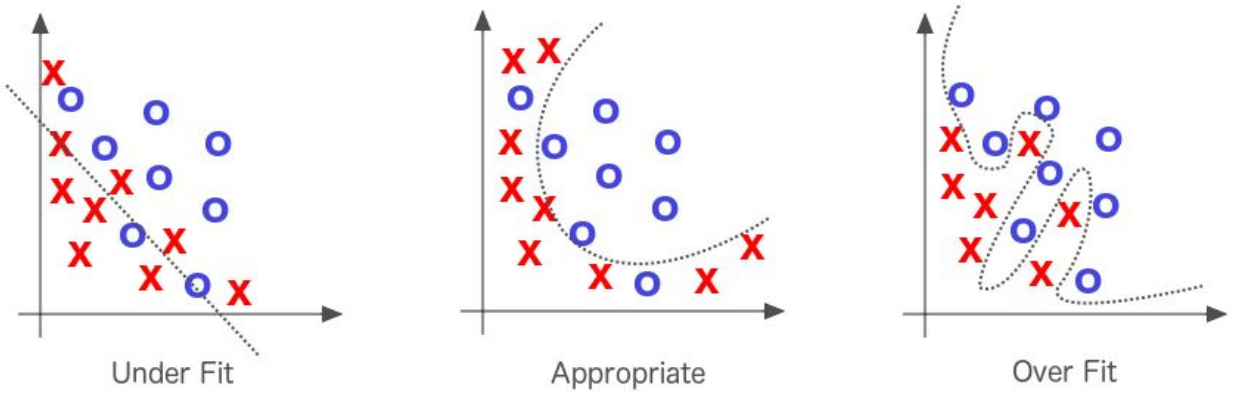
Overfitting

(what you just did)

Learning irrelevant details of the dataset: Overfitting

We say that a model is overfitting if its predictive abilities (output) depend too much on the data which was used for learning the parameters.

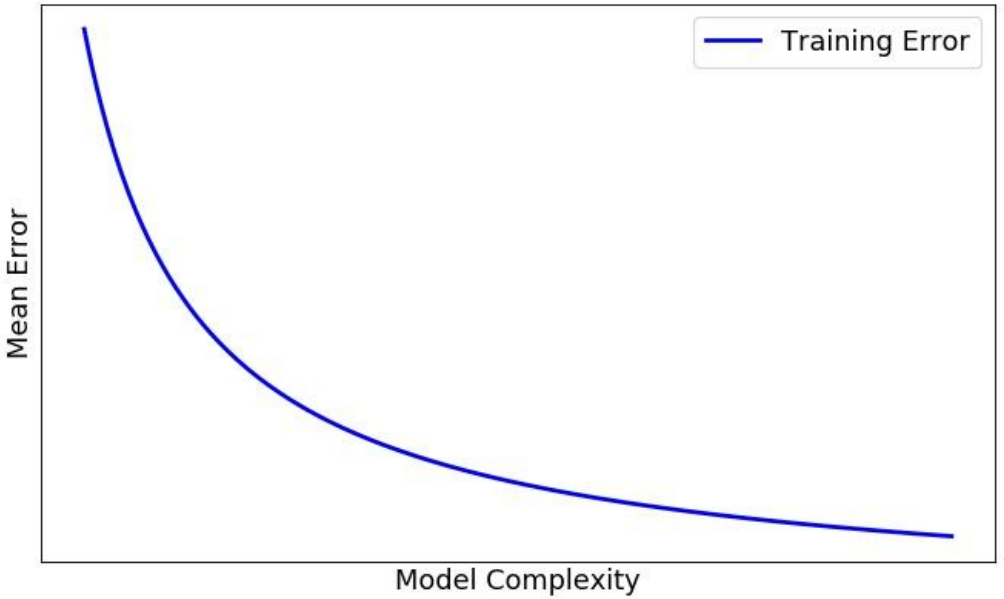


Think about adding one new point to the graph, how well will the model perform?

The more complex the model is, the more likely it is to overfit. Overfitting is a huge issue as the model will seem to perform great when training it and then perform poorly when applied.

