0:13

Alexey

**This week, we'll talk about the unwritten rules for success in machine learning, and many other things. We have a special guest today, Jack. Jack transitioned from software engineering to data science, and he worked as both an individual contributor and in leadership roles. He managed, at some point, teams of up to 15 people – currently, he is the VP of data science and machine learning. And soon, (there's even a date, November 15) he plans to move to become an entrepreneur. Welcome to our podcast!**

0:53

Jack

Yeah. Thanks, Alexey. Thanks for having me. Great to be here.

0:56

Alexey

**The questions for today's interview were, as always, prepared by Johanna Bayer – thanks, Johanna, for your help. Let's start.**

# Jack’s background

1:04

Alexey

**Before we go into our main topic of these unwritten rules, let's start with your background. Can you tell us about your career journeys so far?**

1:12

Jack

Yeah. I started out my professional career in 2015 as a full stack software developer. I was working in a company called Trunk Club. They're sort of like Stitch Fix – they're owned by Nordstrom (online retail). I started out as a software developer, worked there for a couple of years in that role, but then became interested in data science and machine learning, and made a transition within the company to data science.

1:42

Alexey

**Do you remember what made you interested in data science?**

1:46

Jack

Yeah! Actually, I had sort of always been interested in data **science** – going back to my undergrad days, when I was a physics major and took a lot of stats classes. But this was 10 years ago, so at the time, data science was not well-established – there was no Master’s degree, there was no Bachelor's degree – there was no clear career path.

So I decided to do software engineering (a Master’s in software engineering) intending to go into data science, but ended up just really enjoying software engineering, so I stuck with it for a couple of years. And then, at some point, though, I started to become interested in machine learning again, and a data science team was spun off at my company. I watched what they were doing and became very interested in that, so I made the transition into that team through an apprenticeship.

2:38

Alexey

**So there was an internal opportunity.**

2:41

Jack

Yeah – it was not easy. I had to be very assertive and ask a lot of times, and then, eventually, landed an informal apprenticeship opportunity within that, where I was doing a side project. But I was able to switch, and did data science at Trunk Club for a year. Then I moved on to a company called GoHealth, which is sort of like Orbitz, or Kayak, but for health insurance. I was a Senior Machine Learning Engineer… [cross-talk]

3:10

Alexey

**Orbitz or Kayak? Orbitz, I think… wait, wait, wait – I was in the States at some point, and I needed to buy an airplane ticket, and this is what you use for comparing prices, right?**

3:20

Jack

Exactly.

3:21

Alexey

**Yeah. Okay. It's like Skyscanner.**

3:24

Jack

Yes. So think about that, but for health insurance – at least in the United States, most of us get health insurance through our employer and you have like three plans to choose from. But for people who are not employed, or people who are self-employed – to choose a health insurance plan, it's complicated. So this platform was basically the selection process for that. In any case, I was a senior machine learning engineer there – started out in that role. The data team was also new at GoHealth. Quickly, I found myself informally managing that team because there was a data science team within GoHealth and the data organization itself was maybe 15-20 people, all being managed by one person.

Naturally, people stepped up as informal managers and that was me for the data scientists and the data analysts around me. I was formally promoted to manager after a year or two and then, again, I was promoted to director after another year or two after that, most of that was around having successful projects launched. After a big project launch – that would generate a bunch of excitement and demonstrate value to the company – leadership would want to invest in more data science and machine learning. And, naturally, that would come to me. [cross-talk]

# Transitioning from IC to management

4:47

Alexey

**Just curious, in retrospect, do you think it was a bit too fast, too quick? You worked for a few years as an individual contributor and then – boom – you became a manager. Now, when you look back, was it too quick or was it just the right pace?**

5:11

Jack

It's a good question. I'll give you two short answers to that. The first one, I'll say is that because I did a lateral move between software engineering and data science and machine learning, there's so much overlap between those skill sets and professional maturity that I would say…

5:28

Alexey

**I assume it helped a lot, right?**

5:30

Jack

It did, yeah. It wasn't like I had just one year of experience before management – it was more like I had four, before being promoted because, again, there's just so much overlap between the two. I would say that. The other part, though, I would say is that being promoted to manager, it almost always feels too soon – and it will always feel too soon, because there just is a shift that is very difficult to emulate a seamless transition. It's a new paradigm, it's a new thing to experience, and it is always going to be a difficult transition. That being said – Sure, it was fairly fast, both good and bad results came from that.

6:09

Alexey

**But you also learned a lot, I assume Right?**

6:11

Jack

Oh, yeah.

