**Week 2 - Validation Notebook**

0:00  
Hey everyone, so in this video I'm going to elaborate on the previous video on generalisation and validation in this Jupiter notebook.

0:09  
So in this notebook, to demonstrate the concepts of validation, cross validation, the risks of using the test set, I'm going to use the empty cars data set as a worked example.

0:20  
So there's bits of it that I'm going to ask you to do yourselves, bits that I've done already.

0:25  
And the task is to kind of read this, try running the cells and understand what's going on with them, get the logic behind the things I've tried looking at this test train or train test split function, comparing training scores to test scores, getting cross validation scores, making a selection on which features to use.

0:41  
And also there's an example here of polynomial regression, which I'm going to ask you to complete on a variable to see if it can improve things over and above a linear regression model.

0:52  
After this, then after I've done the kind of work on the Mt cars data set, I want you to do the same thing with the diabetes data set, which is built into Escaler.

1:02  
So first of all, looking at the Mt cards, which is a data set from a magazine back in 74, I think more the Motor Trend magazine to see what we can do with it.

1:11  
So what there is is there's an output, which is usually what's trying to be calculated, which is the miles per gallon.

1:17  
And there's a load of features.

1:18  
I think there's maybe 10 or 10 other features, which is the number of cylinders in the car, the horsepower, the waste, the axle ratios, quarter mile time and so on.

1:29  
There's a load of these number of gears.

1:32  
So what we're going to do is see which of these predicts MPG best of all.

1:36  
And we'll try also more than one feature to see if that improves the linear regression model.

1:41  
And what we should be doing, as always, is using separate sets for training and evaluating tests.

1:49  
So let's see what happens.

1:50  
So the first thing I'm just going to import the usual stuff that I import, I'm going to read in the data set, the CSV, and I am going to have a look at it.

2:01  
So what we're going to see here is we've got the kind of a blank or just an index.

2:06  
We've got the name of the vehicle and the data for each of the features.

2:10  
Then so Y, which is always going to be our output in the convention that we're using is our response or our output.

2:17  
So we're going to slice that off, which is the miles per gallon.

2:20  
So there's lots of different possible XS that we could use.

2:23  
You know, there's all of these things here.

2:26  
So what we want to do is capture just the X in a 2D array.

2:30  
So what we're going to do is slice off the 1st 2 columns.

2:33  
So we're going to get rid of the name and we're going to get rid of miles per gallon, and we're just going to have this data.

2:38  
So that's slicing off these two columns here.

2:40  
And now looking at the array that were left, which is all of our X, our input data, we can see here we've just got the subset of the overall data set which relates to the inputs.

2:51  
Next thing we need to do is try and split the data into a training set and a test set.

2:56  
And a very handy function that comes with SK learn to do this is train test split.

3:00  
So we're going to import it and then run it where we have to specify what our inputs are, what our outputs are, what we want of our as our test size, which is nought .2 and our Oh yeah.

3:14  
So this is a seed that I'm going to set up.

3:16  
And what the seed means is that every time I run this workbook or this notebook, it's going to give me the same answer and I'll push Then is these numpy arrays, the training set inputs, training set outputs, test set inputs and test set outputs.

3:36  
One thing just to note actually with this data set is that there's only 32 data points.

3:39  
It's actually observations samples, so it's actually quite a small data set.

3:45  
Anyways, let's come to that in a little while.

3:48  
So This is why I seeded it.

3:49  
Two different data sets now are subdivisions that of the big data set, train and test that we can use.

3:55  
So let's just try some different features.

3:57  
Looking at the data set and what's here.

3:59  
Not all of them was are obvious.

4:01  
It's a quarter second and this is axle ratio or something like that.

4:05  
Rear axles and horsepower waste looks like the obvious one that will be good for predicting miles per gallon like a big heavy SUV, even though they probably weren't around back then.

4:15  
It's going to be much more fuel hungry than a lice car.

4:20  
So let's have a look at that separating out just that column.

4:23  
We're going to fish the just extracting the training data for X train from the all X train.

