



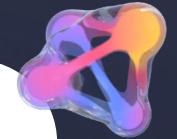
# Iguazio and MLOps

---

Nick Schenone – DATASCI 290

# Agenda

---



- What is MLOps?
- Challenges in an ML System
- Requirements for an ML System
- Design Patterns + Iguazio Implementation

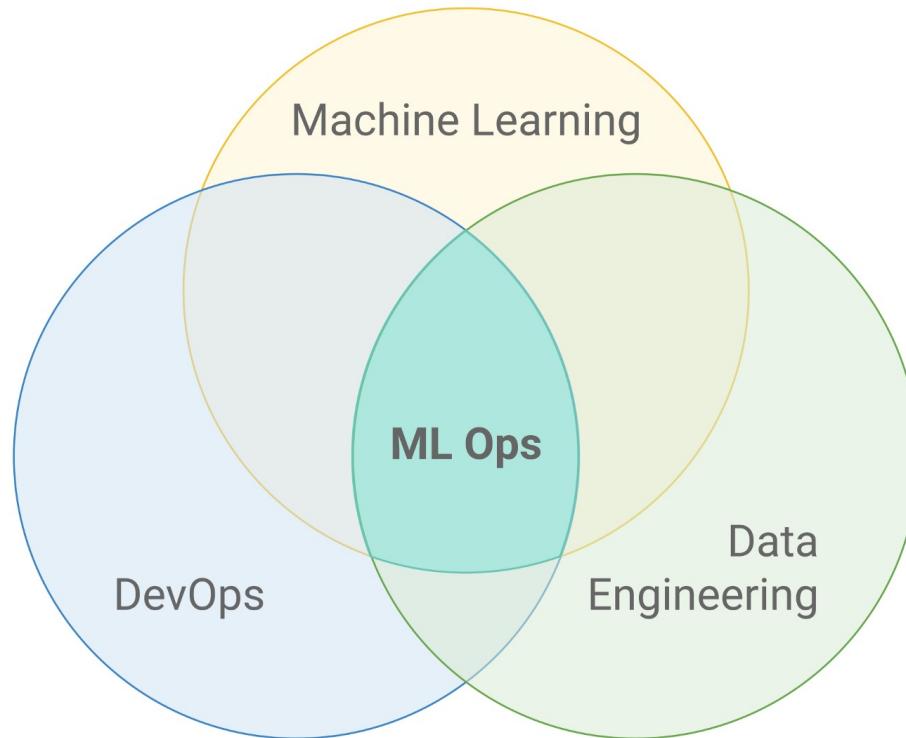
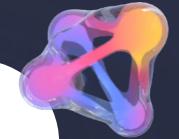




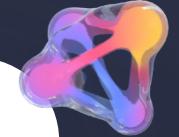
# What is MLOps?

---

Machine Learning Operations



"ML Ops: Machine Learning as an Engineering Discipline".



Set of practices that aims to deploy and maintain machine learning models in production reliably and efficiently



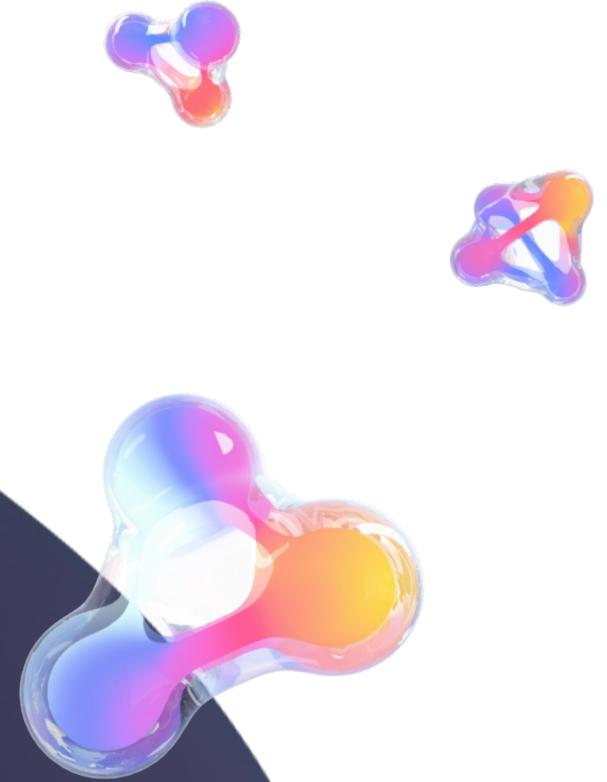
"ML Ops: Machine Learning as an Engineering Discipline".



