Data Science Algorithms In Plain English

Mark Whitehorn

It's all about me...

Prof. Mark Whitehorn
Emeritus Professor of Analytics
University of Dundee

Consultant
Writer (author)
m.a.f.whitehorn@dundee.ac.uk

It’s all about me…

Teach Masters in:

Data Science

Part time
Distance learning - aimed at existing data professionals

Data Engineering

Outline

Data Science is about extracting information from data and we are developing new algorithms for doing this all the time. Algorithms are ways of solving problems, they are not the code; indeed, many famous algorithms predate the development of computers. So, where do algorithms come from? How are they developed? Is each one new and unique? Are they developed from scratch each time or is there some kind of underlying framework?

Precision

We have two hours (including questions!).

I do have to make a fast getaway BUT we have time for question at half time.

I promised “plain English”.

I reckon I am only capable of two out of these three:

Short, understandable and precise.

So please forgive any imprecision.

What is an algorithm?

It is a way of solving a problem. Nowadays we generally implement algorithms as code in a computer language – R or Python are often used for machine learning but (within reason) almost any will do. (I used to use Pascal but I am very old.)

What is an algorithm? - Soundex

A phonetic algorithm that tries to encode homophones identically – so words that sound the same end up producing the same codified value (e.g. P434).

```pascal
Program Soundex;


Uses Crt;

Var
InputString   : String[255];
CodeString    : String[4];
Count,Counter : Integer;
FirstLetter   : Char;

{-------------------------------------------------------------------------}
Procedure TurnTheWholeStringIntoUpperCase;
Begin;
 For Count := 1 to Length(InputString) do
    InputString[Count] := UpCase(InputString[Count]);
End;
{-------------------------------------------------------------------------}
Procedure CatchTheFirstLetterOfTheStringAndPutItIntoTheOutputString;
Begin;
 FirstLetter := InputString[1];
End;
{-------------------------------------------------------------------------}
Procedure CodeTheStringIntoNumbers;
Begin;
 For Count := 1 to Length(InputString) do
   Case InputString[Count] of
     'B','P','F','V'                 : InputString[Count] := '1';
     'C','S','K','G','J','Q','X','Z' : InputString[Count] := '2';
     'D','T'                         : InputString[Count] := '3';
     'L'                             : InputString[Count] := '4';
     'M','N'                         : InputString[Count] := '5';
     'R'                             : InputString[Count] := '6';
     'A','E','I','U','O','Y'         : InputString[Count] := '7';
     'H','W'                         : InputString[Count] := '8';
   End; {Of Case}
End;
{-------------------------------------------------------------------------}
Procedure IfTwoOrMoreIdenticalNumbersStraddleAn8ThenTurnAllExceptFirstInto7;
Begin;

 For Count := 2 to Length(InputString) do    {Starting with the second letter....}
  If  (InputString[Count] = '8') Then       {If you find an 'H' or a 'W'}
  Begin;
   Counter := Count + 1;
   While (Counter <= Length(InputString)) and         {While there is still a character to the right of the 'H' or 'W'..}
   ((InputString[Count-1]) = InputString[Counter]) do {and that character is same as the one to the left of the 'H' or 'W'}
   Begin;
     InputString[Counter] := '7';                   {Zap that character}
     Counter := Counter + 1;                        {Get ready to look at the next character along}
   End;
  End;
End;
{-------------------------------------------------------------------------}
Procedure IfTwoOrMoreIdenticalNumbersSitTogetherThenTurnAllExceptTheFirstInto7;
Begin;
 For Count := 1 to Length(InputString) do    {Starting with the first letter....}
 Begin;
   Counter := Count + 1;
   While (Counter <= Length(InputString)) and         {While there is still a character to the right of the 'H' or 'W'..}
   ((InputString[Count]) = InputString[Counter]) do  {and that character is same as the one to the left of the 'H' or 'W'}
   Begin;
     InputString[Counter] := '7';                   {Zap that character}
     Counter := Counter + 1;                        {Get ready to look at the next character along}
   End;
 End;
End;
{-------------------------------------------------------------------------}
Procedure PassUpToThreeNumbersIntoTheOutputStringAsLongAsTheyAreNot7or8;
Begin;
 CodeString  := '0000';
 CodeString[1] := FirstLetter;
 Counter := 1;
 Count := 2;
 While (Count <= Length(InputString)) and (Counter < 5) do
 Begin;
    If (InputString[Count] in ['1'..'6'])  then
    Begin;
      Counter := Counter + 1;
      CodeString[Counter] := InputString[Count];
    End;
    Count := Count + 1;
 End;
CodeString[0] := #4;  {Manually sets the length of the CodeString to 4}

End;
{-------------------------------------------------------------------------}

{-------------------------------------------------------------------------}
Begin;

ClrScr;
Repeat
Writeln;
Writeln('Please enter a word; just press ''Enter'' on it''s own to exit.');
InputString := '0000';
CodeString := '0000';
Readln(InputString);

TurnTheWholeStringIntoUpperCase;
CatchTheFirstLetterOfTheStringAndPutItIntoTheOutputString;
CodeTheStringIntoNumbers;
IfTwoOrMoreIdenticalNumbersStraddleAn8ThenTurnAllExceptFirstInto7;
IfTwoOrMoreIdenticalNumbersSitTogetherThenTurnAllExceptTheFirstInto7;
PassUpToThreeNumbersIntoTheOutputStringAsLongAsTheyAreNot7or8;

Writeln(CodeString,'     = Coded version of this word.');
until CodeString = '0000';

End.
```

Soundex

```
For Count := 1 to Length(InputString) do
  Case InputString[Count] of
    'B','P','F','V'                  : InputString[Count] := '1';
    'C','S','K','G','J','Q','X','Z'  : InputString[Count] := '2';
    'D','T'                          : InputString[Count] := '3';
    'L'                              : InputString[Count] := '4';
    'M','N'                          : InputString[Count] := '5';
    'R'                              : InputString[Count] := '6';
    'A','E','I','U','O','Y'          : InputString[Count] := '7';
    'H','W'                          : InputString[Count] := '8';
  End; {Of Case}
```

(The full code is provided just in case)

Soundex

'Penguin' raw encodes to P752775

Then we drop the duplicates and vowels to P525

Fred, Freddy and Freddie all encode to: F63

Soundex

Soundex is NOT a classic data science algorithm. But it makes the point that algorithm are not code and are not even about computers. Check out the dates of the patents.

Robert Russell

US1261167 (A) — 1918-04-02

US1435663 (A) — 1922-11-14

Algorithms

So, algorithms are simply a formalised way of solving a given problem.

In Data Science the terms is often used (perfectly appropriately) for processes that we apply to data in order to:

- understand it better
- see the patterns within it
- **make predictions about future data**

Defining terms

Data mining

We look for information in raw data.

Humans have to work out how to do this.

Some of the techniques they develop are unique to a given problem.

Others just happen to be highly applicable and eventually become enshrined into the “Data Mining Hall of Fame”.

Defining terms

Machine learning. A computer system (an algorithm) that can extract information from data without human guidance.

Machine learning often is based on data mining.

We can also say that if a data mining algorithm is used to process a set of test data, and the resulting pattern is stored for later use, then this is machine learning.

Algorithms

So, many data mining algorithms are used extensively in data science. So, let’s take look at a couple.

As an example:

Suppose that we have some data about how much we spend for a client (on advertising) and how much additional profit the client makes.

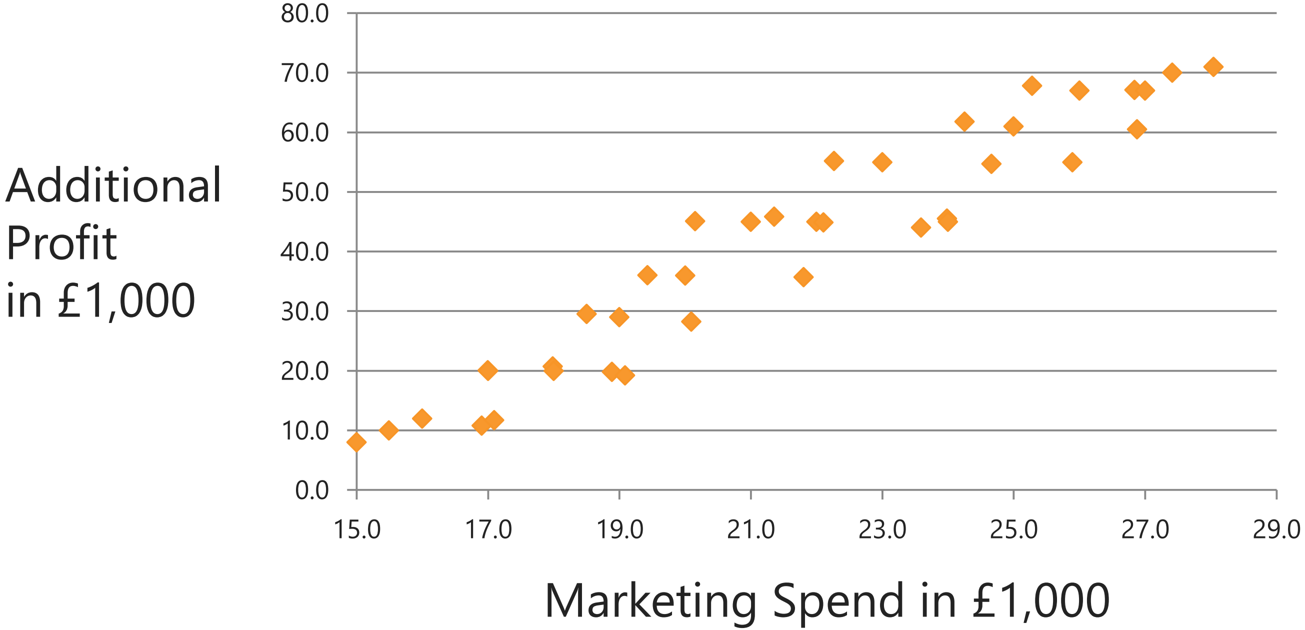
Algorithms

That data can be described as a set of X and Y coordinates.

Spend	Profit
14.2	7.0
15.0	8.0
15.5	10.0
16.0	12.0
16.9	10.8
17.0	20.1
17.0	20.0
17.1	11.7
18.0	20.7
18.0	20.0
18.5	29.6
18.9	19.8
19.0	29.0
19.1	19.2
19.4	36.0
20.0	36.0
20.1	28.2
20.2	45.1
21.0	45.0
21.4	45.8
21.8	35.7
22.0	45.0
22.1	44.9
22.3	55.2
23.0	55.0
23.6	44.1
24.0	45.5
24.0	45.0
24.3	61.8
24.7	54.7
25.0	61.0
25.3	67.8
25.9	55.0
26.0	67.0
26.8	67.1
26.9	60.5
27.0	67.0
27.4	70.0
28.0	71.0

Algorithms

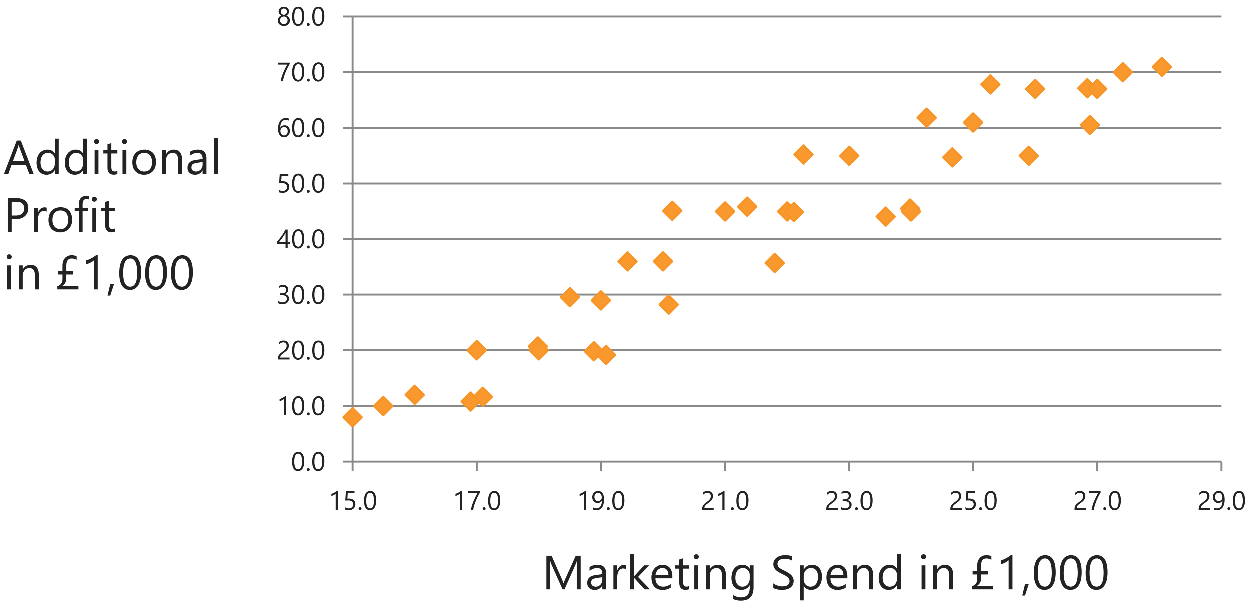
We can plot the data points.



Algorithms

Predicting the future. The client says “If we spend, say, £20K with you, how much additional profit will be see?”

What do we do?



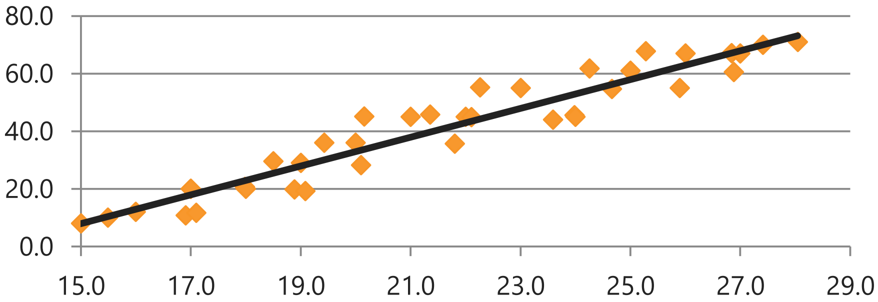
Algorithm Choice

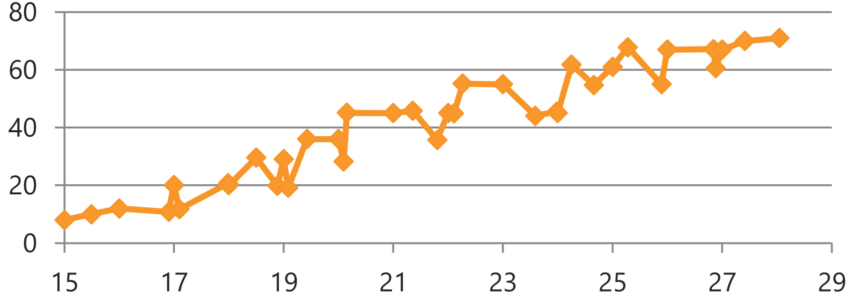
We can fit a line to the data.

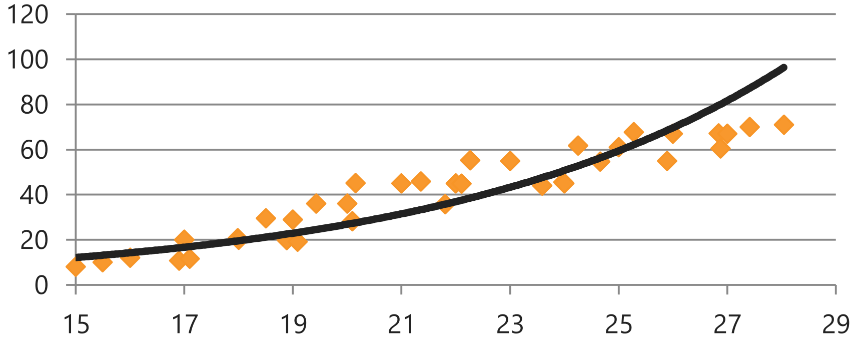
Which do you think is the best line to use in order to:

- understand the data better
- see the patterns within it
- **make predictions about future data?**

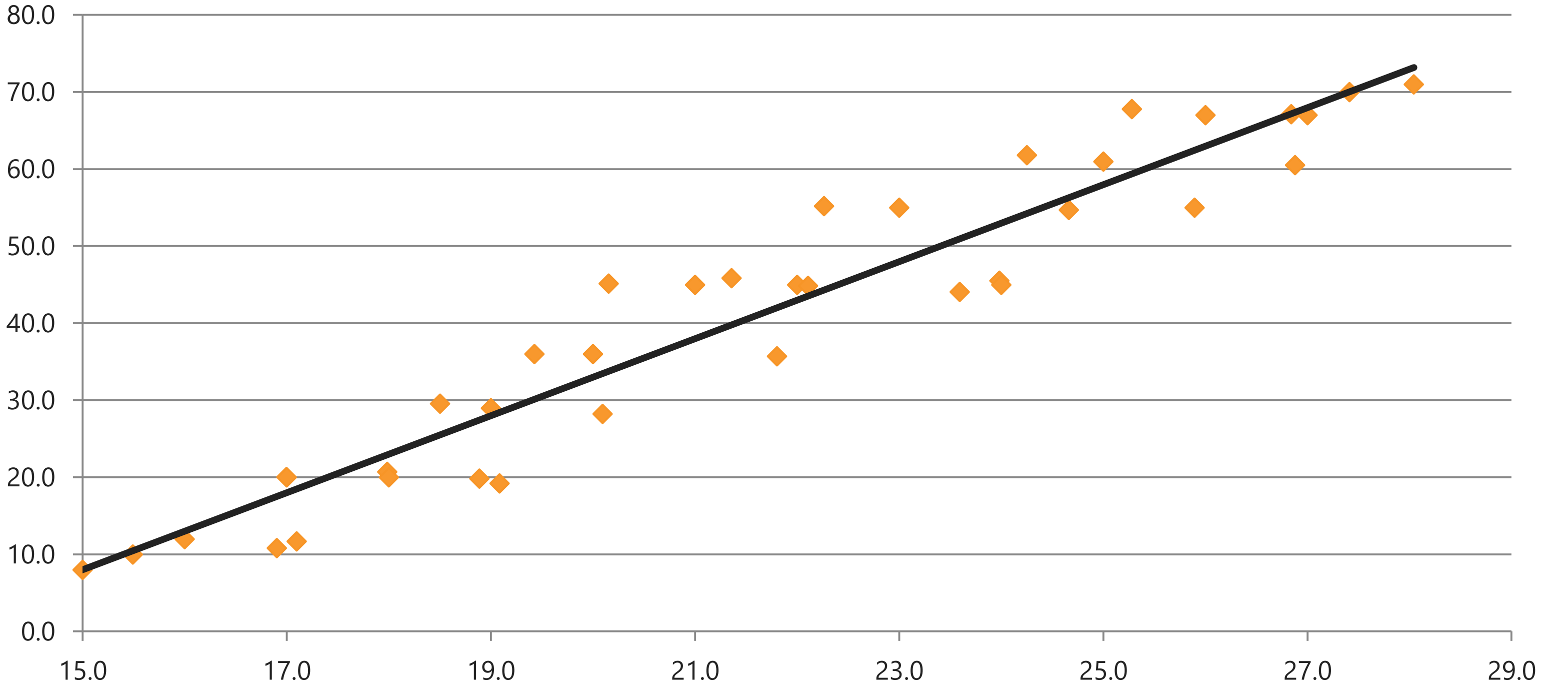
(If you can answer this correcty then you understand the general principle of 'overfitting').







Algorithm Choice



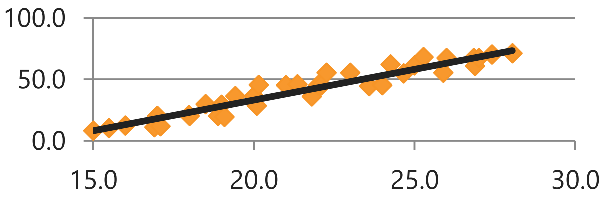
Additional Profit in £1,000

Marketing Spend in £1,000

Machine Learning

Now adding a line of best fit to a graph is something we probably all did in school.

But that simple act is a wonderfully elegant example that encompasses so much that is characteristic of algorithms in Data Science. **Linear regression** is the algorithm and what we did is not only an example of using an algorithm it is also an example of **Machine Learning (ML)**.



Machine Learning

Where we use an algorithm to ‘learn’ a pattern that is inherent in some existing data. That pattern can then used to make predictions about data that has not yet been analysed.

Data modelling (no, not that kind)

Better still, in formal terms, our line is a **data model.**

(Note that the relational model is a data model, so is the dimensional model.
But the term has another use as we are using it here.)

Data modelling

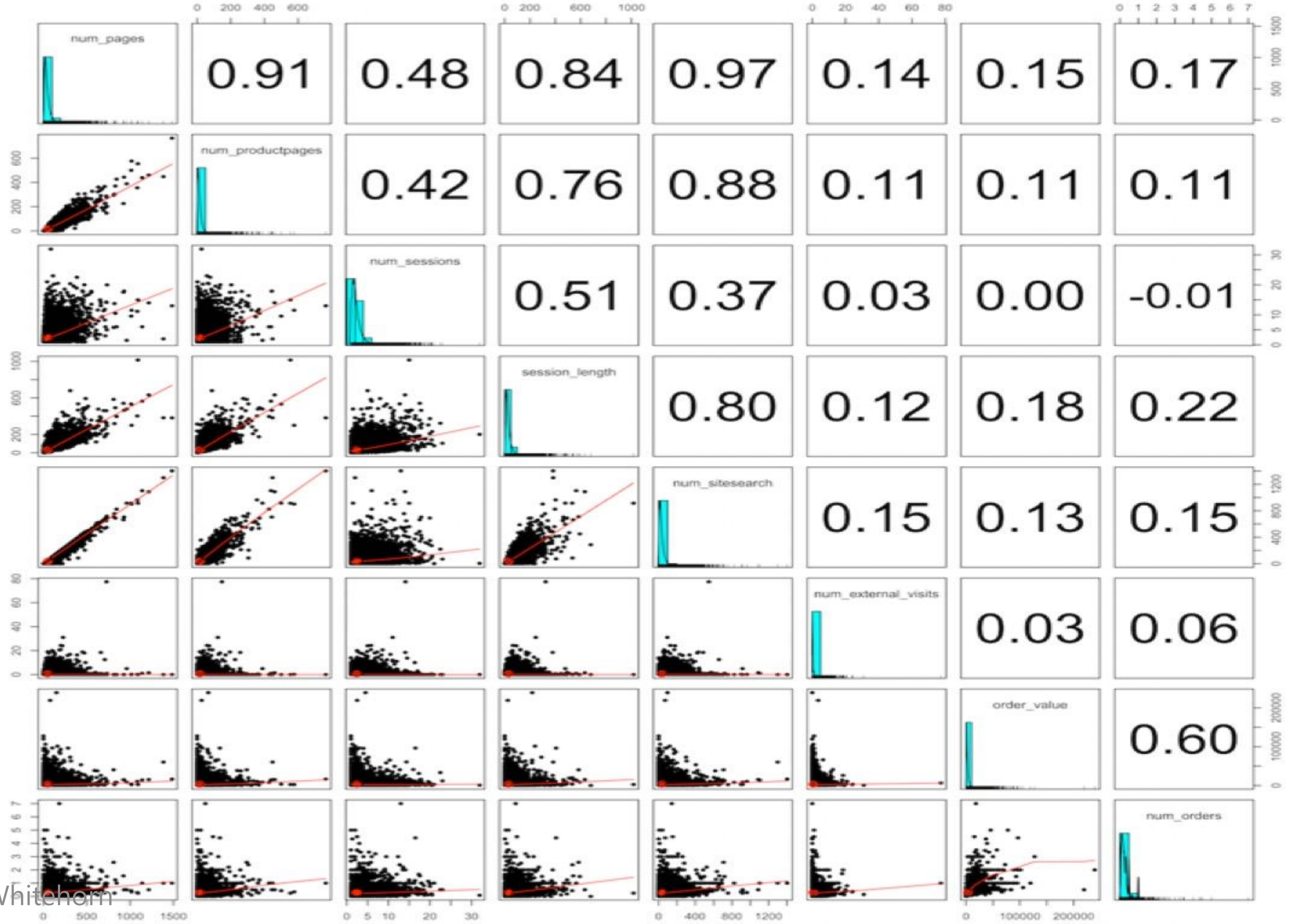
Training data + algorithm = data model

Our XY Coordinates + Linear regression = Line

We use the data model to predict the future.

