1:30

Alexey

**This week we'll talk about MLOps and DataOps. We have a special guest today, Santona. Santona started her data journey as a researcher at CERN, then worked in NLP, and then worked as a Python engineer. She did many things, and one of those things was working at the company behind Airflow. Astronomer, right? She worked in the data space and now she works at Upsolver, where she leads data engineering and science, doing research. She will probably tell us more about that. She's passionate about building, as well as about empowering others to build end-to-end data and ML pipelines, which will be the topic of today. Welcome!**

2:20

Santona

Thank you so much. Happy to be here.

2:24

Alexey

**The questions for today's interview were prepared by Johanna Bayer. As always, thanks, Johanna, for your help. And let's start.**

# Santona's background

2:32

Alexey

**Before we go into our main topic of DataOps and MLOps, let's start with your background. Can you tell us about your career journey so far?**

2:41

Santona

Yeah, absolutely. So I'm a physicist turned ML engineer turned data workflow builder, kind of. I feel like I started as a full-stack data scientist because as a physicist, there was data engineering, modeling, feature engineering, machine learning, and then result presentation. So I think of myself as a generalist, and now I'm going even more meta, in the sense that I want to solve the problems that allow other people to solve their problems. So I'm passionate about building workflow authoring tooling in a way that is with the user in mind, where users are other developers like myself so that we can build the right kind of tooling that helps others do their work as simply as easily and as enjoyably as possible. That's me. [chuckles]

3:39

Alexey

**Can you maybe tell us about it in a few more details? Because it's quite a curious career journey. You worked as a researcher, then you worked as an ML engineer, and then you worked as a (I don't know if it's the correct way of saying this) as a data engineer – you worked in the data space, before that in the ML engineering space, and before that as a researcher. So what motivated you to actually go into ML engineering? You were a full-stack ML person as a researcher – you needed to do all these things. So why exactly ML engineering? How did it happen to you?**

4:14

Santona

Yeah, that's a great question. As I was leaving academia and thinking about what was next at that time, (this was in 2019-2020) NLP was pretty exciting. I mean, obviously, it's still very exciting.

4:31

Alexey

**It's getting more exciting every year. [chuckles] It's becoming more and more exciting.**

4:35

Santona

Yeah, exactly. So I wanted to go in... I was just very, very eager to jump into that field. In physics, I was doing work with massive data and fast-moving data and all of those things, but they were mostly numerical and categorical – the data types. As I did just a few projects here and there on NLP or natural language-based applications, it was really refreshing in the sense that I, as a human, could interpret the models and the predictions' inferences that they were making in an intuitive way, that this algorithm was also doing. I think that's kind of what I latched on to. And so I started talking to folks that were doing NLP, and this opportunity arose where we were building a predictive routing engine using natural language queries to figure out how to best resolve them – how to best answer the questions that people were asking.

Yeah, that was a lot of fun. Building an intent architecture to figure out what people might ask in this entire network, per product and per language, and then curating that over time, training the classification models on those intense – it was just a lot of fun. So I would say the transition was a bit narrower in the sense that I was mostly thinking about ML algorithms, deep neural networks, and sort of the production pipeline in which they were deployed, as opposed to a more end-to-end process. And then the other difference was the kind of data and so on and so forth. But it didn't feel like a huge transition. [chuckles] Because I mean, at the end of the day, it's still data and ML pipelines and serving some end-user purpose.

6:40

Alexey

**Just text instead of numbers. Then you convert text into numbers, and then it's all the same.**

6:47

Santona

Exactly. [laughs]

6:52

Alexey

**Then this is how you ended up being an ML engineer, right? [Santona agrees] Of course, you needed to deploy all these neural nets that you created, and you needed to deploy the chatbot – these things needed to be scalable and whatnot.**

7:06

Santona

Yeah, exactly.

# Focusing on data workflows

7:08

Alexey

**But then you decided to go... I think you said “even more meta,” right? Focused more on data workflows. So how did that happen?**

7:20

Santona

Yeah, I think... Well, a lot of things sort of fell into place. But I started using Airflow at Directly in the production stack, and workflow authoring. It all sort of comes down to how your pipelines are orchestrated, or if the orchestration is abstracted away, that's even better. But a lot of the “engineering” component of data and ML pipelines has to do with “Okay, what are my dependencies? When do things get executed? What do I do when things go wrong (fallback protocols, etc.)?” So, thinking about the pipeline line as a main asset, as a main thing to take care of, is the direction in which I was going. I was also excited by the fact that this was Astronomer, which was a company that was managing the Airflow OSS. It was just also that I joined the data team very early, as it was being formed, so it was a lot of open green fields (open opportunity) as far as what we were going to be able to do.

