COMCAST

Data Governance Using Apache Avro Makes Feature Engineering (and a lot more) Easy

Barbara Eckman

Gabriel Commeau

Outline

- Common pain points for modelers
  - Data Discovery and Interpretation is Hard
  - Moving from Development to Production is Hard
  - Taking advantage of teammates' work is Hard
  - Understanding evolution of modeling pipeline is Hard
- How to address them!
  - Metadata repo for discovery and documentation of data and its lineage
  - Schema-driven data processors for automatic feature preparation and data quality control
  - Metadata repo for discovery and documentation of models
  - Metadata lineage to capture model evolution

COMCAST

Common Pain Points for Modelers

Data Discovery and Interpretation is Hard

- Does your job involve integrating data across corporate silos/verticals?
- Do you spend more time finding and reformatting data than you do analyzing it?
- When you attempt to integrate your data with another team’s data, are you uncertain about what the other team’s data means?
- Are you worried that in joining the two datasets, you may be creating “Frankendata”?
- Does your Big Data ecosystem go beyond a single hadoop provider, or even include public cloud and on-prem?

COMCAST

Moving from Development to Production

- Do you get a bunch of ad-hoc requests, and a few of them occasionally become a cornerstone of some production system?
- Do you have a standard method to transition from data scientists engineering features on their laptop to data engineers productionizing data pipelines?
- Do you rewrite (or copy/paste) data transformation code over and over again?
- Do you have a standard method to ensure your feature engineering is sturdy and well tested?

COMCAST

Taking advantage of teammates’ work

- Have you ever worked up a model and features, only to find out that a colleague has already worked up something very similar?
- Would an easy way of discovering similar modeling activities across the team or the enterprise help you do your job better/faster/more efficiently?

