SuperDataScience

SDS PODCAST EPISODE 993: HOW TO BUILD AI-FIRST ORGANIZATIONS, WITH JACOB MILLER AND JEREMY MUMFORD

Jon Krohn:	00:00:00	Anything you put on the teleprompter, Ron Burgundy will read it. That single line from Anchorman explains why most AI projects fail and what to do about it. Welcome to episode number 993 of the SuperDataScience podcast. I'm your host, Jon Krohn. My exceptional guests today are Jeremy Mumford and Jacob Miller, who serve as lead AI engineer and vice president of platform intelligence respectively at Pattern, a giant Utah-based tech company that IPO'd on the NASDAQ Exchange about six months ago. Jacob and Jeremy are also co-authors of the brand new Wiley book, Architected Intelligence, and this episode focuses almost exclusively on this invaluable new book. Whether you're a hands-on practitioner or not after this episode, you'll have all the key guidance you need to successfully build AI features, AI products and AI first companies. Enjoy.
	00:00:53	This episode of SuperDataScience is made possible by Anthropic, Acceldata, and Cisco.
	00:00:59	Jeremy and Jacob, welcome to the SuperDataScience Podcast. I'm so happy to have you guys here. Where are you guys calling in from today?
Jacob Miller:	00:01:06	From Lehigh, Utah, just outside of Salt Lake City.
Jeremy Mumford:	00:01:10	Yeah, Same here.
Jon Krohn:	00:01:10	Both of you. Yeah. We just got a head nod from Jeremy, which wouldn't translate to our audio only podcast listeners.
Jeremy Mumford:	00:01:18	Well, you know, Jake and I are the same place. I'll let him speak for both of us.
Jon Krohn:	00:01:21	But you're not actually in the same building right now, right? Jacob looks like you're- No, I'm working from home
Jeremy Mumford:	00:01:25	Right now.

Jon Krohn: 00:01:26 Yeah. And then you guys both work together at the same, I guess, is it Lehigh, Utah that you both work at as well? Yeah. Tell us about your business.

Jacob Miller: 00:01:37 We both work at Pattern and we run what's called the Platform Intelligence Team. And so the Platform Intelligence Team is the AI platform underline all the AI tech that Pattern is an e-commerce accelerator. It's the largest e-commerce accelerator in the world, public company. And so we manage the agents, the workflows, the evals, all of those components of the AI platform at Pattern.

Jon Krohn: 00:02:01 Really cool. It must be nice to be working at a business that is so agent forward where you get to really be doing these things. And so Pattern is also in Lehigh, Utah?

Jacob Miller: 00:02:11 Yep. That's where their headquarters is.

Jon Krohn: 00:02:12 Nice. And how long have you guys both been there? Also, tell us about the roles that you respectively have at Pattern.

Jacob Miller: 00:02:20 So I'm the VP of platform intelligence, and so all of the different parts of the platform roll up. And to me, I've been at Pattern six years. And so I started on the data science side, then it became Data Science Plus AI and then evolved into more exclusively on the AI side.

Jon Krohn: 00:02:36 It sounds just like the podcast in the last few years.

Jacob Miller: 00:02:39 Yes. We've been following the same journey, Jon .

Jon Krohn: 00:02:42 Exactly. Yeah. And Jeremy, what's your role? It's a lead AI engineer or something like that, right?

Jeremy Mumford: 00:02:48 Yeah. So I've been at Pattern for four years. I kind of started out doing more traditional data engineering and software engineering for a lot of our ad tech, but my educational background was in data science. So Jacob

eventually pulled me over and said, "We really need your talent set here. We think it'll be a good fit for this team." And it's been kind of a rocket ship journey ever since. It's been good.

Jon Krohn: 00:03:13 That's really cool to hear the rocketship journey at the office. And I think there's like a booster rocket that's about to take off for you guys and take you even further into the stratosphere because you guys, Jacob, Jeremy, you're actually not on air today to talk about pattern of what you guys have been doing directly professionally, although I suspect lots of examples are going to come up as we discuss this, but we're here to discuss the brand new book that you co-wrote together. And at the time of us recording this, we're about a week away from the book being publicly available and it's going to be by the time this episode is live, 100% for sure the book will be about a week fresh. So if you really like that new book smell, then the book to pick up is Architected Intelligence Principles for Building AI First Organizations and Technologies.

00:04:02 So like the main title there is Architected Intelligence. And so yeah, this episode is going to be focused on that topic. It's probably even going to be called Architect Intelligence and we're going to be going into, this is a really interesting episode because you're both highly technical hands-on practitioners, but actually most of what we're discussing in this episode is going to be applicable to any kind of listener who's interested in successful development and deployment of AI projects.

00:04:36 You guys might end up mentioning open source Python libraries or something at some point, but that's not really the main focus. This is more, it's like higher level. It's more conceptual and kind of less code level, right?

Jeremy Mumford: 00:04:48 Yes. And I think that was our goal going in was we've seen a lot of amazing, great content come out with very focused

content on use this library, use this tool, but we felt like there was a lack of information and content around kind of these higher level concepts and principles, especially what we've seen as AI is moving so quickly. Somebody recommends you to use GPT 3.5 Turbo in their tutorial or guide and you're like, "Well, that's now instantly outdated." So our goal going in with the book was how can we actually create content that will last more than a year, that'll last five years, even 10 years, and will continue to be extremely relevant and extremely useful.

Jon Krohn: 00:05:27 Yeah. Nice job finding a topic that'll be relatively evergreen. I think pretty much everything we discuss based on the topics that I have planned for us are concepts that will serve listeners for years and years to come. So great. You guys, despite both working at Pattern, your whole career history is quite different. And so I expect that you guys came at architected intelligence from different angles. So Jeremy, you come from deepfake research at BYU and satellite imagery for analysis for defense use cases. Jacob, you come from 23 years of experience across risk modeling, internal consulting and building an academic data science program. So how did your perspectives come together to create this book and perhaps even more importantly, why did the world need architected intelligence now?

Jeremy Mumford: 00:06:23 So I think the best way to answer this question is to go back to the beginning and tell the story of how this all started. I'm sitting at my desk, Jacob and I, we've been working together a couple of months at that point. Things have been going really well and he comes up to me and he says, "Jeremy, I've got this book that's been in the back of my mind for years. I've been thinking about it and it has to do with all the things that are going on in the world of AI right now. And I feel like now is the right time to write that book, to put it out in the world. I feel like a lot of people really need it and I want you to write it with me. " And I go, "You want me to write it with you?

