CONF42 Python, March 2023

The mythical machine learning pipeline - use feature/training/inference pipelines

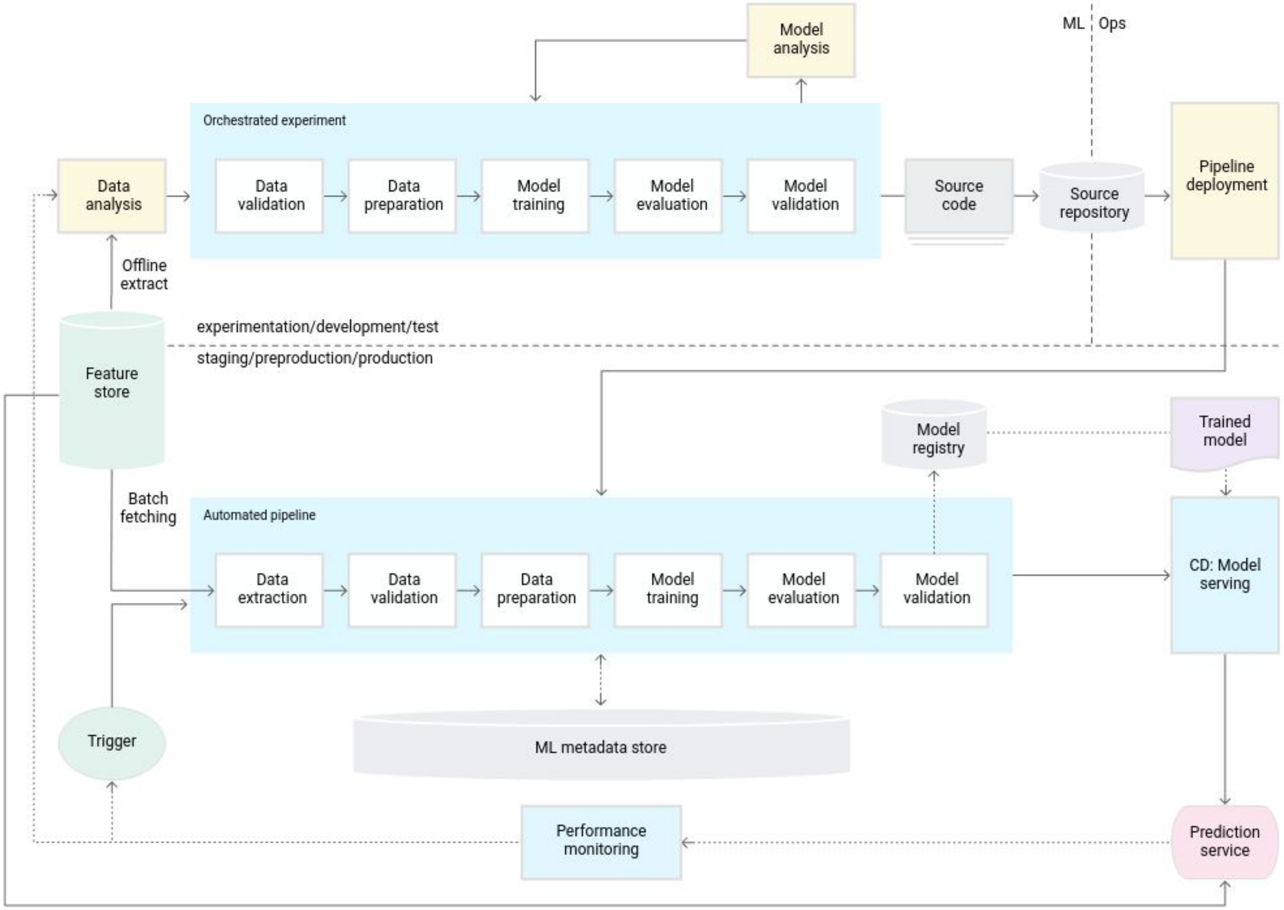
Jim Dowling (@jim_dowling) - CEO at Hopsworks