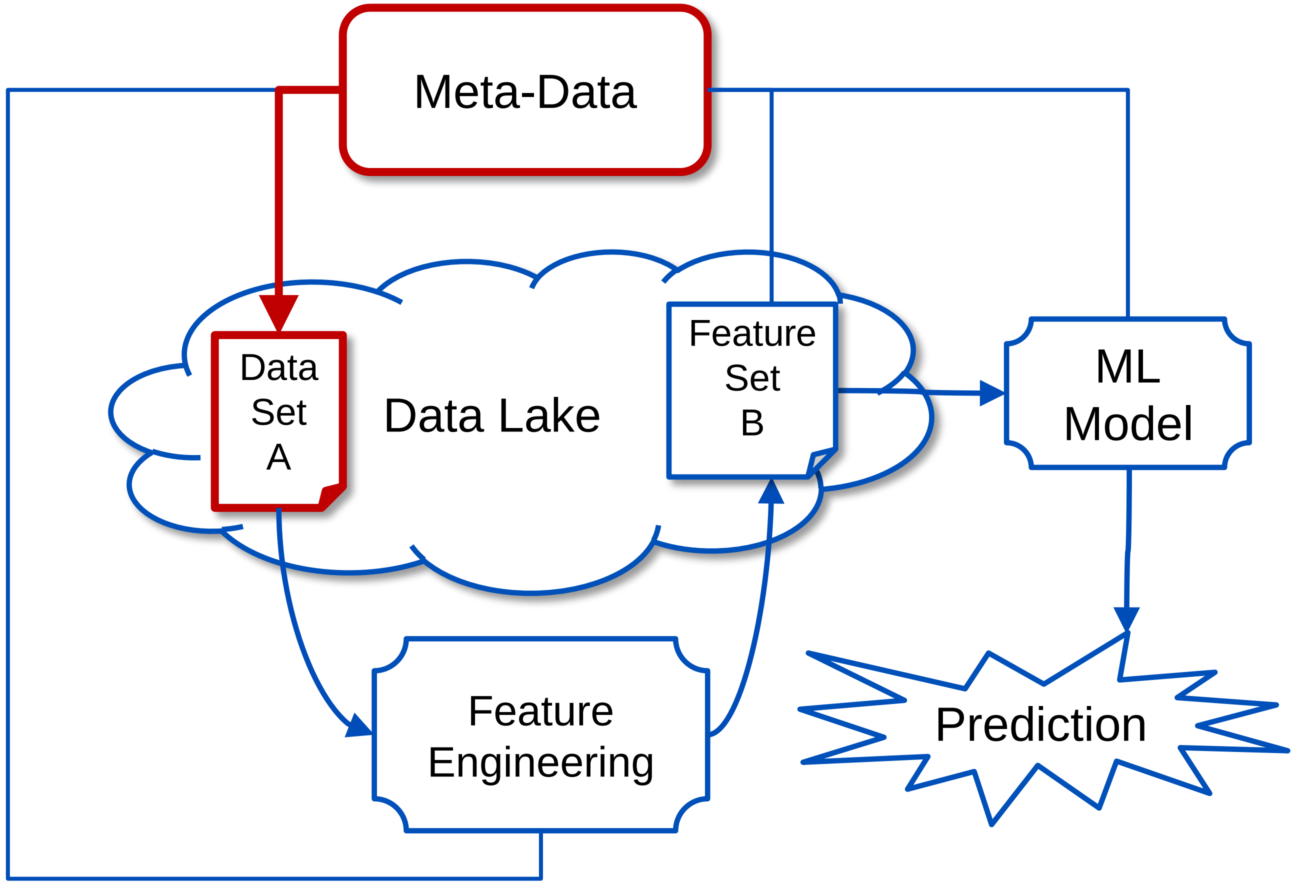
COMCAST

Understanding evolution of modeling pipeline

- Do you always get your model right the first time, or do you occasionally need to iterate and evolve it? ;-)
- Do you have a standard method to manage multiple versions of features and models?
- Do you have a standard method to efficiently keep track of what features go with which model?

How to Address these Pain Points!

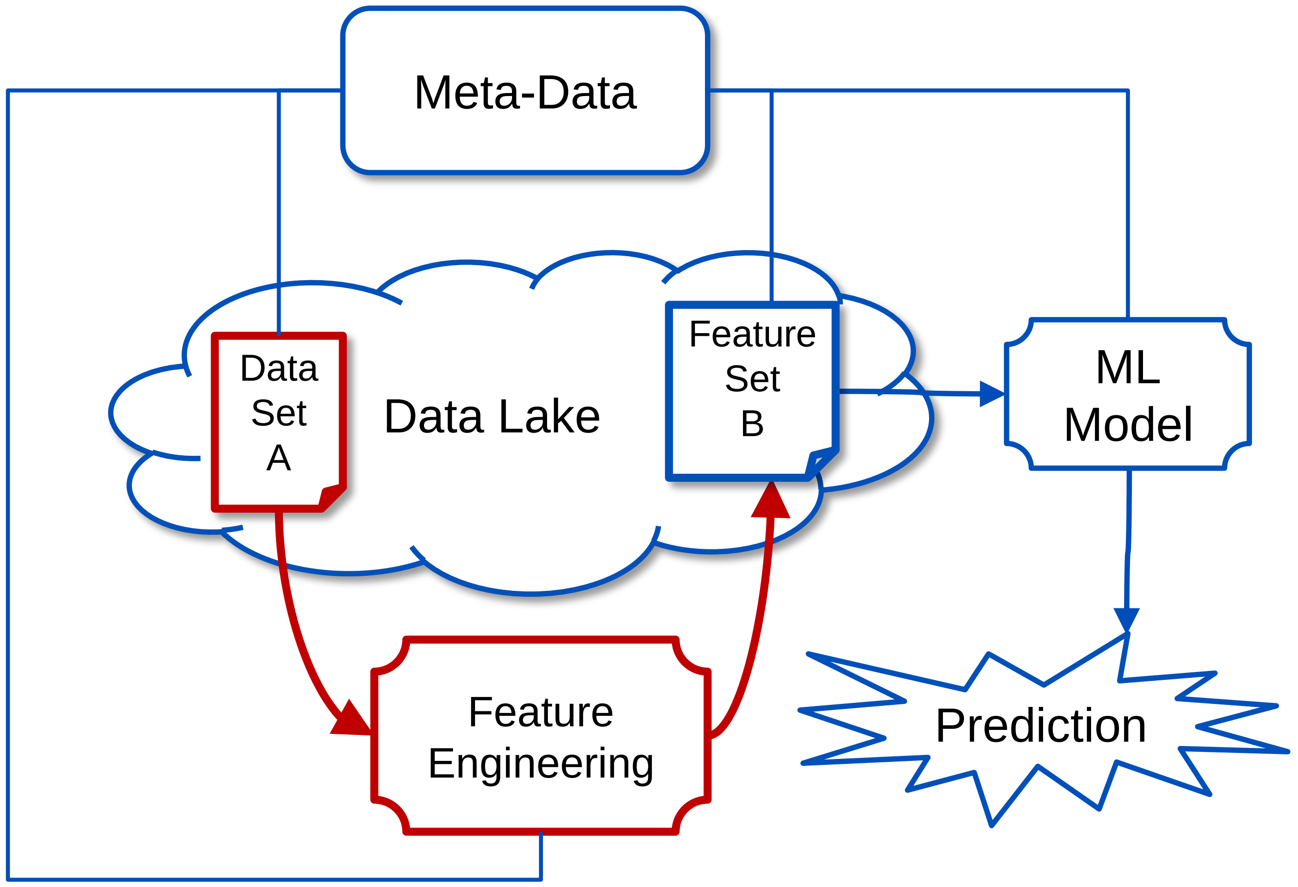
Metadata repo for discovery and documentation of data and its lineage



