EPISODE 1263

[INTRODUCTION]

[00:00:00] JM: The company, Baseten, helps companies build and deploy machine learning API's and applications. Using pre-existing ML models or choosing from Baseten's library of pre-trained models, Baseten helps you instantly deploy API endpoints powered by those models to use in your applications. These API's easily scale and integrate with existing data sources. Baseten's serverless infrastructure enables chaining model outputs and pre and post processing code. They also use a drag and drop UI builder to create custom UI's for the applications all without learning React.

In this episode, we talk with Tuhin Srivastava, one of the founders of Baseten. Tuhen previously founded Shape, and also worked as a data scientist at Gumroad. We discussed machine learning API development, scaling ML-driven applications and the capabilities of Baseten's technology.

A few announcements before we get started. One, if you like Clubhouse, subscribe to the Club for Software Daily on Clubhouse. It's just Software Daily. And we'll be doing some interesting Clubhouse sessions within the next few weeks. And two, if you're looking for a job, we are hiring a variety of roles. We're looking for a social media manager. We're looking for a graphic designer. And we're looking for writers. If you are interested in contributing content to Software Engineering Daily, or even if you're a podcaster, and you're curious about how to get involved, we are looking for people with interesting backgrounds who can contribute to Software Engineering Daily. Again, mostly we're looking for social media help and design help. But if you're a writer or a podcaster, we'd also love to hear from you. You can send me an email with your resume, jeff@softwareengineeringdaily.com. That's jeff@softwareengineeringdaily.com.

[INTERVIEW CONTINUED]

[00:01:51] JM: Guys, welcome to the show.

[00:01:52] TS: Hi, thank you. Thanks for having us.

[00:01:54] AH: Hi, Jeff. Thanks for having us.

[00:01:56] JM: There's obviously a wide set of use cases for machine learning. But machine learning remains very hard to productionize. Tell me about the challenges of making machine learning in production.

[00:02:10] TS: Yeah, absolutely. So just to get started, like I think it's worth thinking about what does it mean to have a machine learning model. So machine learning model is a set of rules or it's an artifact that's coming out of a common machine learning framework like PyTorch or TensorFlow. And really what you end up with is a binary or an artifact in Python that takes some input and produces some output. To get that in production, you have to do a bunch of stuff. So firstly, you need to figure out how to serve that model. And so Amir can talk a bit more about this, about all the intricacies regarding around how you serve a model. But really what that means is getting it behind some sort of web framework and having it serve up an API that you can query either live in-real time or in batch mode. And so that would be in some Python code, in Airflor. It can exist in many forms.

What we really think is like a big challenge associated with productionization of a model actually comes after you've served that API or you've served that model behind an API. And that really comes from, one, how do you get it integrated back in to an existing system. So I think the best way to really think about this is what happens in a traditional use case. So the most tried and use case of machine learning that people often talk about is fraud. So let's say you have a model, you developed it in scikit-learn, you have deployed it now. What needs to happen? So that needs to be integrated back into some business workflow.

We find this to be actually very, very difficult. And so that means writing some rules on the output of the model, putting in front of a human who needs to review the output of the model. And then most importantly, having that wrap back from the human into the model itself. So establishing that feedback loop of data, decision, human, and then back to data. And we see a whole lot of challenges after the model has been created or has been deployed.

[00:04:21] JM: And tell me about some of the opportunities you see of simplifying that process of getting machine learning into production.

[00:04:30] TS: Yeah, absolutely. So the opportunities that we see really is – So from our experience, there's actually been a very, very large skills gap then. So the people developing models are generally the research scientists or data scientists. And so they traditionally come from academic background, or they've been an analyst and become data scientists and learned some machine learning. Hard thing here though is that a lot of these folks, while they're familiar with Python and SQL, aren't necessarily interested or able to do the website code. So like how do I turn this into a Python app? How do I turn it into a Flask app? But also the integration stuff. So like how would I get this to talk to Salesforce or Slack or, most importantly, creating visual interfaces or creating applications that end users can really use. And really, that's where we see our value. Or some of the most leverage that we can give is by allowing the data scientists to do more of those things after that model comes. So really, that's what Baseten is. So Baseten is a fast way for data scientists to build user-facing applications with machine learning models. What we try to do is put the data scientists first. And we give them a set of comprehensive tools so that they can build high-performance user-centric applications with their models front and center. So the opportunities here I like what's the fastest way we can allow data scientists to deploy a model? That is put it upon an API. After that, how can we allow the data scientist to rock that model in some custom business logic using just Python and not really having to worry about an infrastructure? That we see as a very, very large opportunity. And lastly, like how can we allow the data scientists to build apps and to really get their models in some usable form in front of users, or other stakeholders for them? If it's helpful, I can actually even provide an example there. Just going back –

