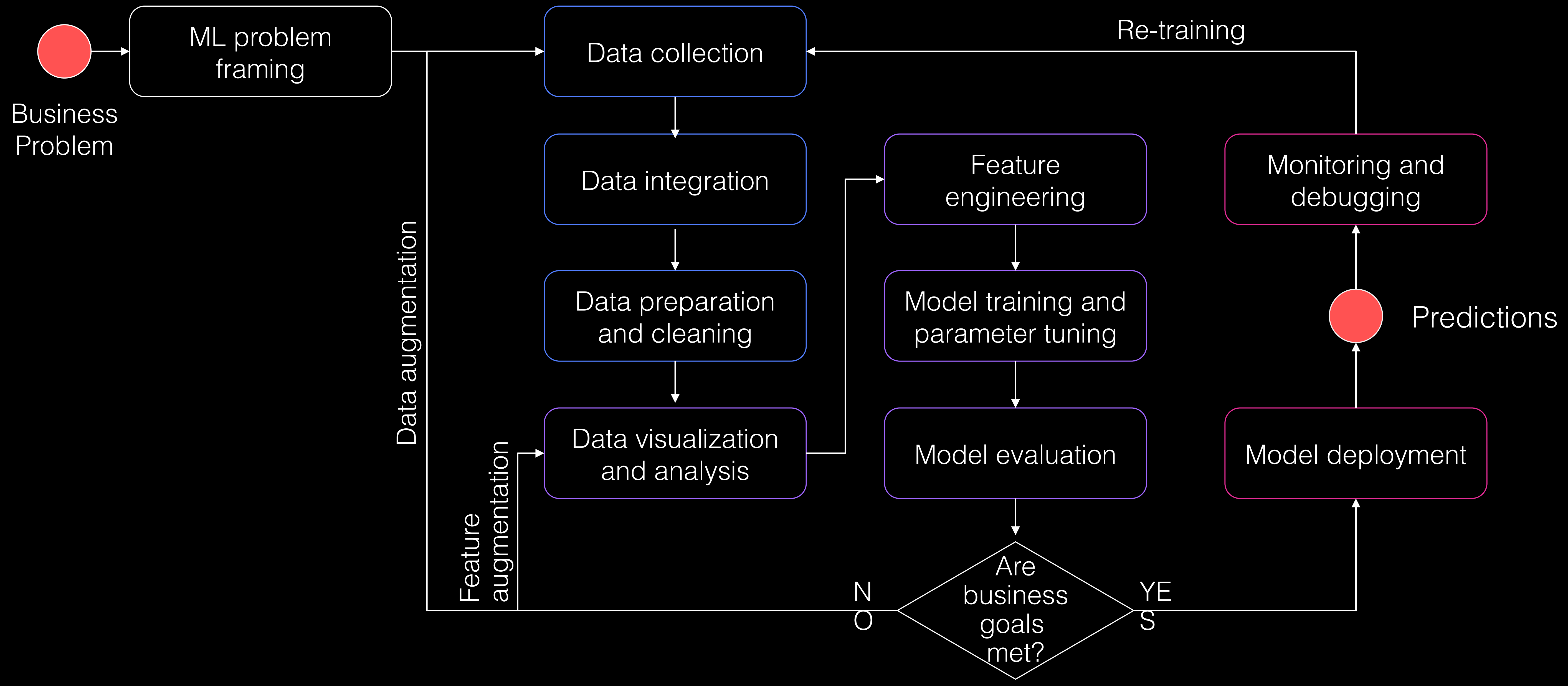
Become a Machine Learning developer with AWS services