	00:06:58	Why me? " I'm like, "There's a lot of really smart people across the country, even in pattern that would be great co-authors. Why me? " And I think this is one of the things I love about Jacob is he is a person with a lot of humility and he recognizes his strengths and weaknesses and he's like, "I have tons of experience in the world of data science and the world of business, but you're coming from a perspective where you're much newer in the industry and you've been focused on more of the engineering side. You still have that educational background in data science where you did all your research." But he's like, "I think that our different perspectives coming together will allow us to write the book that people need." And as we wrote the book over the course of a year, I think that was true because what we've seen across the landscape of AI engineering is an explosion of people getting into building with AI, right?
	00:07:58	I'm sure you've felt this as like a data science podcast. I'm sure your audience has exploded the number of people that are interested in the topic have grown. There's people coming in from all different sides. And what we've seen is people are coming in from the data science side and now they're expected to build applications for users and software engineers are coming in and they're expected to understand how reinforcement learning works and all of these other things. And so we found that our two perspectives coming together was kind of the missing piece of the puzzle that made the book work.
Jon Krohn:	00:08:30	Nice. That makes sense. That was a really great answer where I thought it was going to get a little bit juicy where we were going to get to hear Jacob's weaknesses, but then you were like, his weakness is that he's been in industry so long.
Jeremy Mumford:	00:08:47	But I mean, there's truth to that, right? Of having a fresh perspective. And I think that's what it's important to have sometimes.

Jacob Miller: 00:08:55 And also Jeremy's just super, super obsessed into all of the details of which adding that perspective from top down to bottom up and combining the two of them. And so with components of the book, each of us could bring different strengths to each component of the book.

Jon Krohn: 00:09:12 Fantastic. Well, let's get into some of that book content. Our listeners, I'm sure are dying to get into the nitty gritty. So we're going to start off with some stuff. I'm going to kind of survey what I thought would be the most interesting topics from across the book. We're going to start in chapters one and two and we're going to start by talking about something called the User Agnosticism Tenet, UAT, where you design products and processes so that they can be executed by a human, an AI, or some hybrid mix of human and artificial agent. And I thought that this was one of the more interesting ideas in the book. Walk us through a concrete case where designing for agnosticism, whether the user is going to be human, artificial intelligence or some mix, where that agnosticism saved a project that would otherwise have died.

Jeremy Mumford: 00:10:06 Yeah. I think a concrete example of where this agnosticism tenant can be really helpful is maybe the example of a sales team. You got a sales team, they're trying to automate parts of the sales experience. As a sales team, you're like, "There are an endless number of people we can market to, sell to. How do we go after all of them?" And so AI is really enticing where you can start to automate parts of that outreach parts of that communication pipeline. I think where a lot of people start to see it break down is they automate their pipeline and then they lose the human touch, right? And that's where a project can fail is you've tried to automate something, but you did it in such a way where it's so automated now there's no opportunity for a human salesman to jump into the call, right?

	00:10:56	And then a human is frustrated because they're talking to a robot and the connection is lost. So where we see a lot of people succeeding is where they build out maybe some sort of chatbot experience where someone can start a conversation with a chatbot, get some initial information about the product, about the company, the service, and then at any given point, a human can be pulled into that conversation and pick up where the robot left off. And this is such a powerful pattern that can be used in many different places. You can do it in customer service and sales, you can do it in engineering. If you think about the flow that people are doing with Copilot AI coding tools where it's a flow back and forth between the AI and the human writing code,This pattern is applicable in many different places and if you stick to it, that will help you avoid those failures of getting completely locked out of the system.
Jacob Miller:	00:11:51	Yeah. Otherwise, you end up with a complete refactor where you say, "Hey, we're going to automate this entire thing." You automate the entire thing, but if you decide we want to pull back in certain places and you have to do a massive refactor, it doesn't work.
Jon Krohn:	00:12:03	Yeah. I mean, when I read this tenant, this user agnostic system tenet, it actually seems really obvious to me that this is the way that we need to be designing for in the future for any kind of interface, for any kind of interface, not just the future today. We're already there where with especially now in the past couple of months, the proliferation of CloudCowork, any of these kinds of agentic tools, you have to assume that anything that you might have initially ... If you designed something years ago, it would have been designed for a human user and today you're going to have machines crawling over those same kinds of platforms and trying to interact with it. And if it doesn't work well, you could be shooting yourself in the foot in terms of being able to get automated processes happening on your platform. So thank you very

much for putting this into words and kind of just giving a name to it.

00:13:01 Another part that I like about the first chapters of your book is how you write that failing slowly is costly and failing quickly can become your superpower. So if you're in an organization and we're going to get to speed again, well, at least a few more times actually in this episode, speed is kind of a key thing today it seems, but being able to fail quickly is something that's so important. However, organizational antibodies like the health system, something that was designed to keep an organization healthy, these structures in an organization, let's call them antibodies to give it kind of like a biological analogy, they actively fight against failure. They're always trying to make sure that everything succeeds.

00:14:05 And in this day and age where things are moving so quickly in terms of technologies, capabilities, what you could be doing with your platform, that historically useful antibody is now it's like an autoimmune disease. So what are your tips on forcing failure fast on AI projects?

Jeremy Mumford: 00:14:28 It's an interesting problem because I think enterprises, they have these organizational antibodies as you describe and they start to really want to stick with projects and ideas maybe longer than they should. And I think also they are kind of accustomed to the traditional way of software engineering and building products, which is that the initial product will maybe not have full functionality, but if you stick with it long enough, you can build out to the point where it's useful. But now building with AI and building AI infused products and features going from zero to prototype is nearly instantaneous. And so it requires a mindset shift and an organizational behavior shift that can be hard to execute at the enterprise level. Just to give an example, I remember early on at Pattern, we wanted to build a text to SQL chatbot and having interviewed hundreds of people for AI engineer roles and talking to

people across the industry, I swear every single company tried to do this.

00:15:37 We all tried to build some sort of text to SQL chatbot that would just solve the problem of being able to retrieve data from our endlessly complex databases. And the thing about all these demos is that within like a day you could have an amazing demo that seemed to kind of have something amazing on the surface and so a lot of companies kind of committed to that and they were like, "Well, if we just stick with it, we'll be able to kind of keep going with it. " I think the key thing there is that now that it's so easy and quick to build prototypes, you really need to prove value on day one. If you can't prove value on day one, then you need to toss it out and start all over again. And that's a really hard thing to tell people to just toss out their ideas, but I think we're seeing that shift.