4:31  
Just extracting the waste column and similarly just extracting the weight column to be our test set as well.

4:39  
What we've got to do next is build a model.

4:41  
So we're going to use, as before, a linear regression model.

4:44  
We're going to finish on the X train and the Y training data, where X train is a subset of the data.

4:50  
And what we can do is look at the scores of the model.

4:53  
So this is looking at the scores of the model on the training set.

4:56  
And we can see here it's got quite a good score, nought .77.

5:01  
The score of the test then is not quite so good, so nought .6879.

5:06  
So that's quite a notable difference between the training set and the test set.

5:10  
So maybe it's overfitting a little.

5:13  
See, let's see if there's any other features in the data set that do as good a job as predicting miles per gallon as weight.

5:20  
So we could look at horsepower again.

5:22  
You'd expect that a car with a higher horsepower would potentially drink more fuel than a smaller engine car.

5:31  
So again, we can do a linear regression where we've got our new X train and X test so we can fit the data and then we can look at the score on the training set nought .65 S not as good as what we had with our weight, not as good as a predictor and quite low really on the test set nought .39.

5:48  
So again, it's HP doesn't look or horsepower doesn't look to be as good a predictor.

5:54  
So it's even worse than weight.

5:56  
Not a good one to use.

5:57  
What about drash?

5:58  
I think this is some real axle.

6:00  
If I can remember what that is, let me see.

6:02  
Have it written down somewhere over here.

6:04  
It is the rear axle ratio.

6:06  
So we can see here again, extracting out whatever drash is into X train and X test.

6:12  
Do the same thing again, look at the scores.

6:14  
Even worse again.

6:15  
So we'd like normal .45 on the training and 46 on the test.

6:20  
So best we've tried so far was the weight.

6:23  
Anyways, let's see what would happen if we were to try two variables of combining weight, which looked OK, with horsepower, which was somewhat OK or well, not very good.

6:32  
But we can see that if we combine both of those again with a linear regression where we're fishing the training data or our training and testing data is now the weight of the horsepower.

6:44  
We can see that the model score now is up to nought .83, which if I look back to recall what it was for Model 1, we had nought .77 and nought .68.

6:54  
Just looking at weight.

6:56  
This was just the weight right here.

7:00  
So if I remember nought .77, nought .68, let's see what it looks like for the two.

7:05  
So we have a nought .83 and if we look at the test and nought .78.

7:10  
So that is a really quite a big improvement.

7:13  
It's the best we've had so far.

7:14  
So Model 4 is the best one that we've had.

7:18  
Now, if you notice what I've been doing, actually it's a natural enough thing to do.

7:24  
But what I've really done is I've contradicted pretty much what I said in the lecture.

7:29  
What I'm doing here and trying to choose my features is I'm using the test set.

7:34  
I'm looking, oh, look at this, the test set performance to figure out what is the best combination of features to use.

7:41  
And it's very easy trap to fall into.

7:43  
You know, it's natural to just say, ah, you know, have I over generalised or whatever.

7:47  
And it's very easy to figure like use the test set and seeing here we're essentially hyper parameter selecting, I'm not selecting what order the polynomial is or what, you know, order of linear regression.

8:01  
It is if I'm doing polynomial regression.

8:03  
But what I am doing, another kind of hyper parameter selection is actually selecting the features to use.

8:08  
So I'm using these test scores to make decisions.

8:10  
And I've just used this to make the decision that WT and HP, the combination of those are the best features that I have so far.

8:18  
And I definitely shouldn't be doing this.

8:21  
What we need to do instead is to use some sort of validation set instead.

8:26  
So look at an example.

8:27  
One of the things I'm going to import is cross validation score, this function.

8:32  
And if we look at what cross validation score does, we can look at the documentation is it's going to do, you know, you pass in an estimator, which is going to be our linear regression.

8:40  
There's various different jobs.

8:41  
And what it does is it estimates the score by cross validation rather than by what I'm doing here, which is using the test session.

