Machine Learning Design Patterns for ML Ops

Valliappa (Lak) Lakshmanan
Global Head of Analytics & AI Solutions,
Google Cloud
Twitter: @lak_gcp
lak@google.com

Google Cloud

ginablaber
@ginablaber

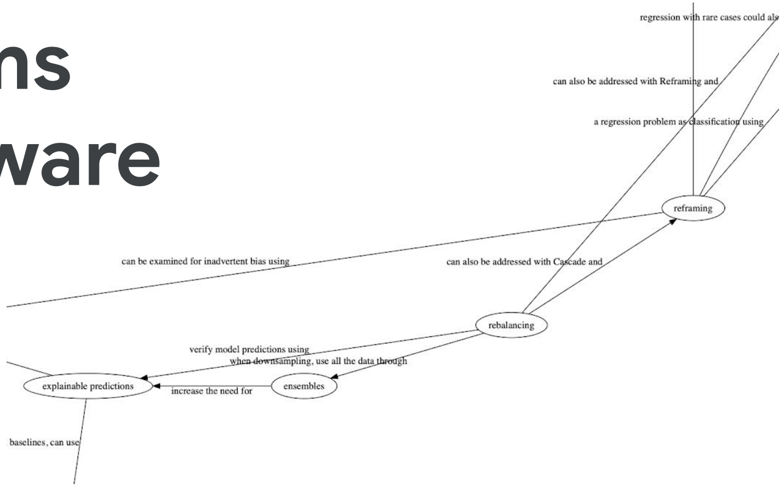
Follow

The story of enterprise Machine Learning: “It took me 3 weeks to develop the model. It’s been >11 months, and it’s still not deployed.” @DineshNirmalIBM #StrataData #strataconf

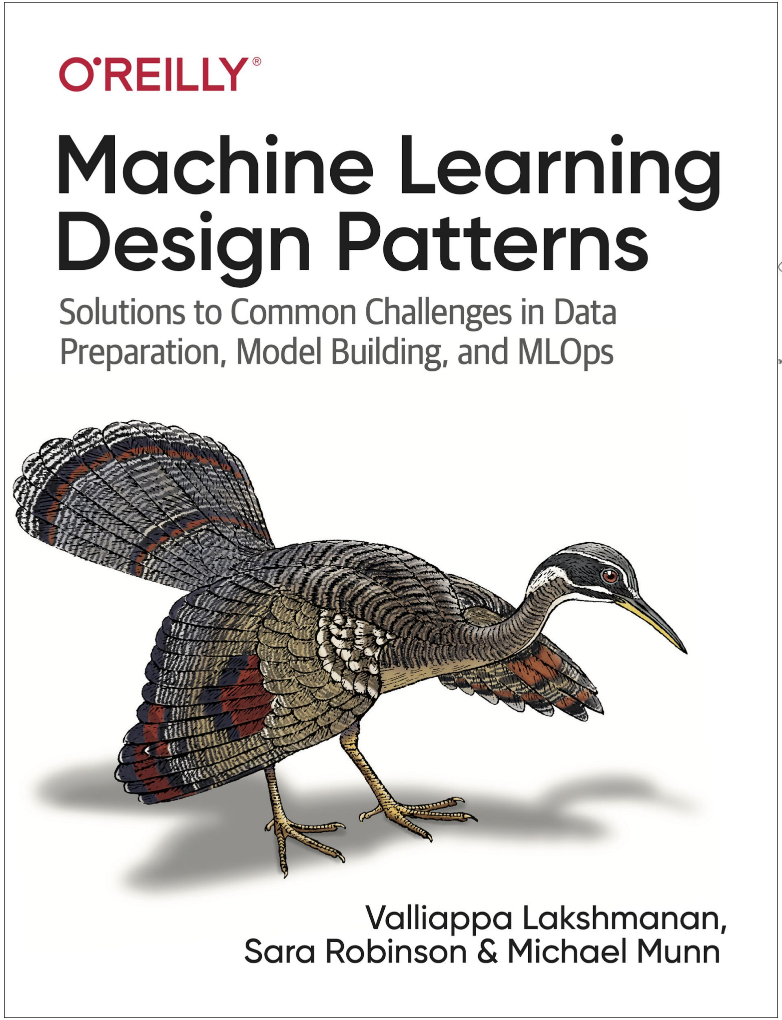
10:19 AM - 7 Mar 2018

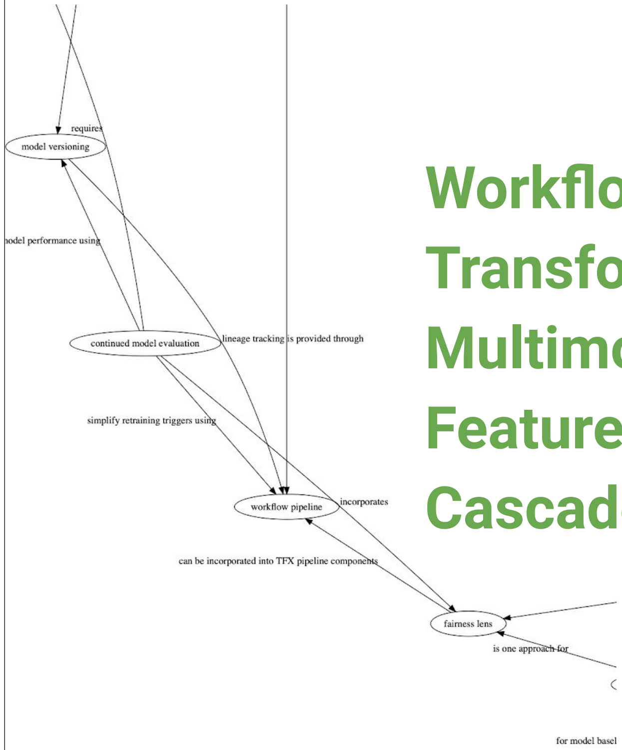
Google Cloud

Design patterns are formalized best practices to solve common problems when designing a software system.