Overfitting: training error

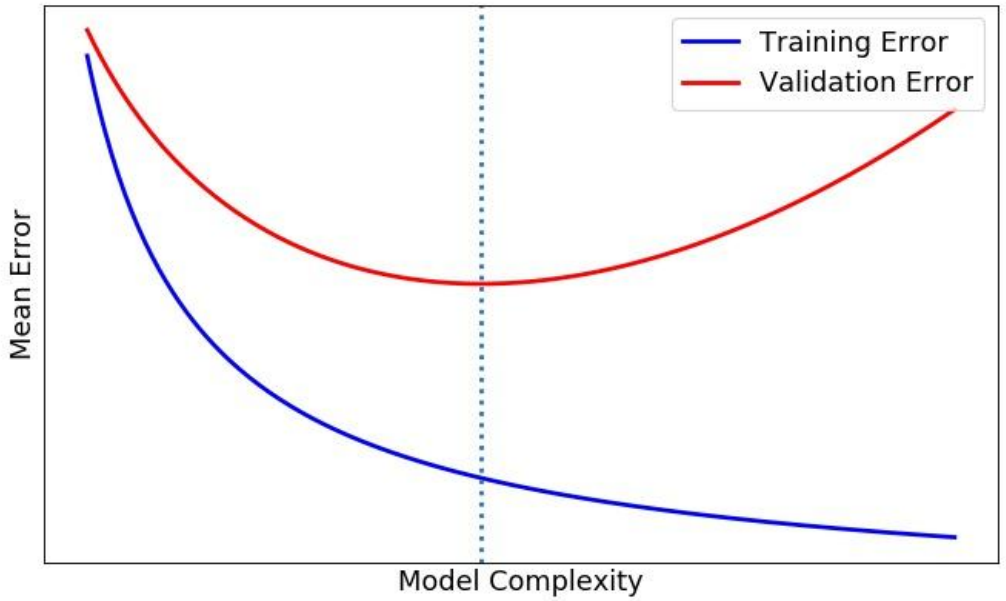
As you have just experienced, **making the model more complex** (e.g. adding more variables, interaction terms etc), **decreases the "mean error" of the model** [blue line, training error].



Overfitting: test error

However, as we have seen in the competition, a low mean error might mean that our model overfits the data: it might start learning irrelevant details of the data and the mean error goes towards 0 [blue line].

This mean that, while the model will seem great on the data we have, **it will perform poorly on new data we have not see before** [red line, validation error]**.** This dataset is called the validation set.



As we increase model complexity:

Error on data we have seen (training) and data we have not seen (validation) decreases. Making the model more complex gives us a better model.

At some point the model starts to overfit. The error on data we have seen (training) continues to decrease but error on unseen data (validation) starts increasing

Prevent overfitting: Train - Validate - Test

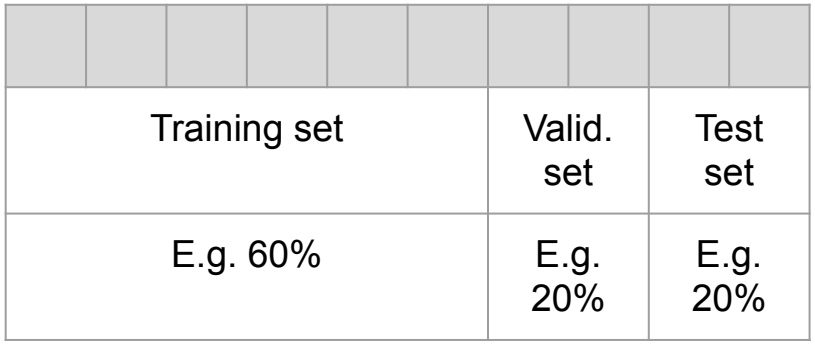
To prevent overfitting, datasets used to train a predictive model should **always** be divided (at random) into:

Training set: Part of the dataset that we use to **train the model**, learning the coefficients (a, b, c, etc) of our logistic regression.

Validation set: Part of the dataset that we use to validate the models we tried

Test set: Part of the dataset that we only use at the end of the analysis to evaluate how our final model will perform on new data. This is the final one we will report