The component of that job that was most fun for me, which maybe I didn't even realize when I was joining, is the user research aspect of it. I was now thinking beyond the use cases that I had come across, be it particle physics or NLP, to “Okay, what is everyone else doing? What is everyone else trying to do with Airflow?” Or more generally, “What are the pipelines? What kinds of use cases are there?” So I really enjoyed learning across domains, across industries, what folks were trying to do and what their pain points were. That got me more and more excited to solve these problems. Following that thread is how I ended up at Upsolver.

SQL is kind of the lingua franca of data. I really enjoy working in Python. The other component of this is that I love going to places where I know the least and then building up my education there. As a physicist (as a CERN physicist), I was working in C++ and some Python. The data structures are these trees – tuples of tuples, nested data structures that we have – our custom storage methods and our custom query methods, and so on and so forth. All of that is to say, I've never used SQL before. [chuckles] I started using SQL to query data directly and then, through the years, I learned better SQL. So that's actually part of the motivation for coming to Upsolver. We author pipelines, or in our platform, you author pipelines just as SQL. You just define the outcomes that you want and then everything that needs to happen to make that possible is handled by us. And of course, we have our own dialect. For example, we have these keywords like “sync” versus “unsync” that sets a dependency – that allows you to say what the dependencies between pipeline components are. That's sort of the thread that I followed. [chuckles]

# Upsolver vs DBT

10:48

Alexey

**I never used Upsolver. I only sometimes receive marketing information from you, in addition to webinars. But I've never actually seen that tool in practice. What you described sounds similar to DBT, right? You have a bunch of SQL queries, the tool looks at these queries and then figures out in which order these queries need to be executed. Is this a correct assessment?**

11:12

Santona

That's a fine analogy to draw. DBT is a good partner of ours, so there's definitely an Upsolver plus DBT story. Specifically, we do a few different things and I want to sort of delineate between them. One is pipeline authoring (workflow authoring). You can go into our UI and author, you can use our CLE Python SDK, or in DBT, actually, you can author Upsolver pipelines. And then we have the actual engine that executes it. So DBT doesn't have its own execution engine.

11:50

Alexey

**So DBT delegates it to the data warehouse [Santona agrees] to Snowflake, BigQuery, whatever.**

11:57

Santona

Exactly. Or to Upsolver. With our new DBT connector, you can write the pipeline in DBT and execute in Upsolver. Then the second component is, we have a data lake where the data actually lives and flows through, and where you make your transformations on. Then the most recent thing that we added to this already pretty, pretty end-to-end tool is a focus on data ingestion. We guarantee high-quality data ingestion, whether you ingest it into Upsolver or Snowflake (which is also a good partner of ours) or just your S3 bucket.

For complex data sources, we've found that that's another pain point that folks are struggling with – if it's CDC. If you want to do Change Data Capture on your production databases, or you have to pull in data from queues like Kafka or Kinesis, and you're usually used to batch data processing in Airflow, for example, then this is sort of a different set of tools that you have to think about or a different mindset. So we're making that very easy – streaming data, large files, nested structures, and so on and so forth, is sort of our specialty. What we do on the ingestion front is guarantee strong ordering, guarantee... The things that data engineers usually have to think about while they're doing transformations, we just take care of the front end.

# ML pipelines vs Data pipelines

13:25

Alexey

**So a lot of data engineering stuff you mentioned. I remember a while ago, we had a podcast episode about MLOps here (I think it was like two years ago), and one thing that the guest, Theo, said back then was that you should never confuse a data pipeline with a machine learning pipeline, because a machine learning pipeline is a very special sort of pipeline. So do not do that. This is a big mistake. I'm wondering, what are these ML pipelines and data pipelines? Can you maybe tell us in a few words what they are, and what the main differences between these two are?**

14:05

Santona

Yeah, absolutely. I think it comes down to the application. What is the ultimate value that you're trying to add to your end users? When I worked at Directly, where our product was this predictive routing engine, where the ML model was the thing that was deployed, and what was receiving data, making a prediction on it, and sending that prediction to end users – that was an ML pipeline. The data engineering aspects of that were very ML-focused, which is to say, as you said, the question is asked on some UI and then comes through to Directly through RabbitMQ. Receiving that question vectorizing it, and doing all sorts of filters on it – that is data engineering. This is very, very different from, for example, analytics engineering, which is happening in Snowflake, let's say, where you're interacting with the data, you're doing SQL transformations and these things.

So it's very focused in some sense, it's a very focused use case, very specialized use case. It can be computer vision, or NLP, or multimodal models, which are also very common. It can be numerical data, but still, the feature engineering that's happening, you usually kind of do the deep dive as an ML engineer, or even as an ML model. You do the deep dive, you figure out exactly how your data needs to be transformed in order to serve this ML use case in production. As opposed to most data pipelines today (maybe someday this will change) but the vast majority of data pipelines today are serving analytics use cases, where you might still have ML as part of the pipeline, but it's not the main... It's not a first-class citizen.

