cloudera

Machine Learning
From idea to productization

Common workflow

Input

Data

Learning

Model training

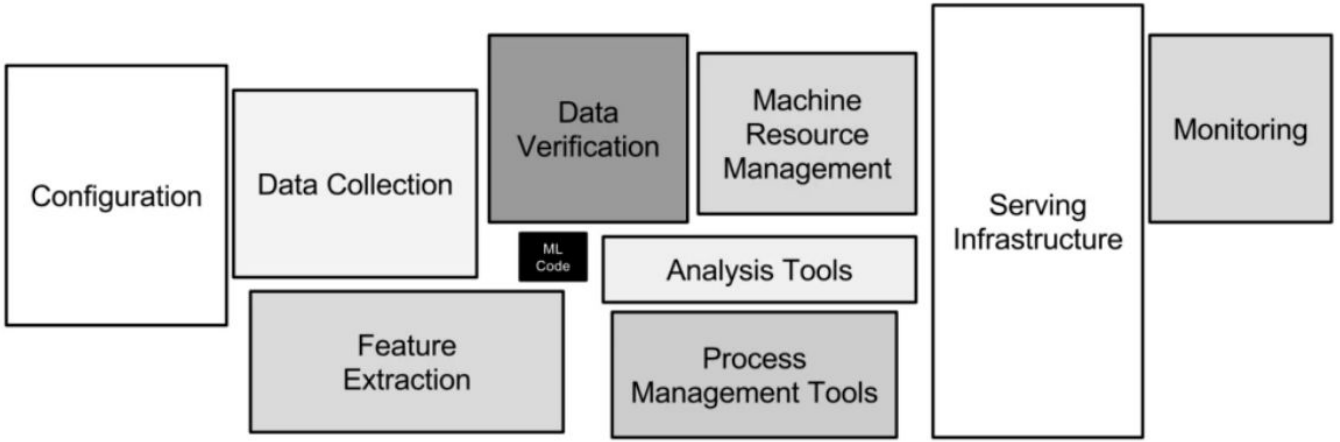
Output

Deployment

What's the problem

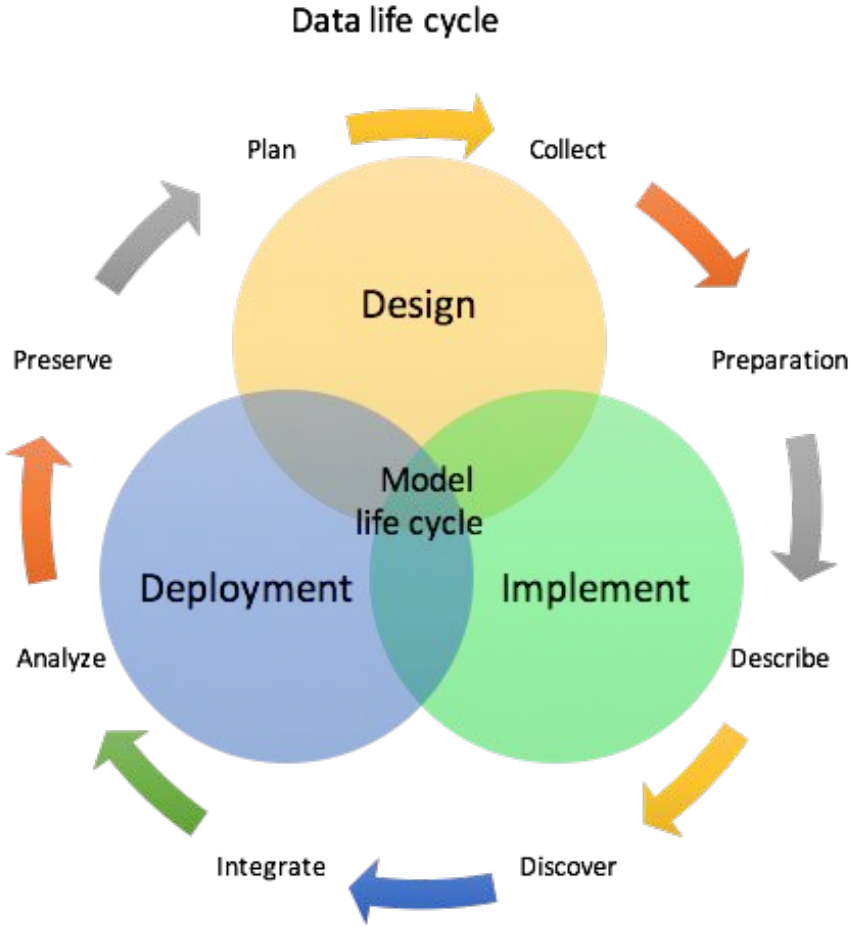
Hidden Technical Debt in Machine Learning Systems

