Module 22: Interpretability and Causality in Models

Video Transcripts

Video 22.1: Module Introduction: Building and Interpreting Managerial Predictive Models (15:15)

So, welcome to this final lecture on deep learning related issues. And actually this lecture at some level was motivated by the fact that we've spent the last five sessions doing deep learning, but at some level, I would like to think of really what we're doing in this particular lecture, it's broadly useful for anyone that really builds predictive models. And so I want to take a step back to kind of give us a little bit of a deep learning specific motivation, but you'll see, hopefully, that this is actually quite a general issue that we're trying to tackle today. So, let's take that step back.

Really what deep learning is about is essentially building complex representations of unstructured data. And then having built those complex representations, relatively simple models actually can do a decent job of regression or classification or whatever it is, Modern deep learning models don't just consume complex data, they can actually generate complex data. Like if you read the general news around things like DeepFix and so on and so forth, you know that deep learning models are capable of generating fake but really complex structures that are realistic. All of this is to say that these models are extraordinarily complex, Really my goal today, stealing a line from Andy Grove, is basically to understand what's going on in these complex deep learning models that are effectively black boxes. And so, what I want to do is to "cut holes" into these black boxes and peek inside, that's really our task for today. So, that's high level, right? 20,000 feet, maybe 100,000 feet, but let's dig just a tiny bit deeper into why we need to worry about today's session on interpretability and causality. And to make the case real, to make the case concrete, I want to focus on one application area. What I'm going to talk about, by the way, is relevant to multiple application areas, we'll touch on some of them later in this session itself. But let's just zoom in on one application area, which is AI being used to power medical diagnostic questions, and let's say medical imaging in particular. So, medical imaging is something which generates data that, as you can imagine, is complex, unstructured, and heterogeneous. So, even a humble X-ray, it's unstructured; it's basically a picture, and it's heterogeneous in the sense that you take an X-ray of me on a whole bunch of different days just because of variations in positioning, variations in whatever's happening in my body, the images you're going to get are heterogeneous. Imagine the heterogeneity you get as you go across a population. So, this data is really heterogeneous, and typically, and it's not just X-rays obviously, there's CT scans and ultrasounds and so on and so forth. Typically looking at this data and diagnosing this data is the job of a doctor, the job of a radiologist or a specialist, or whatever it is. But in order to help these folks out, there's been hope that AI has the promise to essentially aid these folks in their jobs, and ultimately what that might result in is the improvement of health outcomes at lower cost. And that's uncontroversial, that's obviously really a good thing if that were possible. So, just as examples of companies, these are commercial ventures trying to do this stuff and I have nothing to do with any of these ventures here. DiA is a company that does automatic analysis of ultrasounds, Same sort of motivation. Aidoc, same story for brain CT scans. So, they want to automatically analyze brain CT scans and provide decision support for that. Clearly wants to do this for heart CT scans. Exo is a company that's trying to build a cheap ultrasound device where, at some level, deficiencies in the "imaging hardware" are made up for by AI post-processing. And Scorpio is a company that wants to help automate the process of looking at and understanding self-morphology for understanding tumors. So, all of these really noble

ventures, and at some level, you could see how they meet this promise of improving outcomes, reducing costs, and so forth. So, all that's great. If I were, let's say, one of these guys, Aidoc, clearly, whatever, and I wanted to build a predictive model that took in some imaging and came back with some decision support, I'm building the models we've been discussing over the last five sessions. So, it keeps things really, really simple. What's the simplest imaging that you might look at? It might look like a radiograph, a chest X-ray. So, let's say I looked at a chest X-ray and my task was to basically look at the X-ray automatically and to diagnose whether the person's healthy, whether they have pneumonia, whether they have fibrosis, whatever. And the hope is that, like you saw this in the modules you did where we did image recognition with deep learning, we saw that maybe we could build CNNs, convolutional neural nets, for a task like this. So, beyond distinguishing random things in our daily life, you could use it for very serious tasks such as this one. You know what a CNN is at this point, it's a pretty darn, complicated model. It's a very, very complex model. And so high-level picture, 20,000 feet, chest X-ray goes in. It's like an image, it's like a picture, you run your CNN that you've trained on top of it and the CNN predicts, "Hey, this person's healthy, this person has pneumonia, or something like that." And the model in the middle, the F, is a complicated one. It's a CNN, it could have millions of parameters. So, that's the game. Let's contrast this for a second, let's do a 360 almost or 180, I guess, and contrast this with a different predictive modeling task. You're probably scratching your head wondering, "What in the world is this guy doing?" But just trust me. Stay with me for a bit.

