Module 7 – A Model to Predict Housing Prices

Video Transcript

Video 1 (03:03) – Introduction

Hi. In this module, we're going to look at how we can make data useful. Jay Baer once said, 'There's an awful lot of data, but few insights.' And what we want to do is to be able to get insights from our data. Now, we're going to start off by looking at the normal distribution. And that's characterized by just two numbers: the mean and the variance. Now, when we take samples from a population, we get a good value for the mean, we can be quite accurate about the mean. But it turns out that if we don't sample enough from that underlying population, we can going to get a biased variance, and what we want is to correct that.

And so, there's a concept called the unbiased variance. And what we do is instead of dividing by n, the number of samples that we take, we divide by n-1. And that gives us an unbiased estimate for the variance. So, we're going to look at that detail, but then we're going to look at how can we make a predictive model? We're going to look at house prices. We've got data for over 1,000 houses, and we've got over 80 data points for each house, including the price of the house. Now, the price is going to be what we want to predict.

So, we're going to use the other variables, the independent variables to make a prediction of the house price, which we're going to say is the dependent variable. Now, to do that, we'd like to know which of those independent variables is most correlated with the house price, because the ones that are most correlated, or in fact least correlated, you can have correlation both ways, will allow you to make the best prediction. So, I'm going to choose just four columns out of those 80 to make my prediction of house prices. We're going to build together a linear regression model, and it's going to allow us to make predictions.

And we're going to test those predictions against test data that we haven't seen. So, we're going to divide the data into training data, and inter test data. So, we're basically building a machine learning model here. And what I'm going to ask you to do, is to build a better model. So, you can take as many columns as you like to build a predictive model for house price data. As I say, I'm only choosing four columns. You should be able to do much better than me. Well, maybe a little better than me. So, let's go and predict house price data.Okay, bye for now.

Video 2 (13:11) – Sample Variance vs. Population Variance

So, the problem that we're usually trying to solve is that there is a population out there and we can't sample it all. So, we can only take small samples, say, of that population, but we still need to try and infer the statistics that apply to the whole population. So, let's take a look now at an example. We might be talking about the blood pressure, and we're going to sample from, perhaps, different parts of the country. And we'll see, that depending on how we sample, we get different means. So, here on the left, we got a mean of 75. Then, in the middle, 67. And then, on the right, 71. And this will be normal, that we'll often get this problem.

So, let's take a look at how it arises. So, here, we've got a small population. I've just taken some numbers here, and we're going to calculate the mean and the variance. I generated these from a uniform random distribution. So, it turns out that the mean per chance is five, and the variance is 8.2. And you can see, roughly, the variance because we've got numbers like zero and we've got nine. So, we've got quite a variation there. And the way we calculate the variance is we first need to calculate the mean. So, we do that just by summing all the values and dividing by the number.

So, here, we've got 1, 2, 3, 4, 5, 6, 7, 8, 9, 10. We're going to divide by 10. So, we sum them up, divide by 10, and we get a mean of five. Now, we want to shift all these values, so that they're about the mean. So, the mean is going to be the new zero of our X-axis if you like. So, here, we're calculating the mean and we're subtracting it from each of these values. And then, we square that result. So, it's going to be a positive number. And we take the mean of that.

So, that's how we get the variance. Now, let's see what happens when we take a sample. So, I choose five at random. I choose 5, 4, 9, 8, and 6. And if we calculate the mean, it's 6.4. So, now, I subtract 6.4 from each of these values. I do this shift to calculate the variance. And the variance now is 3.4, which is very different to the 8.2. Now, you can maybe see why the variance would be smaller. It's because the mean now is the mean of this sample. So, the variation, since I've only got five values, the variation is likely to be smaller than the population variance. And it certainly is.

Now, let's take another example. So, here, I'm choosing at random. And here, I've got a variance now of 5.6. So, before I had a variance of... What was it, 3.4? Yeah. Now, I've got 5.6. But the real variance is 8.2. So, it seems, for some reason, we're getting too lower value; we get a much lower variance. Now, part of it is because the mean of the sample is going to be different. And also, just because we've got a smaller number, it's likely that the variation, the variance is going to be smaller. Let's see what we might do to correct this.

So, here's an experiment, and I've written a code for this. We generate a population of 200 now instead of just 10. We're going to get 200 from a uniform random distribution. We're going to pick samples of 50, and we're going to calculate the mean and the variance, and we're going to subtract those from the real mean and variance. So, if we're accurate, we should get zero. If, from our sample, we're predicting the real mean and variance, then it should be zero. But notice, while the mean, the most times we do this, the mean goes to zero, but the variance doesn't.

Here, in this case, it's around 20. Now, the variance of the general population is probably of 100 or so. You'll see that there's a systematic error here, because somehow, we're underestimating the variance systematically. So, one solution that's been proposed is to instead of dividing by N, we'll divide by N-1. It turns out that when we divide by N-1 instead of N to calculate the variance, that we actually converge quite well. Here, you see the variance error going to zero. So, let's take a look at this code. So, here, let me first take a look at how we're going to do it on a small scale.

So, here, we generate an array, and we calculate the mean and the variance. And then, we take a sample from this array. So, here, I'm taking five values, and we calculate the mean and the variance of that. Now, this 'ddof', degree of freedom, if we put it zero, we're going to be dividing by N. We're going to be using N, the sample, the number in the sample to calculate the sample variance. And what we'll see is, let's try it with this. So, here, we see that the mean and the sample mean; the mean is five, the sample mean is 5.6.

