A new way to do Proof-of-Value for your models and AI

Create a MVPS (minimal viable prediction service) in two weeks

Presenter;

Jim Dowling - CEO & Co-Founder of Hopsworks and an Associate Professor at KTH Royal Institute of Technology. Co-inventor of the open-source Hopsworks platform.

Hopsworks is an Operational ML Platform. Python-native and high performance; for any cloud or infrastructure.

Silicon Valley
470 Ramona St
Palo Alto
California,
USA

⭐ **Stockholm**
Hopsworks AB
Medborgarplatsen 25
118 26 Stockholm
Sweden

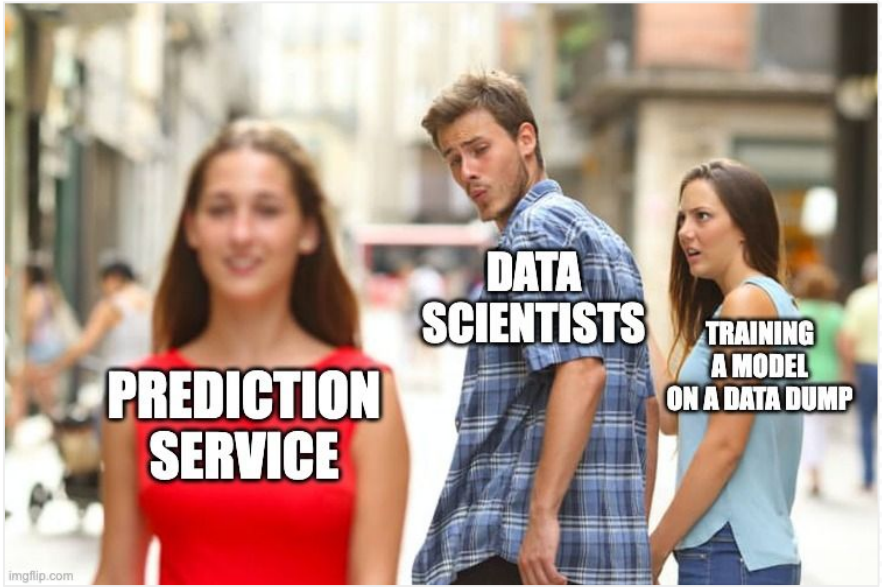
London
IDEALondon,
69 Wilson St,
London,
UK

Allstate. HUMAN AMERICA FIRST CREDIT UNION ARBETSFÖRMEDLINGEN SWEDISH PUBLIC EMPLOYMENT SERVICE Karolinska Institutet PADDYPOWER. GETINGE

What is the scope of a Data Scientist's work?

1. Train a Model on a static dataset and produce an Evaluation Report

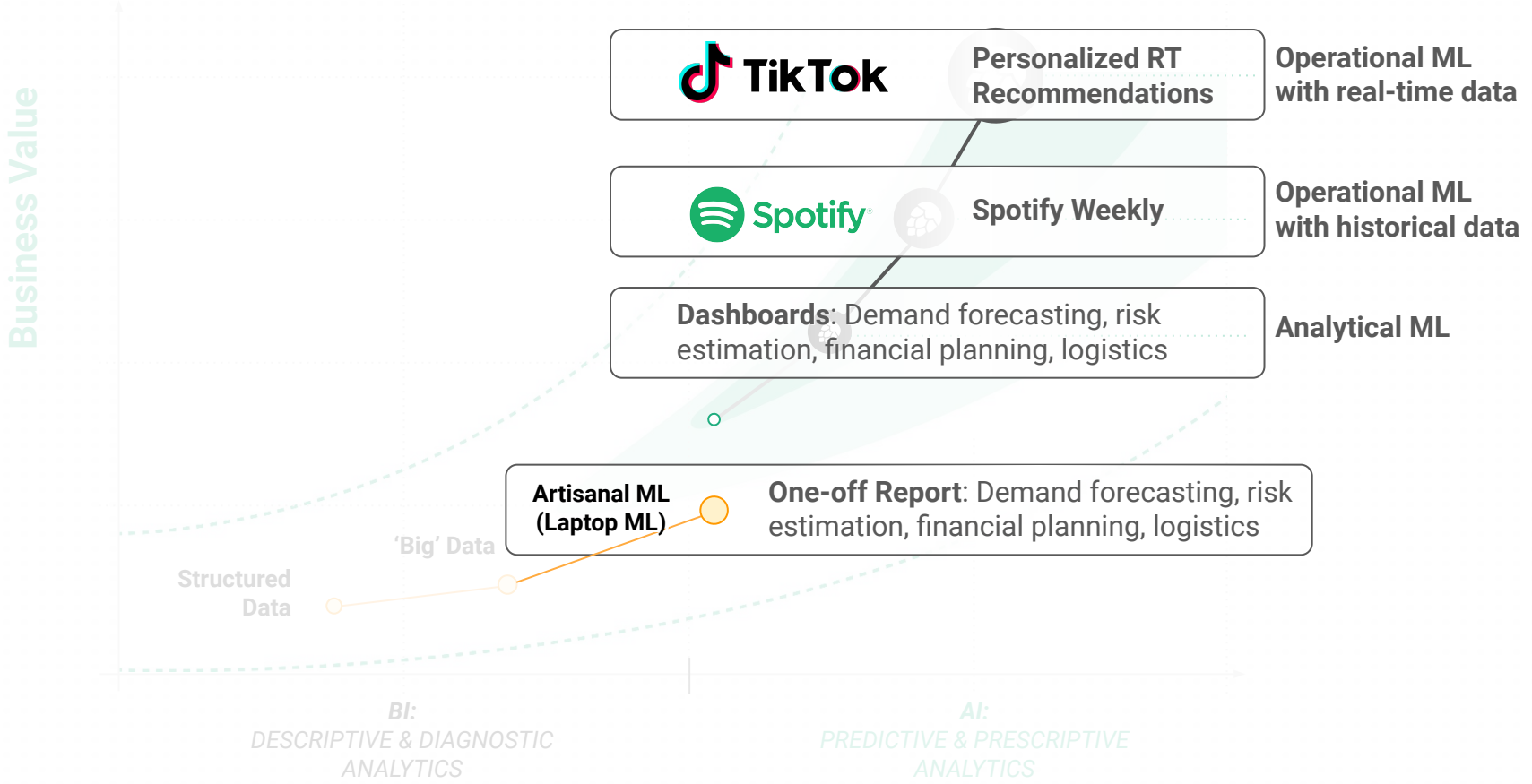
2. Build a [Minimal Viable Prediction Service]
   a. A new language for communicating your work with Stakeholders
   b. Show the model working in its expected deployment environment



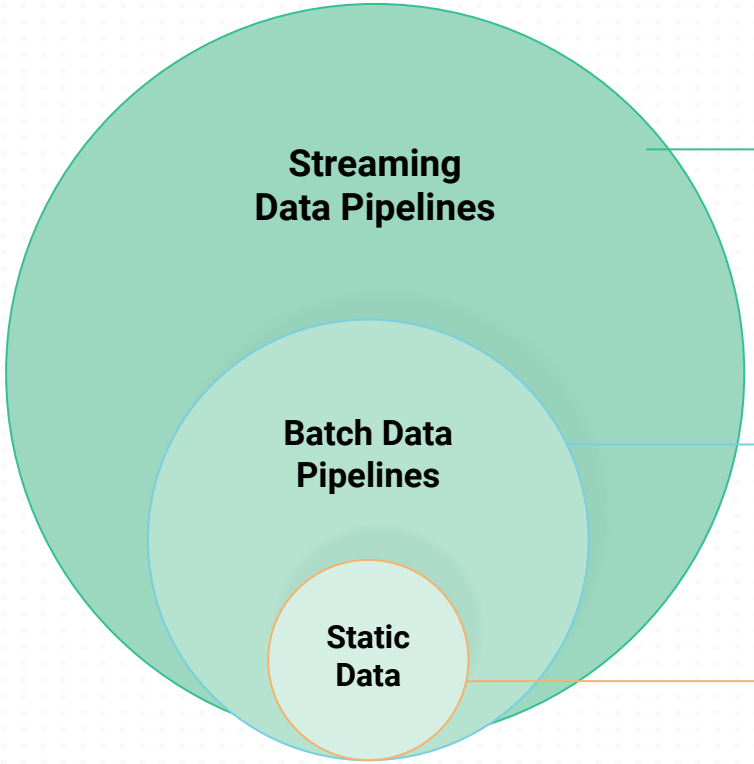
1

CONCEPTS

FTI Pipelines, the Feature Store, and MLOps


Business Value
TikTok
Personalized RT Recommendations
Operational ML with real-time data
Spotify
Spotify Weekly
Operational ML with historical data
Dashboards: Demand forecasting, risk estimation, financial planning, logistics
Analytical ML
Artisanal ML (Laptop ML)
One-off Report: Demand forecasting, risk estimation, financial planning, logistics
'Big' Data
Structured Data
BI:
DESCRIPTIVE & DIAGNOSTIC ANALYTICS
AI:
PREDICTIVE & PRESCRIPTIVE ANALYTICS

