**Week 2 - Generalisation**

0:00  
Hi all, in this video I'm going to talk about generalisation in the context of machine learning.

0:07  
So generalisation is the ultimate goal of machine learning and you want good performance for a model for new data that's data that hasn't seen before.

0:17  
How good the generalisation is is how well it performs on perceived or on previously unseen data, IE data that's not being used during the training set.

0:26  
It's no good if being able to predict stuff that was or predict perfectly stuff it was trained on.

0:32  
You want it to be able to predict stuff that it's never seen before.

0:36  
So the definition is the is that generalisation is the model's ability to give sensible outputs to sets of inputs that I haven't seen before.

0:44  
There's different ways of measuring this error at how good it is.

0:47  
So you could use root mean squared error or the OR squared metric, and we'll look at different error metrics again in the future.

0:54  
But it's similar to the idea that we looked at when we looked at linear regression of checking how far away the predictions of the model are from the original data, from the original outputs Y that are in the data set.

1:08  
But what we're going to like look at in this video is what's the problem with the approach that we've been using so far?

1:15  
I think what's the approach at looking at the outputs Y and are the, yeah, the outputs in the data set Y and the prediction and comparing them.

1:25  
So we're going to investigate that problem and the problems of overfishing and underfishing that are that are related.

1:32  
So one of the issues is that we're checking the generalisation of the model by using data that it was trained on, IE that data that's not previously unseen.

1:42  
As a result, a model might be overfit to the training data.

1:47  
So to give an example, it's kind of like the student who's studying for an exam is just memorising essays or memorising answers to particular maths Quish questions as opposed to understanding or being able to apply that knowledge to new contexts.

2:03  
So another example that just pops into my mind is say you're training a radiographer.

2:07  
Let's say that you train them on 100 biological scans of a tumour and half of them in half of the scans were malignant and half of them were benign.

2:17  
And the radiographer sat down with the scans.

2:20  
You know, they could see the scans and they could see the result of the scans from previous experts diagnosis or something like that, or maybe a biopsy diagnosis.

2:28  
They could see which ones were malignant and which ones were benign.

2:31  
It wouldn't take much for someone with a really good memory if they were looking at these scans over and over again to just actually remember the individual scans.

2:42  
And then if you were to test them and test them on the same scans that you would train them, say you gave them the same 100 scans again and you were going to give them a percentage grade for their overall performance of how well they could classify, You know, those 100 scans that they've already seen during their training, chances are they'd remember a lot of them and they probably will get 100%.

3:04  
But that's no guarantee.

3:05  
You know, they might have just memorised it.

3:07  
It's no guarantee that in the future with scans that they've never seen before that they'll have 100% accuracy.

3:12  
So there's a flaw in evaluating a model using only the training set.

3:17  
And for that reason you also need a test set.

3:19  
And we'll explore why we need test sets and the other partitions of the data that you need to do.

3:24  
We'll look at that in this video for for different things we're trying to achieve while training.

3:34  
The opposite problem to something being overfished is that the model has underfit the data, so it often under fitting happens when you just can't get good performance at all.

3:43  
Even with the training data, you can't reduce your training loss to anything reasonable.

3:48  
It's like a student not being smart enough to do exercises or to do an exam.

3:52  
In machine learning it's generally a symptom of not having a complex enough model, for example not having enough layers or neurons in your neural network.

4:02  
Or to give again the analogy of the student, them not being smart enough to be able to solve the problem.

4:07  
However, this isn't something that happens, especially with students in the real world, as each of us, every human is given a massively complex model to work with, which is the human brain.

4:19  
So the human brain is incredibly complex with, you know, trillions of parameters and so on.

4:24  
It's much more likely, and it's the same with machine learning is that so in machine learning, it can be an underperforming model or an under a model that's not complex enough.

4:35  
But when machine learning, it can also be a poor training mechanism, as in the algorithm you're using to try and learn the parameters of your neural network or of your regression problem are is just underperforming or it's not up to the task or an inappropriate loss function.

4:52  
And to go back to the analogy of the student again, it's not that the model isn't complex enough if the training mechanism is poor, if the analogy is as a student who sits in the library all day long, who's not studying, looking at social media or doing whatever and can't figure out why they still can't do the examples.

5:08  
So often times it's like the human brain is incredibly complex and incredibly powerful, but it's the training mechanism, the building up of the knowledge that hasn't been done the right way.

5:19  
So look then at the definitions and the comparison between an underfish and an overfit model.

5:24  
Underfishing happens when a statistical model or some algorithm, the machine learning model can't capture the underlying trend of the data.

5:31  
It's either not complex enough, the algorithms aren't doing a good enough job, some problem with the loss function.

5:39  
Overfitting on the other hand, happens when a a model captures the noise in the data too well, kind of remembers precisely everything that it is.

5:48  
And we'll see some kind of more intuitively presented examples of what what exactly happens.

5:54  
And visually we'll see overfishing happening.

5:57  
It occurs when the model fits the training data too well.

6:01  
It's essentially it's memorised the data points as opposed to generalised the the data or drawn a kind of an approximate line or a model through it.

6:11  
Over foot fitting, as I said, is a result of an excessively complicated model.

6:16  
In terms of how to build a model, then it doesn't really matter what kind of model we're trying, what kind of problem machine learning we're trying to build.

6:22  
The process is very similar.

6:24  
First you have to acquire the data set, get it yourself, get it online, whatever it is, think a kind of a model to use, be it an SVM or a neural network or a regression model, train the data.

6:35  
And as part of the training, something new we're going to look at is choosing the hyperparameters to tweak the model.

6:40  
So in the in with regression, you might be looking at what kind of polynomial, what degree of polynomial you're going to use.

6:46  
With neural networks, it could be the number of layers of the neuron, the neural network, or the number of neurons.

6:53  
And the final thing is to test the model.

6:58  
No matter how good your model is and how good your algorithm is, the more data you have, the better it's.

7:03  
You can never have too many, too much data.

7:07  
Aside from having the one issue you can have sometimes is having too many features, independent variables, and especially if these features aren't contributing much to the to the output.

7:19  
So say, if I'm just thinking, say within, again, an example of trying to classify a tumour as being malignant or benign, like you could have some really irrelevant data about the patient or about the subject.