Julien Simon
Global Evangelist, AI & Machine Learning, AWS
@julsimon

Lars
Hoogweg
CTO, Lebara

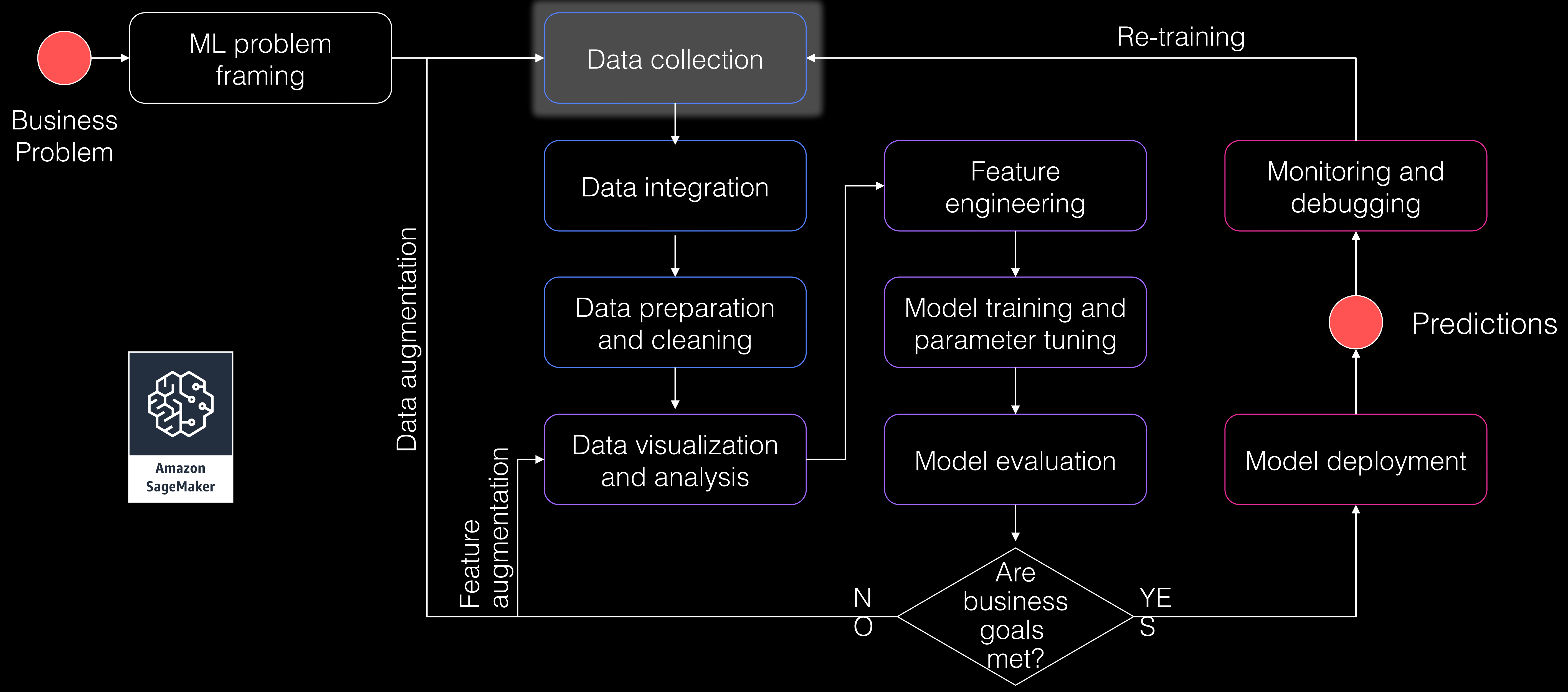
AWS SUMMIT

Machine learning cycle



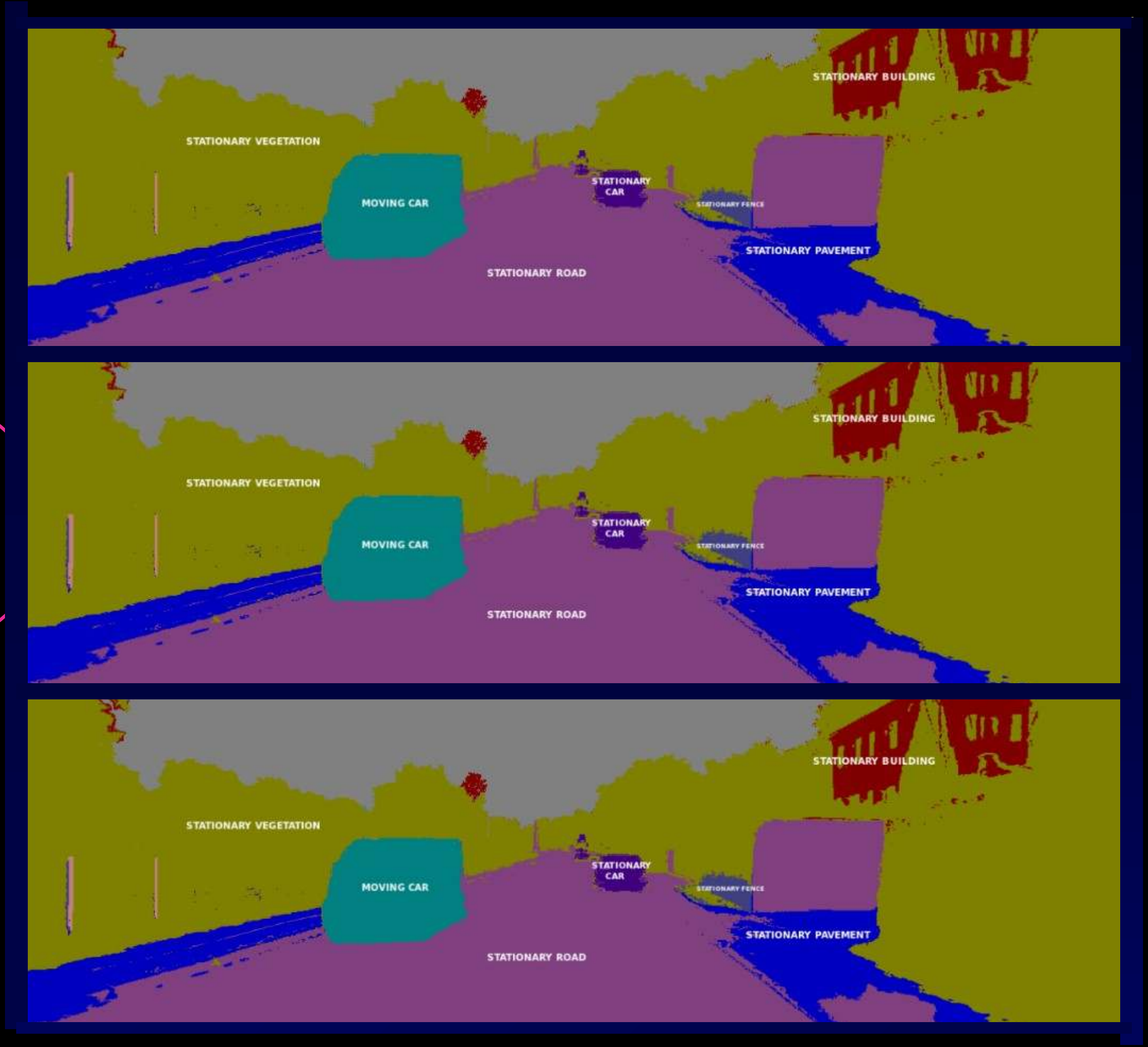
aws SUMMIT

Build your dataset



aws SUMMIT

Annotating data at scale is time-consuming and expensive