So, here's a different modeling task that's completely different from medical imaging, and this is the task of pricing risk. The task of pricing risk is something every insurer needs to do, and the goal over there is to insure loss or insure risk at a fair price. And you want to insure it at a fair price because otherwise you'll go under. So, the rough goal is to say, let's say I'm in the business, for instance, of auto insurance. The goal would be to look at a potential driver who wants auto insurance from you and to understand something like over the next year or the next two years, whatever the term of the policy is, what's the loss that this driver might actually generate? And it stands to reason that this might depend on a whole bunch of things. It might depend on the driver's age, things about their socioeconomic status, their marital status, the kind of car they're driving, what have you. Funnily enough, the sorts of models insurers typically build to measure risk are actually really simple models. They look like the model you're seeing on your screen. The model you're seeing on your screen is something called a generalized linear model. It's a generalized linear model because we're fitting the log of the loss, Not just the loss, we're fitting the log of the loss to some linear function in the age of the driver, the driver's marital status, whether or not this person's driving a two-door car, and so forth. And if you think I'm making this up, I have a little book on the screen. This is a book from 2021 that actuaries in the US might actually want to read as they think about this job of building models. So, I'm not deliberately simplifying things over here; indeed, insurers will use these really simple generalized linear models to price risk. So, this is kind of weird, right? We jumped from... I mean, at this stage in the class you have seen now a plethora of different predictive models, optimization tools, what have you, but predictive models for tasks like this. And you're probably thinking to yourself, "Hey, listen, it's great that this is simple, but are we leaving money on the table?" Because we're not using the best possible model, and as a result, because we're not using the best possible model, our accuracy is not as good as it can be. So, are we leaving money on the table? So, that's potentially the bad. On the flip side, what's really great about this model is it's totally transparent. You can look at this model and you can say, "Hey, makes perfect sense, right?" I mean, taking you all the way back to when we did linear regression, you know how to interpret coefficients, right? This model saying that, "Hey, you know what? The way I'm going to price risk, older drivers, I'm going to price them lower. I'm going to say the loss goes down. Marital status, if you're married, I'm going to price you lower. I'd say the risk goes down. On the other hand, if you're driving a two-door sports car, maybe that loss prediction actually goes up." It's really transparent. I contrasted back with

that chest X-ray thing I was talking about. I mean, you don't have a simple formula like this. The state-of-the-art convolutional neural nets have like 10 million parameters, but good luck to figuring out what those 10 million parameters are doing. Like, you can't, okay? So, that's kind of an issue. Unlike this super transparent model over here where, at some level, the insurer in pricing according to this model is comfortable that they know exactly what the model is doing in this mission-critical task of pricing risk. In this case, you don't, right? You could deploy something like this, but you'd be nervous because maybe you don't understand exactly what the model's looking at. What is the model doing to come up with its decision or its recommendation and saying that somebody has pneumonia or doesn't have pneumonia? In fact, that's not just that, what's going on is even more subtle. So, in particular, when you look at a model like this insurance model, in addition to its interpretability, where you can see what the model depends on, you're actually expecting more, you're actually expecting that these coefficients, the minus 0.1, the minus 0.15, the 0.2, that these coefficients are the right coefficients. What do I mean by that? What I mean by that is when I show you a model like this, you're expecting that given a driver, all else being the same, an older driver is going to be a lower risk. Or put a different way, I take the same driver, nothing changes with that driver, in terms of whether they're married or not, the two-door, whatever, whatever, but as they get older, they end up posing a lower and lower risk to you. You're actually taking that away from the way this model is stated, and in effect what you're doing is that you are asking for causality. i is you're asking for causality. saying this is not just about prediction, it's about causality. All that's being the same, I change one of those things, it's actually going to change my prediction based on the coefficient that I see over there. What might causality mean over here? Over here, with the chest X-rays, with causality, we're perhaps saying, "Hey, look, the things that we're focused on to actually say whether someone has pneumonia or not are not things that are correlates of pneumonia, but rather things that are truly the medical definition of pneumonia." Like you have a patch somewhere in your lung, that patch represents pneumonia. The absence or presence of the patch drives whether I'm saying you have pneumonia or not, that's perhaps what we need over here. We're relying on something that's truly the mechanism of what we're trying to predict. So, hopefully what you're seeing, it's great that we've got these very sophisticated models, these very sophisticated predictive models at our disposal. And these are not just deep learning models, of course, deep learning models fall into this category, but other sophisticated models you've seen earlier in the class. Maybe even a thing as humble as the random forest or a tree or whatever it is, We've got access to all these very, very sophisticated models, but in deploying them in production, we probably care quite a bit about interpretability and raising the stakes. We probably care about causality as well. And so this is what I'm going to devote this lecture to. Getting us, actually, to our first order, basically the state-of-the-art with model interpretability. What we're going to do is we're going to cover two well-regarded approaches, to basically punching holes and black boxes, to rendering those black box models interpretable. And what we'll see is, given this capability, how we can use this capability in practice to audit a model as it were and to understand what the model is doing so that we're confident what we're rolling into production. We'll go even further, almost as a bonus, and think about causality. Now, the thing is that this is an area where our understanding is still developing. And so I want to expose to you what the problems are and how people are approaching this problem today, but you need to watch this space because this is truly something that's cutting edge and evolving.