HOPSWORKS

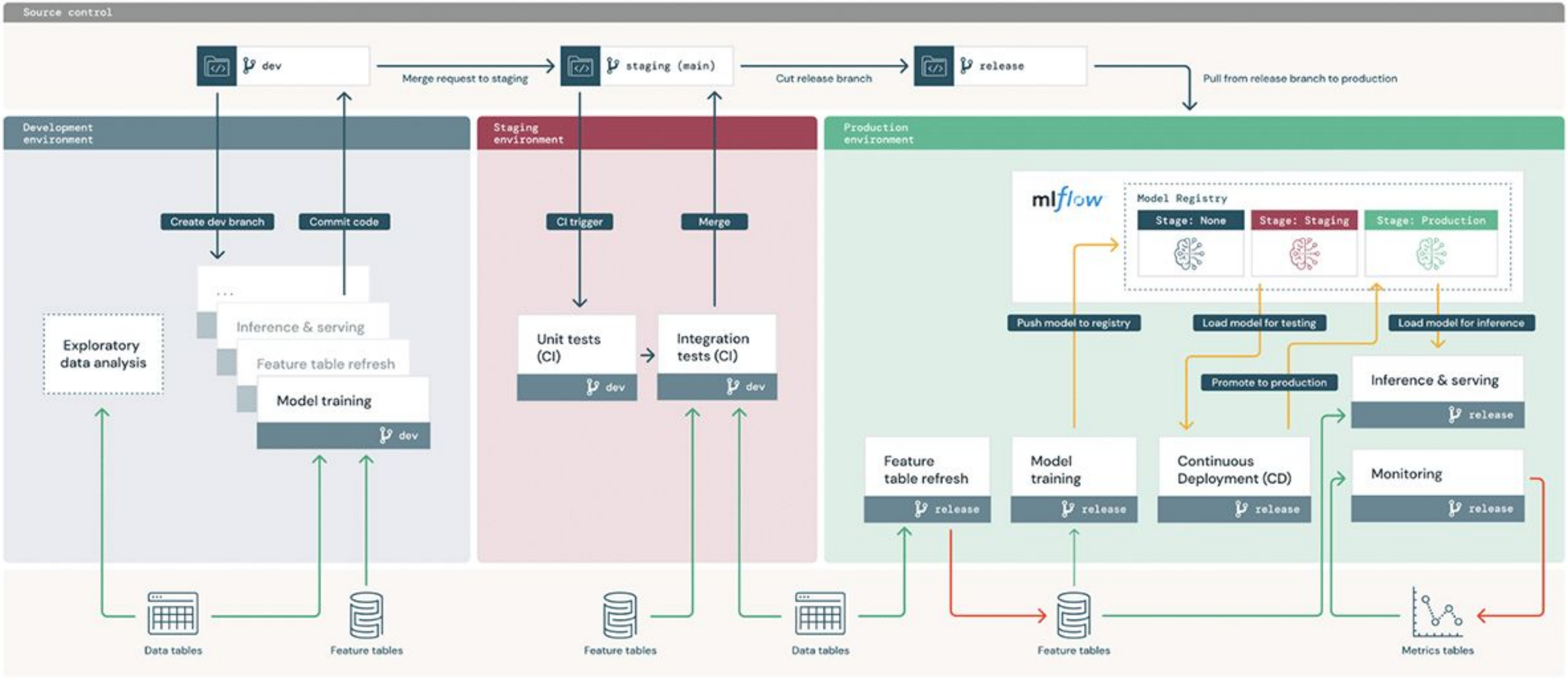
MLOps Today

00 MLOps as it exists today is too hard

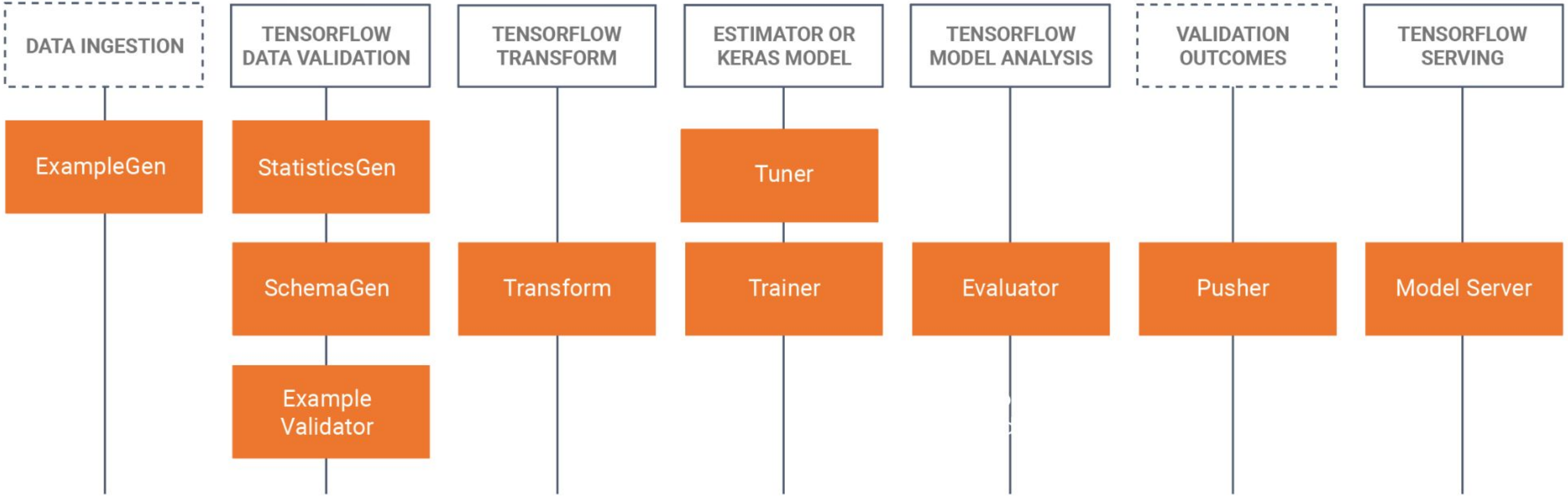
// MLOps according to Google



// MLOps according to Databricks



// The Mythical End-to-End Machine Learning Pipeline



[Image from https://www.tensorflow.org/tfx/guide]