Google Cloud





Workflow Pipeline

Transform

Multimodal Input

Feature Store

Cascade

https://github.com/GoogleCloudPlatform/ml-design-patterns

Workflow Pipeline

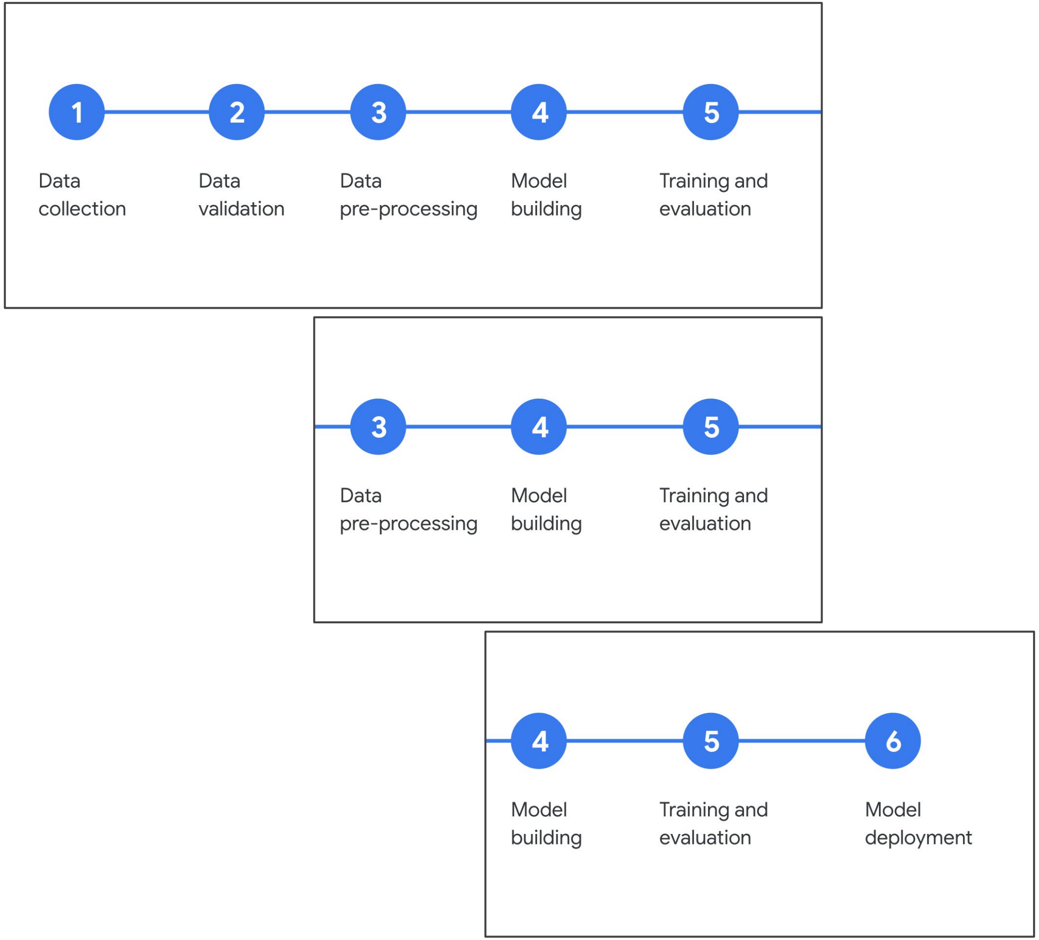
01

Google Cloud

Typical development workflow is monolithic

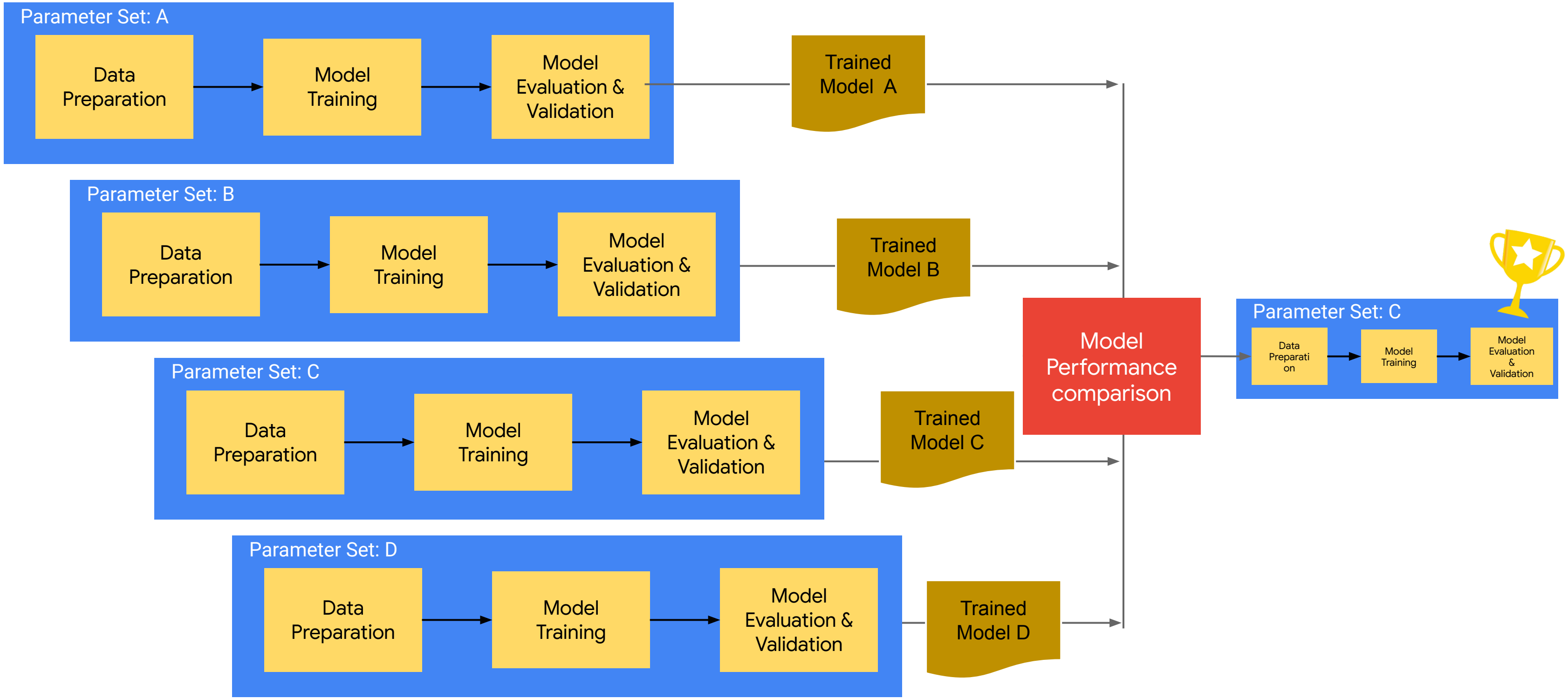
1. Data collection
2. Data validation
3. Data pre-processing
4. Model building
5. Training and evaluation
6. Model deployment

Experiments are ad-hoc and not repeatable



Google Cloud

Hyperparameter tuning has wasteful repetition

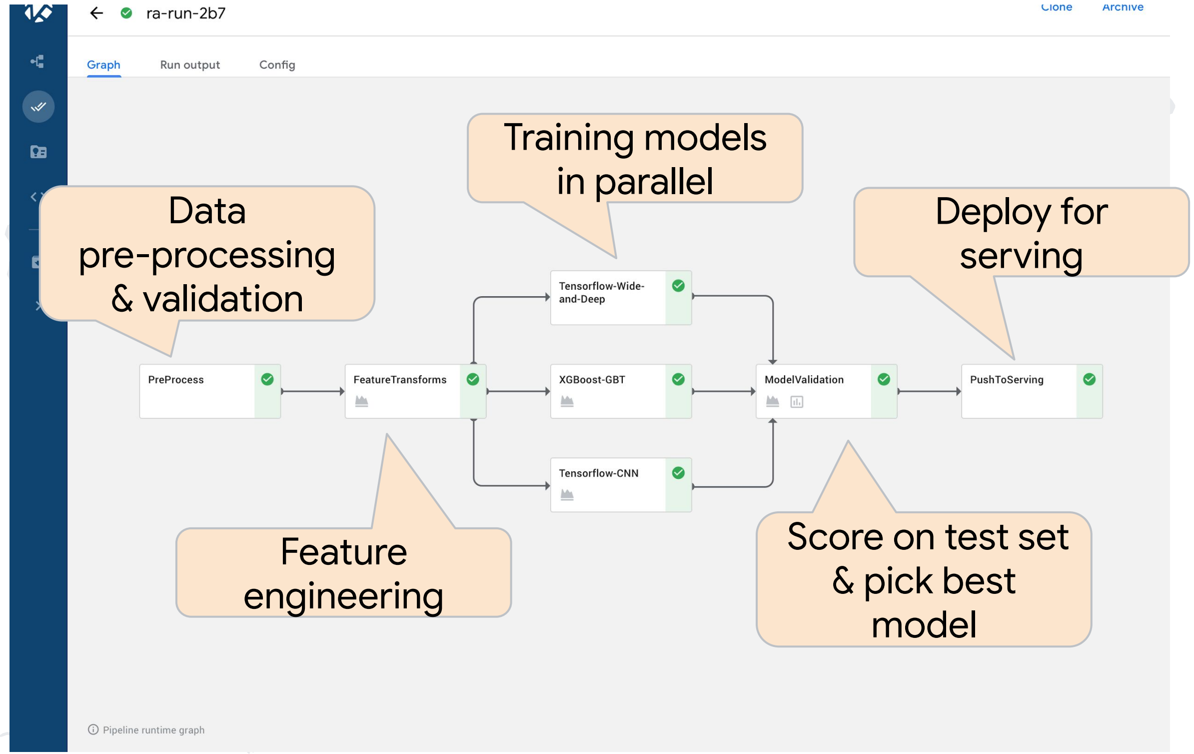


MLOps has some DevOps concepts, and adds data validation & continuous evaluation/training

But MLOps differs from DevOps in important ways

	DevOps	MLOps
1	Test and validate code and components	Also test and validate data, data schemas, and models
2	Focus on a single software package or service	Also consider the whole system, the ML training pipeline
3	Deploy code and move to the next task	Constantly monitor, retrain and serve the model