00:16:26 I look at AWS just this last week, they've kind of announced that they're doing this massive shift from their old AWS bedrock system to this new project mantle and they talked about how they had six engineers over like two or three months just completely rebuild their entire AI infrastac from the ground up. That's the kind of commitment and resolve you need to have in this new age to just say, "We got to start over from scratch and build something with immediate value from the beginning." And so that's really what I think it comes down to.

Jacob Miller: 00:16:58 With the one approach when we're creating something of value, the product cycle is that you define what we want to build, we then build it and then we get feedback and it used to be that building was the bottleneck. So you could be really bad. So if it's defined build and feedback and if build was always the bottleneck, you could always just be bad at definition and feedback because that was never the bottleneck anyway. And so when building no longer becomes the bottleneck, the critical, the crucial step is definition and feedback. And so one of the organizational

changes we've seen help is actually narrowly defining these lanes and say, "Hey, you two, three people, you can make whatever decisions you want. This is what you're chasing after. This is what success looks like. " And you guys can try and fail over and over and over within that lane.

00:17:47 And by involving fewer people, it ends up that those who are working on the project feel much more comfortable trying things out and failing because you're not failing in front of such a large audience.

Jon Krohn: 00:17:59 I really like that framing of, yeah, what was it, building, defining-

Jacob Miller: 00:18:03 Yeah, you define what you're going to build, then you build it, and then you get feedback and you're just repeating that cycle. So to go through that cycle today with AI, it's super, super fast, but going through it traditionally was build was so slow that definition and feedback just weren't important, but now they're critically important.

Jon Krohn: 00:18:19 You could probably even get a lot of the definition right along the way as you're building because everything is so incremental, so slow and smart software engineers, smart architects are thinking about the questions that are the big questions that need to be defined as you start to build and you think, "Oh, there's a potential issue or there's a potential opportunity." But now when you can just take that definition and put it into cloud code, wait a few hours and get your working application, it's a very new world. It's a brave new world where you hear from some of these pioneers that are using codegen tools really prolifically, you hear that they've gone from spending 80% of their time coding to 80% of their time getting spec documents correct.

Jacob Miller: 00:19:08 Exactly. So it shifts the way from that bottleneck of builds to the other steps of the process.

Jon Krohn: 00:19:13 All right, great answers there. Let's move on now to chapter three. And you opened chapter three of your book, Architected Intelligence with a Parable, The Parable of Zillow Offers. Zillow Offers was a project you guys can describe in more detail. You're more familiar with it than I am, but my understanding is that it lost almost a billion dollars on what was at its core a sophisticated AI workflow exposed to an adversarial environment it couldn't see and that wasn't anticipated as Zillow offers was being developed. So tell us about this failure and maybe what could have been architected differently about Zillow offers so that this billion dollar loss didn't happen.

Jacob Miller: 00:20:02 Yeah. Adversarial AI is something that a lot of data scientists are familiar with, but fundamentally with Zillow offers, it's what's called this, it's a winner's curse scenario where they built these really, really great models to understand what the value of a property is and they can make an offer, but if they make an offer that's too aggressive and too high, the customer's going to take the offer because it's a great deal and Zillow loses. But if they provide an offer that's too low, the person doesn't like the offer, they say no, and Zillow doesn't get anything. And so your models have to be very, very good and very honed in order to execute. And they started this as a trial, as a pilot period, and it was actually going really well at first, but then they got hit with COVID and some other changes. Basically, you have model drift, you have changes in the feature space and any type of adversarial position is actually just a change in the feature space for data scientists.

00:20:54 It's like, "Oh, we are out of distribution. I didn't plan for someone to attack me in this way, and we're left in a situation where we lose." And so within this modern AI space, there's actually way more adversarial

environments being built. One of the projects I would hate to be building right now is some sort of customer service chatbot that gives out discounts or free stuff in order to resolve customer complaints because the thing right now is someone posted on Reddit, here's how you can trick this AI agent to give you free stuff. And so you have an army of people out there just ready to take advantage of you and it's a constant shifting in the feature space. And so being aware of what is going on and when you're actually in an adversarial position, are there other people out here who could take advantage of this, could take advantage of me?

00:21:40 It becomes this shifting minefield. And so within those types of projects, having great observability over them and identifying behavior that's outside the norm is critical for those and flagging those immediately and being able to immediately respond would have been super helpful. So in the Zillow situation, many companies that were in data science, I mean, you probably saw this at the beginning of COVID, they literally just tossed their models in the garbage and went, Luke Skywalker style, I'm going to just use the force and turn off my targeting computer and just make all of my forecasting plans and make it on the fly because the models were garbage because the present was not like the past. We had drifted outside the feature space. And that is true for so much more in AI. There's a concept from data science that's core to data science, but within the AI world, within the AI environment, it's much, much more common than it was today or than it was before.

Jon Krohn: 00:22:35 Nice. That was crystal clear, Jacob. Such a nice response. So nice in fact that it's the first question I've asked that not both of you has chimed in on. Jeremy's just stunned by how perfect your answer was.

Jeremy Mumford: 00:22:48 It

Jon Krohn:	00:22:48	Was
Jeremy Mumford:	00:22:48	Perfect. No need to jump in.
Jon Krohn:	00:22:51	Later on in chapter three, you guys make the claim that you should build the hands before the brain, the workflows before agents, which is an interesting claim to make because today it seems like we're hearing everywhere that we should just be diving right into agents. So what are your concerns about people in our field skipping what you view as a foundational step, getting the workflow right first and maybe this is actually a good opportunity to also compare and contrast the differences between a workflow and an agent.
Jacob Miller:	00:23:25	Yeah. So the fundamental difference between a workflow and an agent is the orchestration. For an agent, you give it an objective, you give it a set of tools and you say, use the tools in the correct order of whatever you think is right to achieve this objective. In a workflow, you say, use tool A, then B, then C, and then maybe there's a branch in that tree and you're giving it the path already. And so one of the points that we make is if you can't get a workflow operational and being happy with the results where you're telling the exact path to give, why would you be any more happy with an agent that's just got to make it up on the fly? Now with that being said, there is this natural progression that people are using right now where it goes like skills, then workflows, then agents.
	00:24:10	And so you use, instead of mapping out a process, because that can be tedious or complicated or I don't really know process mapping, because you're trying to get subject matter experts involved here, they can create a skill real quick and it's just a markdown file, describe it. They can use the skill, see how well it performs, and then update the skill and iterate and that becomes this natural landing point for a lot of teams. We're just going to build skills. At a certain point you say, why call these skills

manually? That doesn't really make sense. Turn this into a workflow and then you can get them into a workflow. It's then taking the job and then if you need to dial up the capabilities, you transition from workflow to agent. And so that's the natural path that we see at a lot of companies right now.