Video 22.2: Integrated Gradients (10:31)

So, we said earlier that the Shapley values were great but computationally expensive to compute. We used the example of using CT scans as input and said, "Hey, you know, that input's pretty complex, 256 by 256 by 40 numbers."

And so, we said that raises the need a new type of approach to cutting holes in black boxes, to interpreting black box models. And so, I'm going to introduce a second approach referred to as integrated gradients. This was introduced by a bunch of researchers at Google and finds broad application, especially in understanding complex input that's of the form of images, really common in that particular use case. Why are we doing this?

It's actually two things. One is integrated gradients are going to be computationally much, much less expensive than Shapley values. We're going to be able to work with models that have tens of millions of parameters and really complex inputs. And render those models interpretable, you'll see this, actually. You'll see this as we go along. The second nice thing is, integrated gradients are fundamentally visual in nature. They're fundamentally visual; they're very natural, you'll see this too. And this makes their use very, very easy, very, very compelling. And the high level approach is still as before, we're given some black box model F and the model could be predicting pneumonia; it could be predicting cervical cancer. But given one of these black box models and the starting point for integrated gradients, is basically the question of saying, "Hey listen, can we measure how sensitive our model is to a change in any one of its inputs?" That's the starting point. Isn't that natural? What it's saying is, how sensitive is the model to a change to any one of its inputs. Now again, I love when I look at these things, I love coming back and thinking about the linear model, because it's the simplest model we could look at. If we looked at the linear model and we looked at one of the features of the linear model, we said, "Hey, how sensitive is the model to a change in this feature, what is it? It's the coefficient in front of that feature." Again, exactly what you would expect. If you were looking at one of these linear separable models which we were looking at when we, let's say, did linear regression. Of course, if it's a complicated nonlinear model, it's not as simple as that. There's no coefficient to look at. So, just coming back to our cervical cancer example, just to remind you of it, we had 14 input features, and we're predicting the probability of cervical cancer. And as like a concrete place to focus, we were focused on this IUD feature. But translate it to the setting, what might sensitivity mean? Well, sensitivity might mean something like the following. You might say, "Hey, look, I've got the original set of features X1, X2, X3 all the way up to X14. That predicts some probability of cervical cancer. Let's call that probability Y. And then I might have some other feature. We get some other alternative set of features, X1 prime, X2 prime, X3 prime, all the way up to X14 prime. And by the way, X1, I'm being deliberately fast and loose about what X1 prime, X2 prime, X3 prime all the way up to X13 prime, are. These could be, in fact, this is up to us to figure out what they are. But nonetheless, we've got these two feature vectors and really sensitivity boils down to saying what's the rate of change, so to speak of the predicted probability, what's Y - Y prime. What's the rate of change in that probability as you change the input feature in question. So, as I go from X14 prime, let's say X14 prime is zero and X14 is 1. As I go from zero to 1, how does that predicted probability change? And ignore for a second that these input features are like discrete and so forth. This is just for our thinking. Let's say the model is differentiable with respect to each of these features. Really what we're saying, if we're kosher to take derivatives, we have a specific data point, let's call it X, and we're asking ourselves. What is the gradient of our predicted probability with respect to one of those features X14?

The 14th component of X. Now, just like we had in Shapley values, as I take this gradient, the values of the rest of the features matter. And in Shapley values, we averaged out over a bunch of other features. In integrated gradients,