D. Sculley, Gary Holt, Daniel Golovin, Eugene Davydov, Todd Phillips
{dsculley,gholt,dgg,edavydov,toddphillips}@google.com
Google, Inc.



Sculley et al., Hidden Technical Debt in Machine Learning Systems, 2015.

Challenge: Data driven



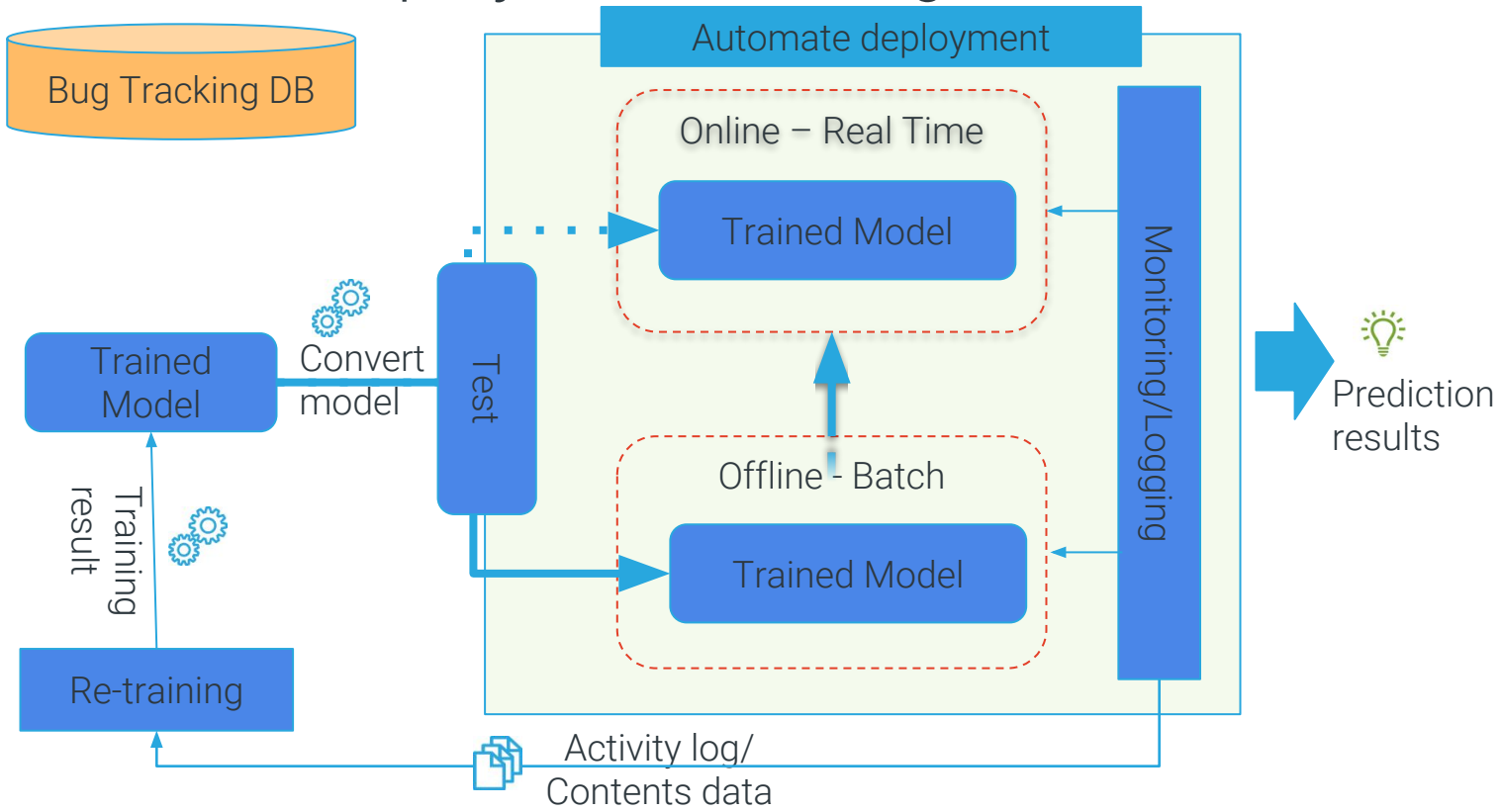
How to improve?