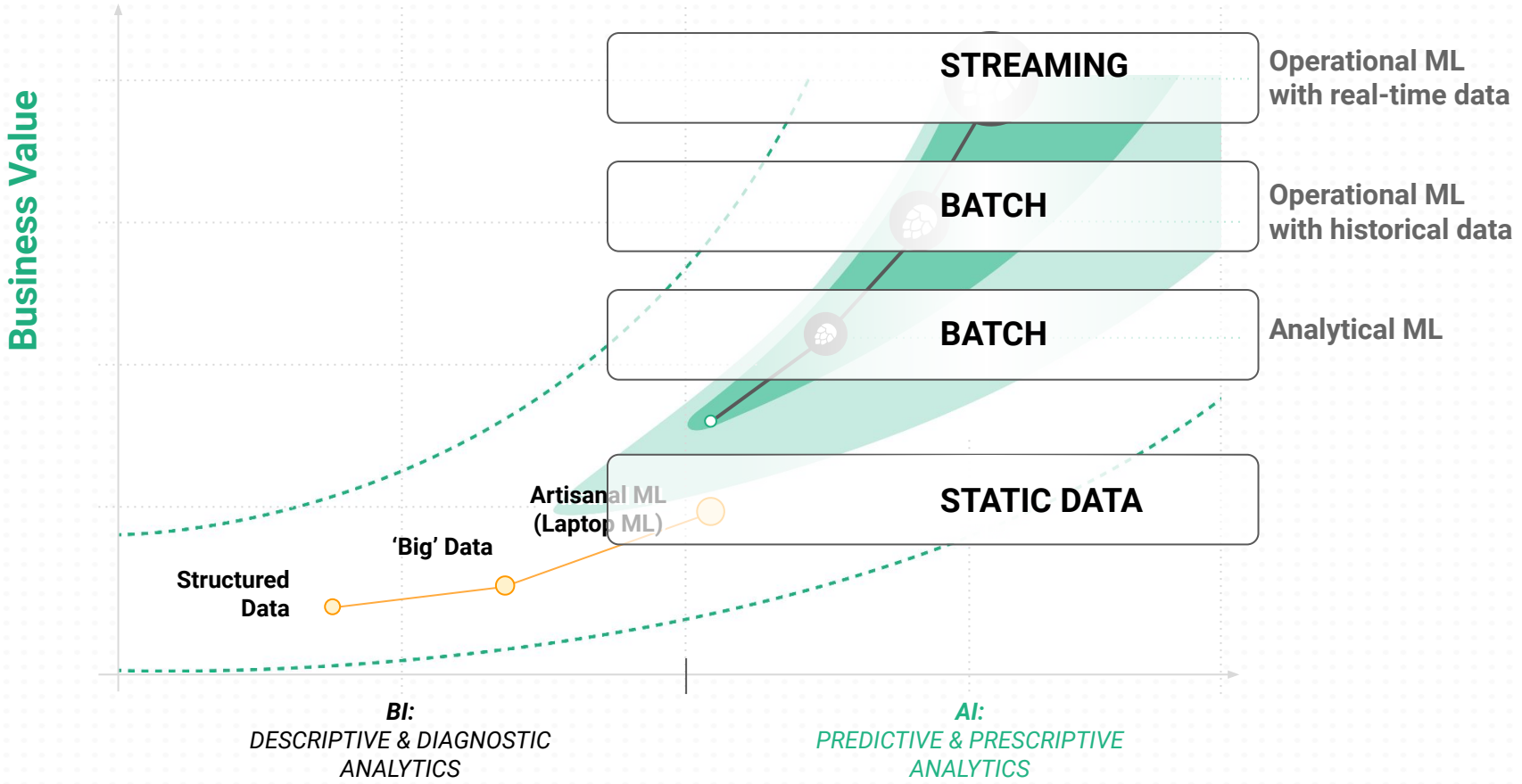
Business Value

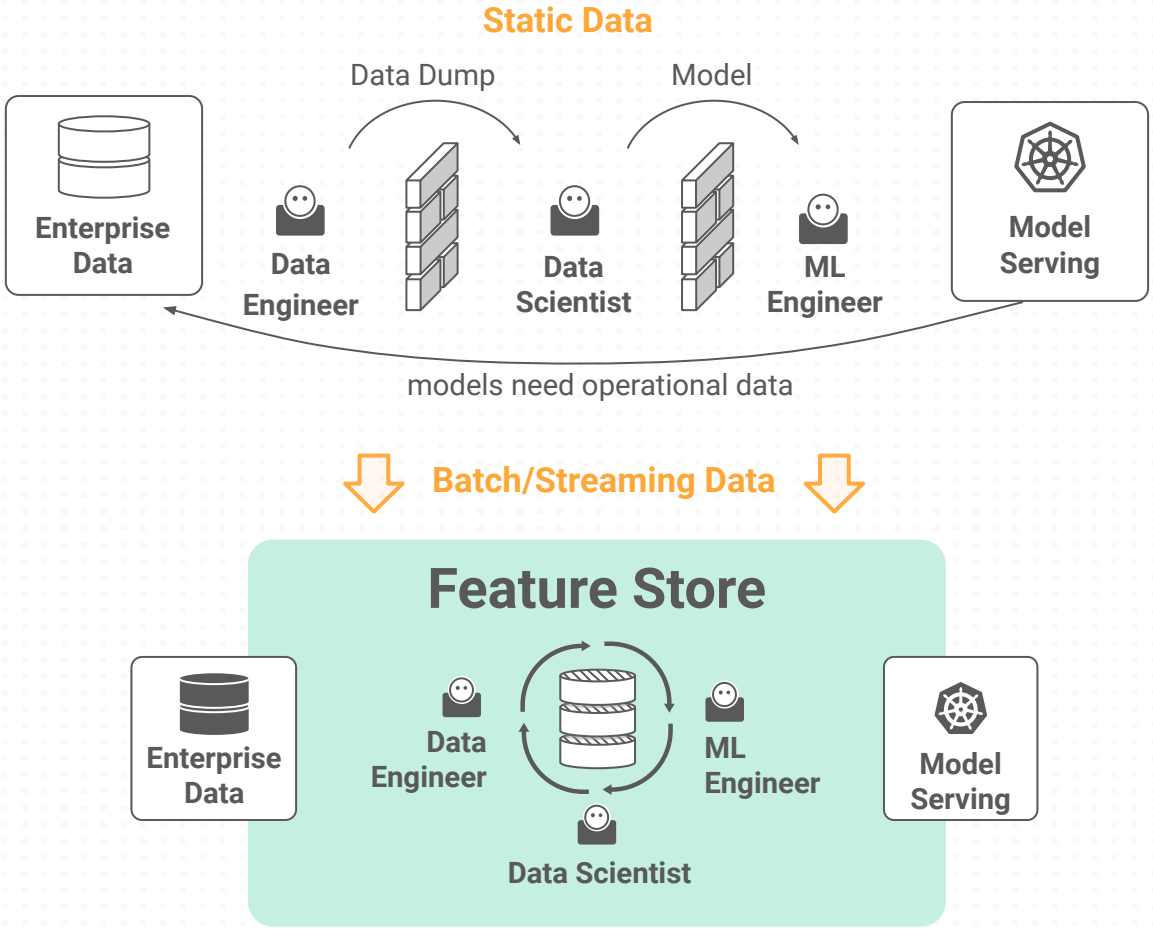


Real time prediction services
Competitive Advantage
High Business Value

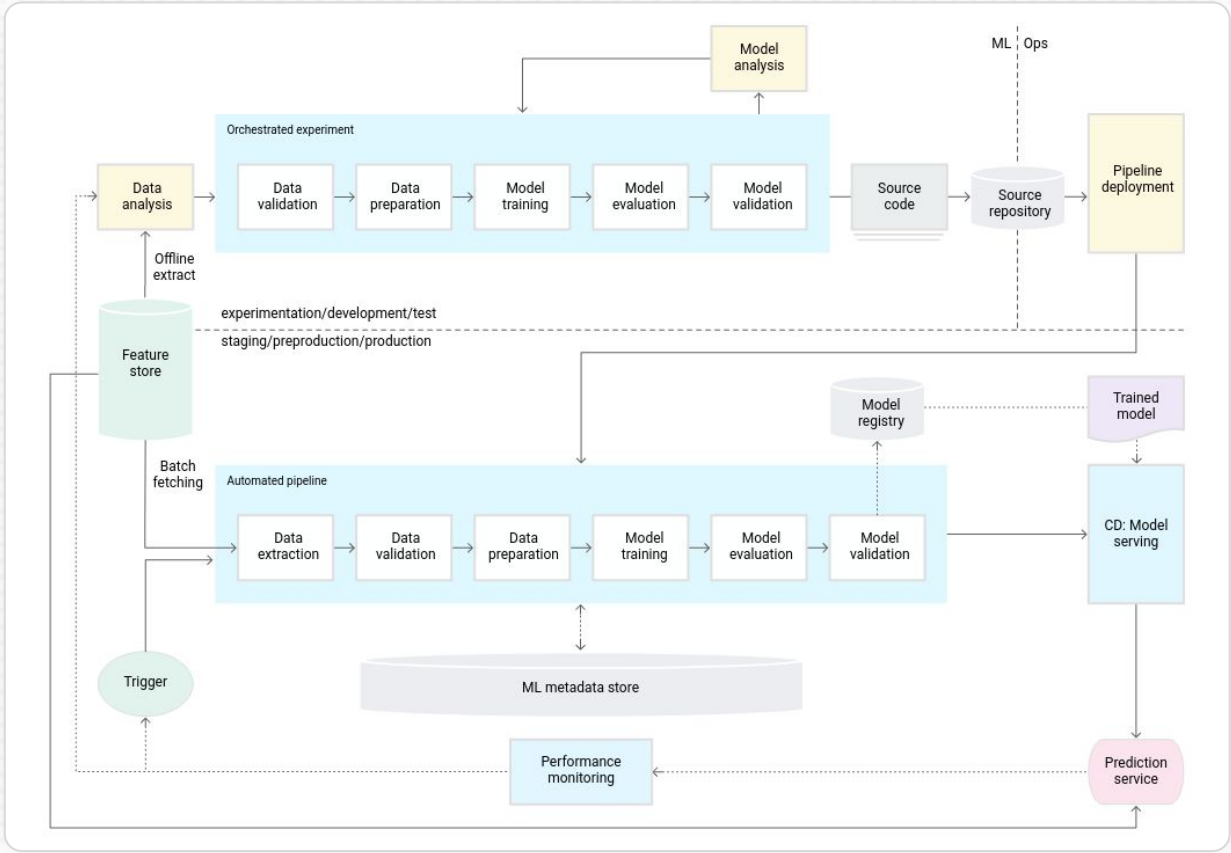
Batch Prediction Services
Spotify Weekly
Demand forecasting, risk estimation, financial planning
Added value
Good Business Value

One-off Reports
Medium Business Value

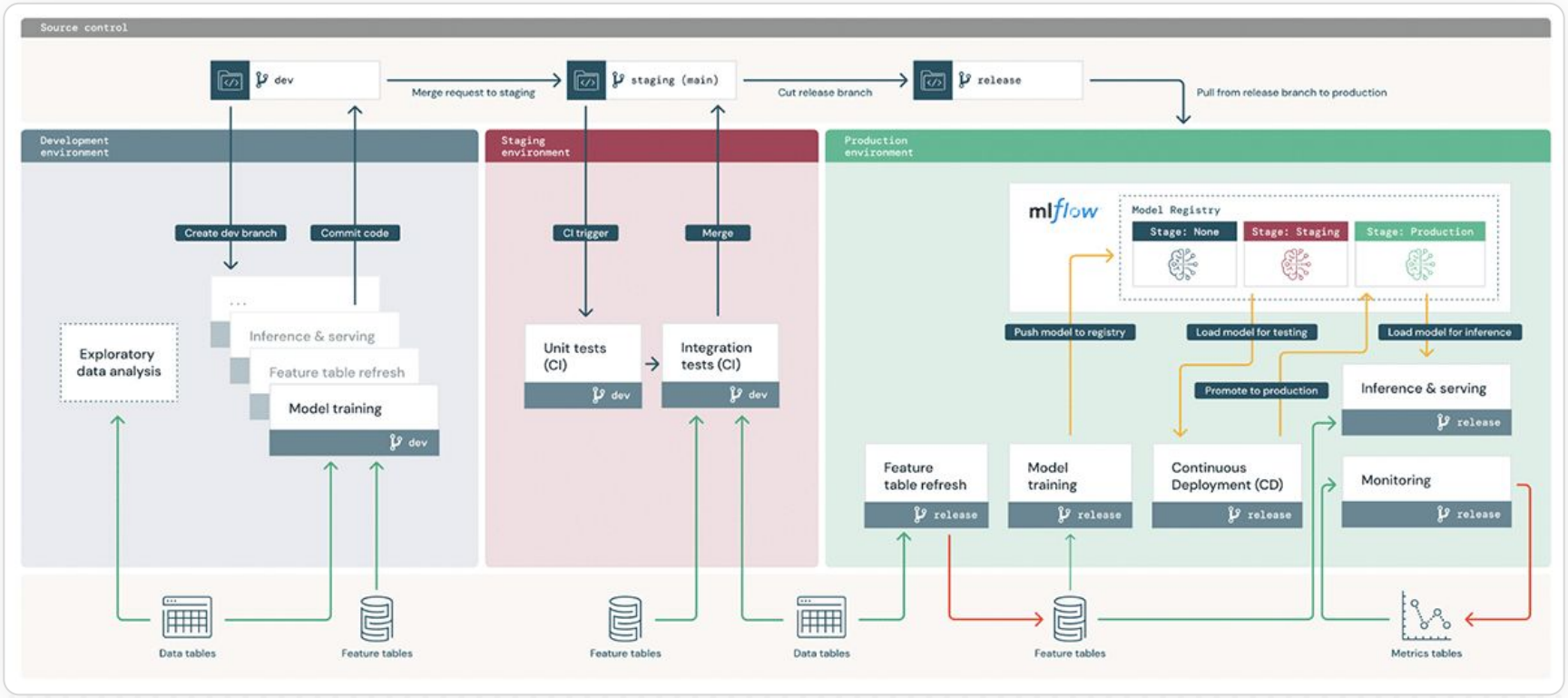

Business Value
STREAMING
Operational ML
with real-time data
BATCH
Operational ML
with historical data
BATCH
Analytical ML
STATIC DATA
Artisanal ML
(Laptop ML)
'Big' Data
Structured
Data
BI:
DESCRIPTIVE & DIAGNOSTIC
ANALYTICS
AI:
PREDICTIVE & PRESCRIPTIVE
ANALYTICS


Static Data
Data Dump
Model
Enterprise Data
Data Engineer
Data Scientist
ML Engineer
Model Serving
models need operational data
Batch/Streaming Data
Feature Store
Enterprise Data
Data Engineer
ML Engineer
Model Serving
Data Scientist