Amazon SageMaker Ground Truth

Build scalable and cost-effective labeling workflows

Quickly label training data

Easily integrate human labelers

Get accurate results

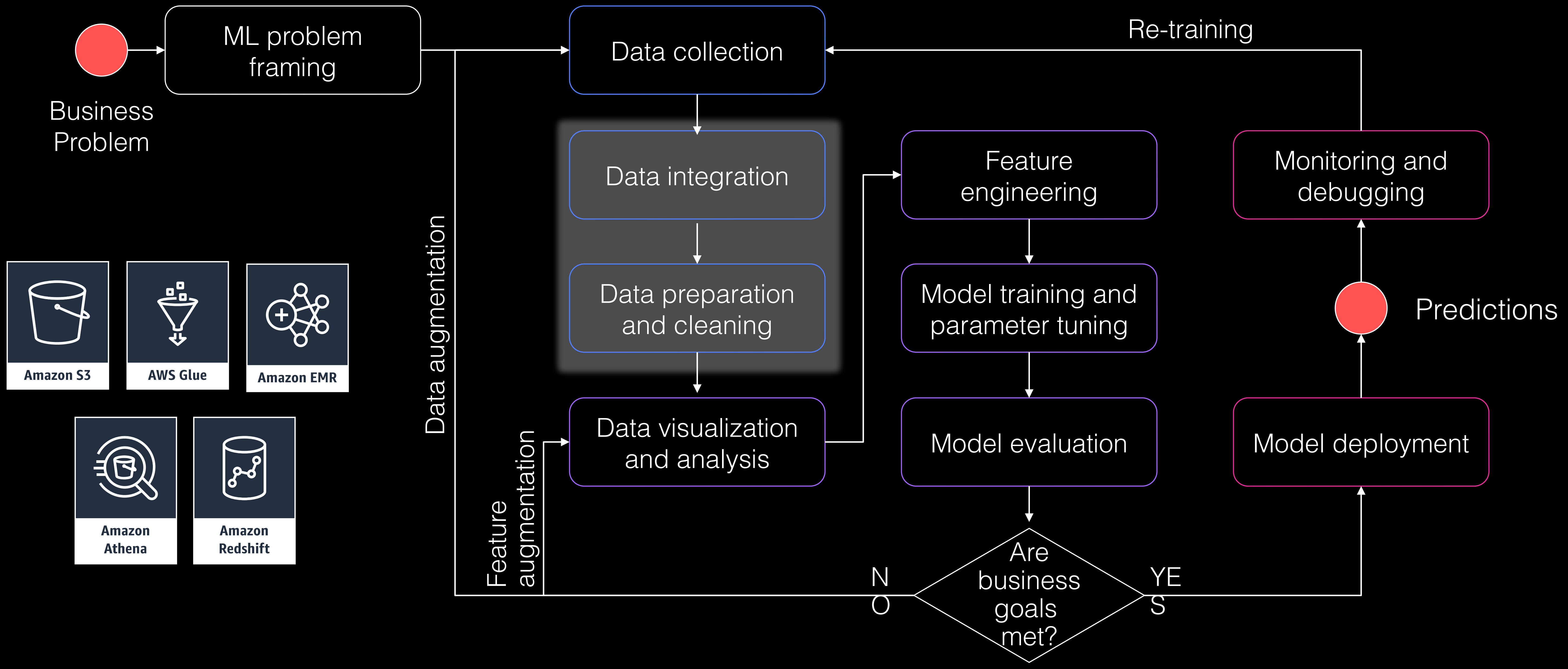
KEY FEATURES

Automatic labeling via machine learning

Ready-made and custom workflows for image bounding box, segmentation, and text