A pipeline is an executable DAG of ML steps

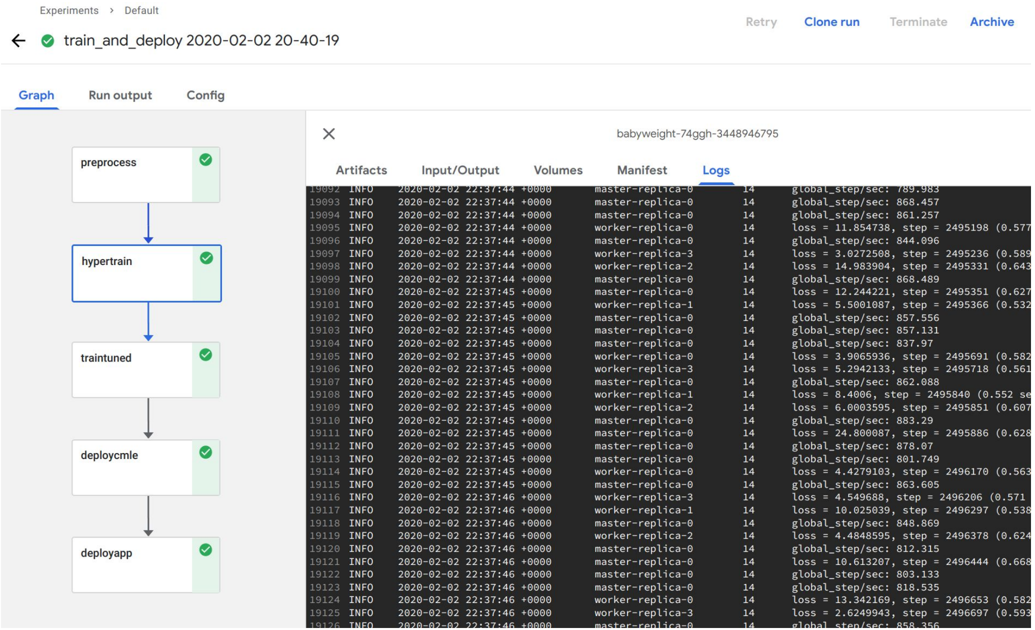


Each step of the pipeline is a container

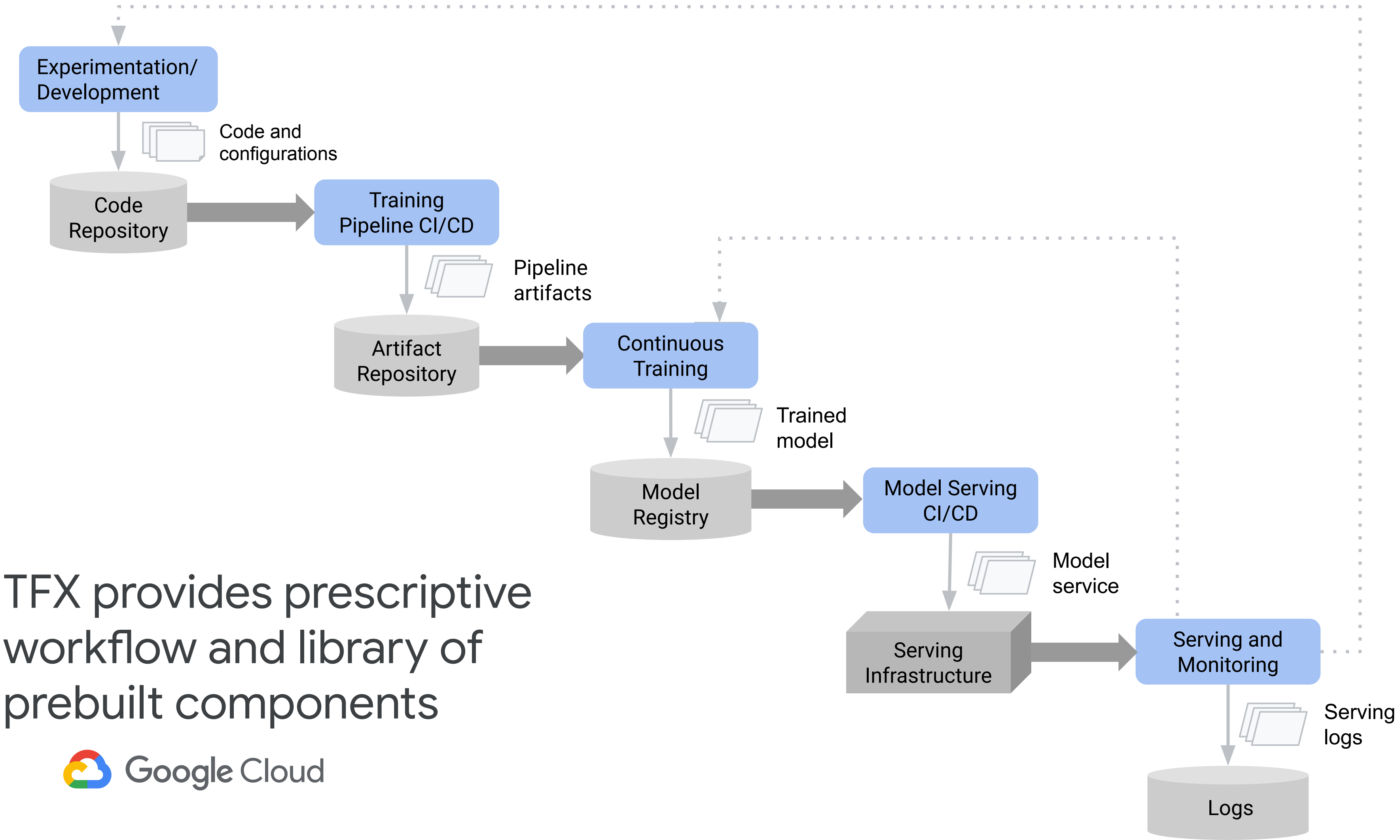
Pipeline runs can be grouped into Experiments

Logs are associated with each step of each run

An artifact repository stores the parameters & data passed between steps



Google Cloud



TFX provides prescriptive workflow and library of prebuilt components

Google Cloud

Workflow Pipeline

Capture ML workflows in a DAG

The pipeline DAG:

- Executable as a whole or in parts
- Can be triggered by events
- Logging and monitoring for each step, run, experiment
- Artifacts stored in repository

Prebuilt components, which:

- Understand training vs. inference
- Avoid rerunning if output artifact already up-to-date in repository
- Run as containers on the pipelines platform

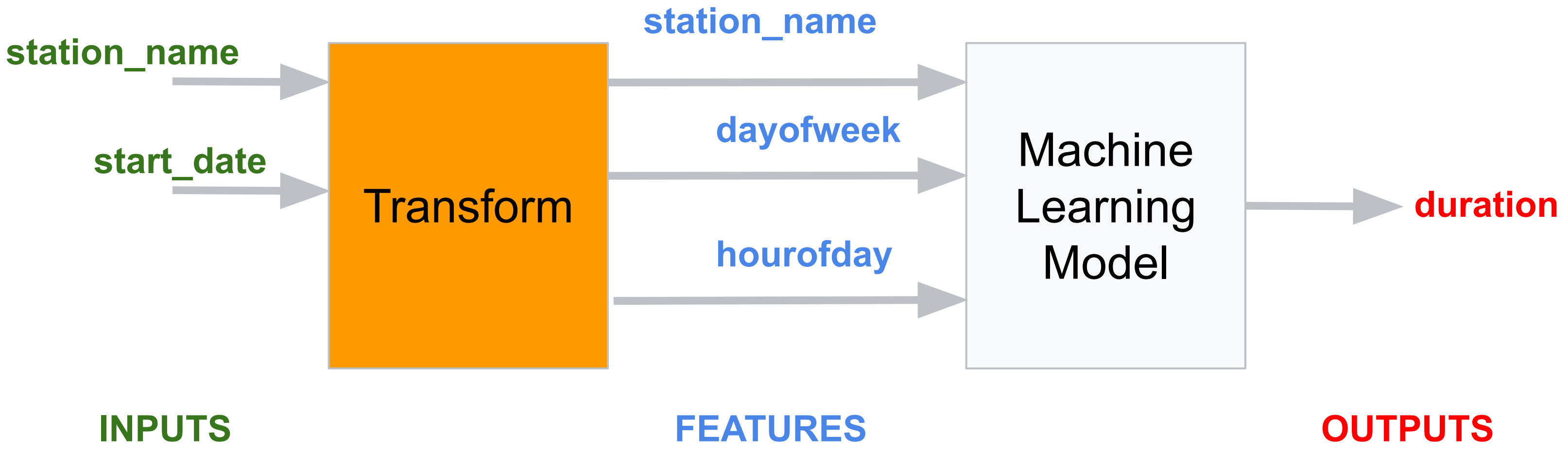
```
pipeline.Pipeline(
      pipeline_name='huricane_prediction',
      pipeline_root='path/to/pipeline/code',
      components=[
          bigquery_gen, statistics_gen, schema_gen, train, model_pusher
      ]
  )
```

```
bigquery_gen = BigQueryExampleGen(query=query)

statistics_gen = StatisticsGen(examples=example_gen.outputs['examples'])

...

model_pusher = Pusher(
      model=trainer.outputs['model'],
      model_blessing=evaluator.outputs['blessing'],
      push_destination=pusher_pb2.PushDestination(
          filesystem=pusher_pb2.PushDestination.Filesystem(
              base_directory=serving_model_dir)))
```