[00:06:32] JM: Please do. Please provide an example.

[00:06:35] TS: Yeah. So like just going back, I'll give you a simple example and then a quite complex example. So the simple example is that I'm a data scientist, and I've come up with a new model that let's say does **[inaudible 00:06:46]** transfer. So that takes some existing – It takes some existing attributes of an image and it takes a new image and superimposes the style from that first image on the latter image. That's great. That's not really usable by anyone except the data scientists.

And so like, today, what would happen is either they create a Jupyter Notebook and have someone come over their shoulders so they can show it how it works. **[inaudible 00:07:11]** did release a paper and create a GitHub repo with this model so that other people can pull it down. What's really missing there is the ability for people who don't really understand what this model artifact is doing to be able to interact with the inputs and outputs of this model. So with Baseten what the data scientists would do is they'll deploy it. So we have a pretty simple client-side library. You call Baseten.deploy. It works on any major machine learning framework. And then from that, they'd be able to – Firstly, now that model is served. So it's ready to be called from somewhere else. But then within Baseten, they could actually write the Python code to surround the inputs and output of that model without having to figure out what's a Flask app, or how will this API scale. And then they'll also be able to really quickly cobble together that frontend so users can drag an image in and see what happens after that model is run. And so that's really like the use case of Baseten, which is one of the easiest way that people can express the value that the model is delivering.

The more interesting use case, although maybe not interesting, but the use case which has more value is around what we see at human in the loop workflows. And so one of the – As I said to you, like let's just talk about a use case. Let's talk about the fraud ops use case. This is something that I spent a lot of time doing from about five or six years ago, which was building machine learning models for fraud ops. And so I worked at an ecommerce company. Every time a transaction would happen, we had to flag it or figure out if it was fraudulent. If it was fraudulent, we'd then go and basically put it in front of a bunch of human reviewers to look at. And if not, we do nothing. If the human reviewer decided that this transaction was fraudulent, it would trigger a bunch of other workflow steps. We'd send an email to ourselves to say, "Hey, this happened." We'd write something to our database, and then we'd refund the transaction with Stripe. We'd then ban the user as well so they couldn't transact anymore.

Now, that's an enormous amount of workflow that I just described to you. You can actually do everything that I describe to you within Baseten. So within Baseten, once the data scientists had come up with their model, they deploy it. We also do some pretty basic version management and model management. So if they want to deploy new versions of the model, once it updates, they could do that if they want to AB test models against each other. They could do that. But

then you can also build all those workflow pieces within Baseten, so the business logic. They could write the Python code with us without having to figure out how to deploy this. And they could also then, on top of all that, build that human review workflow. So they can set up a queue view that the operators can look at and transact off of, and then also the stuff that happened after they're transacted. So the reintegration into the workflow of the core business.

So really what we're hoping to be able to do is to give that data scientist and machine learning engineer a bunch of leverage. So they don't have to rely on other product engineers or DevOps engineers, or backend engineers to flesh-out their services. And really what we hope that we will get out of this is, one, the initiation of that feedback loop that I alluded to earlier. And secondly, there'll be more value being shown from machine learning, because less models will be blocked at just the very, very early stage when training a model.

[00:10:52] JM: So can you generalize this a little bit. Tell me more about what the average user or the average workflow you see as being a less technical, less painful, machine learning deployment process?