7:34  
I'm just thinking of like the number of Disney movies they've watched in their life or something that's completely unrelated.

7:39  
So if you happen to have that feature in your data set, that is going to give you issues because it's just introducing noise and it's going to make it harder.

7:47  
It's going to increase the size of the search space, the space of solutions of the neural network, and it's going to make it harder to kind of find a good solution because it's just this irrelevant data in the mix.

7:58  
So the if you can really do a bit of feature reduction and trying to find the most valuable features and only present that to your network or to your regression model or whatever it is, that's going to make things a lot better In terms of building a model, what we've been doing so far with SK Learn is we start off with the data with X being a matrix with all the inputs.

8:23  
2D matrix Y is a corresponding response variable.

8:27  
It could be continuous output in the form of a regression or a discrete output in the form of a classification.

8:34  
And then what we've the notation we've been using is that a particular Yi is the result of a particular XI where we've got a a number of these data points.

8:45  
So we've been importing our linear model using linear regression, fitting the data.

8:50  
And this is what we were doing previously is predicting the data based upon the training data.

8:55  
And this is the issue that if you're trying to measure the performance of the model based upon the data that you've traded on, you can run into serious issues.

9:07  
So with this right here, Lor was trained on the set of known information.

9:11  
So that's the X.

9:13  
And if we recall the predicted values aren't exactly the same as what is input.

9:17  
So we are going to get some error here between Y and predict.

9:21  
So it's not going to be the same, but it's hopefully going to be close.

9:24  
So you could test the performance by checking how far what's the difference using Whitney squared error or something like that or the L2 error, l ^2 error or squared error.

9:34  
Sorry, look at the difference between predicted and Y.

9:38  
But like I said, there's a problem with this.

9:40  
The problem being that you've trained already on this set of data.

9:45  
Overfitting is the case where where you've done a really, really good job fishing the model to the source data.

9:53  
So the error is really, really small between the source data and the model, but the generalisation of the model is unreliable.

9:59  
Again, graphically.

9:59  
We'll see a good example of this in a few slides.

10:02  
This is due to the model as learned too much from the training data set and it's done a super complex model.

10:08  
It's not really approximating the original trend or the trend in the data set.

10:13  
When this happens, we say that the model is low bias in that it hasn't made any assumptions.

10:18  
I'll talk about what bias and variance are on the next slide, but it hasn't made any assumptions about the data.

10:22  
It's just done.

10:22  
It's very, very best to finish, but there's high variance in that the selection of our data points that we use to train the model creates this high variability in the model that we actually get.

10:31  
Again, that'll be explained on the next slide.

10:34  
Underfitting is where it hasn't learned enough.

10:36  
So often this is the the reason of it is high bias in the original model that you've selected and your parameters and but also it gives you a low variance in the output, which is a desirable property.

10:48  
But so let's talk about what this bias and variance is all about.

10:53  
Let's talk about what bias means, first of all.

10:57  
So bias in the real world means that if you've made or if you have kind of inbuilt assumptions or you've made assumptions about a particular topic or theme or subject or something like that.

11:07  
So in our data here, if we look at our data, this is some data, some values that are ranging over time and we can see that the values seem to go down and then come back up.

11:16  
If we use a linear model to approximate these values, we can see that it's doing some sort of an approximation of it, but it isn't a good fit.

11:27  
So with this linear model, it means that we've a high bias and that we have made the assumption, a prior assumption that this data is linear without actually seeing the data.

11:36  
So we're assuming that the data is linear.

11:38  
That's an inbuilt bias that we have and it's an incorrect, incorrect assumption in this in this regard.

11:44  
So high bias models make a lot of assumptions about the underlying model and it can result in underfitting and we can see that we've underfit the data here.

11:52  
The flip side of bias is that you have a very, very low bias and you haven't made any assumptions at all about the model.

11:58  
You can see here that it's done a really kind of finely tuned job of like fitting the data really exactly in a really kind of low bias way.

12:06  
It's it's like, OK, it's assuming that the curve is almost, it's like estimating the noise of the model.

12:11  
Nearly the model is trying to fit the noise.

12:13  
When in reality this is a more balanced example in terms of bias.

12:19  
It's kind of it's, it's got a probably a fair bias in that it's approximating the data and it's making assumptions that are valid for the particular underlying data in terms of variance.

12:31  
Variance means the sensitivity of the model to the training data.

12:34  
So it means that if we were to select a subset of this, if we just split this data between our test set and our training set, for example, you know, if we were to try and fit this model and we were to like randomly choose 70% of all these dots or something like that, this model will come out almost always exactly the same.

12:50  
It will be this straight line.

12:51  
You know, if you got a real fluke that you just picked the 70% from here to there or something like that, you might get a slightly different line.

12:58  
But if you randomly sample this data and you pick 70%, you're always going to get the same line, more or less.

13:04  
If in this case, you were to randomly sample these points, this line would change quite a bit by and we'll definitely see what polynomial set.

13:12  
If you were to like eliminate a lot of these things, the line would look quite different, you know, more different than the the difference you have here.

13:19  
So again, we've got this trade off between bias and variance.

13:24  
What is the right measure of both?

13:26  
So depending on the model that we're going to use, what you want is some performance that lies between over fitting and under fitting is the most desirable and you want something that approximates the underlying trend of the data, but not going so far as to capturing the noise that's in the data because there's always noise in the real world.

13:43  
Like I said, with the breast cancer, some things are just like random.

13:46  
No matter or any sort of tumour, no matter how well you live your life or how much healthy eating or exercise you can do, there still are noisy factors that can come in and you know, 'cause you to get sick.

14:02  
It's just, it's just there is a noise component as well as there being predictors to these things, there's also just noise.

14:07  
And you don't want the algorithm in the training phase to be learning the noise.

14:12  
The trade off is the most integral aspect of model training is trying to not get an overfit model.

14:17  
And this video is going to look at how we can do that, how we can try and tune our parameters, tune our training so that we're not ending up with something that's overfitting and doing a bad job.

14:30  
Generalisation is bound by these two undesirable outcomes, both where you have a high bias and a high variance.

14:37  
So high variance is obvious that you don't want the model that you get to be highly dependent upon your selection of the data.

14:43  
That's bad news.

14:44  
And also high bias is where it's really learning the noise.

14:47  
It's really making a lot of assumptions about the data.

