SDS PODCAST EPISODE 815: POLARS: FASTER DATAFRAME OPS, WITH MARCO GORELLI

Show Notes: http://www.superdatascience.com/815

Jon Krohn: 00:00:00 This is episode number 815 with Marco Gorelli, Senior Software Engineer at Quansight Labs.

00:00:05 Today's episode is brought to you by AWS Cloud Computing Services, by Babbel, the science-backed language learning platform, and by Gurobi the decision intelligence leader.

00:00:20 Welcome to the Super Data Science Podcast, the most listened to podcast in the data science industry. Each week, we bring you inspiring people and ideas to help you build a successful career in data science. I'm your host, Jon Krohn. Thanks for joining me today. And now, let's make the complex simple.

00:00:51 Welcome back to the Super Data Science Podcast. Today we've got a deeply technical episode for you. I know many of you love that. This one's with the tremendously talented communicator of complex technical topics, Marco Gorelli. Marco is a core developer of the popular Python Libraries, Pandas and Polars, as well as being the creator of the Narwhals Library. He's spoken at several major Python conferences such as PyData. He's taught Polars professionally and he wrote the first complete Polars plugin tutorial. He currently works as a Senior Software Engineer at Quansight Labs. Previously he worked as a data scientist and was one of the prize winners from amongst over a hundred thousand entrants of the M6 forecasting competition. He holds a master's in Mathematics and the Foundations of Computer Science from the University of Oxford.

00:01:36 Today's episode will appeal primarily to hands-on technical folks like data scientists, ML engineers, and software developers. In this episode, Marco details what the hot, fast-growing Polars library for working with dataframes in Python is. It already has 65 million downloads and 28,000 GitHub stars. He also talks about

how Polars offers up to a 100x speed-ups relative to Pandas on dataframe operations, how the lightweight dependency-free Narwhals package he created allows for easy compatibility between different dataframes libraries such as Polars and Pandas, how he got addicted to open source development and this simple trick he used to be a prize winner in super popular forecasting competitions. All right, you ready for this dazzling episode? Let's go.

00:02:24 Marco, welcome to the Super Data Science Podcast. It's awesome to have you here. We're in London, it's sweltering hot. We took the train in from Cardiff to be here. Welcome to the show. I really appreciate you making that trip. So you were introduced to me by Reshma Shaikh, who has recommended many wonderful guests on the show. I said that I was going to be in London and who should I speak to? She wrote back right away and said, "Marco is who you should talk to." And I reviewed your work. I reviewed some of the talks you've done in the past and I couldn't be more excited to be here interviewing you with an episode, particularly focused on Polars, which I've been excited to learn about for a long time.

00:03:04 There also is, there's an interesting connection here. In episode number 765, we had your CEO Travis Oliphant, so we'll talk about your company Quansight later on in the episode, and he's a huge player. He was the originator behind NumPy, behind SciPy, and so maybe someday some of the packages of yours that we'll be talking about today, like Narwhals will also be as invaluable to data scientists. Anyway, that was a long intro welcoming you here. Welcome to the show, Marco.

Marco Gorelli: 00:03:37 Thank you for having me.

Jon Krohn: 00:03:39 Nice. Yes. So Polars, you've been deeply involved in the development and maintenance of popular open source

dataframe manipulation library. So, Pandas is probably the one that are listeners are most familiar with. I think probably anyone who's a hands-on data science practitioner is used to manipulating dataframes in the Pandas library in Python. But Polars is increasingly popular and it's developed by Quansight Labs. Actually, I guess a lot of support for Polars comes from Quansight Labs.

Marco Gorelli: 00:04:18 I think I'm the only person in Quansight Labs who's contributing to Polars. Polars came out of Ritchie Vink, he's a developer in the Netherlands. It was originally his lockdown project in 2020 and then last year he started a company around it, imaginatively called Polars.

Jon Krohn: 00:04:39 Yes, and I guess the idea of the name Polars is that it comes from a panda bear and a polar bear. That must be it, right?

Marco Gorelli: 00:04:47 That's part of it. The other part is that it ends with R-S. Polars is written in Rust and the file extension for Rust files is typically dot R-S.

Jon Krohn: 00:04:58 Let's actually, because I know Rust is going to be important to this conversation, so tell us a bit about the Rust programming language and why somebody should maybe consider using that programming language over other languages.

Marco Gorelli: 00:05:10 All right, yeah, that's a fun one. I started learning Rust, not because I particularly wanted to learn Rust, but because I wanted to contribute to Polars. Just tried using Polars one Saturday afternoon while procrastinating on some life admin. Found a little bug, thought it might be fun to fix it, and got a bit addicted to the process. What people usually highlight about Rust that's nice is memory safety. So Rust has some built-in mechanisms which make it quite difficult for you to make certain kinds of

mistakes, which are a lot easier to make in certain other kinds of programming languages. It's also got quite a readable syntax and nowadays Rust, it's been around for I think at least 10 years, so IDE support is really nice. You get lots of really nice support if you are writing Rust in VS Code, and I think that's why it consistently ranks as one of the most admired languages in the Stack Overflow developer surveys.

Jon Krohn: 00:06:13 It does, it does. That I've seen for sure. And so what is it that makes it easier to program in Rust. I've read, I've never looked at a line of Rust code myself, but I've heard that there's things related to it compiles very easily. You're very unlikely to run into programming errors that you put in.

Marco Gorelli: 00:06:33 Well, it compiles slowly. I don't know about easily. In fact, most people when they try writing Rust, they experience fighting the borrower checker. There's Rust, which enforces certain constraints, which make it hard to run into certain kinds of bugs as you've said. But also it can be a bit annoying, especially at first when you can't understand necessarily why the compiler is rejecting code which to you looks perfectly safe, but it's doing you to save you, and you really appreciate that later. Once you get it working, you're a lot more confident in what you've written.

Jon Krohn: 00:07:11 So it's actually the opposite. It isn't that Rust compiles easily, it's that Rust compiles with quite a bit of complaining. That makes it such a desired language. So it does the complaining for you instead of I guess your downstream users or clients.

Marco Gorelli: 00:07:26 I'd say it's if you're in it for the long run, it's a good choice. If you just need to do some quick experimentation, some quick EDA, maybe not. I think that leads to one of the design decisions behind Polars. So

Polars is written in Rust, but it's got a Python API. The idea is that most data scientists should interact with Polars directly through the Python API. That's something they're probably familiar with that can fit in with the rest of the tool chain, but development of the library itself happens in Rust.

Jon Krohn: 00:08:02 Very cool. So yeah, so back to Polars more specifically. So now we know it's Rust background. We know that even the RS suffix on it is related to the Rust filenames, so that's clever. We know that you develop in Rust in order to be developing the Polars library. For somebody who is a data scientist who isn't necessarily a software developer like you are, but for somebody who wants to be taking advantage of Polars, why should somebody install Polars into their Python instance instead of Pandas?

Marco Gorelli: 00:08:40 I hate to give the boring answer of, it depends, but that's often the answer to lots of technology questions. So my general advice is if it's not broken, don't fix it. If you've got an existing Pandas project that works absolutely fine for you, then I think there's probably better things for you to focus on than rewriting it in Polars. But if you're starting a new data science project, then that's when I typically recommend people, "Okay, this is a good time to give Polars a go."

00:09:08 I think if you start a new project and you try to think in Polars right from the start, you'll end up writing idiomatic code and you'll have a lot of fun. Something a lot of Polars users say is that it's surprisingly pleasant to write Polars code and it's nice to see what the library does for you. The syntax is very nice. I think that's one of the major APIs, major innovations that the library has brought aside from just a phenomenally good implementation.

Jon Krohn: 00:09:38 I would've maybe assumed that the API, that the syntax would be similar to Pandas, but actually what you're saying is it's quite different.