8:51  
So let's try cross validation using weight again.

8:55  
So again, I'm going to use my modify that's going to be linear regression.

8:59  
The scores is what I'm actually going to do is cross validation now by passing in the model, passing in the X train and Y train.

9:05  
And what it's going to do internally is break up that training set into a sub, a smaller training set and a validation set.

9:13  
And it's going to run a couple.

9:14  
I think by default it runs five fold validation and you can see here, I know that because I can see 5 numbers coming at the output where it's done in five different splits and you see there's quite a variation in it.

9:25  
But what you do with your validation and to try and figure out what your best combination of, you know, what your best hyperparameter is, whether that's the best combination of features or the best order of the model is.

9:36  
You can, you know, look at the average of all these.

9:39  
So the scores don't mean, which is the average of all these scores is .57 when we're using weight as our feature.

9:47  
Let's try two variables.

9:48  
The one that seemed to work well for us before weight out of horsepower, we've got a model 6.

9:52  
Now we're going to go across validation.

9:54  
And I can see here, if I look at the scores, it's better.

9:57  
So it went from nought .57 to nought .63.

10:00  
Now this is a legitimate way to make decisions using cross validation.

10:05  
No, I'm not using my test set at all to try and construct or even tune the parameters of my model.

10:13  
So this is the way that it should be done.

10:16  
As I said, using a validation set instead of using the test set to make these decisions.

10:21  
So the two together have a higher cross validation score, nought .63 than just the ones, which means that we should take these two.

10:27  
But it hasn't been an exhaustive search.

10:28  
There's so many parameters there.

10:30  
There was a number of cylinders, displacement.

10:33  
It's a big task of trying to do an exhaustive search of all the various permutations and combinations of all of those features.

10:40  
Let's just try, for example, 3 features.

10:42  
That's not always the case, as I said in the previous video that more features give you better results.

10:48  
So say we already have WT and HP and say we throw drash into the mod, into the model and into the training.

10:56  
We can see that our scores, cross validation scores go to goes to naughty .65, which is a little bit better.

11:06  
But it isn't always the case that more features, as I said, is better.

11:09  
So this is actually better.

11:10  
I can see I've gone from a scores of nought .63 on my validation set to scores of nought .65 on my validation set, meaning that this extra feature has given a bit more information to the model and it's a good thing, but it's not always the case.

11:26  
It just happened in the above example.

11:28  
Let's try a different set of three features.

11:30  
So let's see, Take WT and HP, which is the ones that we found work quite well, and we'll throw in Carp, which is like carburetor or something like that.

11:39  
Can't see it off the top of my head, but anyways, we'll throw that in instead of drash.

11:48  
So we've tried the whole thing again and we can see that naughty .60 is what comes out, which is worse.

11:54  
So it's actually better having less data, WT and HP rather than this extra feature.

11:59  
So our best so far was this model with three inputs, Model 7, WTHP and rash.

12:05  
So what you normally do is you once you've selected your hyperparameters, that's selected your features of the order, your polynomial, then you normally go back and you train on the entire model.

12:17  
That's the union of the new training set and the validation set.

12:24  
So we put it back together, which is going back extracting our from our all X train and our all X test, our X train and our X test, which is not subdivided into validation.

12:35  
We run our linear regression on it, we fish the training data and then we test, you know, width.

12:43  
Now our hyperparameter selected test on the test on the final performance of the model shows that we can get a naughty .79, which is, you know, quite a good score for our test set looking and moving on into polynomial regression to see if we could do even better than linear regression.

13:02  
Now we're only going to look at polynomial regression with a single feature, but even to compare it with linear regression with a single feature as well.

13:10  
So what we're going to do is go back and look at weight as the feature and see what see what happens when we use a polynomial regression model.

13:18  
So some notes about polynomial regression of SK learn.

13:22  
What I have here is that Psychit Learn is the main machine learning library in pretty much out there.

13:29  
And it's amazingly good with lots of different learners, regressions and all sorts of neural network models which can learn and train parameters and models from data that you're provided to it.