1. First, what's the objective of ML system
   - Responsive and reliable
   - Adaptive and scalable
   - Manageable and reproducible

2. Let's start with deployment and work backwards to figure out what steps are needed for production

https://www.slideshare.net/AmazonWebServices/cmp305-deep-learning-on-aws-made-easycmp305

Deployment/Serving



https://eng.uber.com/michelangelo/

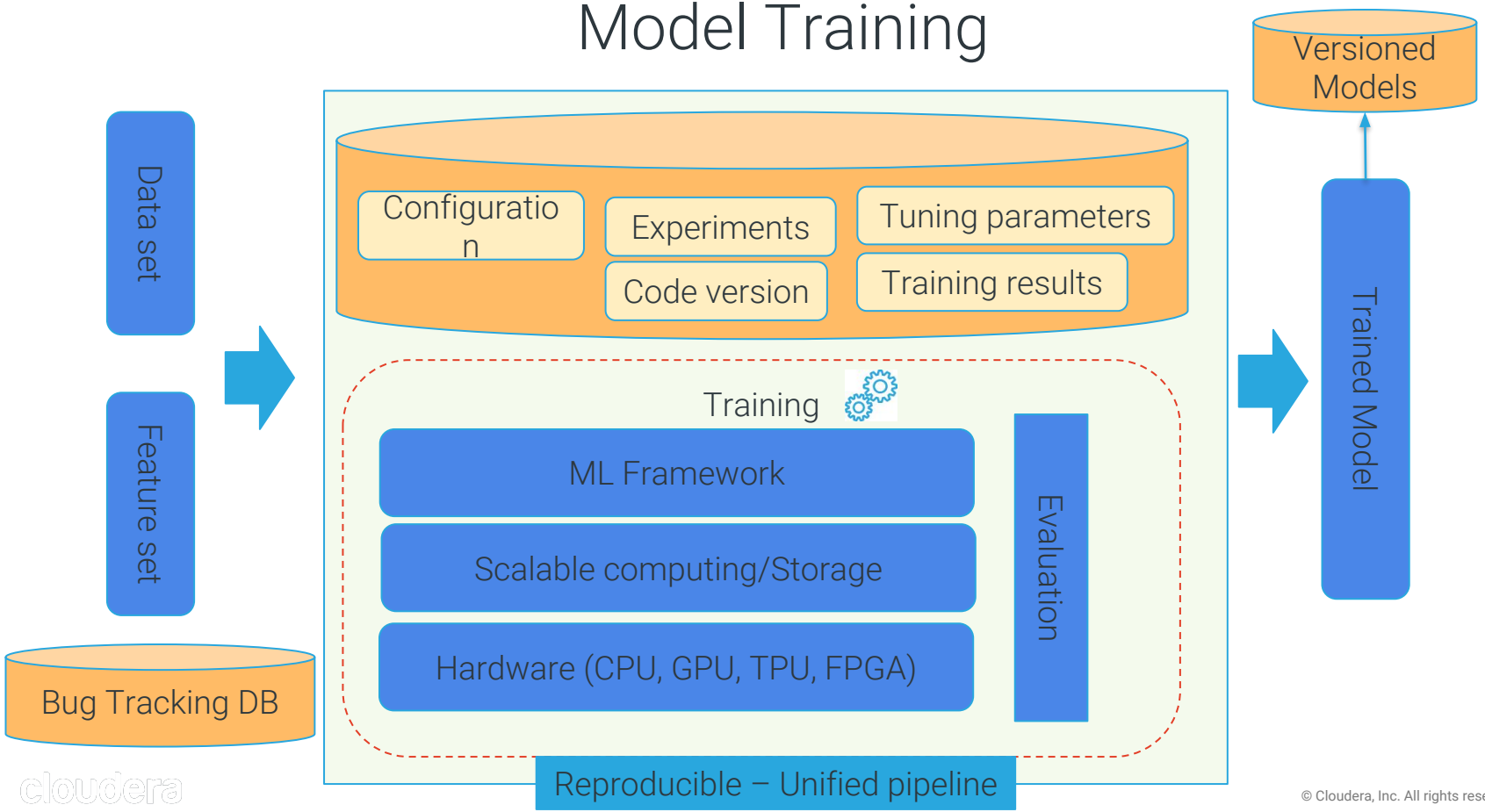
Deployment/Serving

- Responsive and reliable
  - Online or Offline serving
  - Testing
  - Monitoring
  - Feedback loop

- Adaptive and scalable
  - Trade Off between Model quality and Model freshness
  - Deployment frequency

- Manageable and reproducible
  - Continuous improve based on feedback
  - Model versioning