Linear Regression

Linear regression finds the ‘best’ relationship between X and Y. To put that another way, it produces the values of Y from the X values with the least error.


num_pages
0.91
0.48
0.84
0.97
0.14
0.15
0.17
num_productpages
0.42
0.76
0.88
0.11
0.11
0.11
num_sessions
0.51
0.37
0.03
0.00
-0.01
session_length
0.80
0.12
0.18
0.22
num_sitesearch
0.15
0.13
0.15
num_external_visits
0.03
0.06
order_value
0.60
num_orders

Data modelling

Models are characteristically much smaller than the training data.

The line can be defined as an equation of the form:

$y = mx + c$

In our case it is:

$y = 5x - 67$

This is tiny compared to the 10,000 or 10^7 points that we could have used to train the model.

Yet this tiny, elegant, compact model can be used to predict future Y values given an X value.

Data modelling

A client has £20K to spend so we can predict an increased profit of £33K.

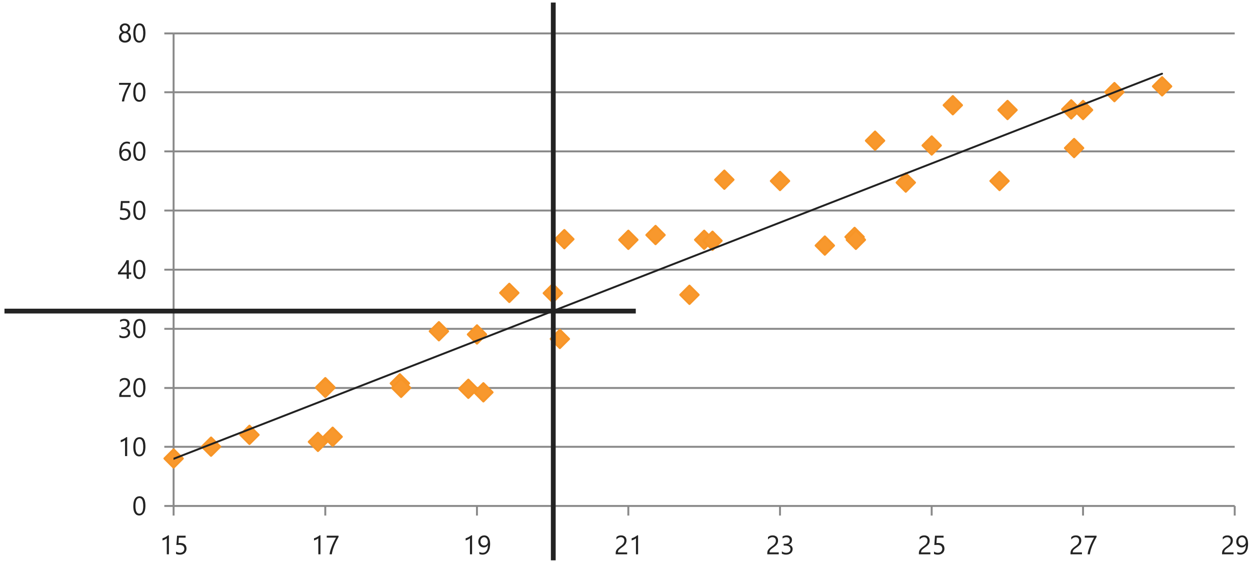
$$y = (5 * 20) - 67 = 100 - 67 = 33$$

Data modelling

Let’s just check. Yup! That looks intuitively correct.

X = 20

y = 33



Data modelling

In practice we would always do extensive testing of our model and very often tweak it.

Training data + algorithm = data model

Check model using test data

We have formal processes for this (ROC curves and so on).

ROC curves

Emma and I talked about ROC curves yesterday. If you missed the talk and I have now sparked your interest, check out the video.

Data modelling

Once we are happy it is good, we start to use the model.

Training data + algorithm = data model

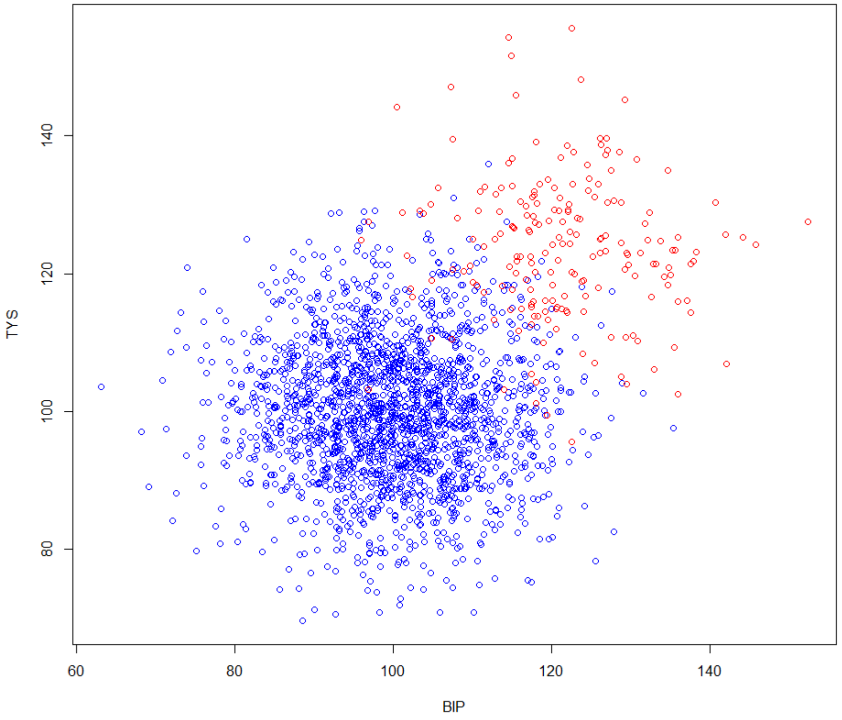
Check model using test data

New data + data model = new information

Example two.

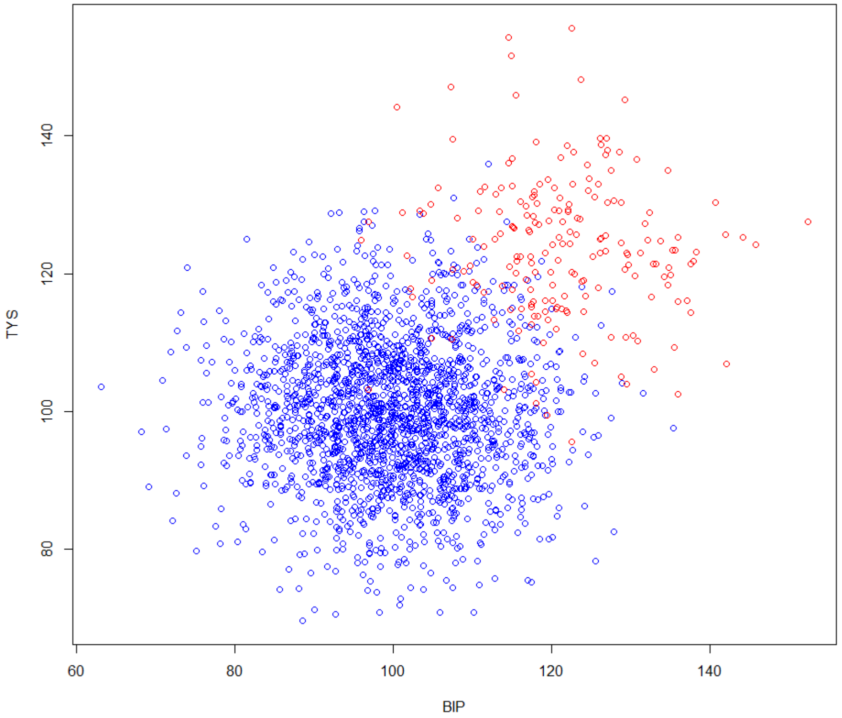
We also have some insurance claims data. BIP and TYS are complex measures of a claim, the result is whether we paid out or decided the claim was fraudulent. We have existing data on 2,000 honest claims and 200 fraudulent.

BIP	TYS	Result
103	109	Paid
125	123	Fraud
117	121	Paid
123	115	Paid
113	119	Paid



Example two.

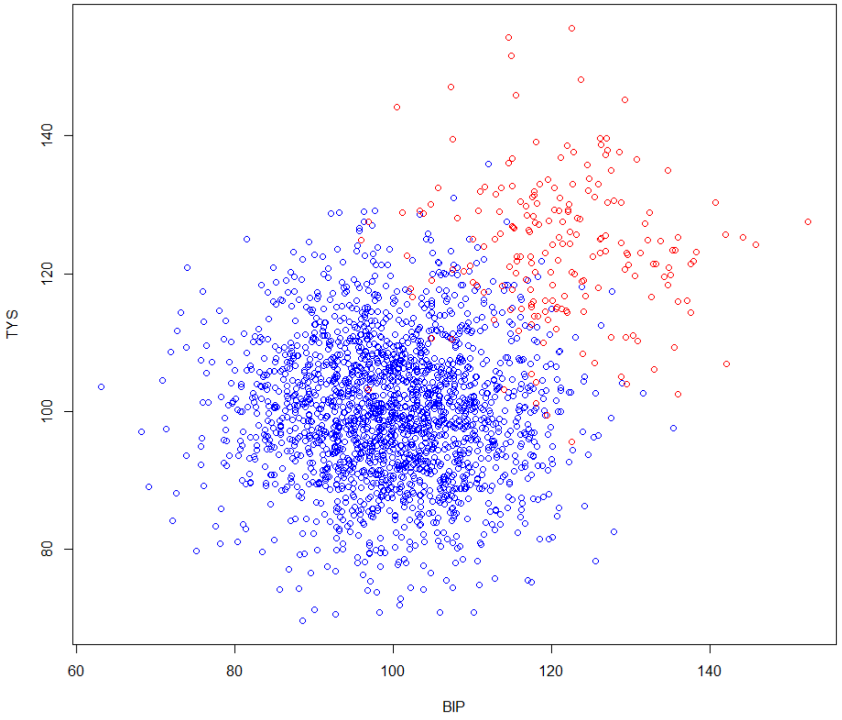
What modelling algorithm would you suggest here?



Example two.

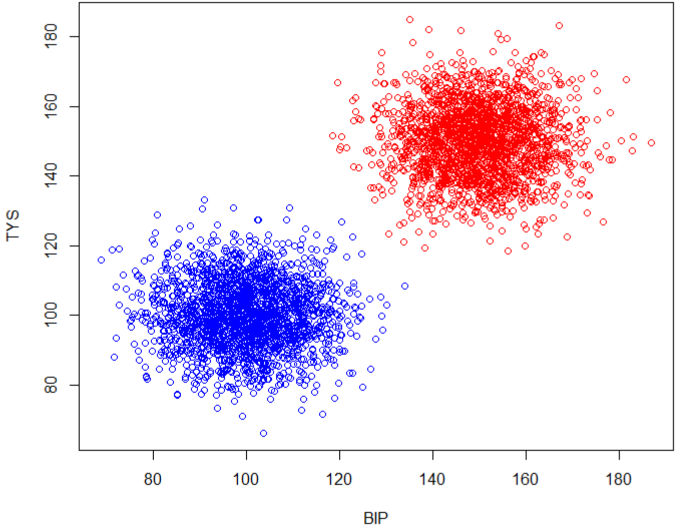
What modelling algorithm would you suggest here?

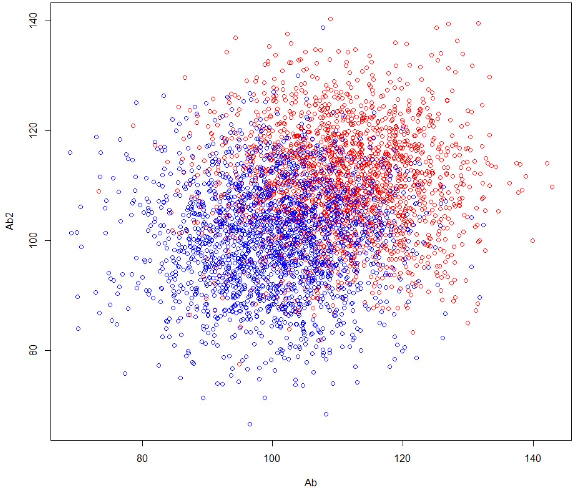
Clustering!



Example two.

Sometimes clusters are separated, sometimes they overlap. Let's take the harder case where they overlap. What is our data model and how do we use it to assign a case to new data?

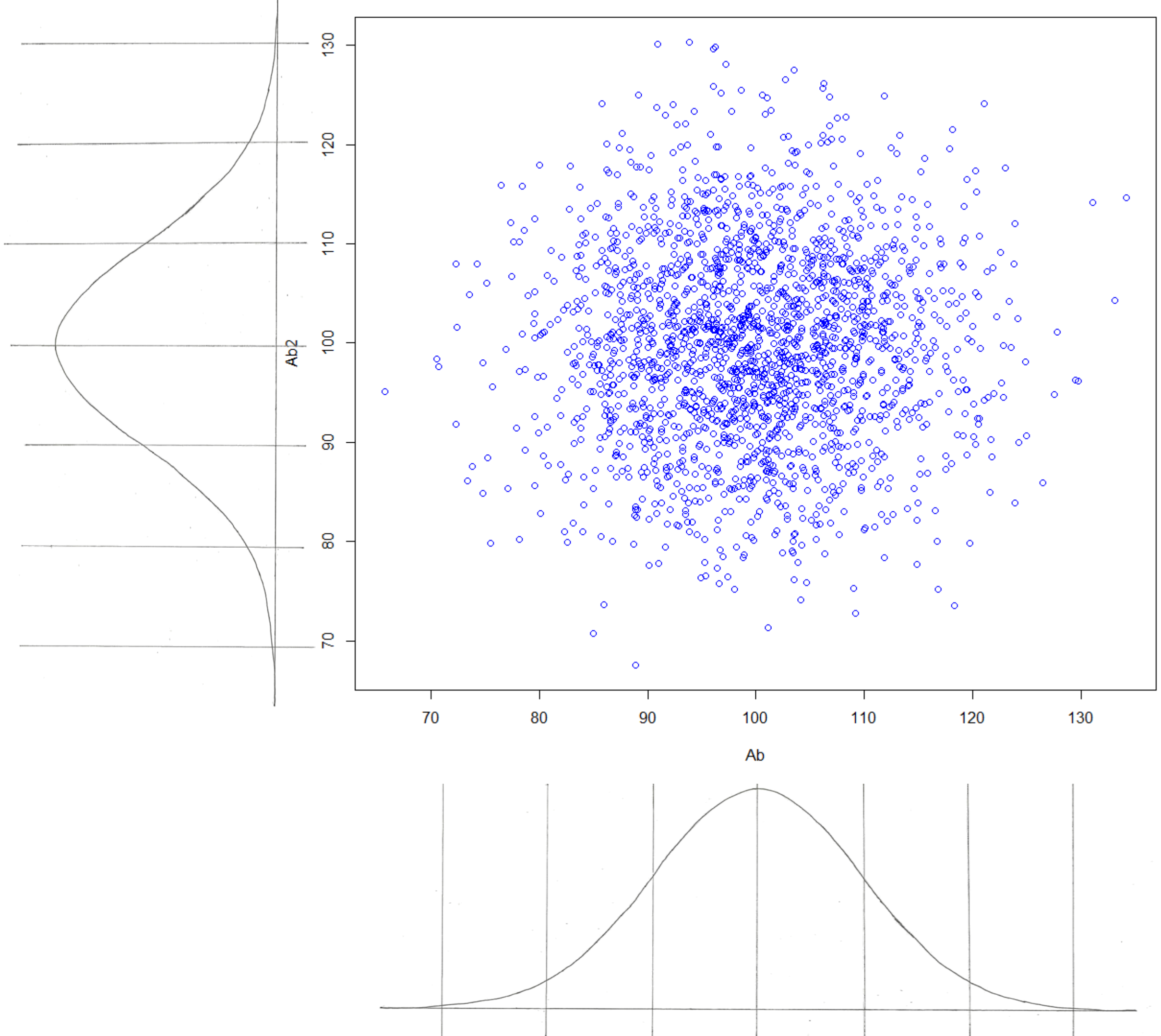




Clustering

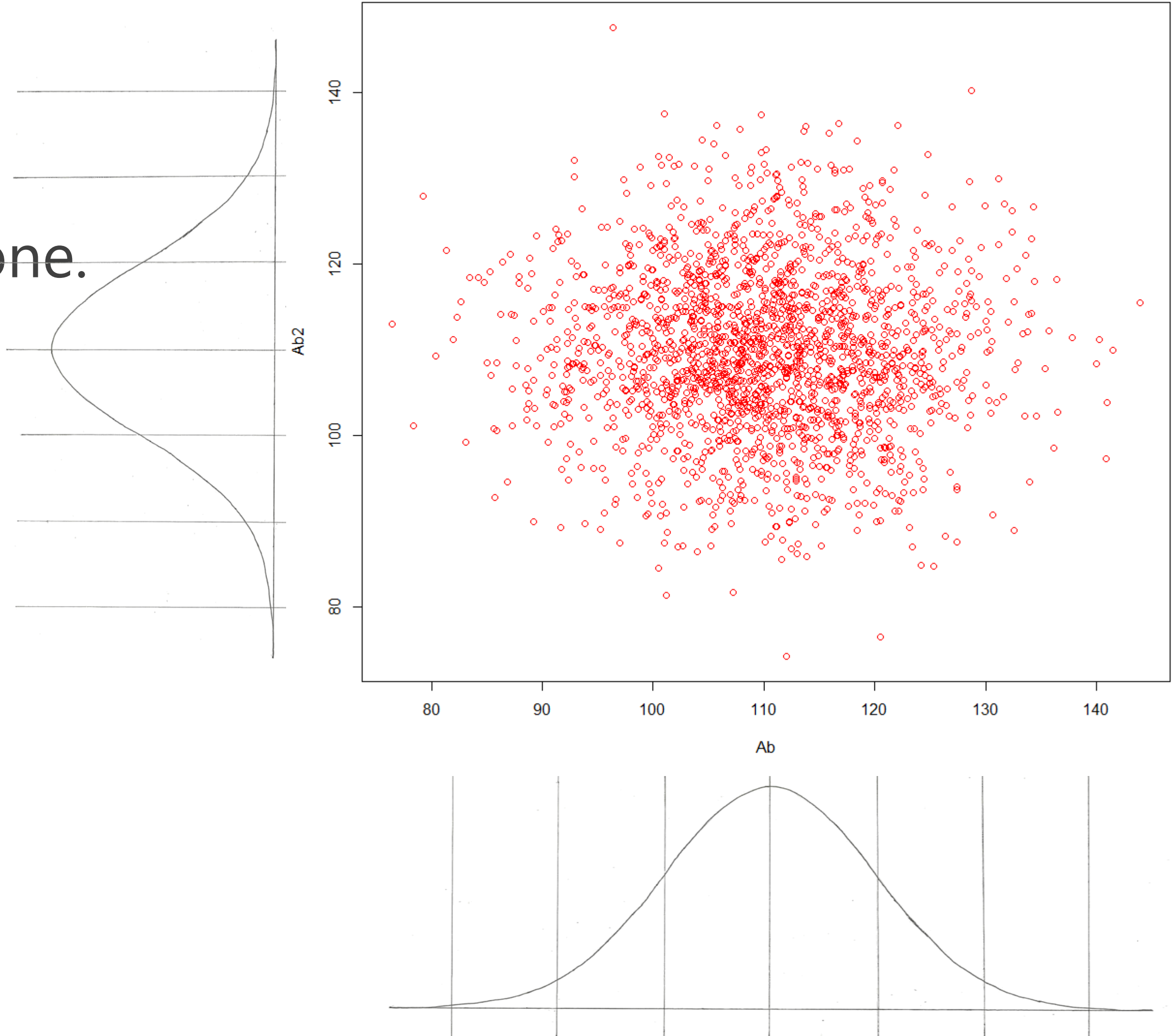
Let's talk zombies.

Any given cluster will have some distribution in each dimension.



Clustering

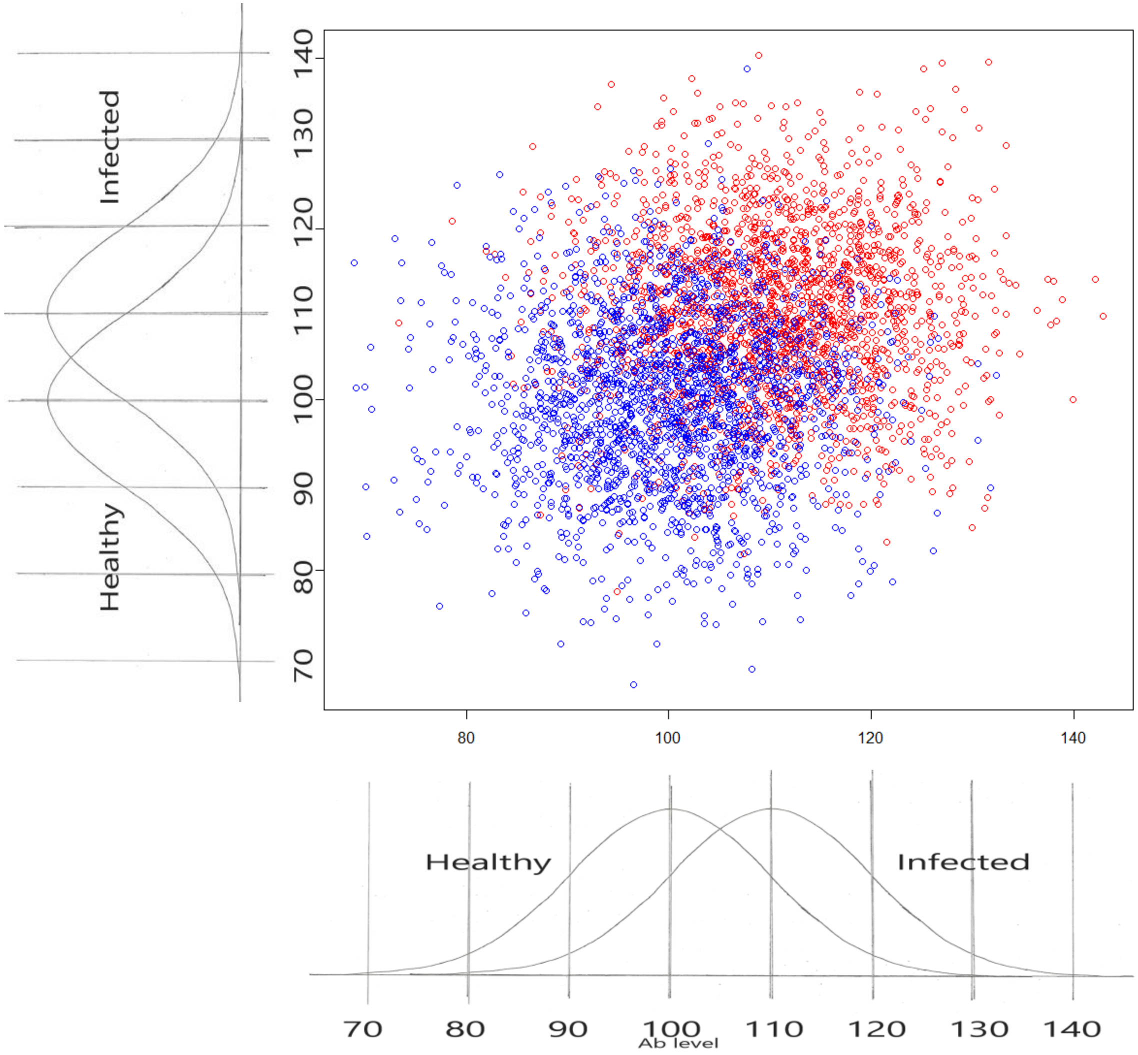
Same for the other one.