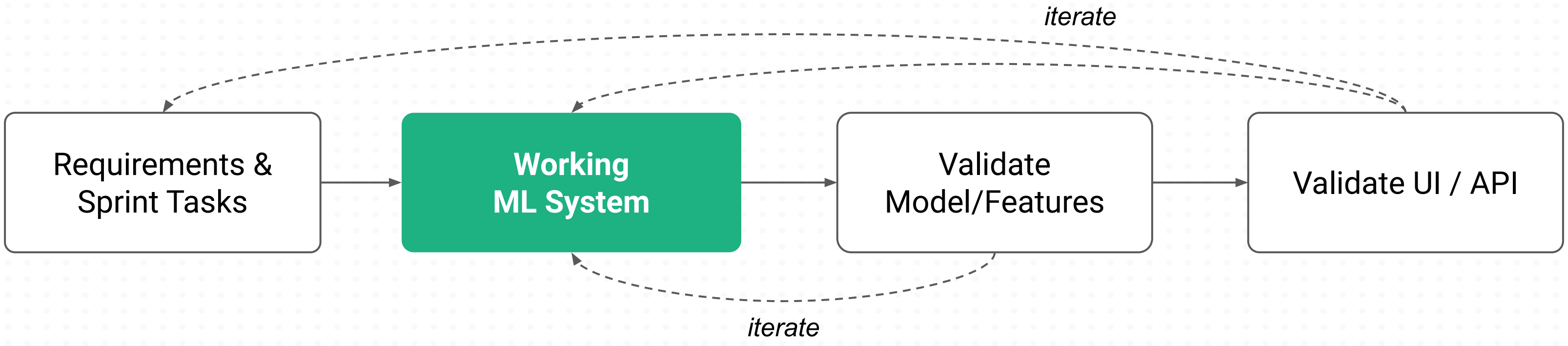
// Problems with the Monolithic ML Pipelines

“Kubeflow Pipelines is a platform for building and deploying portable, scalable machine learning (ML) workflows ….End-to-end orchestration….**enabling you to re-use components** and **pipelines** to quickly create end-to-end solutions without having to rebuild each time” [KubeFlow Website (Feb ‘23)]

- Coupling feature engineering, model training, and inference adds development and operational complexity
- No reuse of precomputed features across multiple models (feature store)
- Too hard to get to a working MVP (minimal viable ML product)

// For developers, MLOps is about Testing, Versioning, and incremental changes

- Get to **a working ML system** with a baseline, then iteratively improve it.
- Automated **testing and versioning of** features and models
- Improve both **iteration speed** and **quality**