Model Training



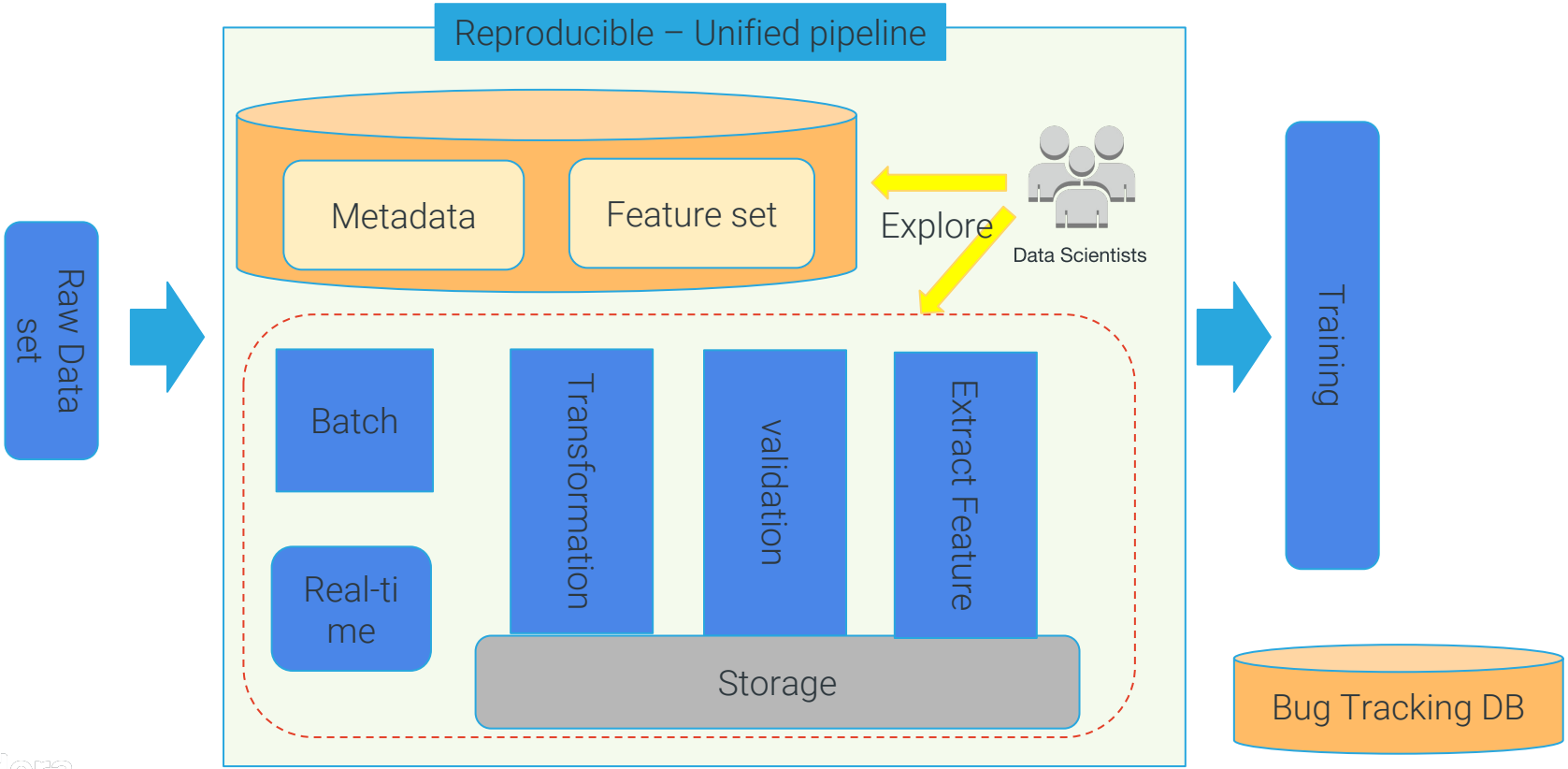
Model Training

Reliable
- Unified process to deploy model
- Mandatory Test/validation
- Regular review
- Model Evaluation

Adaptive
- Consider different training approaches depend on use cases
  - Deep or shallow models
  - Warm-start
  - Early stopping
- Scale out
  - Elastic compute environment
  - Separate compute and storage

Manageable & Reproducible
- Track experiments/models
  - Model configuration
  - Feature set
  - Parameters
  - Training Datasets
  - Training env, hardware, libraries
  - Model versioning
  - Statistics, accuracy
- Track bugs

Data Preparation



Data – Feature Engineering

- Data need to be ingested, cleaned, transformed/normalized, and labeled. This should be done with Consistent pipeline
- Start with original data opposed to deep features

- Document feature set
- Track statistics of the data, inspect data regularly

- Storage consideration, different requirement for different data format

- Privacy and governance

Data – Metadata Management

Manageable & Reproducible

- Metadata management that allow
  - Easy discovery to find available dataset
  - Data re-use and sharing
- Bug tracking for any error found in data pipeline
- Track statistics of the data, as well as manually inspect the data on occasion

- The 5 W's
  - Who
  - When
  - Why
  - What
  - Where

- Sample schema
  - Name
  - Location
  - Format
  - Version
  - Schema
  - Statistics
  - Histograms
  - Annotations

Sample schema to track experiments

Who trained the model

Code version (e.g. git hashes)

Start and end time of the training job

Full model configuration (features used, hyper-parameter values, etc.)

Running environment (e.g. gpu model, docker image)