6:12

Alexey

**When you move fast, you also learn fast – because you have no choice.**

6:17

Jack

Exactly. When I think about the pivotal years of my professional career, there were two time periods. The first was just my first couple of years at Trunk Club, being part of a strong technology organization, developing a lot of really good fundamentals, and you sort of get into your groove. Then the next phase was going from a senior IC role to director within a couple of years. That was just very, very different from the other part. But I learned a lot. That's where you learn a lot of things that are not really taught anywhere. It's very difficult to teach them because everything has a bit of nuance…

# Lesson not taught in traditional school

6:53

Alexey

**There’s no school for VPs of data science, right? [Jack agrees] So you just have to… How do you actually learn these skills?**

7:00

Jack

I mean, for me, it was trial and error. It was realizing what works and what didn't work. [cross-talk]

7:07

Alexey

**How do you even know what worked?**

7:09

Jack

Yeah. A lot of that is what I'll share, in general – just what works, what doesn't work, why it works. In fact, that's a lot of the reasons why I started to be more proactive on social media. But yeah, that was a great learning period for me. You learn a lot of lessons around being able to convince people of value, knowing when and how to articulate accuracy versus something that may be less precise, but more compelling – things like that.

A lot of being a leader is, in some sense, being a salesperson to the rest of the business, where you have to be able to demonstrate value, you have to sell value, you have to translate, “Why is this needed?” Trying to convince a nontechnical stakeholder why you have to spend a month on cleaning up tech debt. You need some sales skills to do that. So there's just a lot of things that are very difficult to teach in any kind of technical curriculum, because they're so opposite of what needs to be emphasized early on in that career.

8:16

Alexey

**Do you have an engineering background? Engineering education – software engineering?**

8:23

Jack

Yeah. My Master's degree was in software engineering and then my first couple of years were software engineer.

8:29

Alexey

**So I assume it's not a skill you picked up during studies, right? [Jack agrees] During software engineering classes, you are not taught how to sell things. You learn how to do Java and algorithms and databases and all that stuff – you're not learning how to sell things. So it was all trial and error, right? You just see how you approach a person and you try to kind of sell a project or something, and then you feel what works and what doesn't. Right?**

9:01

Jack

Yeah, I would say it's a mix of trial and error and it's also a mix of observing what works for others. Another thing that is not really taught very often in any kind of school environment is the importance of technical problem framing and understanding the business and user side of the applications you’re building. One story I like to tell is, when I was a young software developer (a couple years into my career) I felt like I was really strong technically. Then I would look around and look at the people on my team and think, “Okay, who's better at what? How do I make myself better?” And then I would go to my tech lead and see, “Okay, what makes them special?” And something I observed was that my tech lead, at the time, was strong technically – definitely stronger than I was, because they were more experienced. But relative to the people around him, he was not necessarily that much stronger, but there was a clear difference in how much he could produce and the value that he provided.

The difference was that he was really, really good at taking the time to deeply understand the applications that we were building and taking one-two hour meetings with stakeholders to ask them questions and really just understand the aspects of the product as well as anyone else in the business. Because once he understood that, he could then come back to the tech team and be able to transmit, “Okay, here's what we need to do. Here's what we need to prioritize. Here's why this is the way that it is.” That is something that's very difficult to teach.

It's difficult to teach, because when you're learning software development, you're so overwhelmed by everything else that hearing someone try to explain that is like, “Well, that's obvious. I can focus on that later.” It's very difficult to sort of mentally allocate the right capacity to something like that, when you're getting crushed by all of this very difficult technical stuff. So that's sort of something that, even when I post on LinkedIn – if I post something about that, the amount of interest that I'll get relative to posting something very technical, is so much lower. Because most people don't feel like they're struggling with that right now – they feel like they're struggling with the technical parts, so that is the more valuable piece to them. These other parts that really do help differentiate people in their careers often go unnoticed and are not learned until trial and error several years later.

# The importance of people’s perception, trust, and respect

11:33

Alexey

**One of these things that go unnoticed, as you mentioned, is understanding the applications on a deep level, and understanding what generates value. That's one of the things, as you mentioned, this tech lead had that other less senior people did not. Are there other hidden things that are not obvious when you focus on technical things that are actually very important at work?**

12:10

Jack

Yeah. One thing that's extremely important is people's perception of you – another thing that is never taught in any courses I've ever seen. In order to be successful, especially in leadership, but really, in any role – people have to respect your opinion. When you ask for things, they give them to you. That is just an essential part of being a high-impact either IC or manager. And in order to do that well, people have to respect your opinion. Developing a reputation for respect and strong consideration is something that is difficult and is something that I just learned over time how to do.

One simple tip that I give to people is – anytime you're given an opportunity to do a presentation in front of an audience, spend way, way more time on that than you think you should. Anytime you have an opportunity to speak to people that you don't normally speak to, they will base their perspective of you very heavily on that discussion or that presentation because that's really their only perspective into you – that's their only interaction with you. So, if you're giving a presentation to leadership, or even just other teams – if those teams aren't working with you directly frequently, they don't know how good you are – they don't know any of that. You really have to make sure that any time you're given an opportunity to interact with others who you don't interact with on a day-to-day basis, you have to take advantage of that.