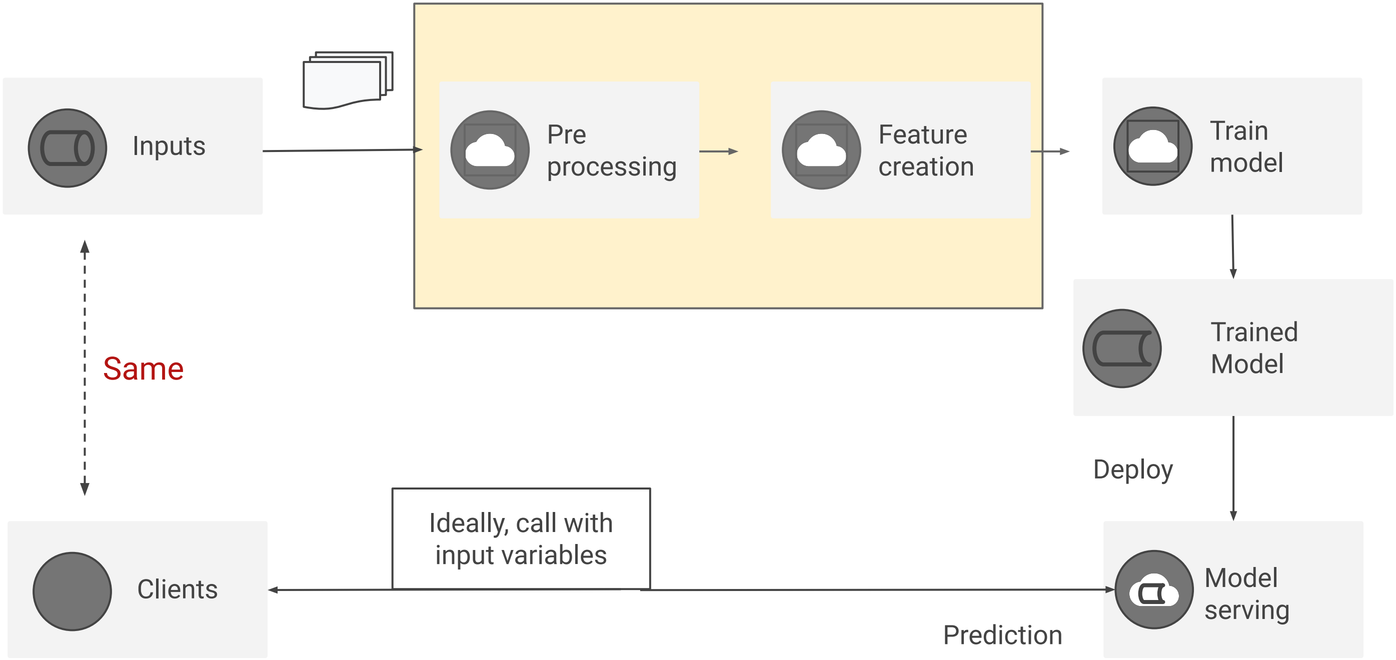
Transform

02

Imagine a model to predict the length of rentals



Who does transformation during prediction?



Ideally, client code does not have to know about all the transformations that were carried out

```
CREATE OR REPLACE MODEL ch09edu.bicycle_model
OPTIONS(input_label_cols=['duration'],
        model_type='linear_reg')
AS

SELECT
  duration
  , start_station_name
  , CAST(EXTRACT(dayofweek from start_date) AS STRING)
        as dayofweek
  , CAST(EXTRACT(hour from start_date) AS STRING)
        as hourofday
FROM
  `bigquery-public-data.london_bicycles.cycle_hire`
```

```
SELECT * FROM ML.PREDICT(MODEL ch09edu.bicycle_model,(
  350 AS duration
  , 'Kings Cross' AS start_station_name
  , '3' as dayofweek
  , '18' as hourofday
))
```

Leading cause of training-serving skew

TRANSFORM ensures transformations are automatically applied during ML.PREDICT

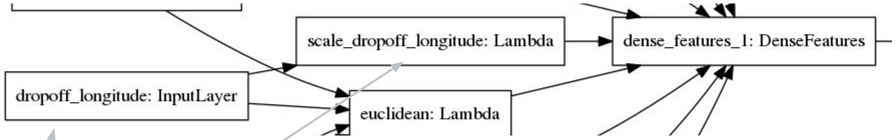
```
CREATE OR REPLACE MODEL ch09edu.bicycle_model
OPTIONS(input_label_cols=['duration'],
        model_type='linear_reg')
AS

SELECT
  duration
  , start_station_name
  , CAST(EXTRACT(dayofweek from start_date) AS STRING)
         as dayofweek
  , CAST(EXTRACT(hour from start_date) AS STRING)
         as hourofday
FROM
  `bigquery-public-data.london_bicycles.cycle_hire`
```

```
SELECT * FROM ML.PREDICT(MODEL ch09edu.bicycle_model,(
  350 AS duration
  , 'Kings Cross' AS start_station_name
  , '3' as dayofweek
  , '18' as hourofday
))
```

```
CREATE OR REPLACE MODEL ch09edu.bicycle_model
OPTIONS(input_label_cols=['duration'],
        model_type='linear_reg')
TRANSFORM(
  SELECT * EXCEPT(start_date)
  , CAST(EXTRACT(dayofweek from start_date) AS STRING)
         as dayofweek
  , CAST(EXTRACT(hour from start_date) AS STRING)
         as hourofday
)
AS
SELECT
  duration, start_station_name, start_date
FROM
  `bigquery-public-data.london_bicycles.cycle_hire`
```

```
SELECT * FROM ML.PREDICT(MODEL ch09edu.bicycle_model,(
  350 AS duration
  , 'Kings Cross' AS start_station_name
  , CURRENT_TIMESTAMP() as start_date
))
```

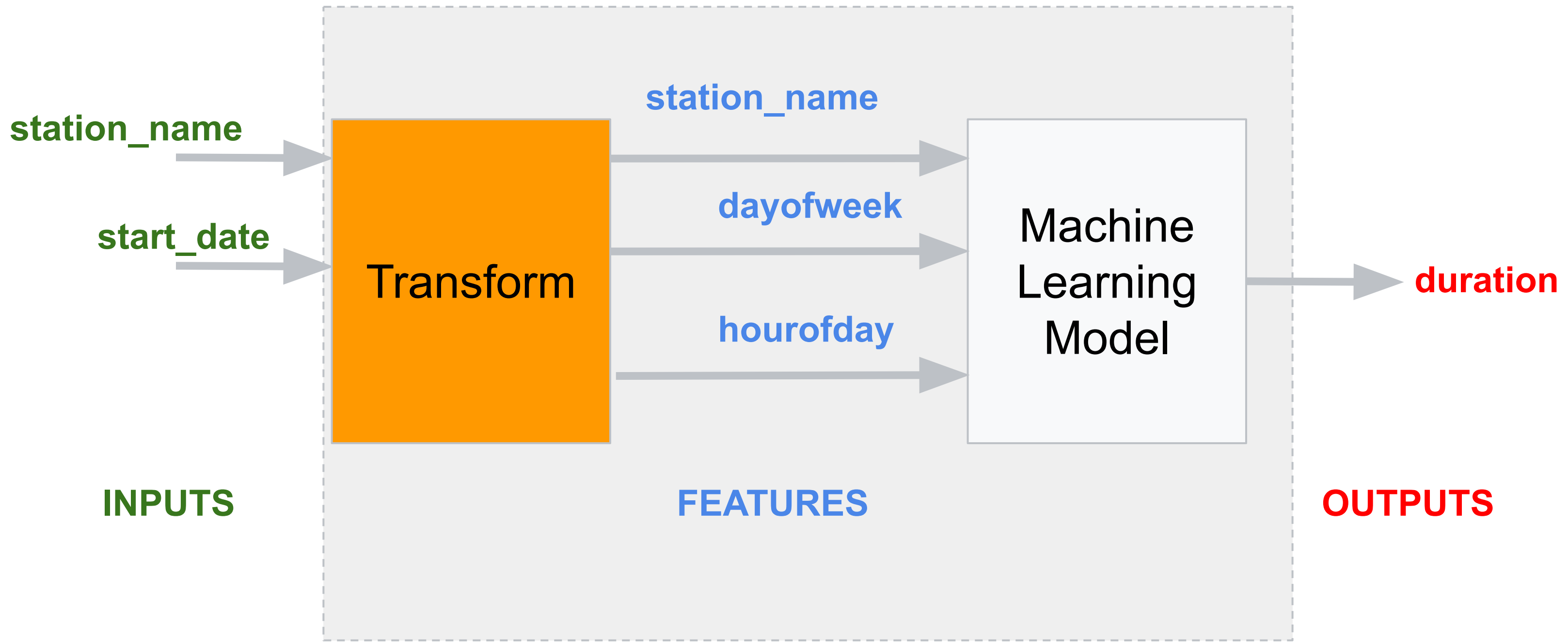
In TensorFlow/Keras, do transformations in Lambda Layers so that they are part of the model graph




```
for lon_col in ['pickup_longitude', 'dropoff_longitude']:  # in range -70 to -78
        transformed[lon_col] = tf.keras.layers.Lambda(
            lambda x: (x+78)/8.0,
            name='scale_{}'.format(lon_col)
        )(inputs[lon_col])
```

Moving an ML model to production is much easier if you keep inputs, features, and transforms separate

Transform pattern: the model graph should include the transformations



Multimodal input

03

Google Cloud

Is this an image classification problem?

The input is multimodal: an image and some structured data

Image feature:

`footage`

Numerical feature:

`time_hour`
10

ML model

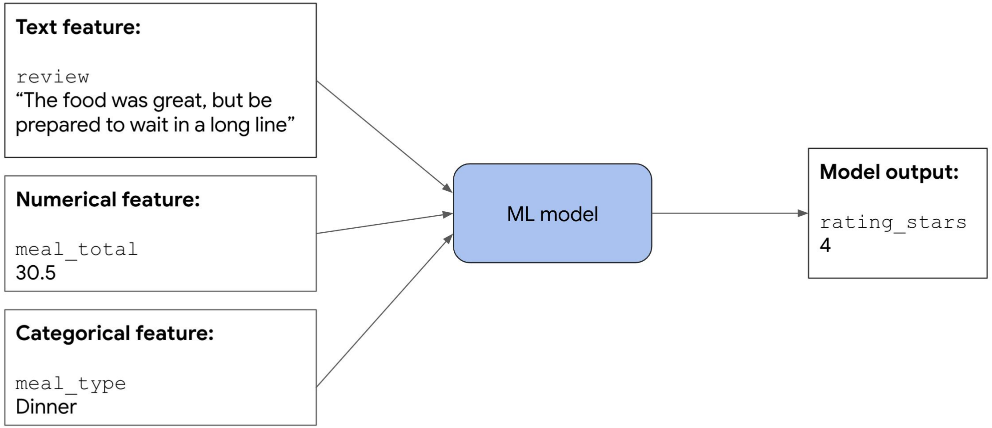
Model output:

`violation`
False

Much clearer in the case of free form text in tabular data

There are multiple ways of representing the text feature (length, sentiment, language, etc.)