The population variance is 8.2, and the sample variance is 6.6. Now, let's divide, instead, by N-1. So, we change this to a '1'. So, when we're calculating the variance now, we're going to be dividing by N-1 instead of N. So, here, now we see that we get the means remain the same, 5.0 and 5.6, but the variance now, the population variance is 8.2. And now, the sample variance is much closer; it's 8.3. So, the statisticians realized this, and they called this the Bessel correction; that we would divide by N-1 instead of N. Let me show you now the code that I used to generate these graphs.

So, in this case, what I'm going to do is I'm going to generate 200 points. Here, I'm setting up some arrays to record the results. So, here, I generate 200 points in the range of 0-100; a random distribution. And I calculate the mean, and I calculate the variance. Now, there, since it's the population, I'm using N for the number in the whole population, because that will give me the correct result. Now, I'm going to take samples. And I take 50 from that array, from the population. I'd randomly choose 50. So, that's my sample. Now, I'm going to do this 100 times.

I calculate the mean of the sample, and I calculate the variance. Now, the first time, I'm using this 'ddof=0'. So, that means I'm going to use the number in the sample. I'm going to use N. I'm going to be dividing by N. While I calculate the squares of the variances, I'm going to divide by N. So, now, I record. I put into the difference between the mean. I append the population mean minus the sample mean. So, I'm keeping track of those. And I'm going to average that array by calculating the sum of it, and dividing by the number of items in that array.

And so, I get a cumulated average of the mean. I get the best mean. Now, with the variances, I'm subtracting the sample variance from the population variance, appending it to array. And I'm doing the same. I'm going to average that out, and hope that it converges. Now, down below, I'm going to plot the best mean and the best variance. So, let's run this, and we'll see the example. So, here, now, we've got... And when we see the error... I'm going over 100 times. And the mean has converged very fast. But it seems that the error in the variance is not converging.

Let's go out. Just to check it, let's go out for 1,000. Let's make N, I'm going to sample 1,000 times now. Let's see what happens. We'll see the variance still doesn't converge. It's at least 20 out, and the variance is probably around 100. So, this is quite an error, and it doesn't go away. Now, let's take a look at if we divide by N-1. So, we put this degree of freedom to be '1'. And if you look up 'np.var', it'll tell you exactly what this means, but it, we'll be dividing by N-1 now. Let's now see what happens.

But you see, it's converging. That it's doing a much better job of getting to zero. So, that's why we're using the sample variance, we're using N-1. For the population variance, we use the number in the population. But for the sample, we use one less. And there's no proof mathematically that that's the way to go. But if you try it numerically, and we've tried with N-2, N-3, etc., an N-1 does a much better job of calculating the population variance. So, you'll need to be careful to use N-1 for the sample variance.

And of course, this is all coded in these days, this Bessel correction, but it gives a much better result. Okay. So, understand the difference between the population statistics and the sample statistics. And of course, we'd like to get as close as we can using the sample statistics to the actual population statistics, which we often don't know and we have no way of calculating, except in the case where we're doing these artificial runs with a computer. Okay. So, that's the difference between population variance and sample variance. It's very important. Okay. Bye for now.

Video 3 (08:03): Plotting Mean Variance Exercise

So, in this exercise, we're going to take a look at the effect of getting the mean wrong when we sample a population. We talked about this before that, because we're sampling, it's highly likely that our means of those samples are going to actually vary. And that mean error is going to affect the variance. So, let's take a look at how that happens. So, here, for example, we've got a population of red and blue dots. So, that's our total population. And we sample and we take the red dots. So, I've marked the sample here. Now, you can see that the total population is -2, -1, 0, 1, 2.

And so, the mean of that is zero and the variance is going to be -2 times -2, 4, plus another -1, -1, that's five, and another one and another four. So, the total is going to be 10. And we've got five points. And remember when we're dealing with the total population, we divide by N not we don't need the unbiased one, that's only for a sample. So, we divide 10 by five, and we get

a variance of two. Now, if we look at just the red sample, we're going to calculate the mean, so the mean of the red is one plus two, that's three. And we're going to divide by these three samples, divide by three. So, the mean is going to be one. Now, our values for data are going to be instead of 0,1,2 are going to be -1,0 and +1. So, when we square those, we're going to get a contribution of one plus another one is 2. And even when we use the unbiased estimator. So, N now we've got N is three, we're going to divide by two, we've got two divided by two. So, now our variance is only one. So, even with our unbiased estimator, we're going to underestimate the variance.

The variance should be two. That's the total population which we happen to know here because we chose it. But we only get a variance of one. Let's take a look at this problem. We will do a numerical experiment. We will choose a population. So, randomly choose numbers between zero and nine including nine. And from that we take a sample. Now, I wanted you to write a program to plot out the population variance. So, remember with that is going to be with we're going to divide by N. And then the sample unbiased variance. And I want you to do that when we assume that the mean is 0,1,2,3. So, we're going to get different results each time. So, when it's zero, we're going to be dealing with the data as it is here. But when it's one, five is going to become a four, four is going to become a three, etc. We're going to shift these numbers. I want to understand what happens divide we shift these numbers. So, let's going to look at some code. So, here, our going to give you this code, and I'm going to right for you this variance. We can't use the built in one that NumPy has because it automatically calculates it with a correct mean.