Marco Gorelli: 00:09:48 That's right. Yeah. So the idea of trying to reimplement the Pandas API but with a different faster backend and all of that, it's been tried with varying degrees of success with Polars. I think this is a really nice success story. Ritchie just had the courage to try something different to say, "Well Pandas, it's successful, it's popular, it does what it does. Let's try doing something different. Let's try not having row labels. Let's just not have an index."

00:10:19 I think any of your listeners who are familiar with Pandas, most of them are probably used to having to do reset index every two or three lines of Pandas code in order to get things to work. There are Pandas users who use the index very intentionally and they can make great use of it. You can get performance improvements from using the index very intentionally, but I think the majority of Pandas users, for them, it's probably more of an annoyance than anything else. And so I think Polars has really made a good design decision here. Most users don't need to worry about their rows having labels. A side effect of this is that it makes certain performance optimizations easier and the company is now working on distributing Polars.

Jon Krohn: 00:11:11 The company, Quansight?

Marco Gorelli: 00:11:13 Sorry. The Polars company.

Jon Krohn: 00:11:14 The Polars company?

Marco Gorelli: 00:11:15 Yeah, exactly. So when it comes to distributing Polars, then it should be easier to do that if you don't have to worry about having an index. Whereas companies that

have tried distributing Pandas, like Dask, they do have an index, but it does cause some difficulties.

Jon Krohn: 00:11:33 I see, I see. So there is a Polars company that is commercializing the Polars open source library that anybody can access and install?

Marco Gorelli: 00:11:42 Yeah, that's right. So there's a company that's behind the open source software. Most of the core developers are hired by the company. The open-source software Polars is and always will be open source according to Ritchie. However, they're also going to make some other offerings, like a cloud offering, distributed. These are things that are going to be paid services and that's what the company is working on.

Jon Krohn: 00:12:15 Makes perfect sense. Hopefully this isn't a controversial question, but Quansight Labs knows that you spend a fair bit of your time on Polars project and we have some more questions for you later in the episode on how Quansight supports their employees splitting time between consulting and open source development. Do you know why that works so well? Do you know why you get so much support on developing Polars?

Marco Gorelli: 00:12:42 Well, I've brought into the company some clients who have wanted training in Polars, both for teaching their employees how to use Polars and teaching their employees some more advanced tricks like how to extend Polars with Rust plugins. We've also had some clients who've specifically wanted help with solutions that have heavily leveraged Polars. So for the company it works well to say that they've got somebody who's invested in contributing to Polars who can help clients, and it works for me if I can do a bit of both. Really happy to have this balance at the moment.

Jon Krohn: 00:13:23 Are you stuck between optimizing latency and lowering your inference costs as you build your generative AI applications? Find out why more ML developers are moving toward AWS Trainium and Inferentia to build and serve their large language models. You can save up to 50% on training costs with AWS Trainium chips and up to 40% on inference costs with AWS Inferentia chips. Trainium and Inferentia will help you achieve higher performance, lower costs, and be more sustainable. Check out the links in the show notes to learn more. All right, now back to our show.

00:14:00 Awesome. Another aspect of Polars that I understand, so you've mostly so far been talking about Polars being a great choice for people who want to be manipulating dataframes and have more fun, have an easier time with the syntax relative to what they might impede on. But you've previously, on another interview, you described Polars expressions as functions that only take effect once you put them inside the dataframe context. Can you provide an example of how this lazy evaluation benefits data processing and any maybe concerns people should be concerned about as users when they do evaluate in this way?

Marco Gorelli: 00:14:37 Oh, that's fantastic. Yeah. Expression is really one of Polars innovations. I don't think it's something that Polars invented. PySpark had something similar, some, our libraries I think had something similar, but the way they work in Polars, I think of an expression as a function from a dataframe to a sequence of series. Most users don't think of it in these terms. Most users just think of it as grabbing a column from a dataframe and then doing some operation on it. People usually get an intuition for what expressions do fairly quickly in terms of what advantages, apart from just how nice the syntax is to manipulate. The fact that an expression is just a function, so it doesn't need to be evaluated right away.

00:15:31 It means that when you've got the dataframe context, Polars can analyze the different expressions which you've passed in and they can apply certain optimizations. For example, the classic example that Ritchie gives is if you are taking a column and doing a sort and then selecting the first five elements, then this has got N log N complexity, but you could just do a top K algorithm, and then the complexity there should be linear, I think something like that.

00:16:03 Another example is you might be doing feature engineering, you might be making two features which both start with something very similar. I don't know, take the absolute value of the logarithm of something and then one feature you're doing like shift one in the other feature you're doing shift two. People are often making features where part of the calculation is very similar. So then Polars can do common subplan elimination. It can see that some parts of the expressions are very similar. It can just assign that to a temporary variable, just calculate that once and then reuse that between the different features.

00:16:43 Another advantage of using expressions in DataFrames is that it lends itself very nicely to parallelization. So if you're just making a single operation on a single column, then it's often just not worth it to set up the overhead of doing multi-threading. But if you're calculating, let's say five different features which are independent of each other, then it's quite natural to say, "Okay, we'll do these five in parallel." People can often get 10, 20, a 100x improvements by writing things in Polars compared to what they might've got with some other frameworks.

Jon Krohn: 00:17:27 Wow. That 10, 50, a 100x, that includes the parallelization?

Marco Gorelli: 00:17:33 Including everything. Including parallelization, including query optimization that we get from doing things lazily, just the whole package. It's going to give you quite a significant advantage both in terms of runtime and in terms of memory.

Jon Krohn: 00:17:50 Nice. And let me try to break down that lazy term a bit for listeners who might not know it and maybe in the context of what you just said. So if the code that I was working with was working in an unlazy way, which could be a Pandas dataframe, and if I have a Pandas dataframe with only a hundred rows or a thousand rows and I want to do a sort like you described before, I take the top five after a sort. With only a hundred rows or a thousand rows in my dataframe, in real time I'm not going to notice any problems with that kind of evaluation. But if I have a million rows or a billion rows, then that Pandas dataframe, I'm going to be just sitting there for who knows how long while I'm waiting for that sort to actively execute. But with this kind of lazy evaluation that is supported by Polars behind the scenes, so it doesn't actually execute the code until I ask for some kind of output.

00:19:00 And when I ask for that output, there's lots of performance optimizations behind the scenes like you described in much better detail than I could. But the net effect is that it means that if I need that sort on a huge dataframe to happen because it's not actively executed in a more simple-minded way. It's lazily executed in a more clever way, and so lazy meaning that it doesn't execute until it has to. Because of that lazy doesn't execute until it has to, performance optimized behind the scene execution, you get these huge speedups like you described with the sort scenario. To use computer science terminology, it was a linear increase in compute as your dataframe gets larger as opposed to N log N, which is much more, much more, much more computationally

expensive when things get larger. Did I do an all right job of trying to recap what you said there?

Marco Gorelli: 00:20:01 Yeah, totally. I think you got the spirit of it perfectly.

Jon Krohn: 00:20:05 Nice. All right, so another aspect of Polars that allows it to differ from other libraries is that it optimizes string operations and data processing in particular. Do you want to talk about that?

Marco Gorelli: 00:20:20 Sure. Right. We need to make a little Pandas and NumPy comparison here. So we need to go back in history a bit. Pandas originally built on top of NumPy. NumPy has not traditionally had a string data type. They do since NumPy version two, but traditionally if you wanted to store strings and maybe they're of different lengths and all of that, you're going to have to just use an object data type in NumPy. So in object data type, every element is just a pointer to a string and that comes with all kinds of performance and memory footguns. So that's the historical part then.

00:21:12 In Pandas, this has traditionally been a bit of a weak point. I think since Pandas 1.5, it's been possible to leverage PyArrow to use a specialized string storage. So how that works is there's a really long string behind the scenes and for each string in your series, Pandas is recording where the string for that particular row starts and where it ends and like this, it ends up with better performance and memory characteristics compared to just using the classic object data type NumPy ones. Polars have taken it even further and they've got a whole different kind of string. They've written a whole blog post about this and that enables further optimizations, especially if you've got repeated strings. So that's the deal. Polars makes working with strings really nice. It also just does this natively. You don't need PyArrow installed in order to make use of Polars strings.