It's only because you think that there's some pattern, some trend, to extract, that makes more sense with a regression or with a classification or something like that. So to me, that's the difference. Here, you're starting with data from different sources – again, the source is less relevant – but once your data is in your system, you're really looking at the data, figuring out how different entities in the data relate to each other. For example, you get your product data from your application database. You get your CRM data from Salesforce. You get your Zendesk data, et cetera, et cetera. And then you have to create this model of “Okay, what are the different primary entities in my data? What's the mapping between them? How do things relate to each other?” And then you try to get to a representation of all of that data that can help serve analytics use cases downstream, which might be other data engineers on dedicated teams – there are different models, as we know, there's data mesh, etc. There are different ways in which you can power that, but the main focus is building an understanding of the business and then serving up that understanding to end users.

Often, the users are internal, and your customer success team wants to know when a certain client is going to churn – that's the person you're serving. The other shift there is, “Who's the persona that I'm building for?” So yeah, I agree with the other podcast speaker. I think they're very different. There are certainly elements that are in common. At the end of the day, data is flowing through both pipelines. But the architecture of the pipeline should always be use case-driven, application-driven.

Lastly, you didn't ask this, but I'll sort of add it – we use the word “Ops” And I think that it's a little bit up in the air as far as how you define it. Or maybe there isn't, but I have decided that I use a very simple definition of Ops, [chuckles] which is that it's all the steps that I need to take to consistently deliver value to the end user from whatever the thing is – from data or from ML. So if I have an ML model, great. It already does everything that I need it to do, but getting it from there to actually serving end users consistently and reliably, with little to no downtime, that sort of that's my Ops portion.

# MLOps vs DataOps

18:44

Alexey

**So using your definition, MLOps would be steps you need to take to deliver value from a machine learning model. And then DataOps would be steps you need to take to deliver value from what exactly – SQL queries?**

18:59

Santona

No, I would say from all of your business data. Yeah, I don't know. Do you agree that a lot of data applications today are really that we're trying to build an understanding of the business? The representation... the final... I mean, there's no final state, but usually, parts of your Snowflake database, or whatever else database, that you're exposing to other users are meant to be a complete representation of all the data in the business.

19:37

Alexey

**I do agree. Although being a data scientist, I have a somewhat biased view on this, because most of the data pipelines I worked on were more like, “How do I transform the data in order to feed it into a machine learning model?” Even though it was mostly data engineering pipelines, still, the last couple of steps were mostly training a SciKit Learn model and publishing the model somewhere. For me, my understanding of the difference between a machine learning pipeline and a data pipeline, based on this experience, is that an ML pipeline is a more specialized version of a data pipeline.**

**You have different steps that are ML-specific – most of the steps are not ML-specific, but then you have feature engineering or extracting numbers from text, (the vectorization thing that you mentioned) and then training a model, which is ML-specific, and then publishing this model software. So these last few steps are ML-specific, and the rest or not. For me, the distinction was that it's just a more specialized version of a data pipeline, but based on your definition, or based on what you described, it looks like these are quite separate things. One is focused more on ML, and in the other, you focus more on business understanding. What they have in common is the pipeline part, which is all this orchestration, but the rest are different. Is this a correct understanding?**

21:12

Santona

And they can have the data in common as well. For example, at Directly, primarily our ML product was there, but we also did analytics. We did analytics, we ran SQL queries on the data that had come in – all the historical data – and then what was going on with their ML model, like performance, and so on and so forth. I think that there's definitely overlap, and there's definitely a closed marriage. I mean, definitions are only as good as they're useful.

I think the only useful part of the way (that I like to think of it [chuckles]) is that if you are a data engineer who has been building an ML pipeline, or a certain service of an ML product – let's say not at a startup, where you're doing things end-to-end, but rather at a larger organization, such as Reddit or Netflix or something, where there is a big ML-based product.  A recommendation engine is what serves the end users, and you have ML engineers that are developing models and deploying models and looking after them in production.

There might be separate data engineers, who are called data engineers, and they're more in charge of what's happening upstream. That's another way of splitting it up – within the ML application pipeline, you've got upstream data engineers, and downstream, you've got ML engineers. That's totally a valid data engineer persona and workflow. The one thing I will say is that a data engineer is used to doing different work than a data engineer at a company whose main purpose is something else. Let's say it's not an ML-based model. Hmm... It's really hard these days to find companies that don't have ML as a product. I'm really struggling. So your end-user product is something that's... Astronomer, let's go with that.

It's just a managed Airflow service, there is no ML component to that. So what were we doing as a data team? Our focus there was data engineering for the purpose of building business understanding and serving those up. There, we were thinking much, much, more about, “How do I represent the data in a way that makes sense to our business partners across the organization, not our ML model.” So you're thinking human-centrically. You're thinking in terms of, “What is the mapping here?” And you're doing a lot more data modeling. You're doing a lot more of those views and tables. As you go down the pipeline, you're getting more denormalized because you're building these mappings and complex relationships.