Here, I want to input the mean, so the sample could be the population, or it could be this sample, we're going to feed it in and then we're going to specify whether we want bias or unbiased. So, zero would be when we divide by N length of the sample, and subtract the bias, so length of the sample is say N. Then it remains N. But with the unbiased, we're going to make this one, so we're going to use N -1. So, this will return either a biased or unbiased. Now here, what we're going to do is we're going to loop over and change the mean, we're going to have the mean vary from zero to N. And the N I chosen as 10.

And so here we're going to calculate the variance of the population. So, it's an array and I'm going to append to that array the result of calling this with going to mean varying from to N. So, let's take a look at what that gives us. And we see here that it gives us a value of about 8.2 at the minimum here, where the mean is five. So, this is the population variance. Now, let's take a look and do the same for the sample. So, we're going to clone this, and now, instead of the population, we're going to sample. And again, this is going to be a sample, and now we want to do a scatter plot instead of with blue.

Let's do it with red. This should be sample. Change that to sample and now let's see what it looks like, and we see that the sample always underestimates the population. So, this is why we need a correction because we're always going to underestimate the real population variance with a sample. Now, we could take a look at this and change here instead of using the biased we'll use the unbiased. Well, put a one in here, and let's see if that improves it. So, you see now it improves it somewhat. If you compare these two, you'll see that we're actually getting nearer. We Will definitely get a better estimate with the unbiased variance. Okay, so why don't you repeat that and understand exactly why we need to make this correction to make it an unbiased estimate. Okay, bye for now.

Video 4 (02:29) – Generate a Normal Distribution in Python Exercise

So, in this example we're given some data about the blood pressure of 10 people. So, here we have this data, 76, 64, 62 etc. So, what I want you to do is to make an analysis of this data, I want you to find the mean, and then I want you to find the variance, and we need to use the sample variance, not the population variance because we've only got a few people here and we know there's far more people than 10 in the world. So, we are going to use the sample variance and then we're going to find the standard deviation.

We know if the underlying distribution in the population is a normal distribution, then a certain percentage of that population will be within one standard deviation of the mean. So, I want you to calculate the mean, calculate the standard deviation of this sample, and then look up on a normal distribution, what percentage of the people we would expect to lie within that one standard deviation? And see if that corresponds well with the data that we've got. So, how many people lie within one standard deviation of the mean? I want you to either do this by hand, if you're not comfortable with Python, but it would be better if you could do it in your Jupiter Notebook. You'll need to use a few NumPy functions like square root, and sum, and mean, but it won't take you long to do it. And if you want to really do something special, why don't you end up by creating a pandas data frame. So, let me show you that. Here we'd have the blood pressure data, deviation from the mean, and the square of the deviation. So, that would be very nice if you could produce this pandas data frame. Okay, bye for now.

Video 5 (04:29): Generate a Normal Distribution in Python Solution

So, this was the problem that we were set, and we've got this data. Let me, I going to start with this one here and I'm creating first a NumPy array. So, we've got our blood pressure data in. Now, let's calculate the mean. So, we just need to now get the mean. Now we've got the mean of that data, let's calculate the deviation. So, we take the blood pressure, and we subtract the mean. So, NumPy will operate on this whole array. So, we'll see that the deviation is the actual one. Now we calculate the square of the deviation, and we sum it.

So, now we've got the square of the sum of the deviation. So, this is what we need for our sample variance. Now to get the sample variance, we divide by the number of samples minus one. Okay, so this is the Bessel correction. So, now we've got our variants and we can calculate the standard deviation just by taking the square root, so that's NumPy square root of that variance. So, we have our standard deviation. Now we know how big it is. Let's print it out and see. So, our standard deviation is 7.2. So, that means 7.2 either side of the mean, we can calculate that quite easily. So, now we've got our standard deviation, let's create now a panda data frame. So, I take the blood pressure data and I make it into a dictionary and I'm going to now put that dictionary into the data frame. So, now I've got the data frame. Now I can add another column to it deviation from mean, that's our deviation and we'll get the square of the deviation and that's enough and we can now print that out. So, let's do that. And we have, I think we need to import, there we go.

Okay, so remember I'd already imported NumPy before now I'm importing the pandas because we need them, and we get a nice data frame. Now, we can look at the mean and it happens to be 71.3 and we saw that the standard deviation was 7.2, so we can subtract 7.2 from 71.3 and gets it 64.1 and we can add it to it and that will be 78.5. So, anything between one standard deviation to the left or to the right of the mean remains, that would be 64.1-78.5. So, if we look through our data, we'll see that all of these are between those values.

So, we've got six of the results that are between these values. So, 60% and if we look up what the standard deviation is for a normal distribution, it's around 68, just over 68%. So, we're close to that. So, what I'd say is that probably blood pressure data does follow a normal distribution. We can check that. How would we check it? We need more data. Okay, so the more data that we have, the closer we'll get, and we'll be able to check out whether it's a true distribution or not. And we'll see later, there are other ways of figuring out, is that the best distribution for this data? But for now, that's what I wanted you to do. Okay, bye for now.