Jon Krohn: 00:22:29 And I guess this is of increasing and increasing importance with how natural language processing is becoming more and more and more of data science. So there was a time when Travis Oliphant, who we talked about at the outset of the episode, when he would've created NumPy and SyPy, almost everyone who was using Python, I don't have the stats on this, but just based on my experience and seeing what was happening out there, most of the time you're working with tabular data. And those tabular data, by and large, they were numeric. I mean for sure with NumPy, and Pandas was designed to go a bit beyond that and be able to handle lots of different data types in one matrix structure where you have one column that strings, one column that's numbers and so on.

00:23:14 So more like working with the kind of data that are in a spreadsheet that you might have in Excel. But we are now in this era of data science where natural language processing capabilities are so profound thanks to things like large language models, transformers, generative AI, we have so much more interest in natural language processing than ever before. And so it seems to me like having these string optimizations will come in handy.

Marco Gorelli: 00:23:46 Yeah, definitely. I mean, even if you're not working in NLP, if you're working in traditional data science, you're probably working with some columns which are strings, like maybe you've got a column which tells you the name of your vendor or the name of your supplier and all of that. You can see the difference that this makes with the TPC-H queries. So this is a set of popular database benchmarks. It's originally written for SQL engines, but it's been adapted to dataframes and you can see the difference of running those in Pandas, just a classic data types. And then in Pandas where the only difference you make is to use PyArrow strings instead of the classic object data type. And typically most queries get about

twice as fast, even though in those queries you're not doing anything string-specific. Just doing a join that includes string columns, even if you're just comparing two columns for equality, any operation where strings are there in the middle, it benefits from this.

Jon Krohn: 00:24:47 Very cool. So again, you talked about how you can get big performance improvements. You talked about 2x just now, even in situations where there aren't string operations. You talked not long ago in the episode about a 10x, 50x, or even a 100x speed up in some situations thanks to the lazy execution, other optimizations that exist in folders. Do you happen to have any kind of specific case studies that come to mind for you where these biggest really big performance changes happened? So where you're working with large data sets and you leverage Polars and Rust to get a huge performance improvement relative to if somebody was using say Pandas and NumPy.

Marco Gorelli: 00:25:34 You're in luck. I do.

Jon Krohn: 00:25:38 I didn't prepare him for these questions, so that sounded like it was almost like cheesy and teed up.

Marco Gorelli: 00:25:45 No, yeah, I got a case study that's really in my mind for this. Was recently working with a client who had lots of data that they wanted to geocode. Maybe we should explain to the listeners the meaning of geocoding reverse geocoding. So geocoding is the practice of when you're given an address, you need to determine what's the latitude and longitude of it. And then reverse geocoding, that's the opposite. You're given some latitude and longitude and you want to work backwards and get what's the closest address. So this is something that's used in a variety of sectors from trying to identify landmarks to advertisements. Lots of industries are interested in doing these kinds of operations and typically

the way you do it is with really big data sets. You've got big lookup data sets with lots of addresses, you're trying to match things. And how can you do this quickly?

00:26:40 In particular, this client was quite interested in doing this in a way that could save them money and they were really interested in seeing how far can we get, let's say on a single node or even on AWS Lambda, and we found that actually we could do the entire thing on AWS Lambda. AWS Lambda is a very constrained computing environment. There you've got maximum 15 minutes to complete a job. You've got maximum 250 megabytes for your package size. So that means that installing, let's say the newest versions of Pandas, PyArrow and NumPy all together, it just wouldn't fit.

00:27:19 However we found, well Polars is fairly lightweight. We didn't need PyArrow, Pandas or NumPy for this task. We can make it work. So on the packaging side, we also needed some Rust extensions and I think that's not super easy to get into AWS Lambda, but here this allows me to talk about one of Polars superpowers and that is that you can extend it. You can make Polars plugins, you can write your own Rust extensions for Polars, which you can then distribute onto PyPI and people can just PIP install them as they would any other Python package, and then it just fits in naturally with the rest of your Polars workflow as if it was part of Polars itself. So for this client, we wrote one new Polars plugin for this task. We leveraged a couple of Polars plugins, which the community had already built, which happened to perfectly suit our use case.

00:28:17 And this Polars, I think we were using three Polars plugins, then Boto3, S3FS, some other packages which are just used in AWS for cloud computing and we could fit all of this easily within the limit. Now comes the data constraints because there in AWS Lambda, I think you've got a limit of 10 gigabytes of RAM, but we needed to query

hundreds of gigabytes of data. This is where lazy execution really helps, in particular lazy execution with Polars plugins. So we could say scan all of this data and then use the plugins to determine which rows need reading and only read those so then Polars knows which parts of the data set of all of these hundreds of gigabytes it needs to read based on Rust extensions, which really customized things which would written and in the end it could all fit within the memory and time constraints. I think this could be possible using other technologies, but I am pretty confident that it would not have been so easy. Polars really made it easy for us.

Jon Krohn: 00:29:31 If you're a regular listener, you know that last year I did a European podcast tour interviewing incredible guests in Amsterdam, Paris and Berlin. While all the guests spoke perfect English, Babbel was invaluable for me to learn and practice Dutch, French and German enabling me to get directions and order my meals in the local language. Super fun, rewarding and in some cases an essential skill. Now you can do the same with a special limited time deal. Right now get up to 60% off your Babbel subscription, but only for our listeners at babbel.com slash superdata. Get up to 60% off at babbel.com slash super data spelled B-A-B-B-E-L dot com slash superdata. Rules and restrictions may apply.

00:30:18 That's a very cool case study. To summarize back some of the key points from that, you talked about, you had mentioned actually already earlier in the episode how one of the advantages of Polars is that it can be extended with Rust, but now we got a sense of what that really means. And so these Rust extensions become add-ons that you can very easily install just for the PyPI call, just like if you were bringing in Pandas or Polars. And so, very, very easy to do that. And those Rust add-ons, they can be customized for tasks like going over hundreds of gigabytes of data, identifying relevant rows and then

allowing you therefore to have even running in that highly constrained AWS Lambda environments, which is a small amount of code you're able to execute highly efficiently. It's a really cool case study. Did I get that right?

Marco Gorelli: 00:31:15 Yeah, yeah, absolutely. I like the summarizing. Listen to the last episode on Super Communicators for more on that.

Jon Krohn: 00:31:21 It's true. Yeah, the preceding episode before I was recording with Marco today was episode number 805 with Charles Duhigg. Charles Duhigg is a Pulitzer prize winning journalist. He's a many bestselling authors, a many time bestselling author. He's not multiple authors, not that I'm aware of. I don't think he has a secret pseudonym. But his most recent book is called Super Communicators and we talked about it a fair bit in that episode 805. And it's interesting, because I don't usually have mainstream authors on the show. We have sometimes data storytelling experts like Cole Nussbaumer Knaflic, but those are people who even though her data storytelling book was a huge bestseller, she comes from a technical data science background. Charles wasn't like that, but he was just such a big mainstream author that when he approached me on the show I was like, "Sure, let's do it." Because who can't benefit from knowing more about communication?

Marco Gorelli: 00:32:26 Yeah, exactly. It was a great episode. I'm sure lots of people enjoyed it. On the Rust extensions though, something I'd like to clarify is that it's not as scary as it sounds. I'm sure if we could see people's faces from the audience, we'd be seeing some blank stares. When I've talked about Polars plugins at conferences, that's often what happens. People think, "Who is this guy and why is he expecting us to write a Rust extensions?"