13:41  
And there's loads of utility functions as well that come with it, which make life really easy.

13:45  
It used to be the case, you know, before this, even back when I was doing research.

13:49  
I should write all these functions yourself manually.

13:52  
It can be used by Python using an import S scaler.

13:56  
It's got a very well defined interface which is common across lots of models which make the library really a joy to use and it's definitely helped with its popularity.

14:04  
You know it as this paper.

14:07  
This API paper says that all objects share a uniform common basic API.

14:13  
This is all the different models and algorithm in there with three complementary interfaces, an estimator interface which we use for building and fitting the models, a predictor interface which is making predictions and a transformer interface, which is what we're going to use now with polynomial regression for converting data from one form into another.

14:30  
The estimator interface is the core of the library where all the models and the learning and everything happens and every model exposes this fish method which makes it so easy just to use the models.

14:40  
Sometimes you have to prop pass in some hyper parameters into the model via the fish, but it's just dot fish super easy.

14:47  
All supervised and super all supervised learning algorithms are just offered as objects just like near regression for implementing implement this interface machine learning tasks like feature extraction, feature selection, dimensionality reduction.

15:02  
We haven't seen any of that yet.

15:04  
You all we might also look at those as estimators.

15:05  
So we're going to use polynomial features and the polynomial features is going to allow us to transform the data.

15:12  
So it's common to modify or Philtre data before sending it into the learning algorithm.

15:16  
But some estimators like this one in the library give you a transformer interface and we'll see why this is necessary, which gives you some transform method to turn your X, which you have coming out of your data set into some other form that you feed into the model.

15:29  
It takes some data X and gives you a transformed version of X on the output.

15:33  
And what we're going to see here is where we have our polynomial expansion of X, where we're going to have, let me see what else it says here.

15:40  
We're going to have one feature X.

15:41  
And what we're going to do is X is going to be expanded.

15:44  
Say it was the number two to be \*\* squared four X ^3, 8, and so on up to the power are the degree D of our polynomial.

15:55  
What we have to do then is fit the coefficients or the weights as I said they're called sometimes, often the coefficients of these features in the polynomial.

16:02  
So we need could be W0 or A0 or C0 or whatever that notation is.

16:06  
I have a zero here plus A1X plus a 2X.

16:09  
And you see all this different language and different kind of conventions in different documentations.

16:15  
So what we've done is we've transformed a function of one feature, which is X weight and this into a linear function of many features.

16:23  
So what we have to do essentially, this is just an example of say our, our weight features are are 123 for three different cars.

16:31  
What we're going to have is we're going to have this matrix where one gets transformed into 1 to the power of 01 to the power of one, 1 ^2, 1 ^3, 2 to the zero, 2 ^2, 2 ^3, and so on for three.

16:45  
What we need to do though is 1-2 and three.

16:47  
Like an array like this won't go into.

16:51  
You need an actually a 2D array.

16:53  
And we talked about this previously where X needs to be 2D.

16:57  
So we need to do a reshape of our 1D array to make sure it's a 2D array and then that's our X as the input and then we can do.

17:06  
So that's what I have here.

17:07  
It's just that's what the reshape does is it transforms this into vash and this is the expansion.

17:16  
So what I just did is a demo of this where I've got a demo which is just my no π array.

17:21  
This is my X12.

17:22  
And three, first I do is reshape it to give me what I want, which is this 2 dimensional array where I only have stairs only one element in each sub array, but it still has a 2D array.

17:35  
And what I can do then is do this fit transform demo which what that does is it expands the polynomial we can see here 248163213927.

17:46  
So instead of just having one number weight, weight has now expanded into this kind of polynomial expansion.

17:53  
What I'm trying to do is for weight alone, I'm trying to fit multiple coefficients or coefficient for this, one, coefficient for this, one coefficient for this, coefficient for this, a coefficient for this, and a coefficient for this.

18:04  
And it's a linear regression of each of these because each coefficient is linear.

18:09  
But this is the expansion that I have the transform essentially that I have performed.