// MLOps according to Google

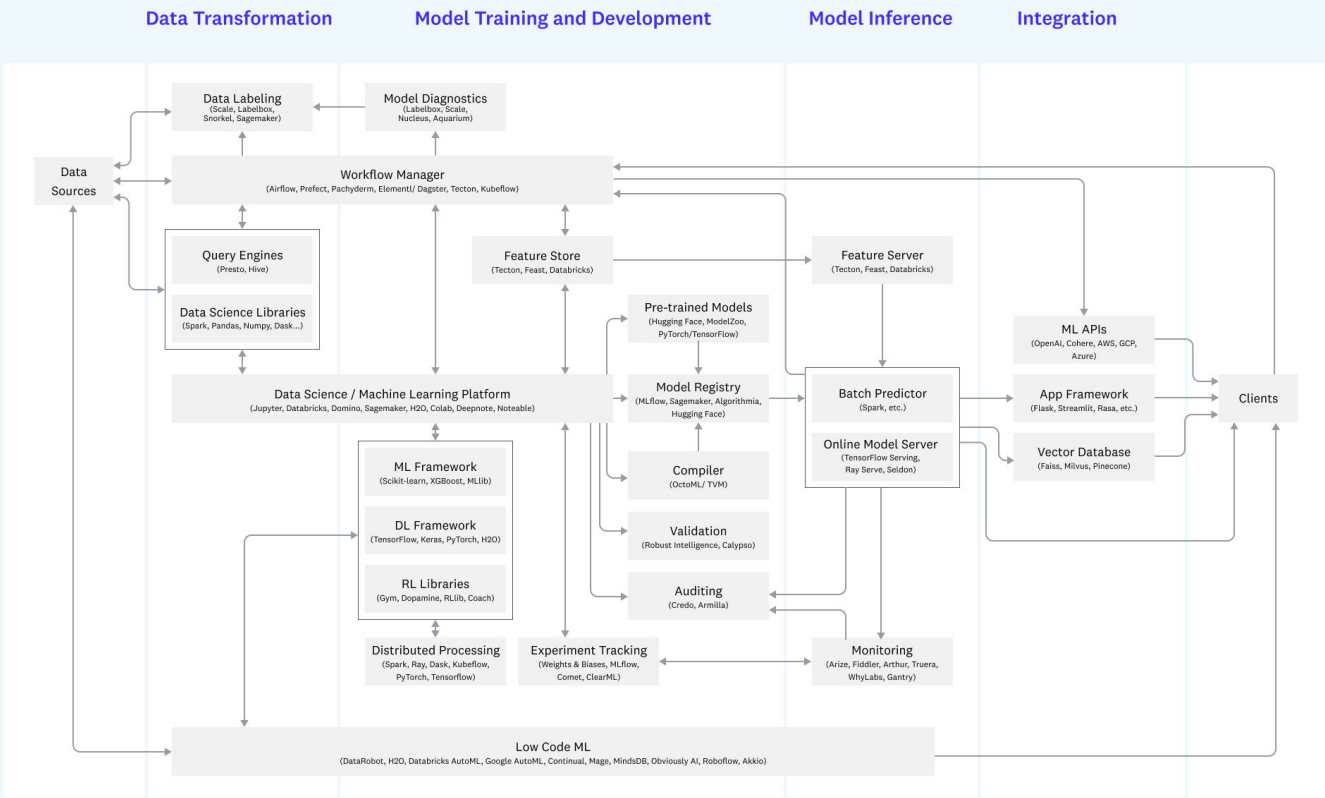


// MLOps according to Databricks

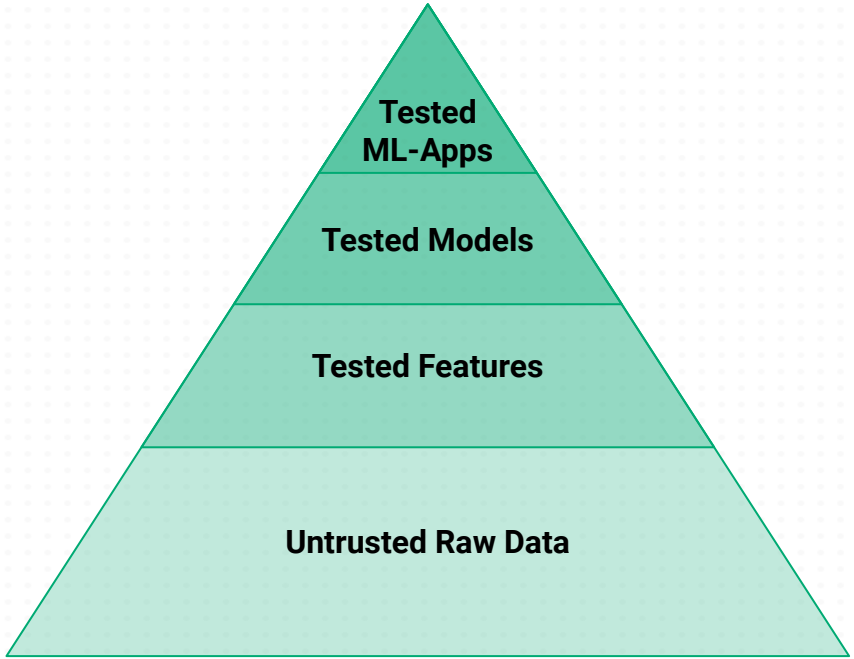


HOPSWORKS

// MLOps according to A16Z

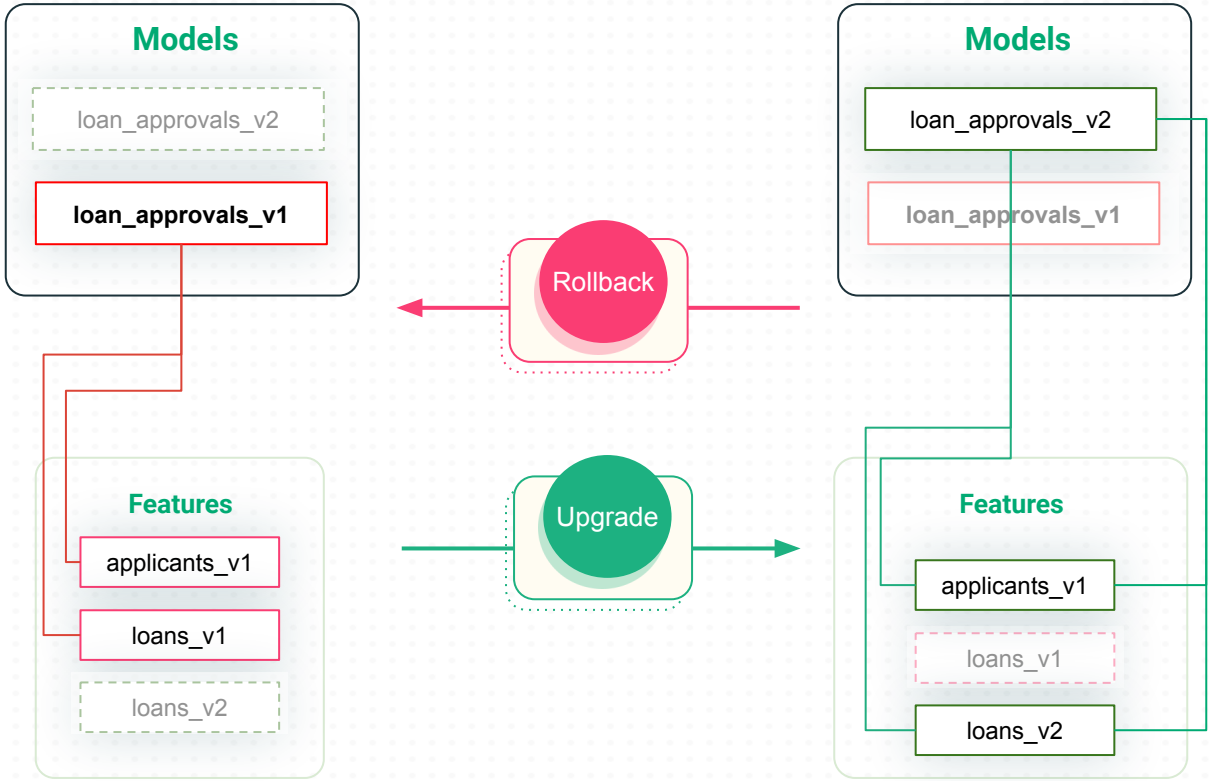


// MLOps according to Hopsworks: Automated Testing

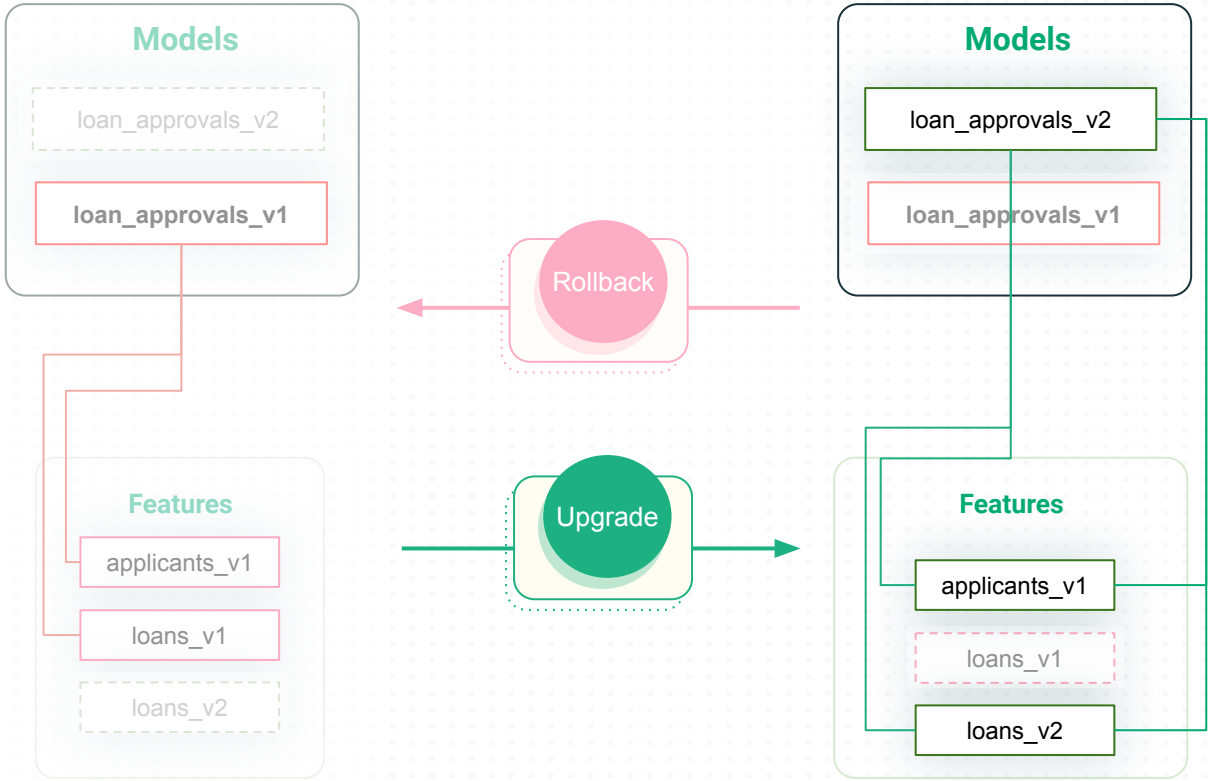


- **ML-Apps**
  build-on models tested with ***A/B tests***
- **Models**
  tested *with* **model validation tests**
- **Features**
  tested with ***data validation and unit tests***
  and corrected with imputation/encoding

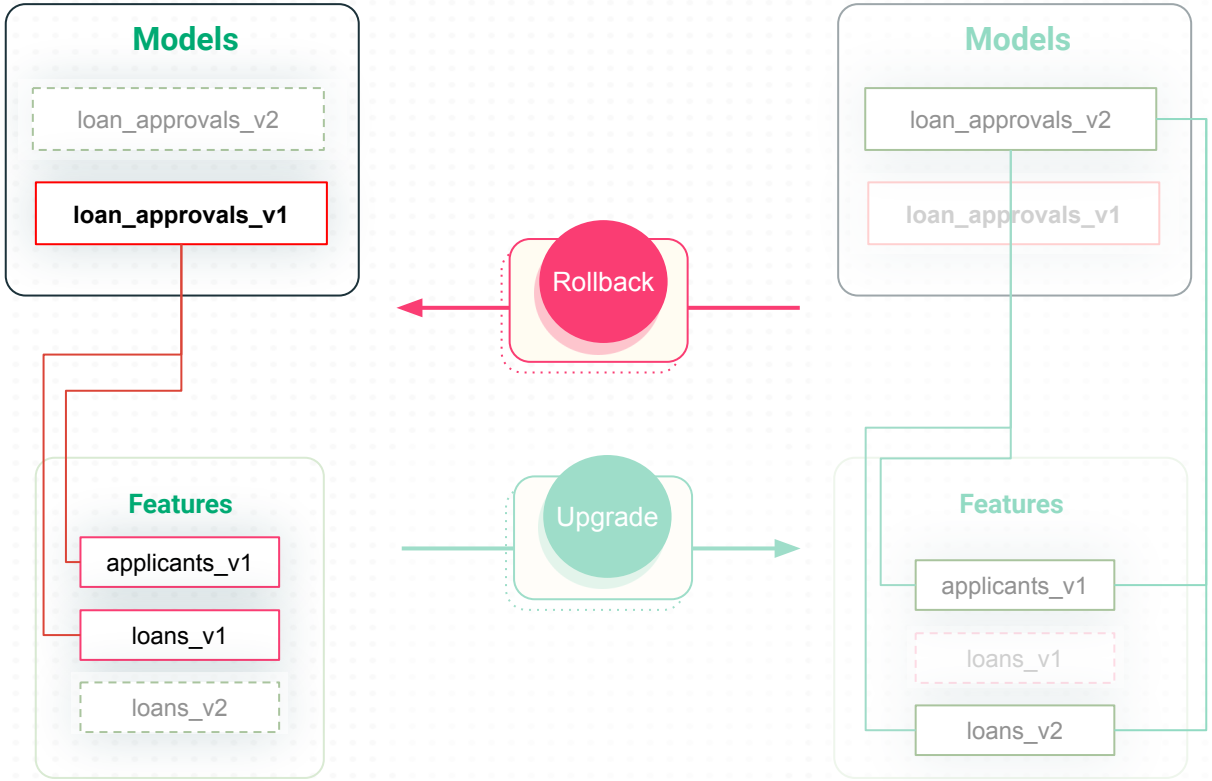
// MLOps according to Hopsworks: Versioning of Features, Models



// MLOps according to Hopsworks: Versioning of Features, Models



// MLOps according to Hopsworks: Versioning of Features, Models



2

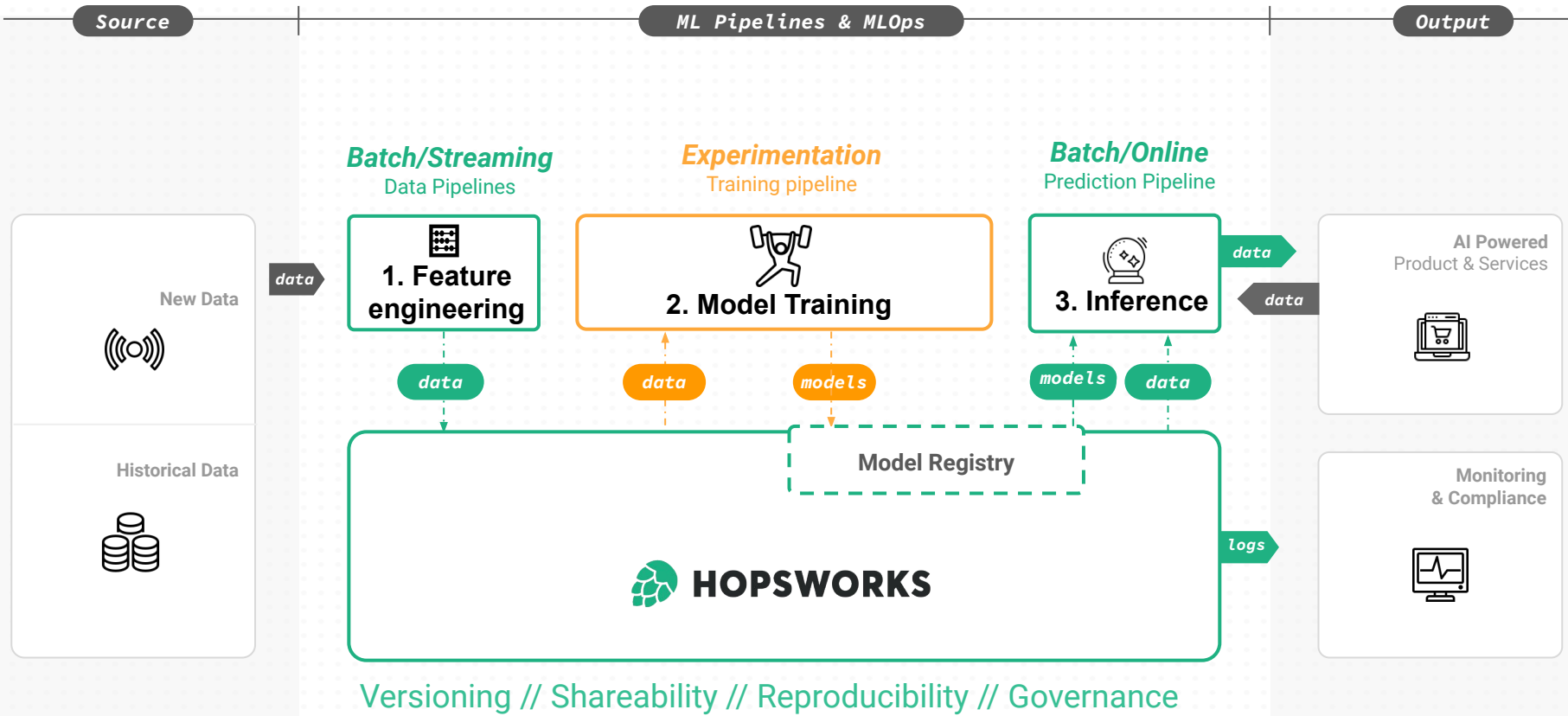
Decompose the monolithic ML pipeline into Feature, Training, Inference pipelines

MLOps made easy

// FTI Pipelines

To build a Prediction Service, you need to write:

`Data`	`=>`	*1.* **Feature Pipeline**	`=> Features/Labels`
`Features/Labels`	`=>`	*2.* **Training Pipeline**	`=> Model`
`Model, Features`	`=>`	*3.* **Inference Pipeline**	`=> Predictions`


Source
ML Pipelines & MLOps
Output
Batch/Streaming
Data Pipelines
Experimentation
Training pipeline
Batch/Online
Prediction Pipeline
New Data
Historical Data
data
1. Feature engineering
2. Model Training
3. Inference
data
data
AI Powered
Product & Services
data
data
models
models
data
Model Registry
HOPSWORKS
logs
Monitoring
& Compliance
Versioning // Shareability // Reproducibility // Governance

3

MVPs

ASSESSMENT

HOPSWORKS

Source

ML Pipelines & MLOps

Output

Data Mgmt

Where is your new and historical data?

Feature Pipeline

What are your features & how are they created?

Training Pipeline

How are your models trained?

Inference Pipeline

How do you make predictions?

AI-Enabled Apps

How are Predictions consumed?

Features (Labels), Models

HOPSWORKS

Source

Where is your new and historical data?

Data Mgmt

Examples:
Data Warehouse, Object/File storage. Other data stores.

ML Pipelines & MLOps

What are your features & how are they created?

Feature Pipeline

Examples:
Airflow, Pandas, Polars, Spark, SQL, Flink.

How are your models trained?

Training Pipeline

Examples:
Scikit-learn, TensorFlow, PyTorch, XGBoost

How do you make predictions?

Inference Pipeline

Examples:
Airflow/Python /Spark Programs, Model Serving

Features, Labels, Models

Output

How are predictions consumed?

AI-Enabled Apps

Examples:
Interactive Apps, Dashboard, M2M Systems

HOPSWORKS

Source

Where is your new and historical data?

Data Mgmt

TRL Assessment
1 - Nothing
2 - Prototype
3 - Accessible
4 - Well-structured
5 - Versioned & documented

ML Pipelines & MLOps

What are your features & how are they created?

Feature Pipeline

TRL Assessment
1 - Nothing
2 - Prototype
3 - Accessible
4 - Well-structured
5 - Versioned & documented

How are your models trained?

Training Pipeline

TRL Assessment
1 - Nothing
2 - Prototype
3 - Accessible
4 - Well-structured
5 - Versioned & documented

How do you make predictions?

Inference Pipeline

TRL Assessment
1 - Nothing
2 - Prototype
3 - Accessible
4 - Well-structured
5 - Versioned & documented

Features, Labels, Models

Output

How are predictions consumed?