[00:11:10] TS: Oh, sure. Yeah, to me, there're really three things that a model needs. So you need a way to get that model out of a Jupyter Notebook where a lot of data scientists do their work and behind an API. So it can be used by other services. So that's the first pillar. The second thing you need is integration. So the ability to integrate the output of that model back into some other workflow or to create a new workflow. So I gave you one example, which was fraud ops. This also applies to say something like the – A great example of this would be a recommendation system. So a recommendation system, the model spits out, takes the input a user, it spits out a bunch of – Honestly, really what it's spitting out is a bunch of weights saying, "Given this user, these are the three products with the highest weight."

So translating that back into something which is usable by another service actually takes quite a bit of business logic. And that's the second part that, the second pillar or Baseten, which is integration logic. The third one is user-facing applications. And these exists whether they're in internal tools, like the fraud ops I described to you, or something external as well, which is the way a way to express the power of your models for other humans to be able to interact with it.

And really, once you have all three of these, you can really kick off that feedback loop that I was talking about such that the actions that the human is taking in that user-facing application can really inform the inputs of the model so you can iterate on that model.

[00:12:52] JM: Great. So tell me a little bit more about what you're building at Baseten. What kinds of tooling have you built to support these simplified deployments?

[00:13:05] AH: Using Tuhen's examples and sort of the flow that he drew out, the storyline that he drew out in terms of steps that a data scientist would need to take in order to bring their model to fruition and to connect it to business value, there are a few systems that we have had to build that help the data scientist in that journey. The first of which is a Python client, shared on PyPI, pip installable Baseten client, that allows the data scientists to pip install Baseten into the environment where their model is trained. And with one easy command line be able to upload that model to Baseten's infrastructure.

The second piece is Baseten's backend infrastructure, which is built from the ground up using Kubernetes for go-to-market reasons actually. We have noticed that a lot of our users and potential users want to deploy Baseten within their own infrastructure, within their own VPC. And because of that, building it from the ground up using Kubernetes has given us a lot of leverage.

What we do in our backend is take the user's model that was uploaded through that first step and serve it, put it behind an API and have it be monitored. So what the data scientist gets at the end of that step is a simple API endpoint where they can call the model. But as Tuhen said, that's just the beginning of the journey. The next step is we allow the data scientists to be able to write Python code, their own custom business logic written in Python, without having to worry about Dockerising it, Kubernetes services, putting it behind flask. Without having to worry about any of those things, we allow them to write Python code, give them access over the environment that executes that Python code in terms of what third party packages are installed there, what versions, and we execute their code.

This is where in particular we're standing on the shoulders of giants, in particular, using the latest serverless development within the Kubernetes umbrella. And in particular, I mean, the Knative project, which is allowing us to, much more easily than before, take code from untrusted

sources and be able to execute them in a secure fashion with isolation and with a scalability built into it.

And finally is the frontend of Baseten, where our end users, the data scientists, can see the different models that they have deployed, can manage those versions, can write Python code, their custom business logic that interacts with those models, using a VS Code in the browser. And this is the final piece of it, is our UI Builder that Tuhen was describing, where we allow data scientists to be able to create user-facing interfaces without having to learn React, JavaScript, HTML, and be able to bring their models and their custom backend logic that they have written in Basedten, bring those to life, put a face on them, and be able to share them cross-functionally and have their end users interact with those applications.

[00:17:32] TS: And just add to that, really like Amir described pretty in depth like the three different parts of Baseten or the tooling that we're exposing for our end users. And I think what's really unique about our approach is that, for all these things, there are pre-existing ways to do these things. I think on a more deployment side, whether it'd be SageMaker, or whether it be people spinning out flask apps just to get something served, and this is already – Like this is a real pain that we have talked to a lot of user about. It's slow. It's cumbersome. It scales the data scientists and machine learning engineers don't have, but it already exists.

I think with the serverless work, again, like we've seen the rise of Lambdas, and Cloud Functions, and more and more stuff is being built using other technologies. But we really think there's like not too much taking the complexity away completely, but just shifting away from the end user so they can just focus on what's important to them. And for data scientists, as a user, that's really glue code that goes around the model its integrations back into existing services. But we're really trying to abstract that away from the user and really allow them to just focus on what matters to them. And then certainly on the frontend side, we've seen a lot of great companies being built in last few years. Like one that we draw a lot of inspiration from is Retool. Retool is amazing. And we've seen what they – I'd say like three or four years ago, everyone would have said, "No, you can't build internal applications in this way." We would have said that, to be honest. We didn't believe that. But I think they've really put on a clinic on how you can abstract away complexities that users don't need to know and really focus on the job that a user is trying to do. And when we bring all these three things together, we really do think that we're

able to give data science and machine learning engineers leverage that they haven't heard before.

