Audio file

Your Recording 15.wav

Transcript

00:00:05 Speaker 2

In this lesson, we're going to look at why Sagemaker is considered a mysterious product and some might even say intimidating.

00:00:12 Speaker 2

So what are we going to look at in this lesson?

00:00:14 Speaker 2

We're going to look at why Sagemaker feels intimidating so we can better understand the complexity of the product. We're going to breakdown Sagemaker into its component parts and understand when we would use a different feature of Sagemaker to achieve a specific outcome.

00:00:28 Speaker 2

And we're going to spend a little time aligning to use cases. Each part of Sagemaker is going to serve a specific purpose. And we're going to introduce personas like the data engineer, the data scientist, the machine learning operations engineer, and understand why different personas might use different parts of the product. Maybe when we've been learning, we've been trying to learn the whole thing as a single practitioner, when in reality we're going to be using different parts of the product depending on who we are.

00:00:53 Speaker 2

So why is Sagemaker seen as intimidating?

00:00:55 Speaker 2

Well, the first thing we need to understand about Sagemaker is it's not really a single product, but actually a collection of tools that can be used by different personas at different times during the machine learning lifecycle. So for example, if I'm starting out my project, maybe I need to clean up some data. Or maybe I'm a data scientist and I want to visualize my data to better understand correlations between features. Then I'm maybe training my model. Then I need to host my model and perform inference. So different personas will need different features of Sagemaker at different times. So approaching Sagemaker.

00:01:25 Speaker 2

Single product as a single practitioner doesn't actually make sense.

00:01:30 Speaker 2

If we consider Sagemaker from the perspective of the persona who is performing an action and what they need to do, in other words having a use case for Sagemaker, then the product makes more sense. Here I can think of Sagemaker being used by a data engineer, but might need to extract some data from a data source and clean it up.

00:01:47 Speaker 2

I could consider a machine learning operations engineer who is focused on deploying models out onto compute platform so that it can perform inference, it can perform predictions. So an Mlops engineer is kind of thinking more like a DevOps engineer than thinking about how we can productionize, getting what we've developed our models out into a usable state that will generate predictions. So they're thinking about deployment and pipelines.

00:02:08 Speaker 2

And then we have the data scientist who's going to be focused on exploratory data analysis or EDA for short, and feature engineering and model training. So these three personas are all aligned to different tasks, which means they will be using different parts of the Sagemaker product.

00:02:24 Speaker 2

Now, when you're learning Sagemaker, you may have started to look into the user interface in the AWS management console, and this may have led you to some struggles to understand what was happening. For example, I know when I started looking at Sagemaker, I started to see how to create a training job and I clicked on training jobs and there's a button saying create training job, but it was asking me for inputs that I had no idea what it meant. And once I started working with data scientists in the enterprise, I realized that they are starting from a position of code first, the data set, their Python code and using tools.

00:02:54 Speaker 2

Like scikit learn, pandas, OK, so they're using tools to perform a specific job. And ultimately if they're using scikit learn, they probably were training their models locally on their laptops. So when we start to think about training in Sagemaker, it's asking us for inputs that unless we know what training looks like when you're doing it in code, it's very hard to translate that concept into a UI LED interaction.

00:03:17 Speaker 2

So we'll find that most Sagemaker practitioners are not using the UI. They're not using the button that says create processing job or create training job. Instead, they've gone with a code first approach because that's where they are comfortable. They have come from that environment working locally. And then when they start to use Sagemaker, they're starting to use the compute features, the scalable compute features of the cloud to perform these actions. But they are still doing it with Python, they're still doing it with Pandas, psychic learn, and Jupyter notebooks. It's just where the jobs are running has moved.

00:03:46 Speaker 2

From local computer to cloud. So we in learning Sagemaker should be thinking more like a data engineer or a data scientist and trying to accomplish jobs and then just thinking about, well, how would I accomplish that job in that environment? So you really need to learn data science and how we do creation of models first so that you can transpose that idea and think, OK, well how does Sagemaker do it differently?

00:04:10 Speaker 2

When we talk about a code first approach, we will be making use of Python And a particular software developer kit called the Sagemaker SDK for Python. This is a dedicated SDK that is in addition to the regular AWS SDK for Python, and this is about abstracting high level tasks in machine learning so that they're easily created in just a few lines of code.

00:04:29 Speaker 2

We'll be working initially in Jupiter notebooks. Now we've got a whole lesson on Jupyter Notebooks. So if you've never seen one before, don't worry, we're going to be seeing exactly how they work. But for now, we need to know that Jupyter Notebooks are kind of like an integrated development environment a little bit. How maybe a developer might use Visual Studio Code. Well, data scientists might use Jupyter.

00:04:46 Speaker 2

And that will give us an indirect of Python environment where we can run our Python code. That will still leverage pandas and scikit learn. But it will now make use of this Sagemaker SDK so that it can interact with the Sagemaker compute platform and delegate out processing, processing for data cleanup, processing for feature engineering, processing for training.

00:05:05 Speaker 2

So the best practice for us is to have a code driven workflow. That's how 95% of practitioners using Sagemaker are using the Sagemaker platform. So having those UI buttons to create training jobs and create processing jobs is a bit of a distraction. So I would suggest we don't go down that road but instead work like proper data and ML professionals in a code first approach.