14:49  
It's not like they're trying to build a general model.

14:52  
Detecting whether the model suffers from either one of these is the sole responsibility of the model developer, and if part of the training, you need to account for this when you're building the model.

15:01  
You need to be watching out for these things and tuning the hyperparameters and tuning the training so that you stop at a particular time before you get to finely tuned.

15:13  
One of the major problems with overfitting is that it doesn't just misclassify new data, but it often does so with a very high confidence.

15:21  
And we'll see again in our visual example where this is the case, like here, the data is so well fitted that it's very, very confident all the way through that it's able to do a good job.

15:32  
It's like a super confident model that thinks it's doing great while here at least it knows it can always look at the difference between the model and the original Y values.

15:42  
Same here I can, I can say, oh, I'm amazing.

15:44  
I've got like almost a 0 error.

15:46  
But here it can be like actually I'm doing a pretty good job in this range of time and I'm doing not such a bad job in this little window of time.

15:54  
But here I'm doing a pretty bad job and here I'm also doing a pretty bad job in this window of time.

15:59  
You know, I'm way off here.

16:01  
So it can kind of an estimate of confidence that in certain regions of time it's like, OK, this is my guess, but I know it's a pretty low confidence guess.

16:08  
You can give a measure of confidence with the prediction.

16:10  
Similarly here, I think this is where the predicted value is, but I'm pretty sure it's quite wrong.

16:17  
You know, so you can give you can you've got a more accurate estimate of confidence with this high bias or this like under fitted model, but over fitting, you get this big confidence coming in with the algorithm as well.

16:34  
When the model is of more modest complexity, it kind of knows that there's noise in the data.

16:38  
It may have certain samples in the training set that end up on the wrong side or are a long way from the model regression.

16:44  
But the model can report this.

16:45  
The model can say, yeah, like I predict this output in this range, but I'm only like 20 percent, 30% sure or like it's within this tolerance, this prediction.

16:54  
That's so that's an extra problem that you get with an overfitted algorithm.

16:59  
It's getting really good scores and thinks it's brilliant and it can often be really, really wrong.

17:06  
So if we can't test our model on our original training data, how do we do it?

17:11  
The way we do it is using a test set that we know the answers for already.

17:15  
And we normally do this by extracting it out of our data set at the beginning, splitting our data set into a training set and a test set.

17:22  
It doesn't matter if if it's our output is a continuous number and a regression or a discrete number in a classifier.

17:31  
It's a similar idea.

17:32  
We might use different formula for accuracy and performance.

17:35  
We've seen or squared we've seen or MSC.

17:38  
We've seen a few different ones.

17:41  
It's the same principle.

17:42  
Our test set is always kept separate from the training set.

17:45  
When it's used then to check the performance of the model, it's brand new information that the model has never seen before.

17:51  
We still know the correct answers.

17:53  
So we can verify like how close is the model to the actual the the model's predictions to the correct answers.

18:00  
And then we can compare the predictions with the results of the test set.

18:04  
And that's a really accurate way to check the performance of the algorithm.

18:09  
So this is a good strategy, but we'll see in the next couple of slides how even this strategy can fall down.

18:16  
We need to, not that it falls down, but we need to further develop it further, especially when it comes to parameter hyperparameter tuning.

18:23  
Anyways, test sets often times when you're given a problem like say in a machine learning conference, and these conferences happen all over the world and often some of them there's annual machine learning competitions of like how well your model can do a classifying X wires at, you know, there's image classification problems for deep learning, neural networks and all these things.

18:42  
They always keep the test set very, very secret.

18:45  
The organisers of the conference that they won't reveal it until everyone's in the room with their models.

18:50  
They'll all of a sudden make it available, get them to run it through the model, and only then the the training set would be made available in advance.

18:58  
So people have months to try and build their model, but the test set is always held secret until that day of the competition.

19:04  
Training set is used to fish, test set is used to evaluate the model and check how well it generalises.

19:11  
Often you're only given though one set of data, like if you download a data set online and then you have to decide the best method to partition the data set into test and train and also ultimately into what we'll see the evaluation set as well.

19:23  
Coming up for parameter tuning.

19:25  
If you want to split your data before we get to evaluation data, we'll just look at the test and train split.

19:33  
Say if you've got a sample size of M, each item is in the data is denoted I and we know that the X ith vector, that's the vector of all the features for a particular sample.

19:42  
And we know that the Yi is the response of that.

19:47  
So we looked at an example of we'll see it again with cars where you know, maybe you can predict the miles per gallon based upon like the number of cylinders in the car, the weight of the car, the engine displacement, all these kind of features.

20:00  
So we split the data into two sets, training set and a test set.

20:05  
It's always good to have your training set significantly larger than the test set.

20:09  
8020 or 7525 are good splits.

20:12  
And then what we do is we'll randomly, and this is really important to do a random select which items are in the training and which are in the test.

20:19  
Often when you download a data set, it might be ordered by.

20:24  
You'd see like say if you ever looked at the Irish data set where it's all the versicolors, all are all together and all the Virginia cars are all together and all the get the blank of what the last one is.

20:33  
They're all.

20:34  
So oftentimes if you just take the last 20 or 25% of the data, you're going to get all one type of flower.

20:40  
Or similarly, if you have data that's broken down by like socio economic class or by whatever, if you just isolate the last 25% as a training set or as a test set, chances are you might exclude a whole set of data from the training process that you won't train on a certain set of flowers or a certain socio economic class or pattern of weather or what, whatever it is.

21:01  
And then you're going to test on that something that's never seen before.

21:04  
So it's really important to randomly shuffle your data, fit the model of the training set, evaluate on the test with SK Learn.

21:12  
The way you do it is you import this train test split, which is a really handy function that's built into SK Learn that breaks out the two data sets.

21:23  
You end up with your X train and Y train.

21:25  
That's your training set inputs and outputs, and your X test and Y test, which is your test set inputs and outputs pass in your original X&Y and your breakdown.

21:34  
You can also do it like this where you specify the train size or the test size, and these values that are returned are always going to be numpy arrays.

21:45  
The full model procedure looks like this, where you have your original data set, your XS, and your resulting YS.

21:50  
Split them into training a test.

21:52  
Your training set goes into your model builder, which you'll eventually build your model.