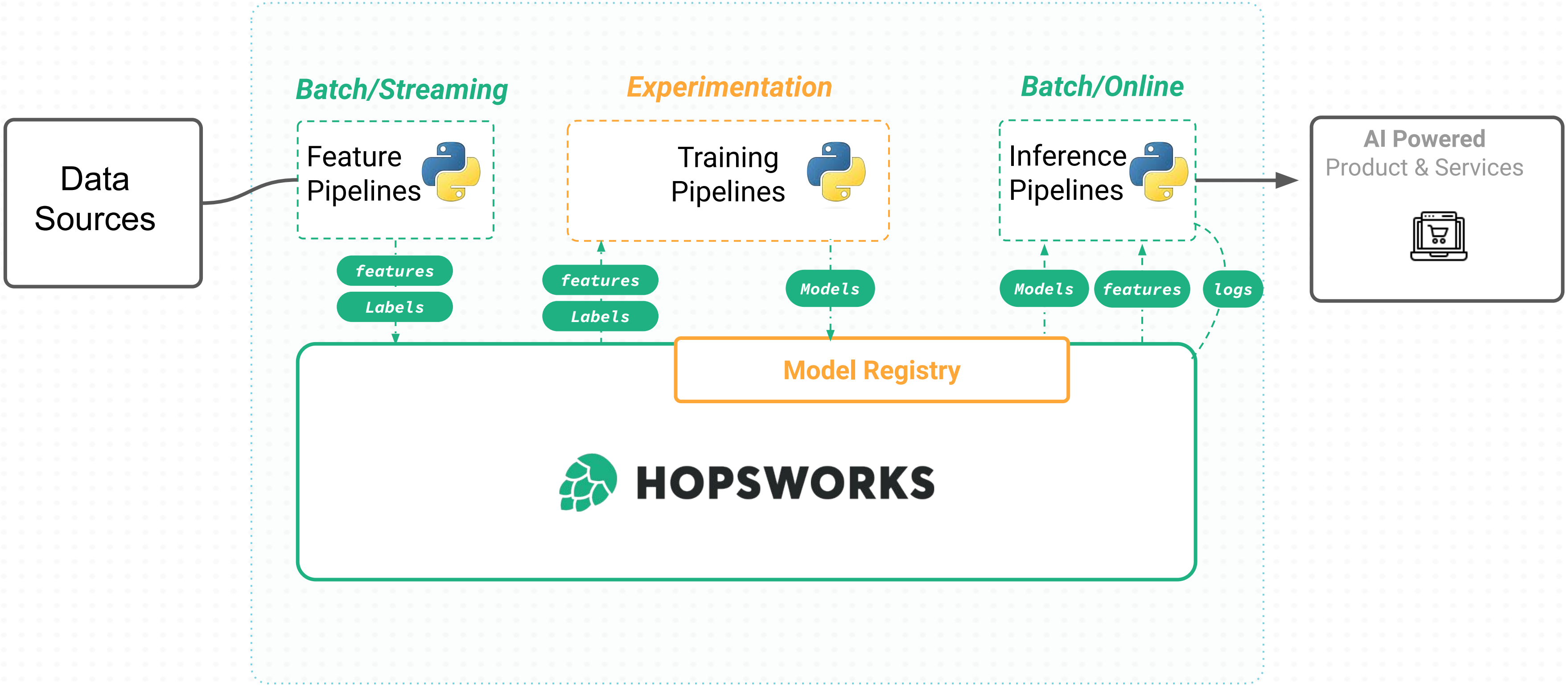
HOPSWORKS

01

MLOps in Python

without the infrastructure

// Feature Pipelines, Training Pipelines, and Inference Pipelines



02 Feature Pipelines

// Feature Pipelines - transform raw data into DataFrames

Feature engineering

Initialize Feature Group metadata

Validate and write feature data

```python
window_len = "4h"
cc_group = trans_df[["cc_num", "amount", "datetime"]] \
    .groupby("cc_num").rolling(window_len, on="datetime")

# Compute aggregations
df_4h_count = pd.DataFrame(cc_group.mean())
df_4h_count.columns = ["trans_freq", "datetime"]
df_4h_count = df_4h_count.reset_index(level=["cc_num"])
df_4h_count = df_4h_count.drop(columns=["cc_num", "datetime"])
df_4h_count = df_4h_count.sort_index()
window_aggs_df = window_aggs_df.merge(df_4h_count,left_index=True, right_index=True)

# Initialize metadata of feature group
window_aggs_fg = fs.get_or_create_feature_group(
    name="transactions_4h_aggs_fraud_batch_fg",
    version=1,
    description="Aggregate transaction data over 4hwindows.",
    primary_key=["cc_num"],
    event_time="datetime"
)

# Write Dataframe to the feature group
window_aggs_fg.insert(window_aggs_df)
```

// Feature Pipelines: Python or Spark or SQL or Flink?

Spark

Good: scale, testing, complex features.

Bad: resource estimation, debugging, operations.

SQL

Good: low operational overhead, easy to implement aggregations

Bad: Complex features (e.g., embeddings) become very complex with UDFs

Python

Good: low operational overhead, rich library support, complex features possible. Pandas or Polars.

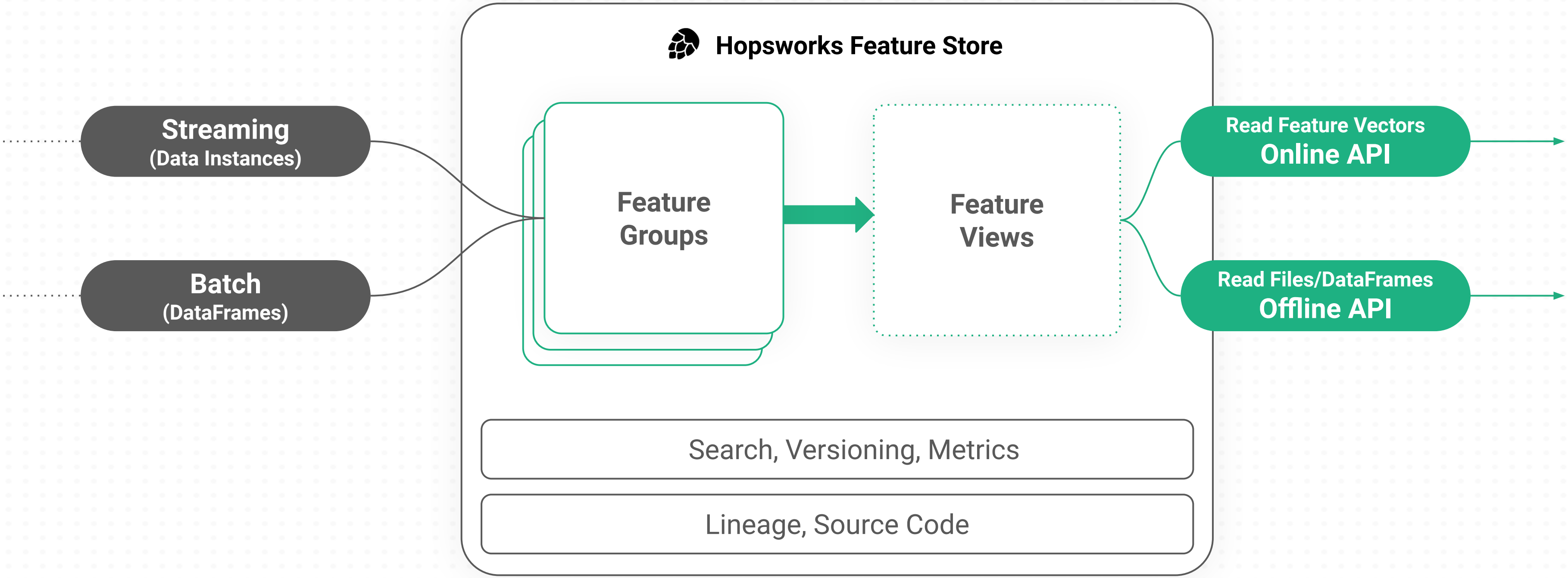
Bad: does not scale (~10 GBs with Pandas, Polars ~100 GBs),