00:05:24 Speaker 2

Most of the users of the Sagemaker platform only use maybe around 70% of its capabilities. So again, think about the specific use cases like training a model for detecting fraud or training a model to predict prices based on historical values. I could have a linear regression type idea. There are many features in Sagemaker that cope with every possible use case, whether it be labeling data or outsourcing of labeling data or training large language models. But for the moment we're concentrating on the core functionality which will be used for ingesting tabular data.

00:05:54 Speaker 2

In order to train models for linear regression and logistic regression and host those models for inference on the platform.

00:06:00 Speaker 2

When we think about the machine learning pipeline, that's preparing our data, taking it into a training process where we're building our model, maybe we're iterating through different versions of building our model and evaluating which one is the best and giving the best predictions, and then deploying that model on a compute platform and ultimately monitoring it to make sure the predictions produced are still accurate.

00:06:18 Speaker 2

But as we think about that machine learning pipeline, there are different parts of the Sagemaker AI platform that will align to different parts of that pipeline.

00:06:26 Speaker 2

So for example, in the data preparation stages, I could either use low code tools like Sagemaker Canvas and the Data Wrangler, or I could do a code first approach and create a data processing job within Sagemaker to do those data preparation tasks. As I move into training my model, then rather than using the training capabilities of my local device, I can train my model in the cloud. Now essentially what Sagemaker will be doing in the background is spinning up a virtual machine, running my training job in that, and then I get the output of its results. But the wonderful thing there is those training jobs will take advantage of.

00:06:56 Speaker 2

Never compute requirements I have. So if I need 24 CPUs and a TB of RAM to do my training, then I will only pay for that virtual machine while it's running and I don't need those compute capabilities locally. So to do that, Sagemaker offers me this concept called a training job and that will allow me to perform remote training.

00:07:13 Speaker 2

Once I've selected my model, I want to then store the model somewhere. So I've got a dedicated model registry for that, and we'll talk more about that later, But that's going to help me manage different versions of my model artifacts. Maybe think of it like GitHub for models. And if I just want to experiment with my model, I want to invoke training jobs. I want to check accuracy of my model. I'm going to be doing that as a code first action. So I'm going to be using Python And I'm going to want to have an interactive environment. So I'm going to need a Jupyter Notebook. Now, Jupyter notebooks need to run somewhere. I could run it locally on my laptop.

00:07:43 Speaker 2

But what would be better would be running any hosted environment and Sagemaker can provide me a hosted, what we call Jupyter Lab environment. And that's just a hosted environment in which I can run this interactive Jupyter environment that allows me to run my code in a secure location. So we started to think about, OK, my data assets, which are maybe in cloud, maybe an S3 buckets or a data warehouse. If I'm then creating training jobs that need to ingest that data and those training jobs are up in the cloud and my Jupyter notebook is up in the cloud, I can secure the perimeter of this data zone for easier.

00:08:12 Speaker 2

Than I can if I've got data scientists pulling entire datasets down to their local devices. So we can do this in a formal, performance and secure way within the Sagemaker environment.

00:08:21 Speaker 2

As we move to deploy a model so it can be hosted somewhere on a compute platform so that we may generate predictions from it, well, Sagemaker again provides a feature for that. They're called Sagemaker endpoints. And again, when we think about what might be happening here, it's simply a virtual machine is being created in the background. Your model is being deployed into that virtual machine and we're hosting it for you on your behalf.

00:08:41 Speaker 2

So I can see as I look at this, I go, OK, Hmaker endpoints are not going to be used by data engineer because they're going to be working at the front end of the pipeline at Sagemaker endpoint might not be used by a data scientist.

00:08:53 Speaker 2

But a machine learning operations engineer whose job it is, is to get models out to production, yeah, they're going to be focused on Sagemaker endpoints. They would be focused on monitoring the traffic and making sure that the model is still producing good predictions. They would be focused on automation of updating models when new ones come out. And that would introduce you into Sagemaker pipelines. So already as we think about these Sagemaker features of Sagemaker Canvas, Sagemaker Data Wrangler, Sagemaker Studio, Jupyter Lab, Sagemaker training jobs, Sagemaker endpoints, Sagemaker Monitor, that's a big list of features, but.

00:09:23 Speaker 2

We're beginning to see that we've got a better idea about who might use what feature and at what point in the pipeline.

00:09:30 Speaker 2

So let's separate that out. Our data engineer now might use the Canvas or data Wrangler. Those are low code tools, not requiring you to know very much Python, if any, or data processing jobs invoked from Jupyter. Jupyter being our interactive Python environment, very much aligned with what the data engineer is doing at the front end of our pipeline.

00:09:46 Speaker 2

Are data scientists then start the exploratory data analysis and then we'll move into feature engineering and training. When they're doing feature engineering, they're probably using data processing jobs to offload processing of large amounts of data. When they move to training, they want to offload the training to Sagemaker training jobs, again offloading the compute about where those jobs occur. They register their models so that they are available for deployment. The Mlops engineer can then concentrate on getting that model deployed into a production account. They will automate, probably using Sagemaker pipelines, the creation of a Sagemaker endpoint so that we have.

00:10:17 Speaker 2

Somewhere that our model is hosted and can generate predictions from.