here's what we're going to do. We're going to say we start at some baseline value of X. Maybe it's the the vector of all zeros or something like that. But we start at some baseline feature vector. We draw a straight line from that baseline feature vector to the X we want to get to. And all along that straight line, you take gradients and add those gradients up. So, the averaging that we saw in Shapley values, in a sense gets replaced by an averaging along a straight line over here. Now that sounds a little bit abstract, so let me actually make it really visual for you. So, here's really important question. I built something to detect animals in pictures. And let's say my model looks at a picture and figures out if there's an animal in it, and if there is an animal in it, it tells you what animal is in the picture. So, presumably it looks at the picture on the right and it says, "Ah, you're looking at a panda." Now, how might integrated gradients work over here? So, remember, the X that you care about is the panda. That's, the input image. And let's say as a baseline image, we take the blank image, a black, just just black. So, there's just the empty image, which is all black pixels, nothing else. And now, for any given pixel of X, and so I can draw a straight line, so to speak, from the original baseline image, which is some large massive vector to the original panda image, which is a large massive vector again, but just with values that are not just, 255-250-5255, they're about pixel values that capture what's going on in the image. And what I do is along that straight line, I simply take a gradient of my prediction of what's in the picture with respect to what that particular pixel. And so, in particular on that straight line path that I've illustrated the path. On that path, going from Xi baseline to Xi original, this is the i'th pixel. You're actually taking gradients of 'th with respect to the value of the pixel and just adding those gradients up along the path. So, that's integrated gradients. Again, we've gone into the math over here and hopefully the math makes sense. All we're doing is we're looking at sensitivity by taking gradients but as opposed to taking gradient at a specific image, you're averaging it over a path from some original baseline image to the image in question, and the integrated gradient folks had some really nice arguments on why this is the axiomatically right thing to do. But as a user of this technology, maybe you can put this aside. What we want to get out of this is basically this is going to tell you, for every pixel in the image, which of those pixels actually mattered in making the prediction that the model made. So, let me just say that again. Let's imagine now for a quick second this is just some machinery, some algorithm, forget about how it works. This is some algorithm and what this algorithm is supposed to tell us is you give me an image, and for that image, I'm going to tell you how important each of the pixels in that image was to my making the prediction that I made. And so, looking at the panda, what pixels, what parts of this picture was I looking at to actually say that it's a panda? If I actually ran integrated gradients on this panda image, and a model that's trained to recognize animals and pictures, what I would get is a quote and quote attribution mask, which is really just a plot of the absolute value of the integrated gradient across the image where large values are colored white and small values are colored black to see what this model was actually looking at the detect a panda. Now I'm not like a zoologist or anything like that, but if you look at this picture over here, makes sense to me. It's picking up the natural thing I think about when I think of a panda. Like the weird looking like big oval eye thing, and it's mouth and nose or whatever it is. It's got that characteristic shape that it's pulling out to say, "Hey, there's a panda in this image." And so, if you looked at something like this, you'd say, "Yeah, okay, the model's doing something reasonable; it's doing maybe what I would do to figure out whether or not there's a panda in the image." If, on the other hand, what might have been an example of something wrong if it were looking at like, I don't know, the grass, to figure out that this is a panda; that would probably not make sense. But that's not happening here, thankfully. It's looking at the face of the panda, the shape of that face, the patches of light and dark on the panda's face to figure out whether or not this is a panda. So, we've now seen two tools to take a black box model and render it interpretable. What I want

to do next is watch these tools in action. I want to leverage these tools next in the spirit of auditing black box models that we might actually be using in production. I want to see how one might leverage these tools in production.

Video 22.3: Using a Model on Chest Radiographs: Part One (4:01)

So, we're pretty dangerous because we were armed with integrated gradients and Shapley values. We've used integrated gradients to understand and audit pretty sophisticated convolutional neural net, that was designed to detect emotions in an image. And we saw that, fed a whole bunch of pictures of Arnold and the Mona Lisa.

It was able to at least focus on the right spot to make its decisions. Now, as I said, I wanted to pivot to doing something much more real, much more in line, as it were with what was motivating. And so, I'm going to come all the way back to this task, we built predictive models to look at medical images and say useful things about these medical images. So, let's say we built a model that's designed to look at chest radiographs, X-rays, and determine whether or not the person in the X-ray has pneumonia. Are they healthy or do they have pneumonia? This is really simple. Radiologist, this is a laughably easy task. We're going to build a CNN to try and do this. Remember, from a modeling perspective, it's not that easy, because it's this massive heterogeneity. And so, let's say that we've built a model and we want to roll the model out. A natural question we might ask is, this classifier that we built, this predictive model that we built, how generalizable is it? And so, in particular if I train on data from one spot, from one hospital or whatever, it's going to work well in other spots. And so to that end, I'm paraphrasing a published piece of work that took 30,000 or 31,000 odd chest radiographs from the NIH Clinical Center. About 13,000 from Mount Sinai Hospital in New York, and about 3,700 chest radiographs from the Indiana University Network. So, this is a hospital network associated with Indiana University. So, natural question here would be, I built a model, my model was trained, let's say on NIH data, would it still work well at MSH? Would it work well at IU? Or it was trained on MSH data, would it work well at the other places? So, you can ask that question. You can literally do this because you have the data. So, it turns out that, let's say your training set were all the chest radiographs from MSH, from Mount Sinai and you tested on Mount Sinai, you get an accuracy of about 61%. Now this is actually not that bad. Okay, why? Because the fraction of times you actually throw up a flag is really small. The incidence of pneumonia is perhaps not that high. But you take that same model, that was trained on MSH data and you try it out on data from NIH and IU, on data from the NIH and Indiana University, the accuracy is terrible. It's 18% accuracy at NIH. Like I don't know what that is, like 9 point something percent accuracy at Indiana University. So, it's really not very good at all. Certainly not something you would ever dream of putting into production. So first off, I'd like you to pause your video over here. I want you to stop, okay? And you're mature enough as a modeler, as a data scientist at this point to diagnose maybe what's going on. I want you to look at this and come up with hypothesis for what might actually be happening over here.

