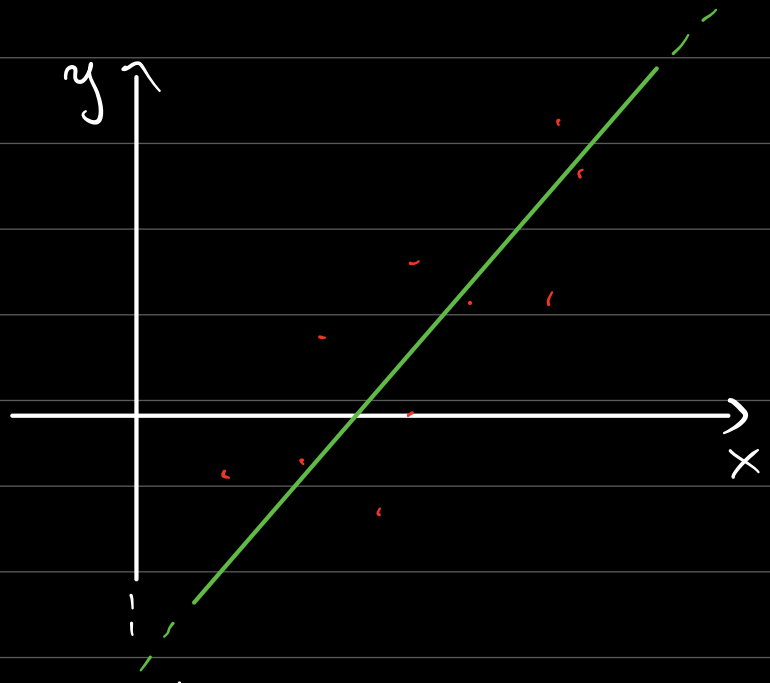
Logistic Regression

So far we have seen Linear Regression, an algorithm that was trying to fit a line given some points:



As any "regression" problem, it was predicting any possible value on the y axis, from $-\infty$ to $+\infty$.

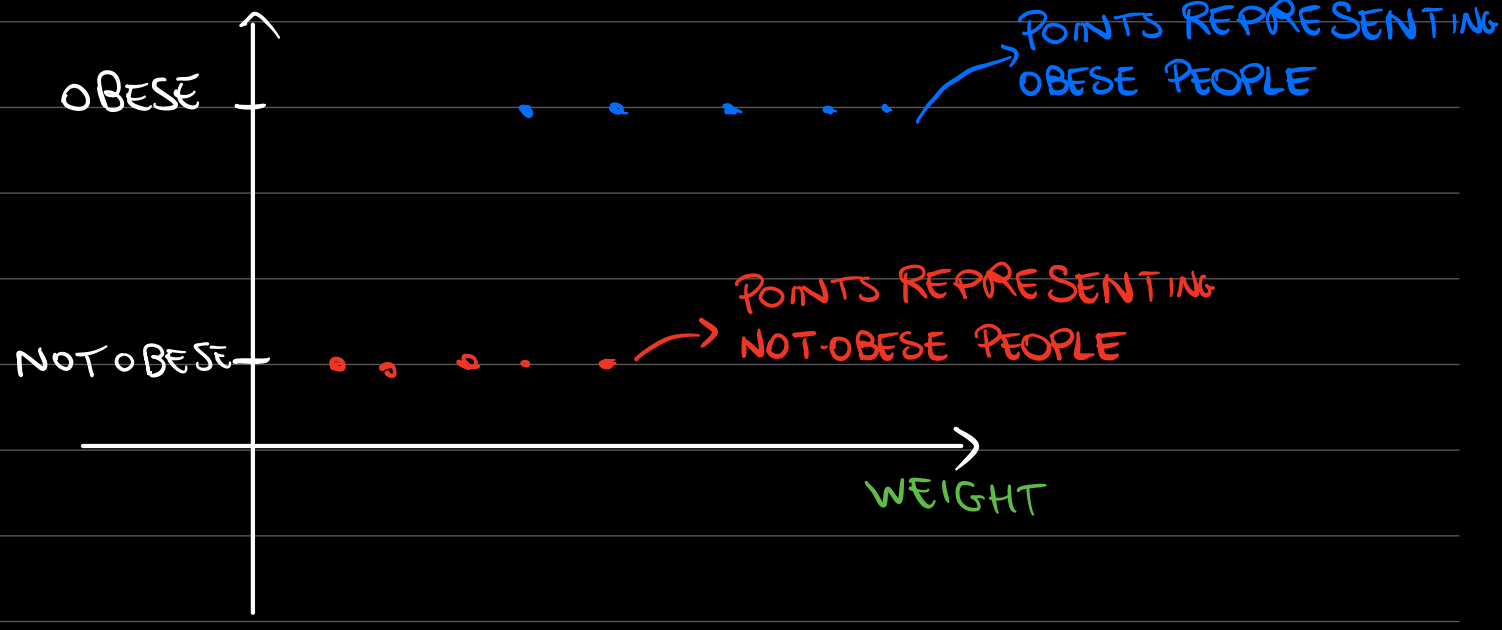
But what if you have a binary classification problem?