	00:32:51	And my claim is it's not that difficult to make a Polars plugin. Polars has some really complicated Rust code inside it. But the Polars plugins mechanism is that complexity is abstracted away enough that if you can express your business logic in Python, it's not that much of a stretch if you just know the basics of Rust to translate that into a Polars plugin. I've got a Polars plugins tutorial online. If you just search a Marco Gorelli Polars plugins, I'd like to think you can find it. Freely available resource and it can teach you how you coming from a Python background can learn just enough Rust to write your Polars plugins. It covers how to distribute them, different data types, some performance tips.
Jon Krohn:	00:33:47	I'm sure we'll include that in the show notes. I don't talk about her enough on air, but she also deserves praise. I frequently actually come on and talk about our researcher Serg Masis who, like you are already experiencing, Marco, today, has done incredible research digging out lots of great topic areas and specific technical questions, which is super helpful. But someone else that is invaluable on the show is our podcast manager Ivana, who goes through and anytime we mention things like this, the guest mentioned some blog post of theirs or some tutorial like you just mentioned that they can download and she goes and makes sure that it's there for you in the show notes, keeping everything organized and on time. And that's how we get 104 episodes a year all released on time for many, many years in a row thanks to her. So anyway.
Marco Gorelli:	00:34:33	Nice one, Ivana.
Jon Krohn:	00:34:34	Yeah, exactly. So back to the amazing Serg topic flow and the kinds of questions that he has covered. You mentioned geocoding in your last example, and we didn't actually even really get back to how that was... After you've kind of gone through the performance

optimizations, what was the net effect for the geocoding? Maybe we should draw a line under that quickly.

Marco Gorelli: 00:34:58 Yeah, sure. The net effect was that the client was able to go from having to make expensive API calls from having to run something on a cluster with multiple nodes to just being able to run something in the most constrained possible computing environment. So for them it was a saving in terms of time, in terms of compute resources, in terms of maintainability. They were really happy with the solution.

Jon Krohn: 00:35:24 Very cool. And so when I think about geocoding, it seems vaguely related. You maybe like me travel around the world a lot. This idea of moving around, something that moves around as we move around is the time zones. That's a huge pain not only for our body clocks, but also for anybody who's developing software and having to manage over many time zones. So you've cautioned previously against manually managing time zones. So you did this in a conference talk. Could you elaborate on the challenges that you faced when trying to manually handle time zones and why software like Polars is a more reliable solution?

Marco Gorelli: 00:36:10 Sure. I remember seeing a colleague trying to manually put in a if-then statement to deal with daylight savings time. Yeah, that's a bad idea. Chances are you won't be able to get it right. You won't get the direction. If you then need to communicate with people in different countries, like in the US they observe daylight savings time at a different time than we do with the UK.

Jon Krohn: 00:36:34 I think Arizona doesn't observe daylight savings time at all.

Marco Gorelli: 00:36:37 Oh, right. Yeah. Lots of countries don't observe it at all. I think in Morocco they do, but they go in the opposite

direction. Time zones are a mess. And then you find that countries have changed time zone offsets. Maybe, like in the UK we are a plus zero most of the year, sometimes plus one. Some countries at some point in time they decided, "Yeah, we're going to go from minus 13 to plus 11, that's it." And you might think that it's manageable to do this by hand, but it's really not. You want to leverage a third-party library to do this. So Polars has full support for time zones in the sense that any operation which is time aware you can do it, respecting time zones. My advice with time zones is you should avoid them if you can. They're an absolute mess and they also come with a bit of a performance hit. It's just a necessity.

00:37:34 Unfortunately, you can't necessarily avoid time zones. If your boss is asking you for a daily sales or something and your company is selling at every hour of the day, then you better do it in a time zone-aware manner. If you take the trick which some people suggest of convert to UTC, do your analysis and then convert back, then you're going to miss subtleties due to daylight savings time and other things. So just convert to UTC and back might be a solution in some cases, but definitely not in all cases, which is why it's important for software like Polars to have good support for time zones. Personally, one benefit that I got from getting involved in time zones is that it led me to get involved in Polars. So side note, open source tip. Sometimes people ask, "How do you get involved in open source? How do you start contributing to something like Polars that looks so complicated?"

00:38:29 And my advice is if you've got some topic that's valuable and that's interesting to you but is boring to other people, then that's your competitive advantage. When I started contributing to Polars, I noticed that a lot of the time zone stuff just hadn't been done. The other maintainers just didn't find it very interesting or found it frustrating and all of that. And I was like, "Okay, well I don't know Rust,

maybe this can be a bit of a win-win situation. I'll help you with your time zones, you help me with my Rust." And yeah, it worked. Just started fixing stuff up, learned a lot about Rust in the meantime and got totally addicted to the process. So I need to pair that word of caution with my open source tip.

00:39:17 You might get started with something, you might find that people are appreciating what you do, but it's very difficult to then not get addicted to it. I call it a legalized drug, just something to keep in mind. A lot of people who then do a lot of open source, they end up doing a lot of it in their spare time and it's not so easy to draw a balance. I can't offer very good advice when it comes to drawing a balance between life and open source because I'm not there yet, but working on it.

Jon Krohn: 00:39:47 Yeah, I understand what that can be like. It's definitely not the same thing. So I have not done much open source contributing at all. It's been many years since I've done any. I mean, I open source Python scripts in Jupyter notebooks when I do tutorials. So for YouTube videos that I make on machine learning foundations or introductory deep learning tutorials, I do open source my code and sometimes people make issues and then I resolve them or there's some small amount of collaboration. But that doesn't really feel to me like the kind of open source collaboration that you're describing if you're working on Rust or Polars and it's this big ecosystem with I imagine hundreds of contributors and everybody has their own little piece and everything needs to work together and execute properly. And so I haven't done that kind of stuff to a significant extent.

Marco Gorelli: 00:40:47 I mean that is really enabled by all of the modern tools we have, like GitHub, all of the CI minutes that GitHub gives us. Otherwise, it'd be so difficult to coordinate between hundreds of contributors. What I find the hardest is the

people part, like API decisions. This part is really difficult. Something I noticed recently in Pandas is there's a function which does not behave as its docstring says it does, and it's been like that since at least 2019. And now what do we do? Do we correct it? But then it's going to break people's code who are relying on it? Do we update the docstring? But then the behavior that it does have seems rather odd. It is just so difficult to make that kind of decision. Whatever you do, you're going to anger some people. Yeah, that's the hardest part of open source. In the end, technical issues are relatively easy compared to some of the people ones.

Jon Krohn: 00:41:44 Yeah, version issues are a pain for sure. So anyway, we kind of digressed over here from, we were talking about time zones and you got talking about how your interest in time zones, it was a stepping stone for you to learn about Rust but also to contribute to the Polars time zone functionality. And so it was a win-win. Then since then you've been addicted to open source contribution.

Marco Gorelli: 00:42:11 Something like that. Yeah.

Jon Krohn: 00:42:14 What is it about the way that Polars handles time zones now that you've been working on it that is, how does that look and feel for me differently as a Polars user?

Marco Gorelli: 00:42:25 Compared to before I started contributing?

Jon Krohn: 00:42:27 Or compared to other alternatives out there?

Marco Gorelli: 00:42:30 Well, yeah. Compared to other alternatives that work, I'd like to think you shouldn't notice much of a difference. Compared to before I started contributing, the difference is that the time zone is typically taken into account when you're doing calculations. So remember some of the early bugs that I would see was something like if I tried to calculate the daily average, then the daily average is done

on UTC time rather than on the local time. It's like, okay, yeah, we need to fix that. And then it's like, oh, but if I pass this data then it just errors because of an ambiguous date time, there should be a way to resolve that ambiguous date times. So when we do daylight savings, we shift the clock back at one point of the year and we shift the clock forwards at another part of the year. So when we turn the clock back, then we're essentially repeating the same times multiple times. I am using the word times but to mean different things. Sorry about that.

Jon Krohn: 00:43:34 Yeah, exactly. When the clock goes backward, for example, you end up having, I think it switches at 2:00 AM or-

Marco Gorelli: 00:43:44 Yeah, exactly. So 2:30 or something. It's going to happen twice.