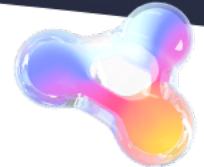
# Challenges in an ML System

---

## Technical Challenges



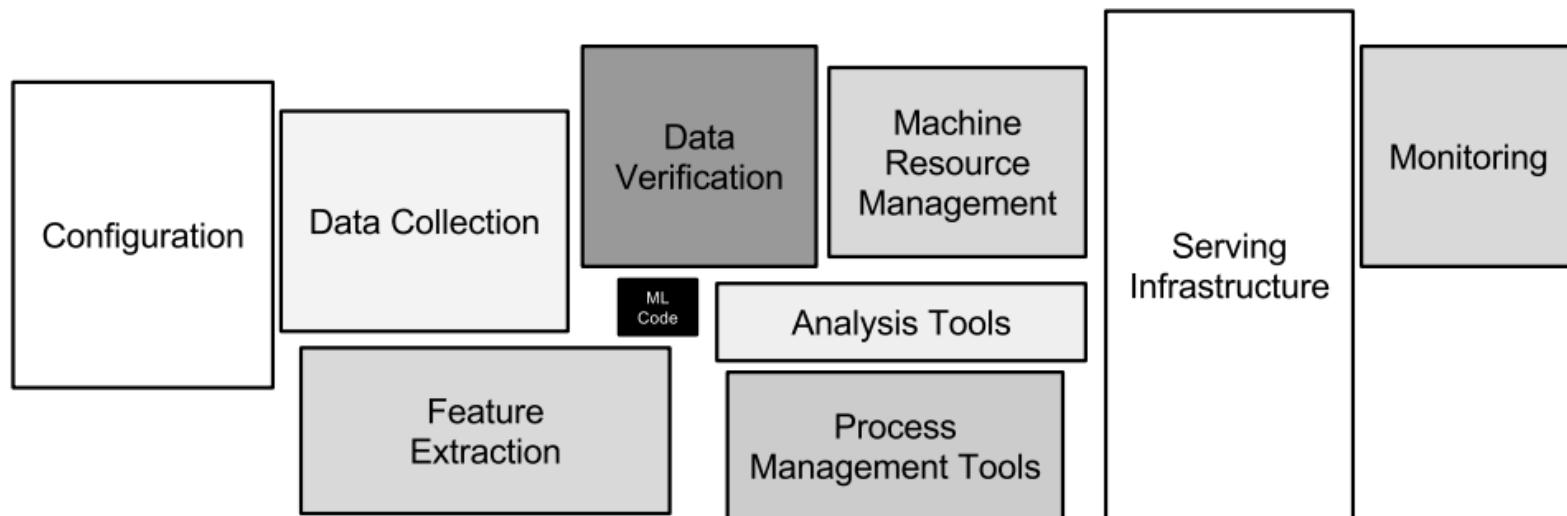
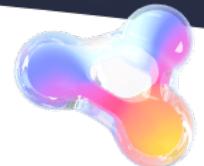
# ML Systems are More than Code



- Data
- Feature Engineering
- Training/Serving Code
- Models
- Metrics/Hyperparameters
- Pipelines
- Docker Images
- Logs

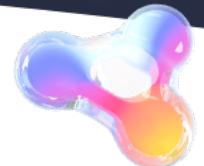


# ML at Scale is Hard



Hidden Technical Debt in Machine Learning Systems (2015)

# Research ≠ Production



	<b>Research</b>	<b>Production</b>
<b>Model Architecture</b>	Cutting edge	Tried and true
<b>Main Concern</b>	Performance (accuracy, f1, etc.)	Latency, performance, bias, explainability, etc.
<b>Data</b>	Benchmark or Synthesized	Real data (requires cleaning)
<b>Changes in Data</b>	Static	Constantly evolving
<b>Prediction Type</b>	Batch	Real-Time + Batch