Binary classification means that you want to predict two values: either 0 or 1, by giving some input

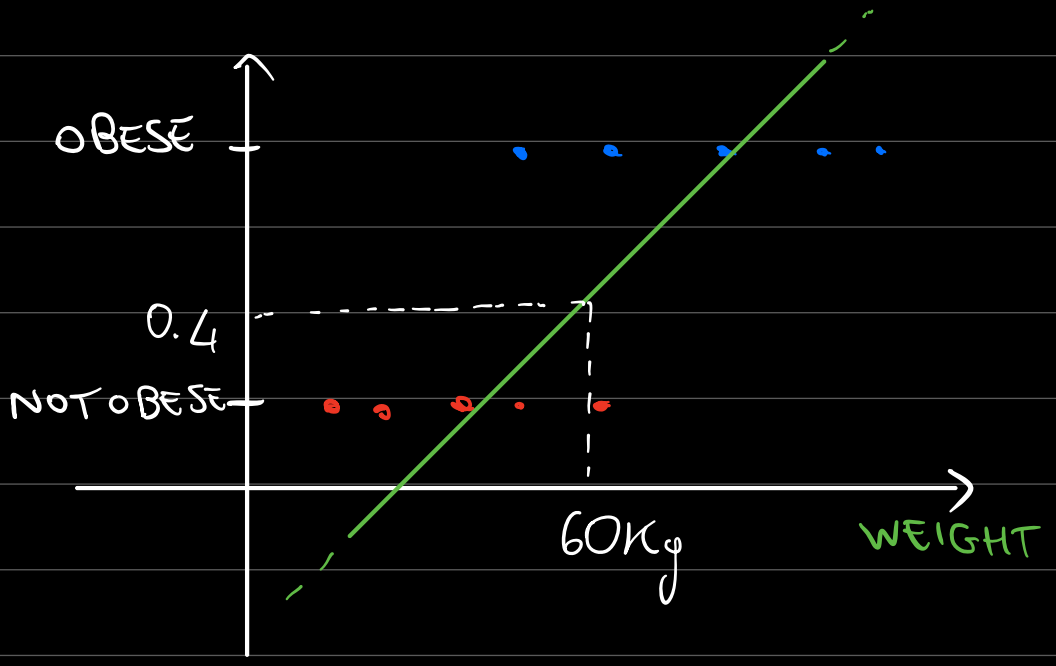
features.

Example:

You want to predict if a person is obese given his/her weight.