Reference to training and test data sets

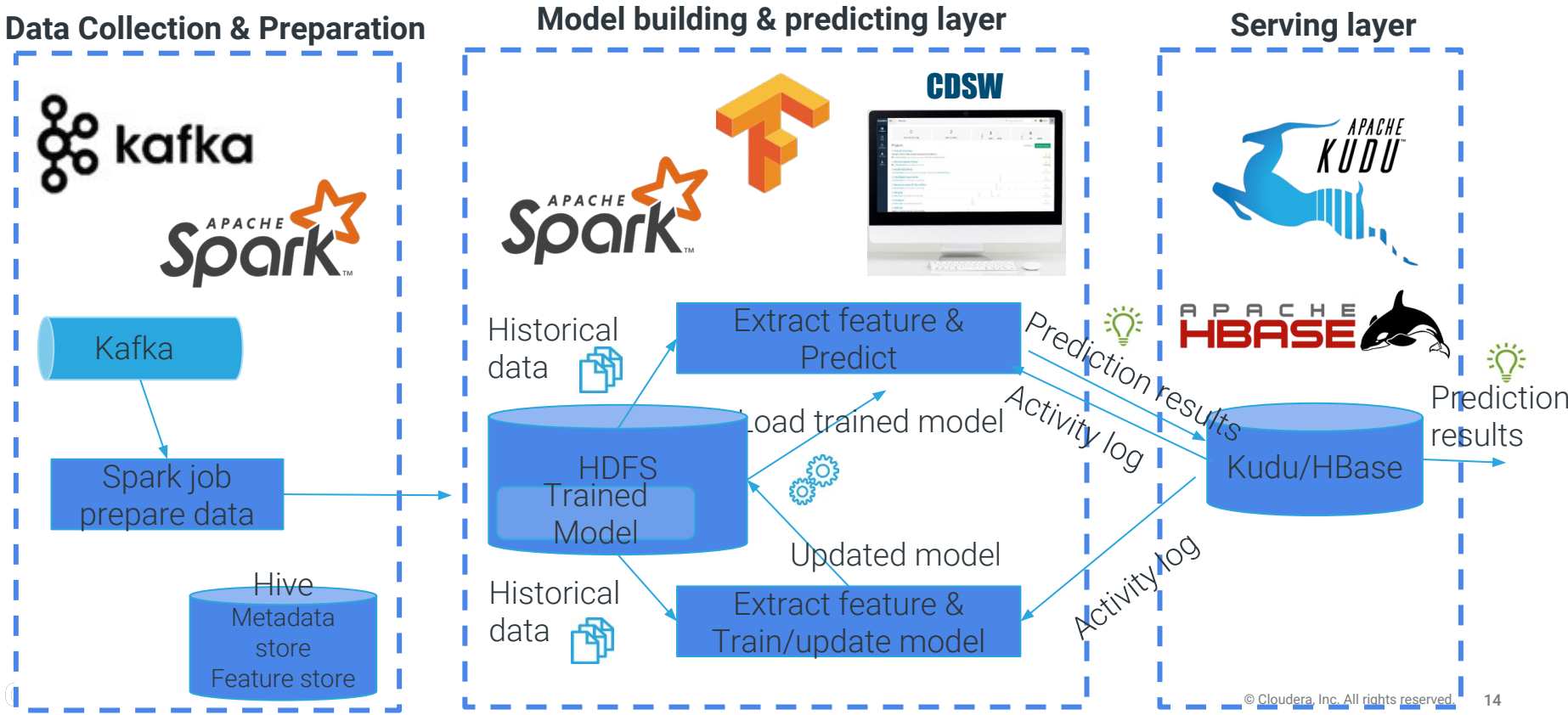
Distribution and relative importance of each feature