Jon Krohn: 00:43:47 Twice in one night.

Marco Gorelli: 00:43:48 And if you're trying to pass a string which contains 2:30 on the 25th of October 2020, how do you know which 2:30 it refers to? Must be an absolute nightmare being a policeman and having to go through reports where people are trying to reconstruct what happened anyway. Anyway, if you're doing this in Polars, there needs to be a way to deal with that. So I introduced an ambiguous argument to the date, time function similar to what there is in Pandas, and it at least gives you a way to deal with it.

Jon Krohn: 00:44:18 In a recent episode of this podcast, the mathematical optimization guru Jerry Yurchisin joined us to detail how you can leverage mathematical optimization to drive commercial decision-making, giving you the confidence to deliver provably optimal decisions. This is where Gurobi Optimization comes into play. Trusted by most of the world's leading enterprises, Gurobi's cutting-edge

optimization solver, lightweight APIs, and flexible deployment simplify the data-to-decision journey. And, thankfully, if you're new to mathematical optimization approaches, Gurobi offers a wealth of resources for data scientists, including hands-on training, comprehensive Jupyter-notebook examples, and extensive, free online courses. Check out Episode #813 of this podcast to learn more about mathematical optimization and all of these great resources from Gurobi. That's Episode #813.

00:45:08 Very cool. So Polars might be something that people have heard about for the first time from this episode. While even in my introduction to Polars, I assumed that pretty much anyone who's a hands-on data scientist knows Pandas is very likely even used Pandas. So with this rapid development of Polars, where do you see it heading? It's blossoming in popularity. I hear people talking about it more and more. I think that's a big part of why I was like, "We've got to get Marco on and have a Polars episode." We haven't talked about that yet and I feel like it's something everyone needs to know about. So where do you think Polars is heading? Where do you think it's going next in its evolution? I don't don't know if I've kind of teed you up enough with this question. You might already have some thoughts on how to answer.

Marco Gorelli: 00:45:59 Sure. So I think there's two parts of Polars we need to talk about. One is the implementation itself and the other is the API. The implementation follows its own API of course, but the API can take on a bit of a life of its own, just like the Pandas API took on a bit of a life of its own. So Pandas follows the Pandas API, but then we saw Modin come along, which also follows the Pandas API, and then FireDucks and QDF, and now we're seeing that Polars might be going in a similar direction. Modin is a dataframe library which historically has promised to distribute your Pandas code and now they're also offering a Polars API. Now they haven't released details of what's

going on under the hood with regards to the engine, but the fact that that Polars become popular enough that they're like, "Okay, we need to do something with this."

00:46:53 It's a good sign. We're seeing that Nvidia are contributing GPU support to Polars. So earlier you were talking about how when you've got lazy execution, at some point you want to actually see the results. You need to tell Polars that you want to see the results. What the team is working towards is the ability that when you tell Polars you want to see the results, you can tell it, "Compute this for me, but do it on GPU." So then you're making use of both GPU acceleration and query optimization. I don't think the world is ready for such levels of speed.

00:47:29 So in terms of where are we going, I think that the library itself is going to grow, but I think the Polars API, I'd like to see it become a bit of a dataframe standard. I'd like to think that when people make new dataframe libraries and there will be new dataframe libraries. I don't think Polars is the last one. I'd like to think that their API will be much more similar to the Polars one than to the Pandas one, which has dominated the dataframe API space in Python up until very recently.

Jon Krohn: 00:48:00 Very cool. It's nice to hear your insights into what's happening next and these interplay between different libraries and technologies facilitated by say with Modin, facilitating broader distribution, with Nvidia supporting execution on GPUs. Something that must be very near and dear to your heart, or at least to your addiction is another open source project that you created that allows for compatibility. This is Narwhals. So last year you described Narwhals as an extremely lightweight compatibility layer between Pandas and Polars. So what is the problem that you're addressing there with your Narwhals library?

Marco Gorelli: 00:48:52 Sure, thanks for asking about it. It's still slightly cracks me up that we've got a library called Narwhals that we're actually talking about. It started off as a little weekend project and I was feeling a bit silly, so I called it after a viral Mr. Weebl song, but now it's being adopted by some people. So I guess we're stuck with the name. Maybe before I go in too deep, just a slight correction. I think this year I described it like that. I only released it back in February. It's a very young project, but it's quickly gaining a bit of traction.

Jon Krohn: 00:49:20 Yeah, I'm sure you're right. That must be a rare research error.

Marco Gorelli: 00:49:26 Maybe not a research error, maybe confusion with another similar project. Before Narwhals, I was involved in a group called the dataframe Consortium, which was trying to make a DataFrame standard, like some dataframe API that different dataframe libraries could implement and then people could write dataframe agnostic code on top of. It's difficult to get different people to agree and the stakes here were pretty high. I just wasn't able to agree with most people there. I found myself disagreeing with most of the participants about nearly everything. I wanted to bring things decidedly towards Polars. They wanted things to be not exactly like Pandas, but they didn't want things to deviate too much from Pandas. They didn't want things to deviate too much from what most people were familiar with and from what would be difficult for them to implement. So in the end, after having agreed to disagree, I said, "Well, let's take all of these ideas which the consortium had rejected and let's package them as its own thing. Let's call it Narwhals and let's see what happens."

00:50:31 And the idea is it's like what the dataframe standard was trying to be. So just some API which different backends can implement and which a library can then use to just

define its transformations, to just define its dataframe logic. And then the user can bring their own dataframe, pass their own dataframe in, and they can just use it seamlessly as if that library was written specifically for their dataframe. So someone comes along with Pandas, they can just use it. Someone else comes along with Polars, they can just use it. The Pandas user doesn't need to have Polars installed and the Polars user doesn't need to have Pandas installed. This is what we're aiming to enable. What kind of surprised me was that interest in it happened a lot faster than I was thinking.

00:51:18 So within about a month or two we had a scikit-lego, which is like a medium-sized library for some extra things which don't quite fit into scikit-learn. They decided to adopt it, which aside from the fact that it's kind of nice as a Polars user to be able to use a library without having to convert the Pandas, it also made a massive performance difference in some cases. So Polars, because of the reasons we described at the beginning of the episode, it really excels at feature engineering can do things in parallel. It can do common sub-planet elimination. And so for the feature engineering functions in scikit-lego, doing it directly in Polars as opposed to having to convert to Pandas, doing the operation in Pandas and then converting back to Polars. It can make, I had one benchmark where I was seeing even a close to 150x speed up.

00:52:17 It was really quite massive. So yeah, I was pretty happy with that. And then what made me fall off my seat just a couple of weeks ago was the major visualization library, Altair have adopted Narwhals. And so, like this that made NumPy optional, Pandas optional, PyArrow optional. I think PyArrow might've been optional from the start, but if you were trying to plot a Polars library, you were required to have PyArrow installed, whereas now you could just pass a Polars dataframe to Altair. You don't

need NumPy, don't need Pandas, don't need PyArrow, and it'll just plot it natively. So for Polars users, they just need this very lightweight library and they can make beautiful, possibly interactive plots, and this is exactly what I was hoping to enable with Narwhals, just a better adoption for Polars and other newer dataframe libraries at no cost to their existing Pandas users.

Jon Krohn: 00:53:12 Very cool. Let's talk about actually Altair for a second year because that is a library. It's a Python library?

Marco Gorelli: 00:53:22 It's a Python library, yes.

Jon Krohn: 00:53:25 I'm used to thinking about really the only two, well, okay, maybe I can think of three plotting libraries off the top of my head. Obviously, Matplotlib. Seaborn, which has been popular for years is a slightly prettier... Because Matplotlib is, it's just with all of the base, if you just stick with all the basic pre-installed configurations, you end up with pretty unattractive plots with quite abrupt colors next to each other, whereas Seaborn out of the box creates beautiful plots. The other library I could think of off the top of my head for plotting is Plotly, and I actually can't off the top of my head, remember why I would use Plotly.