AI-Enabled Apps

TRL Assessment
1 - Nothing
2 - Prototype
3 - Accessible
4 - Well-structured
5 - Versioned & documented

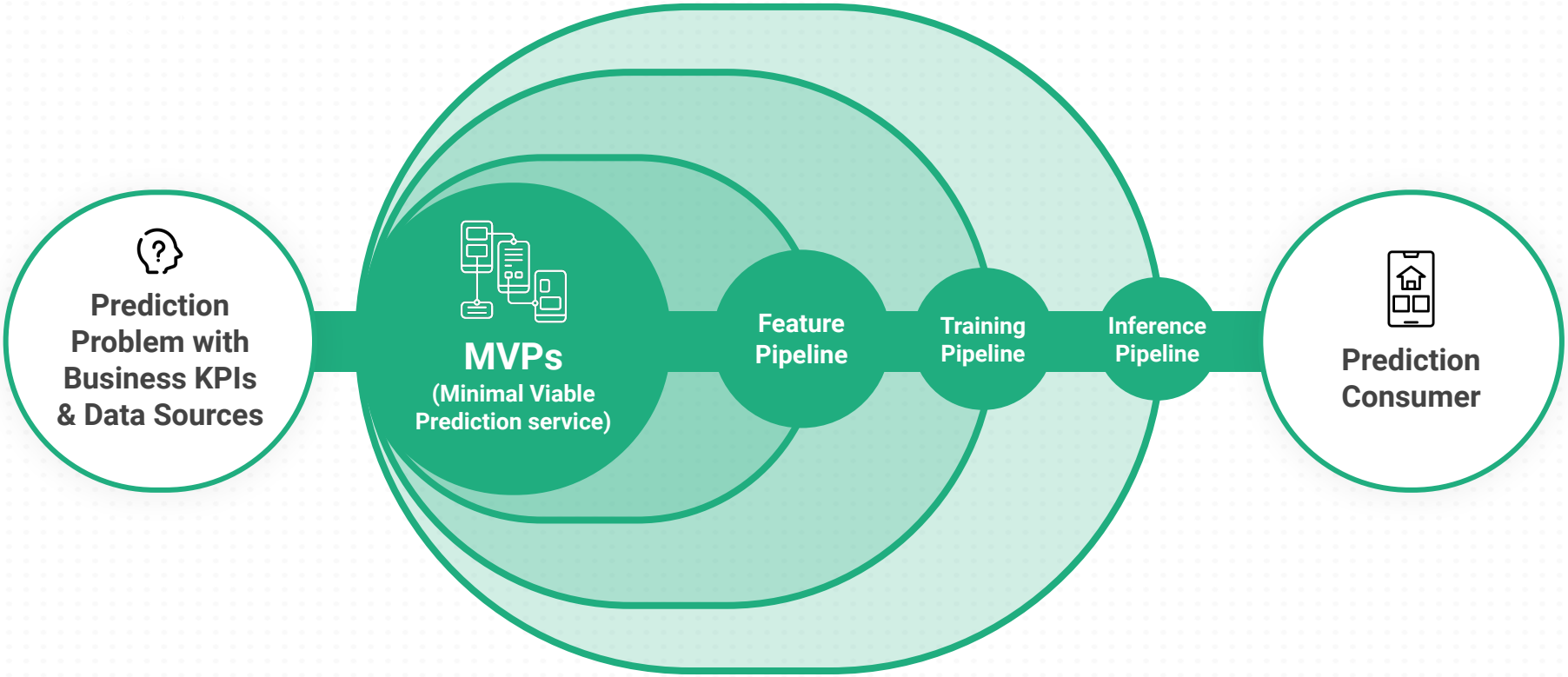
4

CONCEPTS

MVPS and FTI Pipelines

Feature, Training, Inference Pipelines

Minimal Viable Prediction service



MVPS (minimal viable prediction service) in two weeks

1. **Identify the Prediction Problem**
   a. ROI and measurable business KPIs tied to ML Proxy metrics
2. **Identify the Data Sources**
3. **Write the Feature Pipeline(s) (and orchestrate them)**
4. **Write a Training Pipeline (start with a baseline)**
5. **Write an Inference Pipeline (and orchestrate or serve)**
   a. For your batch or online ML Application
   b. Use a UI to showcase the value of your model to stakeholders

Hopsworks Feature Store

Data Science

Azure Machine Learning

DataRobot

Amazon SageMaker

Google AI

databricks

Model Serving

Google AI

Azure Machine Learning

DataRobot

databricks

SELDON

Amazon SageMaker

Feature Engineering

DataRobot

python

Spark

Apache Airflow

snowflake

Data Discovery & Compliance

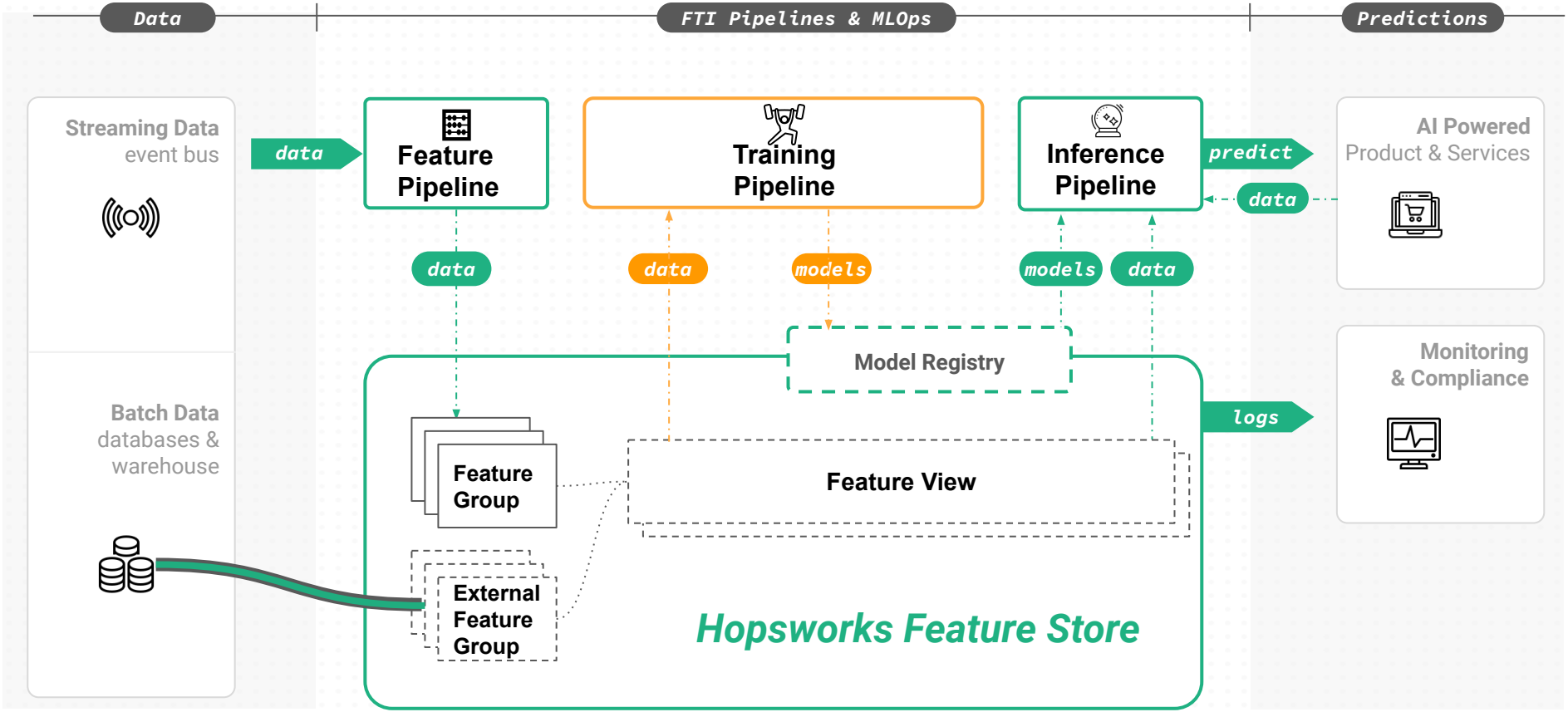
Informatica

collibra

Alation

HOPSWORKS

Hopsworks: write to Feature Groups, read from Feature Views



What we are building

https://github.com/logicalclocks/hopsworks-tutorials

WORKSHOP

1

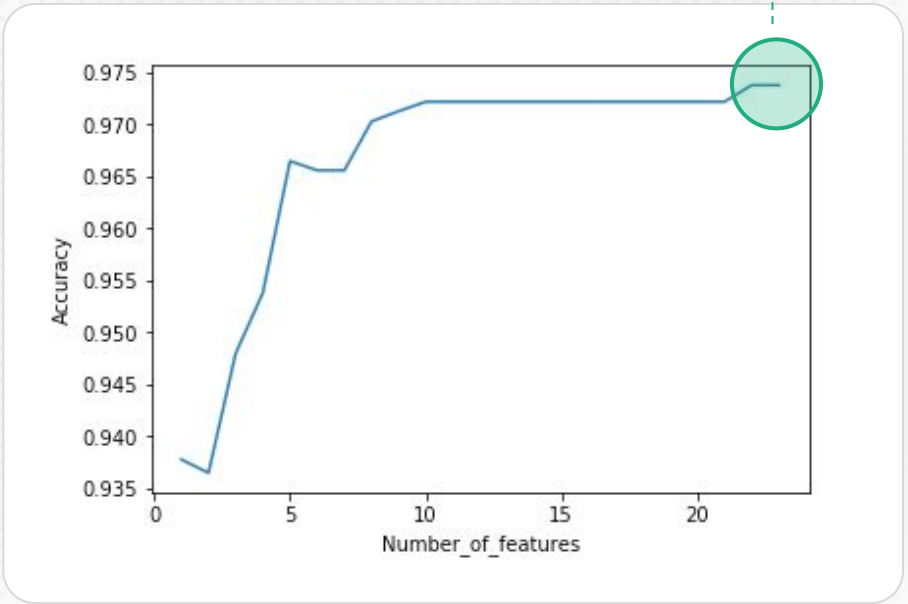
WORKSHOP

Identify the Prediction Problem and the Business KPIs

- **Data science projects start with a business problem** and one or more metrics that should be improved to generate a ROI (return on investment)

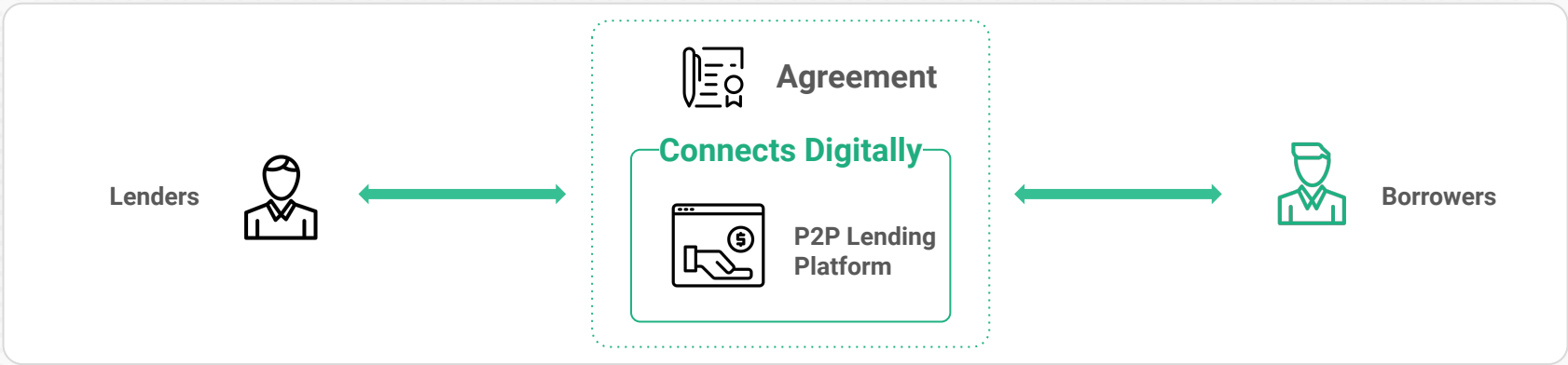
- **Define the business metric** you want to impact and the **ML proxy metric** you are going to optimize (aka target metric).

- Your job is to **translate the business problem** into the right data science problem.



Loan Approvals Prediction Problem

- LendingClub is the world's **largest peer-to-peer lending platform**.
  - It makes data on its loans publicly available
- We would like to **predict** if a given loan **should be approved or not**



2

WORKSHOP

Identify the Data Sources

Feature Group - loans

id	issue_d	int_rate	purpose	loan_amnt	loan_status
<entity_id>	*<event_time>*	*<numerical feature>*	*<categorical feature>*	*<numerical feature>*	*<categorical feature>*
string	*datetime*	*double*	*string*	*double*	*boolean*
122	2022-01-01	5.3	debt_consolidation	$142.34	fully_paid
123	2022-04-01	2.3	wedding	$12.34	charged_off
124	2022-07-01	3.1	credit_card	$66.29	charged_off
125	2022-10-01	4.3	debt_consolidation	$112.33	fully_paid

Feature Types

Row

Feature value.
Store unencoded to maximize reuse over many models.

Feature vector.
Row of feature values

Possible Label
Target for a prediction problem

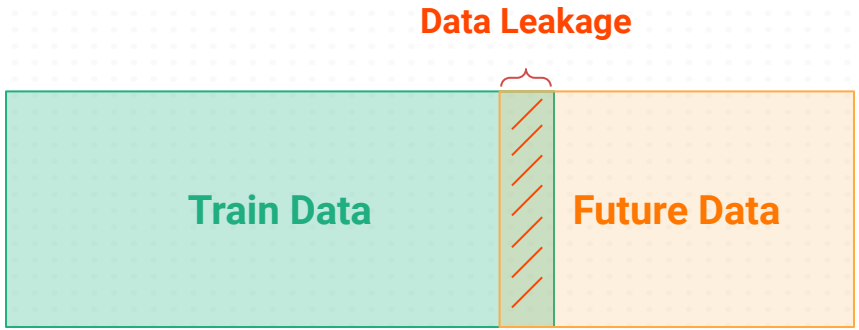
Feature Group - applicants

id	earliest_cr_line	verification_status	home_ownership	annual_inc
<entity_id>	*<event_time>*	*<categorical feature>*	*<categorical feature>*	*<numerical feature>*
string	*datetime*	*string*	*string*	*double*
122	2001-01-01	Not Verified	mortgage	$133,340
123	2011-04-13	Verified	rent	$129,023
124	2007-08-22	Not Verified	mortgage	$77,033
125	2000-11-21	Source Verified	own	$88,929

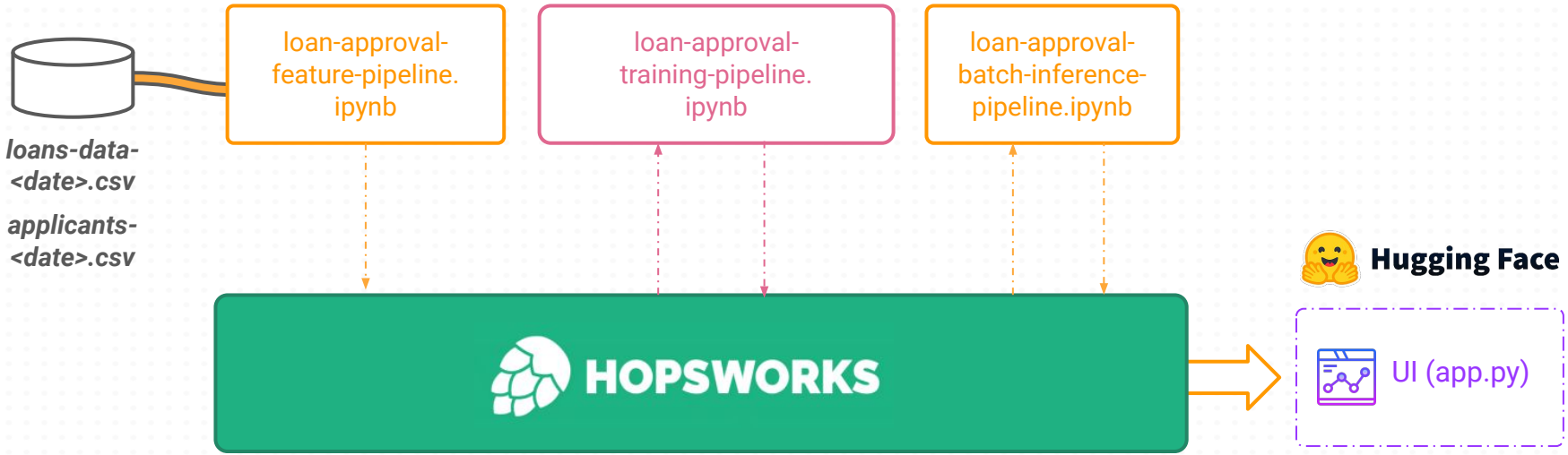
Data sources should have timestamped data

When you train a model, if you use the latest credit report for the applicant (not the credit report at the time the loan was issued), you can have data leakage.