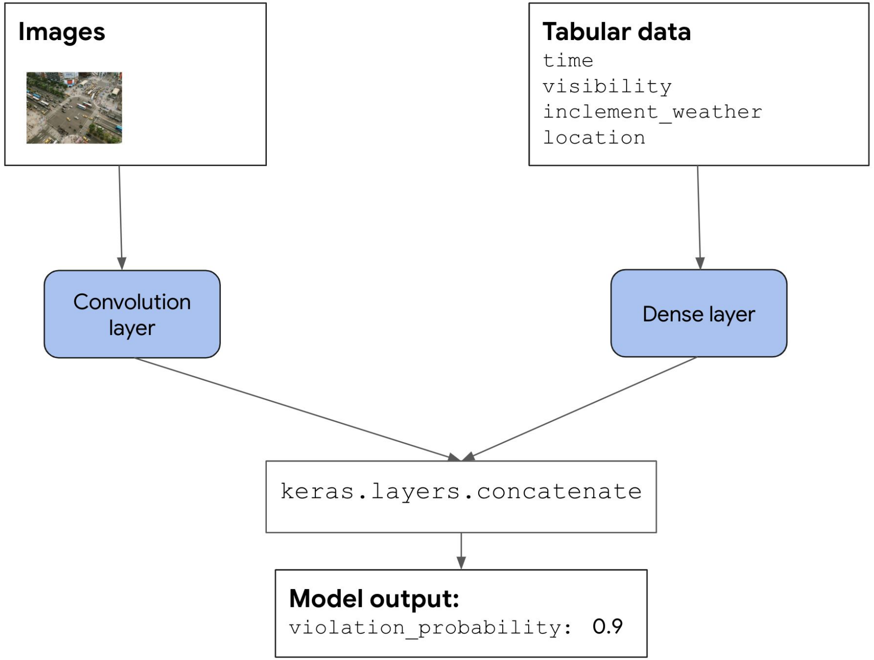
Plus, most models will need a mix of structured and unstructured data



Google Cloud

Concatenate the multimodal representations

Concatenate the inputs as a layer in the model, so that it is part of the model graph (like the Transform pattern)



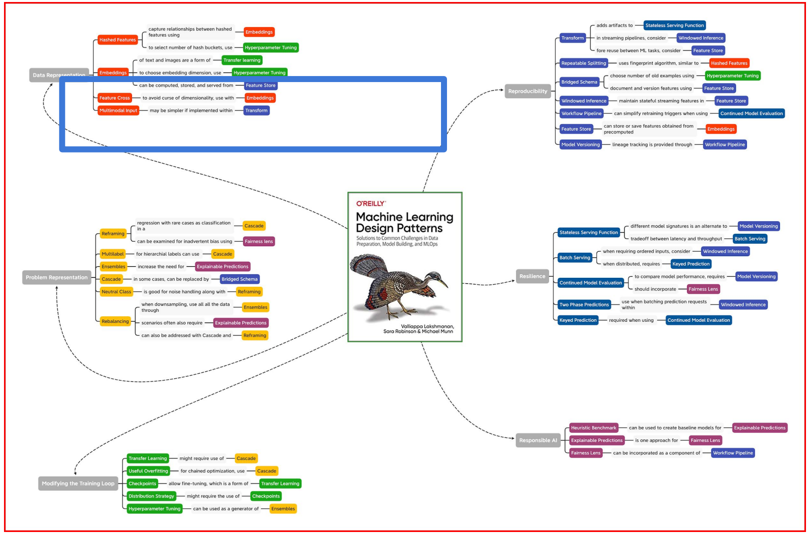
Google Cloud

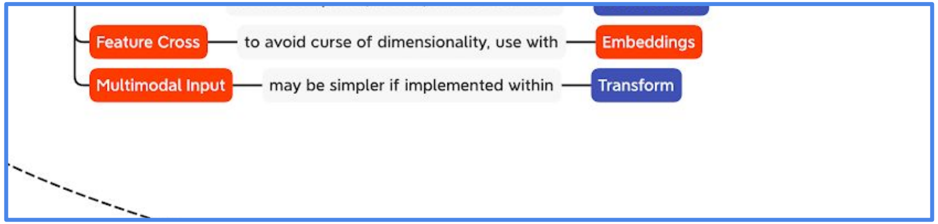
Chapter	Design pattern	Problem solved	Solution
Data Representation	Multimodal Input	Choose between several potential data representations	Concatenate all the available data representations
Problem Representation	Cascade		
Reproducibility	Transform	The inputs to a model must be transformed to create the features the model expects and that process must be consistent between training and serving	Explicitly capture and store the transformations applied to convert the model inputs into features
	Workflow Pipeline	When scaling the ML workflow, you need a way to run trials independently and track performance for each step of the pipeline.	Make each step of the ML workflow a separate, containerized service which can be chained together to make a pipeline that can be run with a single REST API call
	Feature Store		

Simpler if used within

Is part of

Connections between patterns





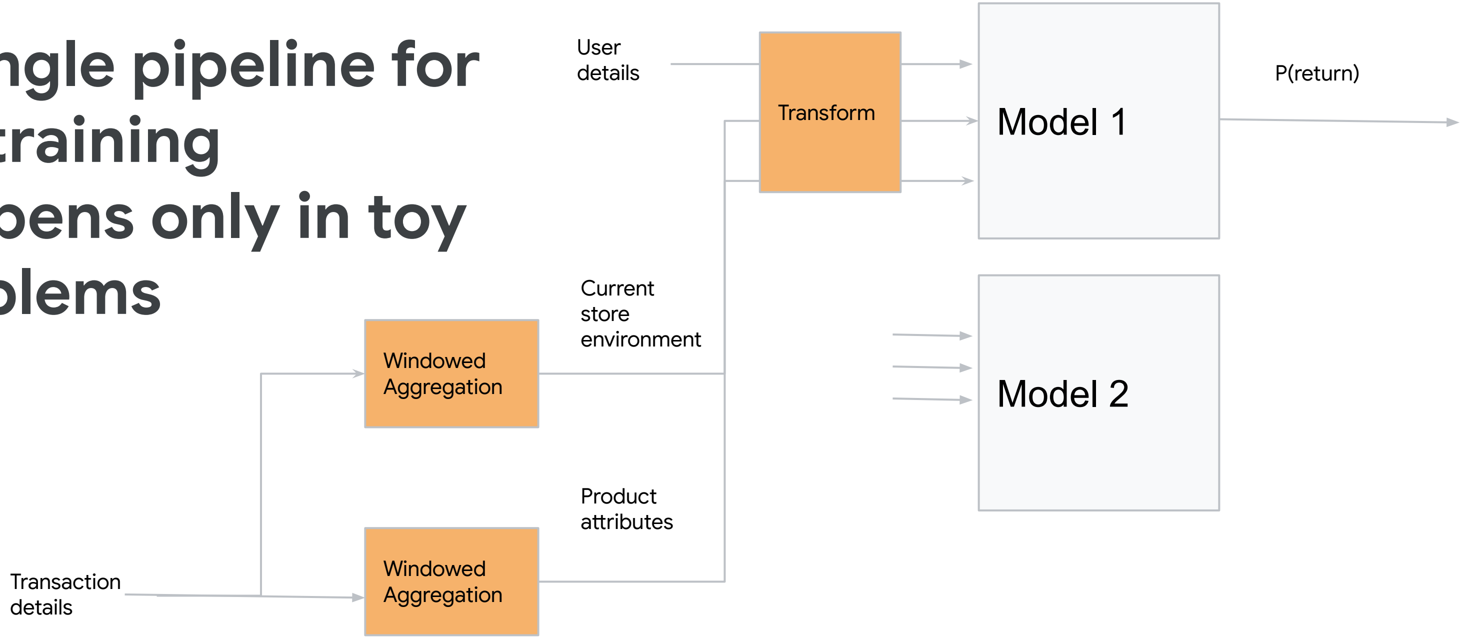
https://github.com/GoogleCloudPlatform/ml-design-patterns/blob/master/08_connected_patterns/machine-learning-design-patterns.png