It's just a very different kind of workflow, in my opinion – having experienced a little bit of both, granted. I definitely want to learn from folks who have done more of either. But I wouldn't hire someone who is building data models and SQL primarily for analytics use cases into a data engineering position for an ML use case unless they were eager to make that switch, and unless they were eager to learn whatever gaps there were. It's not a one-to-one mapping between those two data engineering roles.

24:57

Alexey

**Even the tools are different, right? For ML to use cases, you might have tools like Airflow, Spark, Kinesis, Kafka, RabbitMQ, and all these things you mentioned. But for the first case – for developing business understanding – the tools are different. The tools could be DBT Upsolver and these kinds of things. I think when you were saying that, here, the main goal is to build a pipeline and develop this business understanding and think, “How do I represent the data in a way that's understandable for business?” I think there is a term for that right now, called 'analytics engineering,' right? Am I correct?**

25:43

Santona

I mean, I think so, yeah. I was pretty excited to see the term come out. I don't see it being used as often. I don't see a lot of people with analytics engineer as a title, but I think it makes a lot of sense.

25:59

Alexey

**We have a data engineering course at DataTalks.Club and one of the modules there is about DBT, which we call it analytics engineering. It seems like it's kind of synonymous – people who call themselves analytics engineers usually use DBT and if you use DBT you're kind of an analytics engineer. But I think this comes from the DBT lab. So they came up with this term to kind of differentiate between usual data engineers and these business-oriented data engineers.**

# Tools used for data pipelines and ML pipelines

26:43

Alexey

**Okay. So we started talking about the tools already. What I wanted to ask is, what sort of tools are usually there? I gave a few examples, but maybe you have some more examples? What kind of tools do you need for different pipelines – for data pipelines or for ML pipelines?**

27:07

Santona

Yeah. I mean, I think you hit the nail on the head. The common stacks today are still... there's some orchestration engine, it could be Airflow – I mean, nowadays, there are Prefect, Dagster, Mage, etc. Depending on what kind of transformations you're doing, you could also use, let's say, Spark or Absolver or something like that for the ML pipelines. It all depends on how you break up your workflow. That's the thing I love about the modern data stack. [chuckles] There are things you hate about it as well, but you can sort of pick and choose your different pieces and build your own. So it's like building your own adventure, kind of. You can definitely... I think it's less about the tooling. But on the other hand, I will fully agree with you that DBT would be hard to use in an ML pipeline, I think. Probably. I'm sure there, there are teams that are doing it. Again, that's the fun part.

But the warehouse aspect of the data landscape, I think, is much closer to analytics engineering and analytics use cases than ML. Nowadays, feature stores are becoming a thing, and vector databases and stuff. Obviously, we were building those things for NLP ML applications. Even three years ago, we were building those in-house. It's a data lake in an S3 bucket with the hive meta store, and so on, and so forth. Nowadays, there are managed services for that, which I think is cool. Any sort of abstraction, or any time you can take away some work and abstract it away, I think it's great. But it's more focused around... as a human, you don't really need to fully grok and understand the data – how it lives and how things relate to each other – as long as you have that layer of metadata that retains that information. You can have a programmatic querying of it and usage of it.

# The “modern data stack” and today's data ecosystem

29:16

Alexey

**You mentioned one thing – the modern data stack. You said it's a good thing because it consists of many different components, and you can kind of replace these components. But what is this modern data stack? I see this term used quite often. For me, modern data stack is... You have DBT, you have Snowflake, and then some things for ingestion, and then you call this thing a modern data stack? Is that right?**

29:47

Santona

Yeah. I think that's certainly a common way to define it. For example, Upsolver would be the third thing – Upsolver for ingestion and then Snowflake and DBT. That's your modern data stack. That's really all you need for your analytics use cases. But it's a choice, right? It's a choice that you're making. You also have these platforms, like Databricks, that have different pieces that allow you to do all of those things. I mean, you should always make build and buy-in decisions by looking at exactly what you need for your use case. It's hard for me to attach the idea of a data stack with specific brands and specific companies. Again, what I love about today's data ecosystem, and perhaps it's incorrect for me to use the term “modern data stack,” more generally, in today's data ecosystem (new three-word acronym) [chuckles].

What is cool is that there are these specialized tools that you can pick and choose from. Again, if I do an ML application, I don't really want to be working with Snowflake and DBT, because then I'm going to feel a little bit lost because I don't have Python in my hands. Granted, Snowflake now has better Python support. But that's the whole thing – rather than thinking about tools, which then you're at the mercy of the tool to come out with the thing that you're used to, or the thing that you need, you can instead say, “Okay, fine. My data lives in Snowflake. Great. I can still get it out and work in my Python environment and then bring in my ML libraries and do the rest of the work.” To me, that's the modern data experience.