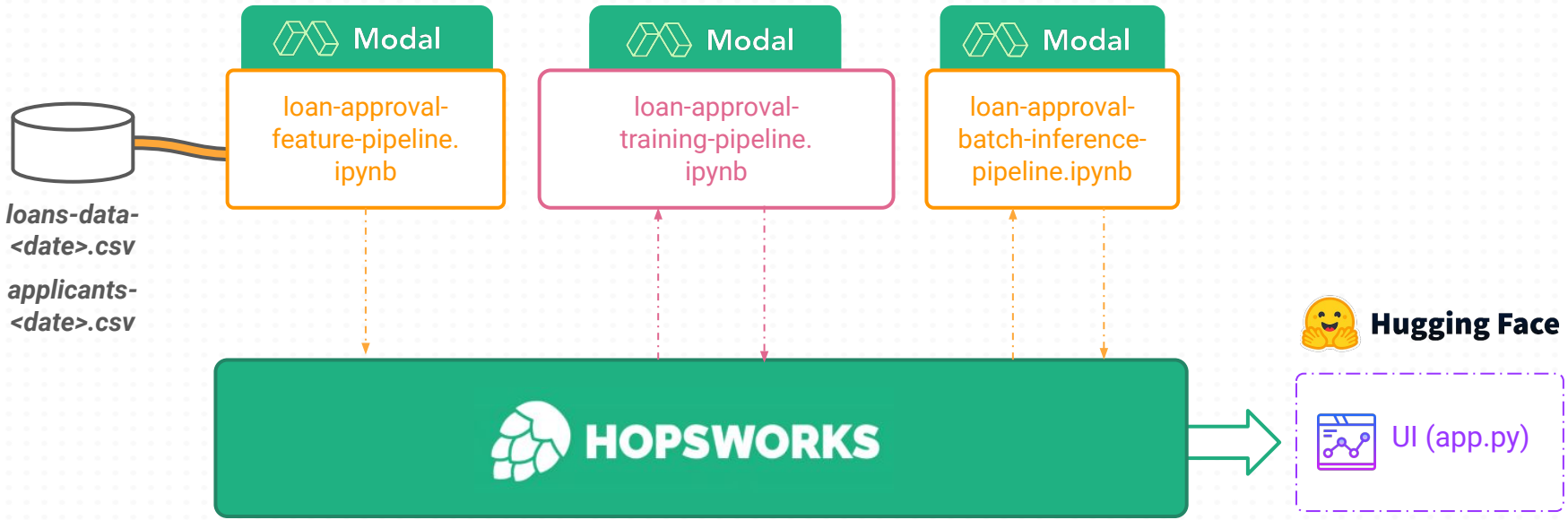
```
/loans-Apr-2008.csv
/loans-May-2008.csv
…
/loans-Aug-2016.csv
/loans-Sep-2016.csv
```

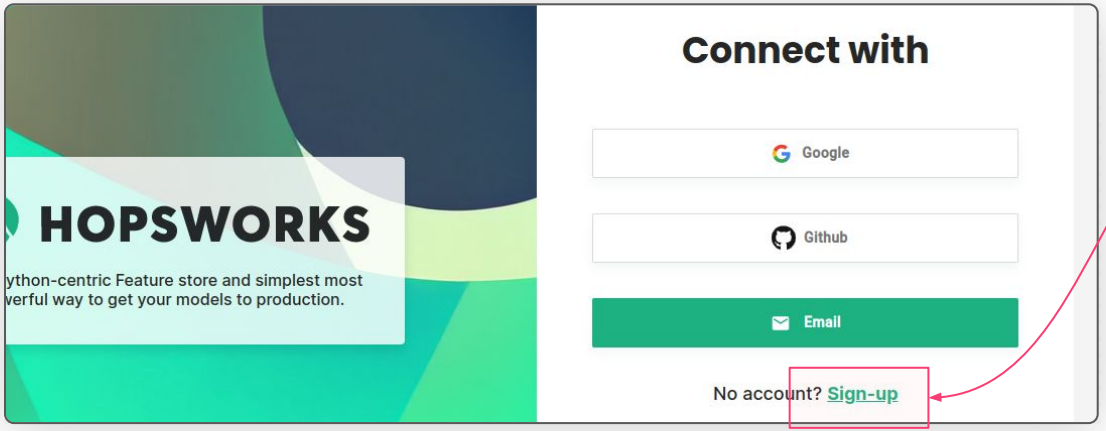


A Loan Approval ML-powered App

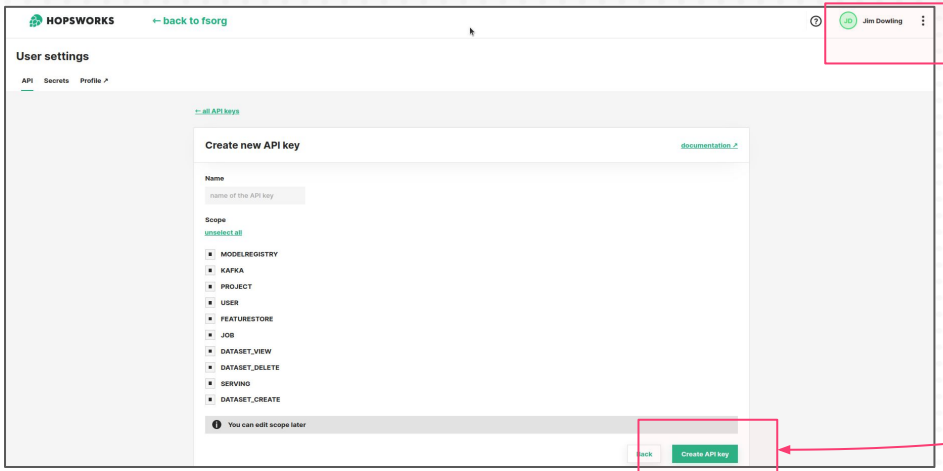


An Orchestrated Loan Approval ML-powered App





1. First, create an account on https://app.hopsworks.ai



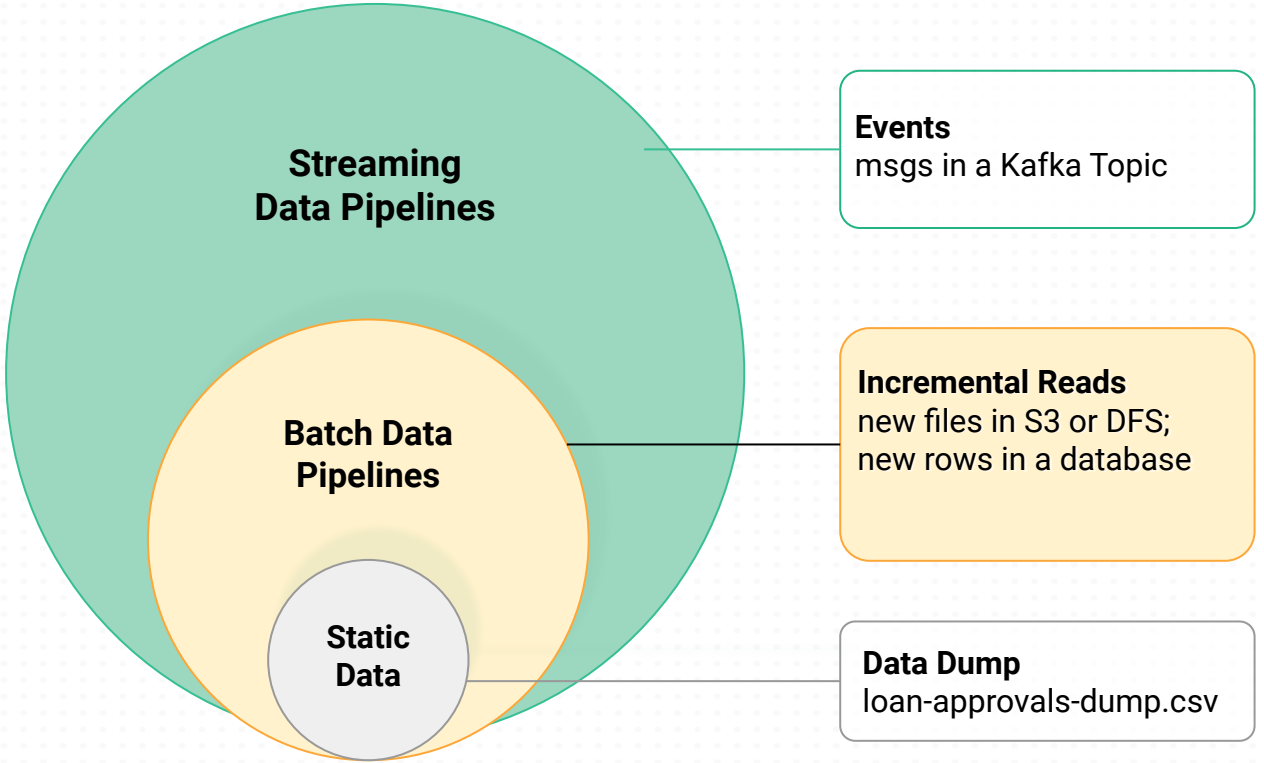
2. Click on “User Settings”

3. Create and Save an “API Key”

3

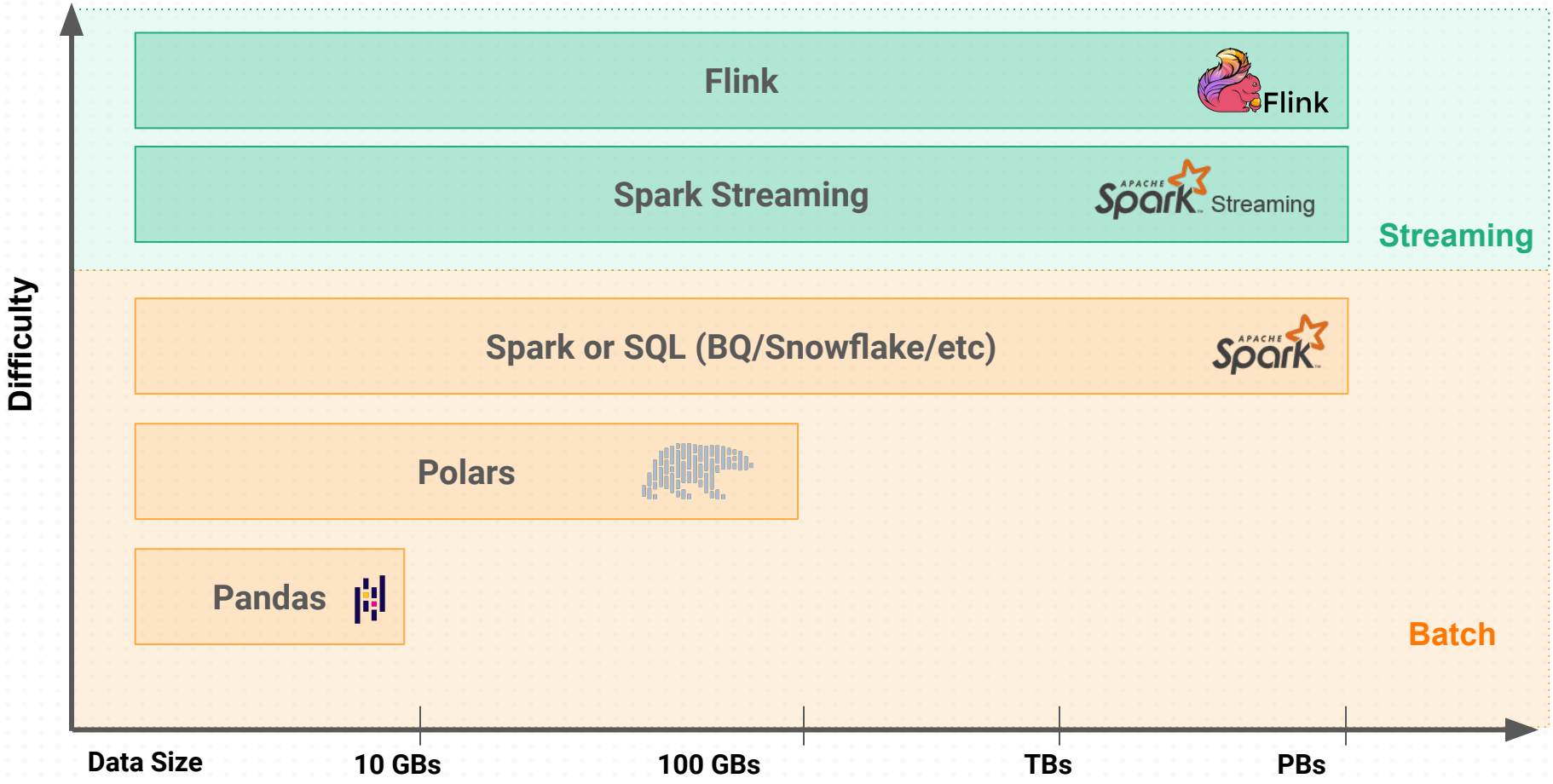
WORKSHOP

Feature Pipelines


Streaming
Data Pipelines
Batch Data
Pipelines
Static
Data
Events
msgs in a Kafka Topic
Incremental Reads
new files in S3 or DFS;
new rows in a database
Data Dump
loan-approvals-dump.csv

HOPSWORKS

Which framework for your feature pipeline?



HOPSWORKS

Loans/Applicants Feature Pipeline in Pandas

```
def main():
  loans_df = # 1. read today's loans data in as a Pandas DataFrame

  # 2. create features for loans in Pandas DataFrame

  loans_fg = fs.get_or_create_feature_group(name="loans",
                                            version=1,
                                            description="Lending Club Loans",
                                            online_enabled=True,
                                            primary_key=['id'],
                                            event_time='issue_d',
                                            expectation_suite=expectation_suite
  )
  loans_fg.insert(loans_df) # 3. write Pandas DataFrame to Feature Group

  # …
```

HOPSWORKS

Loans/Applicants Feature Pipeline in Pandas

```
def main():
…
  applicants_df = # 1. read today's applicants data in as a Pandas DataFrame

  # 2. create features for applicants in Pandas DataFrame

  applicants_fg = fs.get_or_create_feature_group(name="applicants",
                                                 version=1,
                                                 description="Loan Applicants",
                                                 online_enabled=True,
                                                 primary_key=['id'],
                                                 event_time='earliest_cr_line',
                                                 partition_key=['earliest_cr_line_year']
  )
  applicants_fg.insert(applicants_df) # 3. write DataFrame to Feature Group

  # …
}
```

Feature Functions in a Python module

features/loans.py

```
def zipcode(postcode):
    zip_code=int(postcode)
    l=len(str(zip_code))
    if l==5:
        return str(zip_code)
    return "0"


….
```

features/applicants.py

```
def pub_rec_bankruptcies(number):
    if number == 0.0:
        return 0
    elif number >= 1.0:
        return 1
    else:
        return number


….
```

One Python module per Feature Group

HOPSWORKS

Schedule your Feature Pipeline with Modal

Modal

```
stub = modal.Stub("loans_daily")

image = modal.Image.debian_slim().pip_install(["hopsworks"])


@stub.function(image=image, schedule=modal.Period(days=1),

secret=modal.Secret.from_name("jim-hopsworks-ai"))
def main():

    …


if __name__ == "__main__":

    stub.deploy("loans_daily")

    with stub.run():

            main()
```

Define program dependencies and program schedule, env variables

Deploy main() function as a scheduled program on modal

Limitations of cron-based scheduling with Modal

- What happens if a run fails?
- How do i re-execute a failed run?
- How do I schedule in response to an event (like a file written)?