18:14  
OK, so let's see that in action with some code.

18:17  
So same thing as I had before.

18:19  
I'm going to extract my training out of all X train and just take the weight.

18:24  
Similarly, my text test is just the same thing extracted.

18:29  
What I'm going to do now is train with a polynomial feature with an expansion of two.

18:33  
This is an order of two polynomial.

18:35  
That's where I have a square term in it.

18:38  
And like I said in the previous video, square term is called a quadratic polynomial.

18:43  
So X trained Poly is polynomial features of degree 2 or order 2.

18:49  
And I'm doing calling this function fit transform and X train.

18:52  
This is what I did essentially here.

18:53  
This is an order of 5 S whatever 3 to the power of 5 is 243, didn't know that before.

18:59  
But anyways, 2 to the power of 5 is 32 here.

19:02  
What I'm doing is expanding everything to the order of two.

19:04  
So I'm assuming, making the assumption that's not a linear straight line, I've got some sort of a curve with one Inflexion point that's going to fit my data.

19:14  
So my model Poly 2 that's of degree 2 is a linear regression.

19:18  
What I'm trying to do now is fish the parameters to X train Poly and Y train.

19:24  
Y train hasn't changed at all, it still is just the outputs for the just using just for these samples.

19:35  
So what I want to do is print the score for this for the training data, print the score for the test data.

19:43  
And I can see with a degree of two, I can see that we've got a training set performance of .85 point 6-8 that kind of is telling me it's a little bit overtrained potentially or overfitted.

19:56  
So what we can do is to explore what is actually the best hyperparameter.

20:01  
Again, I've just selected 2 here at random made of quadratic, but maybe a cubic or A to the power of four or a 5th order or 5th degree polynomial might do a better job.

20:10  
So what we can do is loop over this parameter D the degree.

20:15  
The order of the polynomial and loop over all of them and rescore record the scores in an array to see what is actually the best polynomial.

20:22  
So what we'll do is initialise the array as all zeros.

20:25  
So what I'm doing is I'm setting up, I'm going to explore up to the Max order is going to be 10.

20:29  
I'm going to explore from first order polynomial up to 10th order polynomial.

20:35  
So that's my I'm just setting up some arrays here, which is zeros to hold the result for each of these things.

20:42  
So what I'm going to do is for every loop, I'm going to loop through I in degrees, which is I going up to 10.

20:48  
And I'm not checking a zeroth order.

20:50  
I'm just checking all of them.

20:51  
So what I'm going to check is an order of one, which is linear quadratic, cubic power of four, power of five, and so on.

20:59  
So what I do is I do my polynomial features expansion, which is pass in my I to figure out what I did here.

21:06  
Remember is passed in five in this case and two in this case.

21:09  
This is what the parameter I'm trying to detect.

21:12  
So I'll fit the transform X train and X test.

21:16  
I'm going to feel my linear regression.

21:20  
I am going to fish with the data.

21:22  
I've just expanded my transform data.

21:25  
I'm going to do a predict on my training data, a predictor, my test data.

21:29  
And then what I'm going to do is I'm going to compare my predicted outputs.

21:33  
That's my X training predicted outputs and tested predicted outputs to the original data to see how well the model is performing.

21:43  
And I'm going to look at two metrics, mean square error and also going to look at the score which is the OR squared error score is the one I'm most interested in.

21:51  
So if I look at the training scores, I can see here that these are my training scores across all the different degrees.

22:00  
So starting with the linear regression and so on.

22:03  
And what I'm going to do is I can also look at the tests.

22:05  
But the most the best way to look at this isn't really a set of numbers.

22:09  
It's actually to plot out what does the training performance look like and what does the test performance look like as I go work along the polynomials.

22:17  
So that's what I'm going to do here is I'm going to plot, look at plots.

22:22  
I'm going to keep going until the error are, Oh yes, I'm not going to look at the 0 value.

22:28  
I'm just going to go from error train one and error test 1, which will give us the degrees with the lowest error on the test set.