- Require well-documented schemas on data ingest
  - Use of common schema elements, schema compatibility measures
- End-to-end metadata repository
  - Rich metadata on all datasets (kafka/kinesis topics, RDBMS's, hive tables, object store buckets/files, etc)
  - All datasets associated with schemas they embody
  - Build lineage and metadata capture into the data flow
- Data and Schema lineage
  - Evolution between schema versions
  - Processes that move/transform one dataset to another

COMCAST

Schema-driven data processors for automatic feature preparation and data quality control



- What's a feature actually?
- Generic data analyzer
  - Extended schema definition
- Generic data processors for Avro:
  - Filter
  - Computation
  - Aggregation
- Generic data quality control:
  - Simple data quality checks
  - Advanced data quality checks
- More:
  - Merging datasets
  - Any frequent operation

Schema-driven data processors for automatic feature preparation and data quality control

Some Avro schema:

```
{
    "name": "PAW",
    "type": "record",
    "fields": [
        {"name": "timestamp", "type": "long"},
        {"name": "id", "type": "int"},
        {"name": "data", "type": "string"}
    ]
}
```

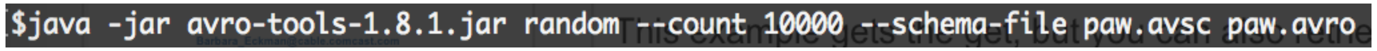
Processing events generically:

```
def process_events(condition, events):
    return filter(condition, events)
```

Creating filter:

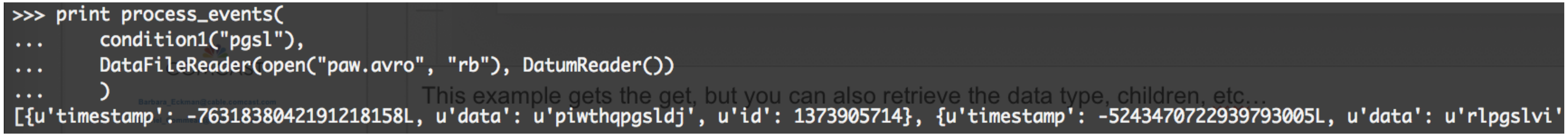
```
def condition1(str_test):
    return lambda event: str_test in event['data'].lower()
```

Generating a random file:



```
$java -jar avro-tools-1.8.1.jar random --count 10000 --schema-file paw.avsc paw.avro
```