Video 6 (13:13): Correlation and Covariance

So, let's take a look at how data varies. We talk about data being correlated. And we also talk about the covariance of data; how it varies. So, here, we've got two variables: we've got Variable A on the X-axis and Variable B on the Y-axis. And you can see here in the first one, there seems to be a weak correlation between A and B, in the sense that they're... You can imagine drawing a line through them, but it's a weak correlation. Now, this one seems to be a stronger one. And certainly, when we get to the bottom row, here in the bottom right, we see that Variable A increases and Variable B increases at the same time. So, A and B seem to be highly correlated. And maybe one causes the other, but maybe not. Just because they vary together doesn't mean that one causes the other. We need to be careful about causation. But certainly, they seem to vary together. Now, what does it mean? Here in the middle, on the bottom one, we've got... As A increases, B decreases. So, they vary together, but we'd say that it's more of an anti-correlation. And in fact, we see that the correlation coefficient is -0.8, whereas over here, on the bottom right, it's 0.9.

So, again, up here, top right, we see -0.6. So, as A increases, B decreases. Now, it still means that there's a correlation that one varies with the other strongly, but that it goes the other way; as one increases, the other decreases. But that's fine. Okay. So, that's roughly what we're talking about with correlation. And you'll often hear the word 'covariance.' Don't be afraid of it; just say it over and over again. It just means that one variable varies with the other. That it either increases with the other, or decreases. So, we'd like to get some kind of metric that will tell us how things are varying, whether they're varying together very tightly or whether that correlation is less.

So, here, we're showing X and Y. And in the left-hand one, obviously as X increases, Y increases. So, there's a correlation between those. And on the right, we see that there's a kind of negative correlation, because as X increases, Y decreases. So, let's see if we can find some metric based on these points, these data points that will tell us what kind of correlation we have, whether it's more like it's on the left or on the right. So, let's take these two points. Here, we've got '(-1, -1)' and '(1, 1)'. Let's use this equation. We'll just multiply the X value by the Y value, and sum it over all the data points. And then, divide by the number of data points. So, here, we've got the first data point is '(-1, -1)'. So, when we multiply, that's '1'. And the second point, '1' times '1' is '1'. So, we have '2'. And we divide by the number of data points, '2', and so we get this metric, '1'. Let's see now with the other case what we get. So, here, we've got our first point is '(-1, 1)', multiply X times Y, and we get '-1'. And the second point is '(1, -1)', and we get another '-1'.

We add them together, and now we've got '-2' divided by '2'. So, this metric is '-1'. So, this would seem to be able to distinguish, at least, between these different kinds of lines, that we can see a positive correlation here. And here, we see a negative correlation. Now, we're going to need to modify this formula a little bit, because I just chose very convenient values for these data points. But let's take a look. Let's take a look at the difference now between covariance and the correlation coefficients. So, we usually move the data to be with respect to the mean. That's the first thing. So, we know how to calculate the mean. We just sum all, for example, the X values and we sum all the Y values. And then, we have these transformed coordinates with respect to the mean. So, that's the first thing we always do. Now, with covariance, once we've done that, the covariance coefficient is defined as the sum of all these X's times Y's. And then, we divide by the number of points. Now, the problem with this is that this can vary from plus infinity to minus infinity, depending on, for example, the numbers that we're using for the X coordinate or the Y coordinate.

So, let's take a look at that first. I've got two arrays. The first one, '(-1, 0, 1)'. Let's call that the 'x'. And the 'y', '(-1, 0, 1)'. Now, if I do a calculation here of the covariance. So, I multiply the X by the Y's. And I divide by... I've got three points; divide by N. So, I calculate the covariance, and it's 0.66. Now, the problem is if I scale these, for example, and make this 100, then the covariance is now 66.6. So, at the moment, we're very dependent on scale factors that are involved in these two variables. Typically, we could be plotting, for example, the number of rooms in a house compared with the sale price. So, the number

of rooms would be a number between 0-10. And the sale price might be 100,000 to a million. And so, our covariance would vary a lot. So, what we'd like to do is to automatically scale this. And we can do that by dividing the corrected coordinates, so this is the coordinates with respect to the mean, by the variance, the square root of the variance. So, the variance is 'sigma squared'. So, for the X coordinate, it's 'sigma X squared'.

And so, this would be the square root of that; just 'sigma x'. So, we can rewrite this in different ways, but they're all equivalent here. Now, this will make sure that the value ranges from '1' to '-1'. So, I think of the correlation as the normalized covariance; that we've got this scale factor. Let's take a look at it. So, this was where we calculated covariance. And we can print that out. But now, let's calculate the sum of the XYs. So, I've got three of them. And the X's. So, this would be the variance, 'sigma squared'. And here, the same for the Y coordinate. And I'm going to divide now by the square root. So, I've got sigma, I'm really dividing by '(xx * yy)'. And let's take a look at now what happens when we have this as '100' and '100'. So, notice the covariance is still '66', whereas now, the correlation is '1'. So, let's get rid of that. Let's even put a 1,000 or 10,000 in here. And we still get '1'. Okay. So, that's showing you what's called the Pearson Correlation calculation. And it's scaled, so that's always between '1' and '-1'. Okay. With the data frame when we calculate the correlation... Now, I'm pulling in the data frame with X and Y.

This multiply X by 1,000. And calculate the correlation. And you see that the X, Y correlation and the Y, X correlation, and they're always the same, is '1'. So, the correlation coefficient is normalized. So, it's always between zero and one. Let's take a look at three of them here, A, B, and C. So, here we are, A, B, and C. So, this is a row. So, we're saying, at some point in time, the measurement is taken of A, B, and C. And the question is, is there a correlation of some kind between these? Well, we can plotter them. So, we can plot some of them on a scatter plot. For example, A against B.