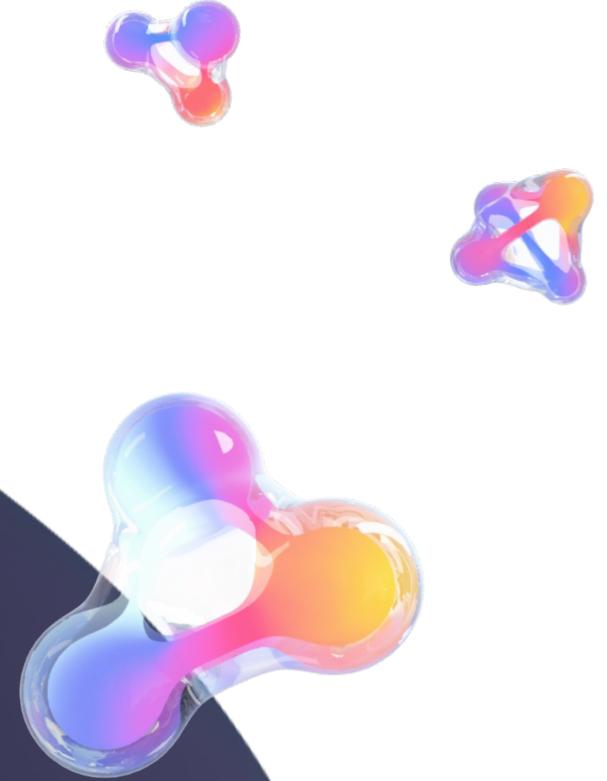


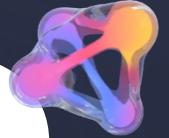


# Challenges in an ML System

---

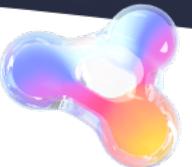
Business Challenges





1. *Your ML System exists to drive business value*
2. *ML and MLOps is an investment*

# Productionizing an ML Model Takes a Lot of Time and Effort

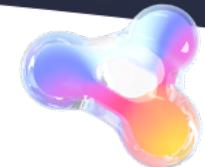
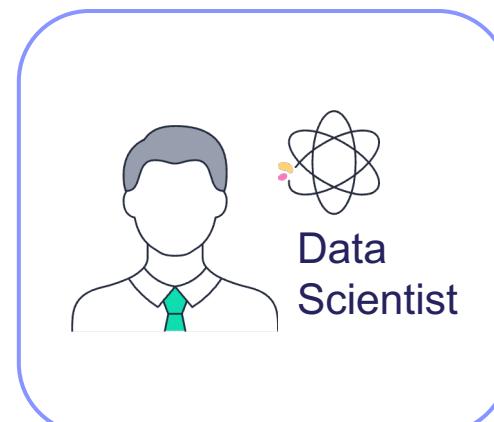
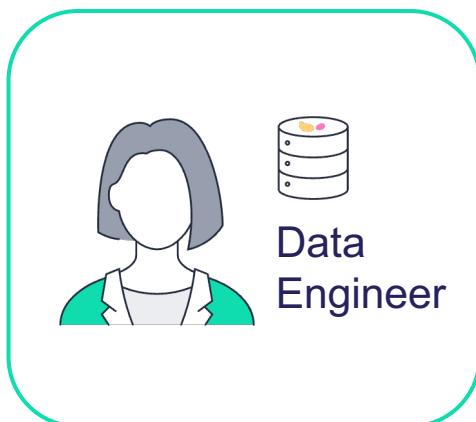


## DevOps

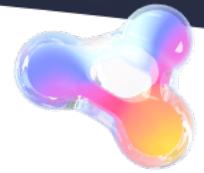
1. Write code + local testing
2. Build code and Docker image
3. CI/CD Pipeline
4. Add logging and monitoring
5. Harden security
6. Provision servers + OS
7. Handle data/event feed
8. Handle failures/auto-scaling
9. Handle rolling upgrades
10. Configuration management



