spot!



Performance Metrics

When developing a ML model, it’s important that we know how well it’s working.

We can use various metrics to assess how well the model has learned, depending on the type of model we’re building.

Today we’ll look at a few of the core ones for classifier models, but we’ll be covering other performance metrics later in the course.

Accuracy

When we’re building a Classifier (a ML model that makes a classification prediction), the three most commonly used metrics to assess performance are Accuracy, Precision and Recall.

Accuracy gives the percentage of predictions that the model makes correctly in the Test Set. In other words :

accuracy = total correct / number of predictions

🟢	🔴	🟢	🔵	🔴	🔵	🟢	🔴	🟢	🔵
GREEN	RED	RED	GREEN	RED	BLUE	GREEN	BLUE	RED	BLUE
✓	✓	✗	✗	✓	✓	✓	✗	✗	✓

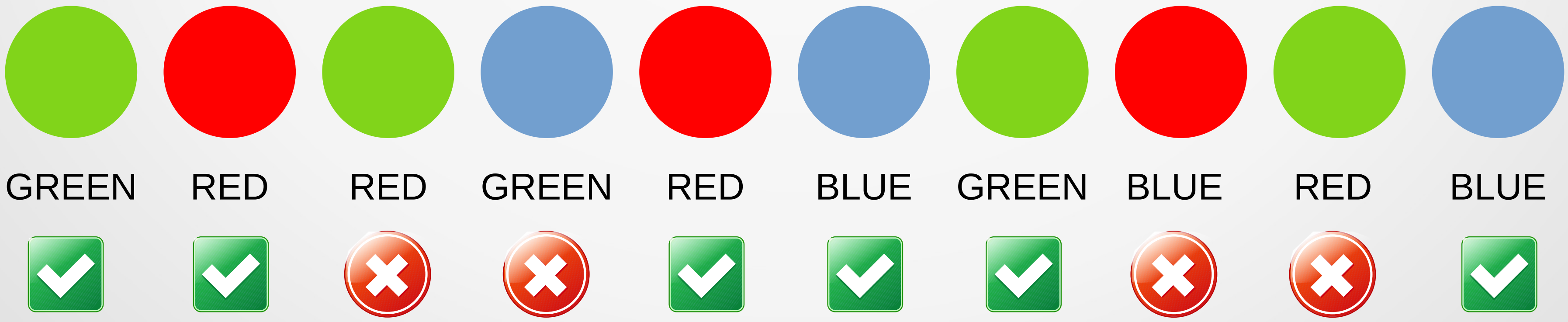
What is the accuracy of these results?

Precision

Precision gives the percentage of correct predictions for the classification of interest, out of the total number of times this classification was predicted (both correctly and incorrectly). So :

$$\textbf{precision}_c = \textbf{true positives}_c \,/\, (\textbf{true positives}_c + \textbf{false positives}_c)$$

Precision is a useful metric when classes aren't evenly represented.



🟢	🔴	🟢	🔵	🔴	🔵	🟢	🔴	🟢	🔵
GREEN	RED	RED	GREEN	RED	BLUE	GREEN	BLUE	RED	BLUE
✓	✓	✗	✗	✓	✓	✓	✗	✗	✓

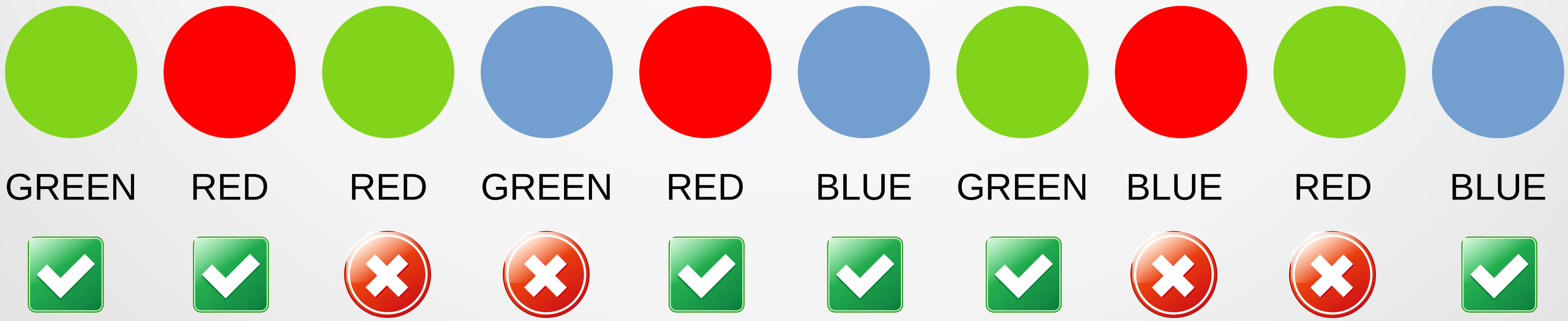
What is the precision for classifying blue circles in the example above?