00:10:21 Speaker 2

So if we look at that from a persona centric view, you can see we've got our three personas and then we think about what Sagemaker features they use and the activities they are using them for. The data engineer is going to make use of either the local tools Canvas and Data Wrangler or the code first tools of Sagemaker Studio and data processing jobs. But ultimately what they're using those tools for is to prepare our data and that might be to transform the data. Maybe the data resides in an original source format of Jason and we want to convert it to CSV. Maybe we want to handle missing data or handle outliers, something like that.

00:10:51 Speaker 2

Before we hand off to the data scientist.

00:10:54 Speaker 2

The data scientist is going to use different features within Sagemaker. The data scientist will make use of Sagemaker Studio, an environment in which they can run Jupyter notebooks. But their focus is going to be on feature engineering, on exploring that data, understanding correlations between features, and ultimately training a model. So they want to make use of Sagemaker training jobs, and once they've created their model, they'll want to register that model into a version control repository that we call the Sagemaker Model Registry.

00:11:19 Speaker 2

They will also be interested in the Sagemaker Model Monitor because once a model is in production, we want to know it's still producing good results.

00:11:25 Speaker 2

And that might create a feedback loop to the data scientist if the model in production starts to drift away from giving us good predictions.

00:11:32 Speaker 2

So there's our data scientist, feature engineering, EDA Moto training and monitoring model performance.

00:11:38 Speaker 2

The Mlops engineer is going to be focused on productionizing, so we know that they're going to make use of Sagemaker endpoints to host the model for inference. Again, that's making an assumption that we are going to use Sagemaker AI as our inference platform. There's nothing that says we have to. We could train a model in Sagemaker and produce a model artifact and then host it somewhere else. That's OK. It's simply that the platform gives us that as a feature, as a capability, and you can decide if you want to use it or not.

00:12:04 Speaker 2

There's certainly a big advantage in using Sagemaker as a hosting platform, because then you've got end to end automation on a single platform.

00:12:10 Speaker 2

Within your AWS account.

00:12:12 Speaker 2

Remember, the Mlops engineer is focused on getting the model to production. So if we make changes to our model, we would really like those changes to be detected and ideally automate the deployment. So when we think about CICD and DevOps, well, think of that from a machine learning perspective. I changed some code in my feature engineering and change a new model. I end up with a new model artifact. Wouldn't it be good if we could then approve that model and it would simply automatically deploy And Sagemaker pipelines we're going to see later on will do that for us. Machine learning engineer is also going to be focused on, well, can we deploy?

00:12:42 Speaker 2

Version of the model? Does somebody need to approve it before that automation occurs? So they will be concerned about using model registry to manage those model versions.

00:12:50 Speaker 2

Let's think about how Sagemaker might look in a production environment, because it will look a little bit different compared to how we might be learning Sagemaker.

00:12:58 Speaker 2

When we're learning Sagemaker, we tend to perform everything in a singular account.

00:13:02 Speaker 2

Now, in an enterprise environment, it's probably similar during project development. In other words, you'll perform your data processing, you will perform your training jobs, you will perform your data cleanup, you will perform your registration of a model into the model registry, probably in some kind of development account.

00:13:19 Speaker 2

But that's where things start to differ when you're learning compared to in production. Because in production we tend to use AWS accounts for separation of concern. So maybe if I want to deploy into a pre production or user acceptance type environment that might be in a separate AWS account. Production services would likely be ring fenced into another AWS account beyond that. So when we think about what Sagemaker features will be used in each account, it might look a little like this where you might have all those Sagemaker features used by the three personas.

00:13:49 Speaker 2

In the development account, but once you look into the pre-production or test environment, maybe it's only Sagemaker endpoint which will host your model for inference, might be used in test and in production. Maybe it's just the Sagemaker endpoint plus the Sagemaker model monitor. And that's so that we can monitor that inference that model is creating so that we can make sure that it is still producing accurate predictions. In any enterprise environment, we generally do want to have monitoring and a feedback loop so that we can correct anything as it starts to go wrong. So it's not just Sagemaker features used by different.

00:14:19 Speaker 2

Personas Sagemaker features will be used differently across the different AWS accounts that have been created for separation of concern and security purposes.

00:14:29 Speaker 2

When we think about automation and CICD and continuous integration, continuous deployment, what we need to think about is what will any automation pipeline look like so that the changes that are made during development and once we create a model and store it in the model registry, there has to be some kind of pipeline in the background that can detect that and then automate the deployment and update of your Sagemaker endpoints in your pre production and production accounts.

00:14:54 Speaker 2

I will be looking at that in a lot more detail when we reach the lesson on Sagemaker pipelines and Sagemaker projects.

00:15:00 Speaker 2

So what have we learned in this lesson?

00:15:03 Speaker 2

We have seen that Sagemaker gives us tools that can help us for the entire machine learning lifecycle. Rather than thinking of this as a single product, it's a collection of tools that will aid a persona such as data engineer, data scientist, machine learning operations engineer, with the activities they need to perform at a specific stage of the machine learning pipeline. Whether it's data preparation, exploratory data analysis, training a model or hosting a model or monitoring a model, Sagemaker is going to provide the tooling needed by that persona to achieve that specific task.

00:15:32 Speaker 2

We have seen that the way that we are going to approach machine learning projects with Sagemaker is typically going to be a code first approach. Meaning that we are going to start our project working in tools like Jupyter notebooks, using Python to call upon the features of Sagemaker, to process data, to train data, to create an endpoint, to host our model. All of that's going to be driven from Python code calling the Sagemaker SDK to achieve those specific tasks. If you're trying to learn by clicking around in the AWS Management Console for Sagemaker, it's going a hard way to learn how to use the.

