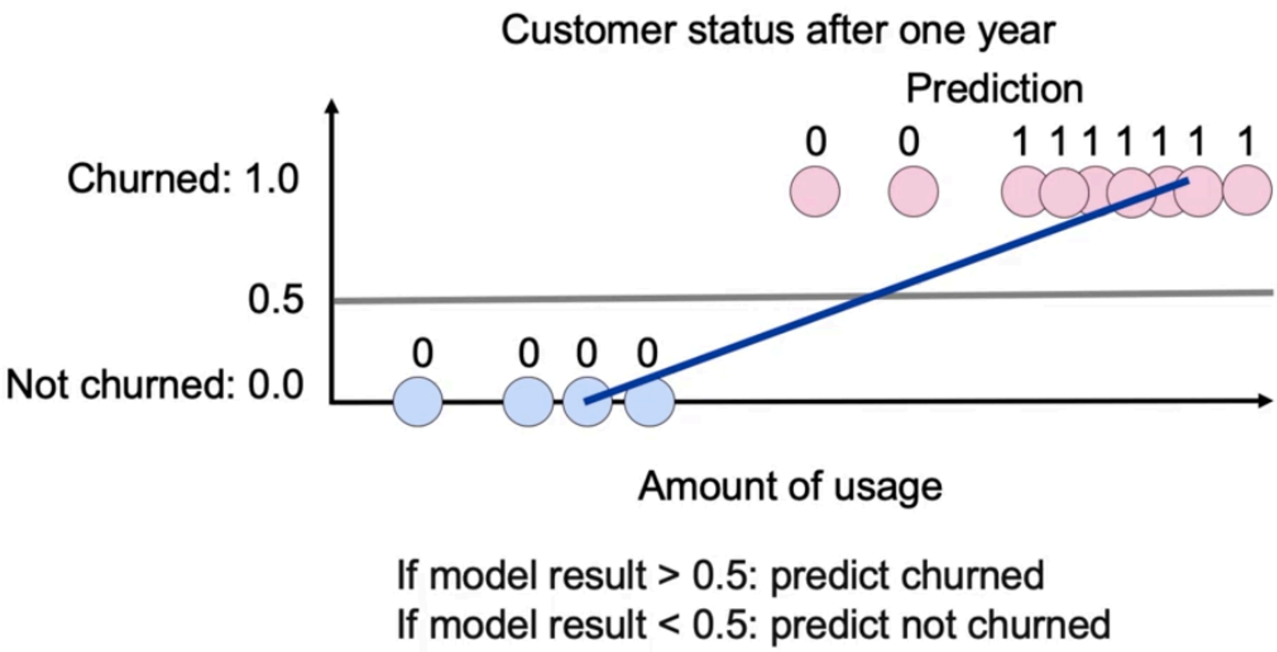
2) Logistic Regression

Useful for binary classification. Should NOT be used for regression

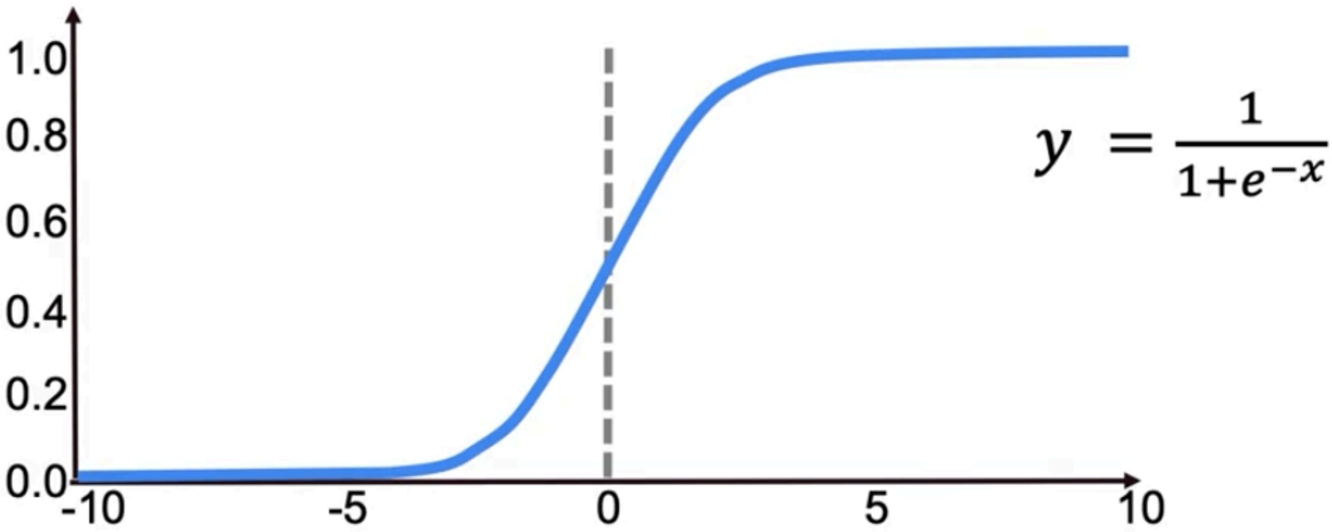
If we will use simple linear regression, it can misclassify data (cause of points/outliers far away) like so:

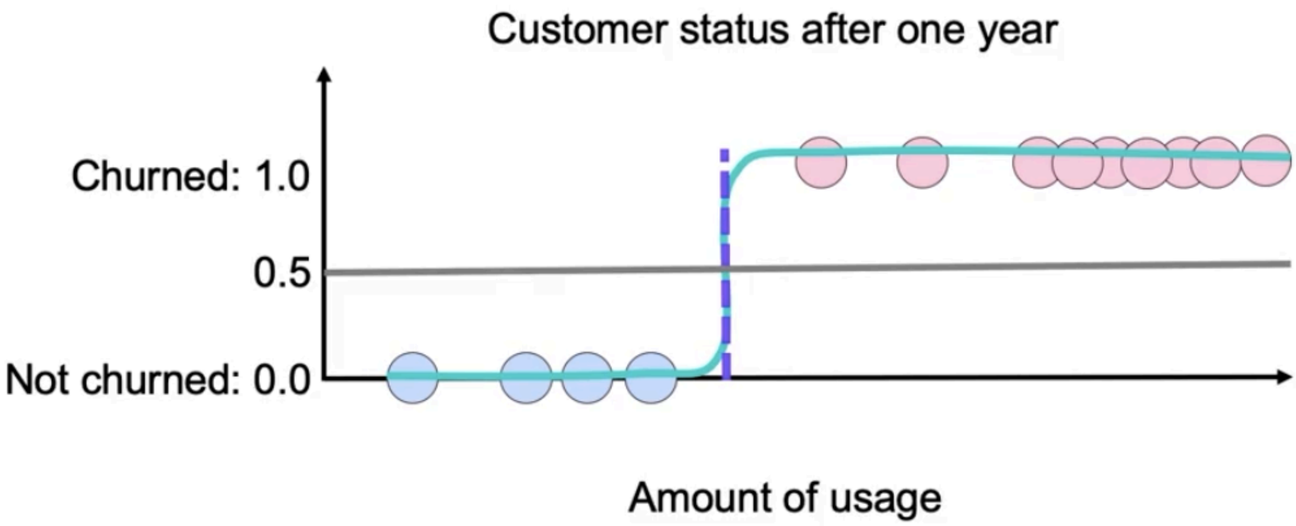


So we need to weight the samples (that are far away) much lower. Thats where the logistic function (ie sigmoid function) comes into play

Sigmoid function

It looks as follows:





$$y_\beta(x) = \frac{1}{1+e^{-(\beta_0+\beta_1 x+\varepsilon)}}$$

So rather than the slope being `mx+b`, it is 1 / (1 + $e^{-(mx+b)}$)