Clustering

So we can use the distributions to assign probabilities to new points.

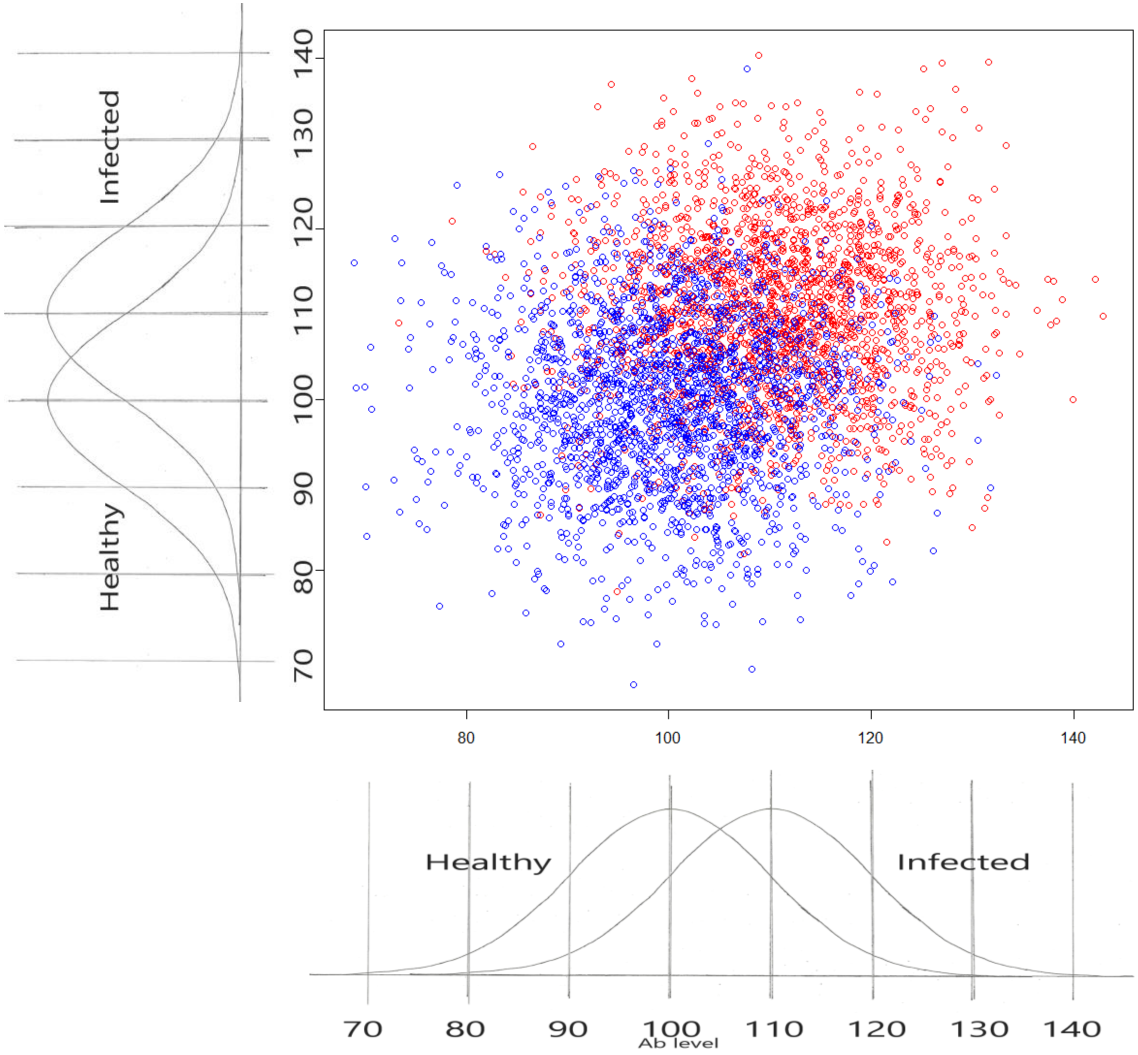
What is the model here?



Clustering

So we can use the distributions to assign probabilities to new points.

What is the model here? It is simply the mean and the standard deviation of the four distributions.

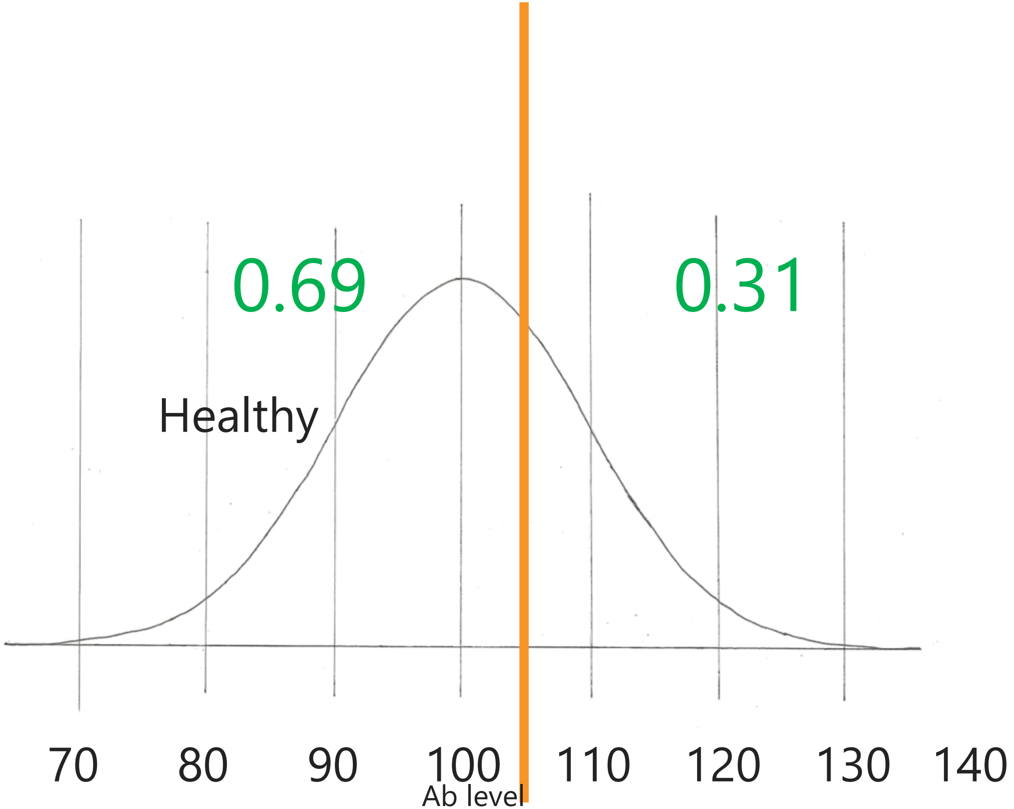


Data modelling

Training data	+ algorithm	= data model
Our XY Coordinates plus the classification	+ Clustering	= Cluster distributions

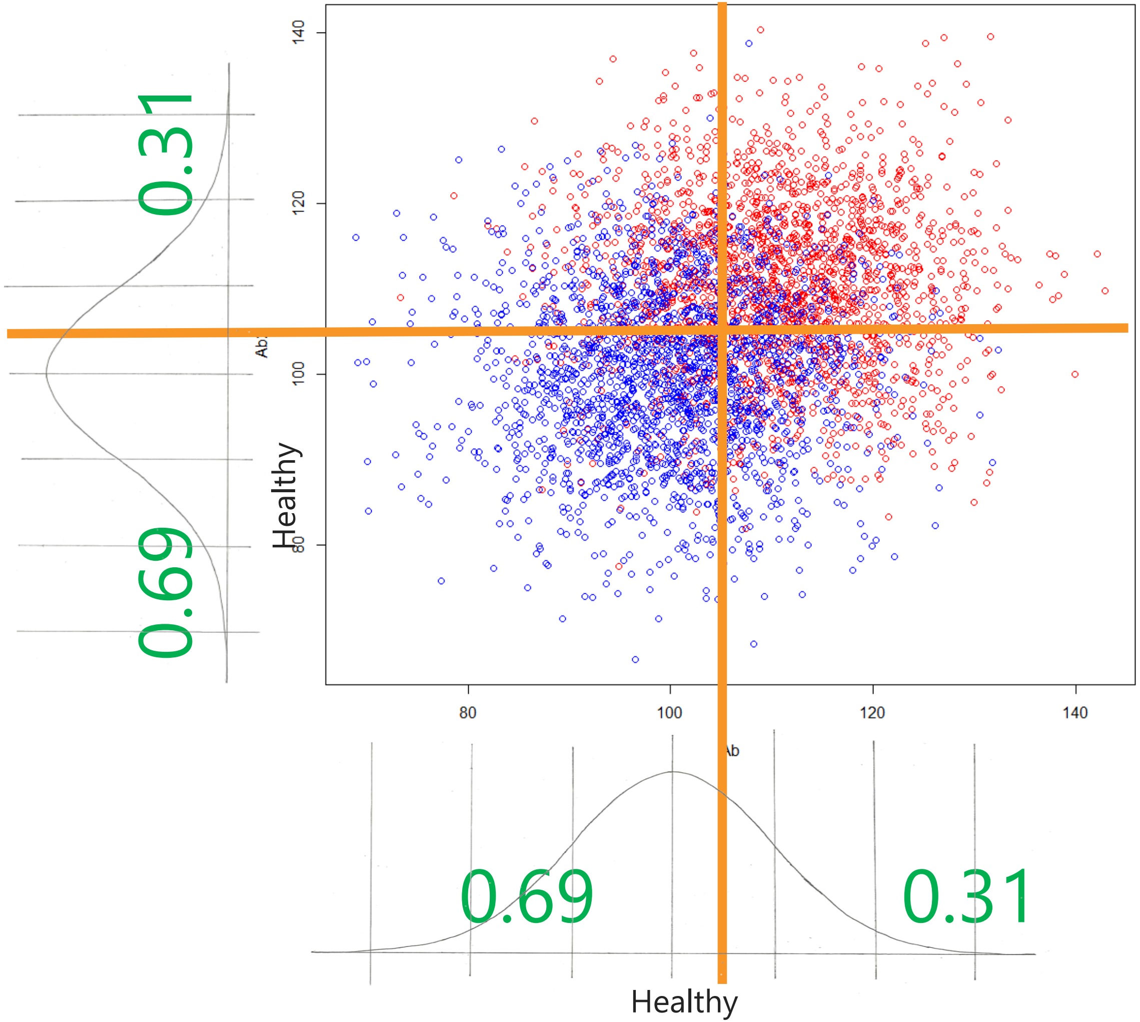
More about efficiency

We can say that 69% of the healthy people have an Ab of 105 or below and that 31% of them have an Ab > 105.



More about efficiency

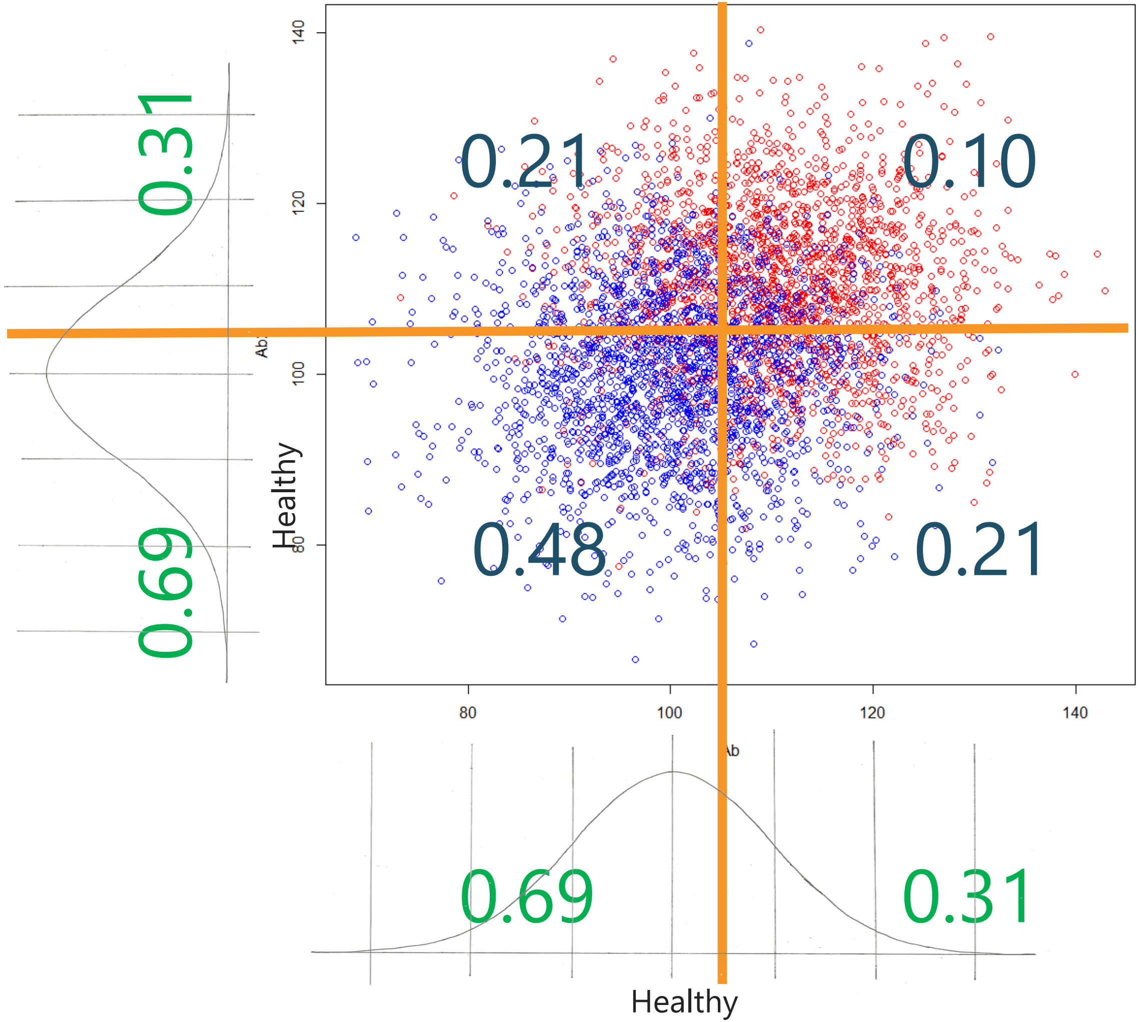
If we assume that the two factors are independent (which seems reasonable) it is then to calculate the numbers.



More about efficiency

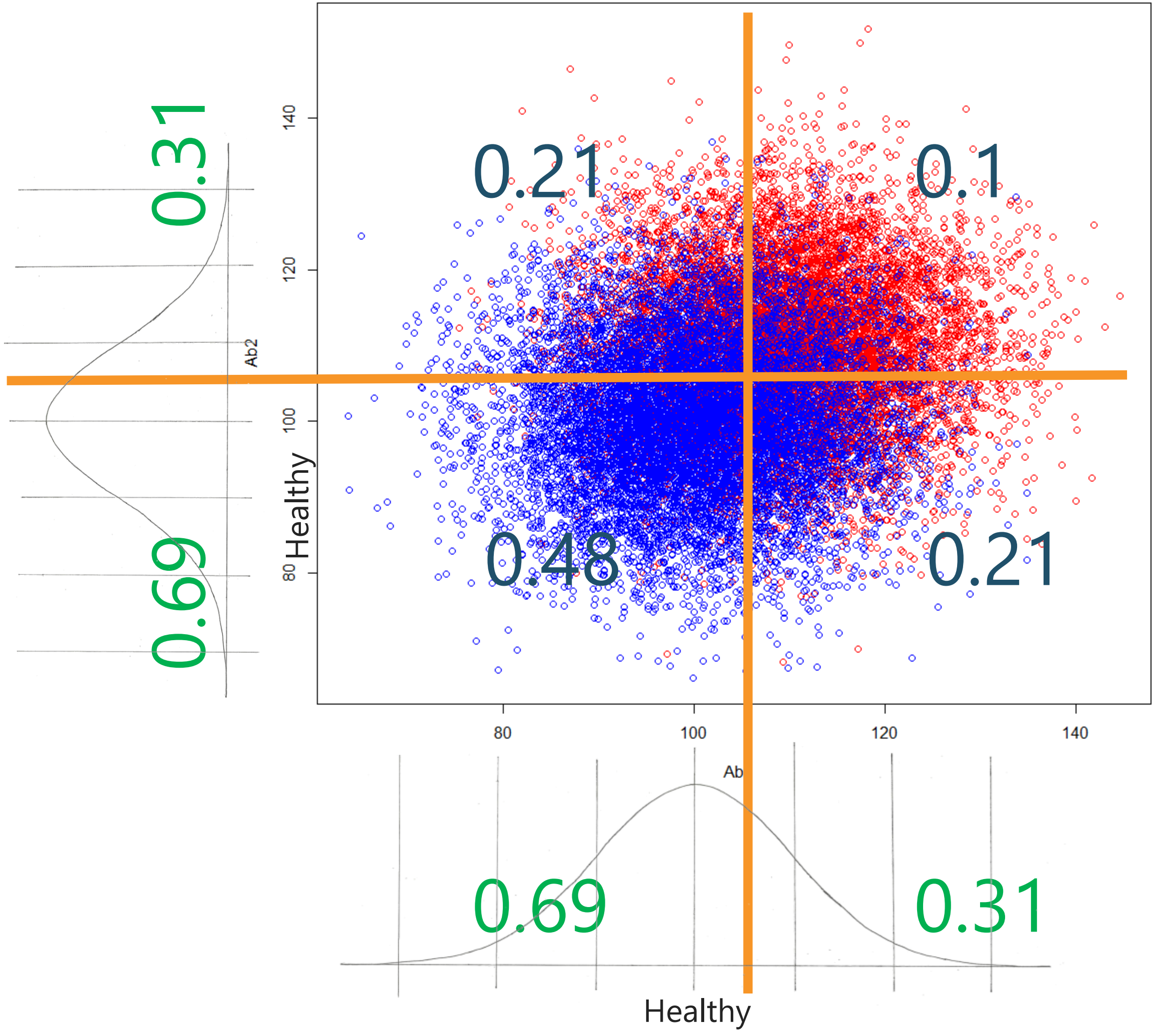
If we assume that the two factors are independent (which seems reasonable) it is then to calculate the numbers.

Does this look right?



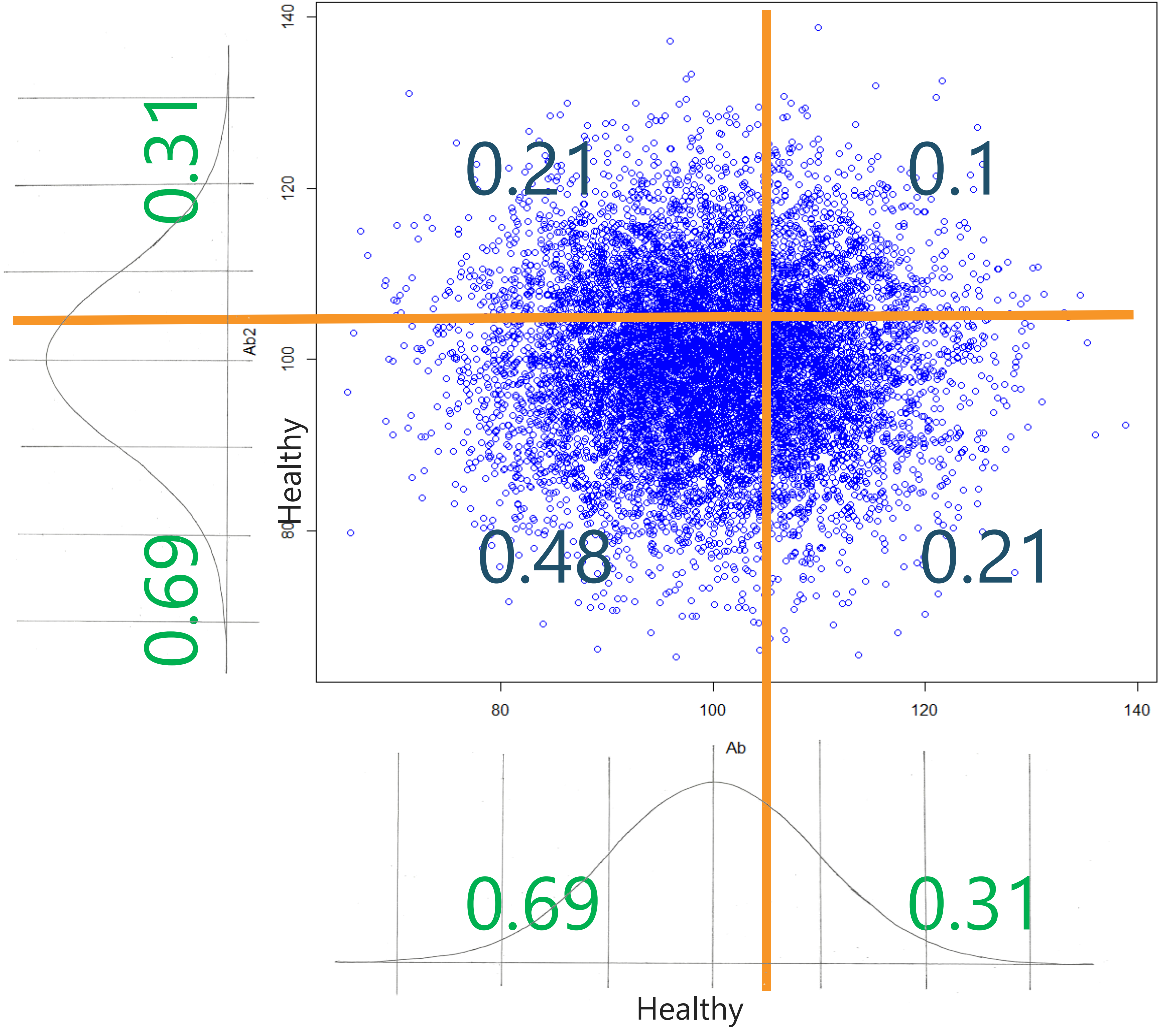
More about efficiency

The effect is easier if we increase the number of patients to 10,00. Note I am plotting the healthy patients last so that they overlay the infected. But that is OK, we are estimating the healthy ones.