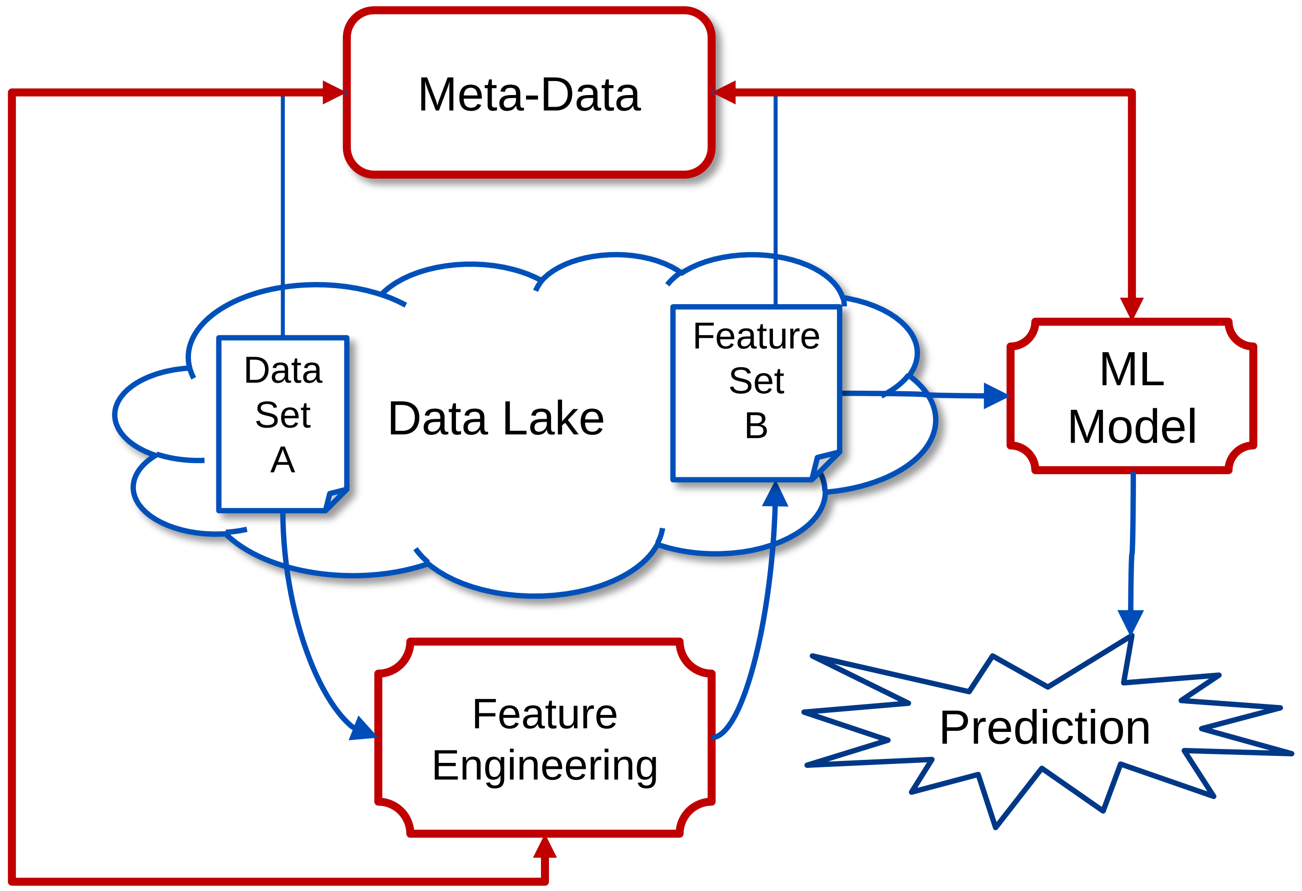
Demo!



```
>>> print process_events(
...     condition1("pgsl"),
...     DataFileReader(open("paw.avro", "rb"), DatumReader())
...     )
[{u'timestamp': -7631838042191218158L, u'data': u'piwthqpgsldj', u'id': 1373905714}, {u'timestamp': -5243470722939793005L, u'data': u'rlpgslvi'
```