00:24:51 It's start with skills, very accessible, very simple. You then take those skills, package them up into a workflow and if you need it, roll it up into an agent.

Jeremy Mumford: 00:25:00 I think there's a few other problems with helping people understand this distinction between workflows and agents and helping them understand they should start with workflows. One is the boundary is very fuzzy, right? You can have agents calling workflows, workflows calling agents. It can be fuzzy on where exactly the line starts and ends with which one is which. And then I think the other thing that's challenging is the industry, naming products, naming techniques. You see a lot of stuff marketing itself as an agent builder when in my personal view, I view them more as workflow builders. And so it can be very hard then to try to go and explain this to people and say, "Oh, congratulations, you built a workflow." And they're like, "Well, I thought I built an agent." And you're like, "Well, I guess we can call it an agent, but it looks like a workflow and it talks like a workflow.

00:25:51 So let's call it a workflow." And I am very pro workflow. I think there's so much power in having a deterministic proces that you can rely on, you can count on and it makes things faster, more reliable and often cuts costs. Think about setting something up with Claude code that's just on a routine that everything is orchestrated by the agent. It's all passing through the context window. I can get really expensive. There's a lot of power in building out a Bash script and you can have the agent write the Bash script for you, but do we have to have everything go

through the context window? Does the agent have to handle everything? That's where you start to see these horror stories of people saying like, "It went off the rails in week three and deleted my production environment." It's like, let's stick with deterministic workflows first.

00:26:40 Once we've kind of built out our skills and we're confident, then we can kick it up to the next level and go tackle the kind of craziness of non-deterministic, unpredictable agents.

Jon Krohn: 00:26:52 Yeah, this makes a lot of sense to me. Having an LLM involved in a workflow that you could imagine how you could code this up as something more deterministic is a really bad idea from a cost, from a latency perspective, from a risk perspective. I think there's a risk for those of us working in data science and AI that we can kind of end up in the situation where instead of kind of typing that really short email and just saying, "Yes, that sounds good," we're like copying out of Gmail into Claude or using Cloud cowork to be coming up with an answer to the simple thing instead of just typing it up ourselves. And so because LLMs are so widely useful, we can end up throwing them into any situation within an organizational workflow when it's serious overkill. And yeah, we end up with all of these downsides with cost, latency, risk.

00:27:53 So try to use your brain still people.

Jeremy Mumford: 00:27:58 And if I can share just a quick story from just an example, I remember we were working on this project and it was all AI, all workflows, just a huge mess of AI everywhere and we kept getting user feedback that they were not happy and we kept fine tuning the prompt, fine tuning the prompt. Finally, I just turned around and I said, "Okay, we're just going to rip the AI out of this part. We're just going to replace it with some canned statements and user complaints disappeared overnight." We just replaced it with a decision tree of like, if it is

above this score, we'll say this. If it's below this score, we'll just have some sentences where we fill in different variables and user satisfaction just like shot through the roof. And so I'm like, I feel like half my job as an AI engineer sometimes is ripping AI out of places and people should not be afraid to do that sometimes.

Jon Krohn: 00:28:45 Really great example. You guys have had so many concrete examples so far in the episode. It's been a delight and so easy to understand what you're describing when you give these clear examples. You mentioned there that you're an AI engineer, but you talked earlier in the episode, Jeremy, about how you were previously a data engineer. And so this next question is related to that specifically. You can tell me about Titan Logistics. So you can tell our audience about the Titan logistics story. This is from, we're kind of now in the middle of your book. We're like tackling chapters four to six here out of, I guess you have 12 chapters in the book. If I'm remembering correctly, I'm getting some head nods. So yeah, we're getting to kind of the halfway point in the book and the halfway point in the episode. Wow. And so yeah, tell us about the Title logistics story where they were losing money on both the hedge and the physical supply chain at the same time until an AI firm asked for the boring data.

00:29:40 So fill us in on more about that. But I think this is an interesting example because I think this happens a lot where organizations are looking for advice on AI, but what they really need is an intervention on their data engineering.

Jeremy Mumford: 00:29:57 Yeah. I think this is such a great example that we have in the book and it's so powerful because I feel like a lot of people in data science have maybe already felt this story in their personal lives, but maybe people getting into AI engineering or discovering it for the first time. I feel like I've heard so many stories from people in the data science

industry where they talk about, I got hired at X company and I was asked to do data science and I walk in on my first day and I ask, "Where's the data?" And they go, "What data?" And they dive into the company data and it's a mess and they spend the first two years of being a data science doing data engineering. I think this is the big key thing, you have to have solid data in order to make AI work effectively.

00:30:47 I think data science has kind of figured this out over the years and now we're hoping that we can help kind of pass that lesson on to AI engineering, which is that data is the foundation of everything. We've got a quote that I really like in the book from the movie Ron Burgundy, like Anchorman and

Jon Krohn: 00:31:07 It

Jeremy Mumford: 00:31:07 Says-

Jon Krohn: 00:31:09 What was that Shakespeare play? Oh yes, Ron Burgundy.

Jeremy Mumford: 00:31:12 So Anchorman, legend of Ron Burgundy. We've got a quote in the book that says to this effect of, "Remember, whatever you put on the teleprompter, Ron will read it. " And so whatever data you put in front of your AI, they'll take it as truth. And so data engineering is critical with AI because the data you put in front of your AI is what they will take as the truth. And so really a lot of AI engineering starts to fall back to data engineering. The difference between before and now though is previously we were dealing with structured data in the case of Titan logistics and our data science friends, it was a question of where are the numbers, where's the data? Now with this new generation of AI engineers, it's a question of where are the PDF files? Where are the images? Where are the Google Docs and which ones are authoritative?