For production, you probably need a more feature-rich scheduler like Airflow/Dagster/etc for your feature pipelines

3

WORKSHOP

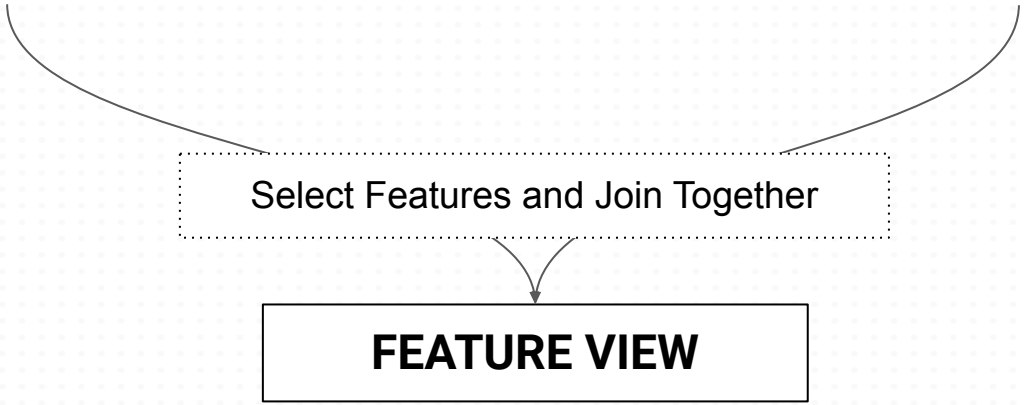
Training Pipeline

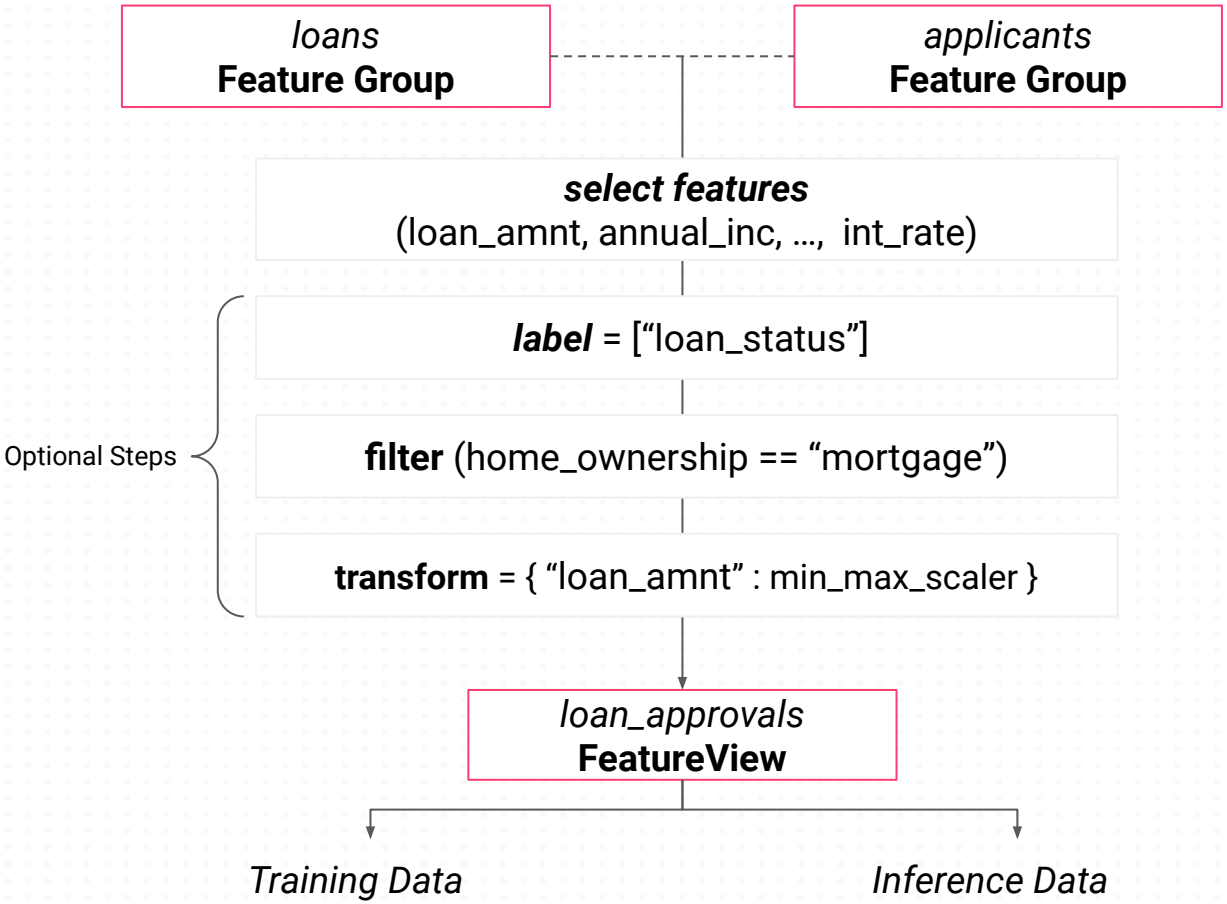
Feature Selection

Useful Feature	Redundant Feature	Irrelevant Feature	Prohibited Feature	Infeasible Feature
It has predictive power for my prediction problem	A similar feature is already selected	The feature has no predictive power	Feature cannot be used. May be context dependent	Useful feature that can't be computed for some reason

id	issue_d	int_rate	purpose	loan_amnt	loan_status
<entity_id>	*<event_time>*	*<numerical feature>*	*<categorical feature>*	*<numerical feature>*	*<categorical feature>*
string	*datetime*	*double*	*string*	*double*	*boolean*
122	2022-01-01	5.3	debt_consolidation	$142.34	fully_paid
123	2022-04-01	2.3	wedding	$12.34	charged_off
124	2022-07-01	3.1	credit_card	$66.29	charged_off
125	2022-10-01	4.3	debt_consolidation	$112.33	fully_paid

id	earliest_cr_line	verification_status	home_ownership	loan_status	annual_inc
<entity_id>	*<event_time>*	*<categorical feature>*	*<categorical feature>*	*<categorical feature>*	*<numerical feature>*
string	*datetime*	*string*	*string*	*boolean*	*double*
122	2001-01-01	Not Verified	mortgage	fully_paid	$133,340
123	2011-04-13	Verified	rent	charged_off	$129,023
124	2007-08-22	Not Verified	mortgage	charged_off	$77,033
125	2000-11-21	Source Verified	own	fully_paid	$88,929




loans
Feature Group
applicants
Feature Group
select features
(loan_amnt, annual_inc, ..., int_rate)
label = ["loan_status"]
Optional Steps
filter (home_ownership == "mortgage")
transform = { "loan_amnt" : min_max_scaler }
loan_approvals
FeatureView
Training Data
Inference Data

HOPSWORKS

Feature Selection - create a Feature View

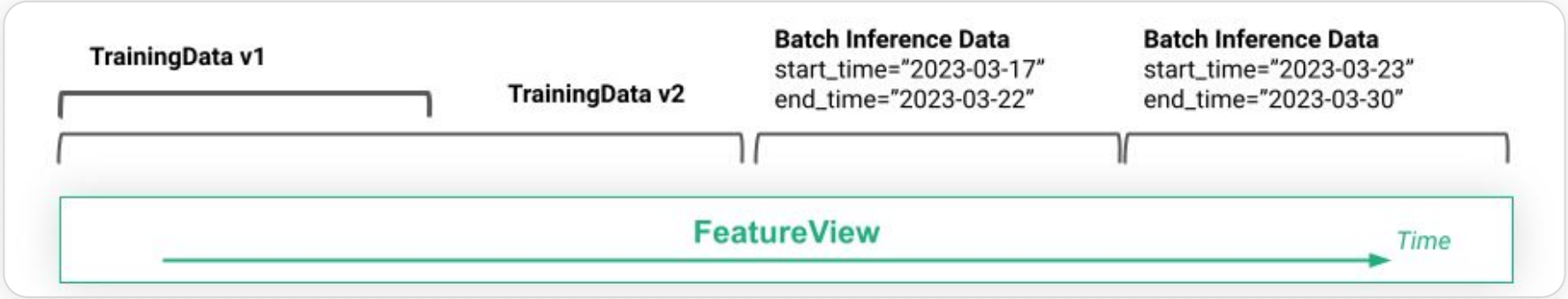
```
fg_loans = fs.get_feature_group(name="loans", version=1)
fg_applicants = fs.get_feature_group(name="applicants", version=1)
query = fg_loans.select_except(["issue_d", "id"]).join(\
            fg_applicants.select_except(["earliest_cr_line", "id"]))
```

select features from feature groups

```
fv = fs.create_feature_view(name="loans_approvals",
            version=1,
            description="Loan applicant data",
            labels=["loan_status"],
            query=query
            )
```

create feature view from selected features

Feature Views - Create Snapshots for Training or Inference



HOPSWORKS

Feature Views - Create Snapshots for Training or Inference

Feature View - loan_approvals

id	issue_d	loan_amnt	home_ownership	…	loan_status
<entity_id>	*<event_time>*	*min_max_scaler*	*ordinal_encoder*	*…*	*min_max_scaler*
string	*datetime*	*double*	*string*	*…*	*boolean*
1	2022-01-01	$142.34	mortgage	…	fully_paid
2	2022-04-01	$12.34	rent	…	charged_off
3	2022-07-01	$66.29	mortgage	…	charged_off
4	2022-10-01	$112.33	own	…	fully_paid
5	2022-11-01	$222.01	mortgage	…	fully_paid
6	2022-12-01	$9333.00	rent	…	fully_paid
7	2023-01-01	$773.00	mortgage	…	fully_paid

Time

Training Data-v1

Training Data-v2

Batch Inference Data

Feature vector

Feature Views - Model-Dependent Transformations

FeatureView for Model-FeedForwardNet

id	issue_d	loan_amnt	home_ownership	...	loan_status
		StandardScaler	*OrdinalEncoder*	...	*LabelEncoder*
<entity_id>	*<event_time>*	*double*	*string*	...	*boolean*

Model-Dependent Transformations

FeatureView for Model-XGBoost

id	issue_d	loan_amnt	home_ownership	...	loan_status
		<none>	*OrdinalEncoder*	...	*LabelEncoder*
<entity_id>	*<event_time>*	*double*	*string*	...	*boolean*

Model-Dependent Transformations

FeatureView for Model-LogisticRegression

id	issue_d	loan_amnt	home_ownership	...	loan_status
		MinMaxScaler	*OrdinalEncoder*	...	*LabelEncoder*
<entity_id>	*<event_time>*	*double*	*string*	...	*boolean*

Model-Dependent Transformations

Model Training with Feature Views

```
X_train, X_test, y_train, y_test = fv.train_test_split(test_size=0.2)
```

get train & test set

```
numeric_transformer = Pipeline( \
  steps=[("imputer", SimpleImputer(strategy="median")), ("scaler", StandardScaler())])

categorical_transformer=Pipeline(steps=[("encoder",
OneHotEncoder(handle_unknown="ignore"))])

preprocessor = ColumnTransformer(transformers=[ \
        ("num", numeric_transformer, numeric_feature_ids), \
        ("cat", categorical_transformer, categorical_feature_ids)])
```

model-dependent feature encoding in a pipeline obj

```
clf = Pipeline(steps=[("preprocessor", preprocessor), ("classifier",
LogisticRegression())])
clf.fit(X_train, y_train['loan_status'].ravel())
```

Pre-process then train model

HOPSWORKS

Store Models in a Model Registry

```
accuracy = roc_auc_score(y_test, clf.predict(X_test))
joblib.dump(clf, 'lending_model/lending_model.pkl')

input_schema = Schema(X_test)
output_schema = Schema(y_test)

fraud_model = mr.sklearn.create_model("lending_model",
    metrics={'accuracy': accuracy},
    input_example=X_test.sample().to_numpy(),
    model_schema=ModelSchema(input_schema=input_schema, output_schema=output_schema))

fraud_model.save('lending_model')
```

serialize model

get model schema from DFs

package up model and upload to model registry

4

WORKSHOP

Batch and Online Inference Pipelines

HOPSWORKS

Batch Inference Pipeline

```
fv = fs.get_feature_view(name="loans_approvals", version=fv_version)
df = feature_view.get_batch_data()
```

download inference data

```
mr = project.get_model_registry()
model = mr.get_model("lending_model", version=model_version)
model_dir = model.download()
model = joblib.load(model_dir + "/lending_model.pkl")
```

download model

```
predictions_df = model.predict(df)
```

make predictions

Online Inference Pipeline

```
fv = fs.get_feature_view(name="loans_approvals", version=fv_version)
mr = project.get_model_registry()
model = mr.get_model("lending_model", version=model_version)
model_dir = model.download()
model = joblib.load(model_dir + "/lending_model.pkl")
```

get feature view & download model

```
def approve_loan(id, term, purpose, zip_code, loan_amnt, int_rate):
    arr = fv.get_feature_vector({"id": id}, passed_features={"term": term,..})
    y_pred = model.predict(np.asarray(arr).reshape(1, -1))
```

make a prediction with precomputed and user-supplied features

5

WORKSHOP

Prediction Service / ML-enabled App

HOPSWORKS

Simple UI in Python with Gradio

```
demo = gr.Interface(
    fn=approve_loan,
    title="Loan Approval",
    description="Enter your details to see if your loan is approved or not.",
    inputs=[
        gr.Number(label="id"),
        gr.Dropdown(term, label="term"),
        gr.Dropdown(purpose, label="purpose"),
        gr.Number(label="zip_code"),
        gr.Number(label="loan_amnt"),
        gr.Number(label="int_rate"),
        ],
    ],
    outputs=gr.Image(type="pil"))
```