31:51

Alexey

**“Modern data experience” or as you already mentioned “today's data ecosystem”. [Santona chuckles] So what are the building blocks of this data ecosystem? You mentioned Snowflake, which is a data warehouse, which is the place where we eventually put all the data – the data is already cleaned and ready to be used for analytical purposes. But what are the other components of this ecosystem?**

32:17

Santona

Often, there is a data lake, whether it's surface to end users or not, as you said. Even within Snowflake or Databricks, in tabular representation as well, you usually have a bronze, a silver, and a gold – raw, ingested, and modeled, and then “marked,” etcetera, etcetera. So there's that, and you can sort of choose where the data warehouse comes in, in that data pipeline. But often, before we ingest it into Snowflake, we will still stage the data somewhere externally, in a lake, whether it's...

# Staging the data and the concept of a “lakehouse”

32:57

Alexey

**What does it mean to stage the data?**

33:01

Santona

There are different patterns. If you use a dedicated ingestion tool, then usually... I'll start with the definition. Data is coming in from some source, and it's going to be in a raw format. You pull it into a place that is meant to be accessed only programmatically and not by individuals. The access controls are programmatic. Then, from that data staging area, you pull the data into a warehouse, where it's more human-centric.

33:39

Alexey

**You don't want to query the source every time you need the data.**

33:42

Santona

Exactly.

33:43

Alexey

**You put it there once and it's just there. And the next time you need the data, you take it from that place.**

33:48

Santona

Yeah. I mean, that's a way you could design it, for sure. Also, it's part of your data pipeline, right? I think the staging area is useful because, even if it's regular (every day at 1 PM, my data from Zendesk is pulled into my GCS bucket or my S3 bucket, and then another, let's say Airflow DAG, picks it up from there at 3 PM, or the DBT model picks it up.) Does it really work with S3? Probably.

34:30

Alexey

**I don't know. [chuckles] [inaudible] Yeah.**

34:33

Santona

But yeah, it pulls it into my warehouse. The staging area is useful because, if you do have catastrophic failures, then that's a place to catch it. Or if you need to wait for some data, like let's say some service is down and you need to wait for the data, and so on and so forth. At Upsolver, we're sort of turning that idea on its head.

We're enabling you to do all of those things, like stopping the pipeline if there's a catastrophic failure, or if there are bad quality rows in your incoming data, choosing what to do with them, such as drop or warn or etc. In this ingestion portion, without having to think about the stage, or without having to think about the actual, underlying lake underneath it – we actually don't need a lake in the way that we've written it – the data does move pretty much continuously through without really stopping in the Upsolver SQL lake, but it gets a lot of the benefits.

Anyway, the main thing with a staging area is that it's a holding area. No one's supposed to be learning anything – no one's supposed to be querying it – no one's supposed to be making business decisions on top of it. It's like a buffer zone. I think over the years, we've iterated toward more abstracted staging and more hidden staging. In Fivetran and Upsolver, your data will come in and be staged, but it's within the service. You don't have to think about setting up an S3 bucket or a GCS bucket for that staging. So again, abstraction is great.

36:18

Alexey

**It's happening, but under the hood. As the data engineer, the user of Fivetran or Upsolver or another ingestion tool, I do not really think about that. It's happening under the hood, but all I care about is that the data is being moved somewhere. Right?**

36:38

Santona

Yep, exactly. Yeah, the data is being moved into my warehouse or the “lakehouse,” basically – the lake that I want the data to persist in and want to be able to query.

36:50

Alexey

**And if something happens, then I rely on the tool to re-do this thing. I don't manually go to this buffer zone (to the stage) to do things. I don't even know about its existence, the tool just pulls it from somewhere and re-does the work.**

37:08

Santona

Yep, exactly.

37:10

Alexey

**That's cool. That's very convenient. [Santona chuckles and agrees] Okay. And this is how the data ends up in a warehouse, but it's in raw form. Right? We haven't processed it yet. [Santona agrees] That's the first initial step for a data pipeline. What happens next with this data?**

37:30

Santona

Sorry, when you say “We haven't processed it yet,” we haven't done any complex transformations. [Alexey agrees]

37:38

Alexey

**What kind of transformations do we actually do at this stage? Do we do any?**

37:43

Santona

In the staging/ingestion stage?

37:46

Alexey

**Yeah.**

37:47

Santona

Yeah, we don't really think of them as transformations. They're more of various cleaning or quality assurance mechanisms of sorts. For example, this is where we would dedupe data. It's the kind of thing that you'd have to, later on, say, “Oh, let's do a select distinct on my table,” you don't have to do that anymore because every entry is guaranteed to appear only once and then exactly, once consistency is strong [inaudible]. So those are the kinds of things that we do. The other thing that you can do with Upsolver (I'm not so sure about Fivetran and others) – this is where you can set your PII strategy (your governance strategies). So if you have a field that you want to mask, or if you have a field that you want to hash, you can configure those things through the Upsolver UI. So when the data appears, let's say in Snowflake, which is for, again, for human consumption, (the data is already sanitized in some ways) the things that are hidden, any duplicates are dropped and data are strongly ordered.