00:16:01 Speaker 2

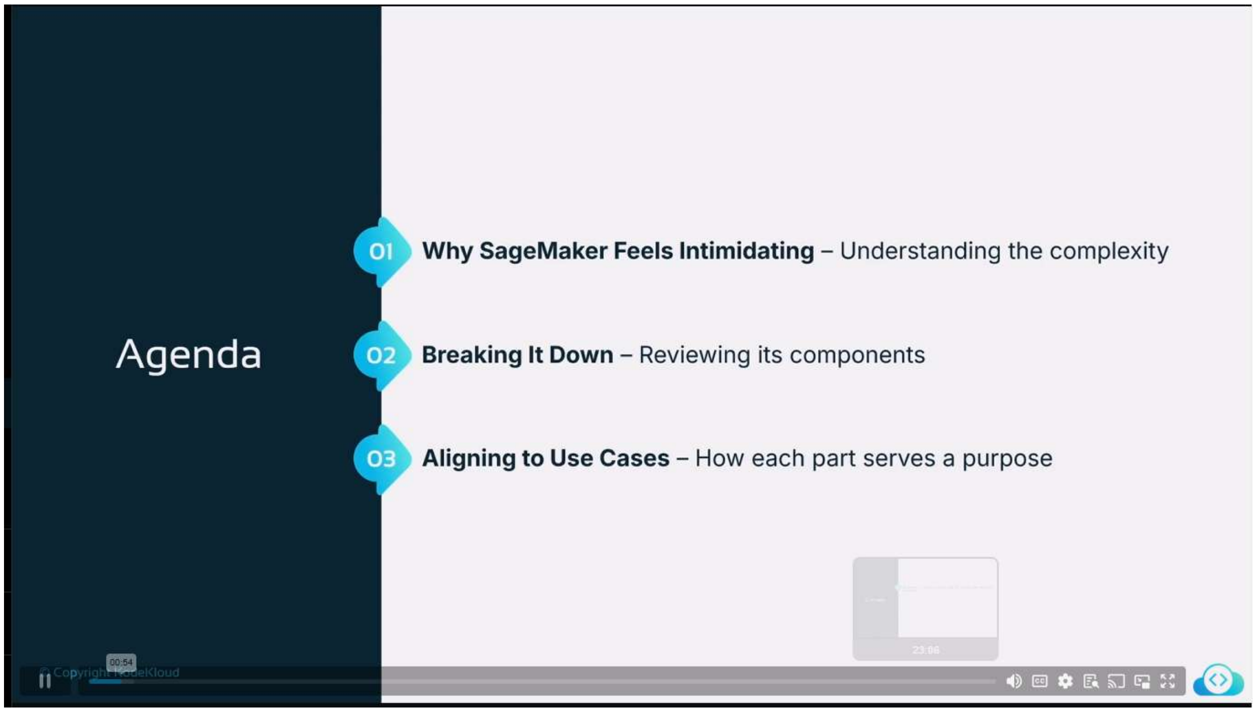
And it's not how most enterprises are using the product. It's a convenience, but it's really not the way that you will be using the product.

00:16:09 Speaker 2

We've seen that Sagemaker is really about helping those different personas as they move step by step through the machine learning pipeline. If you don't have a model idea to train, then you don't have a use case for Sagemaker. Sagemaker only makes sense when you're trying to develop a model and get it into production. And even then, in an enterprise environment, it's unlikely that a single practitioner will be using all those features. Those different personas will do their specific jobs in Sagemaker throughout that pipeline.

00:16:35 Speaker 2

So hopefully we're demystifying Sagemaker piece by piece here for you. I don't worry. As we go further into the course, we're going to expand on each one of those Sagemaker products that we've mentioned. See you there.