Feature Store

04

Google Cloud

A single pipeline for ML training happens only in toy problems



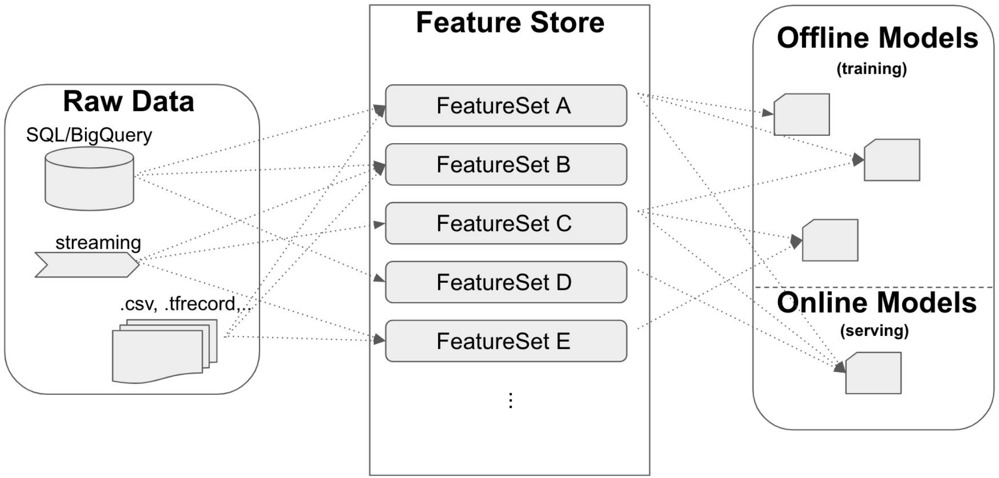
Features get reused
Aggregations get reused
Transforms get reused
Some Transforms are real-time

Feature Store: Centralized location to store feature sets (data storage)

Lambda architecture:

- Store/serve features for low-latency inference
- Store/serve features for large batch access in training
- Metadata layer for versioning of different feature sets

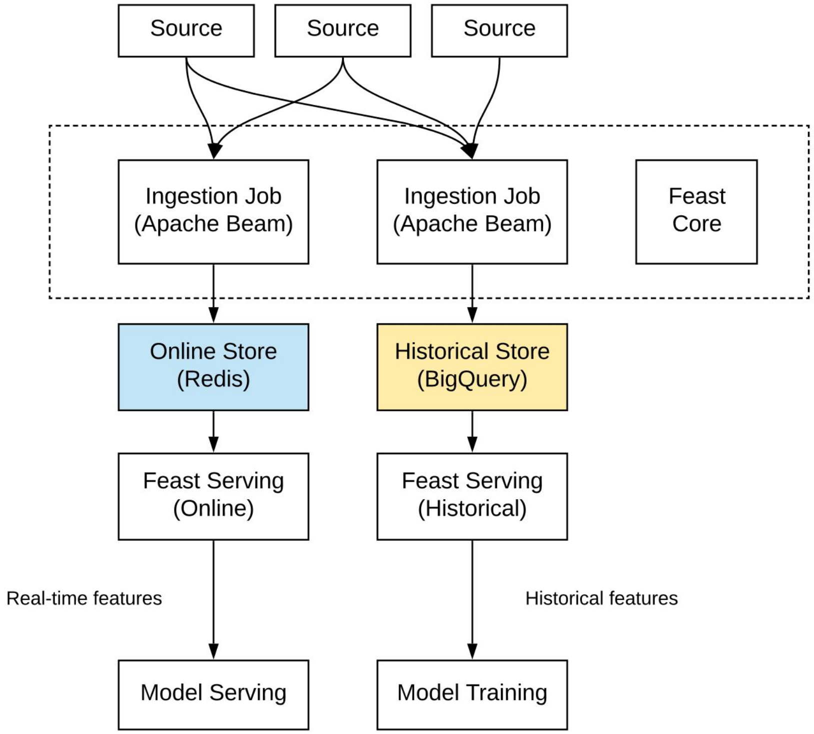
API to manage loading and retrieving feature data.



Google Cloud

Feast

Developed by Google and GoJek



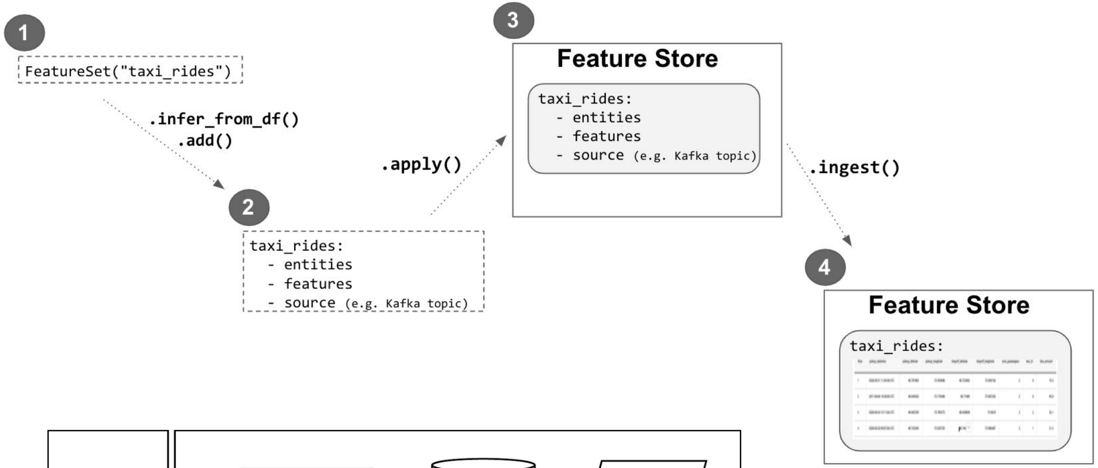
Google Cloud

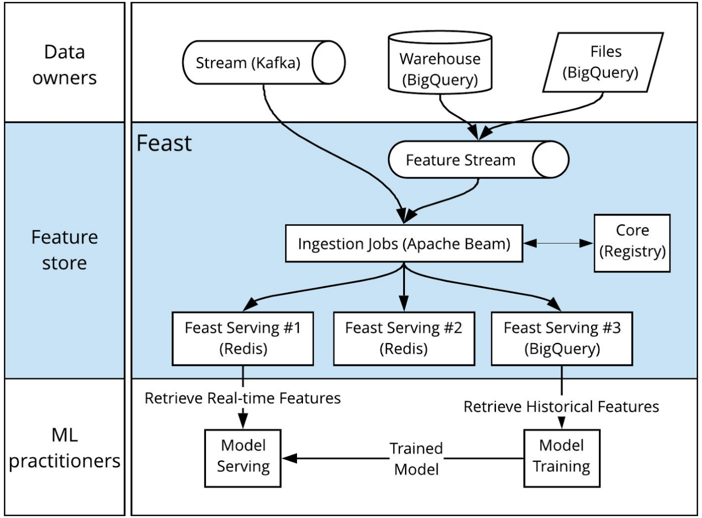
Ingesting into Feast

Four steps:

1. create a FeatureSet
2. add entities and features
3. register the FeatureSet (returns JSON schema)
4. ingest feature data into the FeatureSet.

Feast, like TFX, uses Beam for feature creation



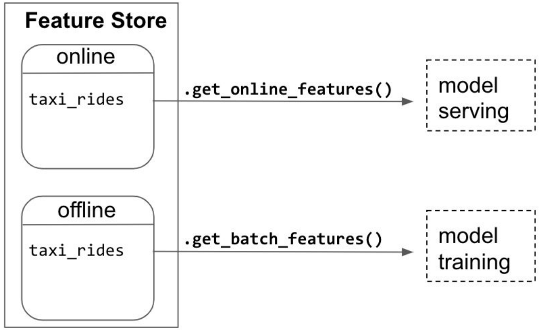


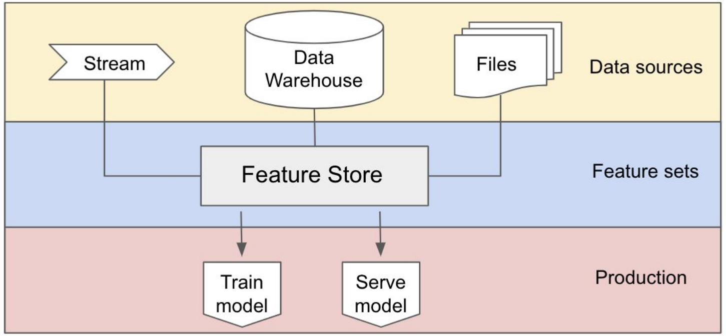
Retrieving from Feast

Feature data can be retrieved either offline, using historical features for model training, or online, for serving.