That's where the sales stuff comes into play. Maybe saying things as they are is not always the best thing. You want to sell them, you want to promote them. Really, what you should be thinking about is, “What is the ultimate action that I want to be taken?” If you're thinking about the month of technical debt cleanup – if you try and explain the exact reason why that's so important, you probably won't get what you need. So you need to be able to communicate in another way and you need to be able to put on a persona that is well-respected so that you get what you want. That's another aspect that is just more of a “softer skill,” but it's critical to success. Some people can get to the top through just technical skills, but they're very few and far between. Most people who get to the top are doing so through nontechnical means.

14:45

Alexey

**So look for opportunities to speak in front of people who you don't interact with regularly and when you get this opportunity, invest a lot of time into the presentation that you give. Right? [Jack agrees] And you should probably avoid technical things. Like in your example of technical debt, you shouldn't say, “We need to refactor a lot of code because our classes are too long.” Right? You should probably come up with a good metaphor. Instead of explaining the code, you should probably find a relatable idea from the real world, and then explain based on that.**

15:25

Jack

Yeah, and even extending that –something that I like to do is to think through what people care about. If you're working professionally and – let's say, you're interfacing with marketing. Marketing will obsess over things cost per acquisition – all they care about is conversion and CAC, and things like that. So if you're trying to present to them something related to machine learning, data science, or whatever it is – if you can talk to them in terms of impact to CAC or something that they will care about, you'll get a lot more attention.

16:02

Alexey

**Impact to what, I’m sorry?**

16:04

Jack

CAC – cost per acquisition.

16:08

Alexey

**Cost per acquisition.**

16:10

Jack

It's the amount of money that marketing needs to spend to acquire a customer, on average. It's like their primary KPI or key metric.

16:20

Alexey

**So, when you speak with marketing people, you need to learn their vocabulary. You need to use words like cost per acquisition and so on, so that they can already relate to what you're talking about. Right?**

16:44

Jack

Right. Once you start using that terminology or vocabulary, now they’ll feel like you understand them, and will start to have more respect for you. Because what often happens is, technology people speak technology, marketing people speak marketing, and both sides feel like the other doesn't understand what they think is important, and therefore does not understand anything important. So you kind of talk past each other a lot. But if you're able to speak their language, you're able to get their respect, and now they'll actually start to listen to you more seriously, because they feel like you know what matters to them.

# How soft skills are relevant to machine learning

17:22

Alexey

**Okay. We'll see how relevant it is – I hope it is relevant to the actual discussion we plan to have today, which is about the rules for success in machine learning. Probably it is related. I just want to summarize. If you want to be a technical leader, there are some qualities that are good to have, like understanding the application, and then there’s the important thing of people's perception of you – you need to gain respect from them. For that, you need to appear in front of other people and speak their language. You use their terminology and then they treat you as one of their own. This way, you get respect. These things that we discussed – they're pretty interesting to discuss. How related are they to successful machine learning projects?**

18:16

Jack

Very, very well-intertwined, especially with machine learning. In any kind of project, but specifically machine learning – one of the challenges with machine learning is just how complex it is, and how much support you need from every part of the business for machine learning to work. Any software developer will tell you like getting support for a new project is difficult and that's absolutely true. But the level of support that you need to execute a machine learning project that actually works and provides value is – I'm going to pull out a number here, but – maybe X3 of what a typical software project does. And that's because there's so much level of care that needs to be involved with generating data, transforming it, setting up that pipeline, building the model, building the prediction pipeline – there are just so many things that can go wrong, and so much effort that needs to be taken, that, in order to allocate the resources for that, you have to have a lot of motivation and a lot of buy-in from stakeholders.

Being able to sell is a very crucial skill of pretty much any machine learning person, because in order to gain enough support for your projects, you have to be able to get people excited about the potential of what you're going to build. That's something that I learned maybe a year or two into my career in machine learning – a lot of the projects I wanted to build were dead in the water because I couldn't get enough support. So what I started to do was figure out the ways in order to generate excitement and get that support that I needed. One example is being able to get a proof of concept that I can use in a demo very quickly, to show on a real sample dataset (like a prediction in real time). That is very effective at getting interest between people because they can see the potential value of it.

But going back to your question of how these skills intertwine with successful machine learning or the rules of machine learning. You have to understand that you don't just get handed machine learning projects and your only task is to execute on it. That does not happen. A huge part of your role in machine learning is to be able to communicate back value to buy yourself the bandwidth and the resources that you're going to need in the future. That's a very often overlooked aspect.

