Visibility and Monitoring for Machine Learning Models

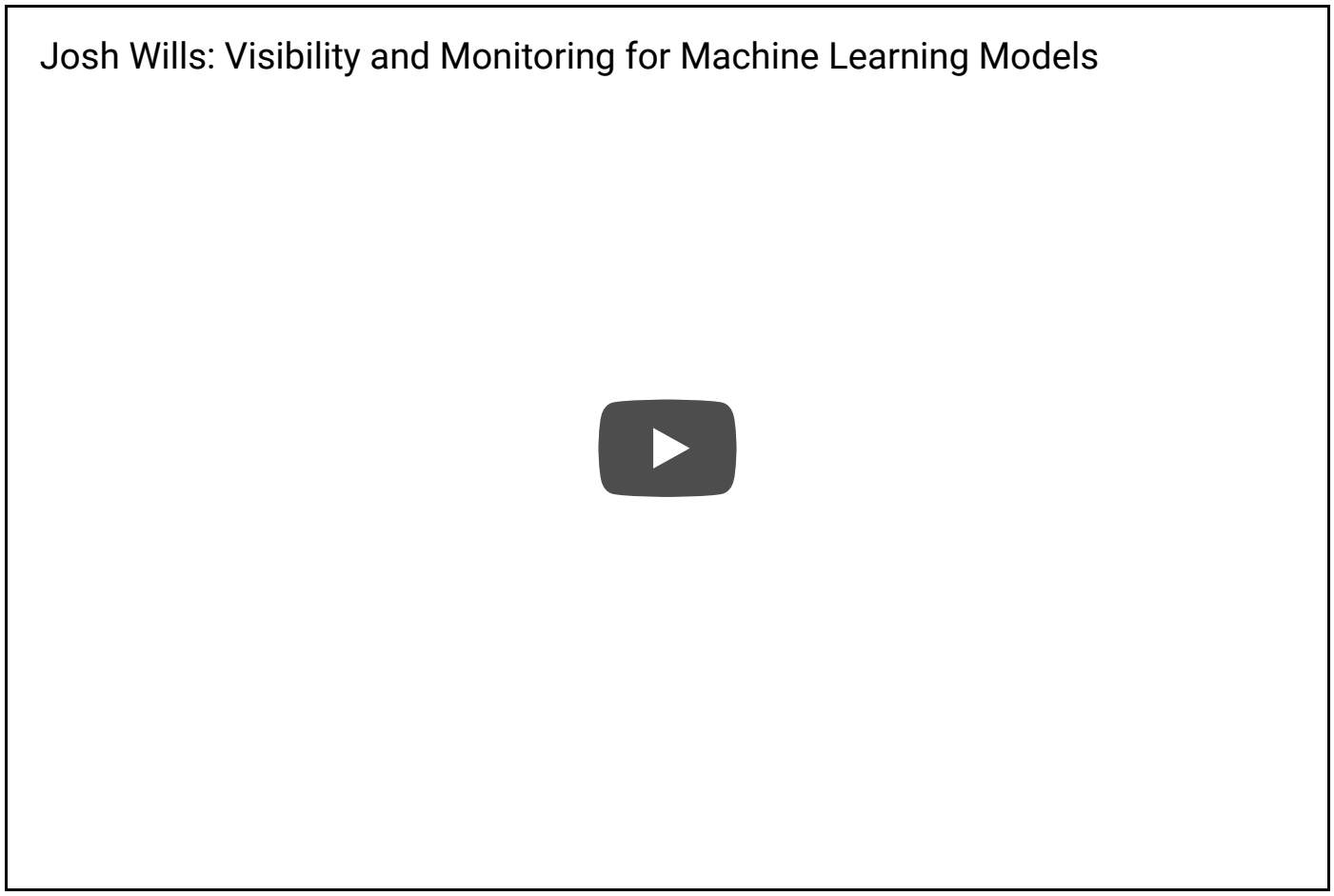
Kim Harrison

Josh Willis, an engineer at Slack, spoke at our January MeetUp about testing machine learning models in production. (If you're interested in joining this Meetup, sign up here.)

Josh has worked as the Director of Data Science at Cloudera, he wrote the Java version of Google's AB testing framework, and he recently held the position of Director of Data Engineering at Slack. On the subject of machine learning models, he thinks the most important question is: "How often do you want to deploy this?" You should never deploy a machine learning model once. If the problem is not important enough to keep working on it and deploy new models, then its not important enough to pay the cost of putting it into production in the first place.

"The tricky thing, though, is in order to get good at machine learning, you need to be able to do deploys as fast as humanly possible and repeatedly as humanly possible. Deploying a machine learning model isn't like deploying a regular code patch or something like that, even if you have a continuous deployment system." -Josh

Watch his entire talk below.