	00:32:04	Which ones are the truth? Which ones are throwaway junk? How do we chunk it? How do we embed it? How do we search it? All of these questions are now forming the bedrock of AI engineering in 2026.
Jon Krohn:	00:32:16	So yeah, thank you for that clear example, Jeremy. And this actually brings me perfectly to the next question that I was going to ask about. So you talked about how with that anchor man legend of Ron Burgundy example where the agent in LLMs in general, they know what they know from data that they've been trained on. And so you argue in your book that hallucinations are usually gaps in what you call cannon. So the body of information that an AI system accepts as authoritative truth not flaws in the model. So the AI is only improvising, this is a quote from your book, because we fail to give it the right script. And that's an important reframe. How do you get engineering teams to stop blaming the model and start treating hallucinations as a data curation problem?
Jacob Miller:	00:33:10	With hallucinations, it's often like you said, it's this categorical error that we see over and over from AI engineers where if the model hallucinates, they just assume that the model messed up when it's most often for the model to hallucinate when it didn't have any good reference. If you tell the model like, "I live on the third floor, what floor do I live on? " It doesn't say you live on the second floor, but if you haven't mentioned what floor you live on at all and you haven't given any of that context and then you ask it, it might make something up or it might try to infer from information that's somewhat unrelated. And so when you look at a hallucination with modern models at least like in 2026 models, what that indicates is actually we didn't provide any good context about that question about that task and the fact that it hallucinated is actually throwing up a flag to say, "We are missing critical context here," which means we either don't have that context at all or we failed the retrieval

entirely saying that was out there, but we just didn't go out there and grab the right one.

Jon Krohn: 00:34:12 It kind of reminds me how earlier when you were talking about Zillow offers and the world changes so the model that was trained is no longer relevant to the world that we're in. It kind of seems like something similar can happen here with context where you could be, say you're an HR manager and you're like, "Sweet, we're going to get this agent set up so that it can handle all the common questions that I have to deal with every day." I get dozens of emails every day on simple HR questions that people could be looking up in a guide. So let me throw all of the HR PDFs that I can find, all the guides that we have. I'll just throw that all in as context, but within that context, there could be conflicting information. Maybe you've thrown in an HR guide from this year as well as one from two years ago and a policy has changed over that time in some significant way.

00:35:11 It's now the inverse of what it was two years ago, but maybe this document didn't have a date on it. And so the AI model, it might pull out that information from two years ago and confidently give an answer that is the opposite of what's true today. And so you guys, in your book, you suggest a new role type in the organization that I had never come across before, I guess because you guys invented it, which is a role called Canon Manager. And so this Canon manager is making sure that the context that AI agents are working with or LLMs, I guess in general, are working within an organization are correct.

Jacob Miller: 00:35:54 Yeah. The Canon managers in general, we came up with this term, but there are many, many different titles for it across the industry and every organization has just kind of sprinkled their own. And so it's like, well, we'll just consolidate them into one single title of Canon Manager, the person who is in charge of what is true. Not because it can't change, not because that document can't change,

but whatever you tell Burgundy is true is true. He will read the prompter. He'll read the teleprompter.

Jon Krohn: 00:36:26 Technically, not Ron Burgundy though, is it? Is it Ron Burgundy in it that does that? No, it's another character, right? Well,

Jeremy Mumford: 00:36:32 I think someone pranks him putting it on the teleprompter and

Jon Krohn: 00:36:36 Then

Jeremy Mumford: 00:36:36 He incorrectly reads it.

Jon Krohn: 00:36:37 Oh, so it is Ron Burgundy that does it. Yeah. Somehow I think it's been probably 20 years since I watched that movie. I was thinking it was the I Love Lamp guy that would read whatever. My bad, my bad.

Jacob Miller: 00:36:48 I would say that the anti-pattern of a Canon manager, so to describe what it's not before to describe what it is, is let's say there's critical context for an agent to behave correctly. And I would say that the most common mistake that we see is that it's put somewhere in GitHub in a repo saying, "Hey, you should follow these exact instructions that are within this repo and you will do what this engineer has placed there." And then if it goes wrong, that GitHub repository is not very public within the company. It's not very public to say the product managers or others who are responsible for its performance. And if they want to update and change it, it's up to the engineer. So someone would say, "Hey, it's up to the engineer to make sure that this context, that this canon is correct and up to date." But we argue, no, they are not the subject matter expert.

00:37:37 They're not the one who knows the most about it. And actually you have to extract that context out from the deep recesses of some GitHub repository and move it over and put it in front of the subject matter experts that

actually know the most about it and provide them the tools to minimize that friction and proactively identify those conflicts so that they are accountable and responsible and they feel that they are the ones to make sure that this agent or this workflow performs the way that it should.

Jeremy Mumford: 00:38:06 Yeah. So it boils down to how do you make it really easy for anyone at the company, people in HR, people in leadership, people in engineering, no matter what your role is, how do you make it really easy for them to publish and maintain and update knowledge that is accessible by agents that are answering questions for other people across the company and it's hard. It's hard to make that experience easy for people to make it accessible, powerful, and accurate.

Jacob Miller: 00:38:33 And for us, it's been so easy for the olden days where it's like, "Hey, use the company Wiki and no one uses the company Wiki, right? It just doesn't work." And it's this new world where actually if we had a super well-functioning company, Wiki, we could do a million things today that we otherwise couldn't. And so the responsibility for it dials up so much. The impact, that accountability and for us, probably the biggest unlock for our company and many others that we've worked with is AI tools to minimize the friction to contribute to these Wikis. And so it's like, oh, here's a conflict. AI can recommend how to consolidate that conflict and recommend changes so that they can accept the diff or they can adjust the wording in one little place. And so minimizing that friction in this new world we found is absolutely critical.

Jon Krohn: 00:39:24 Great. So now that we've talked about data, let's talk about the models themselves. So this is chapter seven and eight in your book are focused primarily on models and an interesting analogy, a sobering story that comes up in those chapters is the fine tuning trap that

Bloomberg ran into with their Bloomberg GPT. So they spent an eight figure sum training a model, Bloomberg GPT, which for a few weeks was a really big story because they had trained this model on all their financial data and it outperformed GPT3 buy a lot on financial queries for the most part. So it seemed like an example of this niche model that was going to provide a lot of value and then a few weeks later, GPT4 came out and nobody's used Bloomberg GPT since. So eight figure sum for a few weeks of utility, not a great investment probably.

00:40:24 So given how rapidly frontier models advance, what's the checklist that our listeners should run through before committing to a fine-tuning project today, especially if it's going to end up being expensive?