# Siloed Work is Not Efficient



# Auditing and Compliance



Reproducibility

Explainability

Bias





# Requirements for an ML System

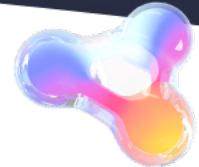
---

Addressing Technical and Business Challenges



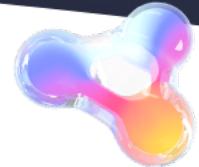
# ML System Should Be

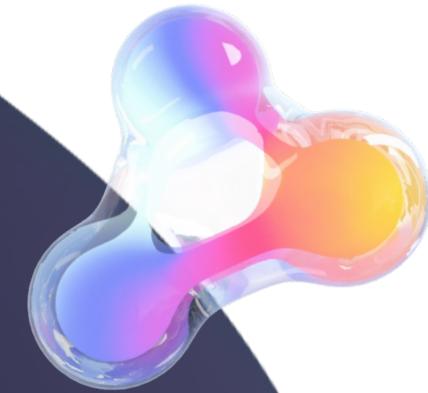
- **Faster, easier to use, more reliable and more efficient than doing things by hand**
- Able to keep track of all the pieces (data, code, etc.)
- Able to run at scale
- Able to facilitate research and production needs
- Reproducible, explainable, and (relatively) bias-free



# ML System Should Be

- Automated end to end – I should be able to train and serve a model with a push of a button

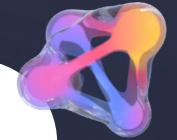




# Design Patterns

---

+ Iguazio Implementations

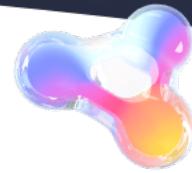


- Abstractions to Containerize Code/Model
- CI/CD and Git as Source of Truth
- Feature Store as a Hub for ML\*
- Monitoring Drift + Automated Re-Training\*



# Abstractions to Containerize Code/Model

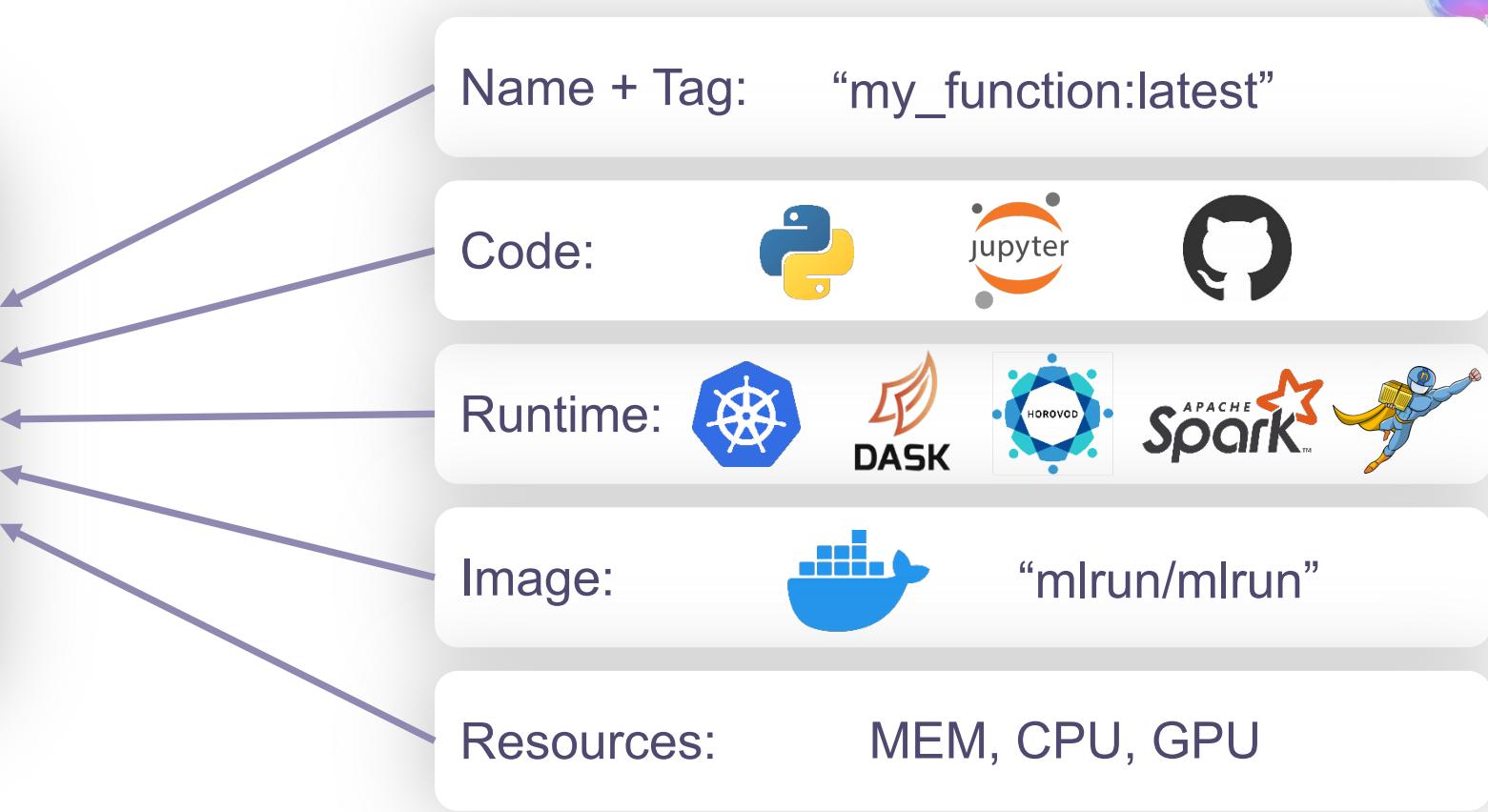
- Spend more time writing code than worrying about infrastructure
- Abstract underlying implementation from the interface
- Allows DS to easily containerize pipeline steps and deployments with high level syntax – WITHOUT in-depth Docker/K8s knowledge
- Kick off jobs/pipelines/deployments from Jupyter notebook or laptop