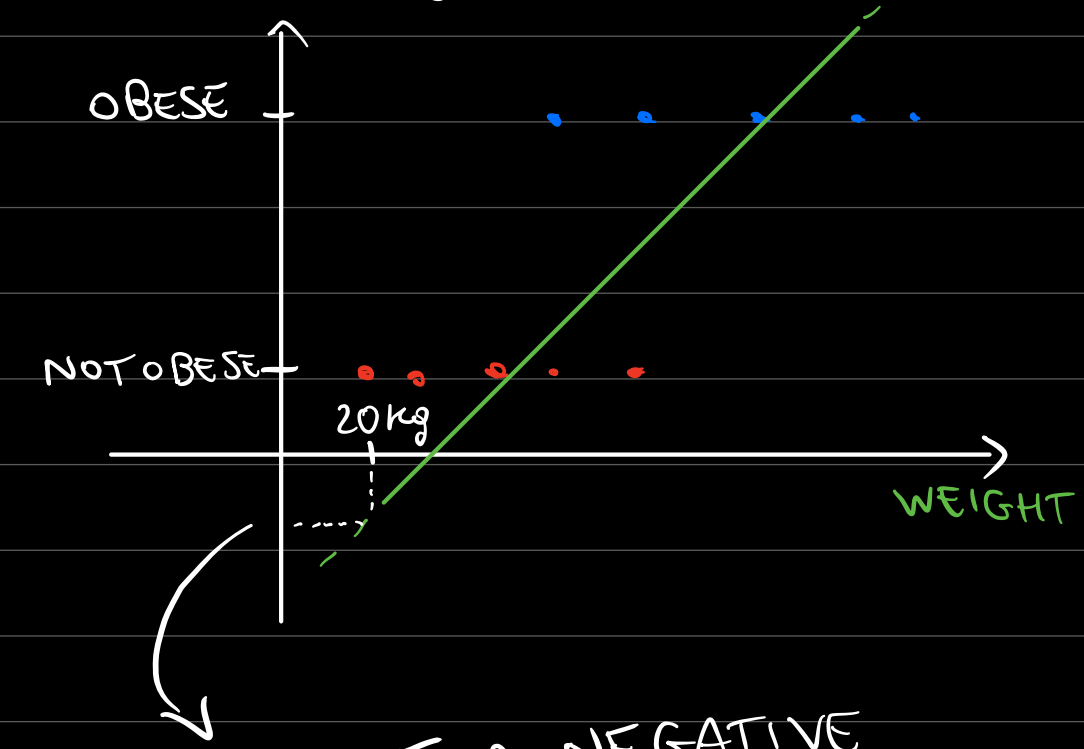


this means that given a weight, let's say 60 Kg, you want your model to tell you either 0, if not obese, or 1, if obese.
Let's toy with a line:

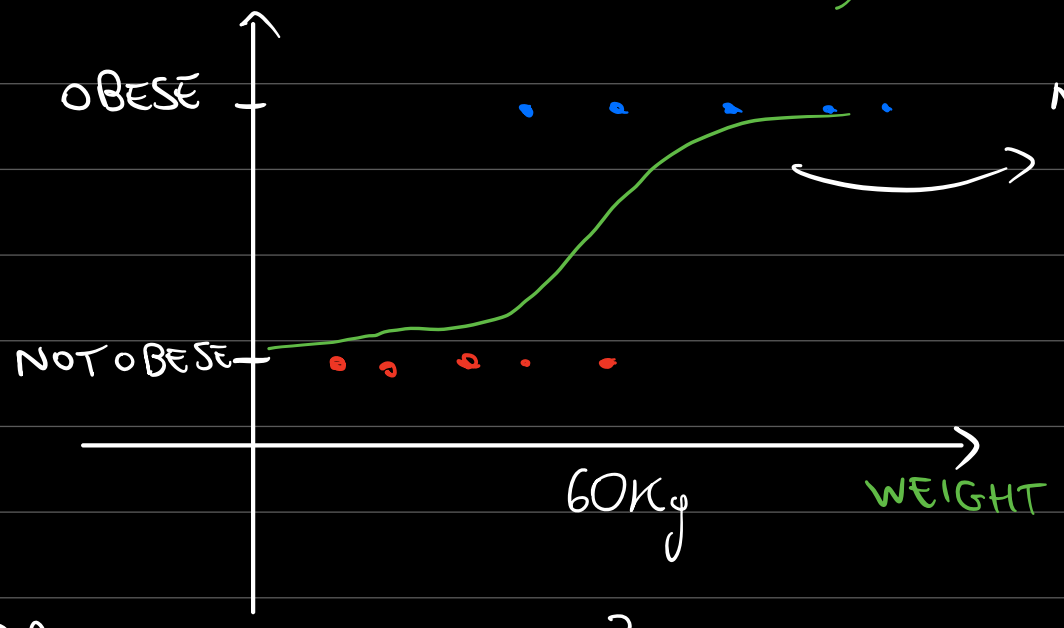


Mmh, it's 0.4, not 0 or 1. Ok, this may not be that much of a problem, because we can round the number.

But what if you have something like 20Kg?



So we need something that could warp that line into something like



Why is it nice?