39:02

Alexey

**There is some initial pre-processing, which might be enough for some use cases. It's already ready for some consumption. But maybe for more complex queries, for more complex reports, then a data engineer or an analytics engineer, needs to take this data and do some extra transformation.**

39:22

Santona

Exactly. Yeah.

# Transforming the data after staging

39:23

Alexey

**That is the next step – transformation. Right?**

39:25

Santona

Right, that is the next step – transformation/some degree of modeling. The way that I've done it, and I think it's generally a fairly good way of doing it is – your data is going to come in from a number of different sources, even at small startups. If you have an analytics team, you're going to be able to bring in like 10 different sources of data. [chuckles] So the next stage after your data has landed in your warehouse or (lake house) is to figure out... Well, you have primary keys coming in from the data source, but what are the mapping keys, the foreign keys? What is the relationship between each of these, let's say tables, that have come in and what can we build on top of that? “What do we actually want to answer for my business?”

Usually, there are very specific questions and this is why it's important for the data engineers and analytics engineers to talk to the end users – the other teams within the visit the business partners – To understand, “What exactly are you trying to answer?” Then I go, sort of back-propagation and figure out “Okay, to answer this question, I need to pull in data from this table, this table, not that table,” and so on, and so forth. So that's the kind of modeling where you're thinking in terms of business entities and business questions. So I think that's what's next.

40:55

Alexey

**Here, this is the transformation/modeling phase, when we actually prepare data in order to show it as a report or a dashboard. Here, we need to work closely with business people, we need to talk to them, we need to understand what all these keys mean – foreign keys, like everything you mentioned – and we also need to make sure that business people, who consume this information, also understand what's happening in the result we give to them. Right?**

41:25

Santona

Yeah, and there are optimizations that you want to do as a data analytics engineer. You want to make sure that your data isn't “super” denormalized [chuckles] and that the same thing doesn't exist in many, many different places. The lookup indices across your tables should be efficient. I mean, there are lots of things to sort of... for example, in a data and motion pipeline, if your sources are streaming, then this is the stage where you think about what downstream use cases rely on these streaming upstream use cases, and how do to keep them in sync.

Let's say I need to combine two different Kafka queues to answer a question. How do I make sure that that transformation in the downstream application gets the relevant data from each of those two streams? That's another thing, sometimes in the batch world, we don't have to think about that, because things are happening in batch. But in a tool like Upsolver, where we're combining batch and streaming and have this “streaming-first” mindset, then this is where you would really think about, “How do I do transformations that respect and stay true to the data dependencies and still answer the questions?”

# What happens after the modeling phase

42:55

Alexey

Okay, **so we have the ingestion phase, then we have the transformation/modeling phase, and then? Do we have anything else after that?**

43:05

Santona

That depends. [chuckles] Actually, we said one thing that I want to double-click on. [chuckles] Dashboards and reports and other deliverables, I think are actually a little bit further downstream, and we shouldn't really be thinking about them at the transformation/modeling stage. The requirements are important. I want to know what dashboards folks are wanting to see, or more specifically, what questions they want answered. This is something that I said earlier in a LinkedIn post, and it resonated with folks – you shouldn't come in with dashboard requests to your data team, you should come in with, “This is what I wanted to know. I want to know how this entity that I care about is going to change, or what's the trend,” and so on and so forth. And then the data team works with you to figure out exactly how that breaks down, what the metrics are, and how we want to present them.

But I think that the whole process is still a little bit further downstream from the modeling because, at the modeling stage, you want to get to a place where you feel that the main entities of your business, for analytics purposes, are covered – are being regularly updated. Then on top of that, there's a second layer of transformations, where you're just going from “Okay, I'm taking these entities, and I'm just writing the transformation that answers my question.” And in some places, this is called “mart,” or you could go a bit more directly from those entities into your dashboards, but there are these two layers. There's ingested data, models data, and then there are answers.

# Human-centric vs Machine-centric **pipeline**

44:57

Alexey

**Ingested data, model data, answers. I wonder how different it is for a machine learning pipeline. Because in an ML pipeline, we also have a modeling phase, but it's a different kind of modeling phase. But I guess some things are similar, right? We might still have some ingestion phase. Maybe it's not our team who is doing that, but somehow it's happening, and then we have the data that we can use for transforming the data in such a way that we can use it in order to train a machine learning model. So I guess it's somewhat similar.**

**We have ingestion, then we have transformation, then we have modeling, which is a different kind of modeling because in the case of a data pipeline, the modeling is more about what kind of things we have in the database – in our data. But here, the modeling is actually training the model. Then finally, there's all these other things like serving the model instead of creating dashboards. So it's kind of similar. Right? But the tools we use are different.**

46:01

Santona

It's kind of similar. But if my application is an ML application, I don't really need to model the data, as you said, by the entities. I don't have to build this business understanding for the ML model to then use that to answer questions. Because by that time, it's already cognitively simple and there are other ways to do it or with simpler ML models. In that sense, the feature engineering or data engineering component for the ML application is more focused around, “How do I best featurize the data for the ML model, not for human beings?” Largely, I feel like it's a matter of whom the data model that I'm building is for.