TRANSCRIPT

How's it going, everybody? Good to see you. Thanks for having me here. A little bit about me, first and foremost. Once upon a time, I was an engineer at Google. I love feature flags, and I love experiments. I love A/B testing things. I love them so much that I wrote the Java version of Google's A/B testing framework, which is a nerdy, as far as I know … I don't know. Does anyone here work at Google? Any Googlers in the audience? I know there's at least one because my best friend is here, and he works at Google. As far as I know, that is still used in production and probably gets exercised a few trillion times or so every single day, which is kind of a cool thing to hang my nerd hat on.

I used to work at Cloudera, where I was the director of data science. I

primarily went around and talked to people about Hadoop and big data and machine learning and data science-y sorts of things. I am Slack's former director of data engineering. I've been at Slack for about two and half years. I am a recovering manager. Any other recovering managers in the audience? I was going up the management hierarchy from first line management to managing managers, and I started to feel like I was in a pie eating contest, where first prize is more pie. I didn't really like it so much. Wanted to go back and engineer. So about six months ago, I joined our machine learning team, and now I'm doing machine learning-ish sorts of things at Slack as well as trying to make Slack search suck less than it does right now. So if anyone's done a search on Slack, I apologize. We're working hard on fixing it.

That's all great, but what I'm really most famous for … Like most famous people, I'm famous for tweeting. I wrote a famous tweet once: Which is a proper, defensible definition of a data scientist? Someone who is better at statistics than any software engineer and better at software engineering than any statisticians. That's been retweeted a lot and is widely quoted and all that kind of good stuff. Are there any … Is this sort of a data science, machine learning audience or is this more of an engineering ops kind of audience? Any data scientists here? I'm going to be making fun of data scientists a lot, so this is going to be … Okay, good. So mostly, I'll be safe. That's fine. If that guy makes a run at me, please block his way.

So anyway, that's my cutesy, pithy definition of what a data scientist is. If you're an engineer, you're sort of the natural opposite of that, which is this is someone who is worse at software engineering than an actual software engineer and worse at statistics than an actual statistician. That's what we're talking about here. There are some negative consequences of that. Roughly speaking at most companies, San Francisco, other places, there are two kinds of data scientists, and I call them the lab data scientists and the factory data

scientists. This my own nomenclature. It doesn't really mean anything.

So you're hiring your first data scientist for your startup or whatever. There's two ways things can go. You can either hire a lab data scientist, which is like a Ph.D., someone who's done a Ph.D. in statistics or political science, maybe or genetics or something like that, where they were doing a lot of data analysis, and they got really good at programming. That's fairly common data science standard. A lot of people end up that way. That wasn't how I ended up. I'm the latter category. I'm a factory data scientist. I was a software engineer. I've been a software engineer for 18 years now. I was the kind of software engineer when I was young where I was reasonably smart and talented but not obviously useful. I think we all know software engineers like this, smart, clearly smart but not obviously useful, can't really do anything. This is the kind of software engineer who ends up becoming a data scientist because someone has an idea of hey, let's give this machine learning recommendation engine spam detection project to the smart, not obviously useful person who's not doing anything obviously useful and see if they can come up with something kind of cool. That's how I fell into this field. That's the two kinds. You've got to be careful which one you end up with.

Something about data scientists and machine learning. All data scientists want to do machine learning. This is the problem. Rule number one of hiring data scientists: Anyone who wants to do machine learning isn't qualified to do machine learning. Someone comes to you and is like, "Hey, I really want to do some machine learning." You want to run hard the other direction. Don't hire that person because anyone who's actually done machine learning knows that it's terrible, and it's really the absolute worse. So wanting to do machine learning is a signal that you shouldn't be doing machine learning. Ironically, rule two of hiring data scientists, if you can convince a data scientist that what they're doing is machine learning, you can get them to do

anything you want. It's a secret manager trick. It's one of the things learned in my management days.

Let's talk about why, briefly. Deep learning for shallow people like ourselves. Deep learning, AI, big stuff in the news. I took a snapshot here of the train from my favorite picture, "Back to the Future, Part III," a truly excellent film. Machine learning is not magic. Machine learning is, it's basically the equivalent of a steam engine. That's really what it is, especially deep learning in particular. What machine learning lets us do is stuff that we could've done ourselves, manually, by hand over the course of months or years, much, much, much faster in the same way a steam engine lets us move a bunch of rocks from point A to point B. It's not something we couldn't do. We knew how to move a bunch of rocks from point A to point B. That's how we built the pyramids and stuff like that. But this lets us do it much, much faster and much, much cheaper. That's what machine learning fundamentally is.