Video 22.4: Using a Model on Chest Radiographs: Part Two (9:58)

Okay. So, you've thought about this for a second, and you're probably saying, okay, the data from MSH is probably not representative of the data from NIH and IU, and that's why we're kind of screwing up over here, right?

In fact, you'd be correct. Okay, here is sort of a summary of the data from MSH, NIH and IU, right? So, each of them is organized into sort of three columns, one for IU, one for MSH, one for NIH. And I just want you to focus on two things. So, focus on the role that looks at the average sort of age in the data set. You'll immediately see that the average age of patients in the MSH data set is 63 years, whereas the average age in the NIH and IU data sets is sort of mid to late 40s, right? These are very, very different age groups, right? So, it stands to reason these are

very, very different sort of demographics of people. Similarly, by the way, if you look at the incidence of pneumonia, a whole 34% of the MSH data set is people that do have pneumonia. Whereas the IU and NIH data sets, these are very healthy people and not a bunch of them, a very small fraction of them actually have pneumonia, right?

In a weird way, by the way, what should this tell you, right? Like as a baseline, just saying not pneumonia would have been accurate, like whatever percent of the time, right? So, it just kind of puts in stark contrast, how poorly the model we were actually talking about was doing because a baseline would be like 98% accurate, 99% accurate on IU and NIH, Okay. So, you're looking at this and you think to yourself, Okay, right. If indeed the MSH data was not representative, maybe what we need is to make the data set more representative. Now, I could sample from all three and then test on all three, but then that would kind of be cheating, right? That wouldn't be testing necessarily, how the model generalizes to a population that truly has not seen. So, what I'm actually going to do instead, is I'm going to train on data from MSH and NIH and then test it on data from IU with the hope that the NIH portion of the training data is more representative of the IU folks, right? It's roughly the same age group, roughly the same incidence of pneumonia, right? So, it's not like the model hasn't seen patients in that age group with that incidence of pneumonia before. So, I'm going to do exactly that, right? And so now you see two new rows in the table where I've trained on MSH and NIH. I evaluate on MSH and NIH and my accuracy is actually higher and presumably the reason for this is that we're kind of putting together a larger data set that's perhaps capturing more features and more of the heterogeneity, that helps with prediction, right? That's it. So, you can see that prediction accuracy has improved from 61% to 73%. Again, by the way, maybe not something to be super thrilled about still, but at the very least it's improved. On the other hand, when I look at MSH and NIH and I test on data from IU, the accuracy still got off, right? It's like 23 some odd percent. So, now we don't no longer have the excuse that we had earlier, right? The excuse earlier was, hey, the model had never seen patients that were younger and I'd never seen otherwise healthy patients that have pneumonia, and so how would you expect it to learn? It doesn't really have that excuse anymore because I've trained it on data from MSH and NIH. So, what's going on?

All right. So, if you sort of think through how we even got to this juncture, we did all this stuff to render models interpretable, right? So, we do exactly that. Take the model that we're using over here and run something like integrated gradients on it, right? You don't have to run exactly integrated. Run something like integrated gradients on it and look at what the model is looking at in order to make its prediction. So, here that is, and this is drawn from a paper that kind of highlighted this point, all right? And so really in shades of red on panels B and C, you could sort of see both of these are patients by the way, that have pneumonia. You can kind of tell from the patches, right? But both of these are patients that have pneumonia and you can see what the model was looking at in red to predict that these patients had pneumonia. Now, I'm not a radiologist. I'm certainly not a medical doctor of any sort, but I know that you don't get pneumonia in your arm, right? And what we're seeing are your shoulder.