1 / (1 + $e^{-(mx+b)}$) can be further represented as

$$P(x) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x + \varepsilon)}}$$

Logistic Function

$$P(x) = \frac{e^{(\beta_0 + \beta_1 x)}}{1 + e^{(\beta_0 + \beta_1 x)}}$$

$$\frac{P(x)}{1 - P(x)} = e^{(\beta_0 + \beta_1 x)}$$

Odds Ratio

and the odds ratio can further be represented as

$$\log\left[\frac{P(x)}{1 - P(x)}\right] = \beta_0 + \beta_1 x$$

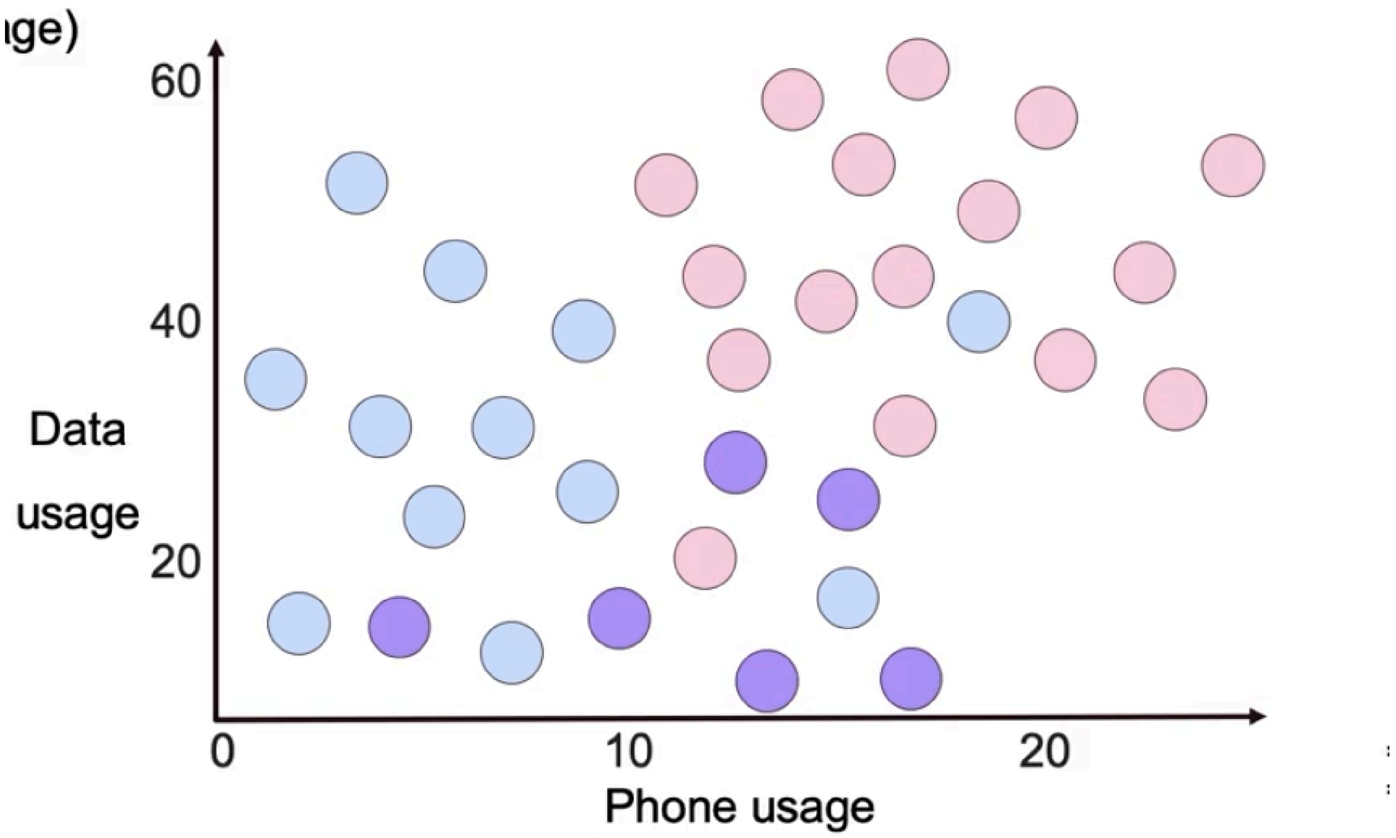
Odds Ratio

One vs all

This is a method used when we are not dealing with binary classes. Here, for every class, we put that class's points in a separate class and all the other points in another class