# How to put on a salesman hat in machine learning management

20:48

Alexey

**So you mentioned that machine learning projects are quite complex, and they acquire 3X (arbitrary number) – they require a lot more effort than a traditional (usual) software engineering project. This is not to mention that these projects often fail because you don't know in advance whether the project will succeed after you put so much effort into it. How do you sell that? By doing this POC and showing that it kind of sort of works?**

21:17

Jack

So there are a couple of things going for you with machine learning that you can leverage to your benefit. Right now, this is actually the best time ever for this because large language models have generated excitement across the entire planet. At this moment, it is much easier to sell people on machine learning than it was four or five years ago.

21:35

Alexey

**Because there’s this hype everywhere – open any website and you see GPT, LLM, whatever.**

21:42

Jack

It's a double-edged sword, where there are benefits and negatives to that. However, even so, it is still difficult to sell projects, because, like you said, there is so much risk involved. The connection that people make with machine learning is that they want to automate their decision-making, or what they think is possible with decision-making. So a lot of people… you can play off of that.

For example, if you wanted to build a model that was prioritizing incoming inbound leads for your sales process – there are people at your company, who spend their days trying to think through, “How do we ensure that we're properly handling the highest quality leads, which are people who are actually going to buy and spend a lot of money? How do we ensure that they're treated well?” Well, if you can build and showcase a simple model that says, “We can detect which leads are going to be high value.” It is very, very easy for people to realize the potential impact or use for that. And if you can give them even just a little bit of evidence that you're capable of building that, then they'll buy into that.

So really, you want to play to what they care about, and then give them some evidence of that and showcase it – usually, visuals are really good selling points here. And if you can give them that, you're much more likely to get that buy-in. If you have stakeholder support, you can typically get the engineering resources – all that stuff.

# The importance of visuals and building a POC as fast as possible

23:18

Alexey

**You said visuals are important to them – what do you mean by that? Is having a demo with a user interface where they can play around important, or did you mean something else? Or did you mean planting a picture in their head or something else?**

23:36

Jack

Yeah, that's a good question. Let me make another comparison using the domain from a different company I worked for. I worked for Wayfair a couple of years ago, and… I’m using this context, just because it's easy to understand furniture.

23:52

Alexey

**They sell furniture, right?**

23:54

Jack

Yeah. The model that I'm going to describe, I didn't build, but here's a hypothetical model and here's two ways that you could pitch it. One would be effective and one would not be. So let's say you wanted to build a model that could detect someone's preferences for styles of furniture. Let's say that I like rustic furniture (farmhouse or something like that) whereas maybe you like modern furniture or something. Let's say that I built a proof of concept model and wanted to pitch this.

I could show people the accuracy of items bought that were related to the style of furniture that I said they liked. So somebody purchases a couch – with 70% probability I can predict which type of style it is based on my model. That is not a good way to sell your model. A better way to sell your model is to go and show three examples of stylistic preferences that a customer has (that they’ve purchased in the past) and then show them the next three items that they might purchase in the future because they have similar styles. This is a good way to visualize.

25:10

Alexey

**So you pick a random customer and say, “Okay, this is person X and this is what they like – this kind of furniture. These are their preferences. Based on that, we think that the next three orders will be these things.”**

25:26

Jack

Right. What that does is give your audience (your stakeholders) – it shows them loosely how the model is thinking about things. Maybe two different coaches are the same material or something – they can start to mentally wrap their head around, “Oh, here's how the model is able to think.” And then they can take that, and they can generalize that to all these other things that they care about, where it would be useful to have an automated decision-making process that could detect that.

So really, you're trying to get them to understand the model and what it can do and then let them take that notion and [audio cut off] using their own internal knowledge. Communicating things in accuracy… At the end of the day, those are the things that matter, but they're not great for selling, because that's not how your stakeholders are thinking about things.

26:15

Alexey

**If you start talking about accuracy – you say, “Okay, this model is 70% accurate,” which may or may not be a good number, depending on the model, but to the stakeholders, it might sound scary like, “Ooh, 30% error rate. 30% of the time, it will make mistakes. Oh, that’s bad.” How do you handle that? Do you talk about that? [chuckles]**

26:45

Jack

It's a good question. Actually, of the two problems, that is the easier problem – trying to communicate error rates and things like that. Typically, what I like to actually do is not communicate accuracy, because it is the easiest one to pick apart and say, “This is not good enough.” What I try and do is discuss precision and recall, because when you discuss… [cross-talk]

27:07

Alexey

**They're quite technical, right? You need to give very good illustrations to people. Because even as a data scientist, I remember studying machine learning at university and I was always lost at precision recall. For me, it was super confusing – I always confuse the two. It took me some time to actually feel comfortable with these two metrics.**

27:29

Jack

Yeah. I don't actually use the words “precision” or “recall”. But what I am discussing is precision recall and what I would do is show, for example, a sliding bar of and say, “How much of a threshold do you want to draw between over-guessing or under-guessing?” Typically, I'll use precision recall, and then discuss… The goal is not necessarily to give them exact numbers, but it's to inform them that we can control the trade-off that we're making and they're more comfortable with that, because it means “Oh, well – if we're over-predicting, we can just take that slider back.” They remember the slider, they remember that there is a threshold, and it gives them more comfort to know that they're in control.