And what you're seeing over here, is that in both cases, the model was mysteriously looking at the patient's shoulder to predict whether the patient had pneumonia or not. This looks insane, right? Actually, there's more to the story. You'll notice that, you know in the top right of the image, there's like this little circular thing. It's a little metal piece that folks that run the X-ray will actually place right on your person to kind of make apparent what your kind of orientation in sort of the machine or against the machine was, right? And different hospitals use different metal tokens. And so what the model was picking up as the first ordered thing was figuring out what hospital it was looking at. That's it, right? It was basically looking at the token to figure out whether it was looking at MSH or NIH and kind of based on that sort of making a prediction of pneumonia or not pneumonia. And this, if you sort of think about it,

that's not crazy because it learned that more patients at MSH have pneumonia, far fewer patients at NIH have pneumonia, and so having just tacked on to knowing whether the patient was from MSH or NIH, it was able to sort of make a decent prediction, of course, in terms of actually doing imaging, that's worthless, right?

That's actually worthless. It's not doing imaging at all. It's playing some statistical trickery by taking advantage of the discrepancies so to speak in incidents of pneumonia in the two hospitals. And so this is an example where auditing the model at hand very quickly revealed that what we were looking at was just garbage, right? This was not a good model, very, very far from anything you might want to use in practice. And the story doesn't just stop here. Here's another quick story, some folks built a deep learning model to detect bicycles and camera images, and this is for use in autonomous vehicles, so that these vehicles can kind of stop a bike, or rather spot a bike and you stop appropriately or something like this. And so they built models to actually figure out, okay, here's the image, there's a bike in it, right?

And then they discovered that, when the image looked at sort of images like you're seeing on the screen, it should be evident to you that's a bike and a person on a bike. It struggled with actually figuring out that there was a person in the frame riding a bicycle. Again, effectively sort of looking at this sort of integrated gradient style things was able to kind of reveal what was going on. In predicting a bicycle, the model was primarily looking for two adjacent circles, like the thing on the left. Makes sort of sense, but that's not exclusive, right? And so, because it was so zoned in on looking for two adjacent circles, when it looked to the right right, it did not see two adjacent circles. This lady had a, whatever that is, like a messenger bag or something like that on the side of her bike. It occluded the wheel and so the machine could only spot one circle and said, no bicycle. Again, this is even more dangerous, right? Because you'd imagine something like this is running in an autonomous fashion, not spotting a bike, probably not a great thing. Okay. And so, what you're sort of seeing here, net, net, right? This is a good place to pause by the way, and just reflect on everything we've taken away, right? Like, take a deep breath. Where are we? We started out with this challenge of sort of saying that these deep learning models, very sophisticated. Effectively, they're black boxes, and shouldn't we be like a little bit concerned about them? And I think the overwhelming answer is yes. Okay. You can't just deploy a black box model in the wild without understanding what's kind of going on in the black box, right? We've seen several examples of this that should kind of make the reason for that amply clear. The good news is we're not flying blind. We have the tools to cut into the black box, to cut holes in the black box, stealing from Andy Grove again to kind of figure out what's going on inside that black box, right? And so I would say, like building a good model that works really well is, one part of the job, as we sort of think about productionizing it, there's a whole host of other things to worry about. One of them is sort of, do we know what's going on with the model? Can we interpret the model, and we have the tools to actually, take a black box model now and render that black box model interpretable.

Video 22.5: Beyond Interpretability With Causal Models (14:00)

So, we're in a good spot.

We've got all these tools to dig into black box models and figure out what these models are doing. Render them interpretable. And by rendering them interpretable, figure out whether we're happy with what they're looking at or not so happy. Now what I want to do is kind of up the stakes. I want to up the ante, and I want to ask for more than just interpretability. I want to ask for causality. And so that just sounds like a word, causality. We all have some intuitive sense of what that means. And in fact, by the way, there's many formalizations in machine learning of what