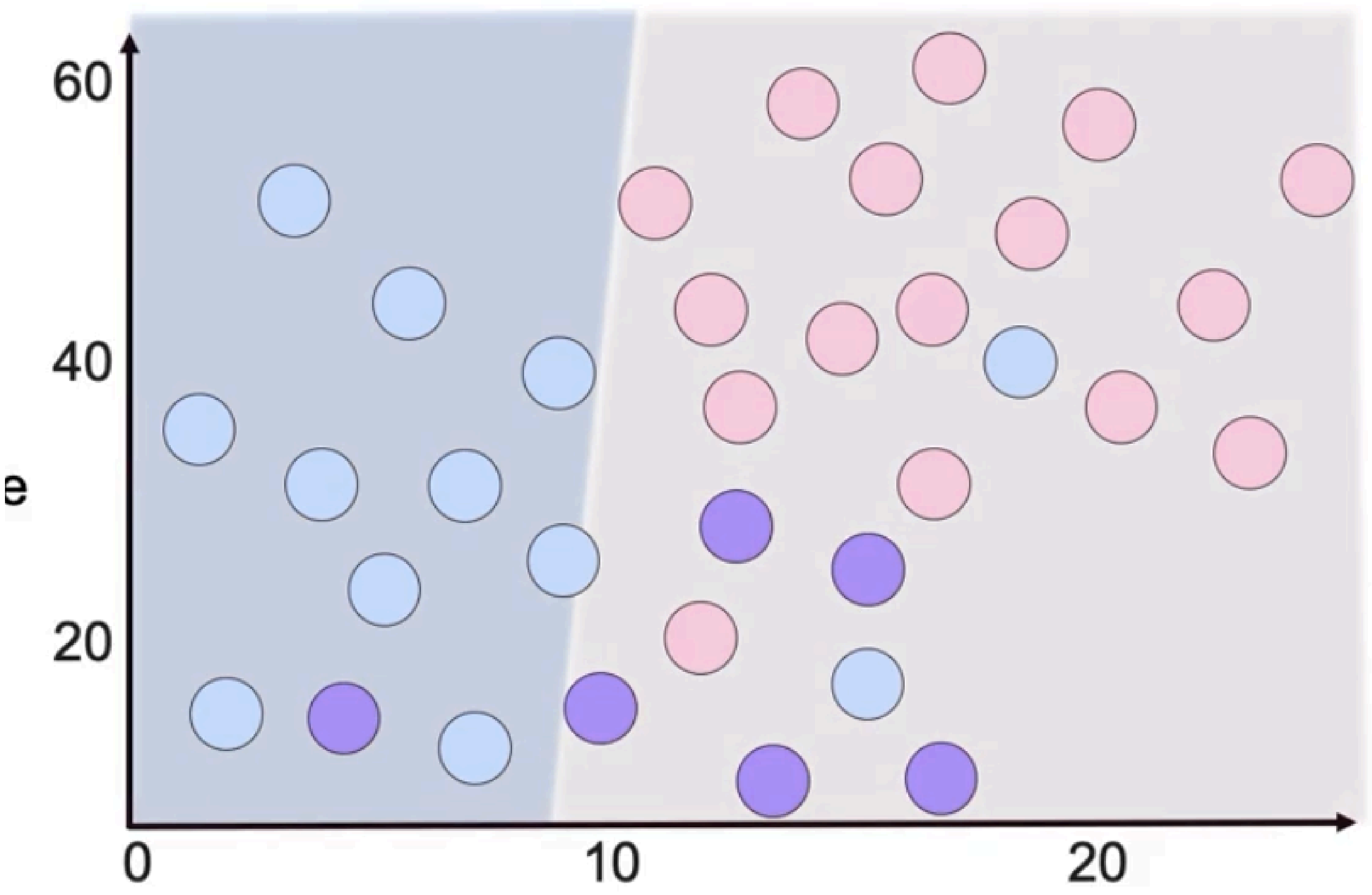
So for example, suppose we have 3 classes (a,b,c). So first, we will train a model with all points of `a` being put in `class1` and points of `b` and `c` being put in `class2`. This model will represent the probability that a given point will be of the `a` class. Similarly, we will train two more models, one representing the probability of a point being of class `b` and the other representing the probability of a point being of class `c`. We will then make our final prediction based on which model gave us the highest probability