- IT'S ALWAYS BETWEEN 0 and 1
- FOR BIG VALUES IS CLOSE TO 1
- FOR SMALL VALUES IS CLOSE TO 0

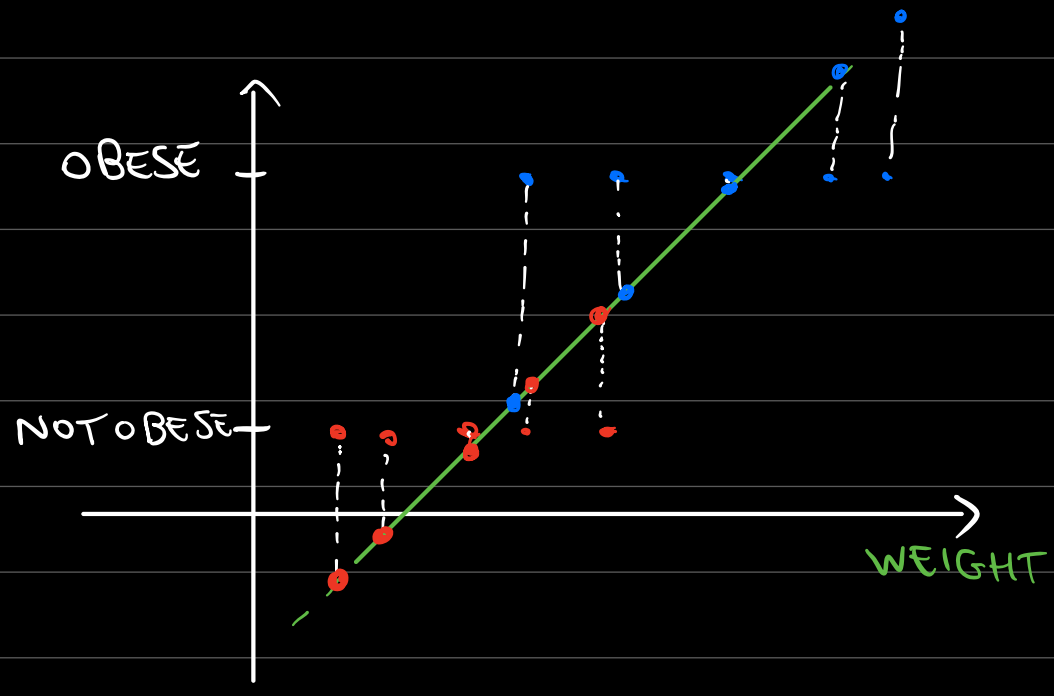
What else is between 0 and 1?
Probability!

BAM! (StatQuest quote)

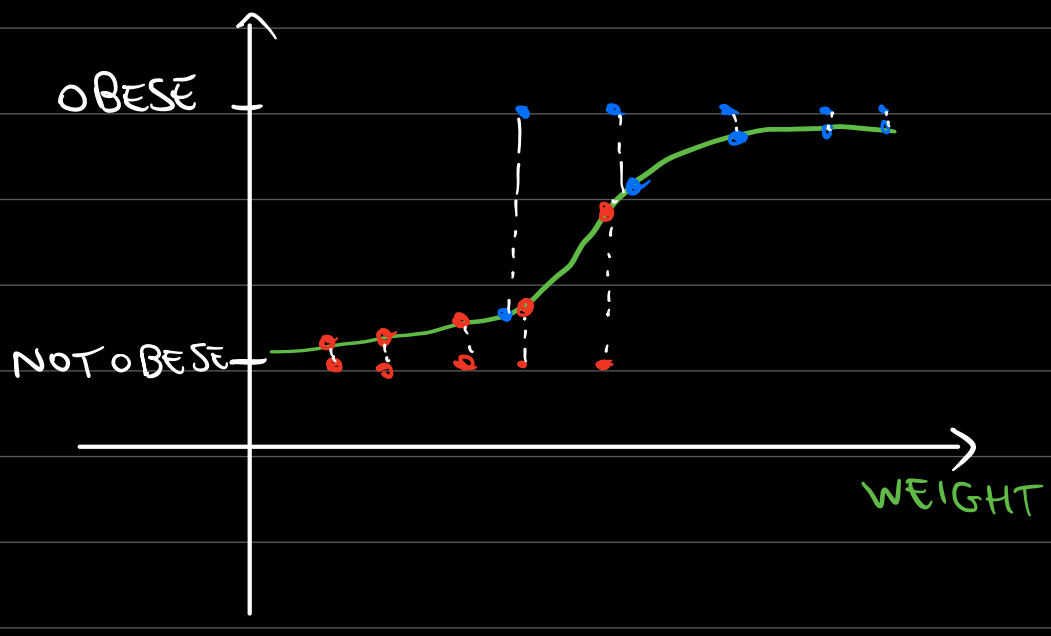
And this is good because we can interpret those numbers as "the probability of being obese".