Recall

Recall gives the percentage of correct predictions for the classification of interest, out of the total number of examples that truly have that classification. So :

$$\textbf{recall}_c = \textbf{true positives}_c \;/\; (\textbf{true positives}_c + \textbf{false negatives}_c)$$

Recall is a useful metric when classes aren’t evenly represented.



🟢	🔴	🟢	🔵	🔴	🔵	🟢	🔴	🟢	🔵
GREEN	RED	RED	GREEN	RED	BLUE	GREEN	BLUE	RED	BLUE
✓	✓	✗	✗	✓	✓	✓	✗	✗	✓

What is the recall for classifying green circles in the example above?

Accuracy, Precision and Recall

So, what do these metrics mean in reality? Imagine we are trying to build a classification model to determine whether someone has cancer or not.

A high **accuracy** would mean that the model correctly diagnosed people (as having or not having cancer) most of the time.

A high **precision** of cancer diagnoses would mean that most of the time, when it says someone has cancer, they do have cancer, and it is rarely telling someone they have cancer when they don’t.

A high **recall** of cancer diagnoses would mean that it’s picking up most cases of cancer, and rarely saying someone does not have cancer when they do.

Question – why might a high accuracy on its own be misleading?

Metrics in action

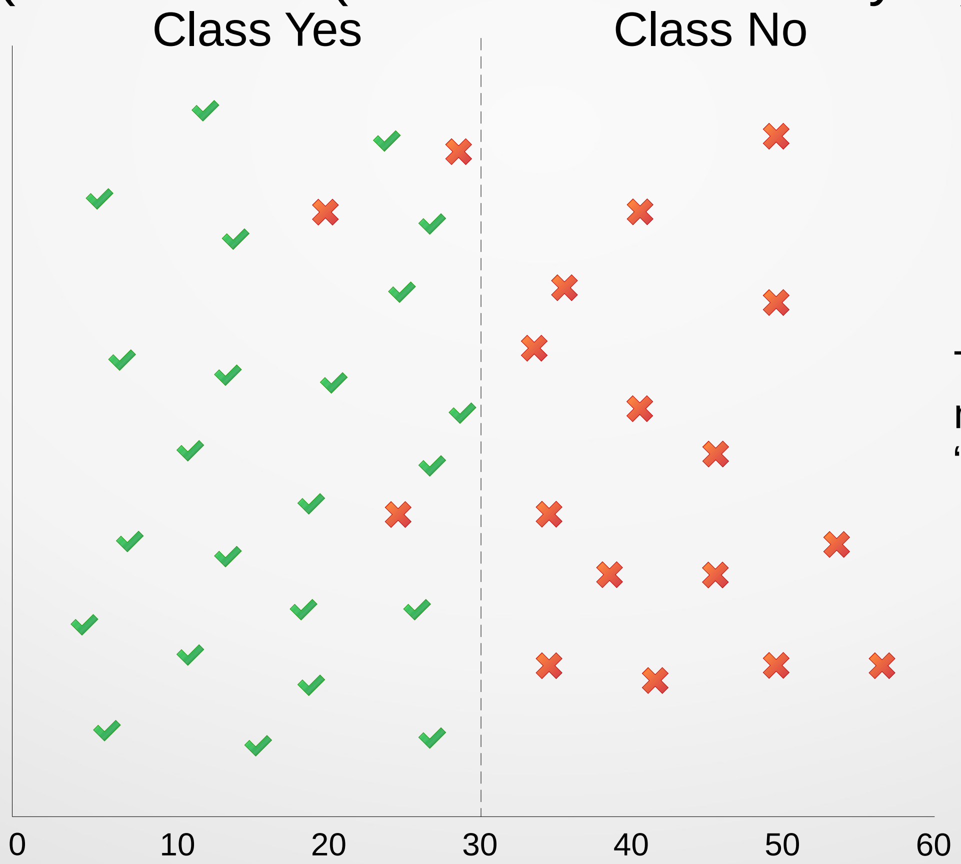
Accuracy = 93% (38 out of 41 correctly classified)

Precision$_{\text{Yes}}$ = 88% (23 true yes / (23 true yes + 3 false yes))

Precision$_{\text{No}}$ = 100% (15 true no / (15 true no + 0 false no))