For example, in a dataset like this:

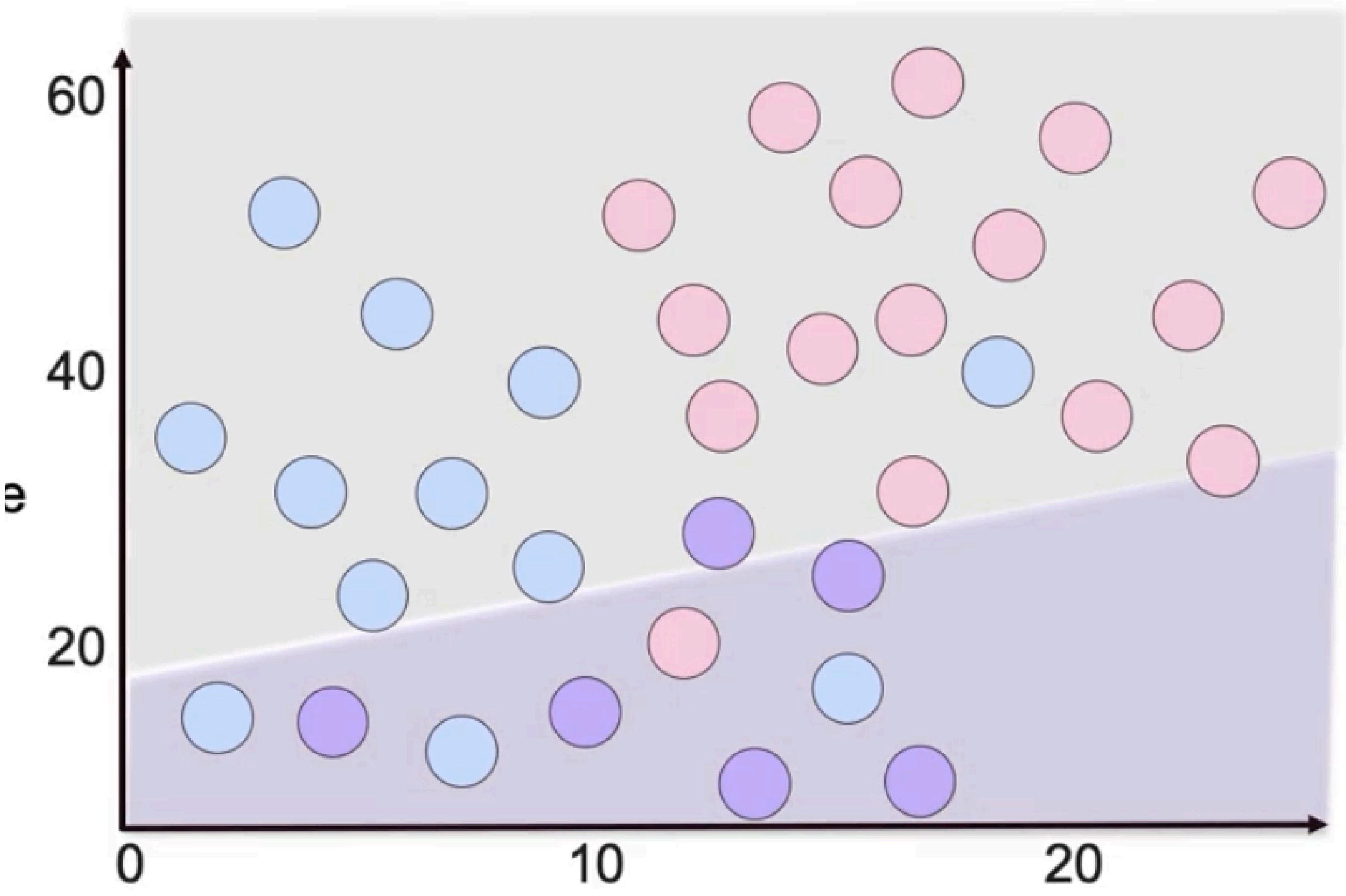


We end up with the following models:

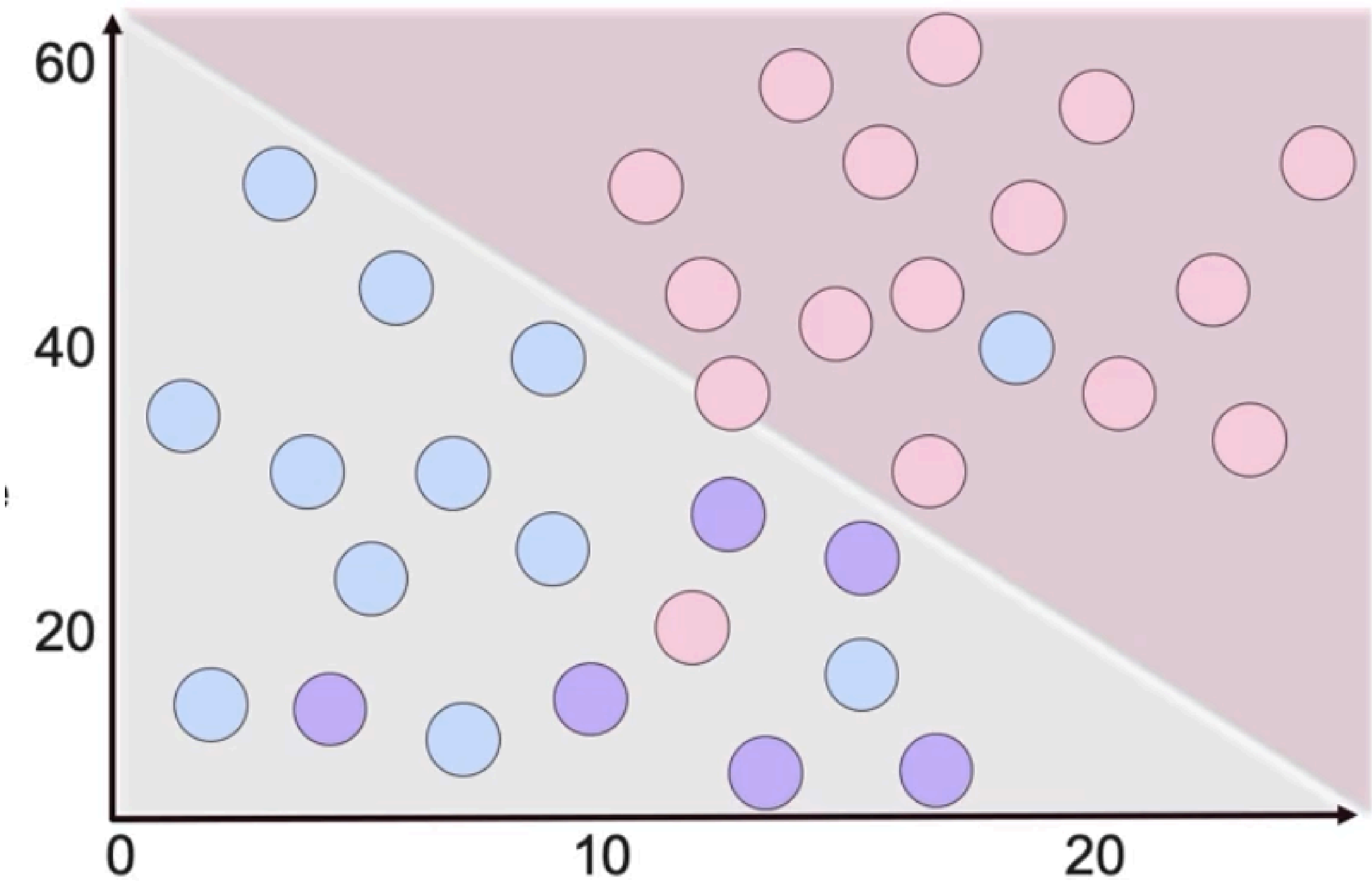
(blue vs purple+pink)



(purple vs blue+pink)



(pink vs blue+purple)



Code

In code, we can pass hyper parameters like `penalty`, `c`, and `solver`

`penalty` is wether we want to use l1, l2, or elastic net regularisation

`c` is the inverse of λ

There is also the `LogisticRegressionCV` class which tunes regularisation parameters via cross validation