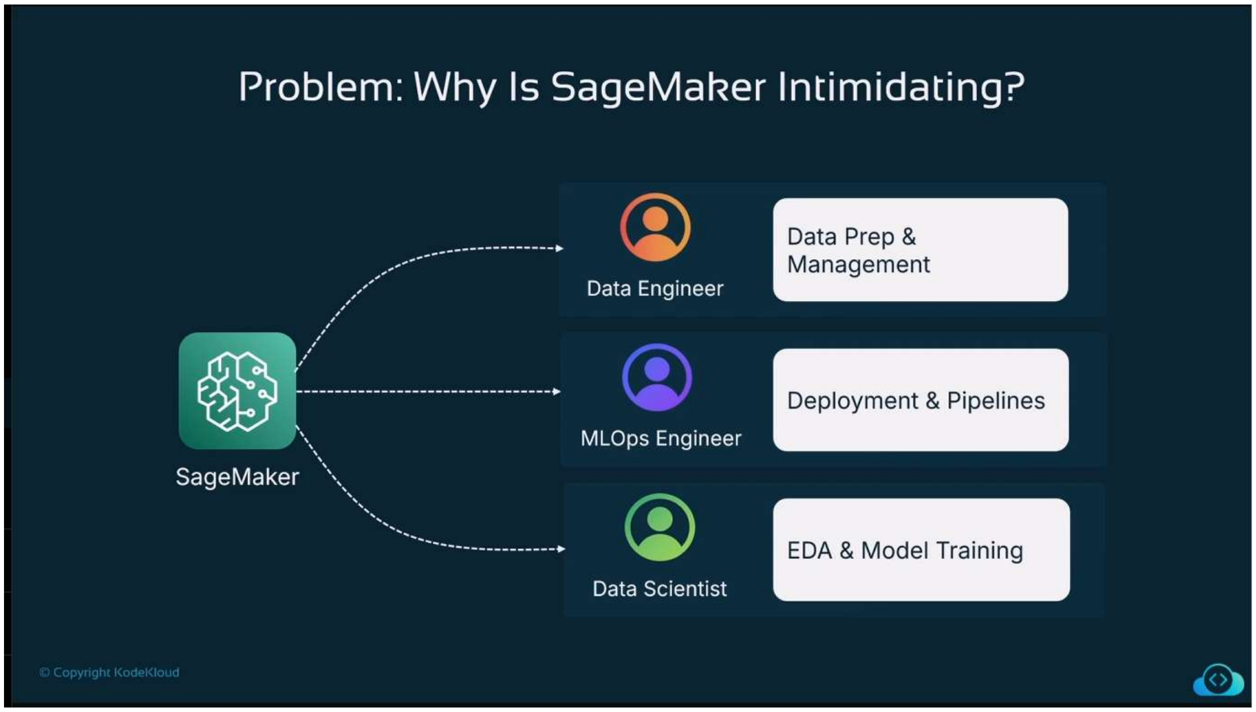


Agenda

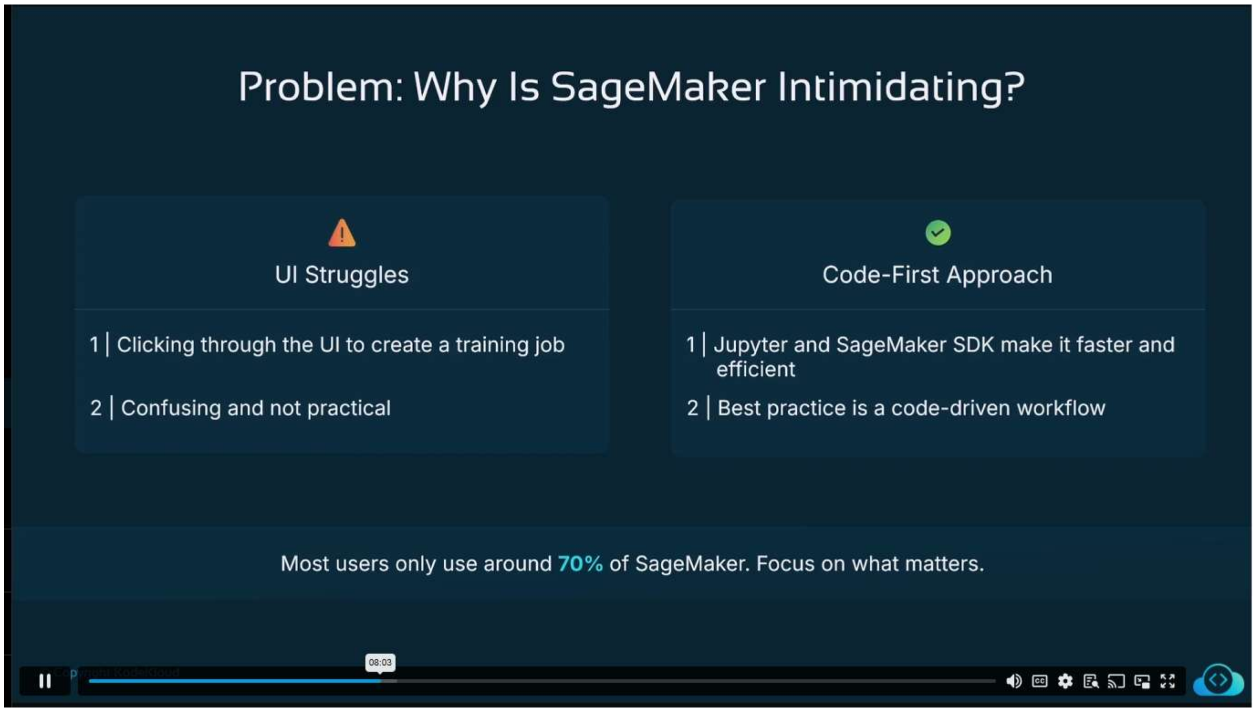
01 **Why SageMaker Feels Intimidating** – Understanding the complexity