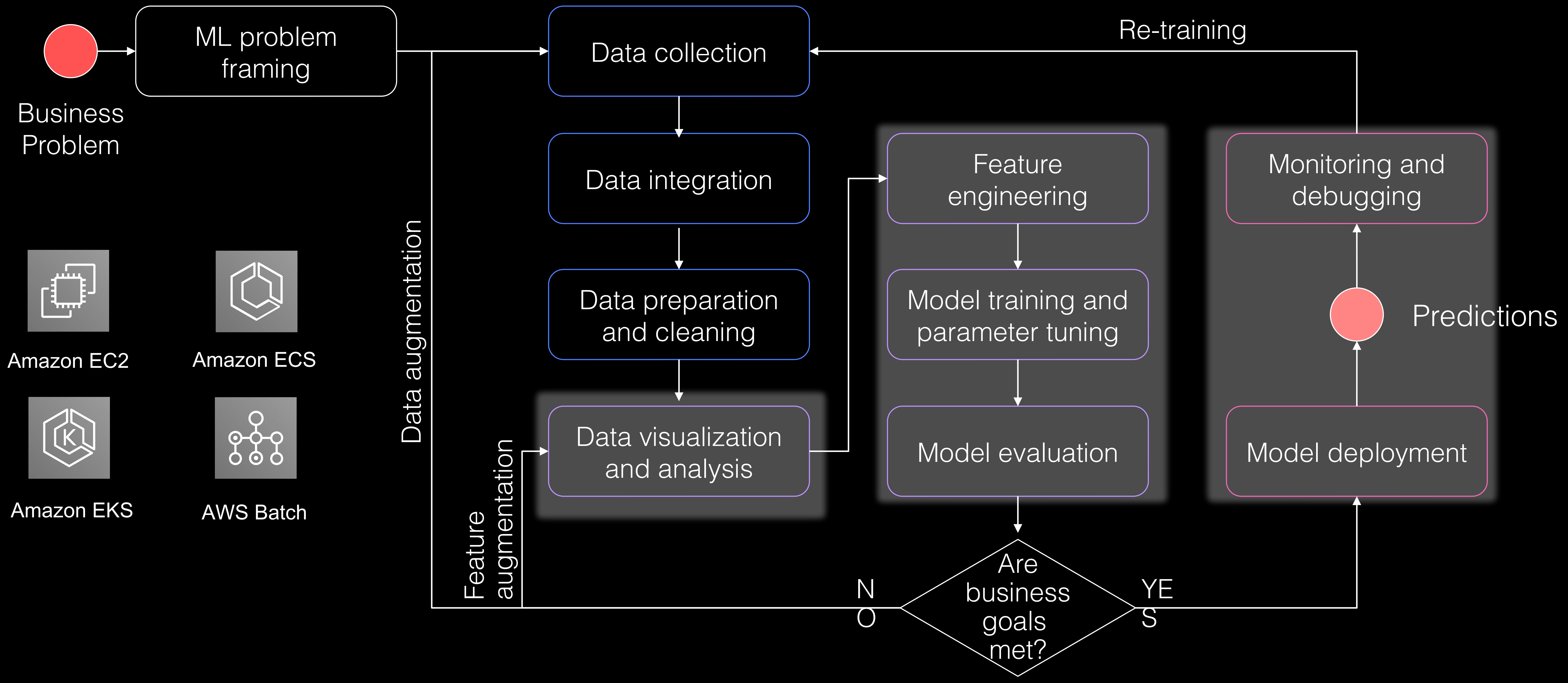
Private and public human workforce

Prepare your dataset for Machine Learning



aws SUMMIT

Build, train and deploy models using compute services



aws SUMMIT

AWS Deep Learning AMIs

Preconfigured environments on Amazon Linux or Ubuntu