More about efficiency

Even easier if we lose the infected patients.



Constrained clustering

Data points can be related in two ways - must-link and cannot-link.

Points related by a must-link have to be in the same cluster, those related by cannot-link must not be in the same cluster. Constrained clustering algorithms understand these relationships and are used to look for clusters.

Constrained clustering

Clearly sets of data can be constructed that are impossible to cluster.

A – must-link – B

B – cannot-link – C

C- must-link - A

Some constrained clustering algorithms abort when presented with this data, others find the minimum constraint violation.

Clustering algorithms

Pages in category "Data clustering algorithms"

The following 36 pages are in this category, out of 36 total. This list may not reflect recent changes (learn more).

A

- Affinity propagation

B

- Basic sequential algorithmic scheme
- Binarization of consensus partition matrices
- BIRCH

C

- Canopy clustering algorithm
- Cluster-weighted modeling
- Cobweb (clustering)
- Complete-linkage clustering
- Constrained clustering
- CURE data clustering algorithm

D

- Data stream clustering
- DBSCAN

E

- Expectation–maximization algorithm

F

- FLAME clustering
- Fuzzy clustering

H

- Hierarchical clustering

I

- Information bottleneck method

K

- K q-flats
- K-means clustering
- K-means++
- K-medians clustering
- K-medoids
- K-SVD

L

- Linde–Buzo–Gray algorithm

M

- Mean shift

N

- Nearest-neighbor chain algorithm
- Neighbor joining

O

- OPTICS algorithm

P

- Pitman–Yor process

S

- Self-organizing map
- Single-linkage clustering
- Spectral clustering
- SUBCLU

U

- UPGMA

W

- Ward's method
- WPGMA

So, what is an algorithm?????

Is it “clustering” or “K means”?

Both.

So, let’s talk about K means as a specific example of a clustering algorithm.

K means clustering

Given a set of observations ($\mathbf{x}_1$, $\mathbf{x}_2$, …, $\mathbf{x}_n$), where each observation is a *d*-dimensional real vector, *k*-means clustering aims to partition the *n* observations into *k* ($\leq n$) sets $\mathbf{S} = \{S_1, S_2, \ldots, S_k\}$ so as to minimize the inter-cluster sum of squares (ICSS) (sum of distance functions of each point in the cluster to the K centre).

https://en.wikipedia.org/wiki/K-means_clustering

Now, I understand that this explanation won't work for everyone so:

K means clustering

in other words, its objective is to find:

$$\underset{\mathbf{S}}{\arg\min} \sum_{i=1}^{k} \sum_{\mathbf{x} \in S_i} \left\| \mathbf{x} - \boldsymbol{\mu}_i \right\|^2$$

where $\boldsymbol{\mu}_i$ is the mean of points in S_i.

https://en.wikipedia.org/wiki/K-means_clustering

K means clustering

Time for some board work

(Except I guess there isn't going to be a board)

Time for some arm waving

K means clustering

```
iris
newiris <- iris
newiris$Species <- NULL
newiris
kc <- kmeans(newiris, 3)
kc
table(iris$Species, kc$cluster)
plot(newiris[c("Sepal.Length", "Sepal.Width")], col=kc$cluster)
points(kc$centers[,c("Sepal.Length", "Sepal.Width")], col=1:3,
pch=8, cex=2)
```

http://www.rdatamining.com/examples/kmeans-clustering

Author Yanchang Zhao

K means clustering

K means is an **heuristic** algorithm which means that…….

K means clustering

K means is an heuristic algorithm which means that the end result is not known as each step proceeds. (Think about playing chess; all chess playing is heuristic.) K means is also **stochastic**.

K means clustering

There is no guarantee that one run will actually find the optimal solution; it can depend on where the initial points are placed.

So the algorithm is often run multiple times. However it can be slow to converge and there are known sets of points (even in two dimensions) that are particularly troublesome.

However, in practice, it is a very good algorithm.

Decision Trees

History

- J Ross Quinlan introduced a decision tree algorithm called ID3

Decision Trees

Used to assign each case to one of several categories

Explains the classification – which variables are used

Decision trees are easy to understand

Often used for the prediction of values of the variables explained

Decision Trees

The process is essentially one of recursive partitioning

Decision Trees

Decision trees tries all possible splits using all possible values of each input attribute

Chooses the most effective split as the first

Several ways of measuring the efficacy of the split – one of which is frequency distribution and another is entropy – look up entropy in this context

Decision Trees

Are all splits binary?

No – eye colour

But any algorithm that can perform a binary split can also split on multiple factors

Decision Trees

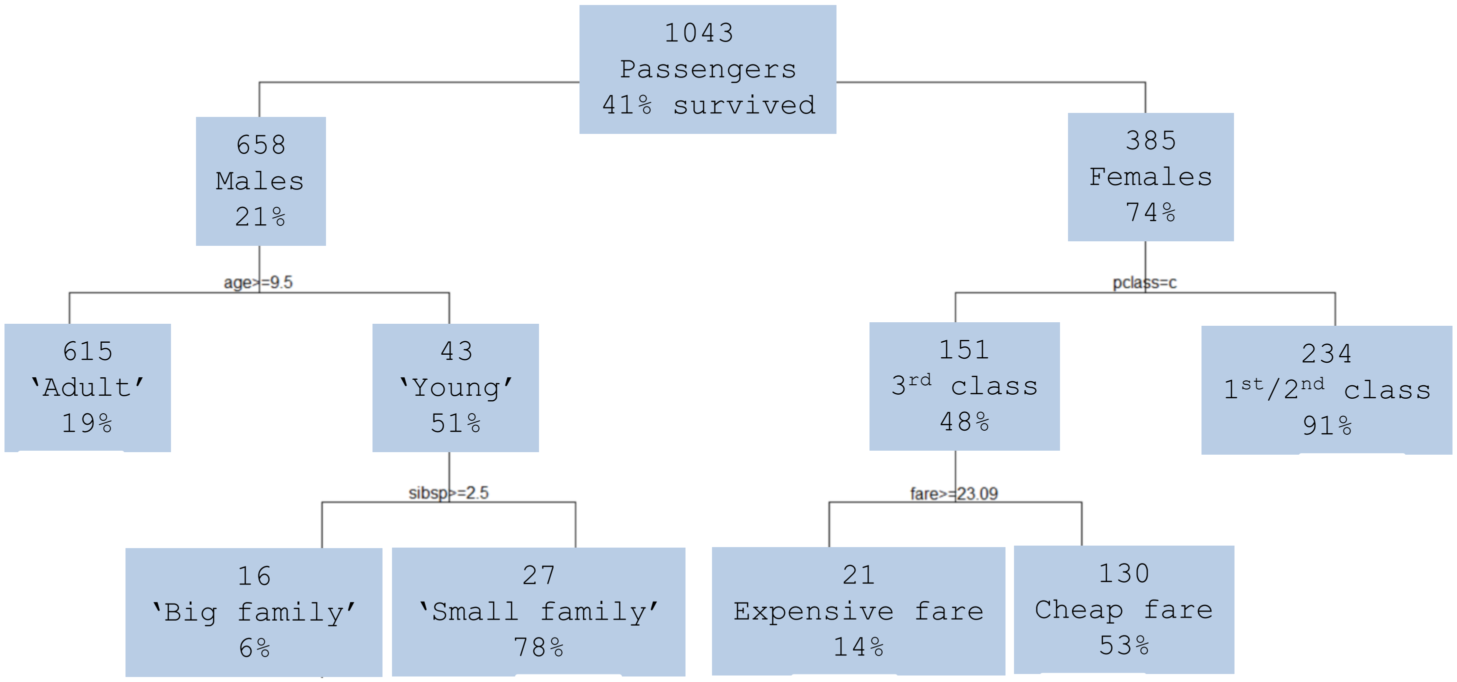
Algorithm design......

- Avoid splits leading to one member
- Watch high cardinality predictors
  - What happens if customer name is used as a predictor?
  - What happens with post code?
    - Should you look for hierarchies?

Decision Trees

```
#Libary
library(rpart)
library(readr)

titanic <- read_csv("~/2018/QA/PWC/titanicdecisiontrees.csv")

tree <- rpart(survived ~ pclass + sex + age + sibsp + parch + fare + embarked,
            data=titanic,
            method="class")

plot(tree, uniform=TRUE,
   main="Titanic")
text(tree, use.n=TRUE, all=TRUE)

str(tree)
```

Titanic



1043
Passengers
41% survived
658
Males
21%
385
Females
74%
age>=9.5
pclass=c
615
'Adult'
19%
43
'Young'
51%
151
3rd class
48%
234
1st/2nd class
91%
sibsp>=2.5
fare>=23.09
16
'Big family'
6%
27
'Small family'
78%
21
Expensive fare
14%
130
Cheap fare
53%

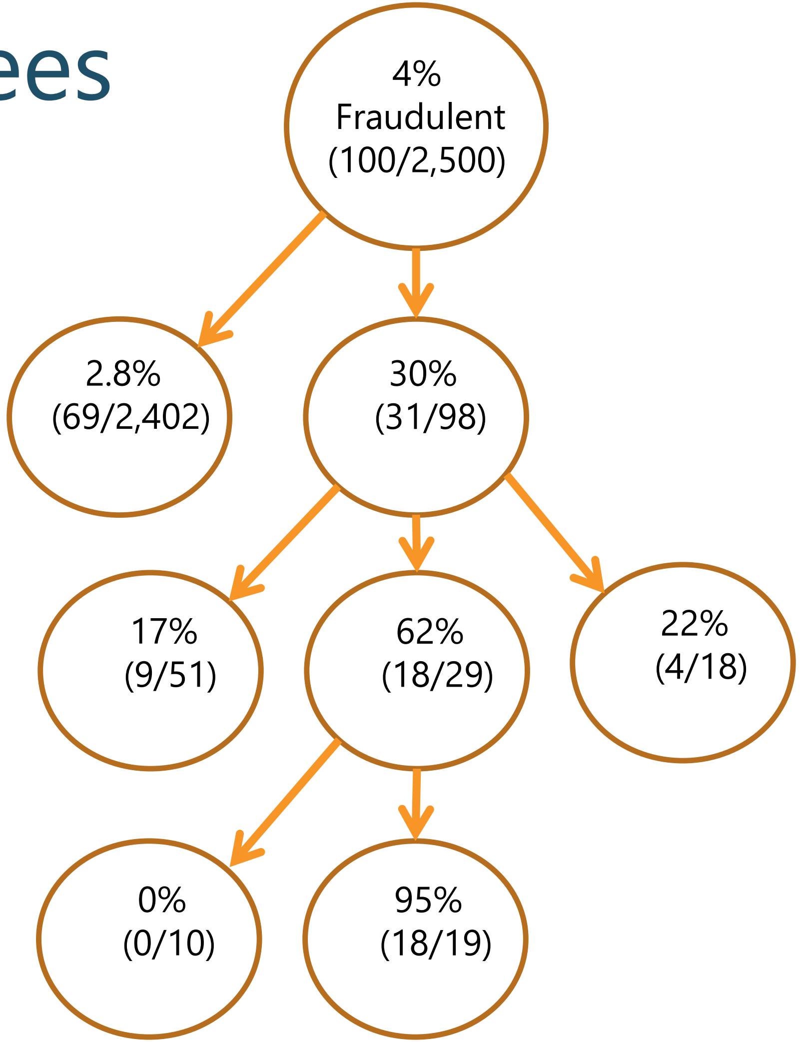
Decision Trees

Input is a table, in this case about insurance claims.

ID	Local	People	Injury	Time	Police	Fraud
1	Yes	2	No	17:35	No	No
2	No	2	No	12:06	Yes	No
3	Yes	2	No	16:45	Yes	No
4	No	4	Yes	19:45	No	Yes

Decision trees tries all possible splits using all possible values of each input attribute.

Decision Trees



Consolidation

Algorithms are ways of solving problems, not the code itself (that is simply an implementation issue).

They are often underpinned by complex maths/statistics. But we don’t need to understand this in order to use them.

BUT choosing the correct algorithm is crucial. Many people don’t. So we do need to understand (in plain English) how algorithms work, what they are trying to achieve.

Consolidation

Many algorithms been prewritten and form the “Data Mining Hall Of Fame”. These are often seen as the “Data Science” algorithms and, indeed, the “Machine Learning” algorithms.

But very often, in my experience, these are simply a **framework** upon which to start building.

Consolidation

Often we use the existing algorithms (e.g. clustering) to create a data model. And then use that to build further models, for example RFI Recency, Frequency and Intensity.

Consolidation

I have shown 2 dimensions in my examples. But often we have far more. The maths get a little more complex but, oddly, not exponentially so for each dimension.

The data we use (the xy coordinates I have shown so far) can be called vectors. This is a good way to think about them as we look at more complex algorithms.

We have looked at linear regression, clustering and decision trees. Let's look at some more algorithms.

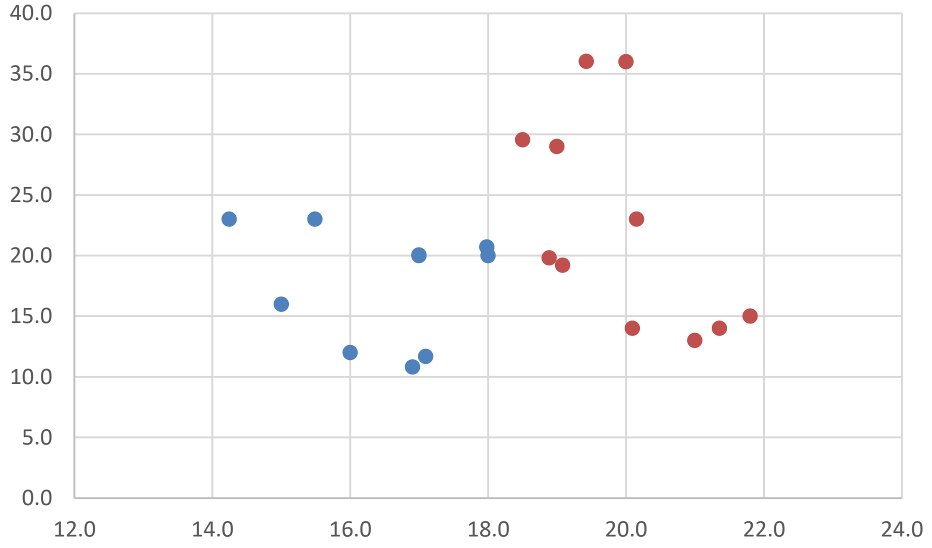
Quick overview of Techniques

Data mining techniques

- Clustering
- Classification
- Decision trees
- Regression
- KNN
- Segmentation
- Association
- Sequence analysis
- Neural nets

KNN (K Nearest Neighbour)

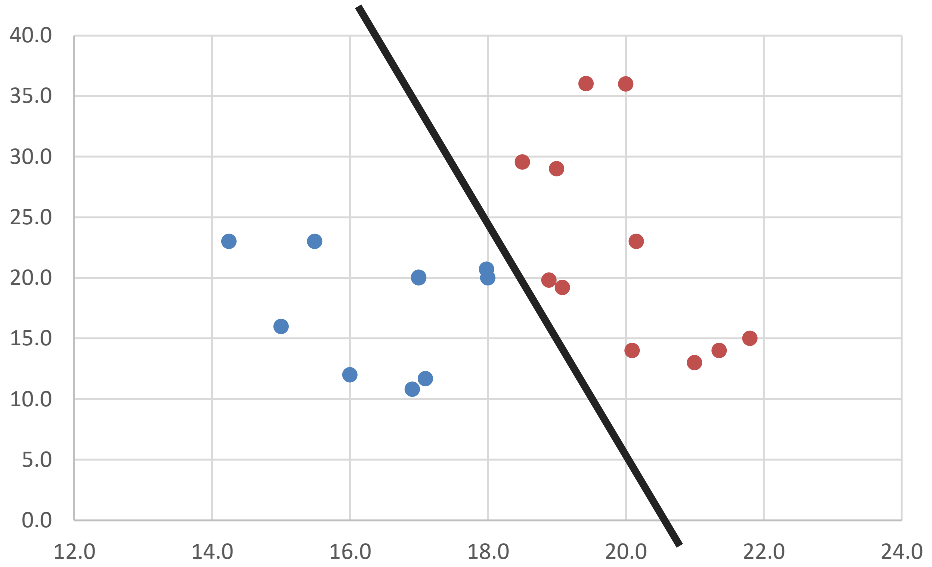
Suppose that we have some two dimensional data that also has a binary classification (such as male - female).



KNN (K Nearest Neighbour)

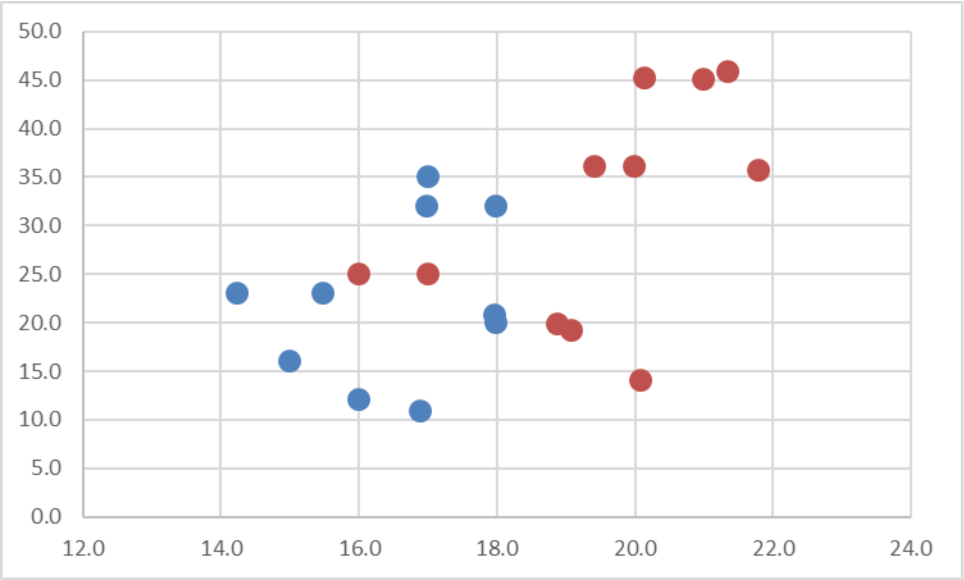
We want to use it for prediction (given these X,Y values, what is the gender?).

We might draw a line:



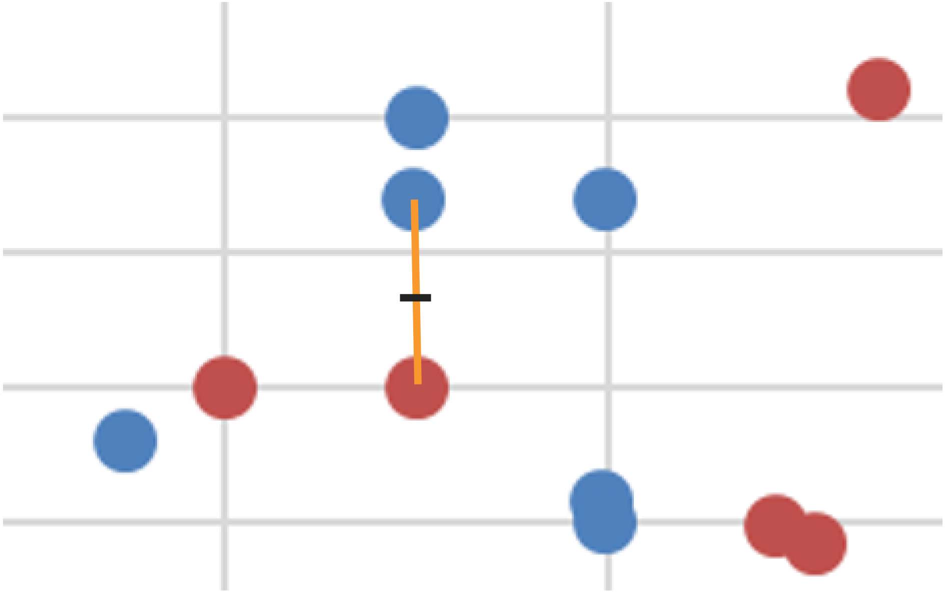
KNN (K Nearest Neighbour)

But suppose that the data is less cooperative. Let's zoom in:



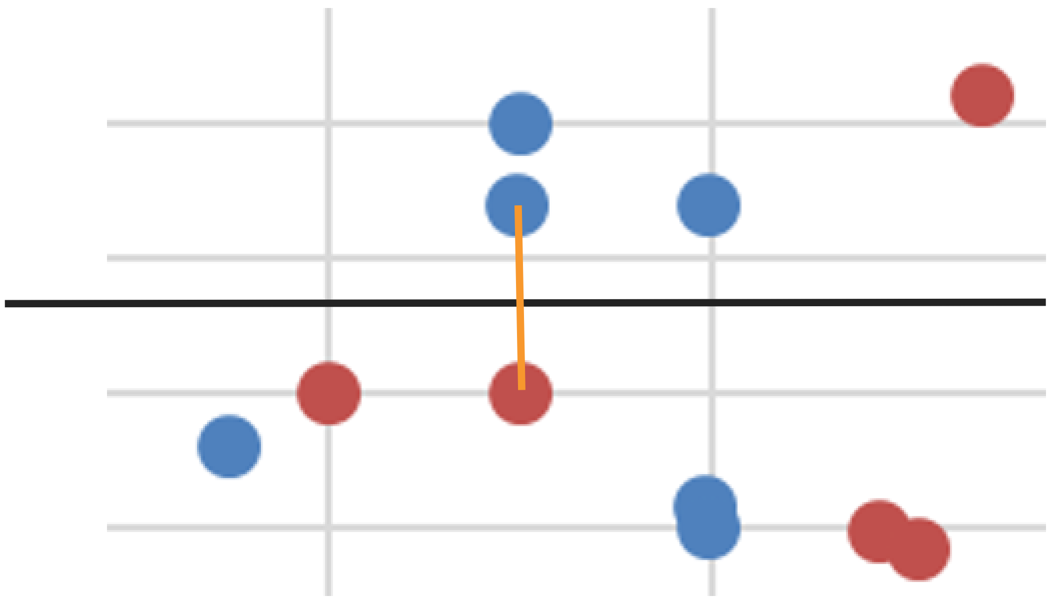
KNN (K Nearest Neighbour)