causality actually means. But let me try and make it real for you. So, this is me watching YouTube. This screenshot is my login. I'm watching YouTube. And you can probably tell that I love cars. I've loved cars since I was a little boy. And I spend a lot of my time, spare time that is watching car video. Now, you look at the right panel, YouTube is in the business of keeping me hooked to my screen. They want me to keep watching videos so that they show me ads and then monetize those ads and so forth. Some level that's the price I have to pay for enjoying these videos. And so if you look on the right, it's recommending that I watch a whole bunch of videos, and it's interesting in that most of these videos are car videos, except for the fourth one from the top, and that's a video from The Office. Now, it turns out that I also enjoy comedy a lot. I particularly enjoy stand up and over here, the fourth video from the top is one of my most underappreciated characters in The Office, it's Ryan. What I'm talking about if you don't know, that's irrelevant this is a comedy video and it's not like the others. Now, here's the thing, if you are YouTube, what is your decision making process? Your decision making process needs to be governed by the following. What set of things do I show Vivac on the right to keep him as glued to his screen as possible. They want me to click on the next thing. And so I've got this list of things on the right, how desirable are these things to me? And in particular, if I moved one of the videos up or moved one of the videos down, I'm obviously going to change the desirability of that video. In particular, it stands to reason that if you move The Office video up all the way to the top, I might actually click on it because I see it. Whereas if it's at the bottom of the screen, I don't click on it and as a result I don't see it. So, let me put that a different way. If you looked at all videos, I can average over all videos, and I looked at click through rates for the position at the very top, that's the position at the top right over here, position one. But when I looked at everything in like position one stands for reason that would kind of convert more. Now look at position two that would convert less, and three that would convert even less and so forth. But here's the thing, how much of this is due to the position for any given video. How much of the success of any given video is due to its inherent qualities? How much of it is due to where the video was positioned? Let me make this even more stark for you. Let's say you have only two videos, amazing videos and crappy videos, and that basically is a very extreme example, and nobody clicks on crappy videos, everybody clicks on amazing videos. But you have two spots, the top and the bottom. Now, if you actually take an amazing video and you only ever show it in the bottom spot, nobody's ever going to click on it and so you might actually be fooled into saying, you know what? This is not such a great video. And so the point is, how do I actually separate the position effect from the intrinsic quality of the video itself or the intrinsic preferences of the individual? This is super important because in YouTube thinking about what to show me on the right, it doesn't want to fall into a self fulfilling prophecy where there's some video by happenstance, it's only been shown at the bottom, but in reality, it's a star video. It's actually an amazing video. So, that's really the problem that these folks are trying to solve. Now in trying to solve for this problem, in trying to attack this problem. This is and by the way, this is really what YouTube, this is a problem truly at YouTube and this is how they tried to attack it. They tried to say, look, what we'll do is we'll take a model and we'll train that model with three pieces of content. So, I'll have three pieces of content, three inputs, the content, the video we were looking at and everything we know about that video, information about the individual and information about the position that we're showing you. And we can build some complicated model to go from there, deep network, let's say, to go from there to predicting the probability of a click. But the approach that these folks used is they said, well, look, let's take one of these models and they tried two models. What I'm going to call the direct approach and then what I'm going to call the shallow approach. I'll talk about both in turn. But think about these as being two alternative predictive models. And they fed in content the individual and position as features to each of these models and try to predict the probability of a click as the dependent variable. And at prediction time, in order to get the true intrinsic value of a piece of content, they essentially set the position piece to missing. And by setting that position piece to missing,

they were like, hey, what's the unpositioned biased estimate of CTR that depends only on the content and the individual? So, hopefully this makes sense. There's a lot to digest. I would suggest backing this up and watching it again. All we're saying is what we're going to do over here is train a model that looks like the one on the screen taking the content, the individual and the position, predict the probability of a click, predict the probability of a click, and then at runtime, you just set position to missing so that you get unbiased view of how desirable that piece of content is for that individual, not biased, by the position you're showing it at. Then you get get you an intrinsic sense of how interesting that content is. So, now the question is what model F should we use. And interestingly enough, they tried two models. One, as I said, was the direct approach, the other was the shallow approach. So, the direct approach, by the way, is literally that; it's basically let F be some deep network. That's it. F could be some deep network it sucks in content individual and position predicts the probability of a click. The shallow approach on the other hand, does something slightly different. First, it learns a representation of the content and the individual. Now, we've seen in deep learning that deep learning is very good at taking very complicated unstructured data and generating structured data out of it. Think of that G function as generating relatively structured information about the content and the individual. And we could, for instance, generate this by doing, solving a separate regression problem that simply looks at the probability of a click relative to just information on the content and the individual. So, I look at the probability of a click and I learn this function G that's forced to rely only on the content and the individual to predict whether the video is actually going to be clicked. And having learned G, I can now separately improve on G by adding a very simple function, let's say of the position. And so having learned G, I could say, look, in addition, the probability of a click depends on position let's solve a simple regression problem like a generalized linear model, and we'll learn that the coefficient on position is -0.1, say. So, I'm going to call this the shallow model. If you want to dig into it, this is actually work that YouTube did. The shallow model isn't quite the simple thing I'm actually showing you over here, but rather a shallow neural net. And you can dig into this if you so desire. Why might the shallow approach be better? Why might the second approach where you first train a complex model, but just to learn a good representation of the content and the individual and then having trained that model add on this shallow dependency, so to speak, on position. Why might something like that work better? Well, here's the issue. Imagine for a quick second that the deep learning model was so good that it could predict position based on content and individual. Let's just imagine the deep model could do that. If you told me the. So, let's say I had three pieces content, individual, position. But if you just gave me the first two pieces, I could build a deep learning model to predict the third. If you could do that then the model wouldn't be effective at predicting the value of the effective position because position is irrelevant to such a model. And the thing is that this happens quite often for the simple reason that these deep learning models are really, really complex; they can capture really complex dependencies. And as a result if we're using some algorithm that matches content and an individual to a position, all that the DNN will do is effectively reverse engineer that algorithm. And so that can actually happen pretty frequently. And if that happens, we run into a problem which people that study causality would refer to as the problem that position is endogenous. Position is determined endogenously in the data set. Because position is determined endogenously in the data set, we can't quite trust the model to actually recover causal effects for position. And so as soon as you have these issues of endogeneity, well, one thing is determined endogenously given the other things, you can no longer trust what's coming out of the model to actually give you causal estimates.