If it's for analytics use cases, for humans, then that's a different mindset, as opposed to if it's for powering an ML model and for a machine to pick up – then it's different. In feature engineering, you're absolutely right, I have to think about a lot of the same things. I still have to de-duplicate my data, if a field has nulls too often, then I have to figure out how to fill that in or drop, or whatever decisions I choose to make. But at the end of the day, my output is this sort of vector space – this large (or maybe it's not that large, maybe it's smaller) but all of those decisions I'm making as an ML engineer (as an ML application-focused person) and I am working with the data to get it to a better form for my machine to be able to pick up.

# Applying skills learned in academia to ML engineering

47:57

Alexey

**Now I'm talking with you, and I'm wondering – four years ago (I think you said 2019) when you were switching to ML, maybe you didn't know about all these things – all these different steps like ingestion, staging area, and all that – but you were an Airflow user. So I'm wondering, how did you convince Astronomer to hire you with your background?**

48:31

Santona

Yeah. I joined Astronomer in '21 I think. So I...

48:37

Alexey

**You already worked as an ML engineer mountaineer for two years.**

48:40

Santona

Yeah. And that's where I was using Airflow. [chuckles] I mean, I have data experience in physics as well – just writing data pipelines end-to-end, doing the data transformations that I needed to do, and so on and so forth. So it wasn't really a big challenge. [chuckles] There are things that I learned, of course, along the way, coming out of academia that were different. I think the main thing that I found was that things needed to be translated a little bit. In physics, specifically in particle physics, we used Cron, we had a custom built job scheduler, that was just like most orchestrators and schedulers. Underlying that was just just Cron. We had to set our own dependencies and we had to set our priority on things like, “How important is this job to run?” Because the jobs were really really massive. [chuckles] So when it needs to be run by and then put in a queue, and then it gets reprioritized. All the same considerations – all the same underlying technologies were there, but we just had a different... We had a custom solution for each of the different parts because that's what we needed. We couldn't query our particle collision database with SQL or with pandas or something. I couldn't pull it into a pandas data frame.

So I realized – and this is going out to folks who are maybe making this transition now or just trying to think how their skills relate. It's not useful for me to say, “Oh, I have the software. It's called root. And this is where the data lives. And I open the branch, and then in that data branch, I open the next branch,” and so on, and so forth. That's not useful. That may be what you're doing, but it's not what folks are gonna resonate with. If you instead explain “Well, the tool doesn't matter, but I have this deeply-nested data structure (event data) which is how my data is going to be created in particle collisions and how it has to be stored. And in order to do analysis and run aggregates on it, I have to be able to reach all the endpoints in my leaf nodes. I have to run filters across different events on those properties. Those are my features, and then I do ML on that.”

It doesn't matter what... we had a minimizer for – it was a custom-built minimization engine to minimize the loss function. But you don't have to talk about that. You can say, “Okay, and then I did this minimization. It was a regression, (or it was whatever else) and this is how I got to the answer.” So it's a level of coming out of the weeds, coming out of thinking about exact tools, and thinking about, “This is what I did, and this is exactly what you do as well, so you should hire me because I have experience doing this.”

# Crafting user personas based on real stories

52:12

Alexey

**So the tools were maybe not your typical SciKit Learn tools, it was something else, and the way you accessed data was different, but in the end, you needed to access data – you needed to run some modeling, some training process on top of this data, and then you have a model. These are pretty much the steps you need to take now with SQL and SciKit Learn, right? I understand. Then you worked as an ML engineer, and then somehow you ended up at Astronomer. But you said you used Airflow, so it wasn't difficult for you to convince them to hire you. Or was it?**

52:54

Santona

Yeah, yeah. I think that having used Airflow was a positive [thing] for sure, as was doing pipelines end-to-end. Again, we were just forming the data team, so there were some open questions around what the data team was going to be – what exactly the charters were going to be. Some of the work that I did at Astronomer was NLP analysis – again, ML for the purpose of analytics, so that's different. Or ML for the purpose of showing how you can use Airflow for ML. It was actually a hybrid role at Astronomer. It's a hybrid role at Upsolver as well. The kind of work that I do is really this... I write data pipelines. Sometimes I write data pipelines to show how a data pipeline can be written using best practices for a specific use case or for a specific domain. Sometimes I craft those user personas and user stories and these are based on real people. For example “My friend, who's Head of ML at this Series C startup is doing this. I know, he's pulling in data from these sources. He's using Prefect for orchestration.” Those things.