And we could plot. So, this is just plotting scatter plot of A against B on that axis. And we could plot A against C. And here, we see A against C. And finally, we could plot B against C. So, those are the green ones. Now, to get the correlation coefficients, let's just do... And this will be... And print out 'corr'. And we see here, it gives us the correlation A against B is -0.8, A against C is 1.0, and A against C is 1.0, and then '-0.8' for C against B. So, with data frames, we can calculate the correlation coefficient very easily.

Let's take a look at this case where we're plotting on the horizontal axis, 'Heart Disease', and against 'Ice Cream Sales'. Now, it would be unlikely that heart disease depends on ice cream sales, but you'll often see these correlations that are not causal. That they don't actually... They happen to vary in the same way, but maybe it's not something that's causal. Let's see. So, here, we introduce a third variable, 'Temperature'. So, now, along the X-axis, we have temperature, and we're plotting heart and ice cream sales. And we see that, it makes sense that ice cream sales, the blue line, increases as the temperature increases.

It gets hotter, people want to eat ice cream. So, there's some causal relationship there between ice cream sales and temperature. Probably. Now, the heart disease happens to go down with temperature. But maybe that's a causal reason, too. We'd have to check that. But often, you'll see, as I said, here, we saw this correlation between heart disease and ice cream sales that's not causal. So, you have to be careful to understand if things are causal or not. Okay. Bye for now.

Video 7 (10:10): Central Limit Theorem Exercise

So, let's take a look at a number of experiments that we're going to do, that are going to demonstrate what's called the Central Limit Theorem. So, we're going to have a box. And inside that box we're going to put tickets. In this case, I'm showing the tickets having either a number 0 or 1. Now, we're going to take a sample from what we call this underlying population. So, we don't know what the statistics are inside this box, but if we imagine these were the results of flipping a coin, then we'd probably have 50% 0s and 50% would be 1s.

But we don't know the underlying statistics. Now to discover it, we're going to take samples of that underlying population and we're going to do it in a very specific way. We're going to decide how many we're going to draw out of that population, and we call this drawing of in this case I'm going to choose M=8. This is the number of times I'm going to reach in to create this trial sample. Okay, so we call it a trial or a sample. So, I reach in, I draw out ticket, and I put it into my sample array here.

So, the first one is 1. I place it in the array. I actually put the ticket back into the population because I don't want to change the statistics of the underlying population. So, we put it back in but it's now in that first slot of trial. Now we reach in, take another ticket out, and it's 0 and we do this again and it's 0 then 1,1,0,1 and 0. Now that completes our sample, and we call that one trial with M=8. Now we can analyze that sample, we can calculate its mean, and in this case, it given us a mean of well 1+1+1+1, we'd have 4, so 4 divided by and I've got 8 in my sample.

So, the mean would be 0.5 and we could calculate the variance. And what we've already found is that if I take a fairly small sample of the population, then I need to take an unbiased variance. So, what I would do, I'd divide instead of dividing by 8, I divide by 7 and that would give me a better statistic and we call that an unbiased, taking an unbiased variance. Now, our actual mean for this would have been 0.5 and our variance for this population, if it was a fair coin, the variance would be 0.25. So, it would be 0.5, squared 0.25 and then we'd add all those up. So, each of these, this would be a 0.5. This would be -0.5 but it would be squared. So, that would contribute 0.25. So, all of these would contribute 0.25 and so when we divide it by M, we'd have a population variance of 0.25. But if we did an unbiased sample variance, we'd actually get 0.29. Now, this is because we didn't take, our M is very small here. If we took an M of 1,000, we'd actually get very close to the population variance by taking take the unbiased sample variance.

Now, Abraham de Moivre did a slightly different variation of this. He said, well, let me calculate the mean. So, I'm going to do a trial with M=4, I'm going to calculate the mean, but then I'm going to repeat the trial. So, I'm going to take multiple samples. And he repeated the trial 10 times, each time calculating the mean. So, in these diagrams, these are the means that he calculated. So, he did 10 trials here, and in this case 100 trials. And what he noticed was that he could get a very nice normal distribution, even though the underlying distribution is not normal, taking these trials and calculating the mean gives a normal distribution of that mean. And he got a very accurate mean. But he was also interested in the variance. So, what he found out was that the bigger that the M got, the narrower the distribution, the standard deviation got narrower. So, his variance decreased. And what he found was that the variance was related to the population variance. So, by doing lots of trials, he could get a much better estimate of the statistics of the whole population. So, now let's look at a slightly more complicated case where we're rolling the dice.

We've got a six-sided dice and we're going to create tickets in the box for 100,000 rolls of that dice. So, it's a true dice, so the chances of a 1/6, of a 2/6 etc. Now, we're going to do experiments where we have a trial and we choose M to be a number like 5 or 10 or 20. And what the Central Limit Theorem shows is that when we plot out, let's suppose we choose M=5, and now we do say 500 trials of choosing 5 numbers and we calculate the mean, what we get is a normal distribution. But not only that as the number M increases where our sample size increases, the variance of that normal distribution goes down.