NEW (March 27th) Deep Learning containers

Conda AMI
For developers who want pre-installed pip packages of DL frameworks in separate virtual environments.

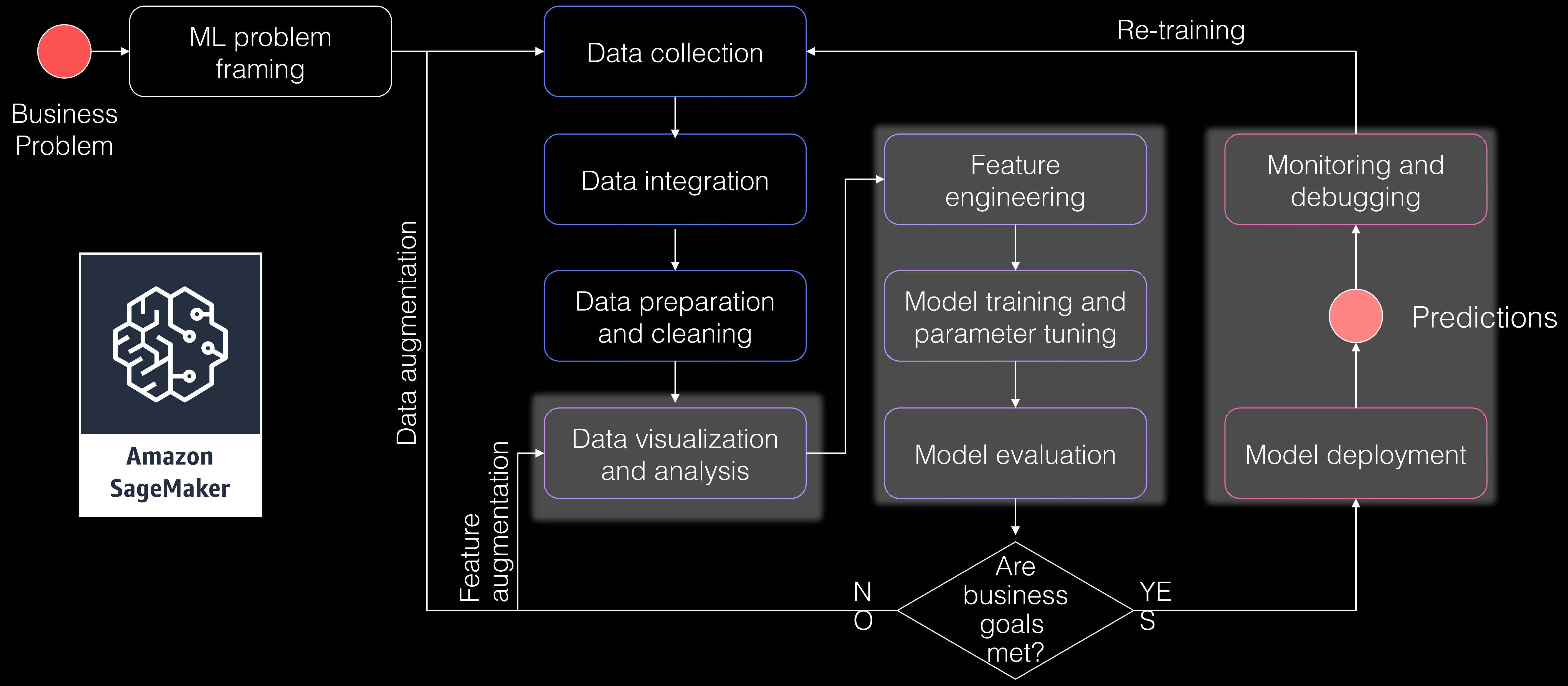
Base AMI
For developers who want a clean slate to set up private DL engine repositories or custom builds of DL engines.