So I write a persona based on that. I write a persona based on my friend who is working at this feature flagging, A/B testing platform, and he has this frustration around... he's identified a data set that is incorrect. And for the last six months, he's been trying to stop all the downstream applications because they're flawed. So I write those pain points down and those user stories down, and then I think about “How do I solve this using the tool that I want to build and improve?” Then the data work comes in. It's after that market user research. It's after thinking about problems. Then I go to, “Okay. Let me actually write this pipeline, and let me use Airflow to do it (or let me use Upsolver to do it).” Now you see that their pain point is solved and it's simple in these [particular] ways. And there it is.

And in the meantime, I'm doing data work, because I'm actually pulling in data. One of the analyses I did at Astronomer was building a predictive model on how long a GitHub issue on the Airflow project is going to take to get resolved. So, it was an NLP analysis of the issue title and the issue comments, and then a multi-class classifier on top of that to say, “Hey, these 10 issues came in the last five hours. This one's going to take days to resolve. This can be done quickly.” So we can prioritize based on that.

55:56

Alexey

**It looks like there is quite some overlap between ML engineering and data engineering. With the skills you got as an ML engineer, you could just say, “Hey, I know how to do this, this, and this. I might not be able to do this thing that you need me to yet, but I will learn. Hire me.” And then “Okay, you seem to be good enough. Let's hire you.” And then you just learn all the rest at work. This is how it works?**

56:22

Santona

Yeah. And this is why I like being a generalist and encourage others to be generalists. All the words that we use and all the definitions and titles, I think can be restrictive. The more important thing is to actually figure out what the work is that needs to be done and what the gaps are between your skill set and that work, and then fill those in.

# A framework of curiosity

56:49

Alexey

**Do you have a framework for doing this?**

56:52

Santona

Um... A framework of curiosity? No, I don't have an organized way of...

56:58

Alexey

**“What is this term? Let me look it up.” Right? This is how it happens? [Santona agrees] Do you follow some resources online? Podcasts or some articles? Or does it just come up in conversations with colleagues? How does it work for you? How do you feed this curiosity? How do you know what you're curious about in order to learn about that?**

57:25

Santona

Yeah, that's a great question. I don't religiously follow any one podcast or publication. But I try to, again, build my own adventure from various sources. If you do it this way, the only concern is vetting. How do you vet the right sources? How do you make sure that what you're learning is a good practice as opposed to a bad practice? And that's tricky. For that, I mostly rely on my network of folks that I respect and admire, and I am very unafraid to just ask, “Hey, what do you think of this?” Or “I read this and this seems like a good idea on principle. Is this what you're doing or are you doing something else? And why?” So all of those conversations together with online resources is what helped me learn. I can't really speak to and say, “Oh, you should go read this or go listen to this.” Maybe someday I will, but not today. [chuckles]

58:26

Alexey

**So you just have a pool of resources, you read from these resources (or consume these resources). For some things, you run them by your colleagues, friends, who can tell you “No, this actually is not exactly correct.” Then you re-think, “Okay, can I trust the source?” [Santona agrees] Or if they say “Yeah, totally. This is how I do this,” then you can see that you can continue consuming from this resource.**

58:51

Santona

Yeah, because then you're getting the real experience from a person who's doing it and building it. I do really like, I should say, companies that have engineering-specific blogs, where they're talking about the problems that they're solving in a blog, outwardly. With that, again, you're getting the real experience with someone who struggled with something and then came up with something. Those are really informative to me.

# Santona's book and resource recommendations

59:16

Alexey

**Okay. Last question for today. Are there any books or other resources that you can recommend, let's say, if somebody wants to learn more about these data pipelines that we discussed?**

59:29

Santona

To learn about data pipelines? Again, I'm a little bit hesitant to recommend specific books. I think that, depending on what you want to do... because reading a book is also a time commitment. [chuckles] So there's that. But I hear that the Fundamentals of Data Engineering, by Joe and Matt, is really solid for learning data engineering skills. There are white papers that companies that are in the orchestration space will publish. I found those to be somewhat useful.

My friend Bas, who I think is still at Astronomer (maybe not anymore) wrote a book on Airflow, basically. That's solid if you want to specifically learn orchestration with Airflow. But I think we're just moving a little fast right now – unless you are a fast reader and someone who learns really quickly – to rely too much on books, because they become outdated so quickly. [chuckles] Yeah just broaden... consume from other sources as well – podcasts like yours, talk to folks who are doing the actual work, and read shorter-form content as well.

61:01

**Alexey  
Okay. Thanks! That's all we have time for today. Thanks a lot for joining us today, for sharing your experience. And thanks, everyone, for joining us today, too. I guess that's it for today.**

61:13

Santona

Thank you so much.

61:14

**Alexey  
Yeah. Have a nice rest of your day and have a great week!**