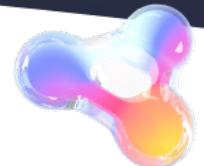
# Turn Code to Serverless Functions



MLRun Function



# Turn Code to Serverless Functions

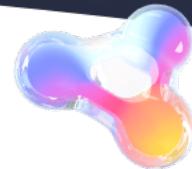


MLRun Function



```
fn = code_to_function(  
    name="test",  
    project="my_project",  
    filename="test.py", # .py or .ipynb  
    kind="job",  
    image="mlrun/mlrun",  
    handler="my_func"  
)  
fn.run()
```

# Advanced Configuration



## Assign Resources

```
fn.with_requests(mem="1G", cpu=1)  
fn.with_limits(mem="2G", cpu=2, gpus=1)
```

## Specify Scaling Behavior

```
fn.spec.replicas = 2  
  
fn.spec.min_replicas = 1  
fn.spec.min_replicas = 4
```



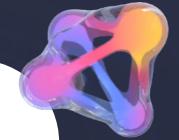
24

## Mount Persistent Storage

```
fn.apply(  
    mlrun.platforms.mount_pvc(  
        pvc_name="data-claim",  
        volume_name="data",  
        volume_mount_path="/data"  
    )  
)
```

## Pod Priority and Node Selection

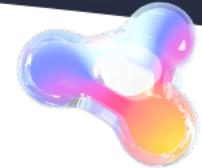
```
fn.with_priority_class(name="igz-workload-medium")  
fn.with_node_selection(  
    node_selector={  
        "app.iguazio.com/lifecycle": "non-preemptible"  
    }  
)
```



# Demo



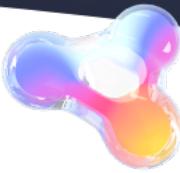
# CI/CD and Git as Source of Truth



- Keep all code and pipelines in one place
- Pairs well with Infrastructure as Code (IaC)
- Restrict who is deploying to dev, staging, prod environments
- Keep a human in the loop to approve model (re)deployments



# MLRun Project - IaC



project.yaml    X    /User    X

```
1 kind: project
2 metadata:
3   name: heart-disease-classifier
4   created: '2022-07-05T16:12:14.068000+00:00'
5 spec:
6   functions:
7     - url: components/get_data.py
8       name: get-data
9       kind: job
10      image: mlrun/mlrun
11    - url: components/train.py
12      name: train
13      kind: job
14      image: mlrun/mlrun
15    - url: hub://v2_model_server
16      name: serving
17    - url: components/simulate_traffic.py
18      name: simulate-traffic
19      kind: nuclio
20      image: mlrun/mlrun
21 workflows:
22   - name: main
23     path: pipelines/training_pipeline.py
24     engine: null
25   artifacts: []
26   source: ''
27   desired_state: online
28   owner: nick
29 status:
30   state: online
31
```