OK, so we could do something like this:

1. PROJECT THE POINTS ON THE LINE

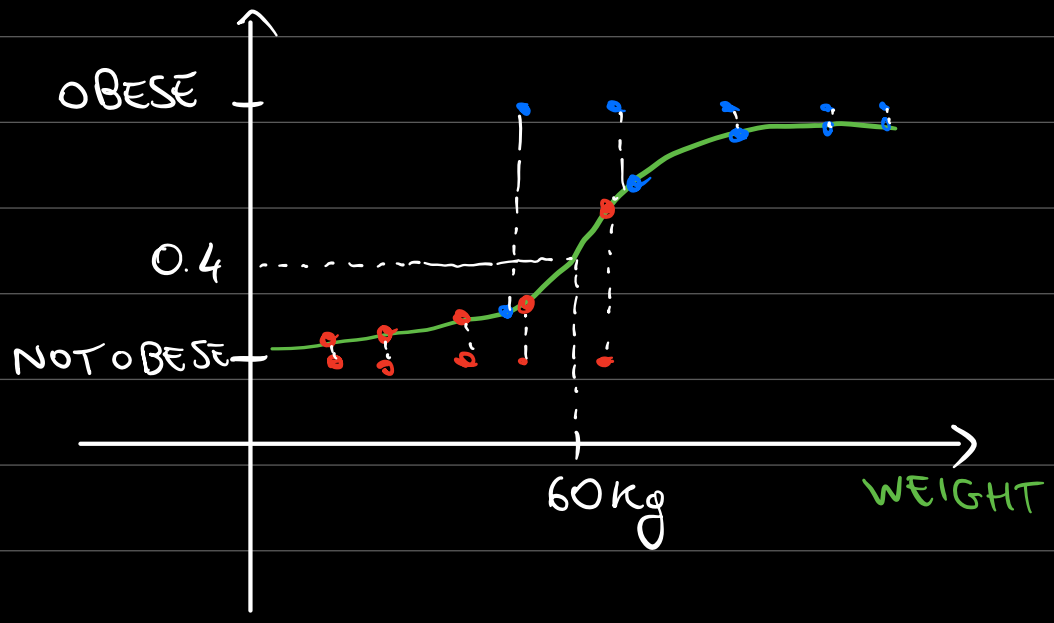


2. WARP THE LINE



3. BAM!

Now, if we ask what is the probability of a person of 60 kg to be obese (according to our model) we get



40%! Nice!

Ok, let's do this steps mathematically:

1. Define a line $y = wx + b$

Example: $y = 5x - 2$

and project the point $x = 60$ to that line:

$$y = 5 \cdot 60 - 2 = 300 - 2 = 298$$

2. Warp the line with the

SIGMOID FUNCTION

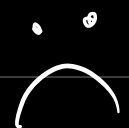
or

LOGISTIC FUNCTION

$$\sigma(y) = \frac{1}{1 + e^{-y}}$$

In our example

$$\sigma(298) = \frac{1}{1+e^{-298}} \approx 1$$



MY MODEL IS TELLING ME THAT IT'S ALMOST 100% SURE I'M OBESE! WITH JUST 60 kg!

Or, my model is wrong! How can I improve it?

Well, I don't have much power, I can only tweak the parameters of the line!

Logistic Regression algorithm does actually this: it iteratively adjust the weights of the line in order to make the least mistake as possible!