02 **Breaking It Down** – Reviewing its components

03 **Aligning to Use Cases** – How each part serves a purpose



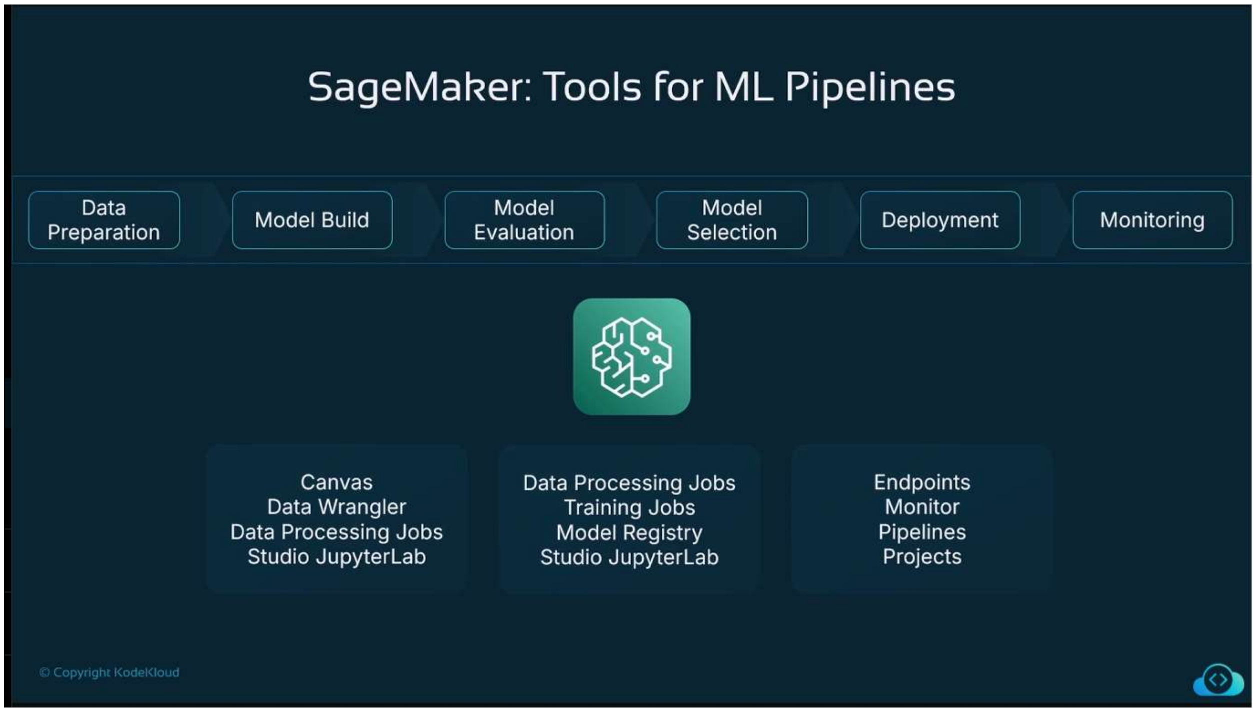
Problem: Why Is SageMaker Intimidating?



Problem: Why Is SageMaker Intimidating?

UI Struggles	Code-First Approach
1 | Clicking through the UI to create a training job	1 | Jupyter and SageMaker SDK make it faster and efficient
2 | Confusing and not practical	2 | Best practice is a code-driven workflow

Most users only use around **70%** of SageMaker. Focus on what matters.



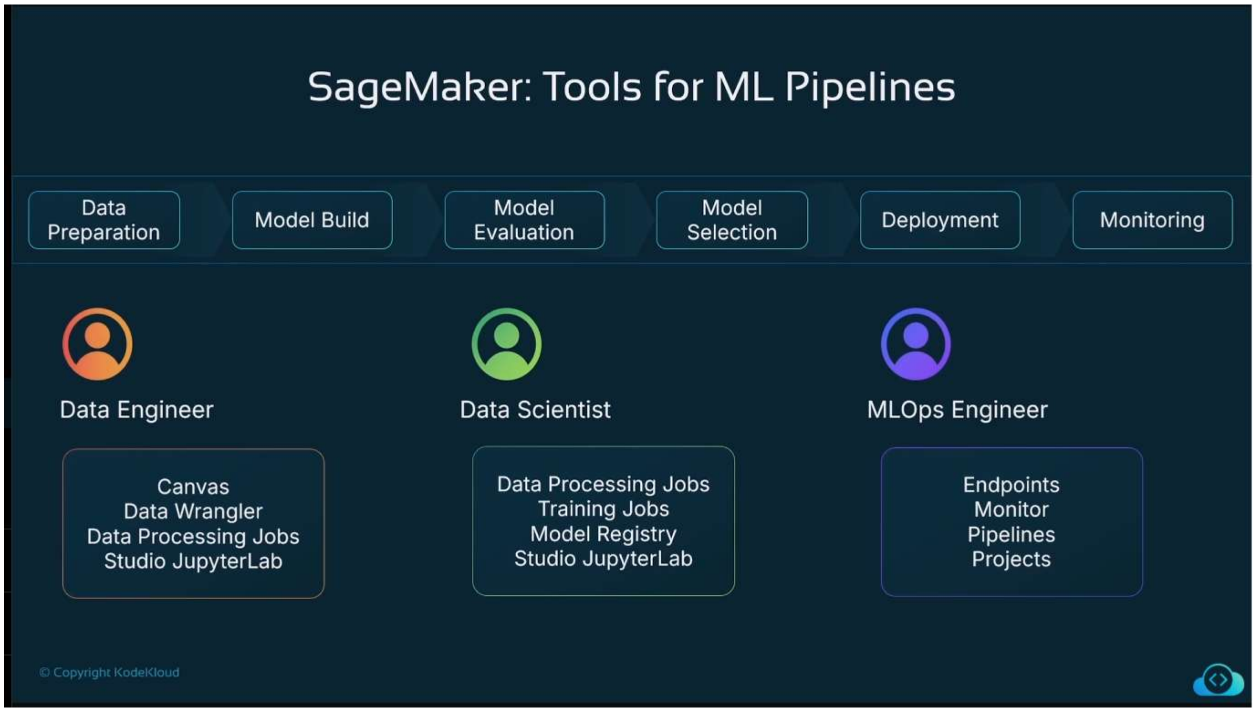
SageMaker: Tools for ML Pipelines

Data Preparation → Model Build → Model Evaluation → Model Selection → Deployment → Monitoring