AMI with source code
For developers who want preinstalled DL frameworks and their source code in a shared Python environment.

TensorFlow mxnet torch Microsoft CNTK Caffe2 ONNX
Keras GLUON PYTORCH Chainer HOROVOD

© 2019, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

Build, train and deploy models using SageMaker



aws SUMMIT

Model options

AWS Machine Learning Marketplace: 150+ off-the-shelf models

Training code

Factorization Machines
Linear Learner
Principal Component Analysis
K-Means Clustering
XGBoost
And more

Built-in Algorithms (17)

No ML coding required
No infrastructure work required
Distributed training
Pipe mode

mxnet Chainer TensorFlow PyTorch scikit learn

Built-in Frameworks

Bring your own code: script mode
Open source containers
No infrastructure work required
Distributed training
Pipe mode

Bring Your Own Container

Full control, run anything!
R, C++, etc.
No infrastructure work required

Using Machine Learning to detect Telco Fraud

Lars Hoogweg
Chief Technology Officer
Lebara

aws SUMMIT

Agenda

About Lebara

Telco Fraud

Using ML for detecting Telco Fraud

Next Steps

About Lebara

- Mobile Virtual Network Operator
- Active in 5 countries across Europe
- Our mission: to make it easier for migrant communities to stay connected to family and friends back home

What is Telco Fraud?

"the use of telecommunications products or services with the intention of illegally acquiring money from a telecommunication company or its customers"

Telco Fraud Examples

- SIM Boxing
  - A SIM box is a device containing a number of SIM cards. These SIM cards are used to terminate (international) calls bypassing international interconnect charges
  - One A-number calling many different B-numbers
- Revenue Share Fraud
  - Generate traffic to high cost, revenue share service numbers
  - Multiple A-numbers calling the same B-number or range of B-numbers.
  - Higher than average call duration
- Wangiri Fraud
  - A special case of Revenue Share Fraud
  - Making random calls from premium rate numbers, letting the calls ring once and then hanging up, hoping that recipients call back

Fraud Detection @ Lebara

- Current fraud detection approach is rule based
  - Fraudsters may change their patterns when they hit these rules
  - We cannot detect the fraud we do not know

- Can we use ML to improve our fraud detection capabilities?
  - Automating fraud detection
  - Detecting new types of fraud?

- How do we find out given our limited knowledge of ML?

Approach

- Organized a three-day offsite workshop together with AWS ML experts
- Working with actual Lebara data: Call Detail Records (CDRs)
- Data set labeled using existing fraud system

- Three groups focusing on three different types of fraud
  - Focus on Revenue Share Fraud for the rest of this presentation

- Training and deploying models with Amazon SageMaker

Call Detail Records (CDR)

- For each call, SMS, data session, top up, etc., a CDR is generated in real-time by Lebara’s Online Charging System
- Lebara streams CDRs using Amazon Kinesis Firehose and stores them in Amazon S3
- So, what does a CDR look like?