21:58  
As you're building your model, you're always kind of evaluating it and see how it goes, But eventually your test set is going to be input.

22:03  
You'll get predictions, and you can get a, a, a measure of performance off the back of that.

22:11  
Something's to note about the test set, like those conferences do, you always need to protect the test set carefully from yourself, from the training process and only use it at the very end, almost never used aside for the final evaluation of your model.

22:25  
This is where it's going to get a bit, maybe stuff new stuff that you might not have seen before.

22:30  
It must never be used to choose any hyperparameter.

22:32  
And I show how it's a very easy trap to fall into to use your test set to tune your hyperparameter, but that corrupts your test set or doesn't corrupt in a kind of taint set, let's say, where so say by hyperparameter.

22:47  
What I mean by hyperparameter isn't so much the coefficient of the weights of saying a linear regression of like MX plus Z or West zero and West one or even the weights in a neural network.

22:59  
What the hyperparameter is is the kind of parameters about the model.

23:04  
So with a neural network, it could be how many neurons are in every layer, how many layers there are at the overall architecture of the neuron.

23:11  
That's not stuff that's not being learned, it's stuff that you kind of tune and set up from the start.

23:16  
With regression, it can be the degree or the order of the polynomial regression.

23:21  
We've only really looked at linear regression so far, but we're going to look at polynomial regression where you have additional terms over and above linear regression.

23:30  
If you were to use the test set, and it's super tempting to do it during your hyperparameter selection process, it no would no longer be unseen data.

23:39  
You use it only for the overall evaluation at the very end.

23:44  
And like I said, competitions, they keep it very secret, very private until the day of the final evaluation.

23:52  
It's all done under very controlled conditions.

23:55  
Just as an example, kind of the challenges that there are when you're trying to break up test sets and training sets.

24:03  
Say you've got 3000 emails that are tagged as spam or not spam.

24:07  
XI is all the kind of features could be the number of times a word and the e-mail as mentioned or something like that.

24:12  
Target set is either 0 or 10 for not spam and one for spam.

24:16  
So each output is going to be either a zero or one.

24:19  
What we do is we split it into a training and test.

24:21  
We Phish, we evaluate.

24:23  
But what happens then if we have way more non spam versus spam emails?

24:28  
So imagine in a worst case scenario where only maybe 5% of your emails are spam and 95% are non spam.

24:36  
So imagine you developed a classifier that just put a 0 as the output every single time.

24:42  
And imagine even if it was 5% you were lucky and you managed to get 5% into your training set and your test set you by just building a classifier that only inputted zero that didn't even look at the input so it just had a there's literally outputting 0.

24:53  
Just Y is equal to 0 or prediction is equal to 0.

24:58  
You could get 95% accuracy on your training set and on your test set.

25:03  
So that shows the importance of really, really balancing, having balanced data sets.

25:08  
And then if you look at balancing data sets, everything kind of gets tricky because you need to represent sort of samples of each.

25:14  
You know, you might have certain percentage of spam emails, certain percentage of non spam emails.

25:18  
What about your spam or your non spam emails that are related to work non spam emails that you're related to, like promotions you're interested in your home life friend emails from friends, you know, spam emails that look like different friends spam emails that look like that they're crypto spam emails or other finance scams or other types of spam emails.

25:38  
Your test set can balloon very quickly.

25:40  
So it's important to scope your problems so that you're only maybe trying to discriminate between, you know, certain types of spam e-mail or whatever it is the way you notice or see under fish the way you detect it.

25:52  
First of all, So what you when you are?

25:58  
So often people compare the score of the training set versus the score of the test set to see if this is happening.

26:03  
So you fit your data with your training, get your training score, get your test score by passing in your test set.

26:10  
If your training score is poor to begin with, chances are you've just done a good or a bad job at model fitting and the model is under fitted.

26:19  
Something wrong with maybe the model the way it's set up.

26:21  
Maybe something wrong with your error metrics, maybe something wrong with the algorithm you used to tune the problems.

26:26  
Something's wrong.

26:28  
What's a bit more hard to detect sometimes is that sometimes you got a really good training score, and almost always your training score is better than the test score.

26:36  
But if you see a big difference between your training score and your test score, as in the training score is much better, then you've got a situation where the model is overfished and that's learned the noise as opposed to learning the underlying trend what we do.

26:50  
So it's a constant issue to look out for this.

26:52  
But there's some common techniques, especially when it comes to hyper parameter tuning.

26:57  
So for under fitting, you need to either train for longer, actually with hyper parameters.

27:01  
I'm going to come to that.

27:02  
If you're not sure what a hyper parameter is that's going to it's, it's coming in the next while.

27:05  
Once I've covered this under fitting, over fitting stuff for under fitting, you need to train for longer, use a more complex model, you know, loss function, all these things you need to tune and look at over fitting.

27:16  
You could reduce the complexity of the model, maybe take out some neurons or make the polynomial and regression a little bit less complex, Decrease the number of features, take out the random features that aren't, you know, given any good information, especially if you have lots of features and not very much data.

27:31  
But if you're throwing away information, you just got to be careful.

27:35  
You got to just make sure you're not throwing out something that's actually important.

27:39  
That's a hard thing to to determine.

27:40  
Sometimes lower solution is get more data if possible.

27:44  
And then often with neural networks you can use a technique called regularisation.

27:47  
I'm not going to cover that in this video, but we'll cover it again in the future.

27:52  
Deciding the complexity of the model is a big decision with neural networks and often it's described as going to be in a black arch, you know with how many neurons, how many layers to use.

28:04  
But there are techniques we can use and especially what we'll look at this week is techniques you can use with regression to decide the complexity of the regression model.

28:13  
So when you have simple models that only take minutes or hours to train, you can basically do a parameter search over the complexity of the model, A hyper parameter search where you can try and afford to try.

28:24  
You've got time, lots of different model sizes and you can look at the kind of performance, the under fitting or over fitting performance of each of them.

28:31  
But you can't just pick the wrong or the one that gets you the lowest training error because we've already seen the problem with that.

28:39  
We just can't look at the training performance because if you have a really low training performance, the model could be over fish got a really bad test set performance.

28:46  
So instead we need a mechanism to check the loss on the data on data or the loss on data that the training algorithm hasn't seen yet.

28:55  
So what is tempting in that case is to use the test set.