Marco Gorelli: 00:54:12 Plotly is nice. Yeah, it makes really nice interactive plots for you pretty much out of the box.

Jon Krohn: 00:54:16 Interactive plots.

Marco Gorelli: 00:54:17 Yeah, I really recommend that one. Yeah, Seaborn's nice as well. Yeah, as you say, it's like a wraparound Matplotlib. With Matplotlib can do anything. I think practically, literally anything, things you just didn't even know were possible. You'll find some answer on Stack Overflow where someone has given you an answer. But yeah, not super user-friendly and Seaborn makes it a lot

easier. My hope is that maybe if Narwhals can become more popular, maybe Seaborn can trust us enough that we can rewrite Seaborn to be Narwhalified and people can pass Polars dataframes into Seaborn. Not currently possible, but hey, let's see.

Jon Krohn: 00:54:56 Yeah, yeah, very cool. I've said very cool way too much in this episode, but I thought that a lot of the things that you've said are very cool. Hopefully my transitions have become a little bit more nuanced after I say that phrase. But with Altair specifically, this is a library that I've only just started to hear people talking about, but it is widely used. And so, why should a listener think about picking up Altair and using that library for plotting interactively?

Marco Gorelli: 00:55:26 I think the API is really nice and consistent and it just makes sense in your head, at least the way that I would think about making plots. They've got a nice grammar. There is a bit of a learning curve. You need to learn these rules, you need to learn about channels and marks, but once you get it, you can make plots and you can make them look nice and you can plot what you want. I think it might not be quite as highly customizable as something like Matplotlib. So if you need to make really super highly customized plots, then maybe it's not the perfect solution. But I think for most data scientists who need to tell stories with their plots who need to understand data, I think it's a really good solution.

Jon Krohn: 00:56:19 This Narwhals project, which you only started on a year ago, it sounds like-

Marco Gorelli: 00:56:23 February.

Jon Krohn: 00:56:24 You only started-

Marco Gorelli: 00:56:25 Less than a year.

Jon Krohn: 00:56:26 Right, right. It already has been picked up by places like Altair. It's already making a big impact. Part of what's so impressive about it is its minimal overhead and its lack of dependencies in its design. How hard or easy was it to get that level of efficiency? Were there some big technical challenges that you faced while developing?

Marco Gorelli: 00:56:51 Sure. So on the dependency side, I don't think it's that difficult. I think it's just a matter of willpower. So if you want to keep your library dependancy free, I think it's usually not that much of a stretch in this case. Suppose that you are a library which receives a dataframe from the user, and you want to know whether it's a Pandas dataframe. The obvious solution is to try importing Pandas, and if that succeeds, you check if it's a Pandas dataframe. But we can actually do better than that because if somebody has passed us a Pandas dataframe, it means that they must have already imported Pandas. So we can just do import sys, we can check all of the libraries that the user has already imported in sys.modules and see if Pandas is in there. And if it's not, then obviously this cannot be a Pandas dataframe, so we don't even need to try importing Pandas.

00:57:42 So what we're very strict about in Narwhals is we don't import anything like this. We don't risk introducing dependencies and we don't risk slowing things down by forcing people to import things, which the object just isn't. So that's a part of it. The other part of it is the overhead. That, I think isn't so immediate because we're translating syntax. The key there is to just have a good mental model of what Polars expressions are. And to me, an expression is just a function from a dataframe to a sequence of series. Once you define it that way, then chaining expressions together, chaining these calls, it's just a matter of chaining Lambda functions, one after the other. You can just need to be very rigorous about recursively applying this definition everywhere, and it all

just kind of happens. There is potential for overhead in the sense that Pandas does a lot of things with the index, which you don't necessarily want.

00:58:49 Pandas is always aligning indices with each other, but Polars doesn't have a notion of an index. So if you want to make Pandas behavior mirror the Polars one with the same API, you need to be careful to avoid automated index realignment. The naive solution to that would be to do reset index all the time, which is in fact, what we see in a lot of users code. Reset index is not a free operation though. So what we do in Narwhals is that the API itself means that when you're comparing columns, they typically are derived from the same dataframe just because of how the expressions API works. So we just do a quick check of whether the index of the left-hand side is the same as the index of the right-hand side.

00:59:35 We don't even compare the values, we just check, "Left index is right index." If it is, then we leave it alone. And if it's not, we set the right-hand side's index to be the left-hand side's index. And what I've observed empirically then is that compared to the naive way of writing Pandas code, we can often make things a little bit faster, which although I need to caveat that. So in Pandas version three, copy on write will become the default. This is an optimization. So once that becomes the default, then writing via Narwhals or writing Pandas code directly shouldn't make a difference.

01:00:14 Before that, I've noticed that Narwhals will often make things a bit faster unless you're dealing with a 100 row dataframe. If you've got something so small, then the overhead of just the extra Python calls within Narwhals is not... It is going to be detectable. You're going to have an extra half a millisecond there. So if you need to write a reactive web application using dataframes, yeah, maybe just use the dataframe library directly. Don't use

Narwhals for anything more than a hundred rows for anything where half a millisecond of overhead is tolerable. Then I think I'd like you to think it's a good solution. And for Polars users especially, that's half a millisecond of overhead. Compared to the overhead of converting to Pandas, it's nothing.

Jon Krohn: 01:01:00 Very cool. Great summary point there. Very cool again. You're getting from me. You have expressed hope for a future where data science becomes dataframe agnostic. So could you explain for us what dataframe agnosticism is?

Marco Gorelli: 01:01:15 Sure. Yeah. Well, you very kindly introduced the topic earlier by bringing up Seaborn. Seaborn just takes the dataframe and then visualizes it. There's nothing about the logic of the library that should be tied to Pandas. So why is it that it only works on Pandas? It does accept other dataframe libraries, but if it receives anything else, the first thing it does is it converts it to Pandas and then it does everything else. There's no theoretical reason why that should be the case. I don't know. What does Seaborn do inside? It takes a column. It does a group by and it finds the sum. I think every dataframe library does that. Certainly every dataframe library that we're supporting in Narwhals. So I'd like to think that we can aim for a future where libraries such as Seaborn can just define their logic, and then the library can be dataframe agnostic.

01:02:10 So long as you are either supported by Narwhals or you comply with the Narwhals API, then your library can just slot in there. And the good thing about standards is that they really enable freedom. Because as much as I love open source, I'm not an open source absolutist. And the nice thing about having a standard out there, about having a Narwhals specification and its API is that someone can come along with their closed source solution and we don't need to know about it. As long as their

closed source solution respects the Narwhals API, then it'll work seamlessly without them having to ask us for permission to do anything, without them having to open source their library. I prefer it if people open source things, but as I said, I'm not an open source absolutist. And I think this is one thing that a standard specification like the Narwhals one can enable.

Jon Krohn: 01:03:04 That was a great summary to give us a sense of why the Narwhals project is so important and this idea of a dataframe agnostic future. And I could imagine that not even just with dataframes, there's probably lots of ways that in development in general, we could have more interoperability between libraries by thinking about commonalities and not having discrete silos of specific projects that are segregated from each other.

Marco Gorelli: 01:03:31 That's the thing. Yeah, it's not just the silos thing. That's an important thing. Lots of projects, they just develop their things, like in writing Narwhals, interacting with people from lots of different projects. It's a lot of fun. It's just a lot of fun to collaborate with communities from different projects. That was just unexpectedly positive benefit of this project and probably the part that I'm enjoying the most.

Jon Krohn: 01:03:58 Nice. So we've heard now a lot about specific open source projects. I've alluded to how Quansight Labs where you work has a hybrid employment model which balances time between community projects like open source developments and consulting work, which brings in revenue directly. So how does this model where you're splitting time between open source and consulting work benefit both the maintainers like yourself as well as commercial clients of Quansight Labs?