So, our error in the mean goes down. Now, you can imagine if instead of rolling the dice where sampling how a population might vote, then we want to know what our likely error is in estimating the mean of that vote whether they're going to vote for one party or another. So, this has practical applications. Here are the results when I choose M=5, that we get this blue curve. So, it's quite flat. The variance is quite wide. Now with M=20, we get the orange, and the variance has gone down and then when we choose M=100, then the variance gets quite small indeed. And so, we've got an accurate result now of the mean. Let's go and take a look at running this experiment, the code that I used to generate this. So, here's that code. Let me show you how we put samples in the box first. So, we just use random.randint and we choose numbers between 1-6, and we choose 100,000 samples. And we can look at the mean, 3.49, it should be 3.5 to be exact. And then the population variance, 2.91. Now here this code and I'll give you this code, we can specify different sample sizes that we're going to take, we'll take 5, then 20, then 100.

And we plot out, we're taking 500 trials each time and plotting out those results. And that's what we get plotting them. And here are the variance. So, with 5 the variance is 0.6, with 20 the variance has gone down to 0.14, and then with 100 the variance has gone down to 0.03. So, this is the variance, remember in the mean. And so now we've got a very accurate calculation of the mean. So, I'll leave this code for you to copy and to run and you can do your own experiments with the Central Limit Theorem. Here's the most important result here, that the sample variance is 1/M of the population variance. So, it's much smaller than the population variance and this allows us from a sample, if we just multiply by M and the number that we're sampling, then we can get a good estimate of the population variance. Okay, bye for now.

Video 8 (00:48): Drinks Data Correlation Exercise

In this exercise, I want you to answer this question. Pick up the data from drinksbycountry, and tell me in Asia, is wine or spirits best correlated with beer consumption? So, here's a correlation table, but I'm not telling you what country this is for. It's certainly not for Asia. Is it the same in North America? So, is it wine or spirits that's best correlated with beer consumption. Okay, so that's the question, I want you to tell me the answer. Okay, bye for now.

Video 9 (03:23): Drinks Data Correlation Solution

So, in this drinks exercise, we've got drinksbycountry data and we need to answer the question, in Asia is wine or spirits best correlated with beer consumption. And then the second question is, is it the same in North America? So, we need to be able to pick out the continent. So, first let's just check what drinks look like. So, we see we've got continent here and we're going to need to pull out Asia. We need to check actually if Asia's in there, but we'll assume it is. So, let's see if we can pull out Asia.

So, out of drinks, we want to do a lock as opposed to an eye lock. And in the lock, we want to pull out drinks, continent equals let's call that Asia. And we're going to check that we've got the right thing. So, they're all Asia's so that's correct. So, now we need to do the correlation and we can just do that, and we'll see here. So, which is best correlated with beer consumption. So, here's beer servings, spirits is correlated 0.69 and wine 0.43. So, the answer to this is spirits are best correlated and it's not 0.69, nearly 0.7 strongly correlated. How about North America? Well, I happen to know that the country is called North America, but I'm going to check. Yeah, we've got north America. So, I'm not going to change it. It could be, it's just a variable, but we need to do. And now, we'll see that it's not spirit servings because it’s a negative correlation, 0.07. It's not correlated at all. So, in this case, the answer is no. It's not the same wine servings are the best and it's 0.4, which is not very well correlated. Okay, so, that's the answer to the exercise. Hope you enjoyed it. Bye for now.

Video 10 (13:35): Building a Model to Predict Housing Prices Project

So, one of the most common uses of data science is to be able to make predictions. And here, we're going to predict house prices from data that we've collected about house prices in a certain area. So, you can imagine that you're going out house-hunting, and you see a house you like, how do you know what it's worth? Well, if you have data about that neighborhood, you can do a very good job to predict what that house will be worth. So, here, I've given you some house data. And this is for 100 properties. We have data for, actually, over 1,000 that we'll deal with later.

But now, we're going to just use some of this data. And that data has over 80 columns. And the column headings sometimes are sensible, in the sense that you can make sense of them just from that column heading, like 'SalePrice'. But sometimes, you're not quite sure what the heading means, like 'MSZoning' or 'MSSubClass'. So, there's a file here, 'data_description.txt', that you can pick up and look at. Now, we're going to go through fairly quickly. But our goal is going to be, we're going to want to predict house prices, so those are the Y's, from these column data. And those are going to be the X's. And we're going to multiply each X by a beta, which we're going to have to find. And we'll choose the betas, so that we get as accurate as possible prediction of the house price. And this is called a linear regression model because this is linear in these independent variables, X_1, X_2. We just have something that multiplies them, and we add them all together, and that gives us a predictor. We're going to go through linear regression in much more detail.

Okay. So, let's read in the data and take a look at the shape of the data. And we see, we've got 100 rows and 82 columns. I'm going to chop that down, so that I'm going to get some training data. And I'm not going to use all 100 rows, I'm just going to use 20 rows. So, out of here, I'm going to pick the first 20 rows, and I'm going to use 'iloc' to do it. I go for row zero to row 20. And I'm going to pick out all the columns. So, that will be my training data.

Let's check its shape. Yeah. Looks good. Let's take a look at the head, see what it looks like. So, here's our data. And we will see at the end, this is what we're going to try and predict, the 'SalePrice'. So, our goal is to use a number of these columns, these X, independent variables, to predict the dependent variable, Y, the 'SalePrice'. We're going to choose a few of these columns. I'm just going to choose two of them, actually. Each row, then, will be multiplied by the same betas. And hopefully, we can get a prediction. So, that if we have the data, the X, independent variable data, for a new house, we can predict its sale price. Okay? So, that's the goal. And we're going to end up by doing a prediction based on two columns only. So, this is going to be our predictive model. We have to find these betas, which we can easily do, but then we're just going to use this data. Now, you can see, I've got zeros in here, so this data is not good. Maybe we shouldn't be using the... this is Masonary Veneer Area. But let's go back to our data. And we're going to do a quick pass through to show you the mechanics of how predictions are made.