Take the direct line between two points, find the mid point



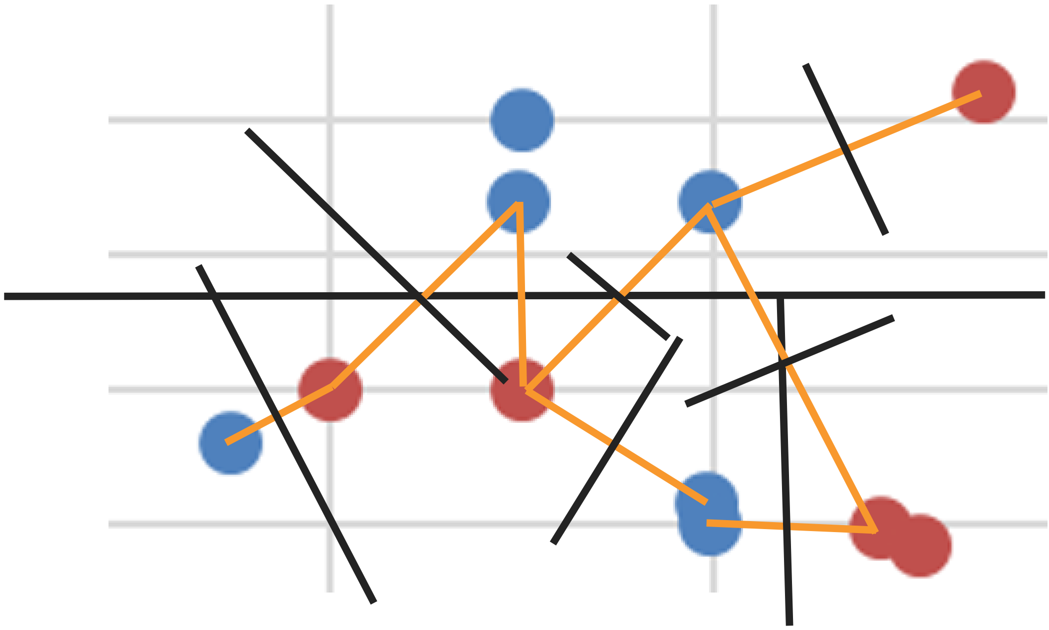
KNN (K Nearest Neighbour)

Extrapolate at right angles



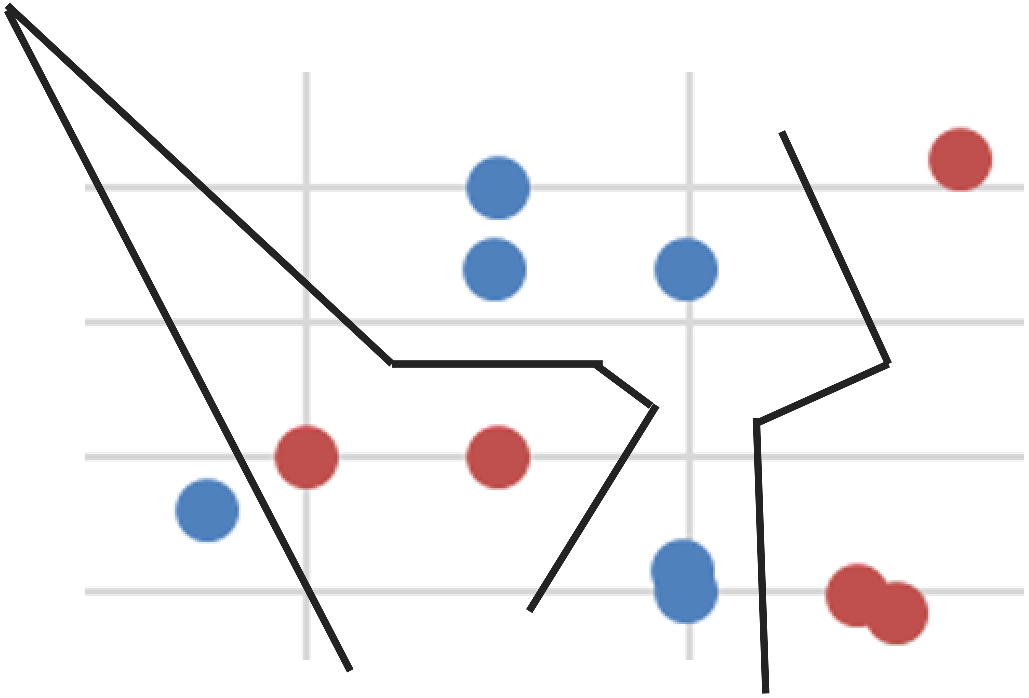
KNN (K Nearest Neighbour)

repeat



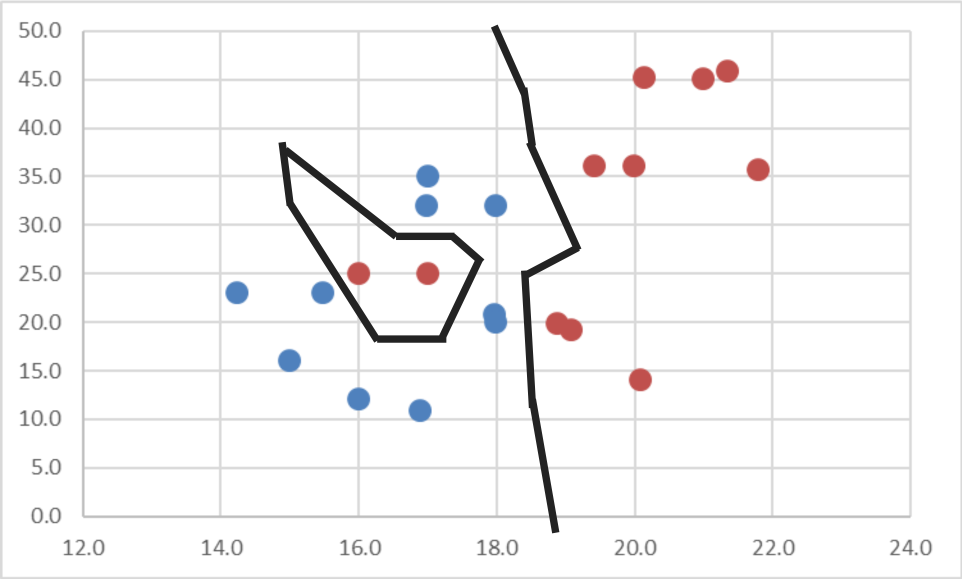
KNN (K Nearest Neighbour)

consolidate



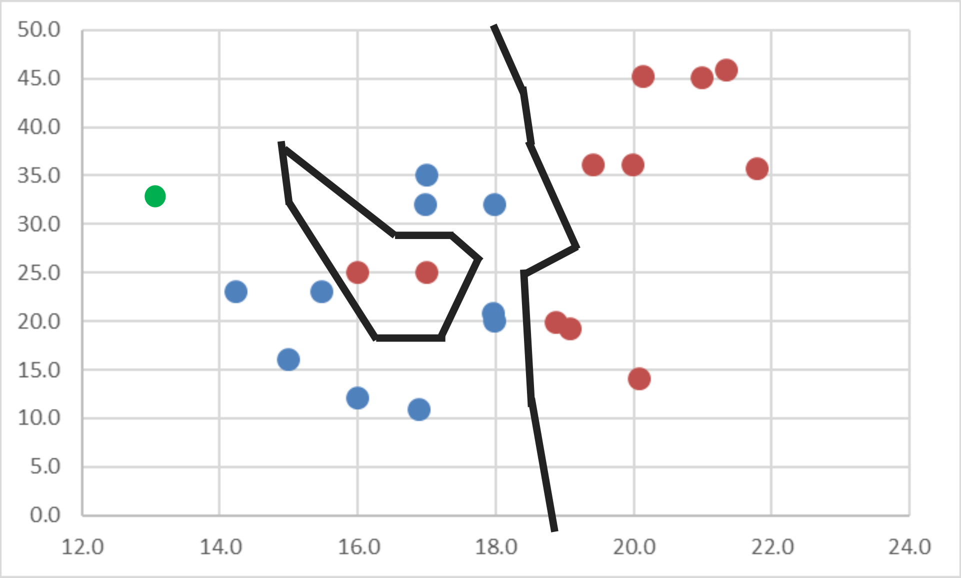
KNN (K Nearest Neighbour)

Zoom out



KNN (K Nearest Neighbour)

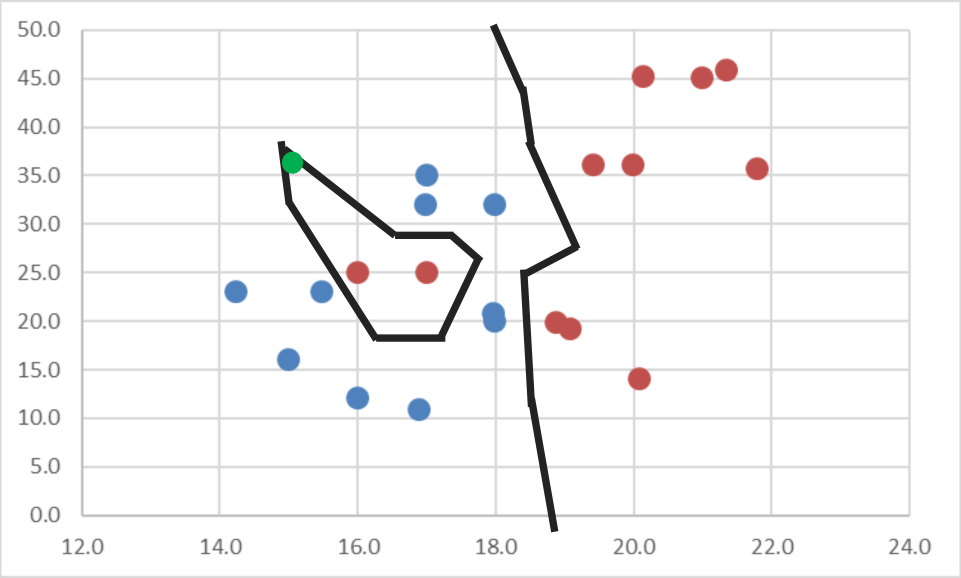
So the unknown data point would be classified as Blue.



KNN (K Nearest Neighbour)

Here it would be classified as Red.

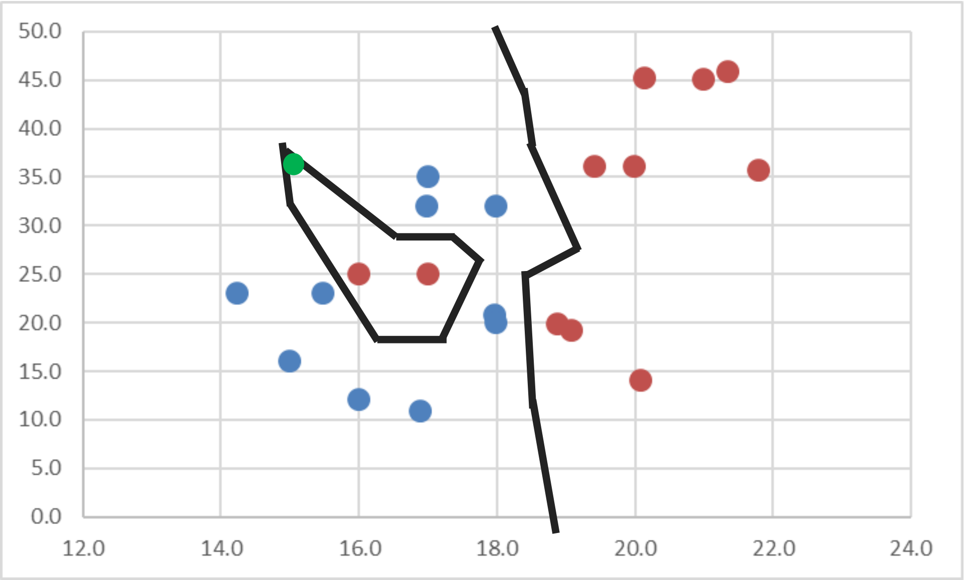
Do we actually like this choice?



KNN (K Nearest Neighbour)

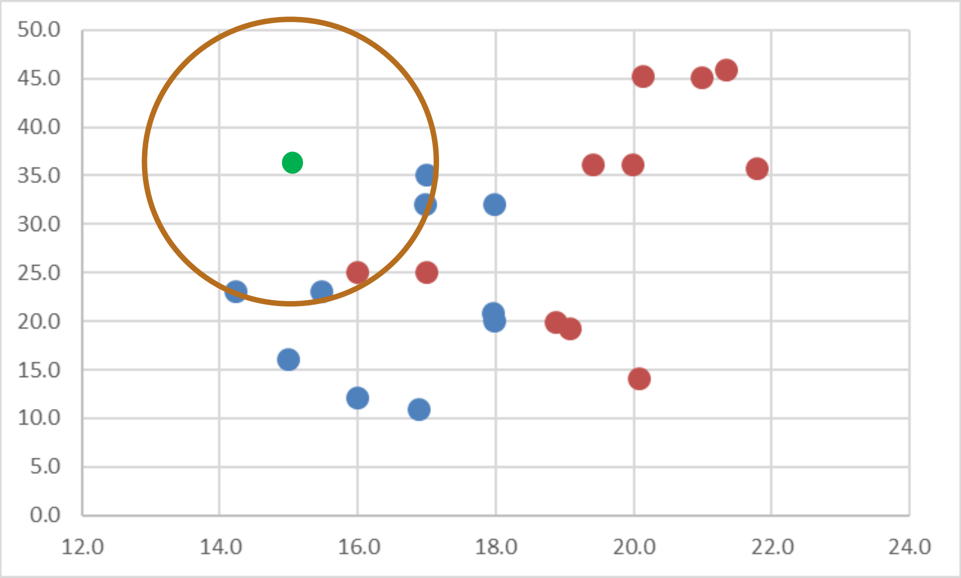
That's OK, we have options!

So far we have been using one neighbour.



KNN (K Nearest Neighbour)

What if we choose three? Answer: this green point would be classified as blue.



KNN (K Nearest Neighbour)

The K is simply the number of nearest neighbours you want to use. What is the correct number?

I don't know but it had better be odd.

Support Vector Machines

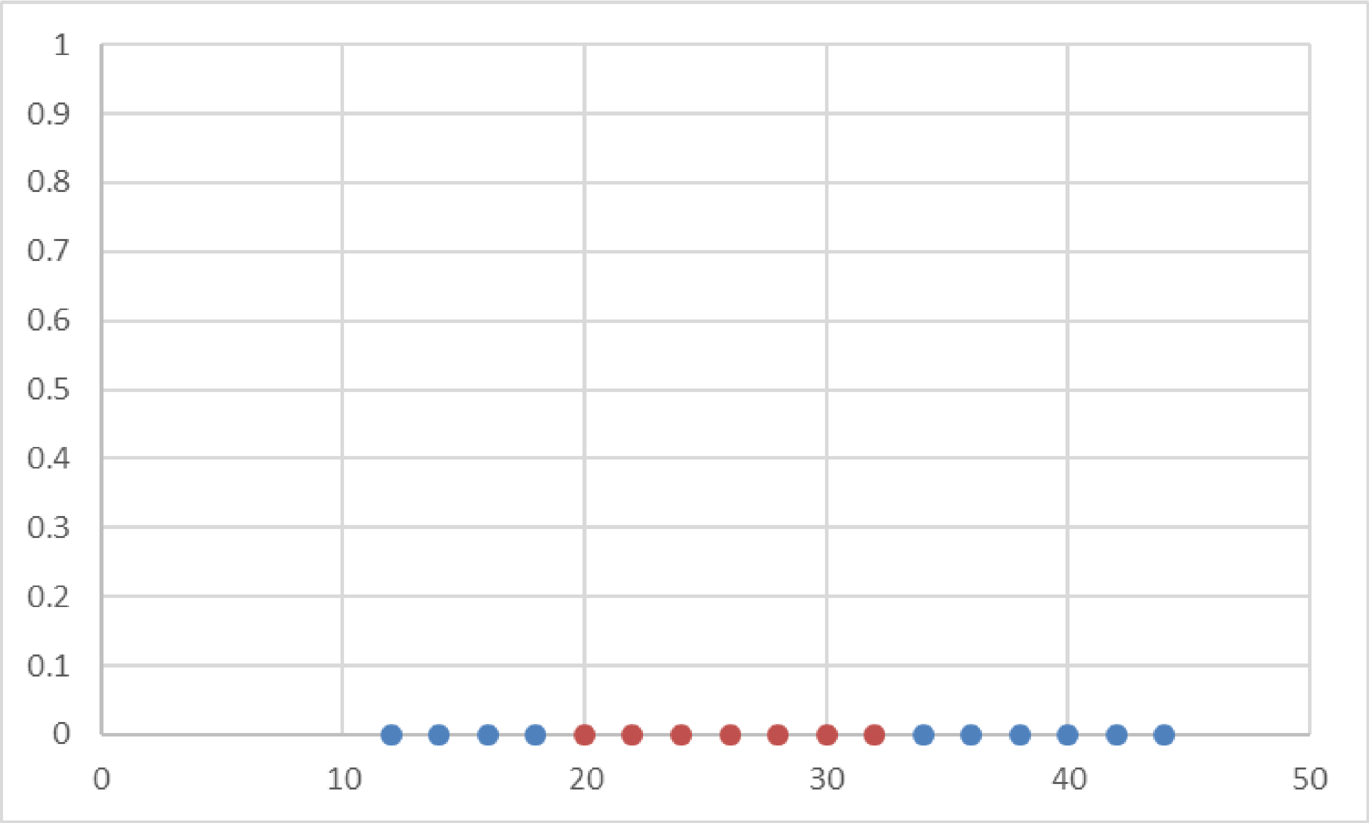
These help to separate out data. More accurately, they allow us to separate data that is somewhat interwoven more easily.

Imagine that we have some vectors that describe some data.

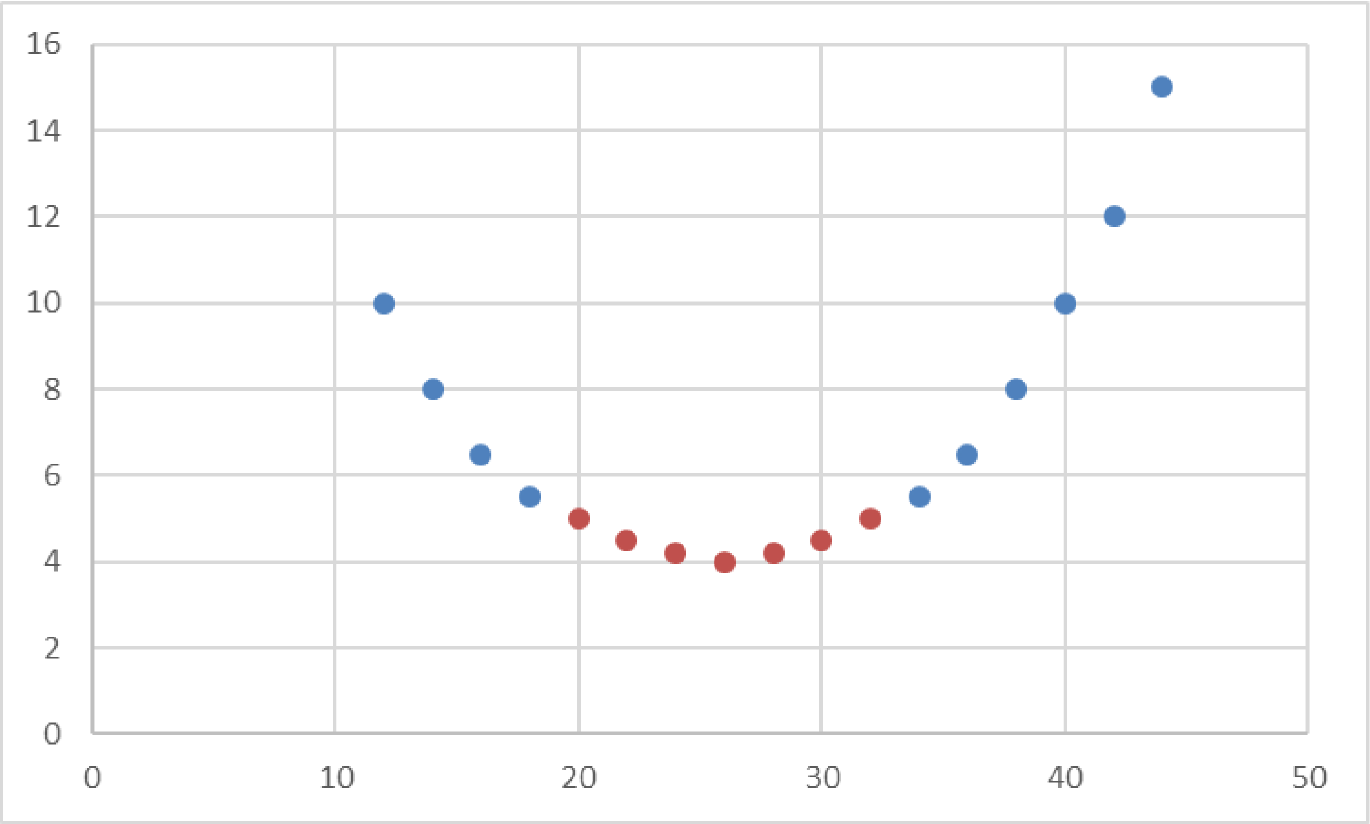
Support Vector Machines

SVMs projects your data into higher dimensions in order to create/derive/learn a hyperplane which can separate your data into two classes.