Flink Challenges

Good: low-latency, real-time feature computation that scales to huge clusters (10k+ nodes)

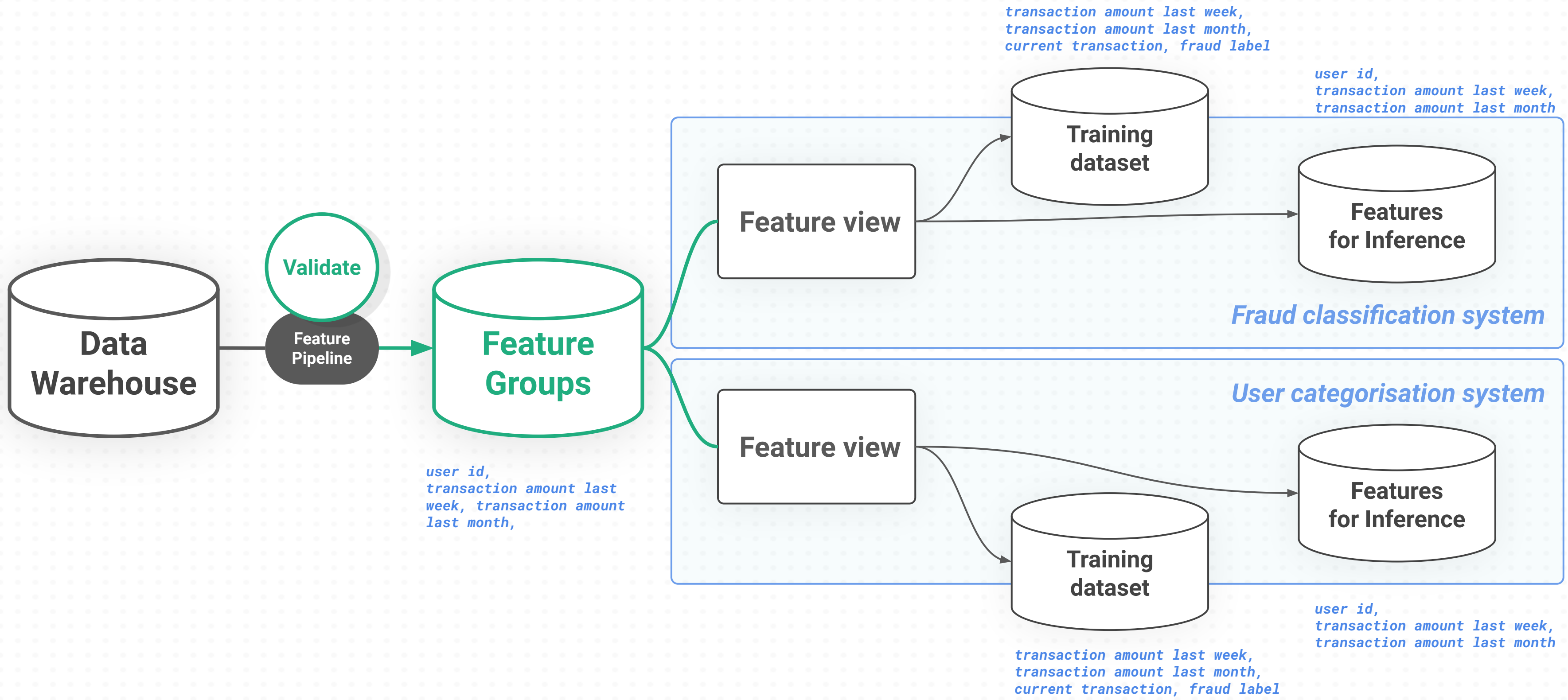
Bad: complex, Java-centric, high operational overhead

// Feature Store: Write to Feature Groups, Read from Feature Views



// One Feature Pipeline - Many Models

A **single feature pipeline** to ingest features into feature group and the feature view can reuse the features for training and inference on **many models**.

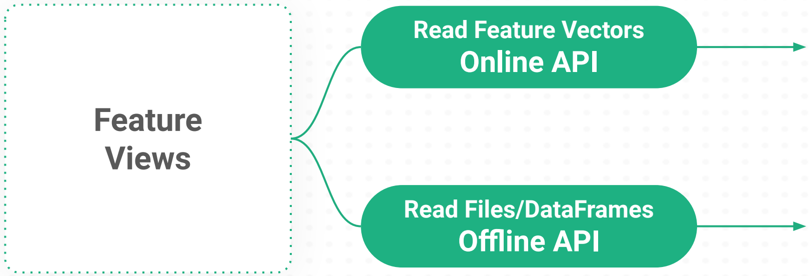


03 Training Pipelines

// Feature Store: Read from Feature Views

What does a Feature View do?