So, we've got that data. Let's take a look at the sale price initially. So, we want 'train'. And out of that, I can get 'SalePrice'. So, that will be... What will that be? It will be a series. Right? It's going to be the whole column. Okay. So, we'll just print out the head of it. There you are. So, 208,000, etc. Let's print it out all just to make sure there are variables in every row. Yes, we've got 20 of them. They're all good. There might be not a number in some of these that we should steer clear of. Okay. Let's plot that just to see what the distribution is of the 'SalePrice'. So, I'll copy that. And we'll do a 'plt.hist'. So, there you see it. So, we've got a number of houses: six and five around 150,000, one around 350, and one around about 100. Okay. Let's select the numeric columns. And then, we'll calculate the correlation factor that will tell us which columns are important for us to include. So, the numeric data will be 'train.select_dtypes'. And I want to include,

I can have a whole number of them, but I just want 'np.number'. So, that will give me all the numeric columns. Let's take a look at that shape of what's left now. So, I've got 40 columns out of the 80 that are numeric data. That's pretty good. Now, let's figure out the correlation. I'll call it correlation, 'corr', and I'll use 'train.corr'. Sorry. And I'll use 'numeric' because I only

want to use that numeric column. So, that's 40 by 40. Now, out of that correlation, I want to find out what's correlated with 'SalePrice'. So, this will give me what's correlated with the 'SaleP'rice'. And out of all of those columns, I want to just pick the most important ones; the ones that are most correlated. So, I can 'sort_values', and I can sort them '(ascending=False)', so I'll have the most important at the top and the least important on the bottom. Now, out of all those, there's going to be a series. I just want to pick, let me pick out the top five. Let's call those, I'm going to call those 'cols' just so we can take a look at them, because it's the columns that are important. So, here they are. 'SalePrice' is obviously correlated with itself.

And its correlation is one. 'OverallQual' is 0.8. The 'MasVnrArea' is 0.78 correlation. Whether there's a full bath, 0.72. And 'TotRmsAbvGrd' is 0.69. Okay. What I can do is, let me get the index of this, because I need an array that has those columns in it. Like I said, I'm just going to choose the top two. So, I'm just going to include these two. So, now, what I want to do, I found the correlations, I want to pick out the X, Y values now for my model. Let's do that. So, what I'm saying is, out of all that numeric data... Actually, I'm going to go back to the training data, because now I know what columns are numeric; I've got them. And so, I'm just going to put in here 'cols'. And that's going to give me this data frame. And out of that, these two are going to be my X's, and the 'SalePrice' is going to be the Y. So, let me get the Y first because that's easy to get. Okay. That's the Y. And now, the X's, I just need to drop out that 'SalePrice' from the X's.

I've got too much in there at the moment. And I want to drop out that 'SalePrice'. And I also want to put in 'axis = 1', so that it knows exactly what to drop. And let's print out my X. So, I'm all set up now to build a linear regression model. From 'sklearn', I need to import the linear model. Now, let's build the linear model. So, 'lr = linear_model.LinearRegression'. Okay. And now, to build the model, I actually need to fit the data. So, I'd prepared the X and the Y, remember? So, now, I've got that. I've got my model. And now, I can make predictions. Okay. So, those are my predictions from the X's. Now, I should really to see how good the model is, I'm going to do two things: one, I'm going to see how good it is of making predictions from the data that we already have, from the data that it was trained on. So, it should do a fair job. But I should then really test it against data that it hasn't seen that I didn't train. So, remember, I only took the first 20 rows of that 100 rows of data.

So, I could easily test it against some of those other rows. I'm going to leave that to you. But here, now, I can tell you how good the model is. There's a factor called R squared. So, 'model.score'. And this will tell us how much of the 'SalePrice' is being predicted by those two dependent variables: what percentage. And it's going to give it as a number between zero and one. And so, you see that 75% of the 'SalePrice' is already being predicted from just those two variables. Let's take a look at doing a scatter plot of the predictions against the actual.

So, this should be a straight line with a slope of one. And it's not too bad. There's a fair amount of scattering there, as you can see. But, considering that we only use two columns of data and only 20 rows, that's pretty good. So, what we're going to do now is we're going to go back over this and we're going to look more closely at those columns. We're going to repair the data. We're going to clean the data. And we're going to see if you can build a better model. I think you can build a better model with this R squared factor as being over 0.8. Somewhere between 0.8 and 0.9 would be good. But I'm going to give you some more help. So, let's go now and repeat this process, but look more closely at that column data, and understand if there are other columns that we should be including that will give us a better prediction. Okay. Bye for now.

Video 11 (14:39): Building a Model to Predict Housing Prices Project Walkthrough

So, last time we went through fairly quickly predicting house price data. This time we're going to go through a little more slowly and look at the data a little more carefully. So, the first thing that we're going to do is we're going to pick up the data_description.text which tells us the meaning of a lot of the columns that we're dealing with. So, I've copied some of them here, sale price is obvious. Subclass, MS Subclass, the building class. We had the Masonry Veneer Area as one of the most correlated variables. So, let's go and pick up the data again.