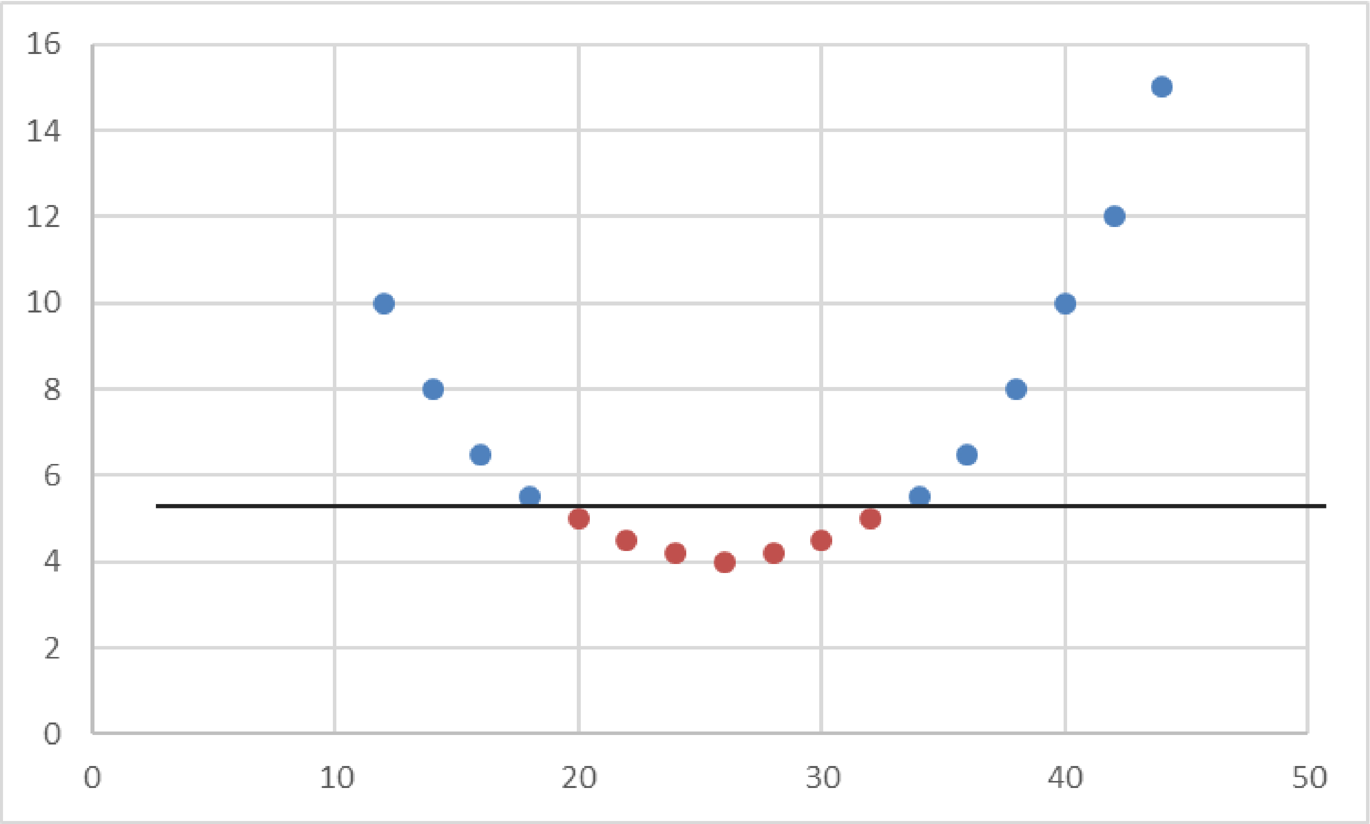
Support Vector Machines



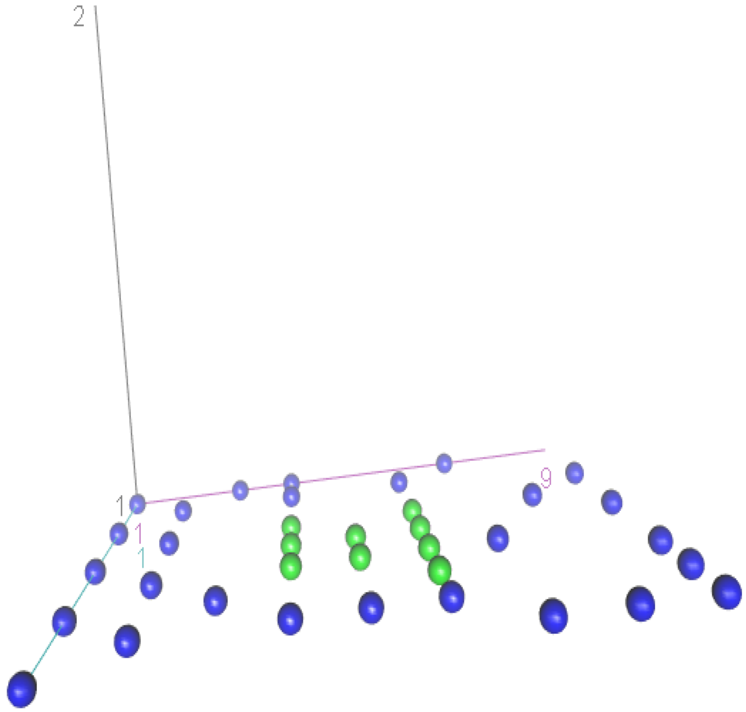
Support Vector Machines



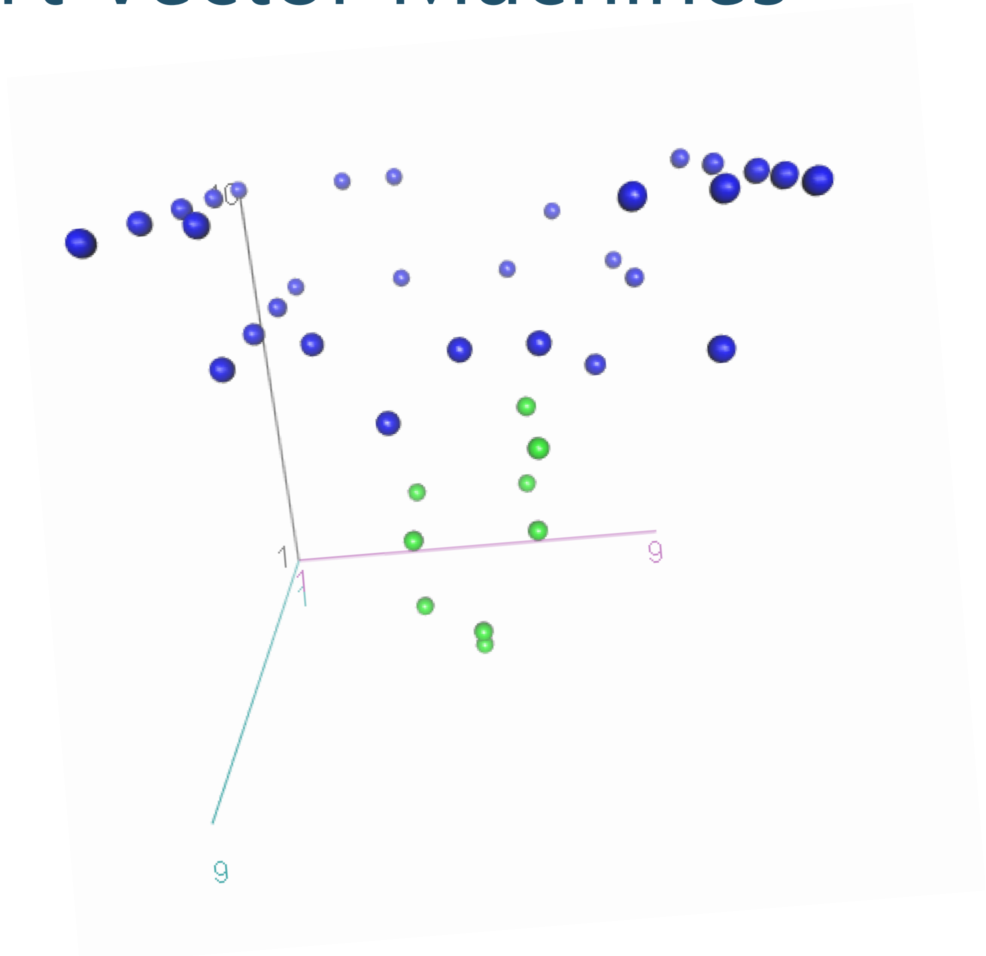
Support Vector Machines



Support Vector Machines



Support Vector Machines



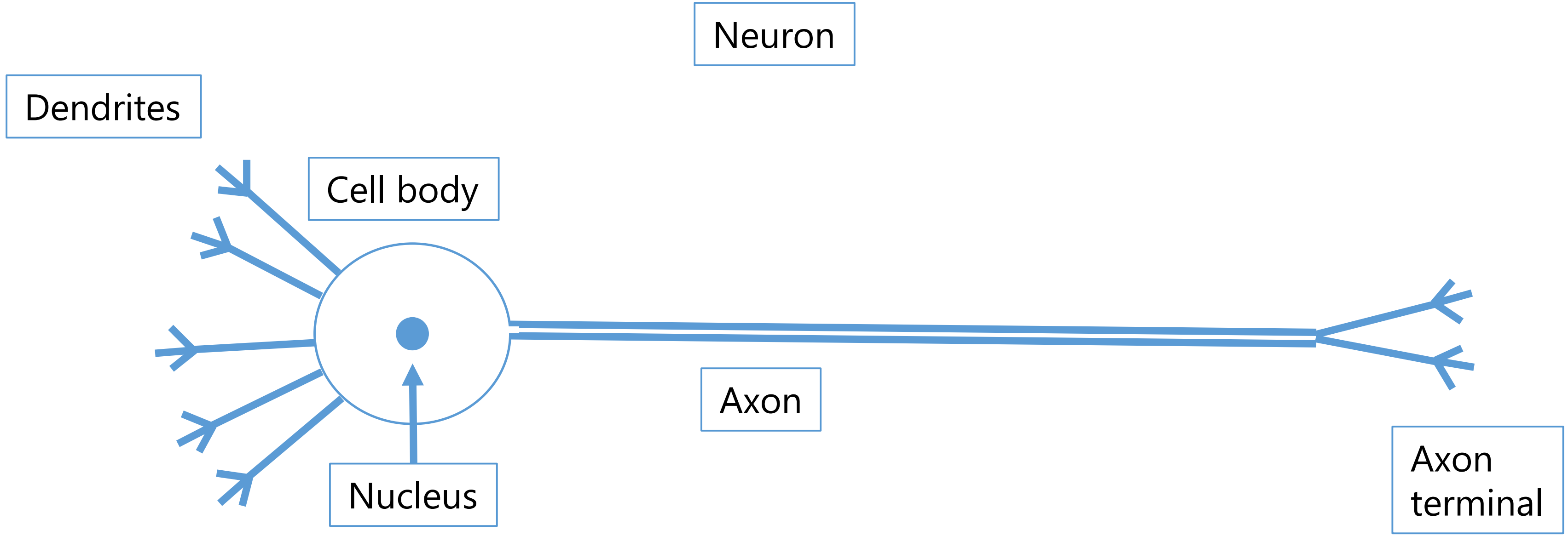
Support Vector Machines

But why are they called SMV?

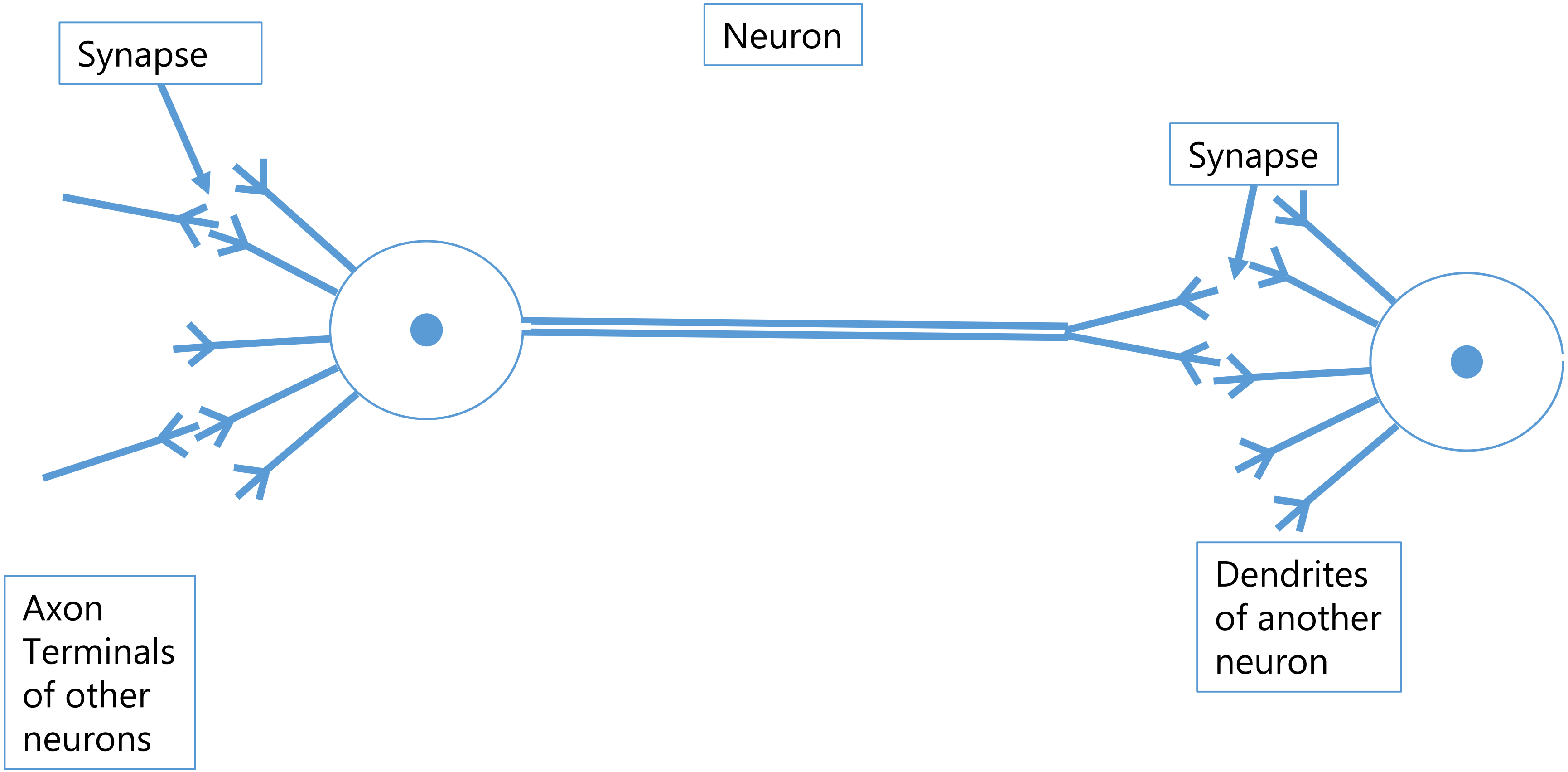
The hyperplane fits between the two classes but there is always a ‘nearest’ red dot and blue dot. These balls are considered to be the vectors that support the hyperplane. Their distance to the hyperplane is called the **margin.** The SMV keeps the margins identical and (obviously) tried to maximise them.

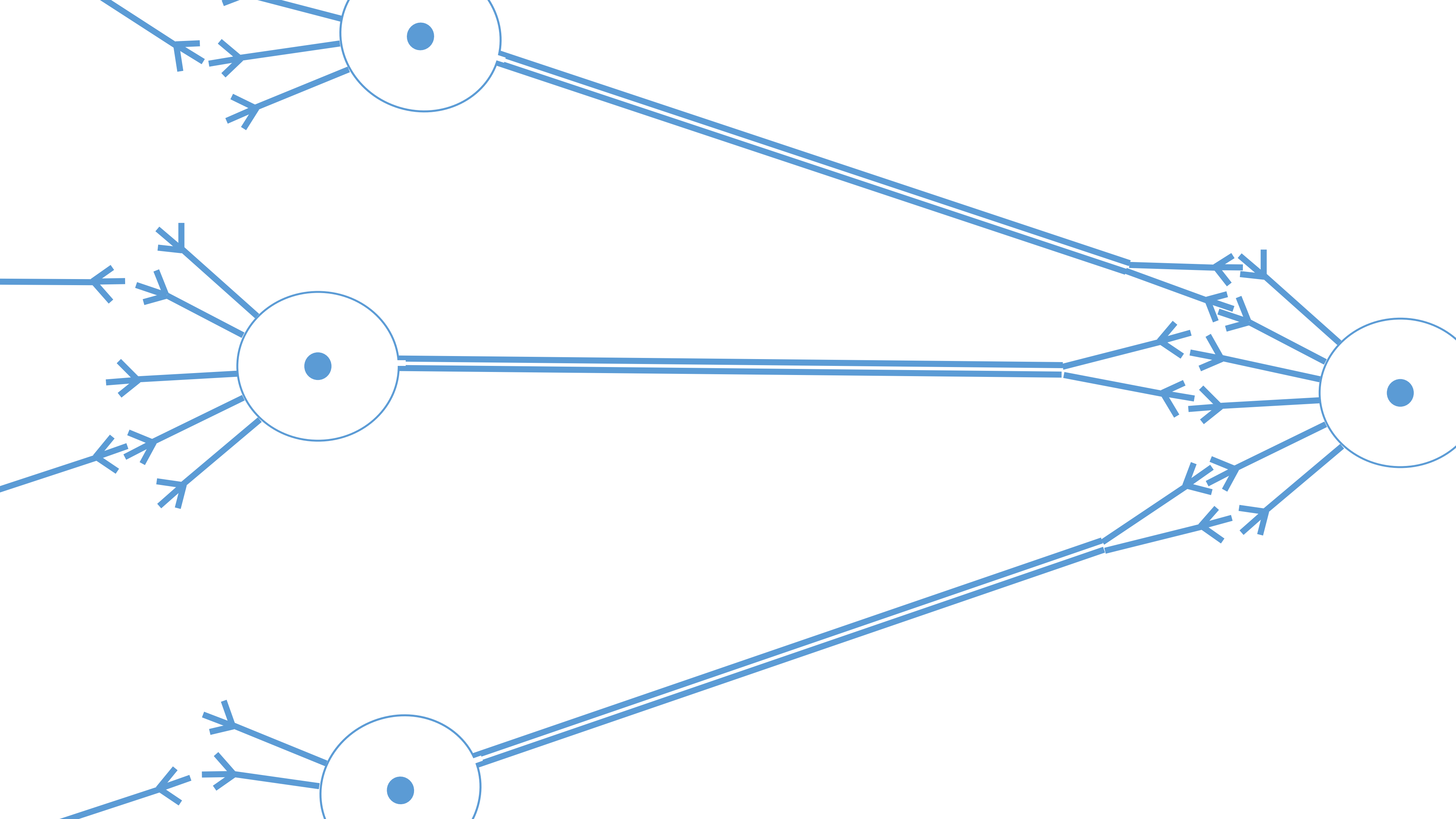
Neural Nets

Real Neurons



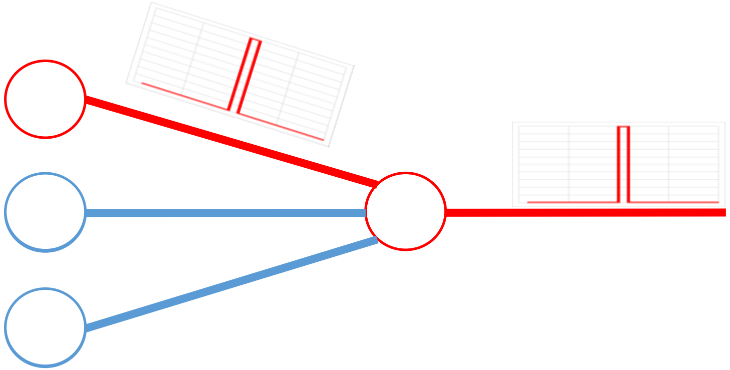
Real Neurons





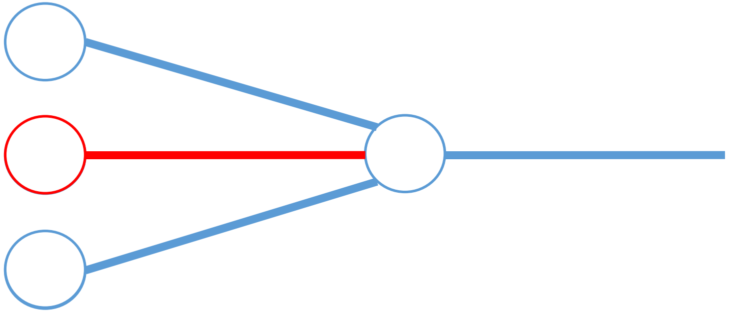
Real Neurons

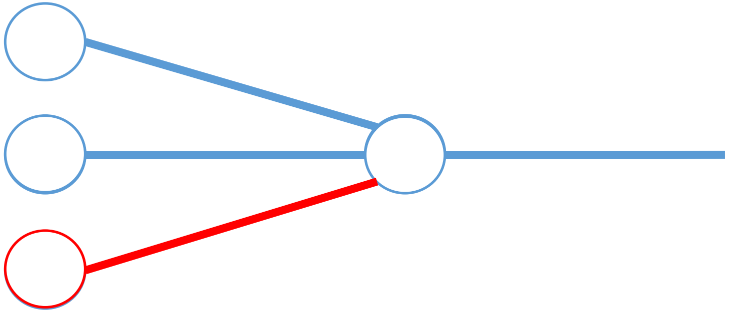
- The output from each neuron is binary, either it fires or it doesn't

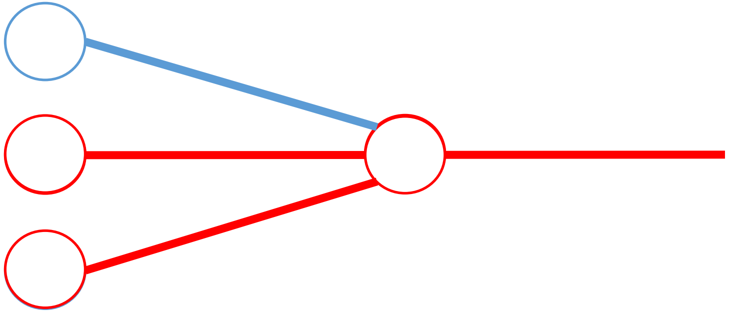


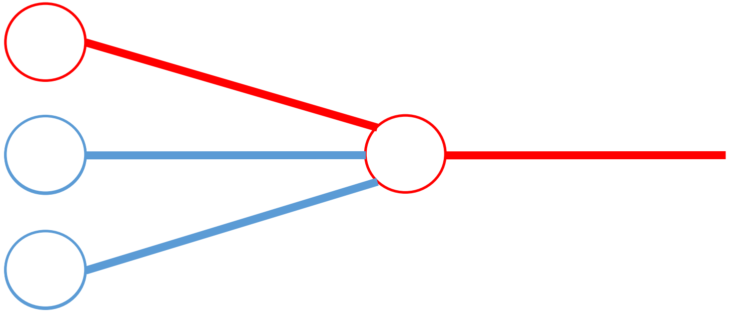
Real Neurons

- The input to the target neuron can have many states. Perhaps the first input neuron is triggered, or the second, or the first and the third together and so on
- Some input combinations will fire the target neuron, others won't. So, while the output of each neuron is binary, the input isn't because many neurons can act as input to one neuron