Jeremy Mumford: 00:40:36 Yeah, I think I have to be careful what I say here because I have really strong opinions about fine-tuning, but I'll just keep it really simple. More people do fine-tuning than they should. I'll just leave it at that. You should really think twice about committing to fine-tuning before you get into it. And so let's talk about that checklist.What should you check off? First off, you've got to make sure, am I an organization that's ready to take on this challenge? Do I have the necessary talent? Do I have the bandwidth to commit to something like this? Because yes, it seems simple on the surface. You're like, "I just throw some data at it. Boom, it's an expert in the problem I care about now." But there's so much that comes into that with all the things we've talked about earlier with feature drift and keeping the model up to date and the challenges of new models continually getting better and better.

00:41:29 I think it really comes down to, do I have a really narrowly defined domain or problem that will not change over time? I think an example of this is anything to do with classification. This is a great place to apply fine-tuning. You're like, "I have an extremely straightforward classification job that I need to do at scale. I'm talking

millions or billions of data points. If you have a problem like this, it may make sense to do fine-tuning. Outside of that, I would really encourage people to hesitate before getting into fine-tuning. I think in 2026, most people have figured it out by this point. Two years ago, I had this opinion and it was a very controversial opinion. Today it feels less controversial, but really think twice before you jump into it.

Jacob Miller: 00:42:17 One of the points that I bring up with teams that are considering fine-tuning is the opportunity cost. It's like, look, you can spend all this time building this fine-tuning model and you might be able to get it from 93 to 95% accuracy. Is that worth the amount of time and maintenance to support that over time versus all the other projects that you guys have been wanting to do? And so yes, if you had unlimited budget and unlimited tokens and unlimited resources, fine-tuning makes a lot of sense, but if you're looking at where's the delta in terms of the improvement in performance, yeah, classification, you definitely see that. And there's other use cases where you see fine-tuning worth it, but in a world that moves as fast as ours and the opportunity cost of all the projects, every time we choose a project, we didn't choose five others.

00:43:09 It better be a really, really good project. And for us, fine-tuning will probably make more and more sense over time as we say the technology gets better and the infrastructure gets better, but generally it's like more people are doing it than should.

Jon Krohn: 00:43:22 I think there's an instinct for those of us who historically as data scientists, for example, have been really excited about training and deploying models. For us now, most of us, 99% of us in the AI space aren't working on frontier models training huge LLMs from scratch. So instead we're kind of like, okay, well, I can be using some kind of Laura adapter and fine tuning this relatively cheaply, but your

time is still very expensive relative to a lot of other options out there. And so just curating the data, the weeks spent probably at a minimum in most cases to get things right in terms of fine-tuning and get that 93% to 95% delta in performance exactly as you say, Jacob, it's going to be the case most of the time that we should have just used something off the shelf, which is kind of disappointing for ... I guess we're looking ... We want to be doing something really cool and really meaty, but as you guys point out in your book, and we're actually getting pretty close to the question where this is going to become the main point, that speed is more valuable than anything these days and it's pretty hard to do that if you're developing custom models, unfortunately.

00:44:46 All right, cool. I kind of got the final word on that topic, although we're getting to the speed thing again later so you guys can chip in again. I've got one last question from the midpoint of the book before we get to that speed thing, which is, well, actually, this relates to speed as well, but it's less about organizational speed. It's more about models. So in your book, you talk about the intelligence cost latency triangle. So if you're listening to this in an audio only format, I'm just going to say that one more time. Imagine a triangle. Don't imagine this too crisply if you're driving a car right now, but imagine a triangle with three points. Wow, try that.

00:45:32 And on each point is a different word. So we've got intelligence, say at the top, cost in the bottom left corner and latency in the bottom right corner. I'm kind of assuming that people by default when they imagine triangles, it's like equal lateral and there's a point up. I don't know why. You guys, did that happen for you guys too? And yeah, this intelligence cost latency triangle, you talk in your book about how you can buy any of those two corners but not all three. So you've got to pick. You can have high intelligence and low cost, but then your latency is going to be high and you guys could work through all

the other possibilities, but basically you can't have a lot of intelligence, low cost and low latency all at the same time. However, despite you saying that, you then go on to provide techniques that you describe for cheating the triangle.

00:46:33 So things like dynamic routing, streaming, semantic caching, ensemble methods. Which of these techniques do you think is the most effective or maybe the most underused?

Jeremy Mumford: 00:46:44 Yeah, I think it's fun to talk about cheating the triangle because it kind of makes you feel like really clever. You're like, "Hi, cheated the triangle." I think probably the most important one is streaming, right? If you look in the world of software development, so much matters with latency and how quickly can you get something in front of a user I can't remember the exact statistic, but it was something about like every 100 milliseconds you add, you lose like 10% of people, something crazy like that. And so every second, every millisecond counts when you're building applications for users and nobody wants to wait 20 seconds for response to generate. So this is like probably the most effective, but I think it's one that's also probably the most commonly used caching as well. Caching is so easy now, especially with the model providers often just doing it automatically for you, whether that's OpenAI or Anthropic, they'll just automatically handle caching on their end.

00:47:43 Of course, you can layer in some more complex semantic caching if you want to get fancy and that can be a very effective tool as well. Probably one of the most underutilized one is ensemble kind of methods and I think I'm starting to see this get more and more popular, especially after Andres Caparthi a few months ago kind of released his council of LLMs. So it's exciting to kind of see that kind of pick up more interest across the industry. I think people are more interested in that now, which

makes me really happy to see. Probably the hardest one to get right is routing. On the surface, it seems so obvious like hard problems should go to a smart model and easy problems should go to a dumb model that's really cheap. But then it's like, how do you classify a problem as easier or hard?

00:48:33 And you see people complain about this where ChatGPT and Anthropic added their adaptive thinking, right? And this is a huge complaining point for a lot of people where they're like, "I don't like this at all. " They're like, having it instantly respond bothered some people, having it think bothered some people, but having it try to guess which to do bothers everyone. And so this is like a really tough one to figure out. And so on that one, I'm like, "I'm going to sit back and let the labs kind of figure that one out for a bit while the rest of us watch." Now that being said, you may have in your case, in your company and your problem set, a more deterministic way of determining how to route problems. And in that case, I'm like, "Go for it. If it's really straightforward, use that routing system.

00:49:18 It's so effective to just figure out which type of model should solve which problem." But if you're trying to figure it out on the fly, that's where you should not use that method.

Jacob Miller: 00:49:30 There's this aside that we put in the book around designing the user experience. Of course, our book is not about optimizing designing user experiences, it's primarily on architecting, but that being said, it is amazing how some designers can get all of the information they need for the AI, go kick off the AI, shepherd you through something else really quickly and then it appears instantaneously and they don't realize that the AI responded to them while something else was going on. And so rerouting these user experiences and basically you're running all of your AI jobs behind the scenes and then designing the user experience flow in

such a way that they don't notice that we hid the latency. And there are products, UX people who are genius at this right now where you see it throughout these products. So in the products that we use, we notice when someone does this and it's like, "Oh, it appeared instantaneously." And we realized that it had kicked off two steps before that.