28:58  
It's like, OK, we'll train it using the training data.

29:02  
We'll keep for every level of complexity of our model.

29:05  
We'll just keep adding and adding complexity.

29:07  
We'll see our training data gets better and better and better.

29:10  
But we can look for over fitting by looking at the performance of the test data.

29:13  
And if the test data starts to get the error starts to get worse, we know we're getting into the territory of over fitting and we need to pull back on the complexity of our model.

29:22  
This process is called hyperparameter tuning.

29:25  
But what I just described there as in using the test set to figure out when to stop increasing the complexity of the model isn't OK.

29:36  
What you're doing is you're exposing in that process the test set to the training process and you're tainting the data.

29:43  
So what we need to solve this is a per set of data, something that we can essentially use as a test set during the hyperparameter training process, during the like selection of the complexity of the model, the number of neurons, the degree of the polynomial and so on.

29:59  
We need an extra set.

30:00  
And we're going to come to that in a minute.

30:01  
But first I want to talk specifically about what these hyperparameters are.

30:05  
Hyperparameters are parameters that are chosen and often chosen before the training happens rather than learned.

30:11  
What's learned, say in the simple example of linear regression is you're learning the slope of the line and you're learning that C, the intercept, the Y intercept.

30:21  
So those things are learnt, but what's decided beforehand is to even use linear regression in the 1st place or should we use a kind of a higher order regression or something like that?

30:33  
So there's in in different models of different types of parameters that you have to choose and the degree, the kind of Holy Grail of machine learning is to find a model, A parameterless model.

30:45  
That's what like scientists are always looking for is harmony.

30:48  
If you come up with a class algorithm, a really good algorithm that can do amazing things, but there's like 20 parameters that you have to tune and have like expert melting level knowledge of the problems faced or whatever to do that.

31:01  
Like that's not so interesting.

31:03  
Even if it's the best algorithm in the world.

31:05  
If you've 20 parameters that really have a big influence on the performance of your algorithm, then that's not so satisfactory.

31:11  
So what you want is the fewest parameters, but all models nearly have parameters that control the complexity of the model.

31:17  
So what is the degree of the model?

31:18  
So with regression, you might have linear regression, which we've seen quadratic equation regression where you have an X ^2 term like linear is you have MX plus Z with the X.

31:29  
With quadratic along with just having an X term, you've got an X ^2 term where whatever the feature is, you're squaring it.

31:35  
And what happens when you square or cube those things like squaring like flips the signs of certain numbers and cubes might turn something from positive back to negative.

31:45  
So once you've got this kind of effect of like squaring or cubing, this makes things a bit more interesting for you.

31:51  
Just don't have a straight line like the linear regression, but you end up getting curves if you remember, I think from quadratic.

31:57  
I don't know if they still are, but definitely when I did the junior search in Ireland, quadratic equations were on the junior search syllabus.

32:04  
You were trying to figure out the like intercepts and stuff like that of a quadratic curve, which has like an upwards or a downward slope or whatever.

32:11  
We'll see some of these polynomials in a bit more detail in a few slides.

32:15  
So which features, you know, what degree of polynomial do we use?

32:19  
Also a hyper parameter is which features do you discard?

32:21  
Some features.

32:22  
That's another thing that you could do kind of an exhaustive search on of all the different combinations of parameters in your data set, which give you the best.

32:29  
And that's like a big space in itself before you even get down to the the nitty gritty of the the learning, learning ratios under the parameters, some algorithms, regularisation.

32:38  
I'll set a talk about that in the future.

32:40  
Number of layers, neurons in a neural network.

32:43  
Let's look at these polynomials for a minute.

32:46  
So what we have usually is we have something like Y is equal to FX, Y is equal to some function of X.

32:52  
And then linear regression, it's the simplest case of Y is equal to the function of X is MX plus C.

32:57  
That's the function we're operating on X and multiplied by X + C.

33:02  
Now I've mentioned before that that's kind of the algebraic way of writing the equation of a line.

33:08  
But that line can be extended into more complex polynomials where we just, as I said, don't have X, but we've X ^2, X ^3, and so on all the way up to X to really whatever power you want.

33:19  
So our function of X here and also the function of the weights is if we just stopped it here and I deleted everything from a here over from my mouse over.

33:28  
This is essentially the equation of a line.

33:31  
This is using the terms W0 and W1 instead of M and instead of C.

33:37  
But this is the equation of a line.

33:39  
Once we put in W2 and an X ^2, what we get is we get something called a quadratic equation.

33:43  
If I stopped that again and deleted everything to the right, this would be a quadratic equation, which is a curve that can go up and inflect to come back down or else can go down and come back up again.

33:54  
You would have studied that, you know, it's kind of junior start level maths in Ireland.

33:59  
What happens then is if you add another term which is X to the power of three, it becomes a stead of A.

34:04  
It's still linear.

34:04  
It's here if you have an \*\* squared term in it, it's called a quadratic equation.

34:09  
It can have a curve with one Inflexion point where the curve changes direction once.

34:13  
If you've X to the power of 3, the equation is known as a cubic polynomial and your cubic polynomial X to the three will have two Inflexion points, X to the 4/4 Inflexion points, so on all the way up to lots of Inflexion points.

34:29  
So this is our hypothesis space as in what form of model is the best to fit our data?

34:35  
Does a straight line actually just be the best fit?

34:37  
Or does, you know, is it just going up and going back down to the quadratic the best or maybe a cubic is the best?

34:43  
You know, how do we figure out which of these polynomial models are actually the best to fit our source data?

34:49  
Just saying again that W is our weights vector and this is the notation that we'll use mostly.

34:54  
One other thing just to say is like, you know, sometimes you see Y is equal to MX plus C and that's the equation of the line in algebra.

35:00  
In machine learning, what we normally refer to instead of MMX and C, we normally refer to them as the weights W0 and W1W2.

35:10  
But in maths and signal processing, which is more my background, well also a background in machine learning, but often the terms coefficients are used instead of weights.

35:18  
So those terms are used interchangeably and I will end up using them interchangeably all like in without doubt.

35:25  
I would sometimes refer to these things as coefficients W 0, W 1, W 2.

35:28  
Sometimes I refer to them as weights.

35:30  
Just as an aside there.

35:32  
So we've got our measures and successes we had before sum of squared errors, what mean squared error, like the R-squared error.