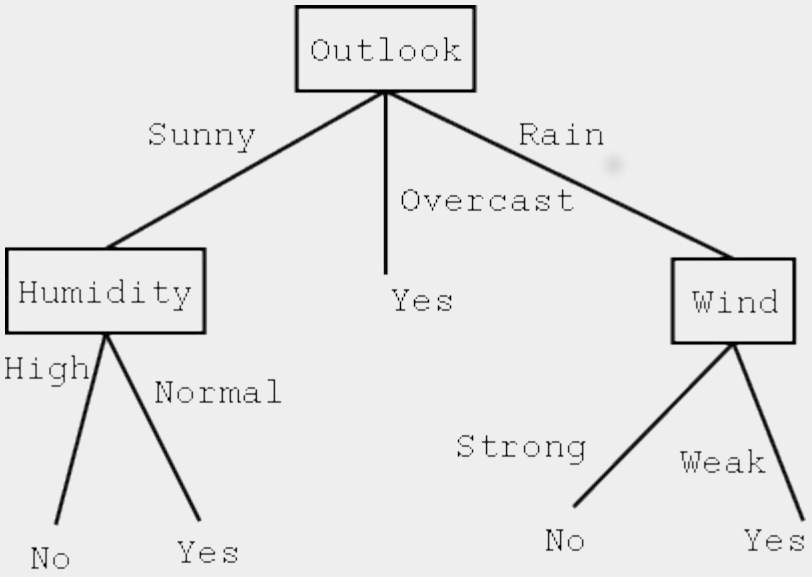
Dataset:



Note the key assumption we make here (and in most predictive models): the data in the test set is a good representation of any new data that we will use our model on. This is why the performance of our model on the test set is the true performance we can expect from our model. This might not always be true though.

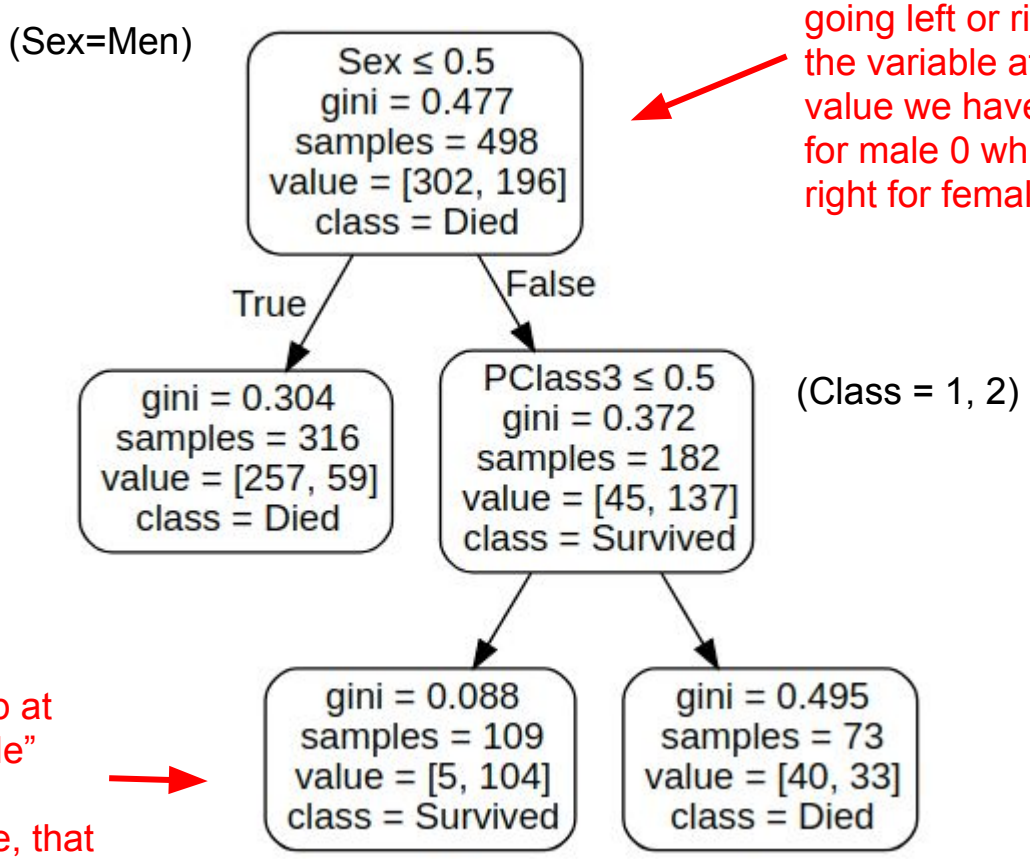
Decision tree

E.g. are people going to play tennis tonight?