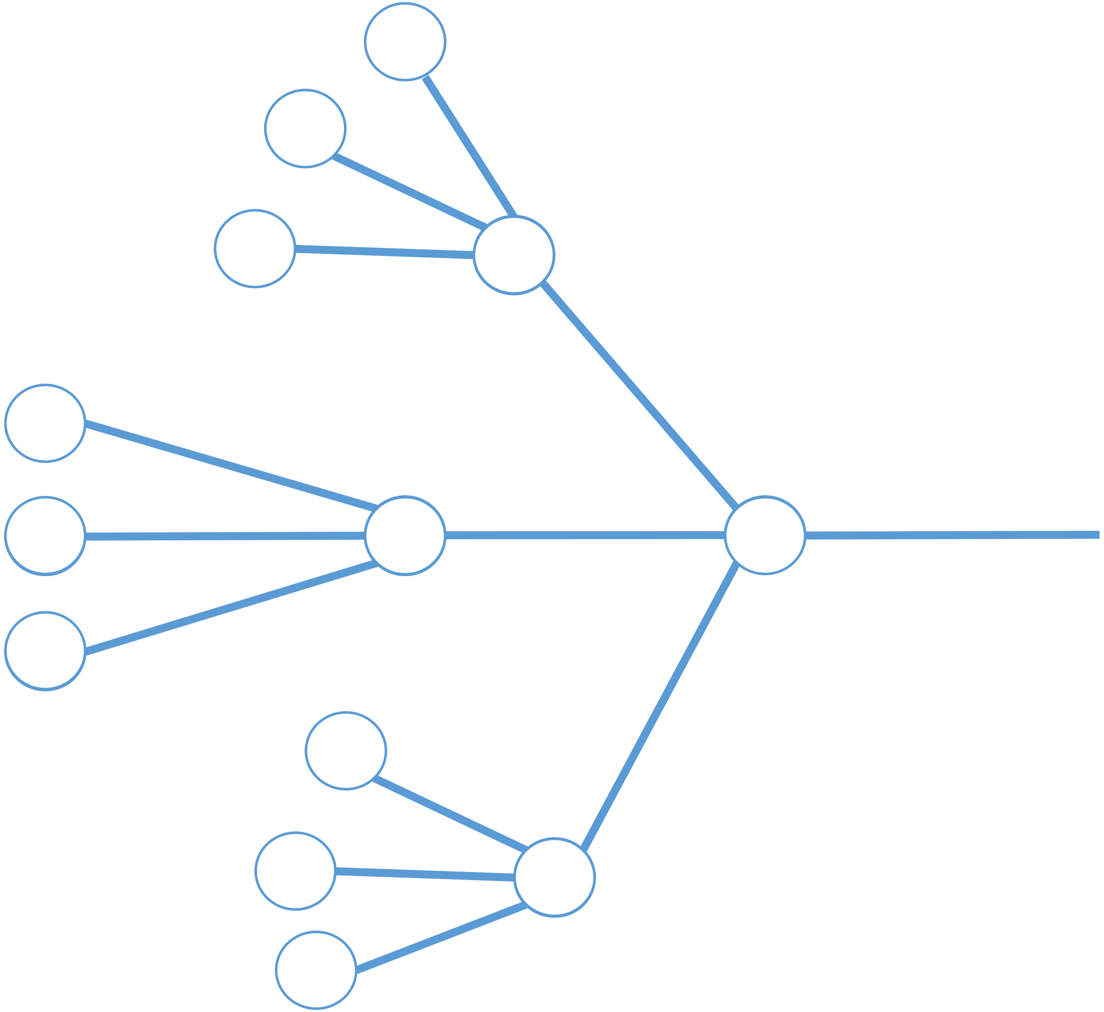




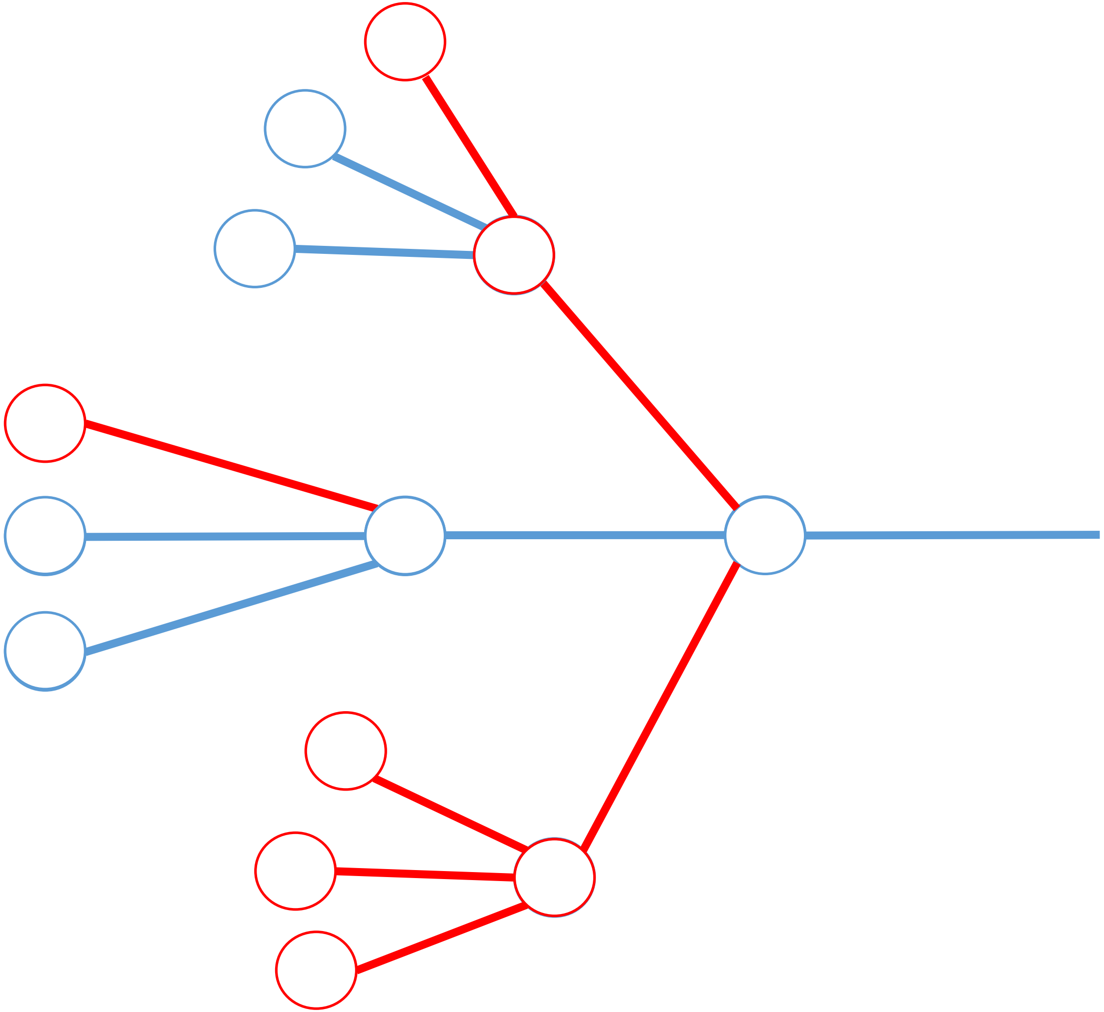




Real Neurons

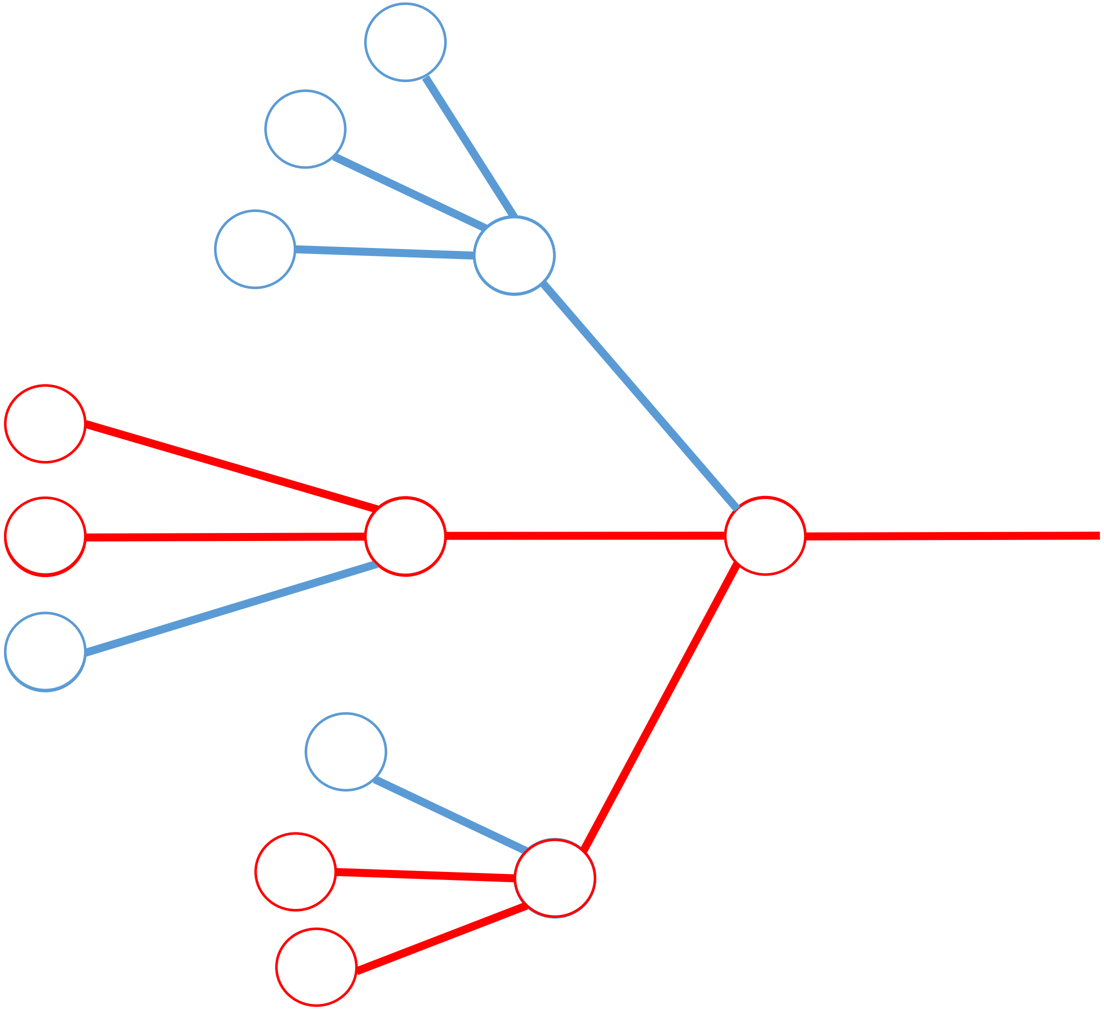


Real Neurons



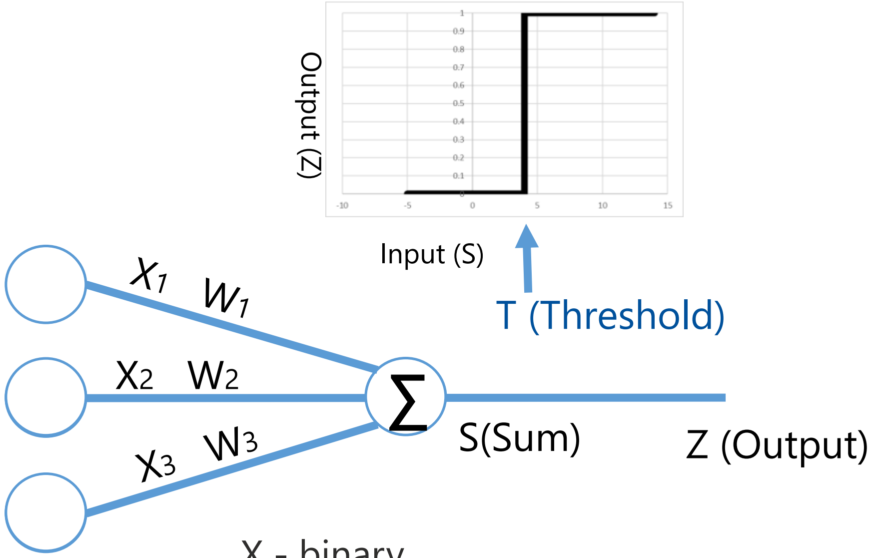
Real Neurons

If we can characterise this behaviour in some way then we can adapt it for use in ANN



Modelling real Neurons

- Each input neuron can provide an input (X) which is binary (it is either 0 or 1), but they have differing effects. That is, some can cause the target neuron to fire on their own, others can't
- We can add a "weight" (W) to model this. The higher the value of the weight, the greater the influence the input neuron has on the chance of the target firing
- These inputs, times their respective weights, are collected together in some way. For the sake of simplicity, we will sum them
- The summed value (S) either does, or does not, exceed the threshold (T). If it does, the neuron fires with output Z which is, again, binary



X - binary

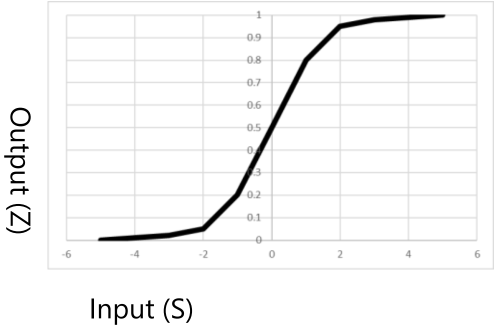
W - continuous

S - continuous

T - continuous

Z - binary

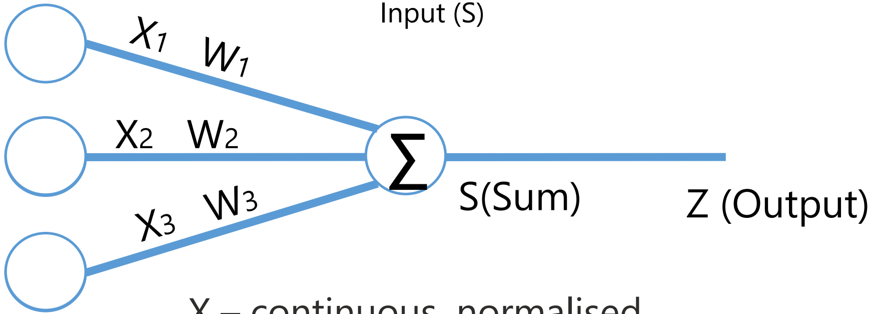
Neurons in an Artificial Neural Network



We set the threshold to zero and render it non-binary. It is smoothed (typically as a sigmoid function but others are used)

One reason we make these changes is so that we can make use of backpropagation. (Paul Werbos 1974). We will look at why we want to use backpropagation later

So both output and input are now continuous, in fact, all the values are now continuous



X – continuous, normalised

W - continuous

S - continuous

T – nominally set to zero

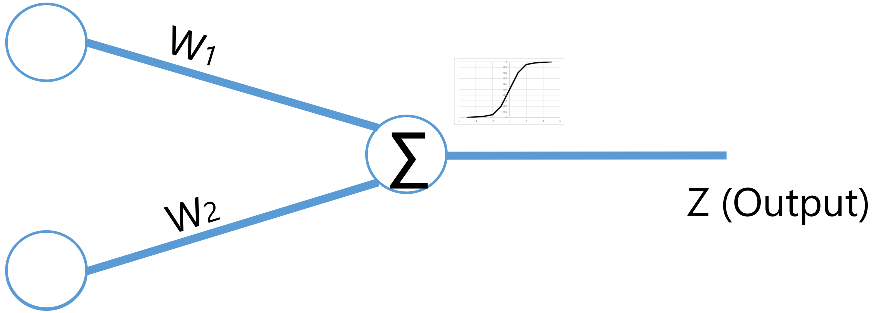
Z – continuous, normalised

Neural Networks

Let’s look at a very simple neural network consisting of just two neurons. This is clearly unrealistic but it shows the principles very well. We will use a classic set of data for this, the so-called Iris Data

We’ll focus on just the first two dimensions, sepal length and width.

Sepal Length	Sepal Width	Petal Length	Petal Width	Species
5.1	3.5	1.4	0.2	setosa
4.9	3	1.4	0.2	setosa
4.7	3.2	1.3	0.2	setosa
4.6	3.1	1.5	0.2	setosa
7	3.2	4.7	1.4	versicolor
6.4	3.2	4.5	1.5	versicolor
6.9	3.1	4.9	1.5	versicolor
5.5	2.3	4	1.3	versicolor



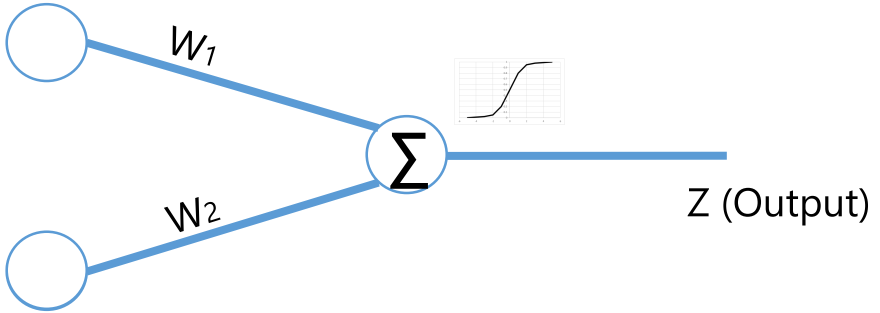
Neural Networks

Let's look at a very simple neural network consisting of just two neurons. This is clearly unrealistic but it shows the principles very well. We will use a classic set of data for this, the so-called Iris Data

We'll focus on just the first two dimensions, sepal length and width. First we normalise the data

Sepal Length	Sepal Width	Petal Length	Petal Width	Species
5.1	3.5	1.4	0.2	setosa
4.9	3	1.4	0.2	setosa
4.7	3.2	1.3	0.2	setosa
4.6	3.1	1.5	0.2	setosa
7	3.2	4.7	1.4	versicolor
6.4	3.2	4.5	1.5	versicolor
6.9	3.1	4.9	1.5	versicolor
5.5	2.3	4	1.3	versicolor

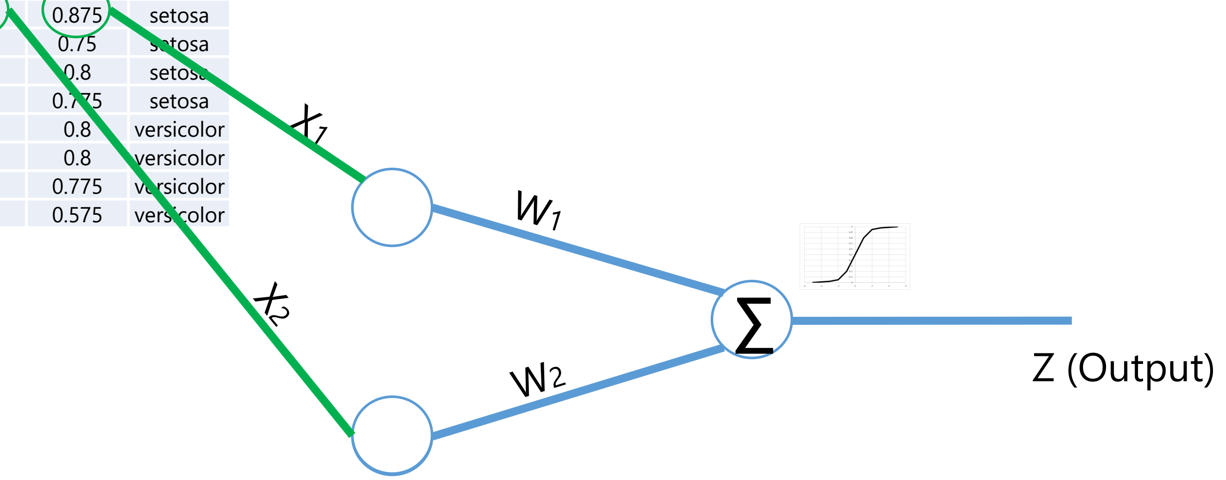
Sepal Length	Sepal Width	Species
0.51	0.875	setosa
0.49	0.75	setosa
0.47	0.8	setosa
0.46	0.775	setosa
0.7	0.8	versicolor
0.64	0.8	versicolor
0.69	0.775	versicolor
0.55	0.575	versicolor



Neural Networks

- The new values become the input values for the two input neurons

Sepal Length	Sepal Width	Species
0.51	0.875	setosa
0.49	0.75	setosa
0.47	0.8	setosa
0.46	0.775	setosa
0.7	0.8	versicolor
0.64	0.8	versicolor
0.69	0.775	versicolor
0.55	0.575	versicolor



Neural Networks

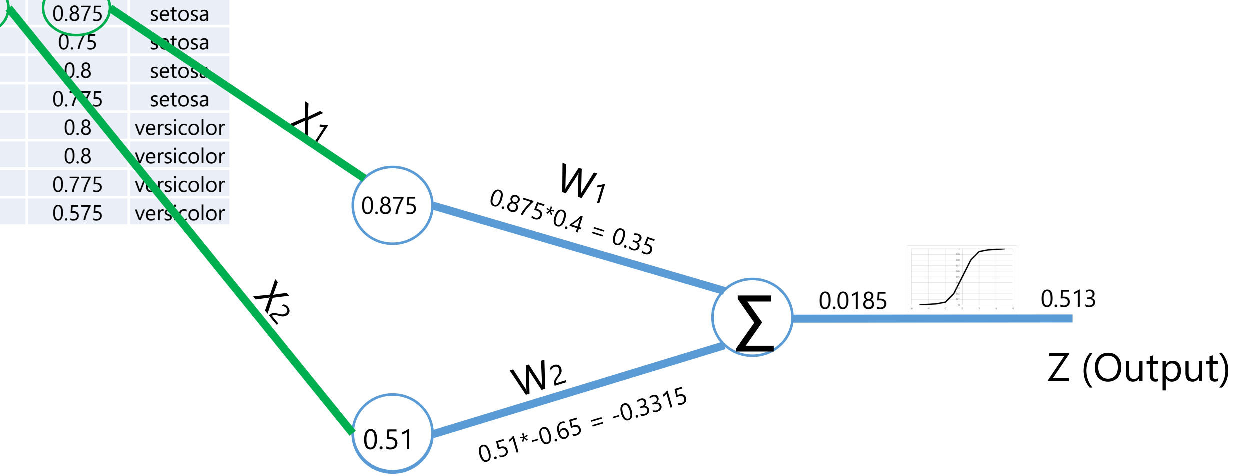
The new values become the input values for the two input neurons

The two weight (W1 and W2) are set to random values between -1 and +1

W1 = 0.4

W2 = -0.65

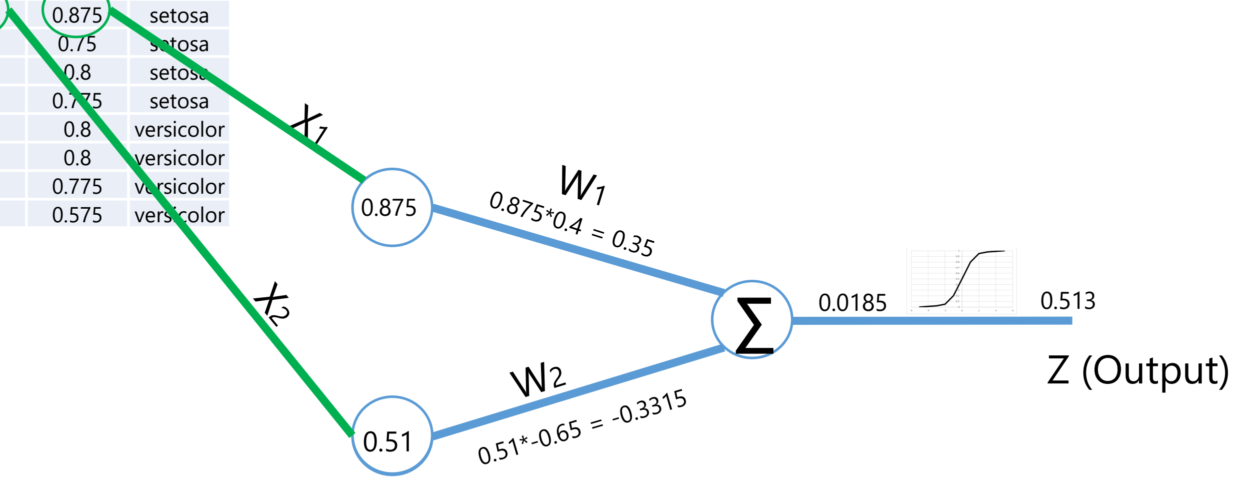
Sepal Length	Sepal Width	Species
0.51	0.875	setosa
0.49	0.75	setosa
0.47	0.8	setosa
0.46	0.775	setosa
0.7	0.8	versicolor
0.64	0.8	versicolor
0.69	0.775	versicolor
0.55	0.575	versicolor



Neural Networks

Let's say that we arbitrarily chose that we want an output value of 0 to indicate versicolor and 1.0 to indicate setosa. We can't change the input values, but we can change the weights.