[00:19:32] AH: Yeah, one principle that we started from the get-go was to make sure that we make easy things easy, but hard things possible. So as an example, when it comes to model deployment, if all you have is, let's say, a scikit-learn random forest classifier, really it should be as easy as one line to be able to deploy that. Lot of magic happens behind the scenes. But to you as a data scientist, the interface should really be that simple.

However, in some cases, your model doesn't fall into these buckets, and is something very custom. And so we've had to create ways for those harder situations to also be handled in a relatively simple way. Same thing when it comes to writing code within Baseten and having Baseten execute that. Sometimes that's really a simple code. And sometimes your code relies on third-party packages, some of which are not on PyPI, and they are yours, and you need to bring those somehow. Again, those are harder things, but those are also possible. So it's striking this fine balance between making the simplest cases be a breeze, but making sure that as soon as the needs of the data scientists diverges from those simple that we don't throw our arms up and say, “We don't support that.”

[00:21:16] JM: So you gave the analogy to Retool. And I think Retool is pretty interesting for changing the way that internal workflows work for these kinds of simple internal tools that people need. How do you expect a tool like Baseten, sort of a workflow reducer or a workflow simplifier? How do you expect that to change the overall cadence or workflow of machine learning deployments?

[00:21:48] TS: That's a really, really good question. I think it kind of strikes at the heart of why we started Baseten. And so why we started Baseten was we've been doing machine learning since – Honestly, like, as a team, like in different capacities in 2009. Our cofounder, Phil, was competing in the Netflix prize back in the day when that had come out. And I think what we've seen over the last decade, it’s like we were enamored by it and by machine learning a decade ago and we were so excited about all the possibilities that it would enable. And we truly think that it's going to have a profound impact on the way that we work and we live really.

But what we've realized is that the impact hasn't been seen. Like over the last year and a half, I've spent countless meetings talking to CEOs of large companies all the way from like CIOs that some of the biggest airlines in the world, all the way down to data scientists of three or four person companies, and machine learning initiatives for the most part are failing. Very few people can point to real machine learning that people are doing. And what we what we hope to do with Baseten is like if by simplifying all these things that we just described to you is that just make the cost of failure a lot, lot lower. And what we hope is that if we make it very, very easy to go from zero to one, so the day you have the simplest of models working, you can have a prototype in front of someone that they can play with. Or as soon as you have a model that you want to iterate with users, whether you can create that interface, or you can create that integration, so they can start to interact with it without you needing like six to eight weeks of engineering effort to make that work. What we hope is like bring this cost to failure down to be very, very low. And if we can in fact do that, I think what will happen is that, one, people won't be scared to try machine learning. The amount of comp teams that we have talked to that are like, “Yeah, we know, this data, we know machine learning can solve the problem.” But we also know that it's more than just coming up with a model. And so they keep putting it off. And so what we hope is that bringing down cost so that we can really just reduce the barrier to entry, there'll be more people willing to fail. And I believe that when more people are willing to fail, they're just going to be more attempts at things. And really, over time, we'll just see more machine learning.

Kind of like drawing that back to the Retool analogy that you referred to, like I do believe that there are more internal tools today because of Retool. And I think the reason for that is they just made it very, very easy to put together something very, very simple. And it didn't matter if it didn't ship. At least as you tried it and at least you were able to make some headway and not give up. And I think like similar to – If I do believe that, that Retool has enabled more internal tools, I do believe also similarly that we hope that more people will do more machine learning.

[00:24:45] HG: And if I may give two examples of this, two different examples, but they do point in this direction where tools like Baseten can help spare more initiatives. One is at a large FinTech startup, neobank, where they already have applied machine learning in particular in the area of fraud. And they have a team of machine learning engineers, ML ops, etc., supporting these efforts. The support structure is already in place for the fraud model in particular. Now, the

data scientists there tell us that they have lots of other ideas where they think applying machine learning can help in the areas of customer support, customer success.