35:39  
There's different ones among the different functions we have to choose.

35:44  
Oh yeah.

35:44  
One other terminology, a mathematical terminology thing is with this degree of the polynomial.

35:50  
So this is a 0° polynomial.

35:53  
If you just have X, that's a first degree polynomial.

35:56  
X to the two is a second degree polynomial.

35:59  
X to the three is a third degree polynomial.

36:03  
So again, a second degree polynomial is also known as a quadratic equation or quadratic polynomial.

36:08  
A third degree polynomial will be known as a cubic equation cubic polynomial.

36:15  
This hyper or yeah, I'm talking about degrees here.

36:18  
But the other word I was always going to say is that you can use for a degree is order.

36:23  
So I would say this is a second order polynomial.

36:25  
A first order polynomial, which is a straight line 1/3 order polynomial will be a cubic.

36:30  
So degree and order are interchangeable.

36:32  
Often in maths as well.

36:34  
The degree of polynomial is the hyperparameter.

36:36  
And what we want is the degree of polynomial that captures the trend of the database.

36:41  
And we'll see a graphical example of this coming up soon.

36:44  
So looking at some examples of different orders of polynomials, if we were to look at which degree of polynomial is appropriate, imagine here we had these data points here in blue where we have our X and we have our Y here, which is our output.

37:02  
So we can see here that the ideal model is the green line here.

37:08  
This is what's being, you know, calculated by an expert calculation or something like this.

37:13  
But here is our red model.

37:14  
These are the model medals we're trying to fish.

37:17  
So if we've a zeroth order polynomial where M is equal to 0, we can see what a zeroth order part polynomial.

37:23  
If I just go back up to the definition of a polynomial, it means that we only have this term here.

37:28  
So the F of X or Y is equal to W 0.

37:31  
So WG 0 is just going to be set to some constant.

37:35  
It's like the MX plus C.

37:36  
It's like taking out the MX and just having the C and say if it's set to the number two or three, or in this case here the number 0.2 or 0.15, no matter what X is, Y is just going to be 0.2, so it's always going to be 0.2.

37:50  
I'm going to see it as a pretty bad approximation of the data here.

37:55  
If we move up to a first order polynomial, that gives us the order of the equation here, which makes it a line, not just a straight line like a zeroth order, but a line with a slope that adds in the M variable here.

38:08  
So this gives us a sloped line.

38:11  
Again, it's a better approximation than the straight line, but it isn't exactly the underlying trend of the data.

38:19  
There's more to the data than a straight line.

38:23  
So if we were go to go up, we actually find that M is equal to 3, which is 1/3 order polynomial.

38:28  
And if I you remember, a quadratic or second order polynomial only has one Inflexion point, while a third order polynomial gives you 2 control points or Inflexion points where the curve is being pulled up here and it's been pulled down here.

38:40  
Like if you imagine a piece of string that's a straight line there and you were to pull the string up here and down here, it gives you this kind of curved shape.

38:47  
And this is quite a good approximation now of the green of the model which is in green.

38:54  
But as you keep going up the orders, let's say we go up to an order of M is equal to 9 all of a sudden.

39:01  
Well, actually I'll jump back here for a minute, is that even though this is a good approximation of the modeller of the data, the model is a good approximation of the data, there still is error between the real world data, the observations and the model, both for the green and for the red.

39:17  
You know, there's going to be naturally a little bit of error there where we're just a little bit off.

39:22  
You can see this data point here is not quite on the modeller on the prediction, but here what we've done is we've got a really, we've gone to a 9th order polynomial.

39:31  
So that's something to the power of 9.

39:32  
And I don't know if I know of anything 2 to the power of 9 is 512.

39:36  
That's the only one I know.

39:39  
So I can see here that as we, you know, we're, we've got a model which is absolutely perfect in terms of matching all the training points.

39:49  
There's 0 error between the training set predictions and the training set actual points.

39:57  
But we can see here it's very far from the actual underlying trend or the underlying model or pattern to the data.

40:03  
And what happens here, and I said it before with the confidence, is that it's thinks because it's got a training error of 0 all the way across the range of X here, it thinks it does an amazing job of predicting something.

40:17  
So if it was to go to whatever this value is here, imagine this is 0.

40:21  
That's nought .1, and that's halfway between a nought point nought 5.

40:25  
It's going to give a value, a predicted value of nought point nought 5 of this could be like this curve could go all the way up here to like 30 or 40 or 20 or who knows.

40:34  
Similarly, we have no idea how low this curve goes.

40:38  
So this could give a predicted value of something like maybe 20 for nought point nought 5 when the real value is something like nought .3 or something like that.

40:46  
It could give something like orders of magnitude and also be very confident about it.

40:50  
So this is a good example.

40:52  
It's a very good visual example of overfishing.

40:58  
If we look at the RMS error across all the different orders of polynomial all the way up to null .9, we can see that the training.

41:05  
So looking at RMS, so this is our polynomial, we've got a variable order.

41:09  
So what we're going to look at is the root mean squared error for each M.

41:13  
That's for each order of polynomial starting at 0, going up to 9.

41:17  
And we're going to look at the performance of the training set, first of the model and also the test set, first of the model.

41:22  
And if you just look at the training set, this is what I was saying at the very beginning of the lecture, where you can run into trouble is that if you only look at the training set, you can see here that this looks like the 9 is the best of all.

41:35  
We've completely solved the problem.

41:36  
We've got a training error of 0, and we're coming down through, you know, bit by bit getting better and better.

41:42  
And eventually we converge on a perfect solution at 9:00.

41:49  
Likewise, if we look or however, rather if we look at the test set, we can see that we improved.

41:54  
The error gets lower as we come through up higher orders up until we saw a good model of three.

42:03  
But as we reach the value of nine, where we get a highly overfit solution, the training error gets really big.

42:13  
It goes up to one which is a really, really high training error.

42:16  
We're we're way off the actual model.

42:20  
So here this is the signal in which you detect overfitting.

42:24  
You can look and do your training and say, OK, this is all good.

42:27  
But in parallel, you can be looking at your test set.

42:30  
And we've already discussed how using the test set is a bad idea.

42:33  
So we're going to have to use something different than the test set.

42:36  
The test set is always hold back as if we use the test set.