Call 'approve_loan' function to make the prediction

user-supplied feature values - entered using the UI

5

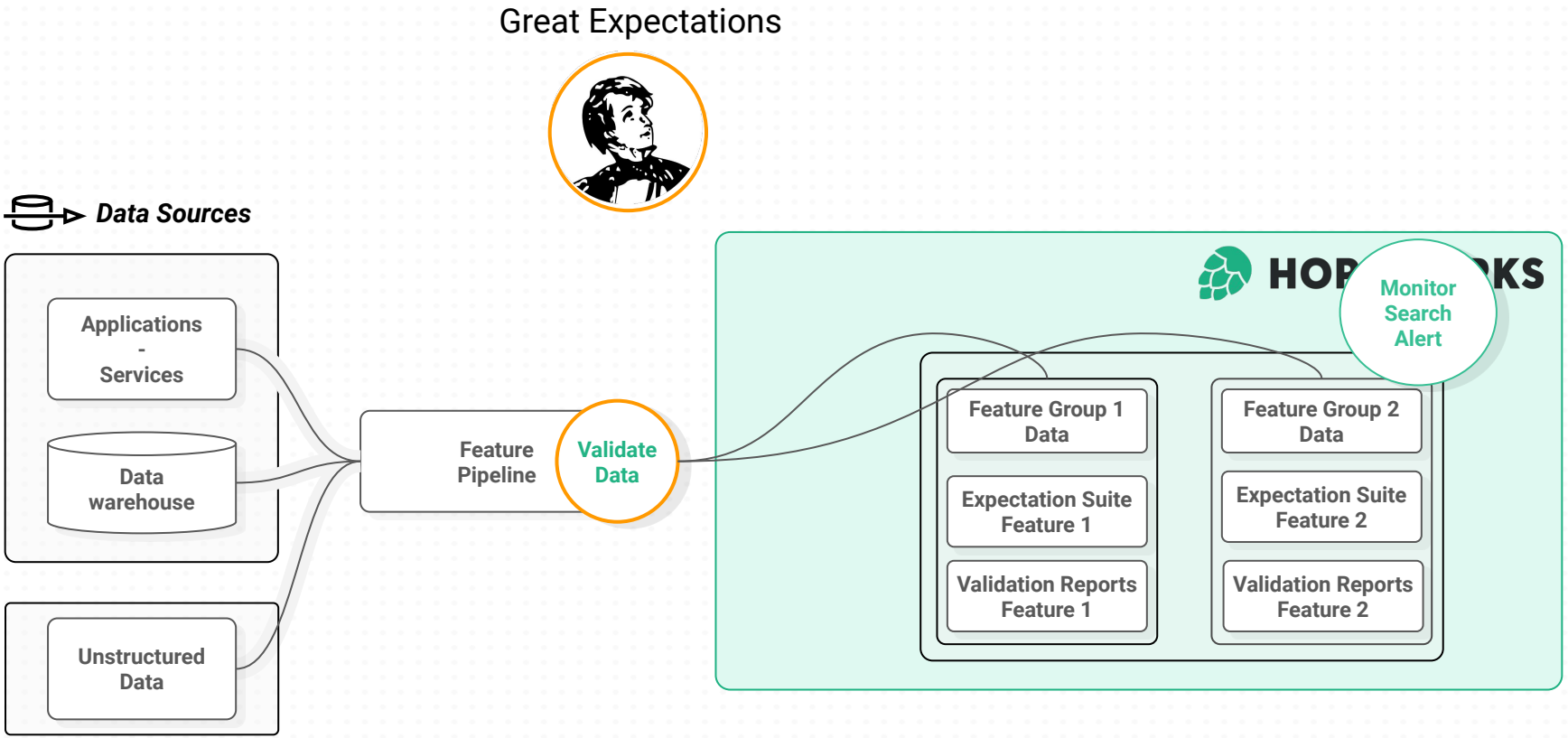
WORKSHOP

Principles of MLOps

MLOps Principles

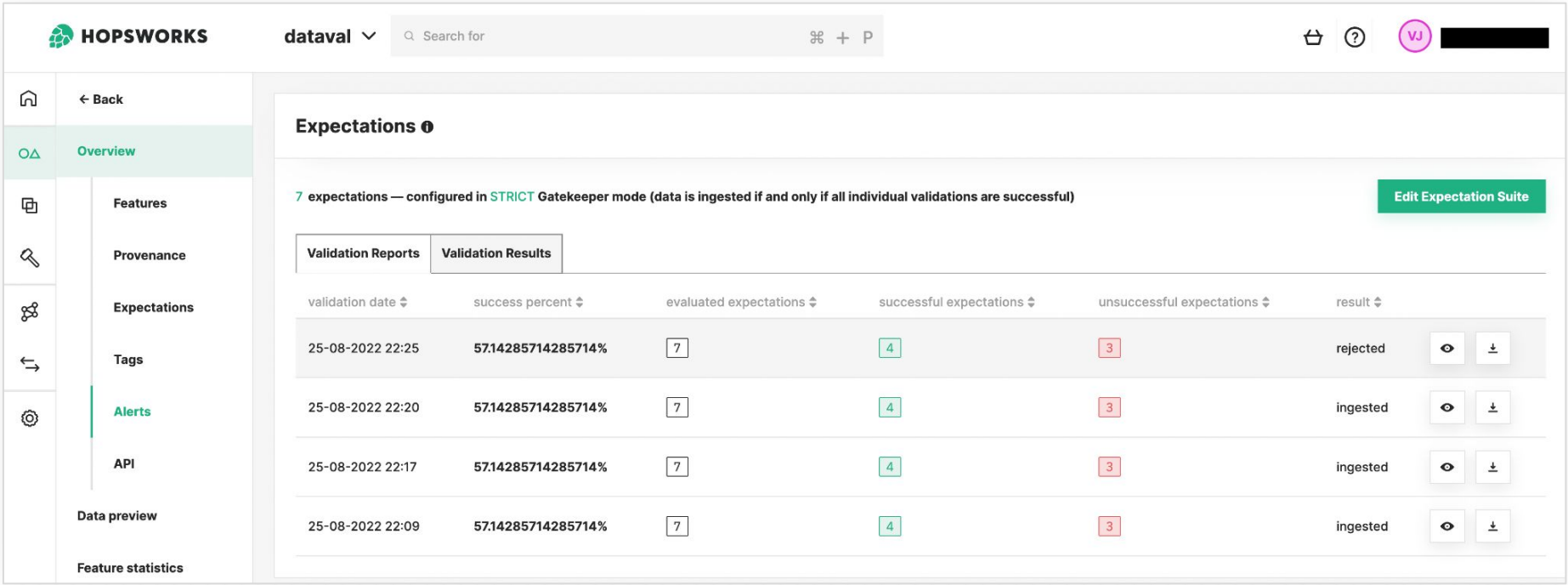
- **ML-enabled products evolve over time**
  - The available input data (features) change over time
  - The target you are trying to predict changes over time
- **Automate the testing and deployment** of ML-enabled Products
  - Safe incremental updates and a tighter iteration loop
- To this end, **features and models must be tested**
  - Tests should run automatically as part of a CI/CD workflow

Data Validation with Great Expectations



https://www.hopsworks.ai/post/data-validation-for-enterprise-ai-using-great-expectations-with-hopsworks