However, the barrier to entry is quite high. If they get a model up and running, trained rather, and they want to now deploy it into their infrastructure, the infrastructure that is built for this fraud model in particular, the cost of getting this new model into this infrastructure is weeks, weeks of working with DevOps, infra, security, and making sure that your model can now be served within the model serving infrastructure that they have. And because of that, what they end up spending their time on is not these new greenfield projects. But instead, they work on making that one fraud model 1% better, more features, feature engineering, etc.

Now, with Baseten, what they could do, what they did, was to actually explore those other initiatives, those other use cases of machine learning, because they saw the path where if it did work, they could easily build applications with it, serve it, and build UI around it, and ultimately ship it to be used by the customer experience in this example. Well, that's one example where there's already ML in place. But the barrier of getting a new model into the existing ML infrastructure is quite high. And tooling like Baseten can help.

The second example is a bit different and is going to touch upon a part of Baseten that we haven't talked about so far. This one is – The second case is a mental health startup, where they have an application that their patients use and they can, as part of the application, provide freeform text. Now, the mental health startup has to review this freeform text and look for signs of self-harm. They have an engineering team. They don't have any ML folks. And they're quite new to ML. But we gave them the idea that, within Baseten, you can actually spin up this review queue tool very quickly. And very quickly, you can add machine learning to it. We have what we call a model made, a library of pre-trained models, open source models, that we have put together and made it very accessible for someone who doesn't know too much about ML, who can come in, see the different use cases that these models can attack, choose one, and very quickly play around with it. Give it new input. See how it behaves.

And so in this case, the case of the mental health startup, we on-boarded them. And very quickly, they ended up with this review queue that they could put in front of the clinical operations team to review the freeform text submitted by the patients ranked by the output of

the machine learning model that they used that determines whether there are signs of self-harm in the freeform text or not.

And all that was, by the way, was a pre-trained zero shot classification model that hadn't seen anything about self-harm in particular, but configured correctly can, right out of the box, without any additional training, perform really well in determining self-harm in freeform text.

And so these two examples I think show that tools like Baseten can really lower the barrier for exploring machine learning and applications thereof and finding new ways to add value where it was not possible or easy before.

[00:30:14] JM: So can you give a little bit more on the interface of Baseten? Like are you a CLI tool? Are you an API? Are you like a low-code visualization system? Give a little bit more about the user experience?

[00:30:31] TS: Yeah**,** totally. I think the good and bad answer to that question is that it's all of the above. The first interface that a data scientist or machine learning engineer would have with Baseten is at the CLI level. So Amir mentioned earlier, it's a pip installable package. It's as simple as import Baseten, and Baseten.deploy. And so you will basically upload your model to Baseten from the CLI.

Once you're there, you're going to end up with basically a very, very simple model management page on baseten.co. And then you'll be able to look at the models. You'll be able to look at different versions of the models. See how to call that model. And I think that's step one. And we see this integration cost, but that's step one. The next part of the interface is basically something that looks like Google Cloud Functions, or Amazon Lambdas where you're basically able to write serverless code and put your model in the middle of that. So you're writing code in the browser. It's a VS Code instance, and Monaco editor that we're using, and you'll be able to write that code and test that code in the browser, even attach a cron job to trigger it. Basically all the things you would need for a simple but powerful backend service. But you're configuring that primarily in the browser. We know like, frankly, that this is probably one of the more controversial parts of Baseten about writing that code in the browser. But our users used two tools like Domino Data Lab and Dataiku, and there is some precedent there.

The third part of Baseten is a drag and drop UI builder. Again, like pretty similar to something like Retool where you're kind of dragging components onto the page and configuring them to interact with user input, and then integrate with that backend service that I had described a second ago.

[00:32:24] JM: Tell me about some of the engineering problems in bringing Baseten to market.

[00:32:32] HG: We have had plenty of those, some of which now that I look back, they seem easy in retrospect. But at first, they were quite challenging. One of the first ones was, again, when we were trying to design that interface that allows the data scientist to give us their model for us to serve it and building it in a way, designing it in a way that allows for, again, the simple cases to remain simple, but for the complex cases to also be possible, and striking that balance between magic and configurability.