There are consequences of that. One of the nasty consequences of it. Machine learning … There's a great paper that I highly recommend you read by this guy named D. Sculley, who is a professor at Tufts, engineer at Google. He says machine learning is the high interest credit card of technical debt because machine learning is basically spaghetti code that you deploy on purpose. That's essentially what machine learning is. You're taking a bunch of data, generating a bunch of numbers and then putting it in a rush intentionally. And then trying to figure out, reverse engineer how does this thing actually work. There are a bunch of terrible downstream consequences to this. It's a risky thing to do. So you only want to do it when you absolutely have to.

Lab data scientists want to do machine learning. Factory data scientists want to machine learning. Their backgrounds mean they have different failure modes for machine learning. There's a yin and yang aspect to it. Lab data

scientists are generally people who have a problem with letting the perfect be the enemy of the good, broadly speaking. They want to do things right. They want to do things in a principled way. They want to do things the best way possible. Most of us who live in the real world know that you hardly ever have to do things the right way. You can do a crappy Band-Aid solution, and it basically works. That's the factory data scientist attitude. The good news, though, of people who want to do things perfectly, they don't really know anything about visibility monitoring, despite knowing a bunch of stuff about linear algebra and tensors, they don't know how to count things. But you can teach them how to do Graphite Grafana. You can teach them how to do Logstash. They can learn all these kinds of things, and they want to learn, and they have no expectation that they know what they're doing, so they're very easy to teach. That's a good thing.

Factory data scientists have the opposite problem. They're very practical. They're very pragmatic. So they'll build things very quickly in a way that will work in your existing system. However, they overestimate their ability to deploy things successfully the way most not obviously useful software engineers do. As a result, they are much more likely to completely bring down your system when they deploy something. So that's what you want to watch out for there.

Another really great paper, "What's your ML test score? A rubric for production ML systems." I love this paper. This is a bunch of Google people who basically came up with a checklist of things you should do before you deploy a machine learning system into production. I love it. Great best practices around testing, around experimentation, around monitoring. It covers a lot of very common problems. My only knock against this paper is they came up with a bunch of scoring criteria for deciding whether or not a model was good enough to go into production that was basically ludicrous. So I took their scoring system and redid it myself. So you'll see down there,

if you don't do any of the items on their checklist, you're building a science project. If you do one or two things, it's still a science project. Three or four things are a more dangerous science project. Five to 10 points, you have the potential to destroy Western civilization. And then finally, once you do at least 10 things on their checklist, you've built a production system. So it's kind of a u-shaped thing.

This is a great paper. If you have people at your company who want to deploy machine learning into production, highly, highly recommend reading it and going through it and doing as much of the stuff they recommend as you possibly can. More than anything, for the purposes of this talk, I want to get you in the right headspace for thinking about what it means to take a machine learning model and deploy it into production. The most important question by far when someone wants to deploy a machine learning model is, how often do you want to deploy this? If the answer is once, that is a bad answer. You should never deploy a machine learning model once. You should deploy it never or prepare to deploy it over and over and over and over and over again, repeatedly forever, ad infinitum.

If the problem is not important enough to keep working on it and keep deploying new models, it's not important to pay the cost of putting it into production in the first place. That's thing one. The tricky thing, though, is in order to get good at machine learning, you need to be able to do deploys as fast as humanly possible and repeatedly as humanly possible. Deploying a machine learning model isn't like deploying a regular code patch or something like that, even if you have a continuous deployment system. The analogy I would use is it's kind of like someone coming to you and saying, "Hey listen. We're going to migrate over our database system from MySQL to Postgres, and then next week, we're going to go back to MySQL again. And then the week after that, we're going to go back." And just kind of like back and forth, back and forth. I'm exaggerating slightly, but I'm trying to

get you in the right headspace for what we're talking about here. It's basically different machine learning models are systems that are complicated and are opaque, that are nominally similar to each other but slightly different in ways that can be critically bad for the overall performance and reliability of your systems. That's the mentality I want you to be in when it comes to deploying machine learning models. Think about it that way.

The good news is that we all stop worrying and learn to love machine learning, whatever the line is from "Dr. Strangelove," that kind of thing. You get good at this kind of stuff after a while, and it really … I love doing machine learning, and I love doing it production in particular because it makes everything else better because the standards around how you operate, how you deploy production systems, how you test, how you monitor have to be so high just across the board for regular stuff in order to do it really, really well. Despite all the horrible consequences and the inevitable downtime that the machine learning engineers will cause, I swear, I promise, it's ultimately worth doing it, and in particular, companies should do it more so I get paid more money to do it. That's kind of a self-interested argument.

If you like to do monitoring, if you like to do visibility, if you like to do devOps stuff in general and you want to do it at a place that's done it really, really well, slack.com/jobs. Thank you very much. I appreciate it.

Kim is a writer and editor. She's pretty excited about the technology that LaunchDarkly is building and sharing that story with the community. Before LaunchDarkly, Kim did marketing for other SaaS companies, like Intuit and Pentaho. She earned her BA in sociology from UC Berkeley.