Model accuracy metrics

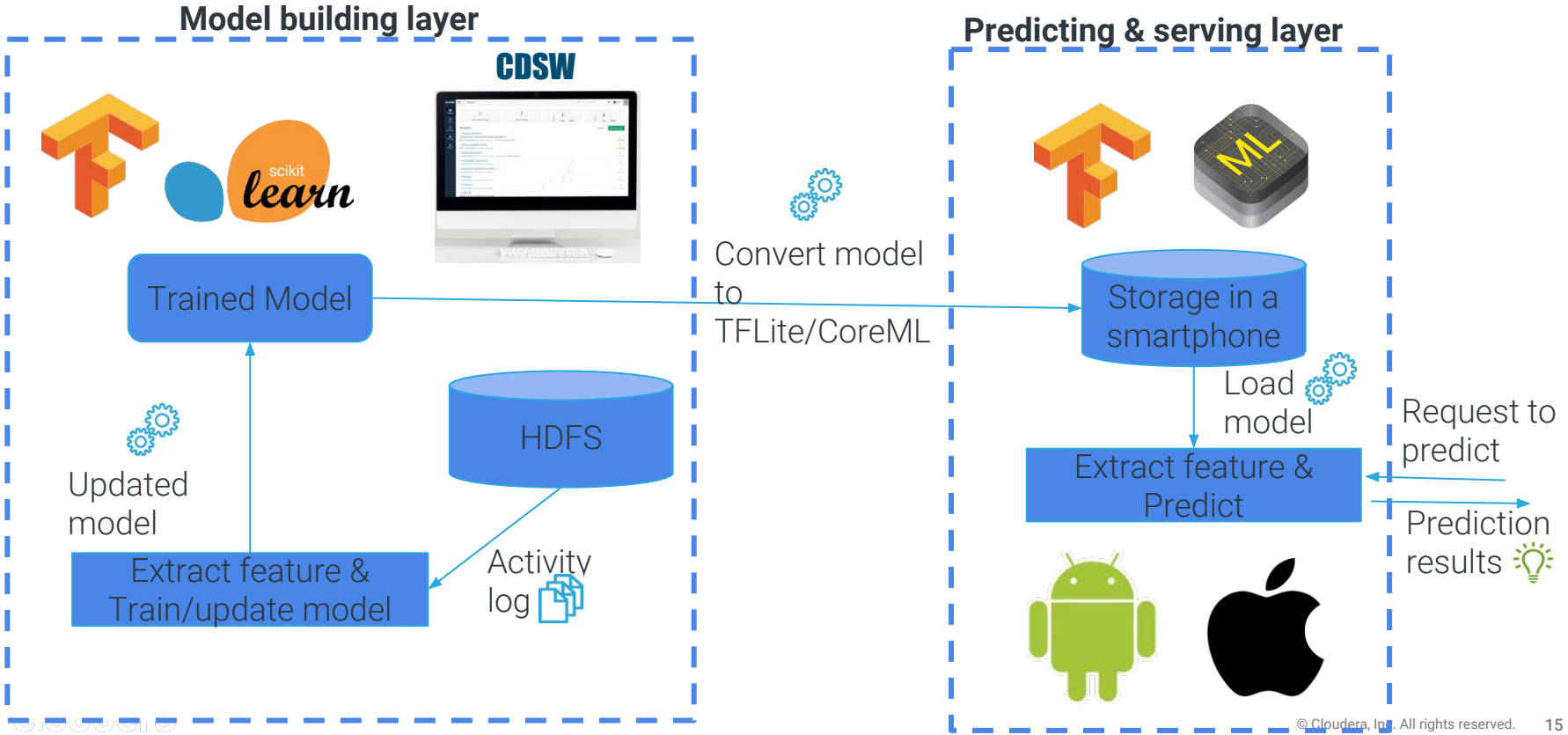
Full learned parameters of the model

Results of the experiment, according to the predefined evaluation measure

Example architecture: Batch serving



Example architecture: Serving on a mobile app



How to achieve this

- Try to build a mini ML platform initially
  - Start simple, easiest use case, get data the fastest way
  - At beginning, consider elastic compute environments take advantage of its elasticity and agility
  - Due to the large data set nature and burst computing need, better to separate storage from compute power to save cost.

- Iteratively improve the platform, expand it by adding missing steps one by one
- Reuse mature toolkits, existing models

- Targeting to build ML as a service for internal users, iteratively automate steps, reuse building blocks

Carlton E. Sapp, Preparing and Architecting for Machine Learning, 2017.

Balance between Tech debt and Agility

- Start small, use the fastest way to get data
- Add process only if you have to repeat it more than 3 times
- Review process and cleanup regularly, remove unused features, configurations, code…
  - Dead code
  - Messy configuration
  - Duplicate/irrelevant features

Sculley et al., Hidden Technical Debt in Machine Learning Systems, 2015

Use Joel test checklist to review your ML system

1. Can new hires setup environment and run analysis in a few days?
2. Can past experiments be easily reproducible (metadata, configuration, feature set, parameters)?
3. Is there a single place to search reusable data sets, code, etc.?
4. Can deployment pipeline be run by data scientists easily?
5. Is there a system to track bugs end to end (ingestion pipeline, unrelated feature, bad model...)?
6. Can data scientists collaborate with each other other than emails?
7. Is there frequent review for dead code path, configuration, pipeline…?
8. Is there a process for detecting bugs (evaluation step)?
9. Do data scientists have access to scalable compute and storage?

https://www.joelonsoftware.com/2000/08/09/the-joel-test-12-steps-to-better-code/

Come up with your own Joel test checklist

References and Recommended Reading

- Hidden Technical Debt in Machine Learning Systems
- Rules of Machine Learning: Best Practices for ML Engineering
- Automatically Tracking Metadata and Provenance of Machine Learning Experiments
- Uber Michelangelo
- Google TFX
- Deep learning on AWS made easy
- Gartner: Preparing and Architecting for Machine Learning
- The Joel Test: 12 steps to better code

Thank you, questions?

cloudera