Let me add a time dimension to this discussion, because I think it makes it more interesting. The very first version of Baseten that we put out there for a few users to use was a proof of concept that was built with as few components as possible as quickly as possible. The backend architecture was pretty much a Django backend running on Heroku. And we wanted to prove to ourselves that this product and this product direction has wings.

Once that was proven to us, thankfully in parallel, we learned a lot from our users in terms of their requirements on how they wanted their code to be executed and the kind of flexibilities that they wanted. And also what I alluded to before, which is what environment, what sort of VPC's, they would be okay with their code to be executed. In particular, not being okay with Baseten to be hosting their code and their data.

And so all of those learnings helped us arrive at the current architecture that is Kubernetes-based from the ground up and uses Knative, and on top of that, Kserving for model deployment and serving and also Knative directly for running the user submitted code in isolation securely and with scaling, auto scaling, including scaling down to zero, built into it.

That journey was one that I'm really glad that we took, even though it resulted in a lot of code that we ended up throwing away. But I don't think we would have arrived at the current architecture if it wasn't for building something that works end-to-end as quickly as possible, putting in front of users and learning from those interactions.

Another very challenging aspect of Baseten in terms of engineering has been this UI builder, this frontend view builder that we were talking about. How do we allow end users who are not engineers have probably never interacted with React, don't know the abstractions there, aren't familiar with those. How do we allow them to build complex user interfaces in a way that they feel like they understand how the different pieces are connected to each other? How their frontend connects to the backend that they have created to the models that they have deployed? That was a product meets engineering challenge.

And what we ended up doing there is taking a lot of inspiration from React and Redux, our learnings from using those ourselves, learning from the abstractions that those provide and exposing those to end users, to our users, to the data scientists in a way that is understandable for them. And also building it all in a way that is composable down the line so that our users can create their own composite components, and reuse those across the applications within the organizations as time evolves. That was another interesting balancing act that we had to do where we wanted to build something that can support that future without having to necessarily build it today. But knowing that we didn't want to throw our frontend architecture away once it got to the points where we wanted to support those more complex features.

So that also took a couple of tries. There was also a lot of throwaway code involved. But, again, building something end-to-end early on. Think of it as imagine you're building a large pipeline that needs to be very thick going from point A to point B, instead of building these large, thick parts, one by one and finally connecting them to each other. What we strived to do was to build a very thin pipe that goes end-to-end and showing that to users, learning from that, and then slowly thickening certain parts of that pipe, which, again, involve some throwaway code. But I don't think, again, we would have arrived at the current architecture if it wasn't for those end-to-end experiments that we did early on.

[00:38:55] JM: The market for machine learning tooling is obviously pretty crowded. And it's tough to get people to try out your tool because there's so much overhead to onboarding and integration. What's your go-to-market strategy?

[00:39:11] TS: Yeah. This is also been something that we've been working on. AndI think you're 100% right in saying that the market for machine learning tooling is very, very – It's crowded. That's for sure. And I think more importantly, everyone says they do a little bit of everything. Early on, we started working with large enterprises and kind of used them as our design partners. And we very quickly realized that those problems didn't necessarily map to the problems of the end user who we were trying to help. And so what we realized was that a lot of the companies in this space, we're also kind of making that mistake where they're going – Whether it's a mistake or whether they're taking that approach of going top down, finding the CIO who wants to do some machine learning and then figuring out the buyer in that way. We think that's going to be necessary for us at some point, especially for larger contracts and for larger installations of Baseten. But really, what we learned, I'd say, like towards the end of last year was that there's not that many tools that data scientists love and machine learning engineers love. And we actually went in as like – I remember like a couple weeks at the end of last year where we sat down and we just reached out to a bunch of folks that we've been talking to and asked them, "Hey, could you described the tools that you're really excited to use?" And we realized that those are very, very small subset of this.

I think data scientist, a lot of them love like the Jupyter Notebooks. All of them love Airflow. What we were really excited by with this possibility, the emergence of this new thing that data scientists were expressing was like they wanted tools that would help them show their work off to other folks. And that's really what's been our go-to-market strategy for the last little bit, which is let's build something that an individual data scientist can adopt by their own. The first time they have a model that they want to show off to people, we want to be their first protocol.