Jeremy Mumford: 00:50:36 And this is so fun because it's almost like we're back in the early 2000s where you hear the story of Instagram where they talked about how when a user was uploading a picture that they wanted to share on Instagram, the moment they added it, they start uploading it in the background to the server while the user types out their caption. And when they hit post, it's just instantly there. And then as like technology's gotten faster, the internet's gotten faster. I feel like us as engineers, maybe we've kind of just gotten accustomed to everything just works fast so I don't need to do all these little hacks, but now we've rearround the clock, we've got to deal with these slow, slow LLMs and it's time to relearn those techniques and lean back into them.

Jon Krohn: 00:51:17 For sure. Really well said guys. Great example around latency with respect to our models. This is going to be my final technical question for you and it's around latency of humans in developing AI solutions. So in chapter 11, which is the penultimate chapter of your book, you take on a question every AI leader is wrestling with right now, which is if everyone has access to the same frontier models, where does durable competitive advantage actually come from? And your answer is that it's not the model, it's not even the data exactly necessarily, it's velocity. How fast your organization can run the loop of build, measure, learn, improve, and to capture this concept, you invoke the gingerbreadman. So run, run as fast as you can to create a moat. So velocity itself becomes your moat. And I think this ties together a lot of ideas from over this episode.

00:52:19 We have been talking about this to some extent, but I feel like it now deserves its own focus. I think a lot of our listeners are wondering where this moat can come from and yeah, if it's not models, if it's not necessarily data, then this idea of velocity itself, I think it's a great solution. How fast are you guys moving right now?

Jeremy Mumford: 00:52:41 We're trying to move as fast as we can. If you look at how the labs are operating right now, Anthropic and OpenAI and others, they're all in on this concept, right? The idea of self-recursive improvement and just like trying to close cycles as fast as possible. This kind of goes back to the critical idea of the singularity, but the good news is like the rest of us companies can take advantage of this concept as well. This is not just limited to people building AI models. This also works for people utilizing them and trying to accelerate their company as quickly as possible. And so everything you can do as a company to complete cycles faster helps you leap ahead of the competition, whether that's shortening the time to build products, accelerating the process of going back to what Jacob said about define build and feedback, anywhere you can accelerate each of those three steps is helpful.

00:53:37 I think at this point we've accelerated the build step a lot. There's still more room to be made there, but I think accelerating the defined step and the feedback step are going to be enormous for companies to continue to increase the speed of this cycle. And if you're just making that 0.1% improvement every time you finish a cycle, then over time that compounds.

Jacob Miller: 00:53:59 So on one side, it's the velocity of how fast you're going through these. And on the other end of the spectrum is how much do you improve with each cycle? And I think this relates to the concept of taste that people hear about or value and are understanding. And so you take a certain subject matter expertise, a taste and understanding of what's truly valuable for your end users

and say, "Hey, if this is what we build, what would we learn and how much would the product improve in being able to pick and choose those?" Because if building is not the bottleneck, then velocity and taste, which is the quality improvement on each one of those cycles is the determiner.

Jon Krohn: 00:54:41 Yeah. For now at least we have a little bit of an edge on taste. We'll see how long that lasts. There was a really popular article, something big is happening. I don't know if you guys have seen that blog post where he talks about how I think that article came out around the Opus 4.6 release and his use of Claude Code where he said, "I felt like I had this edge on taste and I had to get the algorithm to go in the right direction there, but now at least for some use cases, maybe relatively simple, but probably

00:55:15 More and more we're going to find that the post-training reinforcement learning that these models have are allowing them to have our taste." Anyway, I don't mean to be saying you're wrong, but it is interesting. There's an edge that we can have at least for the foreseeable future around the right context for that taste. I think that's something key here where even if we're trying to have Claude have all the right context, we're taking our meeting, recording transcripts and putting those in. There's still going to be some things that are probably missing that we as a product owner or a system architect, we have some extra context that's critical that the algorithm doesn't. Cool. All right. So that is my journey through your book Architected Intelligence available from wherever you buy your books now, published by Wiley, a great publisher. Congrats both of you. For both of you, it's your first book, right?

Jacob Miller: 00:56:14 Outside of textbooks, yeah. Right,

Jon Krohn: 00:56:16 Right, right.

Jeremy Mumford: 00:56:17 And I understand you're going to start working on a new book soon as well, right?

Jon Krohn: 00:56:21 I have started working on it and my publisher is not happy with my pace, but yes, I think I announced this. Did you guys hear this on the podcast or?

Jacob Miller: 00:56:35 Yeah.

Jeremy Mumford: 00:56:36 On the most recent episode, we heard that

Jon Krohn: 00:56:38 You were working on a book

Jeremy Mumford: 00:56:39 With someone named Edward Donner,

Jon Krohn: 00:56:41 Right? Ed Donner, exactly. Many of our listeners will be familiar with him. He actually had an episode on this podcast some 18 months ago we are going to have another podcast episode. I don't know if you guys have this with your book release, but with ... So Pearson is my publisher and Pearson publishes into the O'Reilly platform and something cool that you can do with that is we're going to be releasing rough cuts of each chapter as they're available. I'm hoping that not too long after this episode comes out. I'm going to have the first chapter done and available in the O'Reilly platform for people who have a subscription to that. And then yeah, we're going to have probably about a dozen chapters like you guys had. So it's going to take us a while to get the whole book out. But yeah, Engineering AI Agents is the book and