Canvas
Data Wrangler
Data Processing Jobs
Studio JupyterLab

Data Processing Jobs
Training Jobs
Model Registry
Studio JupyterLab

Endpoints
Monitor
Pipelines
Projects



SageMaker: Tools for ML Pipelines

Data Preparation → Model Build → Model Evaluation → Model Selection → Deployment → Monitoring

Data Engineer

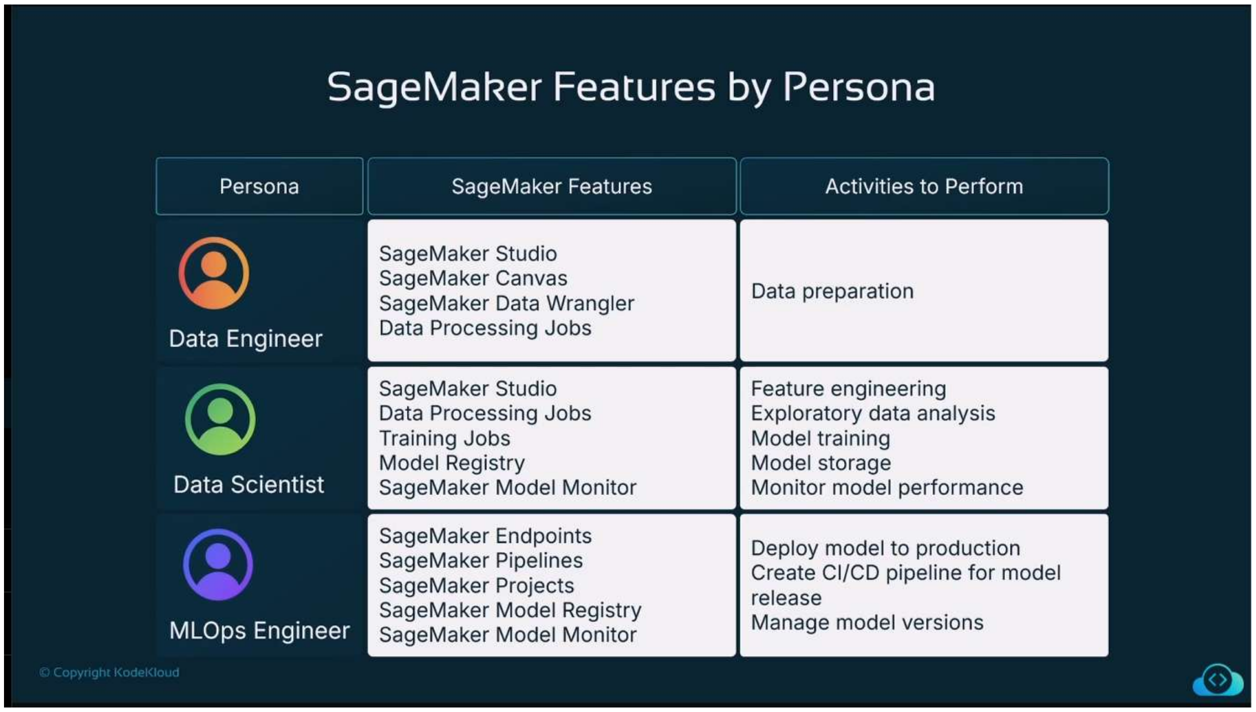
Canvas
Data Wrangler
Data Processing Jobs
Studio JupyterLab