Marco Gorelli: 01:04:32 Sure. Well, you asked how it benefits maintainers first. So let's start with that. We often get started with open source

because we're excited about fixing things, we're excited about adding new features that we might want. But then what happens five years down the line to those features which you've added? Someone's going to have to keep them working, and the reality of open source is that most people make one or two contributions somewhere and then vanish. When it comes to sustaining things for a long time, it's fairly difficult to do this just on willpower, just with volunteer work alone. A lot of these open source projects have become so big, so widely used, they've practically become critical infrastructure and it's just not feasible for everything to be done by volunteers. So fortunately we've been seeing funding come in for open source projects. We've been seeing CZI, the Chan Zuckerberg Initiative.

01:05:33 We've been seeing NASA donate money to open source projects and lots of other companies. It's nice to see that the Python Software Foundation itself has been able to hire, I think even two people to work on Python development as opposed to just being volunteers. So in terms of how it helps maintain us to receive some money, it means that for a lot of the tasks which you just would not be able to do as a volunteer, you can do them. Some big picture things like totally reworking how something functions in Pandas, as a volunteer, if you've just got a couple of hours each Sunday to do that, you're not going to have a chance to do that. You can maybe work on some incremental improvements, but you can't rework how something functions. But if you've got some funding behind it, it can work. When it comes to reviewing other people's pull requests, if you've got time, if you're paid to do that, you can do it.

01:06:30 People are often much more motivated to work on their own things than to review other people's. The other side though is that, yeah, Quansight Labs is not a charity. They don't just out of the goodness of their heart give

maintainers time to do things. It also helps the bottom line because there are companies that then know Quansight as experts in open source. So with open source maintenance, this also benefits Quansight itself. Companies are coming in for training for help with how to use software, but also sometimes with very bespoke features. So they maybe want, they're like, "I really want Pandas to support non-nano-circuit resolution. Can someone please do this for me?"

01:07:23 That was an actual request we got once, and it's not something that was completely delivered by Quansight, but Quansight really enabled it. If Quansight had not been part of the picture, I question whether that would've happened at all. Maybe it would've taken an extra three years for it to happen. So yeah, it's nice that at Quansight then we've got people involved in sales in marketing who know how money works as opposed to just being people coding in their spare time who don't necessarily have the skills or knowledge to deal with that.

Jon Krohn: 01:08:04 Makes perfect sense. And long may commercial supporters of open source work continue. I'm a huge fan of open source work. Most of the libraries, if not a hundred percent of the libraries that I've been programming with for more than a decade now. There was a time when I was using MATLAB.

Marco Gorelli: 01:08:25 I remember that. Yeah.

Jon Krohn: 01:08:28 But, and yeah, I don't think I've written a line of MATLAB code and over a decade. Since then it's been all open source programming for me all of the time. All of the training that I do is in open source. A huge amount of the code that we use at my software company, Nebula for developing our data science models for our whole backend front end of the platform, everything is open source, and so hugely grateful. You mentioned the Chan

Zuckerberg Initiative. Meta also obviously has been pouring huge amounts of capital into training and open sourcing large language models like the Llama series of models and all of these things make a big difference. They allow all of us as data science lovers, listeners to this podcast to be able to do more and make a big impact. So yeah, I hope that this trend continues.

01:09:26 It's great to hear behind the scenes how things work with Quansight Labs in particular, and maybe that will inspire some consultancy owners out there who are listening or other people to be thinking about how you can be supporting open source workers or open source projects in order to help your bottom line while also doing a service to the whole world. Now, a question here that our researcher Serge dug up is that according to a source that he found, only about 3 - 5% of open source contributors are women, which is a really low percentage. I don't know the exact percentage off the top of my head, but I know that the percentage of women say working in data science and in software development is much higher than 3 - 5%. And so, do you have any thoughts on proactive steps that people could take so that open source projects like Pandas, like Polars, like Narwhals, have more diversity than today?

Marco Gorelli: 01:10:34 Yeah, totally. Really important topic to talk about because you are right about these percentages not being aligned. Sometimes people explain it away by saying, "Oh yeah, but it's a pipeline problem. There's fewer women in tech, so obviously there's going to be fewer women contributing to open source," but then the percentage who do contribute is a lot lower than the percentage who are in tech. And there's a variety of reasons for that. I think we can't discount the fact that women do most of the unpaid labor in society, and if open source is a primarily leisure activity, then it means they've got less time for it. It's not the only reason though. I think things are getting better,

but there's a lot of projects where it does feel that, a bit like an old boys club, like the way maybe that people use humor, the kinds of things that people might say or discuss. There's a lot of things that have historically been tolerated that probably shouldn't have.

Jon Krohn: 01:11:37 Right. Locker room talk.

Marco Gorelli: 01:11:38 Yeah. Yeah, exactly. I mean, now it's quite rare to find a project that doesn't have a code of conduct, but we should remember that it was not always the case. It wasn't that long ago that just bringing up the question, "Should a project have a code of conduct?" would spark a whole load of controversy with people saying, "Oh, but it's not really necessary. We haven't had any harassment yet. Why do we need this?" And well, okay, just because you haven't seen it doesn't mean it's not happening. You did ask a specific question though, which is what can we do about it? That's a tough one. So first I'd like to slightly defer to Dr. Maren Westermann's talk at Euro SciPy last year in which she really talks about the importance of mentoring because it's not just about getting people to try contributing, it's about sustaining people.

01:12:33 I was involved for a while with PyLadies London trying to do some Panda sprints, trying to get people from underrepresented groups in tech to contribute. And these sessions were fairly well attended and people made contributions. We got a lot of people involved, but it's very difficult then to sustain people. It requires an active effort. I think unless you are actively going to set aside time and money towards mentoring people, it's very difficult. This becomes doubly difficult in a project which has already been going on for 15 years or something, and which has historically been all male.

01:13:14 At that point, to rectify that you're going to have to put in twice as much effort as if you were just starting from

scratch. Now in Pandas, I don't think we've got much hope of making a significant difference there, to be honest. I mean, we do now have one woman core developer. Realistically, the percentage of women is, I think it's unlikely to get close to the percentage of women in tech, at least without active efforts. And yeah, active efforts, take time, take money.

01:13:57 Unfortunately, Pandas didn't even receive CCI funding this year, so it's going to be tricky. With Narwhals starting from the beginning, we've been a lot more careful about this. I was messaging talented women that I knew saying, "Hey, maybe are you interested in trying out this project? I can help you out, provide quick reviews." And yeah, we've got lots of women who are contributing, who have given commit rights too. Going to do, probably going to take part in the Grace Hopper conference later this year, which is, I think it's primarily aimed at women. So with Narwhals from the start, we're just making this a priority and maybe we'll be able to do things differently. I don't know. We'll see.

Jon Krohn: 01:14:42 Nice. That sounds like a step in the right direction to me. It also sounded like you had some great tips in there for projects in general. So more mentorship would make a big difference. Your technique has been active reach outs to maybe... Because that kind of thing happens where if somebody like you who created this project taps you on the shoulder and says, "Hey, you have the skills I need, would you like to be involved?" That can flip things in someone's mind right away from thinking, "Oh, this isn't something that's for me because maybe I haven't seen people like me do this before or just didn't imagine that I could." But then someone taps you on the shoulder and you're like, "Okay, yeah, maybe I can do this. I can give it a shot." Especially I guess if some mentorship is paired in there. And then you made an interesting point there at the beginning of your answer around more paid roles.