28:17

Alexey

**Do you use something like Streamlit or Gradio for that? Or you use your full stack development skills and build a real one?**

28:23

Jack

[chuckles] No, whatever I can generate in the shortest amount of time that looks decent, is what I do. [chuckles]

28:29

Alexey

**Which is usually Gradio or something like that, right?**

28:31

Jack

Yeah, sometimes. It depends on the organization. If I have more time, I'll do something like that. Honestly, sometimes I'll just go into Google Sheets and show something that's very, very, very basic.

# 1st Rule of Machine Learning – don’t be afraid to start without machine learning

28:46

Alexey

**I see, I see. Interesting. Well, we still wanted to talk about the rules of machine learning and the unwritten ones. There is this famous article from Google, which is called the Rules of Machine Learning. You probably know about that. I remember that rule number one is, “Don't be afraid to start without machine learning.” That's my favorite one. It's kind of funny, because if you're a data scientist or machine learning engineer – how can you advocate for not using machine learning? It's a bit counterintuitive. Then they talk about metrics and so on. So this is definitely a really good article to talk about. But since, today, we’re talking about the unwritten rules, I'm wondering what these rules are.**

29:33

Jack

Yeah, awesome question. There's a lot of them. Actually, let me touch on what you just mentioned really quickly. Because I think, regardless of if it's written or not written, it’s… Maybe let me expand on that, because the importance of that, which is often not written. I completely agree with that statement – it is critical… And I would even go beyond saying, “Don't be afraid to start,” and I would say, “Always try to start without machine learning.” And the reason for that is because, the most frequent reason that I see machine learning projects fail is because whatever you end up building doesn't actually solve any problems.

So even if you build an accurate machine learning model, that does not necessarily mean that whatever it's doing is going to provide value to the business. So before you build machine learning [models], ensuring that whatever problem you're solving is actually worth solving, with or without machine learning – that should always be the first focus. I always emphasize, “You should do a proof of concept, heuristic, rule-based model first. Forget machine learning, forget all the complexities with that. Try to just spin up a manual process or a rule based process that emulates what a machine learning model is going to do, but does so at a much more basic level.” Honestly, nine times out of ten, if you can't make that work, the machine learning model is also not going to work. [cross-talk]

31:03

Alexey

**Yeah, I have a story about that. A few years ago, we wanted to launch a model for predicting the quality of pictures. If a picture has a good quality… It was for an online classifieds site, where people would sell and buy cars. One idea was, to make listings more attractive, we can look at listings with bad quality pictures. And if there are bad quality pictures, we contact the sellers and ask them to improve the picture, which is probably a reasonable assumption – if they improve [the picture quality], then more people will probably become interested in that.**

**Then we started building the model – a deep learning one, of course, because it’s images [chuckles]. A few months later, we were finally ready to test it. It turned out that nobody really cared – the thing that people mostly care about when looking at listings is price, not the quality of images. Plus the sellers weren’t really super reactive to these suggestions.**

**So what could have happened instead, I could have just sat down, taken a sample of images, and said, “This image is good, this image is bad.” We then send an email to the user saying, “Your image is bad. Do something about this.” And see how many people react. One day of work. And then we’d see, “Okay, nobody cares about that. Let's just put this idea aside for some time and focus on something else.” I love this story. I mean, it's something I experienced myself. One thing is somebody when somebody says, you just nod like, “Yeah.” But having experienced this firsthand, it’s a different thing.**

32:53

Jack

Exactly. That is exactly like the type of situation that occurs. Again, it is also evident in software engineering, where a lot of people ask for things – 10 years ago or whatever, the common thing [to ask] was, “Can you build me an app that does this?” That was the question everyone wanted answered, “I have an idea for an app.” Or, after, it’s, “I need a feature that does this.” Eventually, you realize that you have to start vetting who was asking that question and why they're asking it. Because oftentimes, people think they want something and then the reality is that they don't even use it.

That's basically the same problem as that, but much more intense, because machine learning is so much more difficult and expensive – both in terms of actual monetary costs and development time – that making a mistake here is much more costly than in just a typical software development project, usually. So it's definitely very critical. I just wanted to highlight that point, because when you think about the unwritten rules of machine learning, you really have to understand the human nature of getting lost in the tactical details of machine learning projects that all of us [audio cuts out]

34:09

Alexey

**Yeah, and then you establish a baseline. Well, you first prove if this idea is valuable at all, then you can establish a baseline and then you can iterate on top of that and you can see if this is actually an improvement over the previous iteration.**

34:26

Jack

Exactly. Another concrete example with machine learning was – at one point in my career, I built a customer churn model. In a subscription-based company, churn is when they cancel their subscription. For the company in question (I won't mention which one it is, just to keep things somewhat private) we tried to build a churn prediction model, and I did, and it was actually pretty accurate. It could detect who was going to churn. The problem with that was there was nothing we could do about it.