42:39  
Here we're using it as a signal to determine this parameter here M the order of the polynomial.

42:45  
But we can definitely use this technique or a variation of it in order to see what's the best end.

42:50  
If I was to look here, I can see that the this is pretty much a plateau, and this is pretty much a plateau.

42:55  
So probably around an open order of three, anywhere between an order of three and eight.

42:59  
Often lower computational complexity like the simplest model will definitely the simplest adequate model will often give the best generalisation.

43:06  
But any of these models appear to be equally good.

43:09  
So the point is the model performance very, very well in terms of the training data at m = 9, but very, very poorly for the test set, meaning that it's over fish.

43:20  
So what we need is some methods of selecting the hyperparameters.

43:25  
And the reality is, is that there's lots of different methods out there for hyperparameter selection.

43:30  
And that specific area isn't the focus of this module.

43:34  
But we're going to look at one method which is using the validation set and and doing tuning to using feedback from the validation set will be demonstrated.

43:43  
But in the future, if you go into this specific area, you're going to have some more learning to do.

43:47  
Just a note there on that.

43:50  
So the validation set is about, we've already discussed the importance of keeping the test data separate and not tainting it.

43:59  
So what we got to do is split the training data now that we have into another further subdivision into a subdivision which is still going to be a training set and our validation set.

44:10  
So the training date is split into a training set and validation set and a 7030 split is often used.

44:17  
Next step is to train different models, for example different numbers of layers in a neural network or different order polynomials on the training set and then iterate for every iterate through all the possible hyperparameters.

44:31  
So for every hyperparameter, for example the polynomial order M is equal to 123 and so on, you fix the hyperparameter.

44:38  
So Seth one order parallel lenomial to begin.

44:42  
That's a linear regression.

44:44  
Train the model using the training set, validate the model using the validation set, and then store the results for the hyperparameter.

44:51  
So you're going to look at what the training performance is and what the performance of the model and the kind of unseen data, the data unseen in the training, which is a validation set.

45:01  
And we're storing the result.

45:02  
And then we go back to step one and repeat for the next hyper parameter.

45:05  
M is equal to two, M is equal to 3 and so on, where we store for every polynomial, same as what we have in the equation here, we're storing the performance on the training set and also on the test set.

45:16  
And once we have this data, we can look at both of these lines and determine where do we get the best test set performance.

45:24  
What's the polynomial that gives us the best balance between training a test and thereby avoiding overfitting?

45:31  
Choose the model with the minimum error on the validation set and then not always done, but often are probably recommended as well, is that since once you've decided on your hyperparameter, rather than just having your model trained on this training set and leaving out the validation set, you may as well go back and unify or merge both of these into a larger training set and train the model again.

45:54  
So retain the model, retrain the model on the complete training data.

45:57  
That's the union of the training set and the validation set with error loss.

46:03  
And with this technique, we have to be a little bit careful.

46:06  
Hyperparameters are parameters of the model of that aren't trained.

46:09  
They're not trained to during the fish methods, let's say an SK learn where we're doing a linear regression dot fish.

46:16  
Instead, they're set up or chosen a priori, that is, beforehand.

46:20  
As we can see with hyperparameter tuning, we're not exactly training them.

46:24  
This is with the tuning that we're using, but we're not choosing them either.

46:27  
So it's not necessarily a training, it's some sort of heuristic or some sort of method maybe to iterate over them to determine which hyperparameter is best.

46:37  
The tuning that takes place is in some way using the validation set data and it doesn't see the data directly during training, but it is getting a signal from it and in for that reason the validation data is compromised.

46:50  
So it doesn't give us a true picture of the generalisation of the algorithm.

46:53  
And for that we still need to have our test set held back so we can calculate a test error.

46:58  
So this is essentially just stating the rationale for why we need now three sets of data.

47:02  
We need our training set, we need our validation set to use for hyperparameter tuning.

47:09  
And then ultimately, once we've decided on our decided on our hyperparameters, we need our test set at the very end to to measure or to calculate the overall performance of the model.

47:22  
Test set needs to be guarded carefully from yourself.

47:24  
I've said that already only used sparingly and only used at the very end.

47:29  
Model hyperparameters are selected from the validation set.

47:34  
One way of using the validation set is that if you just use a static validation set and data isn't shuffled or whatever, that can introduce bias.

47:45  
So what if the validation set is not representative as in you know, it's at the end of the filing.

47:50  
You just pick the last 20% which might correlate to, like I said previously in the video, a certain socio economic class or a certain, you know, whether it's male or female or some sort of flower, whether it could be all the varsity colours at the end of the Irish data set.

48:07  
If we have a limited data set especially, this is more likely to occur.

48:11  
So the idea of cross validation is to do multiple runs of validation, not just a single one.

48:17  
What cross validation does is it creates S groups of data out of your training data and it uses s -, 1 of those groups to train and it uses the other one, the remaining 1, to validate.

48:31  
You do a number of these runs then by moving or changing which is the validation.

48:36  
So depending on the cross validation method, there's various ways to do it.

48:39  
K fold cross validation is the most commonly used one.

48:43  
You average all the scores for all the different validation training splits, and then whichever is the best out of all of those is the type of parameter that you choose which one has the lowest average error over all the runs.

48:56  
It's an effective method from for.

48:58  
This is really hyper parameter selection.

49:00  
You could also use it for model selection.

49:01  
If you're tuning, you know what type of neuron you're using inside a neural network, but it's slow, especially if you've got a big model and a big data set, models with lots of parameters.

49:12  
And that's why I said that parameterless algorithms are like the Holy Grail in machine learning.

49:17  
You've got this exponential number of runs that even if you decide with this technique of cross validation and with the validation set, which is the optimal set of features to input.

49:27  
If you had a big set of features like is in the maybe the diabetes data set or whatever the race, whether it could be 9 or 10 features.

49:34  
You know, if you do every single combination of them individually and every combination of them in pairs and every combination of them with in threes, fours, fives all the way up, you end up with a huge amount of and if you do a cross validation in all of them, it's a huge amount of experimental work that needs to be done.

49:49  
Different types of cross validation is 1 is leave one out cross validation, K fold cross validation, which is the most common one stratified K4 cross validation, adversarial validation, cross validation for time series.

50:01  
This one, the K fold is the one we're going to be using in this module.