Recall$_{\text{Yes}}$= 100% (23 true yes / (23 true yes + 0 false no))

Recall$_{\text{No}}$ = 83% (15 true no / (15 true no + 3 false yes))



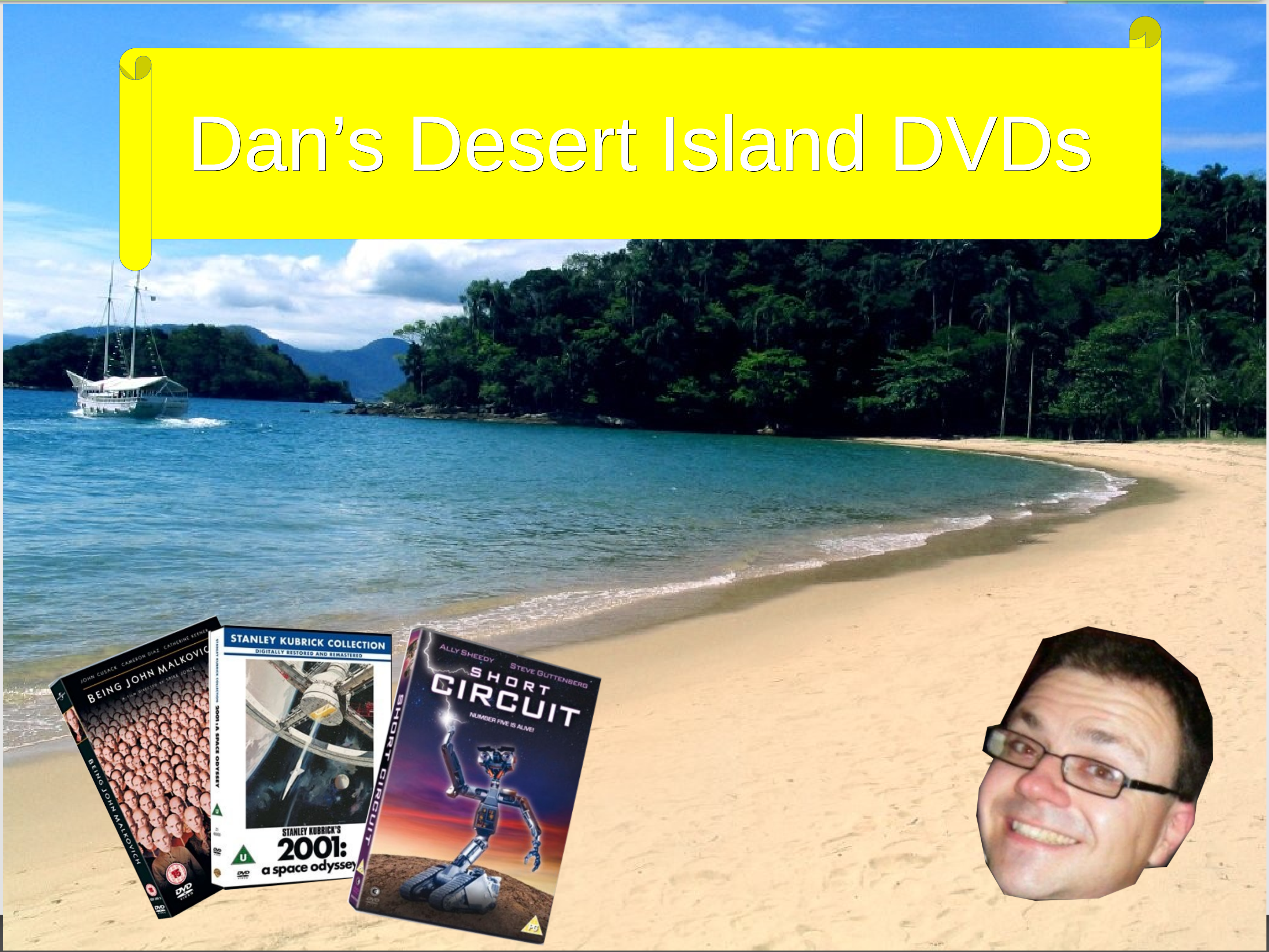
This side, our model says “yes”

This side, our model says “no”

These results suggest that this model is better at saying “you do have this” when you do, than it is at picking up cases where you don’t have it.

Exercise 2

Dan’s Desert Island DVDs



Exercise 2 – Dan’s Desert Island DVDs (1)

After a 10 minute break, you will work in your groups. Dan, having grown weary of the human race, has decided to relocate to a remote desert island. To keep himself occupied, he’s decided to pack a DVD player and a number of DVDs. We’ll suspend our disbelief about some of the logistical issues, such as lack of an electricity supply.