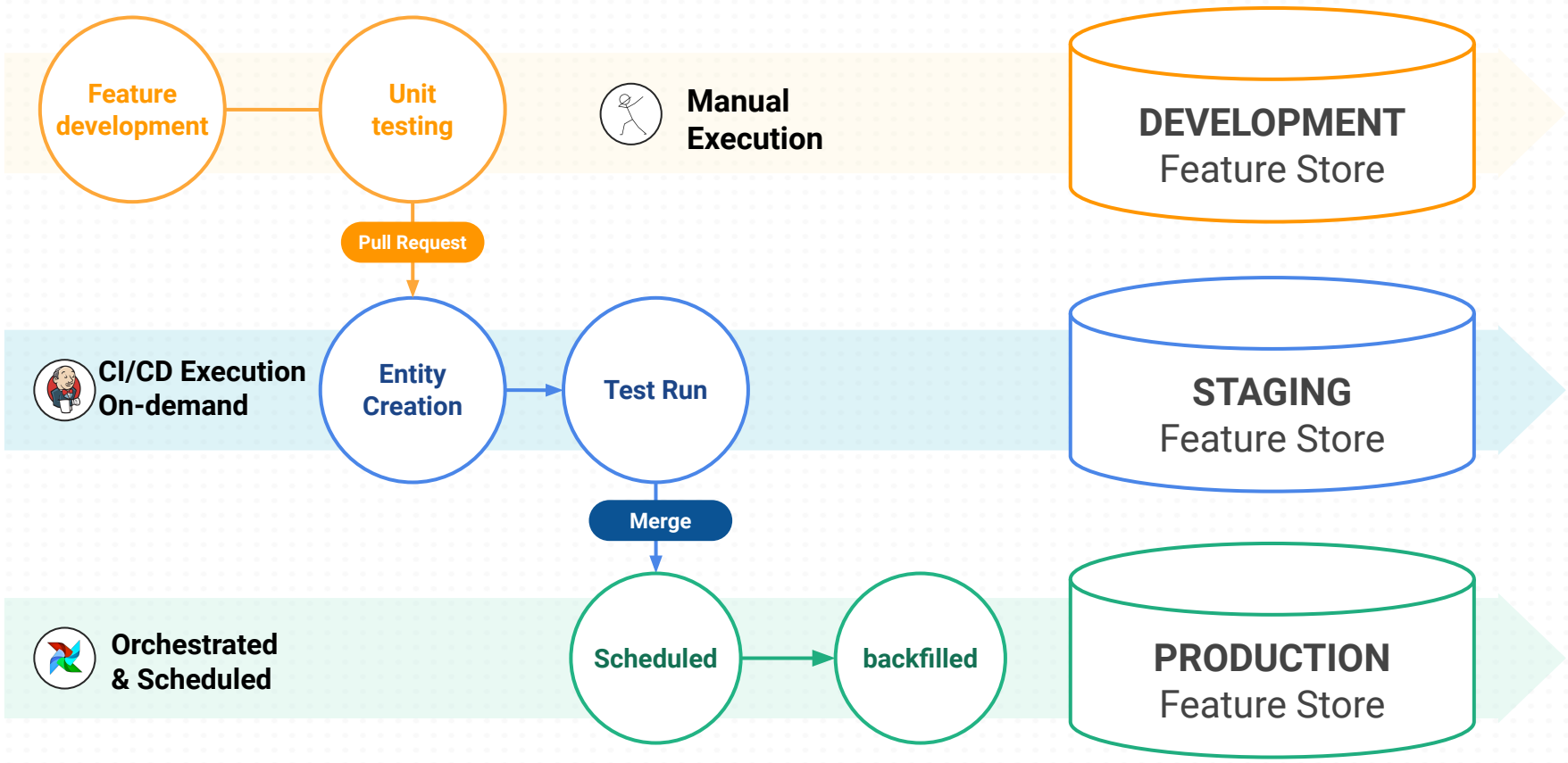
Great Expectations in Hopsworks



https://docs.hopsworks.ai/3.1/user_guides/fs/feature_group/data_validation/

HOPSWORKS

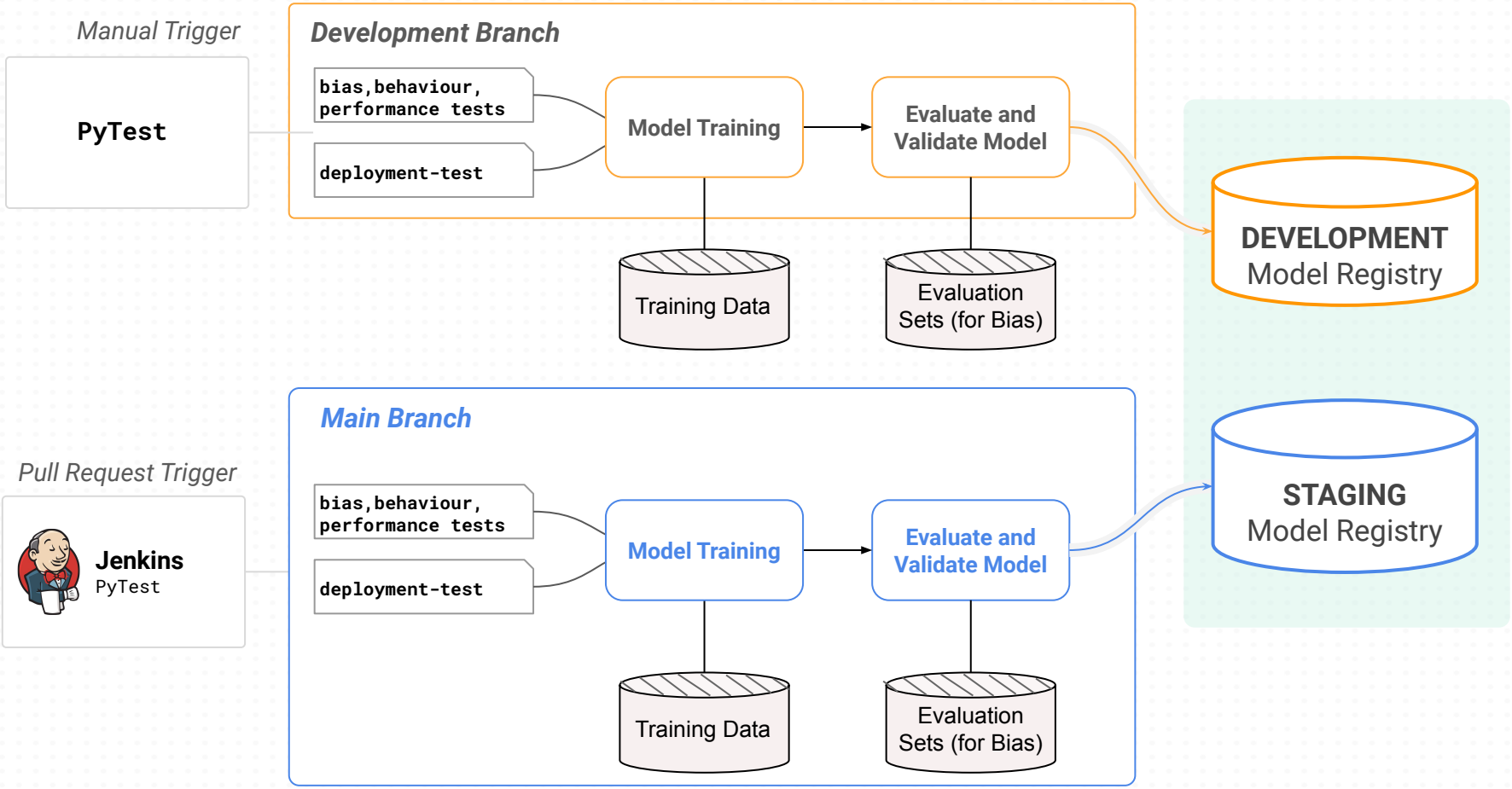
CI/CD for Feature Pipelines



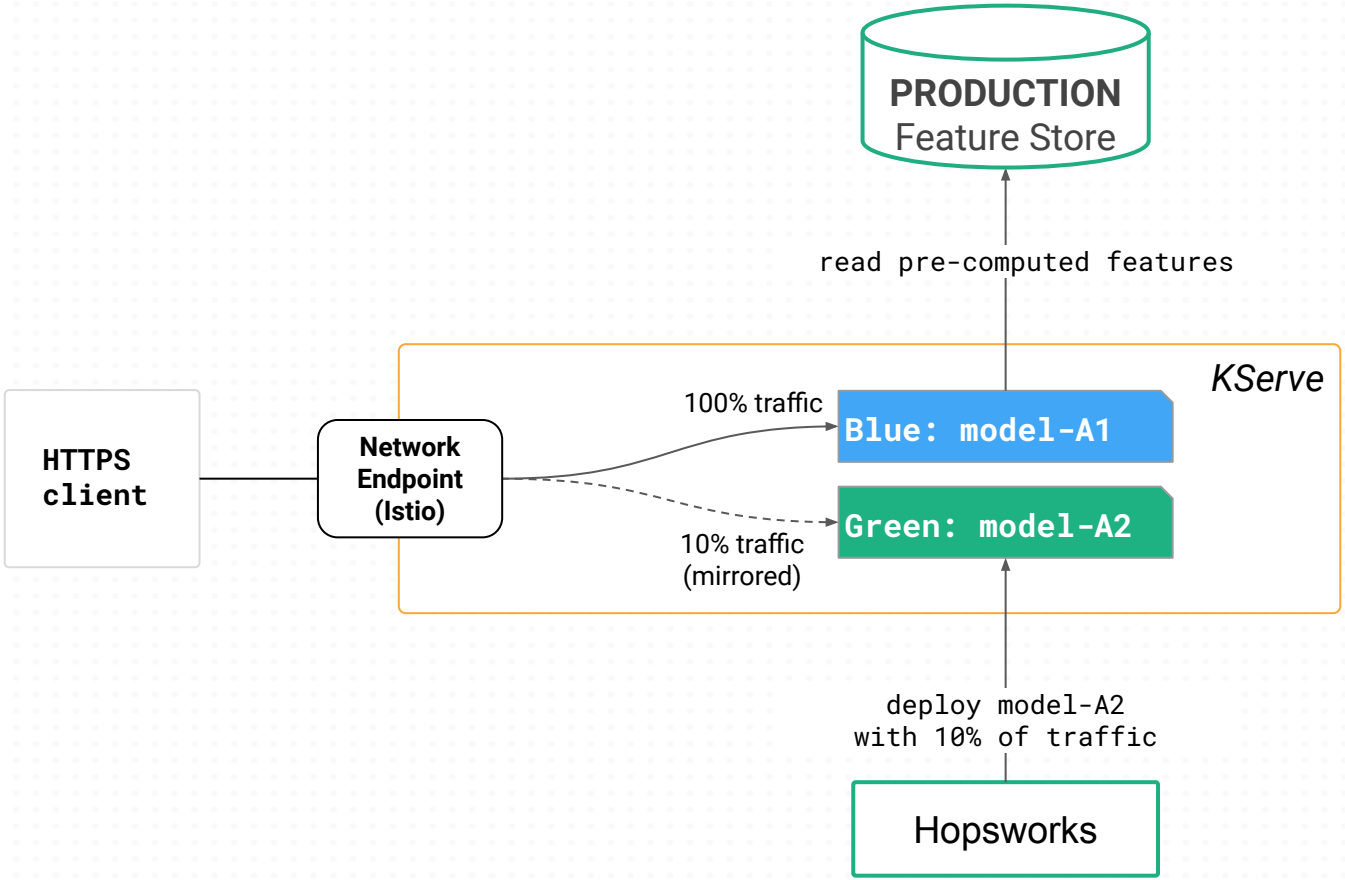
https://www.hopsworks.ai/post/optimize-your-mlops-workflow-with-a-feature-store-ci-cd-and-github-actions

HOPSWORKS

CI/CD for Model Training

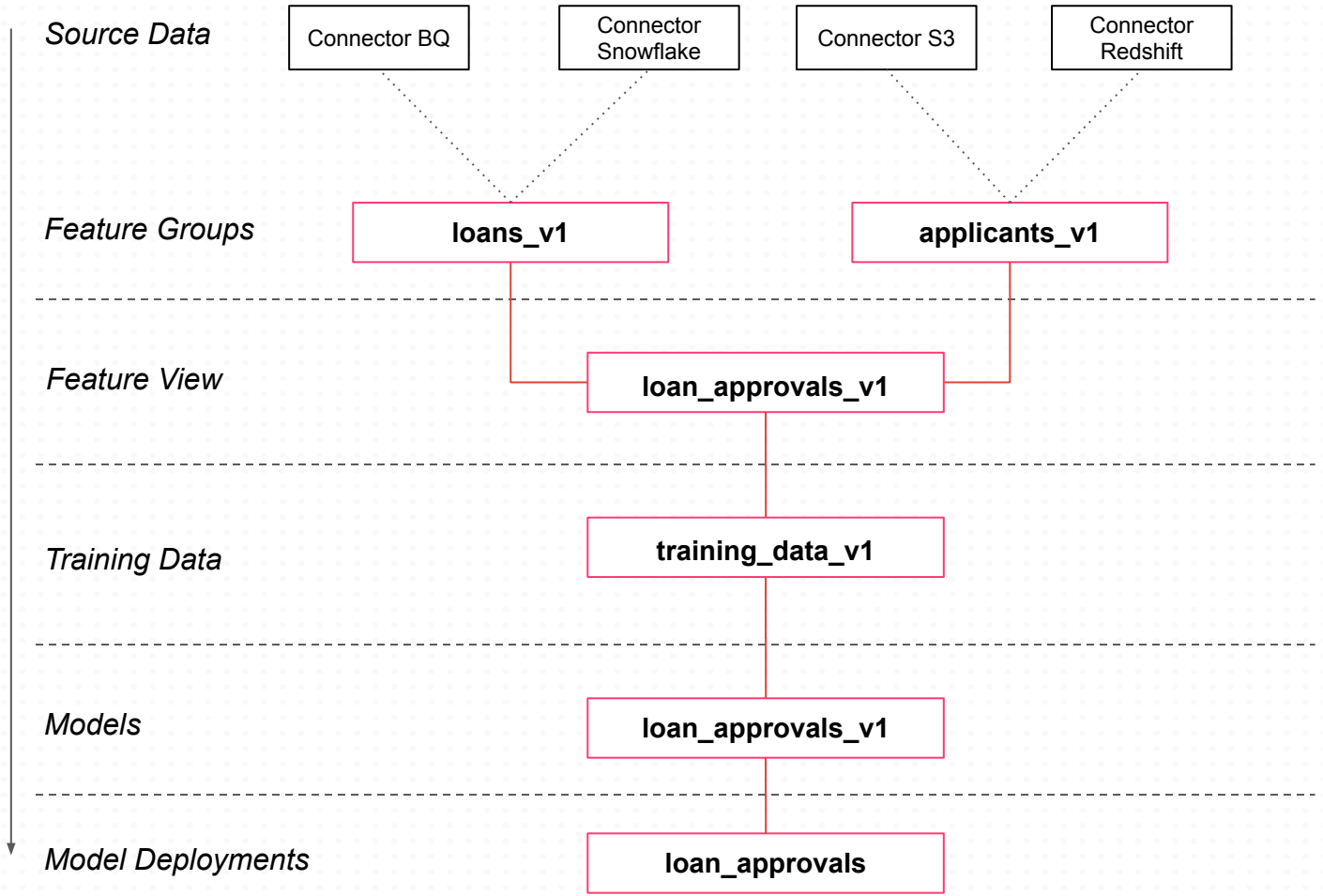


CI/CD for Deployed Models



HOPSWORKS

Lineage for Features and Models



HOPSWORKS

Schema Versioning

- In Hopsworks, you can make non-breaking schema changes that do not require updating the schema version.
- Appending features with a default value is a non-breaking schema change

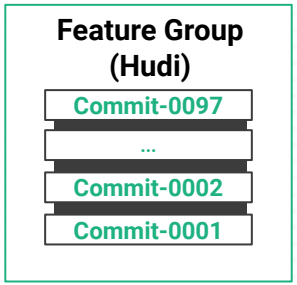
```
from hsfs.feature import Feature

features = [
    Feature(name="id",type="int",online_type="int"),
    Feature(name="name",type="string",online_type="varchar(20)")]

fg = fs.get_feature_group(name="example", version=1)
fg.append_features(features)
```

- Breaking schema changes require updating the schema version for a Feature Group.

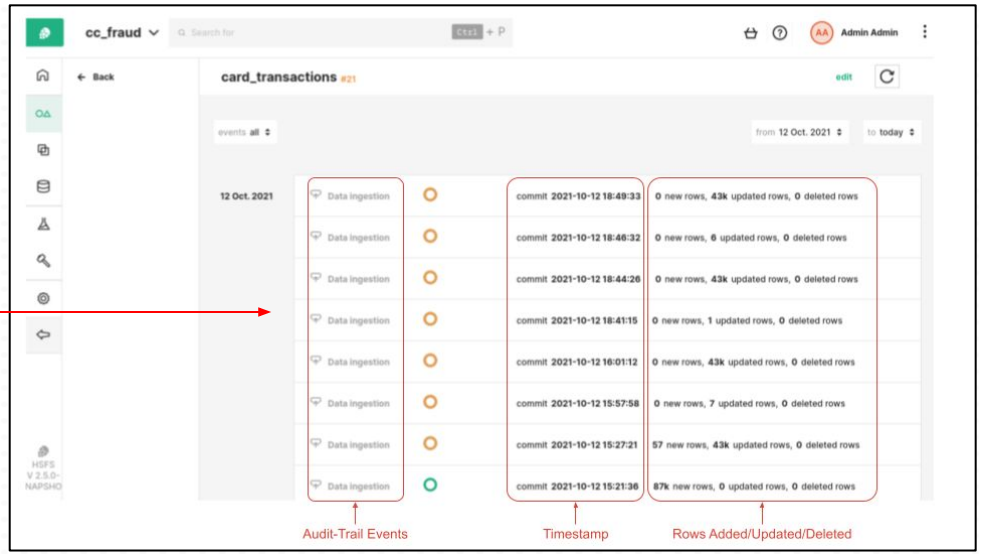
```
fg1 = fs.create_feature_group(name=”example”, version=1)
df = fg1.read()
fg2 = fs.create_feature_group(name=”example”, version=2, features=new_features, …)
fg2.insert(df) #backfill the new feature group with data from the prev version
```

Time-Travel for Feature Groups

Feature Pipeline writes DataFrames

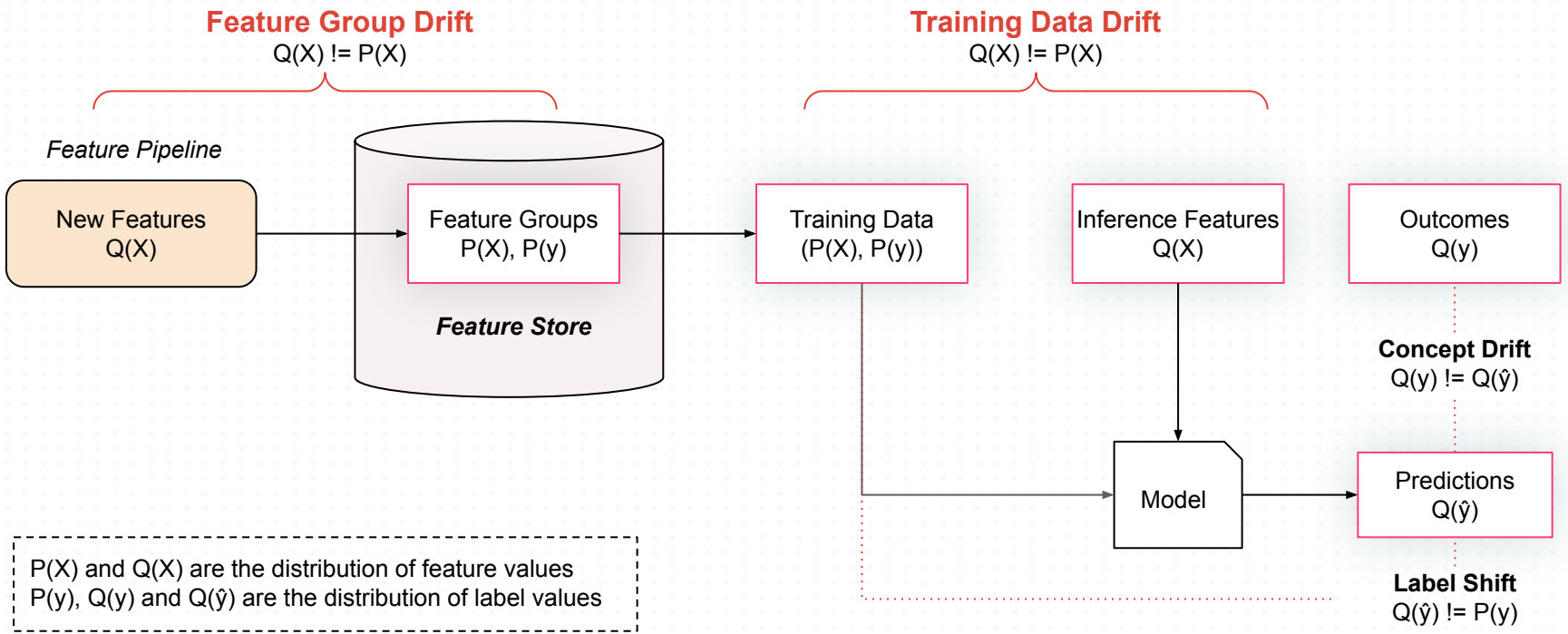


View Commits in the Hopsworks UI in the “Activity” tab of a Feature Group



HOPSWORKS

Feature Monitoring & Alerting (coming in Q2)



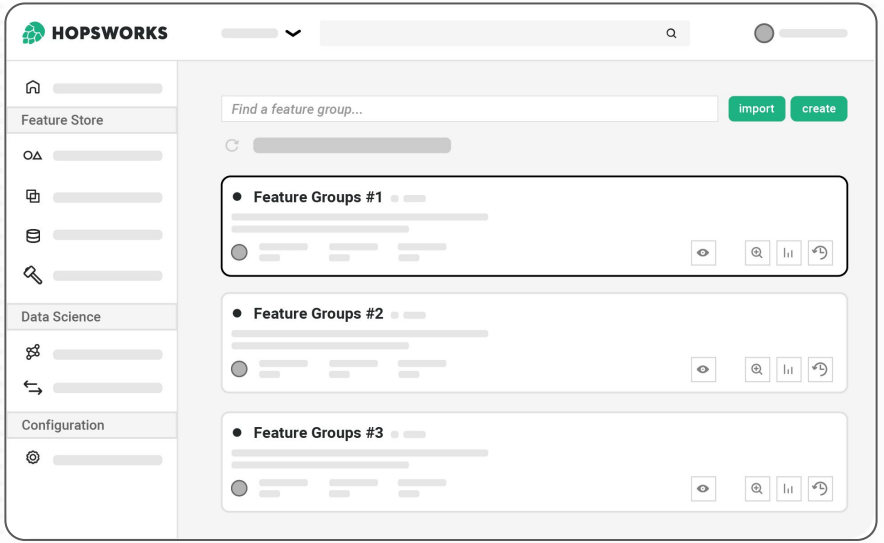
HOPSWORKS // Ready to start serverless or in the cloud

Efficient API, Python API for data scientists

```python
# connect to Hopsworks

import hopsworks

project = hopsworks.login()
fs = project.get_feature_store()
```

```python
# create a feature group

transactions_dataframe = ...

transaction_fg = fs.get_or_create_feature_group(
    name="Transactions_30min_aggregates",
    version="1",
    description="Transaction fatures 30 mins window",
    primary_key=['cc_id'],
    event_time='datetime'
)

transaction_fg.insert(transactions_dataframe)
```

Clear UI for exploration and productivity



Access on app.hopsworks.ai