50:05  
The way it works is that you randomly split your entire data set into K folds for each K fold in the data set.

50:13  
What you do is you build and train your model on K -, 1 folds and you evaluate on the K TH fold.

50:22  
You record the error for each of the predictions and you go through each until each of the K folds has served as the validation set.

50:30  
This is graphically presented on the next slide.

50:32  
It's a bit more clear to see the average of your K recorded errors is the cross validation error, and that's your performance metric for the model.

50:39  
That's how you decide on your hyperparameters and so on.

50:42  
Five and 10 are often just good choices for the number of folds.

50:45  
SK Learn defaults to five.

50:48  
This is with A10 fold cross validation.

50:50  
As you can see, the whole training data is broken up into ten folds and that validation set is kind of progressively moved through the data set where you train on the remaining data and validate on the data that's highlighted there in green.

51:06  
And this just kind of removes any bias in terms of the way the data or like even the fluke of selecting a particular.

51:12  
If you only did this once, it could be that your very first validation set is not representative of the entire sample or that your training set isn't fully representative.

51:21  
So this is just ensures that there's a nice, it's don't it removes any noise by doing it like this and calculating the average over all 10 rooms.

51:30  
So we're getting towards what we need for a final procedure in order to train our entire model, including hyperparameter tuning as part of that.

51:38  
First thing is just keep the test data separate.

51:41  
Then is with your training data.

51:42  
Iterate through all possible hyperparameters, say for linear or rather regression, polynomial regression, for linear, quadratic, cubic, and so on.

51:52  
When you get to for each of these, you fix the hyperparameter, say M is equal to 1.

51:56  
To start, split the data into K folds.

51:59  
Train the model using K -, 1 folds, and evaluate using the remaining fold.

52:03  
Record the error for both the training and evaluation, and you repeat until all folds have been a validation set.

52:10  
Then you take the average error over all of these folds that you do for a particular hyperparameter and store the results for that hyperparameter.

52:18  
Then you repeat from step one with the next hyperparameter.

52:20  
What you're looking for is the hyperparameter that gives you the lowest error on the validation set, the minimum average error on the folds.

52:29  
That's the averaged error on the folds.

52:32  
Not always.

52:32  
Don't have said this before, but it's a good idea is rather than just taking that model and run, running with it straight to the test set is you may as well not that you've got your best hyperparameter, your best polynomial order in this case.

52:43  
Go back and retrain the model on the complete training data.

52:47  
Go back and unify your evaluation and your training set and then go back, retrain with the new hyperparameter.

52:55  
You'll get a better model, which then you can go and use with your test set to find a final performance.

53:02  
A general kind of summary of this procedure.

53:04  
And this applies now not only to polynomials or even just regression is for all machine learning is split your labelled data into training set and test set.

53:15  
We're going to subdivide further subdivide our training set.

53:18  
Select your model.

53:18  
It could be SVM, support vector machines or neural networks or regression or nearest neighbours.

53:24  
Tune the hyperparameters using the validation set using cross validation.

53:28  
Then train the model with the full well.

53:31  
Once you've selected your hyperparameter with the full training set, that's the unification or the union rather of the validation and the training set and then evaluate the model using the test set.

53:43  
Looking at polynomial regression in Python, and I'm going to put up a Jupiter notebook which kind of goes through some of this.

53:50  
Your dependent variable is Y.

53:51  
That's always the output and the inputs are X.

53:54  
You've got a set of data X and a set of response is Y.

53:57  
And we, we want our model.

53:59  
Normally what linear regression is just is equal to Y is equal to AX.

54:03  
Ignore this X is equal to 0, but it's put in to kind of formalise the equation.

54:07  
The reality is that any number to the power of 0 is 1, so it's a 0 \* 1.

54:11  
So it's normally this would be MX because X to the power of 1 is just X + C, but this reduces to C It's a zero in this case because X0 is one that's just a 0 by 1, which is just a 0 AKAC, the kind of algebraic form of for the equation of the volume.

54:29  
So Y is equal to a zero plus A1X plus A2 X ^2 + a three X ^3.

54:34  
And what we want to do is X just say for a particular, you know, if it was a particular input for let's say in the cars data set where you're trying to calculate the miles per gallon.

54:47  
And one of the inputs, X might just be the number of cylinders on a car and the number of cylinders on that particular car.

54:53  
That X happens to be two.

54:55  
So imagine if we were just trying to calculate MPG in 2 dimensions just or like just based upon the number of cylinders for the car and that's the car.

55:06  
Some cars had four cylinders, some cars had two.

55:08  
But in the case of a car that had two cylinders, this would be one because 2 to the power of 0 will be one.

55:14  
This would be two 2 ^2 or this would be 4, this would be 8/16/32, and so on all the way up 64 and so on.

55:23  
So this is what happens when we kind of transform X, which is just a single number to the polynomial degree we want.

55:30  
We kind of see it here where we've got this array of 12345 and we transform it into one gets transformed into 11111 because it doesn't matter how many times you multiply 1 by itself, it just becomes 1-2 becomes 1248 because again, this to the power of 0 is 1/3 becomes 39279 by 9 by 3 is 27 or three cubed is 27/4 cubed is 64 or five cubed is 125 and 5 ^2 is 25 SX, which is just these values here 12345.

56:01  
This is our inputs or our features gets transformed into this bigger array, which is a polynomial.

56:07  
And this because more interesting when this negative values, that's why polynomials have curves in them and they can get bigger and smaller and so on.

56:14  
The weights then all of these are still linear because if you look A0A 1A2, is it still just a multiply and an odd.

56:23  
It just happens that these things here have got a power symbol in them, but these are pre calculators at the very beginning.

56:29  
The actual regression is that we're still trying to multiply this coefficient or this weight by that value, this coefficient by that value, that weight by that value, that weight by that value, and so on.

56:39  
And that's what what polynomial features we'll see in the the 5th transform does is it expands this from a kind of AA1D array into a 2D array where it's got all the polynomial where the orders or the degrees of that polynomial are pre calculated.

56:57  
And then there's a linear regression done on this.

56:59  
It's going to do K fold this lab book or the Jupiter notebook.

57:03  
It's going to do K fold cross validation or polynomial regression to choose the degree of the polynomial.

57:09  
That's going to be the aim, to do some hyper parameter tuning using a validation set.