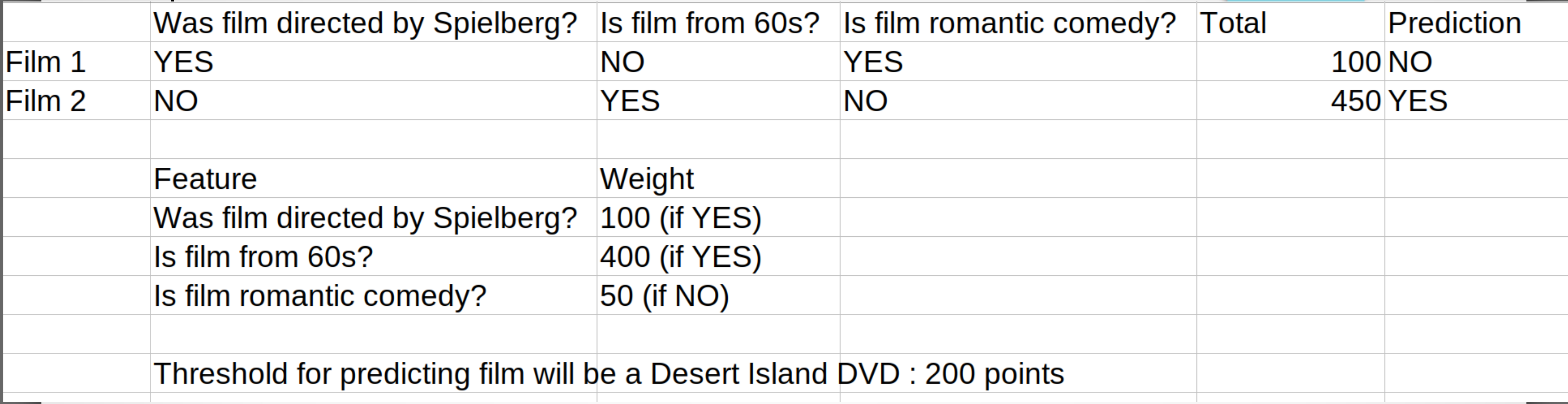
After 1 hour, you will be presented with a series of DVDs. Your task is to develop an “algorithm” that will attempt to classify whether or not Dan would want to take each DVD to the desert island or not. Dan enjoys a lot of films, but he’s got to be selective, as he can only pack so much. So he only wants to take the films he loves the most.

To help you develop your algorithm over the first hour, you will be provided with some “training data” – a list of films along with whether or not Dan decided they would make the “cut” to be taken onto the island. From this, you should decide which *features* you will incorporate into your training (e.g. things like director, genre, year of release etc). Use online resources (such as IMDb) to find out information about the films. For this exercise, your features should be boolean (e.g. “Is Director Steven Spielberg?”, “Was film released in 60s?”).

Your algorithm should give a weight score to each feature, such that a score is given to the feature if that feature is true (or false, or you could sometimes use negative scores). You then add up the scores for the feature values for that film, and use a threshold total score (which you decide) to determine whether the film is predicted to be a Desert Island DVD.

Exercise 2 – Dan’s Desert Island DVDs (2)

Example :



	Was film directed by Spielberg?	Is film from 60s?	Is film romantic comedy?	Total	Prediction
Film 1	YES	NO	YES	100	NO
Film 2	NO	YES	NO	450	YES
	Feature	Weight			
	Was film directed by Spielberg?	100 (if YES)			
	Is film from 60s?	400 (if YES)			
	Is film romantic comedy?	50 (if NO)			
	Threshold for predicting film will be a Desert Island DVD : 200 points				

You should use the training set to try to optimise your algorithm. You may want to consider carving out part of your training set as a *validation set*, and trying out your algorithm on the validation set every now and then to see how well it performs.

You should use the three metrics of accuracy, precision and recall to monitor how well you are training and adjust your algorithm (by changing weights, throwing out unimportant features, bringing in new features, changing the threshold etc). But remember – be wary about overfitting (and underfitting!).

In 1 hour, you’ll be given the Test Set, and you’ll be asked to swap your algorithms with one of the other groups. That group will then apply your algorithm to the Test Set examples, and will calculate your algorithm’s accuracy, precision and recall (they will have 30 minutes for this). Your algorithm **cannot be changed** once the test set has been released (hence why you’ll be giving them to another group to test). At the end of the exercise, we’ll find out who developed the best algorithm!