For the titanic a simple model could look like this

We start at the top and then go down the tree by, at every node, going left or right depending on the variable at that node and the value we have (here Sex so left for male 0 which is < 0.5 and right for female 1)



We do so until we end up at what is called a “leaf node” where we decide which prediction we make (here, that the person survived)

Gini coefficient

Gini impurity measures how often a randomly chosen element would be incorrectly labeled (survived/died) if it was randomly classified according to the distribution of labels in the subset.

$$I_G(f) = \sum_{i=1}^{J} f_i(1 - f_i)$$
$$= \sum_{i \neq k} f_i f_k$$

with f_i the frequency in class *i* (survived/died).

The maximum gini for two classes is 0.5 (each class at 0.5 which gives 2*0.5²) and the minimum gini 0 (the frequency of one class at 0 and another class at 1).

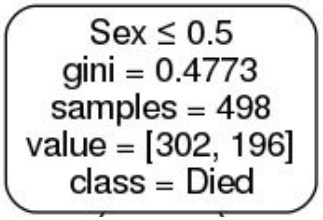
We have 498 people in our training set, out of which, 196 survived. The frequency in the classes *died* and *survived* are thus:

$$f_{died} = 302/498 = 0.606$$
$$f_{survived} = 196/498 = 0.394$$

From which we can compute the gini of our training set:

$$I_G(f) = \sum_{i \neq k} f_i f_k$$
$$= 2 * 0.606 * 0.394$$
$$= 0.477$$

Let's build a simple tree using sex, Pclass1, 2, and 3

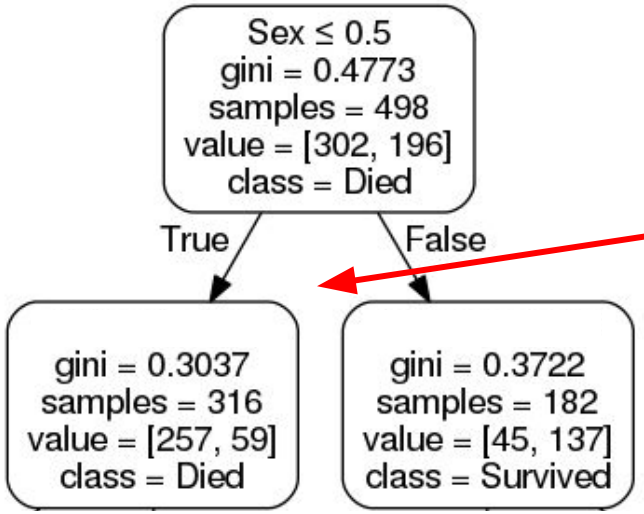


We thus first decide to split the dataset by sex

We start with the initial node with [302, 196] split giving us a gini of 0.477. We then consider each of the possible variables we could split the dataset by:

- by Sex: sample=(316, 182), gini=(0.304, 0.372), wavg=0.329
  - Which gives us two nodes, one with 316 people (men) with a gini of 0.304 (257 of them died out of 316) and 182 people (women) with a gini of 0.372 (45 of them died out of 182). This gives us a weighted average of 316/(316+182) * 0.304 + 182/(316+182) * 0.372
- by PClass1: sample=(369, 129), gini=(0.416, 0.439), wavg=0.422
- by PClass2: sample=(377, 121), gini=(0.467, 0.497), wavg=0.474
- by PClass3: sample=(250, 248), gini=(0.490, 0.336), wavg=0.413

Let's build a simple tree using sex, Pclass1, 2, and 3



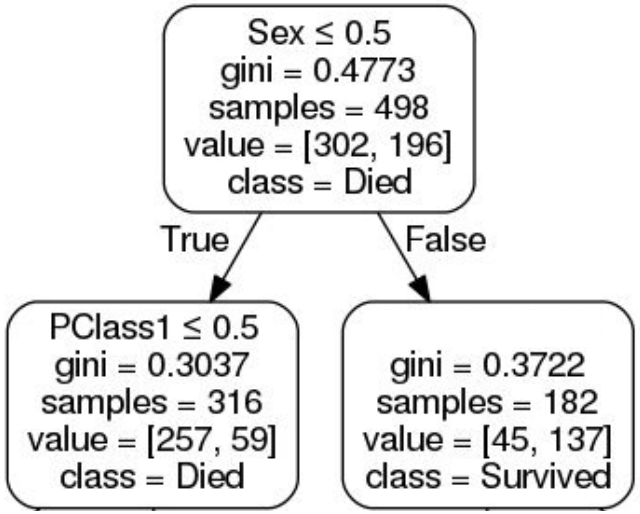
For men, PClass1 results in the lowest weighted average. We therefore decide to split the men now by PClass1

Then, for each of the two nodes, we need to decide which variable to use to split.