00:57:38 Yeah, looking forward to making that. I think Ed is tremendous. He has amassed. He started creating video courses on Udemy about 18 months ago and he's sold 500,000 copies of courses that he's made and yeah, people can figure out how much that's worth because they sell for at least 10 bucks each. It's like, that's a non-trivial amount of revenue in an 18 month period. And he's done that by being so engaged with his audience and yeah, I'm basically just, I'm kind of taking his ideas,

		his video content and working with him to repurpose it into a coherent textbook. And given how valuable his courses have been, we're hoping this book will be a hit too.
Jacob Miller:	00:58:30	Nice. We look forward to it.
Jon Krohn:	00:58:31	Yes. Yeah. So all right. Well, thanks for listening to the podcast, guys. That's nice.
Jacob Miller:	00:58:36	We love it.
Jon Krohn:	00:58:37	Before I let you go, since you are podcast listeners, you must know that I always ask for a book recommendation. Do you guys have one for me today? And you can have one or even more each. I'm not a very strict interviewer
Jeremy Mumford:	00:58:52	Deal. I think for me, I'm going to cheat and I'm going to recommend two books, Observability Engineering by Charity Majors, fantastic book was a huge inspiration for us as we were writing our book and kind of thinking about how can we write a book that is focused on higher level concepts. So that was super inspiring. And then LLM Engineers Handbook, I believe it's by Paul Eutzen and Maxine LeBon, very more practical kind of code-based guide for those trying to get into AI engineering. So highly recommend that.
Jacob Miller:	00:59:33	Yeah. Besides Charity Bager's book, another book that was an inspiration for us as we were writing, centering in on how can we communicate something that's changing very quickly, but communicating principles that are durable and are still relevant five years from now. Prince Fundamentals of Data Engineering from Joe Reese, he's someone that we know. I remember going through his book for the first time a few years ago and that was an inspiration for For me, because there were so many concepts that were enduring around the principles of data engineering. As we were tackling this book, we thought,

how could we approach AI engineering and architected intelligence in this similar way?

Jon Krohn: 01:00:14 I love that. And you guys have definitely accomplished it. I love how this book is going to last for years and years. So even if you're in the 2030s or beyond listening to this podcast episode or given our efforts to be a user agnostic podcast, maybe you're an agent listening in in the future.

Jacob Miller: 01:00:35 Exactly.

Jon Krohn: 01:00:37 If architected intelligence is not already in your canon, then put that in your canon agent.

Jacob Miller: 01:00:44 Exactly.

Jon Krohn: 01:00:47 Nice. Fantastic. Thank you so much. This has been a great episode. Every minute had an invaluable nugget for us to take home with us after the episode. Jacob, Jeremy, for folks who want to glean more invaluable nuggets from you other than obviously purchasing your book, Architected Intelligence, how can people follow you online after this episode?

Jeremy Mumford: 01:01:08 We've got a website called thewybehindai.com. So you can go check that out. And similar to how you're going to be publishing your content through O'Reilly, that's where we publish thoughts from the book in kind of a rougher format or other thoughts that we have around kind of the why behind AI. It's a very kind of fun place for us to share our ideas. We tend to be a little more casual in that environment where there's more memes. So not as many memes in the book, sadly. There's still some fun nuggets in there, but if you come to our blog, you're going to have a lot more fun as far as memes go.

Jacob Miller: 01:01:45 We make it a lot more entertaining. And we're both on LinkedIn and that's where we post updates in terms of things that we're doing and tackling. But yeah, Wiley of course doesn't like as many memes as we would have

liked in the book. And so those are on the blog. It is a more fun way of ingesting the material.

Jon Krohn: 01:02:06 You do have a lot of stick figure cartoons though that still add that lighthearted element to it. So that's really cool. All right, Jeremy, Jacob, thank you so much for taking the time to be on the show. And also this episode would not be complete without me thanking Jon Baum. How do you pronounce his last name? Jonathan Baum.

Jeremy Mumford: 01:02:23 Yeah, Boun. Yeah, Jonathan Baum.

Jon Krohn: 01:02:24 Okay. Nice, nice, nice. Yeah. Thanks so much, Jon , for recommending Jeremy and Jacob as guests. Jon has been a listener to this show for years and years and one of our most active social media commenters. And yeah, maybe at some point we're going to have to have Jon on here for an episode as well.

Jeremy Mumford: 01:02:43 You should. He's a great guy.

Jon Krohn: 01:02:45 Well, yeah. And I know that at the time of us recording this, he is in Boston at ODSE East giving a big two hour long tutorial. So there's probably some content in there he could be sharing with our audience. So yeah, thanks Jon . Yeah. All of these Jays from Jon to Jacob and Jeremy. Thank you other Jon for this jovial episode. Yeah, this was really good. Thanks listeners for bearing with me and all of my Jays. And yeah, thanks a lot, Jacob and Jeremy for being on the show.

Jacob Miller: 01:03:21 Thank you.

Jeremy Mumford: 01:03:22 Thank

Jacob Miller: 01:03:22 You.

Jon Krohn: 01:03:24 Man, oh man, that was a great episode. Really appreciate the well thought through insights of Jacob Miller and Jeremy Mumford. Don't forget to check out their new

book, Architected Intelligence. In this episode, we basically only talked about stuff from their book, including the User Agnosticism Tenet, which means designing products and processes so they can be executed equally well by a human, an AI agent, or any hybrid combo. We talked about the shift in the define build feedback loop now where building is no longer the bottleneck like it always was, which means definition and feedback are where teams win or lose. We talked about why workflows are deterministic, predictable, and cheaper than agents and why the natural progression is skills first, then workflows, and only then agents. We talked about why data engineering is the true bedrock of AI engineering illustrated through the anchor man analogy.

01:04:19 Whether you put in front, whatever you put in front of your model, the model will read as truth. And we talked about why velocity is the only durable moat in a world where everyone has access to the same frontier models. All right, that's it. As always, you can get all the show notes, including the transcript for this episode, the video recording, any materials mentioned on the show, the URLs for Jacob and Jeremy's social media profiles as well as my own at superdatascience.com/993. Thanks to everyone on the SuperDataScience Podcast team, our podcast manager, Sonja Brajovic, media editor, Mario Pombo, partnerships team Natalie Ziajski, our researcher, Serg Masís writer, and our founder Kirill Eremenko. Thanks to all of those fine folks for making another sensational episode for us today for enabling that super team to create this free podcast for you. We are deeply grateful to our sponsors.

01:05:15 You can support the show by checking out our sponsor's links, which are in the show notes. And if you'd ever like to sponsor an episode yourself, you can get the details on how by making your way to Jon Krohn.com/podcast. Otherwise, there's tons of other ways you can support the

show and we really do appreciate if you do. It is the only way we can grow. So please do share this with anyone who would like to learn about how they can be building AI first companies themselves. Review this episode on your favorite podcasting platform or on YouTube. Subscribe if you're not already a subscriber. But the most important thing above all else is that you just keep on tuning in. I'm so grateful to have you listening and I hope I can continue to make episodes you love for years and years to come. Until next time, keep on rocking it out there and I'm looking forward to enjoying another round of the Super Data Science Podcast with you very soon.