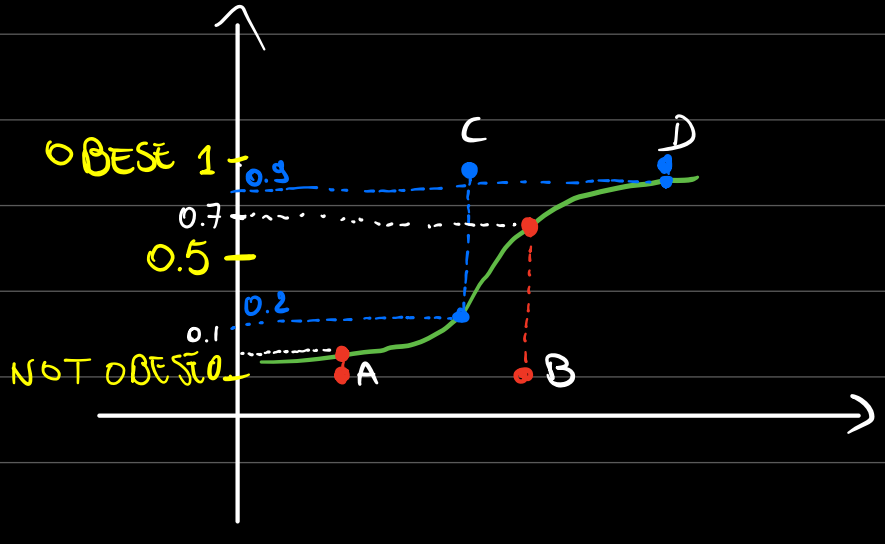
Wait, what are the weights of a line?

They are just the coefficients that determine a line: if $y = ax + b$, the weight are a and b.

In some notations, you will find a to be the weight and b to be the bias term.

Ok, you said we need to minimize the error, how do we compute it?

Let's see an example with four points



I called the four points A, B, C, D.

- A was supposed to be red, so to have label 0. It has 0.1, so the error is $|0 - 0.1| = 0.1$

- B was supposed to be red, so to have value 0, so the error is

$|0 - 0.7| = +0.7$

↓ TRUE LABEL ↓ PREDICTION

- C was supposed to be blue, so to have label 1, but it got a score of 0.2. So the error is $|1 - 0.2| = 0.8$
- D was supposed to be blue, so to have label 1, but it got a score of 0.9. So the error is $|1 - 0.9| = 0.1$.

What is the **total error** of our model? We need to multiply all the errors we got:

$$0.1 \times 0.7 \times 0.8 \times 0.1 = 0.0056$$

But... it's very small!

We can use a clever trick: make the log of the error! In fact

$$\log(ab) = \log(a) + \log(b)$$

$$\Rightarrow \log(0.1 \times 0.7 \times 0.8 \times 0.1) =$$

$$\log(0.1) + \log(0.7) + \log(0.8) + \log(0.1)$$

$$= -2.252$$

↓ OK, more reasonable!

So now we know many things:
- how to create a model warping a line
- how to compute the error

now the question is: how do we update the weights to minimize the error?

Well, we have a formula.
Let's say that our x (the weight of a person) is $x = 60$, and that our model is $y = \frac{1}{2}x - 2$, let's compute our prediction

$$\mathcal{I} = \frac{1}{20} \cdot 60 - 2 = 3 - 2 = 1$$

$$\Rightarrow \hat{y} = \sigma(\mathcal{I}) = \sigma(1) = \frac{1}{1+e^{-}} = 0.73$$

$\hat{y}$ ← our prediction

Let's say that our training sample of 60 kg was not-obese, it means the label should have been 0. So the error is $10 - 0.73 = 0.73$.

To update the weights given this data point we need to use the following formula

$$w_{new} = w_{old} - \alpha \cdot error \cdot x$$

where alpha is a learning rate, that is "how much you want to update at each step"

In our example we have only $\frac{1}{20}$ as weight, and let's use $\alpha = 0.1$:

$$w_{new} = \frac{1}{20} - 0.1 \cdot (-0.73) \cdot 60$$

- $\frac{1}{20}$: old weight
- α
- -0.73: error with the sign
- x

$$\Rightarrow w_{new} = 4.43$$

For updating the bias term it's the same, but you omit x:

$$b_{new} = -2 - 0.1(-0.73) = -1.927$$

So after updating the weights the model becomes:

$$y = 4.43x - 1.927$$

Repeating the process over and over on the data points will create a better model!