We thought that there was going to be something that you could do when we were able to identify someone who was about to churn, but in this case, there was really no action to take that was likely going to make them not churn. So because of that the whole model was useless. It didn't provide any value. And the reason for that, is because it wasn't actionable. Had we gone through a more heuristic-based approach, we would have saved ourselves several months of development time.

35:27

Alexey

**Was it the case because when you try to approach people and prevent them from churning, they're annoyed even more?**

35:39

Jack

I would guess that that is often the case. In this case, it wasn't. It was more so because… The reason that they were churning was not because they were unhappy, but because of external factors that were driving them to churn. [chuckles] Again, I don't want to get too into it… [cross-talk]

35:54

Alexey

**Then you would give away the company, right? [Jack agrees] But in a hypothetical situation, let's say there is a telecom provider and people are churning, because there’s a different provider with better prices, and you cannot lower your prices.**

36:08

Jack

Exactly. Exactly. That's a very good reason

36:12

Alexey

**If you lower [the prices] then you’re kind of working… You don’t generate enough revenue.**

36:19

Jack

Right. Your margin will be off, yeah.

36:20

Alexey

**Yeah. Okay. Well, so the first unwritten rule… I’m wondering – in this podcast, we have transcripts. So, after this is transcribed, this rule becomes written?**

36:37

Jack

[chuckles] Yeah. Well, most of this stuff I'm already posting on LinkedIn anyway. So I guess at this point, it is written.

36:44

Alexey

**Ah, it’s too late. So it’s “commonly unwritten rules” that Jack has written about. Right? [both chuckle] So the first rule is – we'll get caught up in technical details and we… What's the expression? We don't see the forest behind the tree?**

37:04

Jack

Yeah, we lose sight of the end goal.

37:06

Alexey

**Yeah, we lose sight of the end goal and then we just get too interested (too invested) in the details. Because it's exciting to tune all these knobs of a deep learning model, for example – or any machine learning model. Anyway, that's one thing – we need to focus on the end goal and sometimes this means doing things manually. Right?**

37:32

Jack

Yes. Exactly.

# The importance of understanding the reality that data represents

37:33

Alexey

**What’s the next one?**

37:34

Jack

The next one? Here's another unwritten rule. And that is… Well, let me describe the more common, written rule and then let me describe what I think the unwritten version should be – or what the version should be, that is unwritten. The commonly told rule is, “You should be obsessing over data and obsessing over understanding the patterns and the distribution and the nuances to it.” And I agree. However, I think that that is somewhat misleading. I think a better way of phrasing that is, “You need to understand not only the data, but you need to understand the process for the things that the data represents.” That is really what you should be obsessing over. For example, if you are looking at customer buying patterns for an online retail company, you need to understand who your audience is, you need to understand how they like to buy – it's even better if you yourself are a customer. You need to really understand, “What is it that is producing this data that I'm looking at?” From every angle.

Because it's very easy to, again, lose sight of what's actually happening when you're just focusing on numbers and data. All of this makes way more sense, and all of the directions you go with your analysis are much easier to do, if you focus first on just completely understanding the real-world process that the data represents. A quote that I've started saying more recently is, “You need to recognize that data is a shadow of reality, not reality itself.” Because data is just an artifact being produced by something. Inherently, it is always going to be an imperfect representation of something else, and if you have access to understanding that real representation of whatever it is, you should focus on that. Because the data will never tell you everything – it will only be a shadow of what you're actually trying to understand.

So if you can understand customer buying patterns, because you yourself are a customer, if it's understanding the output of a machine at a machine shop – you should be able to deeply understand what the machine does, all of its parts, why it's doing what it's doing. If you can do that, all the data will start to make sense. One example of where I see people trip up with this is that people will find nuances or patterns or significant things in data that are actually just not real – they're just anomalies with how the data is being produced. People will think, “Oh! I just found this new hidden reason for why a customer churns!” when, in reality, it's just sort of because the process for logging the data is biased in some way. Nine times out of ten, if you observe some anomaly in data, it's probably not a real-world anomaly – it’s probably just something to do with either the way the data was stored, or your perspective of the data.