22:34  
This zero here is I just never, I skipped 0 here, if you remember, so didn't populate anything into this element.

22:40  
Remember I initialised everything to 0 here.

22:43  
So this remained A0 because I didn't actually do a computation on a zeroth order polynomial because it just wouldn't make sense.

22:49  
So it would just be a straight line.

22:51  
So this data is irrelevant.

22:52  
So I'm going to skip that data starting from here like I said.

22:57  
So what we're going to do is plush the training performance, plot the test performance, and what I'm drawing here is a line.

23:06  
What I'm doing is I'm looking at the best performance, which is the minimum.

23:09  
This is the minimum in my test error.

23:13  
So I'm.

23:13  
It iterates through the test error, finds the minimum of what you can see here is I'm drawing a line at that minimum.

23:18  
Minimum performance in green.

23:20  
I think G is green, Is that right?

23:22  
Yeah.

23:22  
So what I can see here is the best test performance that I got right here is actually with a first order polynomial that's a polynomial of 1.

23:33  
And I can see here that the test, the training performance continued to improve, but the test performance got worse.

23:39  
So if I was only looking at my training performance, my test performance is getting worse and worse and worse.

23:44  
So this right here a linear regression is actually the best regression to give me the best performance with this weight parameter.

23:54  
So simple linear regression, which is Y is equal to MX plus C or W0 plus W1X, works the best for weight.

24:00  
Notice how the training error here keeps getting better as the degrees increase.

24:05  
But this doesn't actually mean you've got a better model or a better test error.

24:09  
This is expected if you actually go and look or eyeball the data.

24:12  
And that's often the first step in machine learning.

24:14  
It just gives you so much information.

24:15  
It's just to look at it with some scatter plots or pair plots or whatever it is.

24:19  
If we scatter plot weight, we can see here that the data actually does look linear.

24:24  
What these high order polynomials are doing is they're fitting some bizarre really high dimensional curve, but lots of or high order or high degree curve, but lots of Inflexion points.

24:34  
And that's why your test error starts to go through the roof.

24:37  
So it's doing something really, really bizarre.

24:38  
You could have a plot of it and see what it looks like if you're interested.

24:42  
But we can see that actually a linear model is the model that best fits this data.

24:50  
All right, Can anyone tell me what I've just done wrong again here?

24:54  
I've done it wrong twice deliberately in this video.

24:57  
If you think about it, what I'm just after doing is I'm making the same mistake as I did before.

25:02  
I'm using my test performance and it just shows you how.

25:05  
I'm just want to show you how easy it is to fall into this trap.

25:08  
I am using my test error again to predict what is the best hyperparameter and it means that my test set is already corrupted.

25:19  
I have used information from my test set in order to decide what the best model is.

25:24  
Again, this is a bad idea.

25:25  
What the solution is for hyperparameter tuning is to use a validation set.

25:32  
So I'm going to leave some work for you to do here to actually go and do this.

25:35  
So I'll first talk a little bit about the validation set.

25:38  
I'll just recap a little bit on it, and then I'll leave you to do some of the work.

25:43  
I've given you some code suggestions on pseudocode, but I'm going to leave you to do it and also to replicate it on another data set.

25:49  
What we've done is in picking a given degree D as the best hypothesis is that I've just used the test set as information in my training.

25:58  
If you choose the best D based upon minimising the test set error, we have fish for the hyperparameter D on the test set.

26:04  
Bad idea.

26:05  
The test set error will now underestimate the true out of sample error on if say we have more data or new data and also we've contaminated the test set.

26:13  
That's the bottom line by using it to decide on D so it's no longer an uncontaminated or untainted test set.

26:21  
The solution to this is to bring in a new validation set of which are complexity parameter.

26:27  
This is the hyper parameter, D is fish, and we just keep out the test set.

26:32  
Just do not use it, do not touch it until the very, very end.

26:35  
And once you've evaluated on it, don't use that information to go back and do anything.

26:40  
That's your untainted, untouched test set that you only ever make that you don't ever make decisions based on.