1. **API** for ML model development and operations
2. **Define** features, labels, and model-specific transformations
3. **Generate** training dataset, batch inference data, and feature vectors



// Feature View: Point-in-Time Correct JOINs

Label Feature Group

cust_id	**event_time**	category
1	2021-01-01	short_term
2	2021-01-02	short_term
3	2021-01-16	long_term

Transactions Feature Group (weekly)

cust_id	**event_time**	amount_last_week	amount_last_month
1	2021-01-01	14.00	87.00
1	2021-01-07	44.00	80.00
1	2021-01-14	78.00	82.00

Point-in-time Correct JOIN
(no data leakage)

trans.event_time => time	category	amount_last_week	amount_last_month
2022-01-01	short_term	14.00	87.00
2022-01-02	short_term	44.00	80.00
2022-01-16	long_term	78.00	82.00

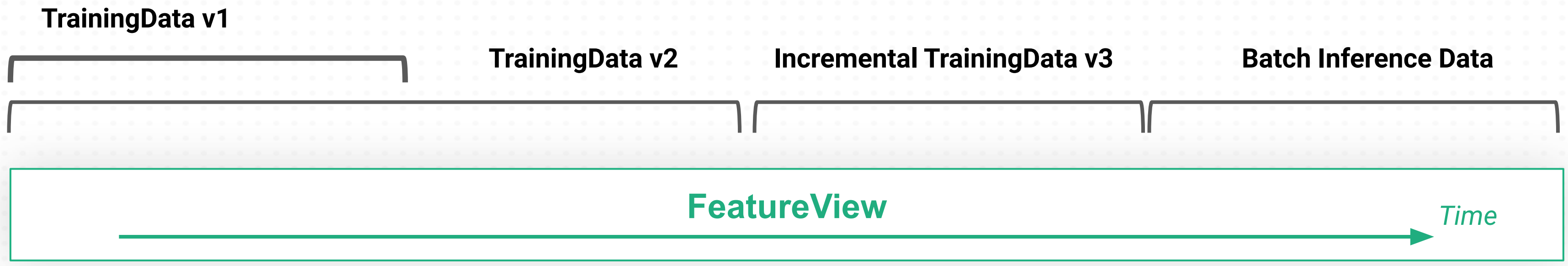
Training Data

// Feature View: Point-in-Time Correct JOINs

```sql
WITH right_fg0 AS (
SELECT *
FROM (
    SELECT `fg1`.`category`
         , `fg0`.`amount_last_seek`
         , `fg0`.`amount_last_month`
         , RANK() OVER (PARTITION BY `fg0`.`cust_id`, `fg1`.`event_time` ORDER BY `fg0`.`event_time` DESC) pit_rank_hopsworks
    FROM `label_fg` `fg1`
    INNER JOIN `transactions_fg` `fg0`
    ON `fg1`.`cust_id` = `fg0`.`cust_id` AND `fg1`.`event_time` >= `fg0`.`event_time`
)
WHERE `pit_rank_hopsworks` = 1)
(SELECT `right_fg0`.`category`, `right_fg0`.`mean_cloud_perc_se3`, `right_fg0`.`mean_cloud_perc_se3`
FROM right_fg0)
```

```python
query = label_fg.select(["category"]).join(
    transactions_fg.select(["amount_last_week", “amount_last_month”])
)
```

// Feature View: Training Data and Batch Inference Data for Models




```python
# Create training data in a specific file format
td_version, td_job = feature_view.create_train_validation_test_split(
    description = 'transactions fraud batch training dataset',
    data_format = 'csv',
    validation_size = 0.2,
    test_size = 0.1
)

# Retrieve previoulsy materialised training data
X_train, y_train, X_val, y_val, X_test, y_test = feature_view.get_train_validation_test_split(1)

# Generate batches of data from a certain time range
start_time = datetime.strptime("2022-01-03 00:00:01", date_format)
end_time = datetime.strptime("2022-03-31 23:59:59",date_format)
march_transactions = feature_view.get_batch_data(
    start_time = start_time,  end_time = end_time)
```

// Feature View: Model-Specific Transformations

Training Pipeline

```
standard_scaler = fs.get_transformation_function(name="standard_scaler")
label_encoder = fs.get_transformation_function(name="label_encoder")

transformation_functions = {
    "amount_last_month": standard_scaler,
    "amount_last_week": standard_scaler,
    "category": label_encoder,
}

fv = fs.create_feature_view(name="category_predictin",
        …
        transformation_functions=transformation_functions)
X_train,X_test,y_train, y_test = fv.train_test_split(0.1)
```

Online Inference Pipeline

```
keys= {cust_id: 1}
feature_vector = fv.get_feature_vector(keys)
```