For the men:

- by PClass1: sample=(244, 72), gini=(0.203, 0.490), wavg=0.269
- by PClass2: sample=(247, 69), gini=(0.328, 0.205), wavg=0.301
- by PClass3: sample=(141, 175), gini=(0.400, 0.202), wavg=0.291

Let's build a simple tree using sex, Pclass1, 2, and 3



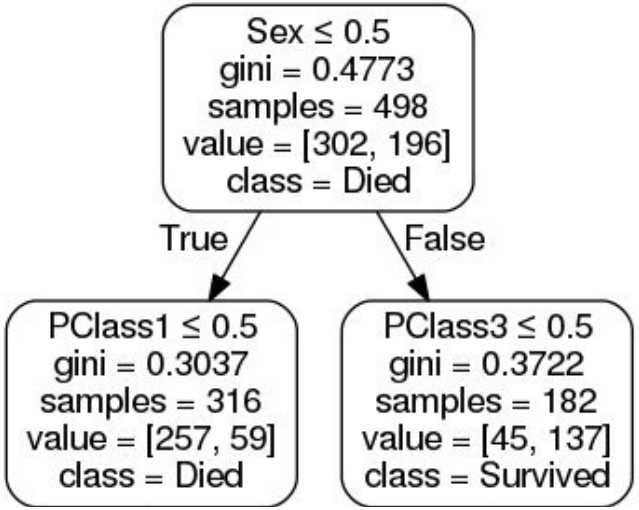
For women, PClass3 results in the lowest weighted average.

Then, for each of the two nodes, we need to decide which variable to use to split.

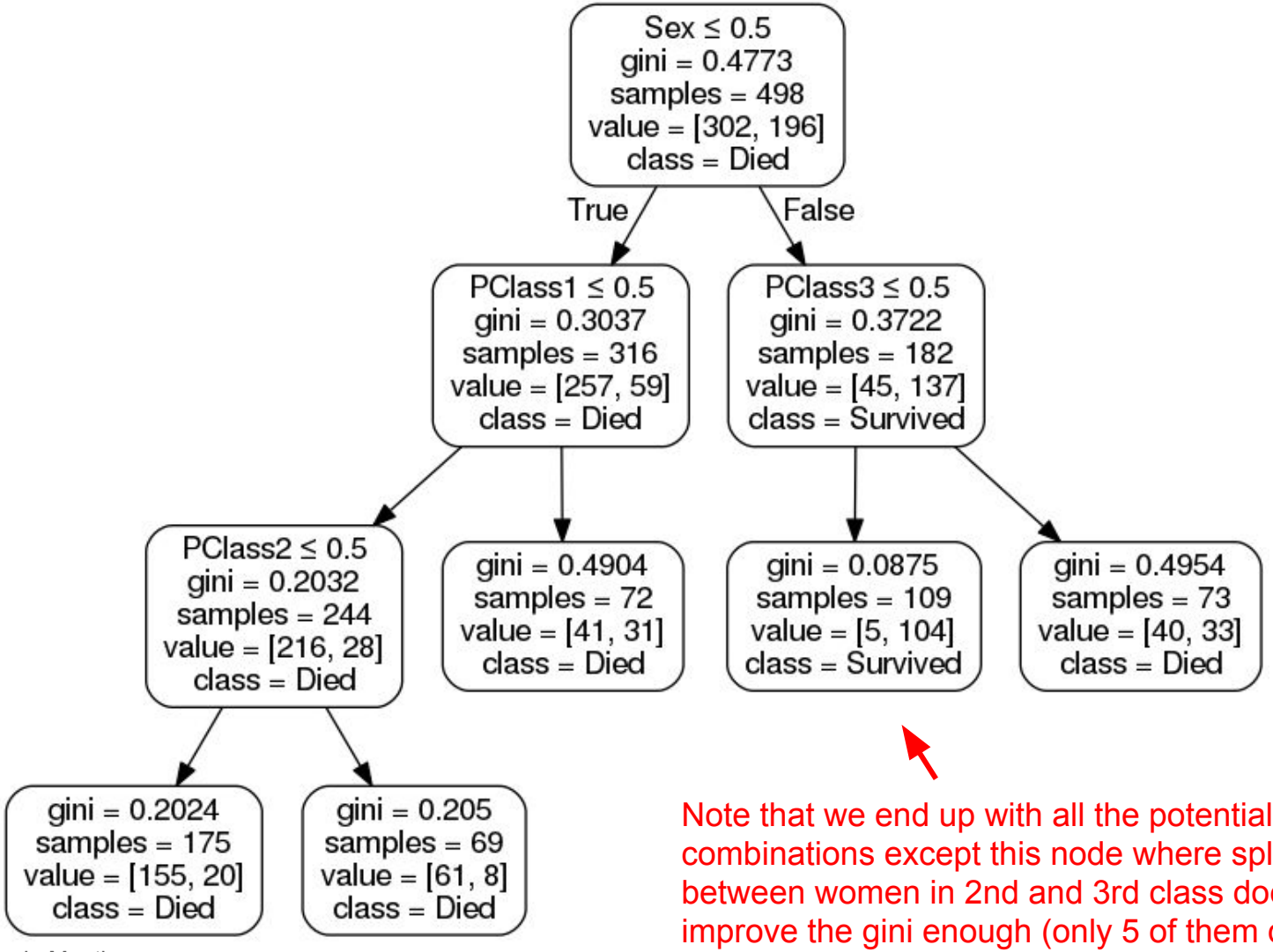
For the women:

- by PClass1: sample=(125, 57), gini=(0.456, 0.034), wavg=0.324
- by PClass2: sample=(130, 52), gini=(0.432, 0.142), wavg=0.349
- by PClass3: sample=(109, 73), gini=(0.088, 0.495), wavg=0.251

And we can keep going



Until we have a full tree



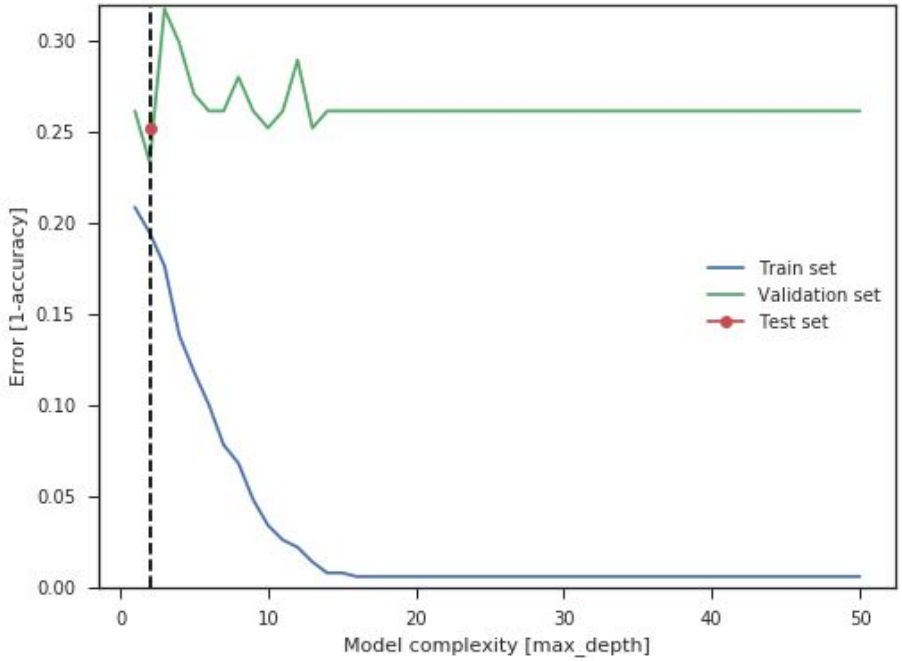
Note that we end up with all the potential combinations except this node where splitting between women in 2nd and 3rd class does not improve the gini enough (only 5 of them died)

But, as always, beware of overfitting

The error on the train set (here 1-accuracy) **will** keep going down as we're making the model more complex (e.g. increasing the depth of the tree)

We therefore use the validation set to explore the set of possible (hyper-)parameters for our model (e.g. the depth of our tree (here 2), etc)

And, then (and only then) we look at the accuracy on our test set (here 0.252), slightly worse than the error on our validation set (0.234)



Yves-Alexandre de Montjoye
Imperial College London
Computational Privacy Group
deMontjoye@imperial.ac.uk