26:48  
So leave out a test set.

26:49  
So what we do is we've got a subdivision, which is our test set, and with our training set, we subdivide that into a training set and a validation set.

26:57  
So what we've done is split the old training set here into a new, smaller training set and a validation set and holding out the test set for final testing.

27:06  
Obviously, we've decreased the size of our data available, and this is particularly relevant in the Mt cars.

27:11  
That's the reason I'm not really doing it on this data set because of only 32 data points or observations to begin with.

27:17  
So if I split this into like 25% test set, which is 8 and then maybe a 25% validation set, which is another rate, I'm only really left with 16 to train with, you know?

27:28  
So it's small and I'm going to leave you to do it, have a go with it and then try it on the on another data set, the diabetes data set, which is much bigger.

27:37  
It's got over 400 samples or something like that in it.

27:40  
But this is a price we have to pay for getting a good estimate of your hype or trying to figure out your parameters.

27:46  
And ultimately getting not even just a good estimate, but an accurate or fair estimate of your out of sample error validation process is what you do is you, you know, train it on different parts of the validation set.

27:57  
So you can shuffle this around.

27:59  
You loop over the complexity parameter D.

28:00  
So for each D, each complexity parameter D, try and do a fish on the new subdivision of the training set.

28:09  
So you get a best fit model where this this is just notation here where the minus indicates that you're just fitting your model on a new training set, which is a subset of the old training set, which is minus the validation chunk.

28:21  
Again, you don't need to worry at this point about this notation.

28:25  
You then test this model on the validation chunk and you get the validation error as we've spoken about before.

28:30  
And that's going to give you the information to tell you what is the best degree D to use.

28:36  
Then you move on to the next degree polynomial tried on tests, degree 2A, degree of three, a degree of four.

28:42  
Keep repeating the process.

28:43  
At the end you compare all the validation set errors, just as we were doing in the wrong way earlier with the test set errors and pick the degree D star which is the best validation set error.

28:56  
And you use that then hyperparameter to train your model ideally on the entire training set.

29:02  
That's the union of this and this.

29:04  
And then go and do your test set performance.

29:05  
Haven't picked your hyperparameter.

29:07  
D we train on the entire training set.

29:10  
Define the parameters are to find the best coefficients.

29:14  
And then we have.

29:16  
So this is again, we're training on.

29:19  
Anyways, you don't need to worry too much about this notation.

29:21  
The validation set then is the set which the hyperparameters fish.

29:26  
This method of splitting, I'm actually just going to take this out right here one second.

29:29  
We don't need to worry about this.

29:33  
The G yeah, take that out.

29:42  
This method of splitting the data is called train validation test split.

29:48  
So what we want to do is based on the training data and predict on validation rather than what I was doing above, which was fit on the training and predict or tune your hyperparameters in the test.

29:58  
So what we need to do now is subdivide our data again and we can use train test split again.

30:03  
So what we'll do is we're going to, instead of train test split your entire data, you're subdividing further the training data.

30:11  
So what we're doing is taking our training data, which is already separate to our test data and subdividing that further into a training with a validation taken out.

30:20  
So we've got an X train, AY train with this V and ish, meaning that we're doing a validation, an X validation and AY validation.

30:29  
And if we look at the shape of it, we're down to a small number.

30:31  
We're down to 18 samples.

30:33  
But here I'm going to hand it over to you giving you some pseudocode.

30:36  
I just said I'd do this to start you off where we're going to.

30:38  
You're going to train now on the smaller training set and fit for D or fine D or tune D on the validation set for each D as you iterate through in your for loop, exactly as it did before, you can store the mean squared errors are the R-squared error, which is the score and look for the one with the lowest validation error.

31:01  
So this one I'm doing here.

31:02  
I just gave you some pseudo code and I just threw this together.

31:05  
I didn't actually test this, but it's I'm going to set it.

31:08  
Leave it to you to try and get this up and running.

31:09  
But as I said, it's very, very similar before.