Model-Specific Transformation functions (UDFs) applied transparently

04 Inference Pipelines

// Tying it all together

The predictor interface

Establish connection to Feature Store

Initialize Feature View

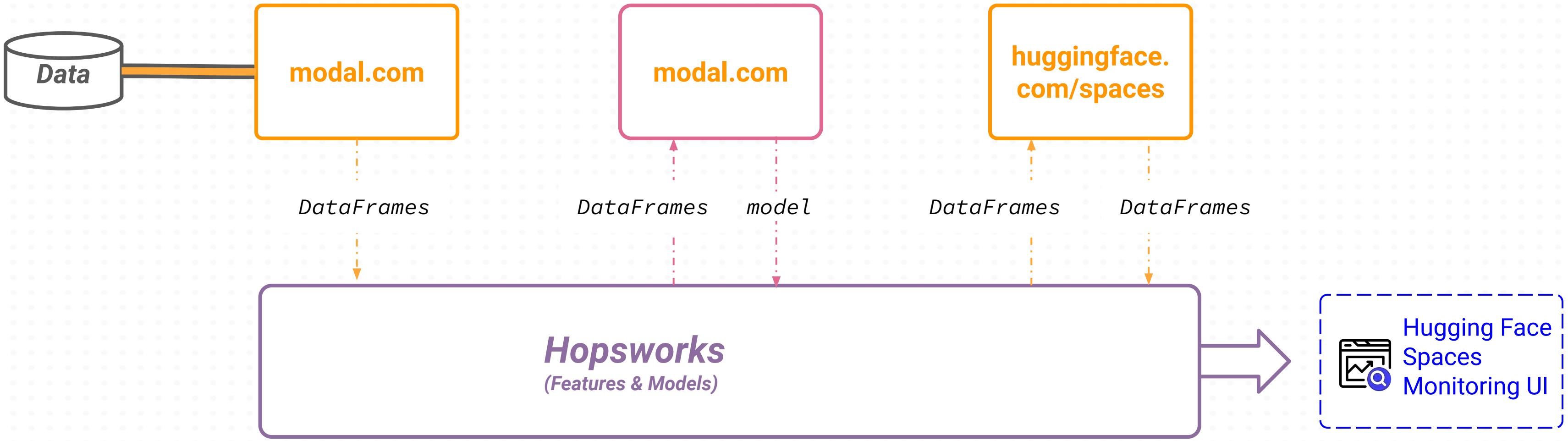
Loading the model from a model registry

Implementing the predict interface using the Feature View

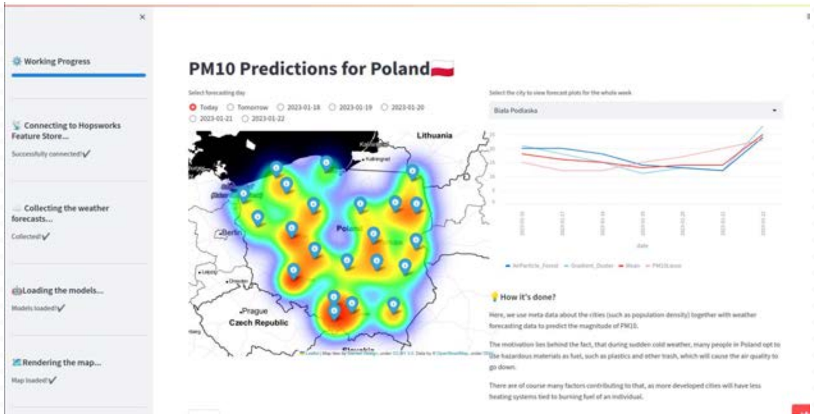
```python
import os
import numpy as np
import hsfs
import joblib

class Predict(object):

    def __init__(self):
        """ Initializes the serving state, reads a trained model"""
        # get feature store handle
        fs_conn = hsfs.connection()
        self.fs = fs_conn.get_feature_store()

        # get feature views
        self.fv = self.fs.get_feature_view("transactions_view", 1)

        # initialise serving
        self.fv.init_serving(1)

        # load the trained model
        self.model = joblib.load(
            os.environ["ARTIFACT_FILES_PATH"] + "/fraud.model.pkl")
        print("Initialization Complete")

    def predict(self, inputs):
        """ Serves a prediction request usign a trained model"""
        # Numpy Arrays are not JSON serializable
        return self.model.predict(
            np.asarray(
                self.fv.get_feature_vector({"cc_num": inputs[0]}))
                    .reshape(1, -1)
            ).tolist()
```

05 Demo

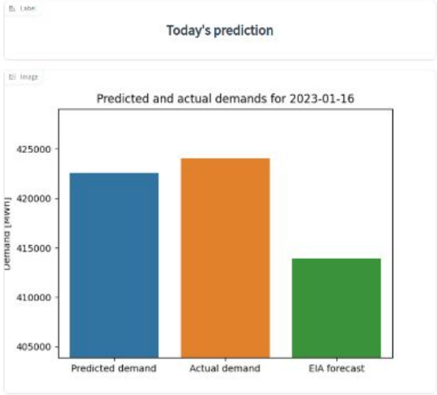
// Serverless Machine Learning with Hopsworks, Modal, and HF Spaces



// Example Serverless Machine Learning Systems



[Air Quality Predictions for Poland]



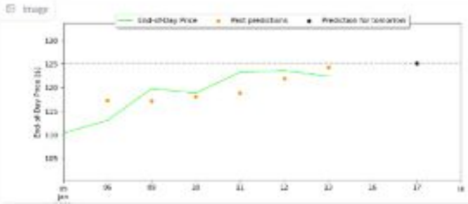
[New York Electricity Price Demand]

[Will your reddit post be liked or not?]