# The importance of putting yourself in the shoes of customers

40:37

Alexey

**So you need to have this domain knowledge. Let’s say you work at Wayfair (or some other store that sells furniture) you need to understand the domain – let's say the customer journey from the moment they sign up, to the moment they receive furniture (hopefully in time? You need to understand this entire journey and all the problems that can happen on the journey? From the first step to the last step? And the best way of getting this domain knowledge is being a customer yourself, right? If you work at a company that sells furniture, go ahead and use that website to order some furniture for yourself.**

41:23

Jack

Yes, absolutely. In fact, I've always… Once I realized this, I started looking for jobs only where I was a customer because I realized how important it was to understand… To put myself in the shoes of the people I was building stuff for. So I really wanted to be able to identify with the customers. My first job… Well, my first job was at Trunk Club, but that was only a year in data science machine learning.

When I was at GoHealth, most of our customers were Medicare recipients, which is people who are 65 and older in the US, so I couldn't really identify with them. I always found that part difficult. But when I was at Wayfair, it was very easy because I was already a Wayfair customer – I already purchased furniture. We had just bought a house, so I was buying furniture all the time. It was just very relevant. And then the same thing…

42:11

Alexey

**Is this the house where you live now?**

42:12

Jack

Yeah, this is the house.

42:14

Alexey

**Is the chair from Wayfair?**

42:17

Jack

Actually, no. In fact, my office is actually – most of it is not Wayfair, but a lot of the rest of the house is. [Alexey chuckles] Yeah. And it’s the same thing with Fi, my current company. We make smart dog collars. I think my dog is probably somewhere in my office but…

42:35

Alexey

**Smart dog what?**

42:38

Jack

A dog collar. Like a collar that they wear around their neck.

42:44

Alexey

**Like a leash, of sorts?**

42:47

Jack

Yeah, it's more of a Fitbit, or a GPS tracker. So if a dog escapes, you can track them. But we also do stuff like, you know, step counts and movement and behavior tracking and things like that.

43:00

Alexey

**Like a fitness tracker but for dogs?**

43:02

Jack

Exactly. You have an app where you can see that your dog took this many steps today, or, “I took my dog for 10 miles of walks the past week. That's five miles less than what it was a week before.” So for people who are dog owners and interested in fitness or data tracking, it's a fun thing. Dogs and data tracking and health tracking are all things that I'm interested in. So to me, it was like, “Oh, this is a product that I am very interested in. It will be very natural for me to put myself in the shoes of a customer because I am one, essentially.”

43:38

Alexey

**And if you had a cat, not a dog, then it would be more difficult for you to relate?**

43:44

Jack

It would be, actually.

43:45

Alexey

**You would have to get a dog.**

43:48

Jack

Yeah. I mean, what's funny is a lot of people who come to Fi do end up getting a dog. [chuckles]

43:54

Alexey

**Well, that’s not bad, right? [chuckles]**

43:55

Jack

No, it's great.

43:58

Alexey

**Okay. So the second rule is, “Understand how the data is generated.” The quote you gave was, “Realize that the data is actually only the shadow of reality.” So you need to remember that and make sure you understand the process – then the data will make more sense and you will be able to understand that the patterns you find are not actually real patterns, but more like anomalies. And if you don't have this domain knowledge, you will not be able to understand that.**

44:32

Jack

Exactly. Yes.

# The importance of software engineering skills in machine learning

44:35

Alexey

**Well, I guess we have time for one or two more rules. I guess you have a bunch of them, right? What's the third one?**

44:43

Jack

Sure. Another unwritten rule that I've written about many times [chuckles] is that there is a lot of emphasis, when teaching machine learning, on mathematics and algorithms and things like that, which are absolutely essential. However, one thing that I think is often under-addressed is that, to be successful in machine learning (or to be high impact) you really need software development skills. At the end of the day, machine learning is software – it is a specific type of software – but when you build machine learning, you are building software, and it is likely going to go into a production state. So being familiar… not just familiar, but actually intermediate to advanced in software development skills, is critical to being successful in machine learning.

This is more of, I would say, sometimes a hot topic that I have, where not everyone agrees with me. But what I can tell you is that in the six years that I've been in machine learning, almost everyone who's been successful in machine learning has been successful largely due to their skills in software development, or that they will spend the time to get good at them. Because it's very difficult to have success, when you are only able to address one part of the machine learning funnel and you have to hand off everything else.

Because it requires that you'll be in a position where there are others who are extremely competent, and can work with you to hand off a model in a Jupyter Notebook and hand it off and then deploy it – there are a lot of steps involved there. To be reliant on others to do that, well, is very limiting in terms of your own success. It also means that it's difficult to put you on projects where there are limited resources. I can't give somebody a full stack ML project if I know that I don't have enough MLOps resources, which is especially true in smaller companies.

46:48

Alexey

**So you need to be some sort of full stack data science machine learning person, ideally. Right. If you can do that, then you can take a project end-to-end. You don't need to be an expert in all the areas, but just know enough to take the notebook and put it as a web service.**

47:09

Jack

Yeah. I want to make a careful distinction that… Because a lot of times what people think that I'm saying is that you don't need to know machine learning that well, which isn't true. You do need to know machine learning. You *also* need to know software engineering or software development. So I would say it's more so that there is a strong balance. Typically, the emphasis is on learning the machine learning side.