An Example Call Detail Record


```
704000001796849823|0|20190504205402|1|31616531654|31624586868|31616531654||
0624586868||1|00|316530200000|204083309537469|||20190504215245|8|0|0|20190504215254|0|
68|1|30454341333043443337||120||||1287862183|310008|31616531654|1|0|20190501|0|0|0|0||31|
99|1|31|99|99999|31|99|2|31|99|2|310008|10000000000000000000|0|0|0||||1000000|
102779300616118593|2||0|120|0|0|192440000|0|0|102779200616117893|0||6372|0|120|179880|0|0|
0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|||||||0|0|0||||||||20190501000000||||0|0|
0|0|0|0|0||0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0||||31624586868||||
ocg2;1556999564;69969258;2||||||||||1|N|N|201905041950|20190504215402|D|11000|11|
102779300616118593|102779200616117893|5010000060964541NLD|5010000060964541NLD|
1003|1|0|0|0|0|0|0|0|0|5010000060964541NLD|102779300616118593|5010000060964541NLD|
0|0|0|||0|0|||0|0||||0|||0|0|||0|0||||0|||0|0|||0|0||||0|||0|0|||0|0||||0||F|S|31616531654|
704000000967825550|1003|0|179880|319903||||0|0|0|0|0|0|||||0|0|0|0|0|0|||||0|0|0|0|0|0|||||0|0|0|
0|0|0|0|||||0|0|0|0|0|0|0|0|||||0|0|0|0|0|0|0|||||0|0|0|0|0|0|0|||||0|0|0|0|0|0|0|||||0|0|0|0|0|0|0|||||0|0|0|0|0|0|
0|||||0|0|0|0|0|0|0|0|||||0|0|0|0|0|0|0|0|||||0|0|0|0|0|0|0|0|||||0|0|0|0|0|0|0|0||0||0|0|0|0|0||0|0|0||0|0|0|0|0||0|0|0||0|
0|0|0|0||0|0|0||0|0|0|0|0|0||0|0|0||0|0|0|0|0|0||0|0|0||0|0|0|0|0|0||0|0|0||0|0|0|0|0|0||0|0|0||0|0|0|0|0|0||0|0|0||0|0|0|
0|0||0|0|0||0|0|0|0|0||0|||||||316530200000|0|0|0|1|73253560
```