Recent Predictions

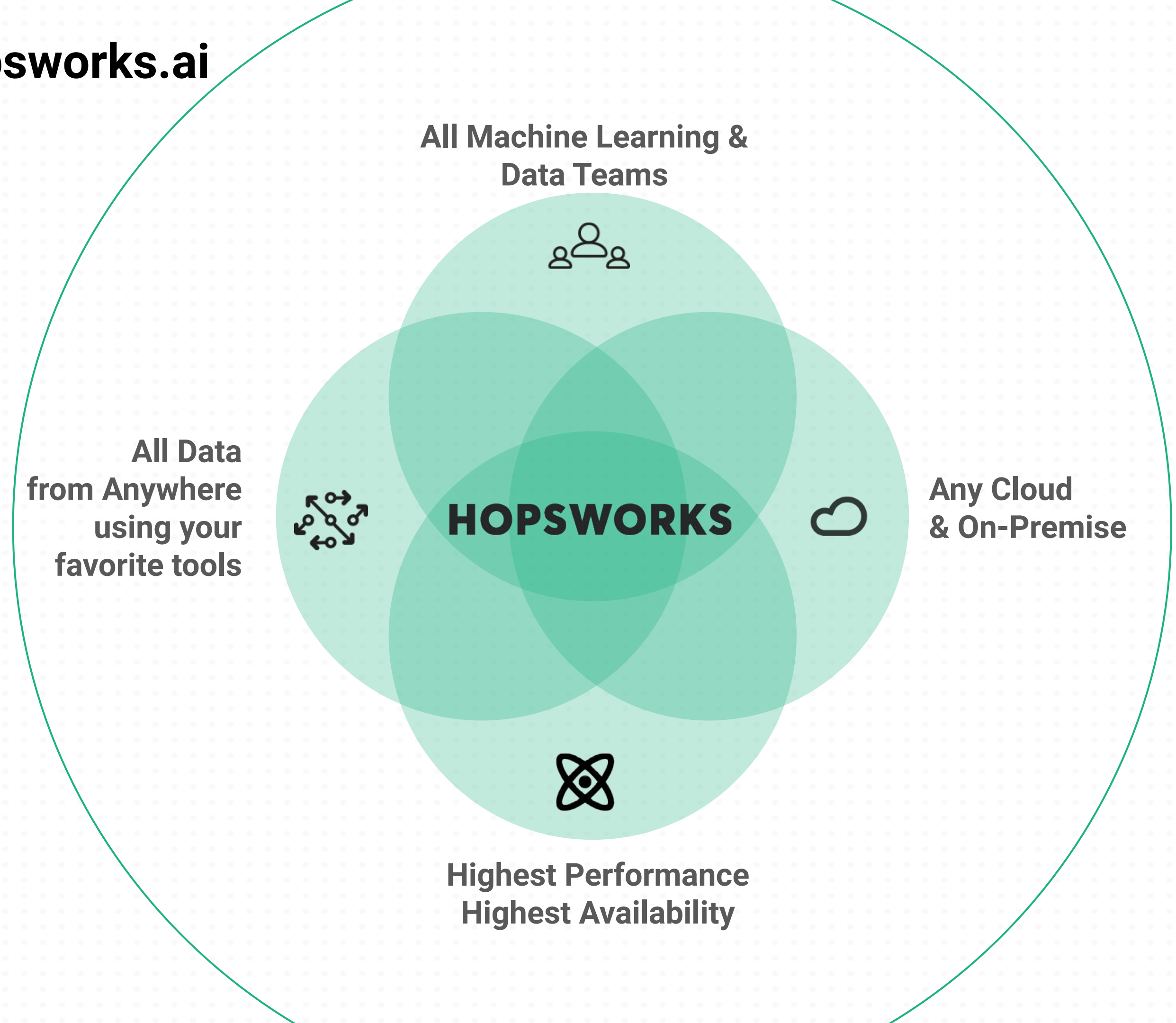
	Date	Pred [$]	Pred [%]	True [$]	True [%]
0	2023-01-06	117.289150	0.062979	113.059998	0.024651
1	2023-01-09	117.124921	0.035954	119.769997	0.059349
2	2023-01-10	118.120352	-0.013773	118.849998	-0.007681
3	2023-01-11	118.876772	0.000225	123.220001	0.036769
4	2023-01-12	121.877310	-0.010897	123.559998	0.002759
5	2023-01-13	124.272293	0.005765	122.400002	-0.009388
6	2023-01-17	125.150664	0.022473	NaN	NaN

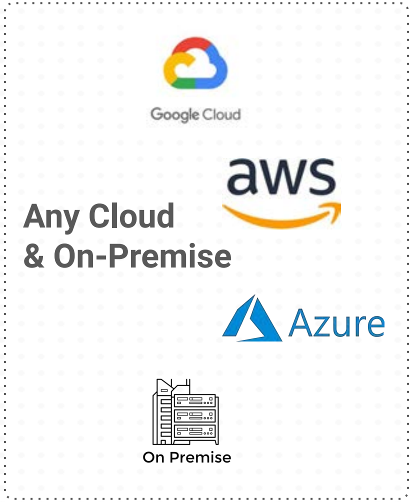
Recent Stock Prices



[Tesla Stock Price Prediction]

// hopsworks.ai





HOPSWORKS

public-hopsworks.slack.com

- Join our slack community
- Explore our latest tutorials
- Ask us any questions

SERVERLESS ML

www.serverless-ml.org