Now, what we really hope is that they'll end up building internal tools within Baseten for their models, and because our architecture allows us, like we'll scale with their use case. But that's to be determined whether we get there. But for now, we're super, super focused on adding value for the individual data scientist and machine learning engineer. And then also, in parallel to that, having those conversations with those larger teams. So we are aware of the use cases when

we do need to make the expansion. So as an example of that, one of the things that you hear a lot when you go and talk to a larger team of data scientists, say, inside that FinTech company that Amir mentioned earlier, is version control. You need ways to version the code. You need ways to version applications. You need separate kind of versioning, and publishing flows and deployment flows of the application itself. The individual data scientist doesn't really care about that today.

And so while we are working with a larger, larger company, a design partner to get those requirements, right now, from a product perspective, we're really, really trying to make it easy for individual data scientists to, one, get on-boarded, to get that aha moment. Basically be able to understand very, very quickly the possibilities hopefully within 10 or 15 minutes of using Baseten and then using that word of mouth or love from that data scientist to kind of build-up credit within an organization and with other teams, if that makes sense.

[00:42:52] JM: Totally. So give me a little bit more about your general assessment for how data science and machine learning workflows evolve over the next five to 10 years.

[00:43:04] TS: Yeah. I think there's like three things I'll mention here. So the first one I would say is there's an increasing prevalence of really really good pre-trained models. I think, if you look at Hugging Face, you're familiar with them. They provide natural language processing models in kind of this standardized format. But really, I'd say every single chat we have with a user or a potential user who's using natural language processing, they're using Hugging Face, and they are often data scientists. So that's the first one, is that an increased prevalence of pre-trained models. And like that's the first thing we're going to factor in. The second one is everything with engineer graduating college today, or a very, very large majority of them have taken a course in machine learning or data science. I think that's really an important point, it's because they were able to – This wasn't the case a decade ago. Most of my engineer friends probably still – For them, machine learning is broadly represented by the very, very public self. So OpenAI, GPT3 and stuff like that. But more and more new grads and engineers know what machine learning can do for them. And they may even know how to train a basic model. And so like taking those two things together, I think the workflow is actually going to expand in two ways. So the first one is they're going to make it really easy for people to get started with

models that already exist. If you can frame the problem, there's going to be stuff off the shelf for you to get started. And then the workflow will involve a tuning step.

By tuning, what I mean is you get a model, you run the model. It spits out a bunch of outputs, and then a user gives feedback to the model about how they classified, how it did. And that model, that is used to kind of update that model itself and fit it to that particular use case. And then really, you really have to start to think, “Okay, if more people can do machine learning, then the tools will kind of update to represent that prior.” And so what we'll see is a lot more tooling, not so much involved around – You will see more tooling that allows people to go from zero to one with models and integrated back at the workflows.

I think the second thing is just leverage. I think a company that I really love is Vercel, which allows frontend engineers to deploy their websites, and really gives them kind of full stack powers without really exposing them, exposing them to that complexity. And I think that's exactly how in my mind the machine learning tooling market will evolve and the workflows will evolve, which will be how can we give data science and machine learning engineer to who are pretty technical themselves? They know a bit of Python code. They know quite a lot of Python, they know SQL. Maybe they know some Julia? How do we interact with these tools and give them leverage to do more? And I think you're kind of seeing this across the sector, even with stuff like Tecton, which is the feature store. That to me is giving data scientists and machine learning engineers kind of like this data engineering superpower. This ability to manage their features without having to think about all the complexities that go into tracking, tracking-computed data over time. They're the two different things.

So the first one will be to accommodate a whole new set of users who know machine learning, but may not be experts, but can use models. What does the new tooling look like to support those users? And secondly, how can we give data science and machine learning engineers more and more leverage so they can focus on the stuff that really matters, which is making their models really good and improving their data, which as a result we’ll end up with better models. And then we hope the better models will feel actual outcomes and then people will invest more into machine learning.

[00:47:19] JM: Okay, well, thank you both for coming on the show. It's been a real pleasure talking to you about Baseten.

[00:47:24] TS: Thank you. I really appreciate of your time.

[END]