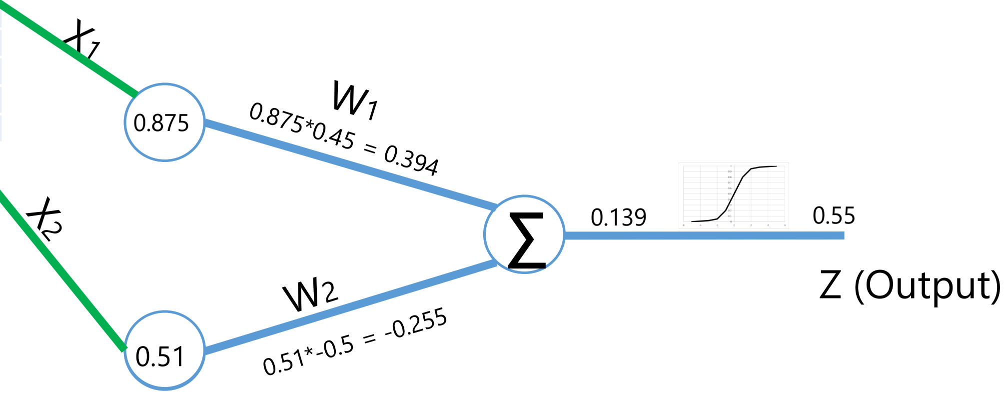
Sepal Length	Sepal Width	Species
0.51	0.875	setosa
0.49	0.75	setosa
0.47	0.8	setosa
0.46	0.775	setosa
0.7	0.8	versicolor
0.64	0.8	versicolor
0.69	0.775	versicolor
0.55	0.575	versicolor



Neural Networks

So we alter them slightly to get closer to the number we want. We go through all of the data and keep cycling through it, gradually adjusting the weights until we arrive at weight values that give us the best separation of the data

Sepal Length	Sepal Width	Species
0.51	0.875	setosa
0.49	0.75	setosa
0.47	0.8	setosa
0.46	0.775	setosa
0.7	0.8	versicolor
0.64	0.8	versicolor
0.69	0.775	versicolor
0.55	0.575	versicolor



Neural Networks

In practice, with our tiny neural net, it would be ridiculous to try to get a reasonable separation, there are simply not enough pathways. But that is OK, we can solve that problem by adding more neurons. And this is what we do in modern neural nets

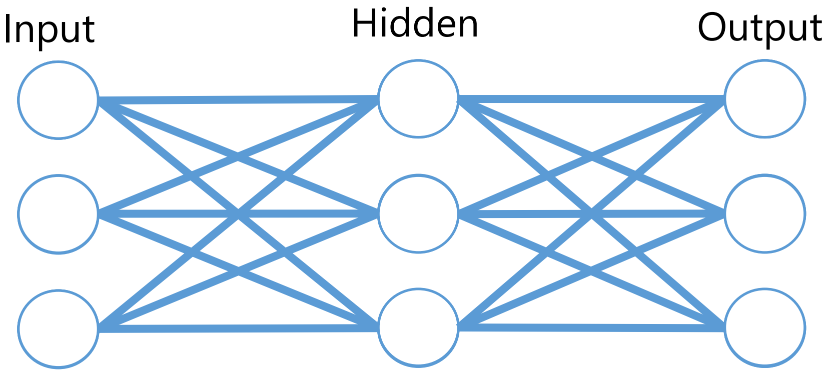
Neural Networks

Modern neural networks can be built in many ways, we will describe a common way

The neurons are layered and the layers named as shown

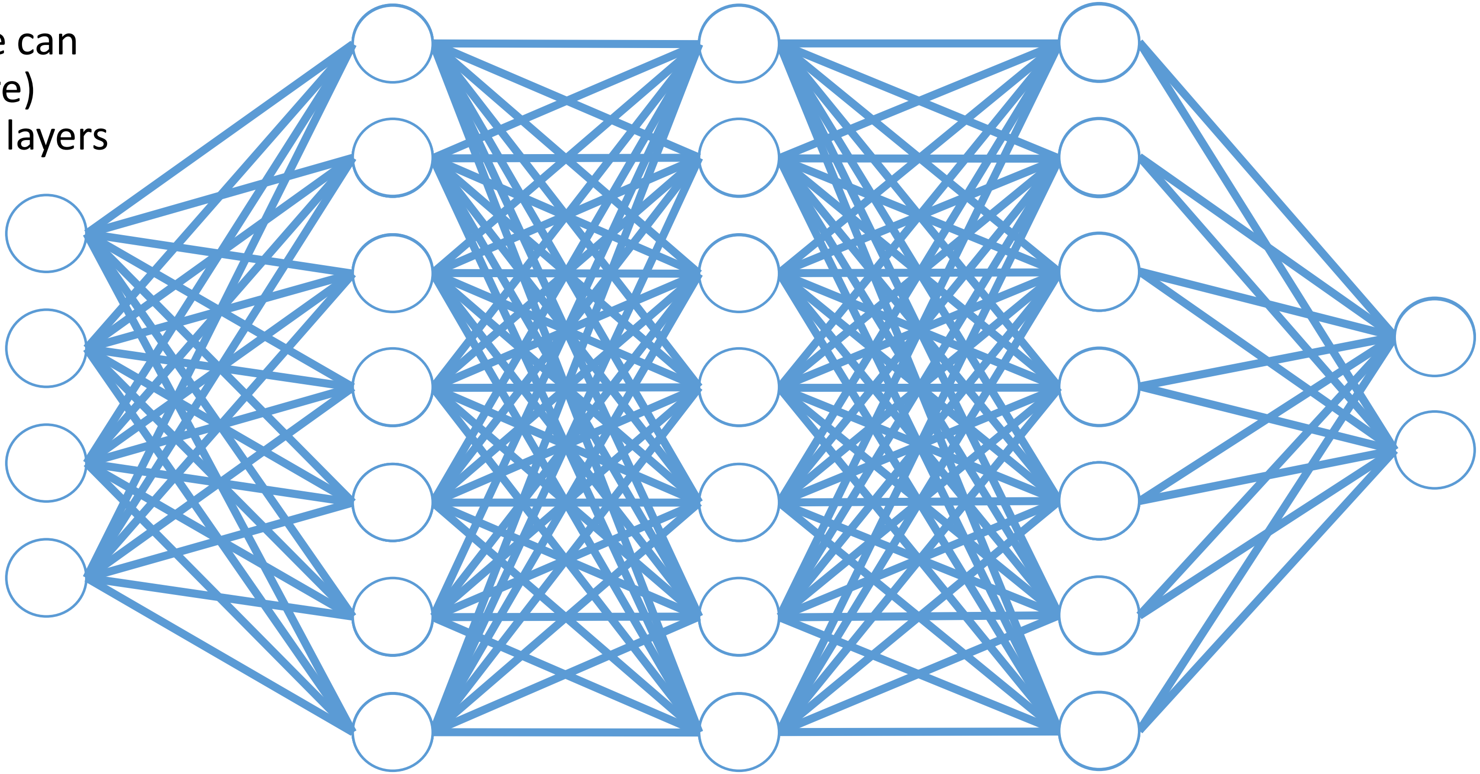
Neurons are connected to every other one in the preceding and succeeding layers

Data is fed into the input neurons, output is read from the output neurons



Neural Networks

In practice there can be (and often are) multiple hidden layers



Neural Networks

So, the good news is that we have more pathways and more weights to adjust. The downside is that the number of possible **combinations** of weight values explodes. There is a trade off here. We need large numbers of weights to give the separation but the number we need to do the job is so large that there are billions/trillions of combinations of values. Finding the combinations of weights that best separate the data effectively is very hard; too hard to do by simply running through all the possible combinations

This problem effectively stopped the development of neural nets for a long time. Happily, in 1974 Paul Werbos, working at Harvard, introduced the idea of back propagation in his PhD thesis. However this took a while to be adopted

Neural Networks

Data is fed into the input neurons

Imagine that you want to look for insurance fraud. You remember this data

(I've removed the Postcode column)

Input columns

Target column

CaseID	Date of Theft	Time of Theft	Value of Items	Number of Items	Fraud
1	23/01/2019	11:30	£350	7	No
2	23/01/2019	12:50	£750	9	No
3	23/01/2019	13:45	£1,230	2	Yes
4	23/01/2019	14:40	£540	3	No
5	23/01/2019	14:50	£450	7	No
6	23/01/2019	15:20	£300	5	No
Etc.	Etc.	Etc.	Etc.	Etc.	Etc.

Neural Networks

We will essentially set up a neural net with one input neuron for each input column. The values in these columns are complex and certainly drawn from different domains. So we will normalise them

There are many different ways of normalising the data

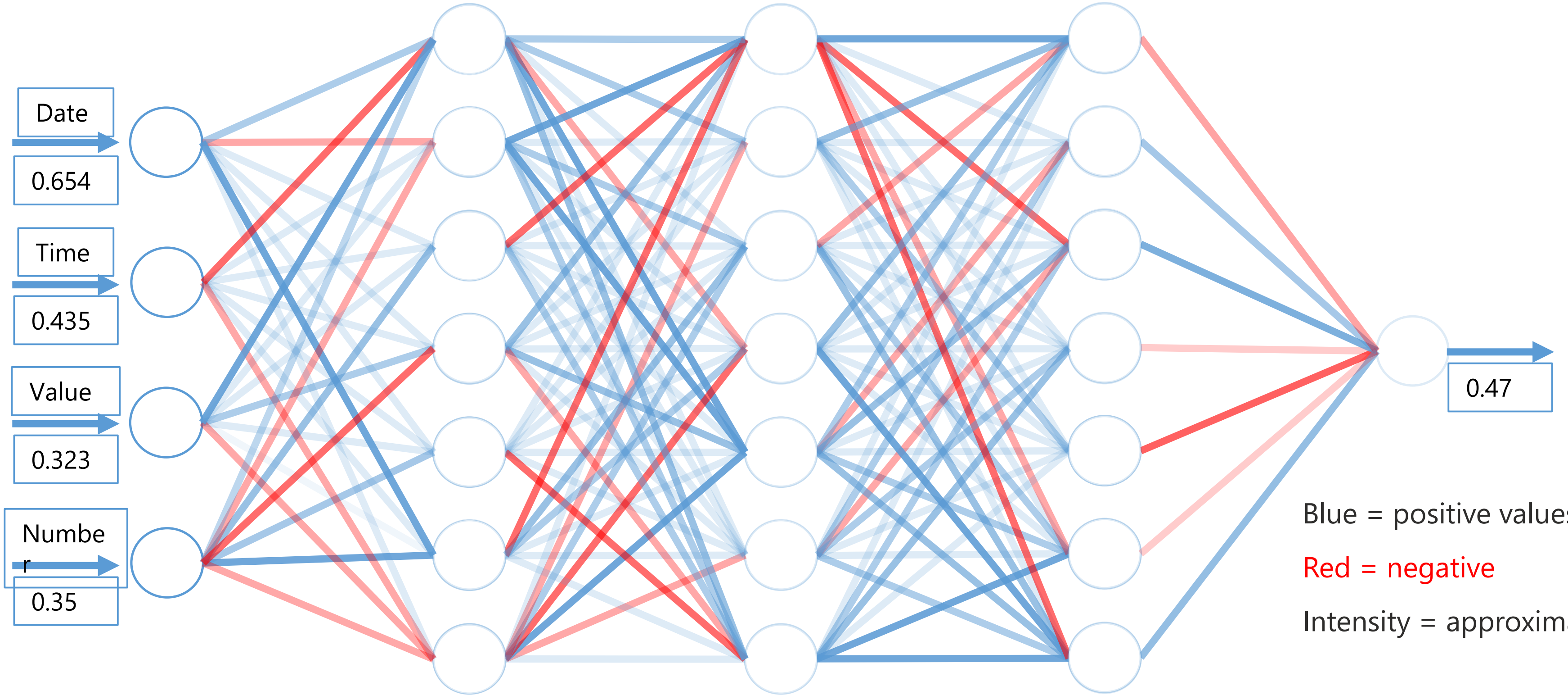
Input columns

Target column

CaseID	Date of Theft	Time of Theft	Value of Items	Number of Items	Fraud
1	0.654	0.453	.323	0.35	0
2	0.654	0.463	.654	0.45	0
3	0.654	0.532	.850	0.1	1
4	0.654	0.567	.453	0.15	0
5	0.654	0.617	.398	0.35	0
6	0.654	0.633	.254	0.25	0
etc	etc	etc	etc	etc	etc

Neural Networks

The normalised data is fed into the input neurons. Before training it might look like this



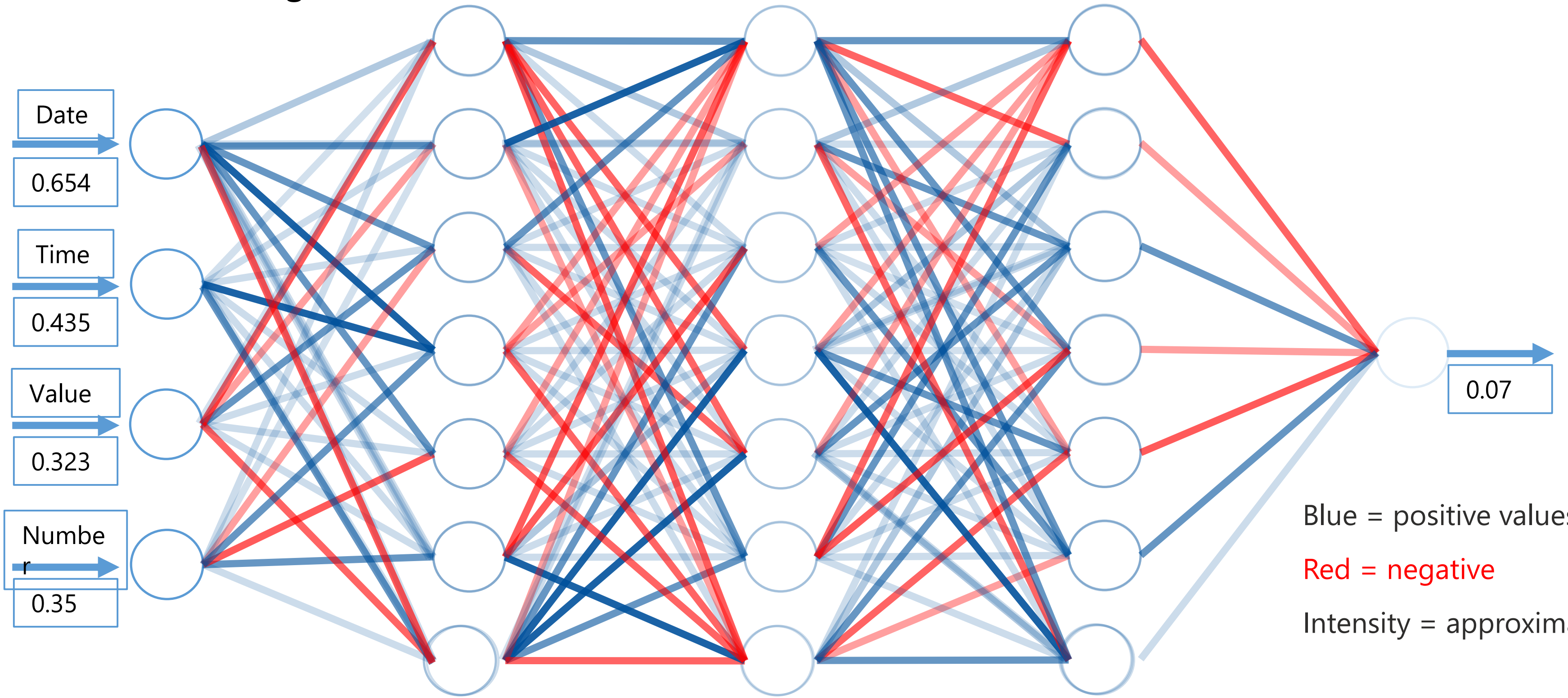
Blue = positive values

Red = negative

Intensity = approximate value

Neural Networks

after training like this



Blue = positive values

Red = negative

Intensity = approximate value

Neural Networks

1944 - Neural networks were first proposed at the University of Chicago by Warren McCullough and Walter Pitts. These early neural nets had weights and thresholds but no layers. There was also no training mechanism. McCullough and Pitts demonstrated that, in principle, a neural net mimicked how a human brain worked and that a neural net could do the same computation as a digital computer. Thus they drew the comparison between the brain and the machine

1952 - McCullough and Pitts moved to MIT as part of the team that formed the cognitive science department

1957 – The world’s first trainable neural network (the Perceptron) was created by the psychologist Frank Rosenblatt at Cornell University. It had adjustable weights but only one hidden layer. Perceptrons were actively studied in both computing and psychology until 1959

Neural Networks

1959 - Marvin Minsky and Seymour Papert (mathematicians at MIT) published a book (Perceptrons) which essentially argued that performing some common computations on Perceptrons was going to be far too time consuming. It is argued that this book destroyed the interest in neural nets at that time. Minsky and Papert went on to become the co-directors of the new MIT Artificial Intelligence Laboratory

1974 – Paul Werbos, Harvard. PhD thesis. Introduced backpropagation

1980s - Neural nets had a resurgence and then again disappeared

1986 - Learning representations by backpropagating errors, *Nature* **323**, 533-536 (9 October 1986) David E. Rumelhart, Geoffrey E. Hinton & Ronald J. Williams (This will ultimately make neural nets computationally viable)

Neural Networks

2010 – MIT seriously considered dropping neural nets from the AI syllabus. Many felt that neural models were not a good representation of the brain and no neural nets had done anything useful anyway. One reason MIT decided not to drop it was to ensure that students knew about them and would not waste time reinventing them

2012 – Geoffrey Hinton (great-(great?)-grandson of George Boole!) published a paper about picture recognition that stunned the world and showed, once and for all, that neural nets could do very serious work. Since then they have become a mainstay of machine learning

Neural networks are simply a complex function

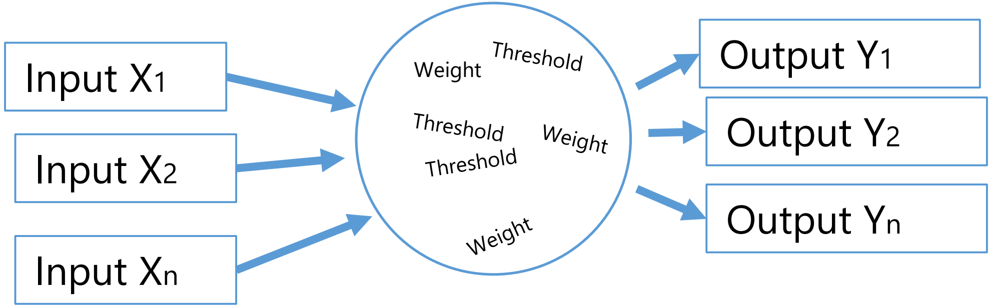
Now, of course, neural networks (and, indeed, our brains) are stuffed with neurons. But we can think of them as simply a collection of weights and thresholds

They have a large number of inputs (X) and those produce outputs (Y)

Which means that we can think of the entire network simply as a function. The output is a vector of Y values which is a function of the input vector (VX), the weights vector (VW) and the thresholds vector(VT)

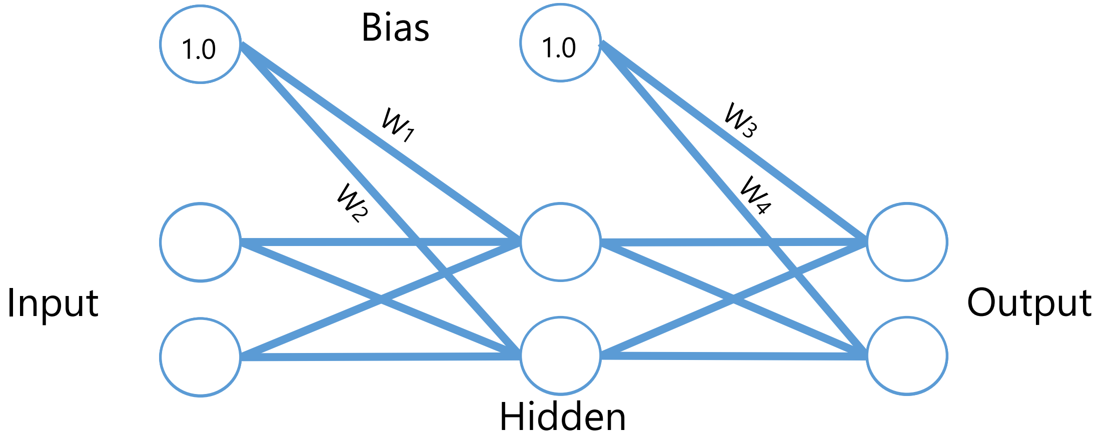
VY = f (VX, VW, VT)

We cannot alter the input vector but we can alter the weights and the thresholds in order to get the output vector we desire



Bias

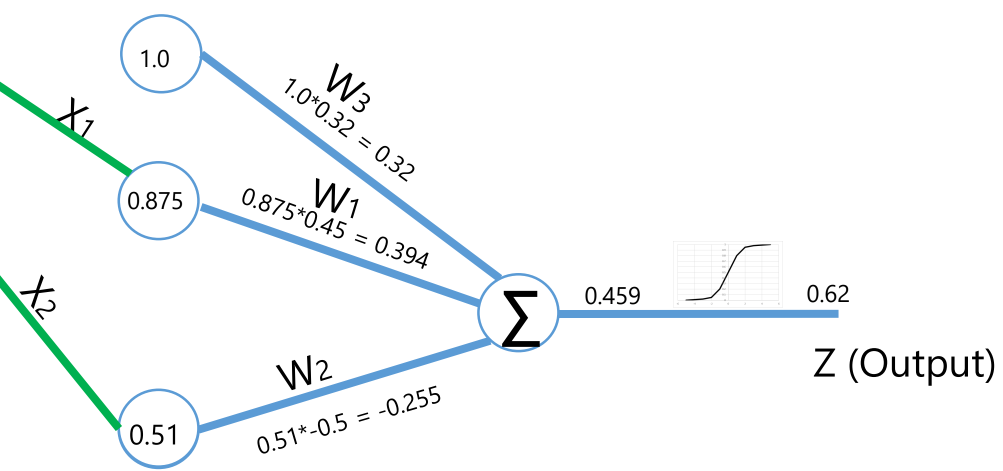
- We have describe neural nets as having simply weights and measures and this can be true. However it is also possible that the model may include a bias. This is simply a constant that is added to the calculation. The effect of adding a bias is essentially to move the activation function. This is sometimes helpful in speeding up learning. The bias values are also multiplied by a weight



Bias

- The weight is used in the usual way

Sepal Length	Sepal Width	Species
0.51	0.875	setosa
0.49	0.75	setosa
0.47	0.8	setosa
0.46	0.775	setosa
0.7	0.8	versicolor
0.64	0.8	versicolor
0.69	0.775	versicolor
0.55	0.575	versicolor



Thank you for listening

FEEDBACK!!!!!!!!!