Two separate REST endpoints

Feast uses Redis and BigQuery for storage for online and batch respectively





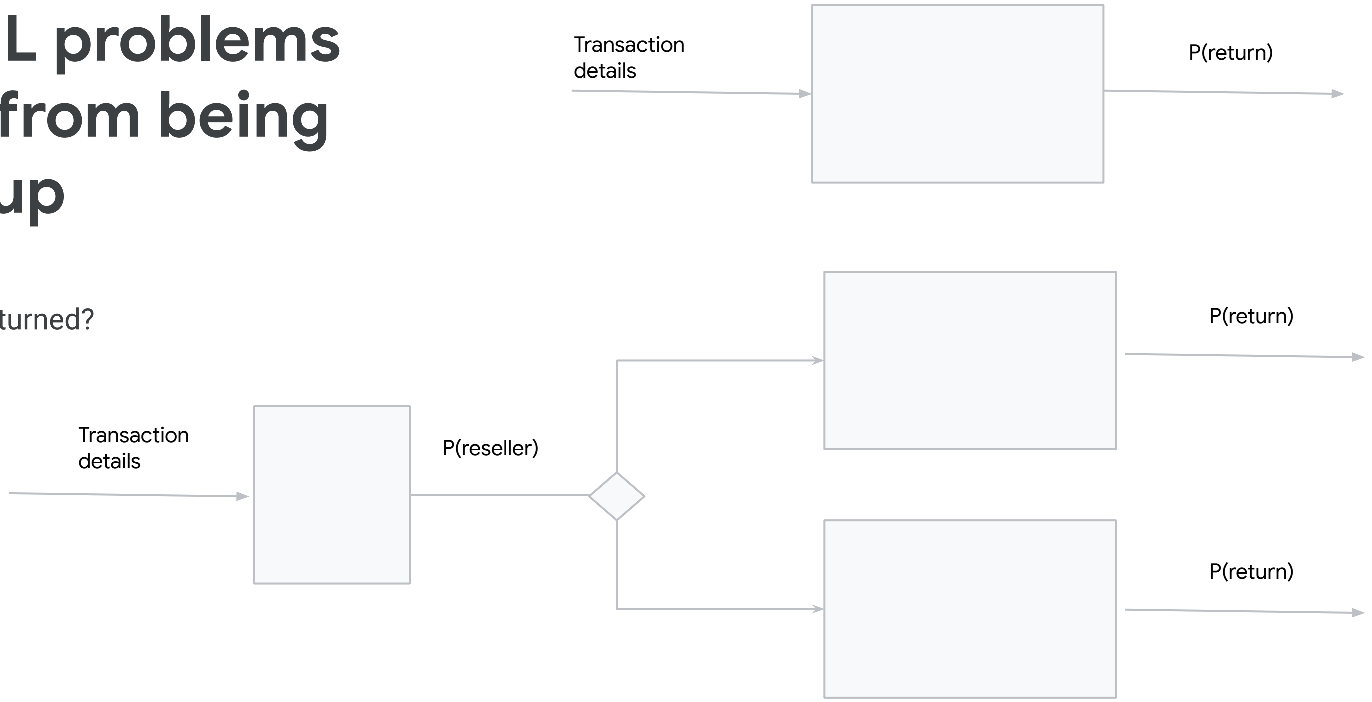
Cascade

05

Google Cloud

Some ML problems benefit from being broken up

Will this item be returned?

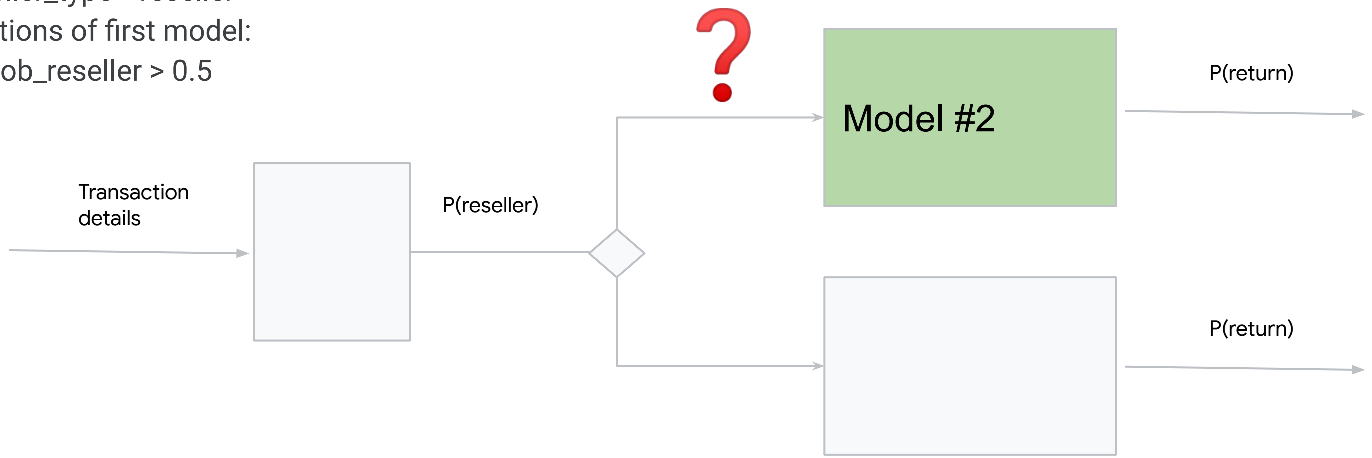


Google Cloud

What is the training data for model #2?

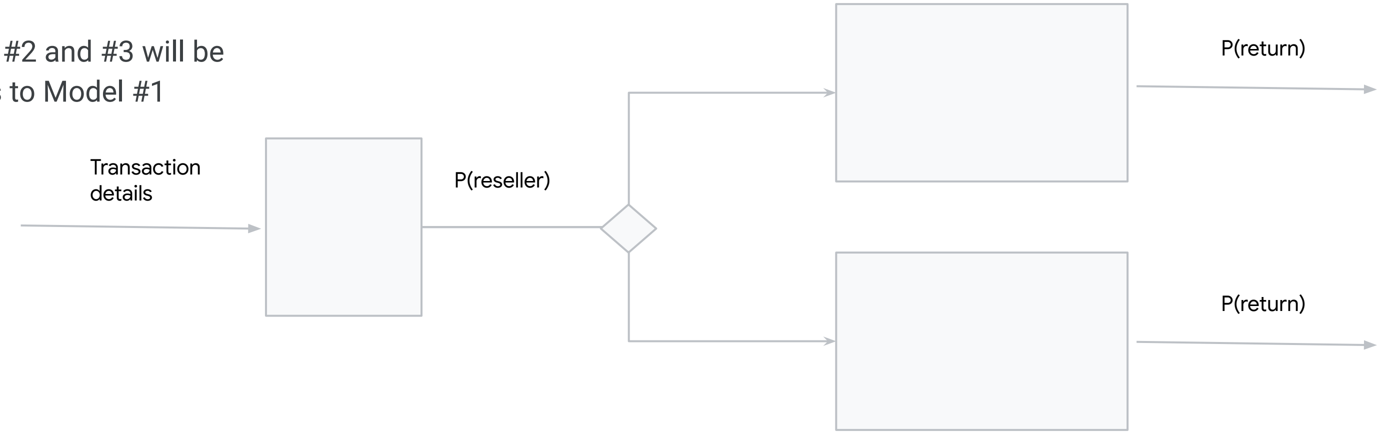
(a) Same as original training dataset:
WHERE seller_type=“reseller”

(b) Use predictions of first model:
WHERE prob_reseller > 0.5



Training this model requires training a Cascade

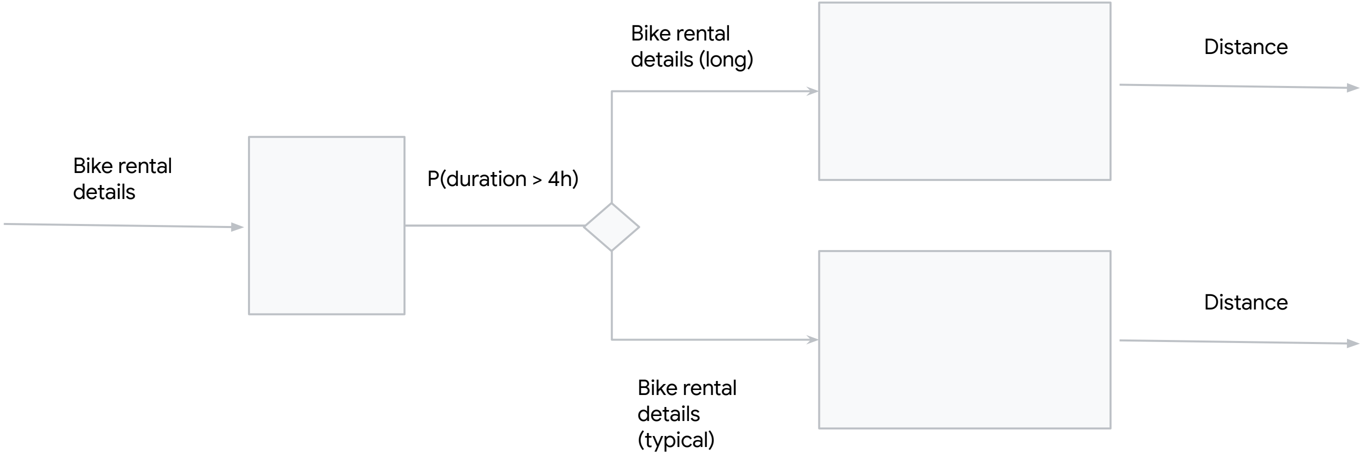
Otherwise, Model #2 and #3 will be fragile to changes to Model #1