And I'll be honest with you, the shallow story over here makes sense heuristically, but we're still trying to understand as a community why something like this might work. So, view this as a heuristic as it were that seems to do a better job of isolating the impact of variables that would otherwise have been endogenous to the problem at hand. In fact,

in the case of YouTube, this works spectacularly well. In using the direct method to actually understand the relevance of content and individual, this was basically flat. It made no difference to performance and making the adjustment, making that position adjustment had essentially no impact on performance. On the other hand, using the shallow method where we were actively trying to understand what the impact of that endogenous variable actually was, so we could say look in the counterfactual we're actually changed the position, would click through have changed. In that case, engagement metrics actually went up, suggesting very strongly that the intrinsic values that were extracted by doing this were more truly representative of the intrinsic value of the content for that particular individual. And as such, this allowed the system to discover perhaps under appreciated content. But what I find really interesting is that this brings us almost full circle, because if we think about that insurance model that we actually started with, it was a simple model. We said, hey, the probability of the logarithm of the loss was the simple linear model in a variety of different features. The YouTube model that we just did is also at some level a linear model, except that it's a linear model on top of derived features in addition to features that are just natural. So, position was a natural feature to this prediction task, the G function over here was some derived feature. It's some derived feature, some index as it were describing the quality of the content or the quality of the content given the individual. But at the end of the day, we were actually able to build a simple interpretable linear model on top of these derived features. And I view this as one slick way of marrying these two opposing points of view. In addition to all the interpretability work we did, at the end of the day really must rely on a simple model that's only looking at simple features where you understand what the model is actually doing. One approach to perhaps enriching the model is to throw in some of these derived features like we just did for the probability of a click. Where these derived features rely on careful modeling using perhaps deep networks as we just alluded to for the problem at hand.

Video 22.6: Module Summary (3:53)

So, that was actually a fun journey. And so I want to just summarize where we've landed up, right?

In fact, I want to take a step back because we've been on a pretty long journey with predictive models. In the last five sessions, forgetting this session, but the five sessions that preceded this, on deep models, particularly at least for me, you're very exciting because we're able to build these super complex models that really get to understanding the structure in our data. That's a blessing. But the curse that comes with this, is that these models are, they're black boxes. Really what we did today was, maybe to some extent, mitigating the fear that these are black boxes, right. So, interpretability and the tools we looked at today, I would like you to take away that these tools let us effectively cut holes in these black boxes, right? Let us understand what's going on within these black boxes. What did we do with these tools, right? Number one, we said, hey, look, with these interpretability tools, we can audit a complicated model and we can look at what the model is looking at in order to make its predictions. Doing this would ultimately drive confidence in the deployment of that model, cache problematic issues before they become real problems, right? Just like we did with the bicycle for self-driving cars, and the example of using a machine learning model to detect pneumonia. But upping the ante a little bit, I'm hoping I also made you aware of the problem that data, especially data that comes out of complicated decision making systems, can have immense selection bias. And so because there's immense selection bias, just like that YouTube example, right? There are aspects of what's going into the model that are endogenous to the process that generated the data. So, position is endogenous because there's some algorithm running behind the scenes that's making positioning decisions based on a whole bunch of other things. And so when this happens, prediction, we can't really do causal attribution comfortably, right? And to be very clear, this problem is not super well understood, but at the very least we should be aware of it, right?

That hey, when we look at machine generated data sets where some of the input variables are endogenous, we can't take for granted that what the model attributes to these endogenous variables is truly causal, so that if we change those things, the predictions would behave in a manner that the model predicts. We saw a hack to actually addressing this issue in the context of YouTube and the hack appears to work reasonably well. But at the end of the day, I think the story here is one of just being cautious, of understanding what the limits to these models are when some of what you're actually putting in is endogenous. And so to summarize, this very exciting journey, with deep learning models where we've seen a whole bunch of different sorts of deep learning models, we in a sense got into a state of the art. We not only understand how to build state of the art models, we also are beginning and have begun to appreciate some of the subtleties and what we need to do to address those subtleties when we take these deep learning models and put them into production in practice.