Data Scientist

Data Processing Jobs
Training Jobs
Model Registry
Studio JupyterLab

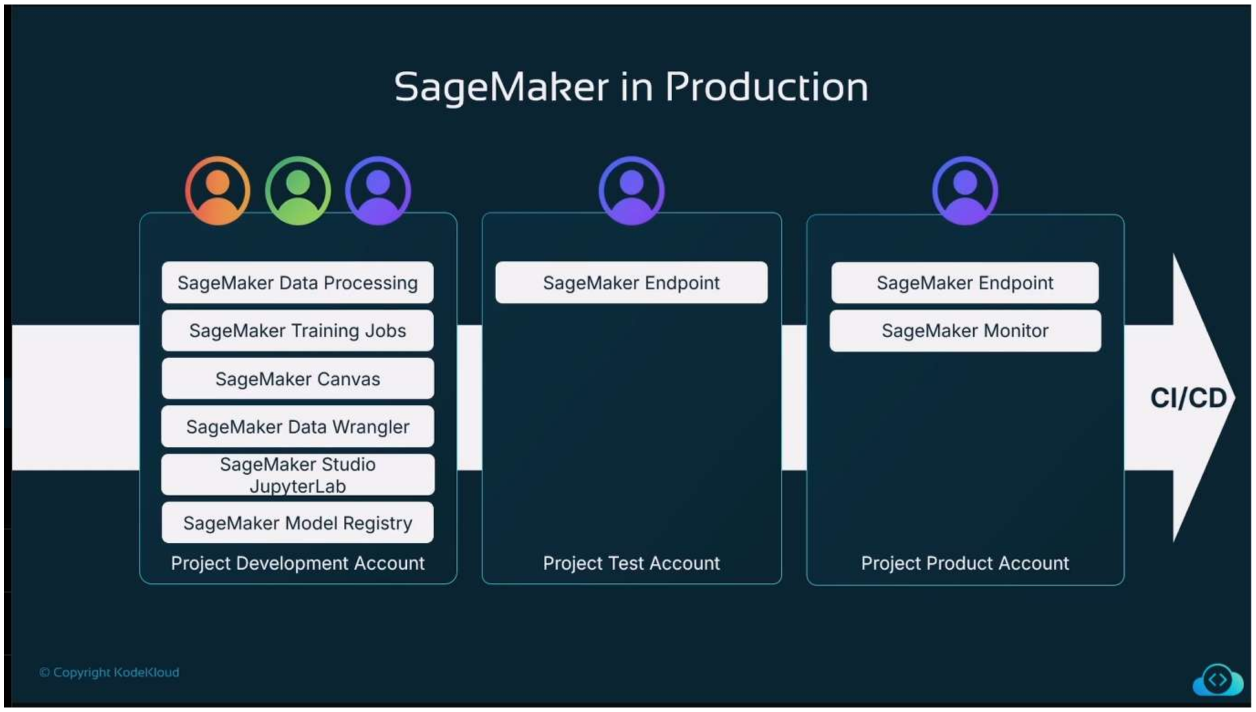
MLOps Engineer

Endpoints
Monitor
Pipelines
Projects

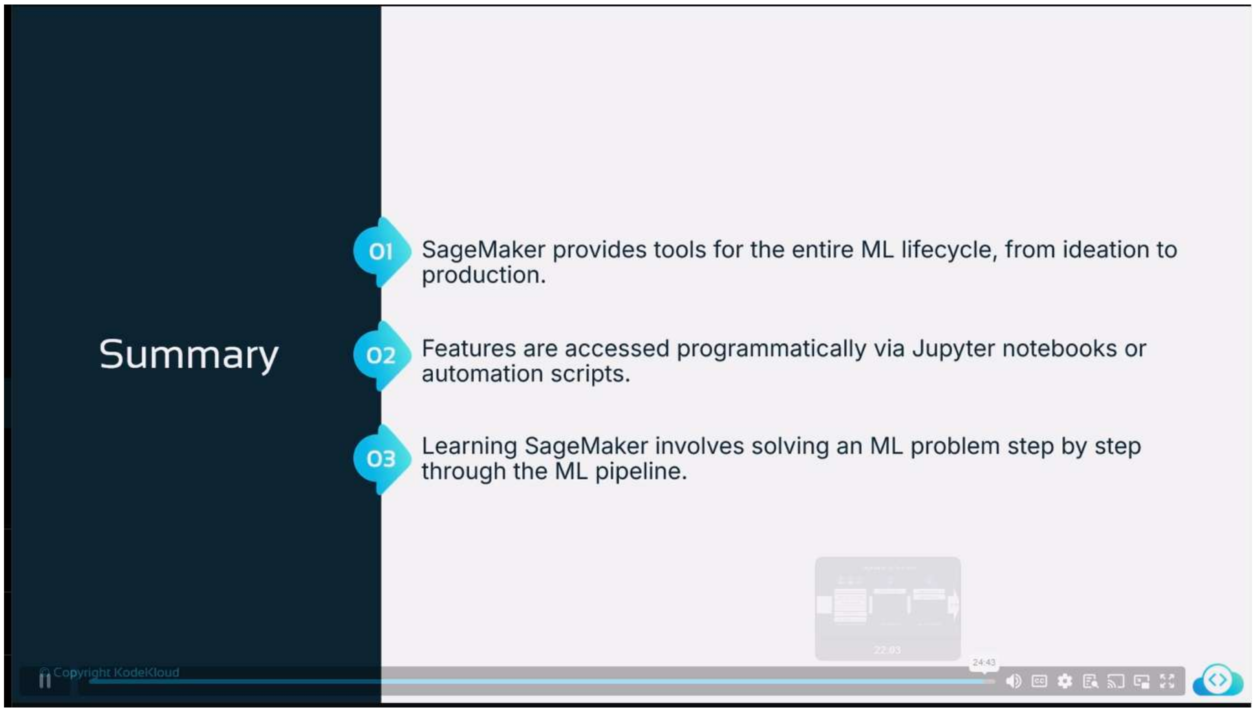


SageMaker Features by Persona

Persona	SageMaker Features	Activities to Perform
Data Engineer	SageMaker Studio SageMaker Canvas SageMaker Data Wrangler Data Processing Jobs	Data preparation
Data Scientist	SageMaker Studio Data Processing Jobs Training Jobs Model Registry SageMaker Model Monitor	Feature engineering Exploratory data analysis Model training Model storage Monitor model performance
MLOps Engineer	SageMaker Endpoints SageMaker Pipelines SageMaker Projects SageMaker Model Registry SageMaker Model Monitor	Deploy model to production Create CI/CD pipeline for model release Manage model versions



SageMaker in Production



Summary

01 SageMaker provides tools for the entire ML lifecycle, from ideation to production.

02 Features are accessed programmatically via Jupyter notebooks or automation scripts.

03 Learning SageMaker involves solving an ML problem step by step through the ML pipeline.