	01:15:34	You didn't say this explicitly, but you said that because in a lot of, maybe all cultures around the world, women disproportionately do unpaid work in the household, in child-rearing, these kinds of things. There are exceptions on an individual basis, but on the society as a whole, this is what we see. And so there's maybe something, some big problem of society that could be rectified over long periods of time, but in the immediate term, having more and more paid open source developer roles would alleviate some of this problem because then somebody doesn't have to be thinking about having this additional unpaid work on top of the paid work that they already do because they can incorporate it into their payroll.
Marco Gorelli:	01:16:30	Totally. Yeah. It needs to be an active effort. It's not going to fix itself.
Jon Krohn:	01:16:34	Yeah. Great answer. I've got one last topic area for you. We've heard a lot in this episode about the brilliant software development that you've done on a data library, on Polars and also in Narwhals, supporting Polars, but you have done some data science work in the past, specifically around forecasting. So you achieved impressive results in several forecasting competitions such as the M5 and M6 forecasting challenges. Do you want to tell us about what those challenges are?
Marco Gorelli:	01:17:08	Sure. Let's see if I can remember. That was a few years ago. Yeah. I used to work in data science. Well, my background's in mathematics, but then realized I wasn't good enough to be a mathematics academic, so became a data scientist.
Jon Krohn:	01:17:26	Zing.
Marco Gorelli:	01:17:27	Then got addicted to open source and became a software engineer. But for four or five years or something like that, I did work in data science and got quite interested in

forecasting. It was quite related to work I was doing in companies, and I just found that taking part in competitions was a fun way to improve your skills. People sometimes say that these competitions are not real data science, and I agree it's not real data science in that it doesn't show you the complete lifecycle of a data science project, but it can teach you some real lessons, which can then be useful when you are doing real data science.

01:18:07 So the M5 competition, that was a fun one. There, you had to forecast Walmart sales. It was, yeah, real data. And there were two tracks to it, the uncertainty one and the point prediction one. I worked with a friend of mine on both of them. We were, just as people often did on Kaggle back then, blending solutions together. And what we generally found, what generally the Kaggle community finds is the most important thing isn't using the most unusual model, but having a good way of cross validating your data, of having a way of estimating how well is your model going to perform on unseen data?

01:18:58 When people talk about Kaggle and real life data science, I think the fact that it teaches you to do cross validation well is the biggest benefit that it'll bring you. Then came the M6 competition, and that was financial forecasting. And there I just took a bit of a gamble. I just figured, well, most people are going to overfit. I don't know anything about finance, if I just submit the simplest possible thing, then maybe it'll land the top 10% and I can put that on my CV.

01:19:31 Unexpectedly, I came second in the first quarter and was awarded $6,000 for that. So yeah, I put that on on my CV to look smart, but I don't know anything about finance. I don't have any insight here. If someone wants to come to me for trading advice, then I can't tell you anything useful other than don't do anything wild. I can tell you about

how to beat other competitors in financial forecasting competitions, but not necessarily how to do-

Jon Krohn: 01:20:04 By keeping it simple.

Marco Gorelli: 01:20:05 Exactly.

Jon Krohn: 01:20:06 So two tips there. So you might be able to out-compete people in forecasting competitions by sticking to simpler models that are less likely to over-fitting and also the importance of cross-validation, which you mentioned there. Which is something if you're not already aware of it, this is where you take, say... A common way of describing cross-validation is with k-fold cross-validation where you split your data into some number of partitions.

01:20:34 And so if you did say five-fold cross-validation, you would train your model on, well, you split your data into five parts of equal size randomly putting samples into each of those five buckets equally sized, and you train on 40%. So you train on four of the buckets and evaluate on the fifth, and then you can repeat that five times, each time leaving a different 20%. So the first 20%, the second 20%, third 20%, going like that through all five 20 percents, and in this way you are training and validating on all of your data, taking advantage of all of it.

Marco Gorelli: 01:21:19 Yeah, maybe just small note. In time series, you need to be especially careful with how you make your buckets so that you're not training on future data and predicting past data, but that's the idea. Yes.

Jon Krohn: 01:21:32 Great point. Glad that you pointed that out there. So this has been a fascinating episode. I've loved it. It's been illuminating to hear so much about Polars from you. You describe in such crisp, clear detail every aspect of what you're talking about and make it so easy to understand.

So really appreciate you doing that, Marco. Before I let you go, do you have a book recommendation for us?

Marco Gorelli: 01:21:54 We're doing fiction, nonfiction?

Jon Krohn: 01:21:56 Whatever. You can do one of each if you really want to.

Marco Gorelli: 01:22:01 Let's be greedy and do that, then. Take two. Okay, so a technical book. I think Programming Rust published by O'Reilly is really good, as is the Rust programming language. So yeah, if you are a Python programmer, want to get into this, want to write your Polars plugin, then it's a really accessible way to get into the language. Fiction? The last fiction book I remember really enjoying is called All That's Left Unsaid by Tracy Lien, just about some Vietnamese immigrants in Australia. This girl, her brother's been murdered, but her family, they're really distrustful of the police, really distrustful of the authorities, don't want to speak to anyone about anything, and she's trying to understand what's happened to her brother. Really good book. Recommend it.

Jon Krohn: 01:22:50 Great recommendations. Thank you, Marco. And for people who would like to follow you on your thoughts after this episode, how should they do that?

Marco Gorelli: 01:22:58 If people want to follow me on social media, they can find me on GitHub at Marco Gorelli. Other social media, I'm on LinkedIn and Fosstodon.

Jon Krohn: 01:23:11 Nice. Fosstodon, one of the many, although I think probably the most popular. Do you think? Kind of, post-Twitter social media places to be?

Marco Gorelli: 01:23:23 Maybe, yeah. People still call it Twitter much to Musk's angerment. Oh, well. Yeah, on there. Well, Mastodon, as, I'm still not totally sure how this federation thing works,

but I log on to Fosstodon.com. So I'm going to call it Fosstodon.

Jon Krohn: 01:23:42 Nice. Cool. Well, maybe we can dig into that kind of stuff, the social media stuff, this post Twitter options, maybe dedicate an episode to that at some point. Thank you so much, Marco. It's been great having you on the show, and thank you again for making the trip to London from Cardiff. Maybe we can check in again in a few years and see how Narwhals, Polars, whatever other exciting projects you've gotten yourself into by then are coming along.

Marco Gorelli: 01:24:08 Sure. Thanks for having me.

Jon Krohn: 01:24:15 Absolutely fascinating technical discussion with Marco today. In today's episode, he filled us in on how Polars excels at feature engineering and allows up to a 100x speedups, especially on large dataframes thanks to lazy execution. He talked about how on string evaluation such as for natural language processing, Pandas is optimized for this natively, so it outperforms the leading data manipulation libraries at Python. That is, NumPy and Pandas.

01:24:37 He talked about how his Narwhals library allows other libraries such as the popular declarative visualization library, Altair to be dataframe agnostic, allowing support for Polars without any detriment to Pandas users. He told us how he won $6,000 in prize money in the M6 forecasting competition by assuming that most teams would overfit their models to the training data. And he talked about how more paid roles, more mentorship and active reach outs could increase diversity amongst open source software developers.

01:25:06 As always, you can get all the show notes including the transcript for this episode, the video recording, any

materials mentioned on the show, the URLs from Marco's social media profiles, as well as my own at superdatascience.com/815. Thanks of course to everyone on the Super Data Science podcast team. You've got our podcast manager, Ivana Zibert, media editor Mario Pombo, operations manager Natalie Ziajski, researcher Serg Masis, writers Dr. Zara Karschay and Silvia Ogweng, and founder Kirill Eremenko.

01:25:34 Thanks to all of them for producing another dazzling episode for us today, for enabling that super team to create this free podcast for you. I am so grateful to our sponsors. You can support this show by checking out our sponsors' links, which are in the show notes. And you yourself, if you are interested in sponsoring an episode, you can do that. You can find the details on how by making your way to jonkrohn.com slash podcast. Otherwise, please share, review, subscribe and all that good stuff. But most importantly, just keep on tuning in. I'm so grateful to have you listening, and I hope I can continue to make episodes you love for years and years to come. Until next time, keep on rocking it out there and I'm looking forward to enjoying another round of the Super Data Science Podcast with you very soon.