Again, because it's so difficult and it's very overwhelming that even understanding that part takes up all of your mental bandwidth, that thinking about software development too, just seems unnecessary. When, in reality, it is very important to understand, “Can you take a piece of code and put it into a web server on AWS?” If you can do that, and you can do it comfortably, you're in a really good spot to be able to deploy machine learning models and be a full stack ML developer.

# Where to find Jack’s content

47:58

Alexey

**What we teach in our courses – what we say in our courses, “Your model can be super good, but if nobody can use it (if it's just in a Jupyter notebook) then it's not good. It's of no use.” You need to be able to do that. All these unwritten rules are already written, as you said, somewhere. You probably publish them on LinkedIn. So can you tell us more about where people can find this content – all these things we talked about.**

48:32

Jack

Yeah. As of right now, it's all on LinkedIn. I write one post every day. I post it every day at 6:15 AM, central time. For the ones that I think are really good and informative… [cross-talk]

48:49

Alexey

**You wake up at 6 AM?**

48:50

Jack

Yeah. Well, I write it the day before, and then I schedule it to be posted at 6:15. But then I’m actually up at six to respond to people's comments. But I leave them in my featured section on LinkedIn. At some point, maybe I'll consider doing a newsletter. I know that's commonly what people do. At the moment, though… I only started writing on LinkedIn a few months ago. I started in late July, which was the first time I ever posted on LinkedIn. Even just now, I'm sort of figuring out what I want to say and how I want to say it. So, at the moment, that's where it is. But if you follow me there, you'll stay in touch with some other places that I'll be putting this data.

49:33

Alexey

**A good idea is – in LinkedIn, you can click on the bell icon, and the moment you do this, you'll get notified about the posts. Because the problem with LinkedIn is, you need to rely on the algorithm to make sure that next time I open it, I see your post. Meanwhile, in a newsletter, you don't have this problem. [Jack agrees] You actually deliver and then it's up to me to decide, “Do I have time to read it now or I'll have time to read it later?”**

50:08

Jack

That's actually a good point. I hadn't actually thought about that. That’s a very good point. Yeah,

50:11

Alexey

**I think you should start a newsletter. Because for me, if I really want to make sure I do not miss your content, I'd rather subscribe to a newsletter rather than rely on the algorithm to constantly show it to me. Because they, one day, might decide that “Okay, let's show more ads on them.”**

50:31

Jack

Actually, that's a very good point. I hadn't thought about that. Maybe I'll consider that now.

# Jack’s next venture

50:37

Alexey

**Yeah. What happens on November 15th?**

50:40

Jack

On November 15, that will be my last day at my company, Fi. I am leaving to start a new business in machine learning, data science, and more generally, data recruiting. There's an idea that I've been thinking through the last couple of years – I've known I want to do this, and I think now's a good time – which is, I think we are in a spot where we need a big shift in how technology hiring is done. I'm very unhappy for many reasons with the current state of hiring, both from an employer perspective and a candidate perspective.

I think there's so much noise in the system that it is very subjective who is considered for a role. People don't feel like they have any control over where they get considered for a role. I don't think people are interviewing for all the things that actually matter in success. Think through all the things I discussed today in terms of what drives success – how many of those things are actually interviewed for well in technology? Very few. I have done hiring differently throughout my career. I've approached it in ways that I think makes sense.

You know, you can take a look at the teams that I've built over the last five, six years, and they speak for themselves in terms of how strong they are. So what I want to do is take this and give it to other employers and other companies as, “Hey, here's a different way of thinking about hiring, where you can actually have a much clearer picture around what success looks like.” And for candidates – to be able to give them more control of, “Let's think through what your strengths are. How do we best showcase them?” Rather than putting you through arbitrary quizzes and questions that may not actually align well with your strengths.

So I'm going to start a business that is in technology, hiring, and recruiting and it's going to focus, in the short-term, on machine learning and data science and maybe some other data roles. But in general, I would like to extend it to any technology role. The idea is to be able to give people more ability to articulate, on the candidate's side, what their strengths are, and showcase that – make that available as opposed to just a traditional resume. And on the employer side, to be able to both understand what it is that they need to succeed, and be able to find that quickly instead of having to do many hours of interviewing on their own.

53:02

Alexey

**We will all subscribe – follow you on LinkedIn – and we will see all the updates about your new endeavor. I don't like saying good luck because you probably don't need luck – you need something like perseverance, more – but luck is also important. So have fun with your new project. And thanks a lot for coming, joining us today, sharing your experience – all these unwritten rules. And thanks, everyone, too. I unfortunately have to run. But it was a great pleasure talking with you. Have a great rest of your week!**

53:36

Jack

Likewise. Thanks, Alexey.