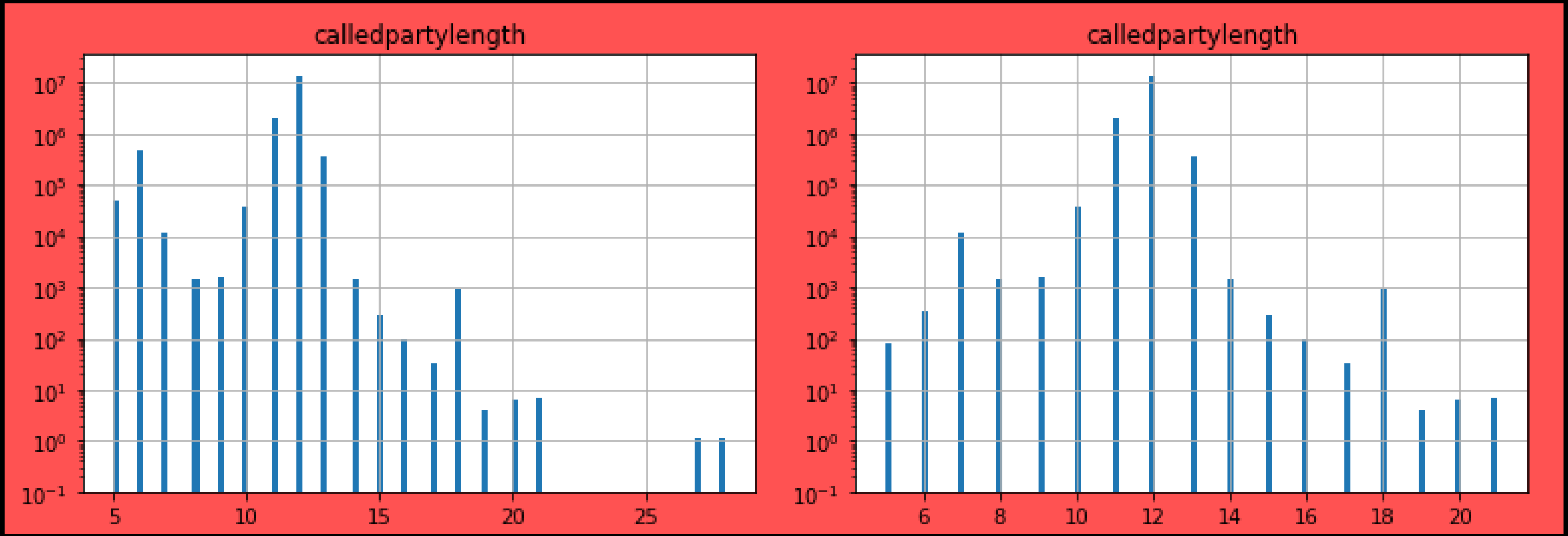
An Example Call Detail Record

704000001796849823|0|20190504205402|1|31616531654|31624586868|31616531654||
0624586868||1|00|316530200000|204083309537469|||20190504215245|8|0|0|20190504215254|0|
68|1|3045434133304344333 7||120||||1287862183|310008|31616531654|1|0|20190501|0|0|0|0||31|
99|1|31|99|99999|31|99|2|31|99|2|310008|1000000000000000000|0|0|0||||1000000|
102779300616118593|2||
72|0|120|179880|0|0|
0|0|0|0|0|0|0|0|0|0|0|0|0|0|
90501000000||||0|0|
0|0|0|0|0||0|0|0|0|0|0|0|0|0
ocg2;1556999564;699692
11000|11|
102779300616118593|10
000060964541NLD|
1003|1|0|0|0|0|0|0|0|0|0|50
000060964541NLD|
0|0|0|||0|0|||0|0||||0|||0|0|||0
704000000967825550|1003|0|179880|319903|||0|0|0|0|0|0|0||||0|0|0|0|0|0|0||||0|0|0|0|0|0|0||||0|0|0|
0|0|0|0|||||0|0|0|0|0|0|0|0||||0|0|0|0|0|0|0||||0|0|0|0|0|0|0||||0|0|0|0|0|0|0||||0|0|0|0|0|0|0||||0|0|0|0|0|0|
0|||||0|0|0|0|0|0|0|0||||0|0|0|0|0|0|0|0||||0|0|0|0|0|0|0|0||||0|0|0|0|0|0|0|0||0||0|0|0|0|0|0||0|0|0||0|0|0|0|0|0||0|0|0||0|
0|0|0|0||0|0|0||0|0|0|0|0|0||0|0|0||0|0|0|0|0|0||0|0|0||0|0|0|0|0|0||0|0|0||0|0|0|0|0|0||0|0|0||0|0|0|0|0|0||0|0|0||0|0|0|
0|0||0|0|0||0|0|0|0|0|0||0||||||316530200000|0|0|0|1|73253560

Timestamp	2019-05-04 21:52:54
A-number	+31616531654
B-number	+31624586868
Duration	68 seconds

Data preparation

- A significant amount of time was spent analyzing and preparing the data
- Removing calls to non-numeric or too long B-numbers
- Filtering out calls to short numbers, like IVR and CS as these are certainly not fraud and may skew the results (many A-numbers calling a few B-numbers)



Feature Engineering

- Creating the variables used to train the machine learning model

- Features that could be used for detecting Revenue Share Fraud
  - Time of day / day of week
  - Count of different A-numbers calling a B-number range within a given time window
  - Ratio of A- to B-numbers
  - Average call duration / standard deviation

Using built-in algorithms in Amazon SageMaker

- Unsupervised learning for anomaly detection
  - Algorithm used: Random Cut Forest
  - Actual (previously unknown) fraud detected!

- Supervised learning using our labeled dataset
  - A needle in a hay-stack: only 1 in every 3000 calls is considered fraudulent
  - Algorithm used: XGBoost
  - Despite the extreme unbalance, initial results are promising
  - Next step is tuning the model to reduce the number of false negatives

Confusion Matrix		Prediction: Not Fraud	Prediction: Fraud
Actual	Not Fraud	114010	0
Actual	Fraud	417	187

aws SUMMIT

Conclusions

- Using Amazon SageMaker, Lebara could get started with limited prior ML knowledge
- Lebara managed to achieve promising results for detecting telco fraud within days
- Besides continuing work on the fraud detection use case, we are looking at applying ML in other areas as well

Hands-on with Amazon SageMaker

The Amazon SageMaker API

- Python SDK orchestrating all Amazon SageMaker activity
  - High-level objects for algorithm selection, training, deploying, model tuning, etc.
  - Spark SDK too (Python & Scala)

- AWS SDK
  - For scripting and automation
  - CLI : '*aws sagemaker*'
  - Language SDKs: boto3, etc.

Demo:

Automatic Model Tuning with XGBoost

https://gitlab.com/juliensimon/ent321/blob/master/ENT321%20-%20short%20version.ipynb

Getting started

http://aws.amazon.com/free

https://aws.amazon.com/sagemaker

https://github.com/aws/sagemaker-python-sdk
https://github.com/aws/sagemaker-spark
https://github.com/awslabs/amazon-sagemaker-examples
https://gitlab.com/juliensimon/ent321

https://medium.com/@julsimon
https://gitlab.com/juliensimon/dlnotebooks
https://gitlab.com/juliensimon/dlcontainers

Thank you!

Julien Simon
Global Evangelist, AI & Machine Learning, AWS
@julsimon

Lars
Hoogweg
CTO, Lebara

Please complete the session survey.

AWS SUMMIT