31:12  
What we have now is we've got an error on the training, an error on the validation score on the training and the score on the validation.

31:18  
That's the number of degrees that we're going to use.

31:21  
So for each, I can't remember, I think I have it up above degrees defined to be 10.

31:25  
So for each degree in D, what we're going to do is expand our polynomial features just like we did before, and we have our scores where we're going to do our training.

31:38  
What we're doing is expanding or transforming our training data and calling it XE.

31:41  
And we're expanding and transforming our validation data, which we're calling XE Val.

31:47  
What we do is we do our linear regression and we only do it now on the new smaller subset of training data, which is our XE.

31:53  
That's our new smaller set of training data.

31:56  
You'll have to go through this a bit slower maybe to really try and get this.

32:00  
And this is our smaller subset of training, the outputs of the training data.

32:08  
Then we calculate our error and our score.

32:10  
And then you'd go about looking over all the degrees, plotting it like it did before, and seeing which of these hyperparameters, which D essentially, or which order of the polynomial gives you the lowest validation score.

32:25  
And I'll have a graph, and I've shown you above how to do that.

32:27  
So plot the training error and the validation error against the degrees of the polynomial.

32:31  
I've done this above already.

32:32  
You just have to reproduce it.

32:34  
Show the really, this is the validation set error at the D which mini or which the?

32:40  
Yeah, the essentially the validation set error with the lowest D, So which is the talking about the kind of sub test here.

32:48  
So then what you do is you go back and fit on the whole training set data, which I have still stored up above.

32:53  
And you'll have to remake the polynomial features just like I did here on not only the validation or the smaller subset of training, but on the on the entire train, which I think is called X train.

33:05  
Yeah, that's it there the entire training data.

33:08  
Then test on the test set.

33:09  
You could try again then with cross validation.

33:11  
One of the things you notice here is that I'm only validating once.

33:14  
Haven't really done across validation multiple times.

33:18  
And again, it's a pretty straightforward thing to add.

33:20  
I've got an example of it up further in it.

33:23  
And this is getting towards a full procedure of how to not only train your model but also tune the hyperparameters of your model.

33:32  
So we're getting kind of the business end of machine learning now where we're able to tune hyperparameters in a systematic way, do it in a way that's careful and respectful to the test set using this validation set technique.

33:44  
Observe the performance over all the different hyperparameters, select the hyperparameters, retrain using the entire training set, and eventually get a really accurate performance of a really high performing model where you've been really taking the model and squeezed every bit out of it.

33:59  
You've optimised all the hyperparameters, you've got as good a result as you can get.

34:03  
Now to do an exhaustive search is really, really hard because you might have to set up, especially with long algorithms to take ages to train because all we're doing here is tuning the features or that mean tuning the degree of the polynomial.

34:14  
But what about the different combinations of features?

34:16  
It's quite hard and it's a sort of black art and one way to do this and it takes experience, but this is kind of the, the one of the fundamental ways of iterating through and essentially doing a brute force search of every potential, in this case degree of a polynomial.

34:33  
What I've also asked you can have a go at doing this is to try with the diabetes data set which has more samples in it.

34:40  
What I've done is loaded for you the diabetes data set in straight away.

34:43  
You can see there's 10 inputs, 1 output which is the compensative measure of disease progression.

34:52  
But nicely here you've got 442 samples to work with.

34:55  
You can do all the similar stuff that what I've done before with diabetes data set.

34:59  
You can try with single features by themselves.

35:01  
You can try with combinations of features.

35:03  
That's kind of the initial feature selection of which combination of features give you the best performance, and then pick one variable and see if a polynomial regression model, just like I did above can improve over a linear regression.

35:16  
There could be some features in there where the data isn't actually linear, that if you use a polynomial, there might be some curve to the data that would give a model that's better at predicting that a straight line.

35:26  
Linear regression is quite a crude method in one way.

35:29  
It assumes your data is linearly distributed when that isn't often even the case.

35:35  
OK, I'll leave it there.

35:36  
Talk to you soon.