COMCAST

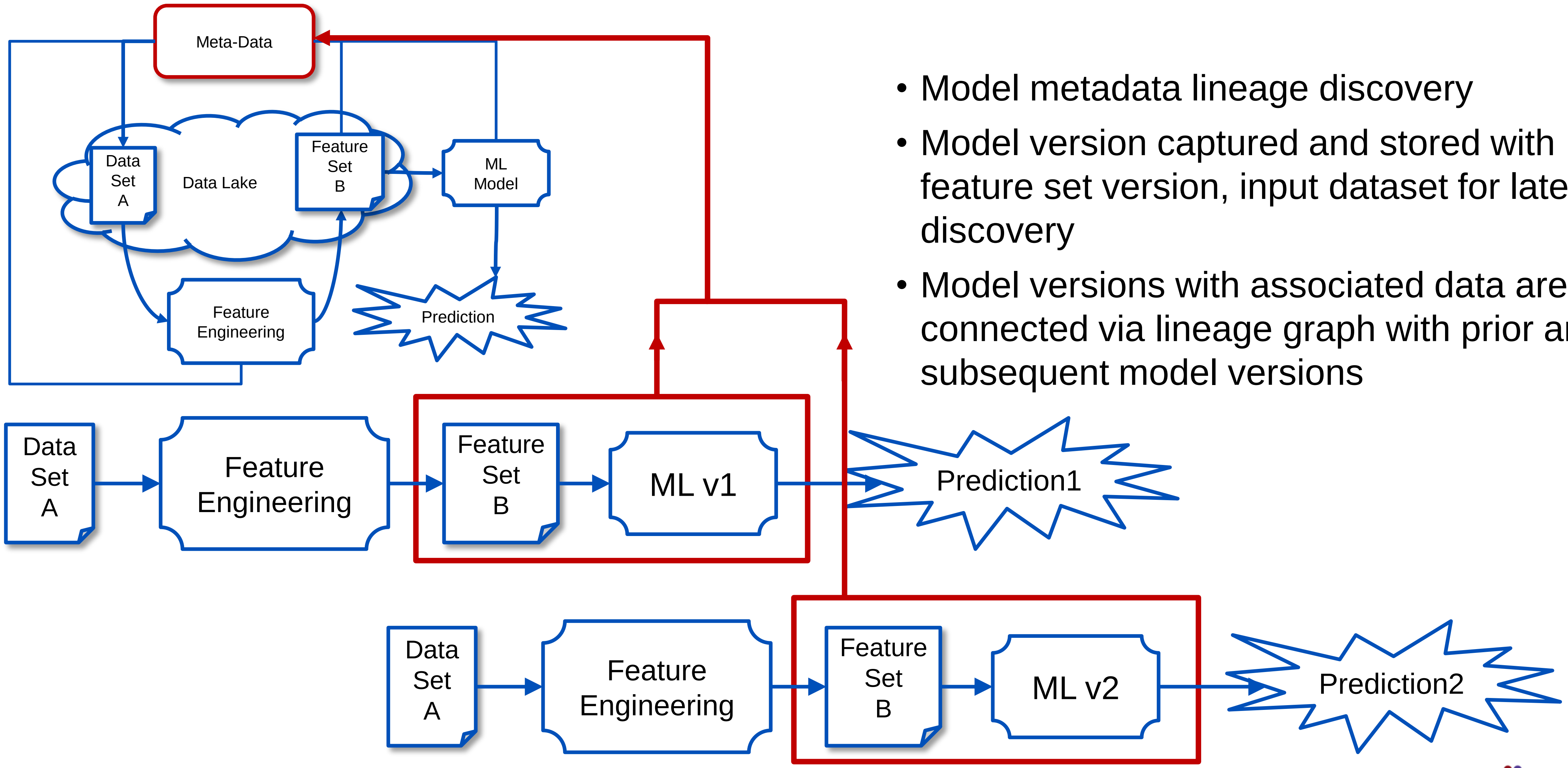
Metadata repo for discovery and documentation of models



- End-to-end metadata repository
  - Models are first-class objects, captured with rich metadata (eg input file schema, feature set schema, model parameters, etc)
  - Feature engineering jobs are first-class objects, captured with rich metadata (eg model, data quality threshold, input file schema, owner)
  - Build metadata capture on models and feature engineering jobs into the ML pipeline

COMCAST

Metadata lineage for capturing model evolution



- Model metadata lineage discovery
- Model version captured and stored with feature set version, input dataset for later discovery
- Model versions with associated data are connected via lineage graph with prior and subsequent model versions

Data Governance Using Apache Avro Makes Feature Engineering (and a lot more) Easy

- Common pain points for modelers
  - Data Discovery and Interpretation is Hard
  - Moving from Development to Production is Hard
  - Taking advantage of teammates’ work is Hard
  - Understanding evolution of modeling pipeline is Hard
- How to address them!
  - Metadata repo for discovery and documentation of data and its lineage
  - Schema-driven data processors for automatic feature preparation and data quality control
  - Metadata repo for discovery and documentation of models
  - Metadata lineage to capture model evolution

COMCAST

COMCAST

Barbara_Eckman@cable.comcast.com

Gabriel_Commeau@cable.comcast.com