So, we'll pick up the data but this time instead of just 20 variables, let's pick up, I believe there are hundred there. So, let's pick those up. Let's take a look at the train.shape rather than the data.shape. Okay it's 100 x 82 columns. Let's add some, I'm going to put some sales in here because one of the things I'd like to do is like I'd like to take a look at the sale price data first to see what we're dealing with. So, let's pull out sale price and we'll just pull it out of the train, and we know that we can do that if we just go with sale price. Okay, let's just do describe on sale price. So, sale price, it's going to be, we know that train is a data frame. So, sale price is going to be a series. So, we can do describe on it. So, they us will give us some data about the sale price. So, here we've got 100 of them. The mean is 173,000, Okay, standard deviation 72 minimum etc. Okay, let's actually do a plot. So, let's do a plot and let's do a histogram and take a look at sale price. Okay, so that's what sale price looks like. It's certainly not normally distributed, it's actually fairly skewed in the sense that it's not evenly distributed about any point here.

So, one thing that we might consider is we might want to scale this. This actually looks like an log, normal distribution as it is. So maybe we should take the log of the sale price and plot that. Let's do that. So, np.log.(salePrice). Let's take a look if that looks, let me just check the sale price. Okay, let's take a look. It looks a little more normally distributed now. So, we could go ahead and use the log of the sale price and then we just need to remember at the very end, if we can predict the log of the sale price, we can certainly predict the sale price. It's just a simple transformation, but whenever you do these, you need to remember what they mean at the end. We can actually look at the skew. So, this was the original data, let's take a look at the skew. So, the original data had a skew of 1.17, which is quite significant. Let's take a look at the log of that and take a look at the skew of that. And so now the skew is 0.09 about one, minus 0.1. So, it's reduced considerably from over one to just about 0.1 . So, let's transform it, and let's make target the log of the sales price data. So, that's our new sale price data. Okay? Let's, for example, pull out the train, the ground living area. Let's do a scatter plot and we'll plot that against we can put 'y' equals target. And that should give us a decent plot. Let's take a look. So, here we've, we can see that there's definitely a correlation between the living area and that target.

Let's take a look at another one. Let's plot the... let's see if the garage area is correlated with the target. Again, it's pretty well correlated actually. So, let me take a look at the... I want to find the nulls. So, and well, we've got a data frame. But what we're going to do is use train and we'll pull out the 'isnull'. And I want to add those up and I want to sort the values and we're sorting values and we want to have those ascending… is false, we want the descending from, okay. So, let's print those out.

And let's just print out, let's print out the first say 20 of those, print them out. So, you see that we've got quite a few nulls here. The pool has 100, Alley has 94. So, certainly we shouldn't be analyzing those. Now as we get down here, we want to see if there's any in, we'll keep this list because we may want to look if we've got any nulls in columns that we're actually going to use, then we better repair them. So, did you see what we did here? We created a new data frame called nulls.

Let's suppose we didn't. Let's suppose we just said nulls equals train and this. Let's see what it would look like. Well, we get the same results. It just doesn't look quite so pretty. It's not a data frame anymore. So, if we did type, i think you'll see that it's a series. It's a panda series. Okay, let's go back and now let's see if we can repair all those nulls. So, what we're going to do, we're going to operate on the train, training data. We're going to select the data types that np.number and we're going to interpolate. Okay, we had to be a little careful here again, we need to put access in here equals one. And that will now give us the 100 rows of the data. Perfect. And we've got 39 columns left in the data. So, now we've got our numeric data and we can carry on with numeric data. So, let's see. Now we need to go back, and we need to find the correlations here. Remember we did this last time. So, this time let's go with six. And let's actually before we do the index, let's take a look at the correlations.

So, now we've got overall quality. Still the living area, garage area is useful to have, garage cars and year's built. There's quite a number. Let's have a look at eight of them. Just, There's quite a number in this sixes. Okay, let's take a look. Well, let's go with six of them. So, that's our calls, and remember we need to do index here to get the names of them. Okay now we've got

these and let's now pull out of data those columns. Okay, so we can go back to data we don't have to do numeric because these are all numeric columns. Correct? So, now we will have this as the sale price. Okay, this looks good. Let's take a look at 'X'. It's a data frame. Okay, It's looking good, its looking good, okay, let's go and carry on now. So, now we've got our sklearn, this is fine. We've got our 'X' and the 'Y's, Perfect! We've got it up to .83 for r squared. So, 83% of the price edge is explained by these variables. These independent variables, overall quality, grand living area, garage area, garage cars he has built. So, we've already improved from .75 up to .83. And we can take a look now at predictions. Perfect! This should look the error. 'Y' minus 'y' predictions should look like a normal distribution. And it does. Let's take a look at the scatter plot. That looks pretty good. Should look like a straight line. That looks very good indeed.

Okay, so let's test our model now against data that we haven't seen. So, we're going to pick up 'jtest.csv'. We'll load it in, and we'll just look at the head and this is data that we haven't trained on. So, what we need to do now is, we'll get the same columns of the test data, we'll pick out the sales price. So, now we're going to make predictions based on this 'X' data. So, let's make our predictions and we'll see how good our model is. And here we are. And our model is .74. So, on the data that we haven't seen we're not as good, but we're still not too bad. .74. we were .83 on the data that we had seen. Okay, so we've seen that by cleaning the data we can get better prediction. Why don't you now attempt the homework? And you can choose how many of the correlation values you want to use. See if you can do a better job than I've done. Okay, bye for now.

----X----X----X----