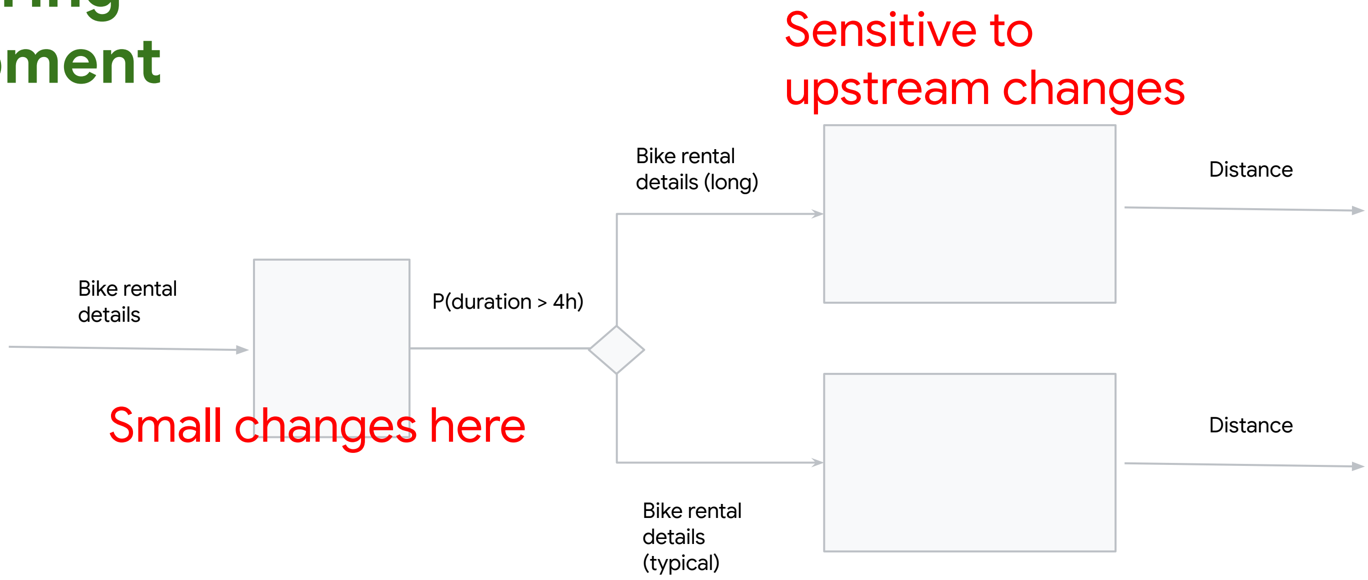
Google Cloud

Use Cascade to handle rare scenarios with less data & simpler models

https://www.flickr.com/photos/mobikefed/3350883056



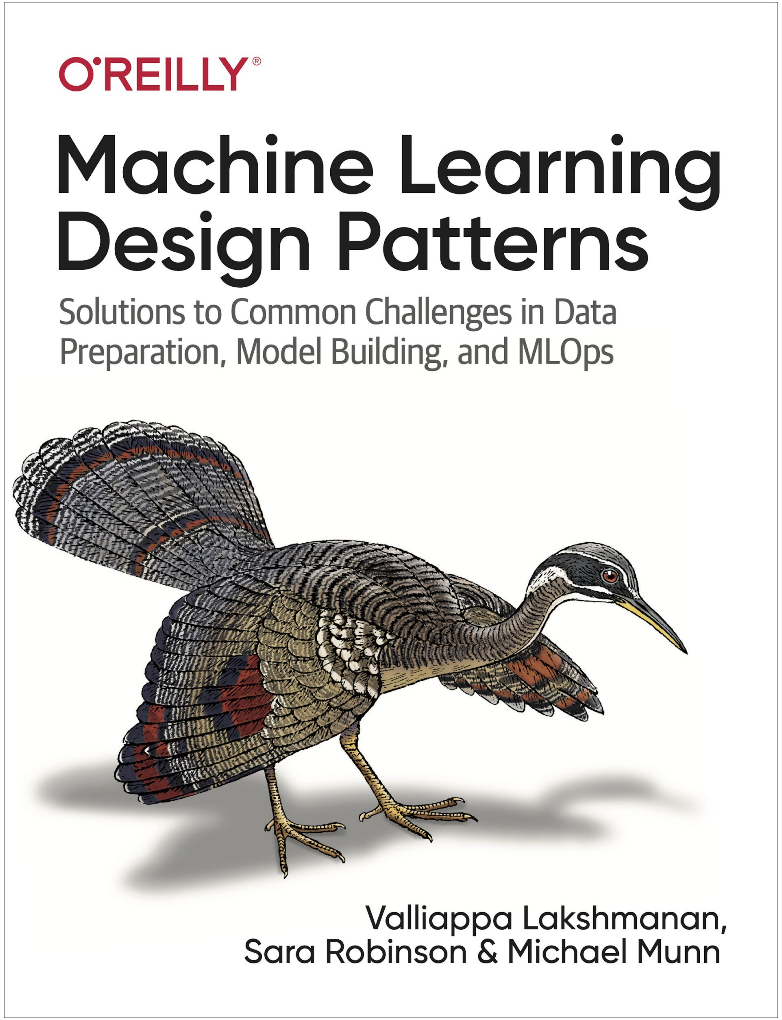
Use Workflow Pipeline to automate Cascade even during development



Google Cloud

Summary

06



Workflow Pipeline

Transform

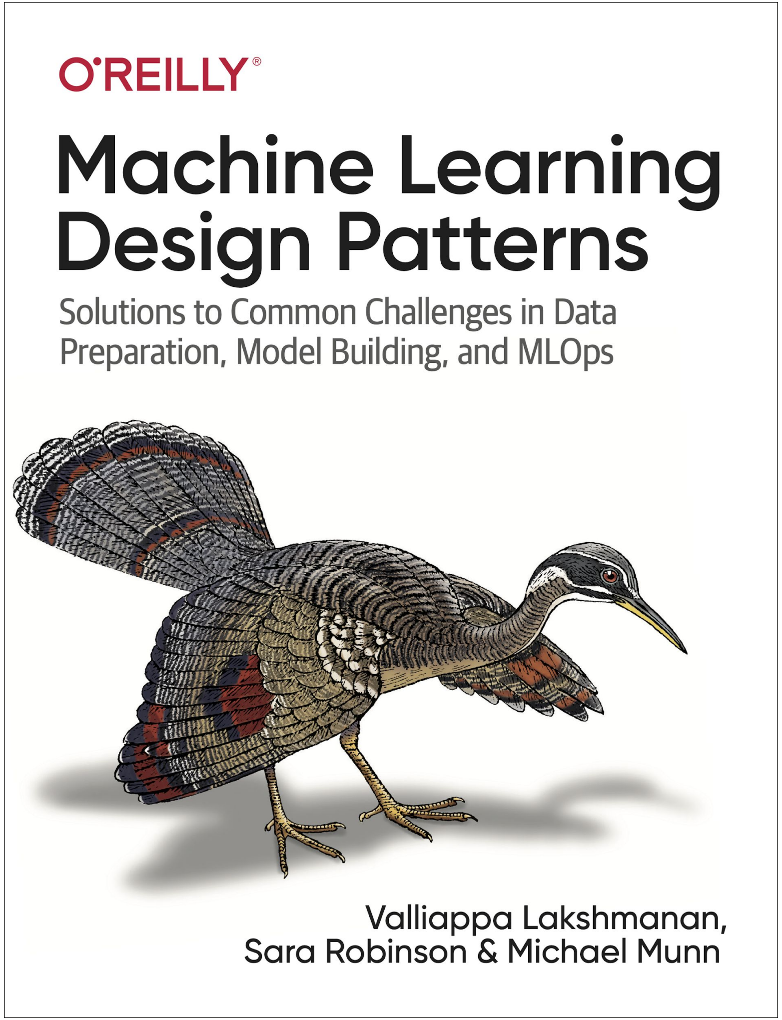
Multimodal Input

Feature Store

Cascade

https://github.com/GoogleCloudPlatform/ml-design-patterns

Chapter	Design pattern	Problem solved	Solution
Data Representation	Multimodal Input	Choose between several potential data representations	Concatenate all the available data representations
Problem Representation	Cascade	Maintainability or drift issues when a machine learning problem is broken into a series of ML problems	Treat your ML system as a unified workflow for the purposes of training, evaluation, and prediction.
Reproducibility	Transform	The inputs to a model must be transformed to create the features the model expects and that process must be consistent between training and serving	Explicitly capture and store the transformations applied to convert the model inputs into features
	Workflow Pipeline	When scaling the ML workflow, you need a way to run trials independently and track performance for each step of the pipeline.	Make each step of the ML workflow a separate, containerized service which can be chained together to make a pipeline that can be run with a single REST API call
	Feature Store	The ad-hoc approach to feature engineering slows model development and leads to duplicated effort between teams as well as work stream inefficiency.	Create a Feature Store, a centralized location to store and document feature datasets that will be used in building machine learning models and can be shared across projects and teams



Read the book (Nov 2020)
https://bit.ly/ml-design-patterns

Follow me on Twitter:
@lak_gcp

Read my blog:
https://medium.com/@lakshmanok

Check out implementations:
https://github.com/GoogleCloudPlatform/ml-design-patterns