## Run with Python

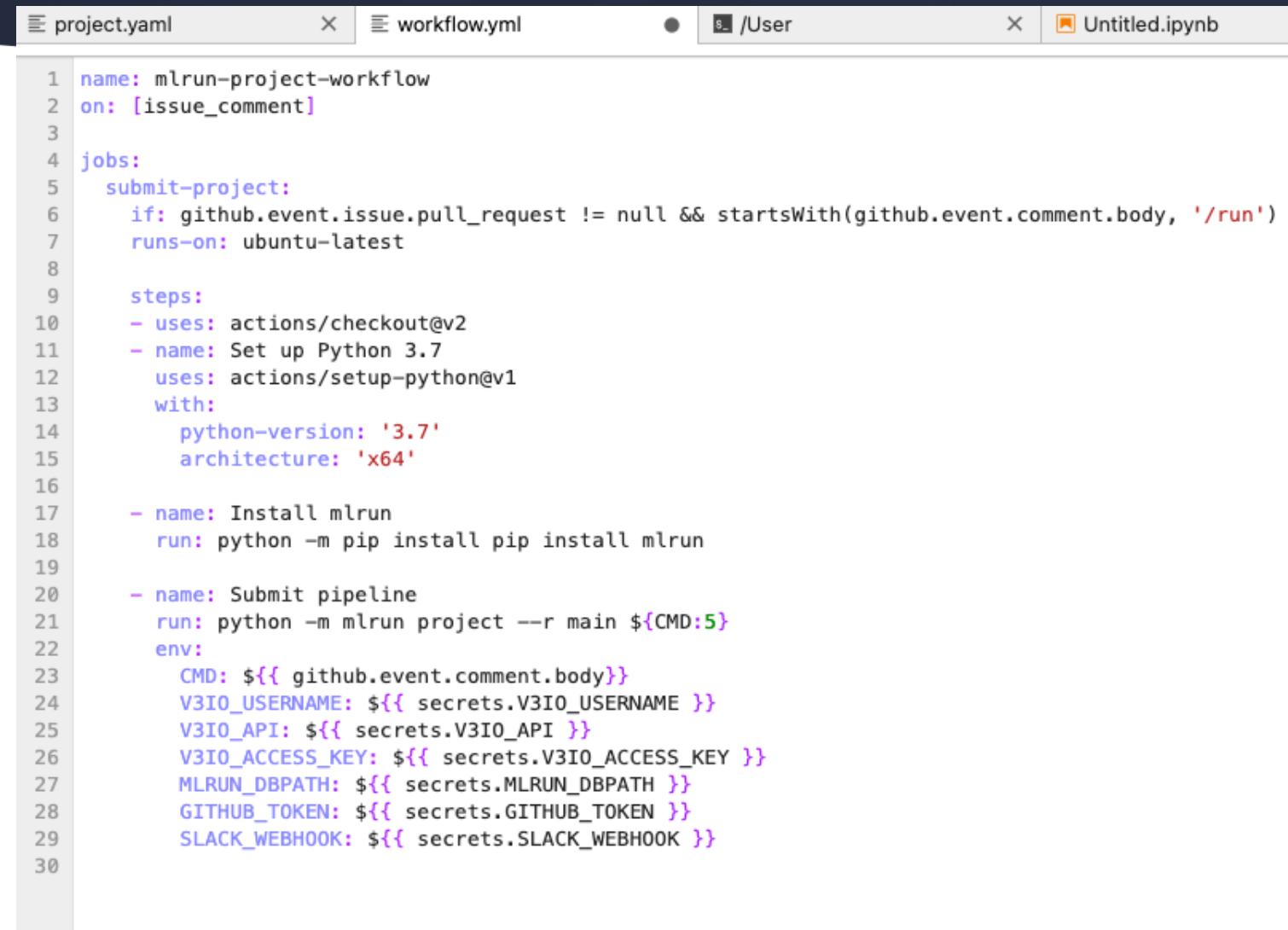
```
run_id = project.run(
    name="main",
    arguments={
        "source_url" : "store://feature-vectors/heart-disease-classifier/heart-disease-vec:latest",
        "label_column" : "target"
    }
)
```

## Run with CLI

```
mlrun project --run main \
  --arguments source_url=store://feature-vectors/heart-disease-classifier/heart-disease-vec:latest \
  --arguments label_column=target
```

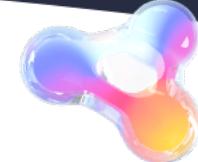


# CI/CD Example – GitHub Actions

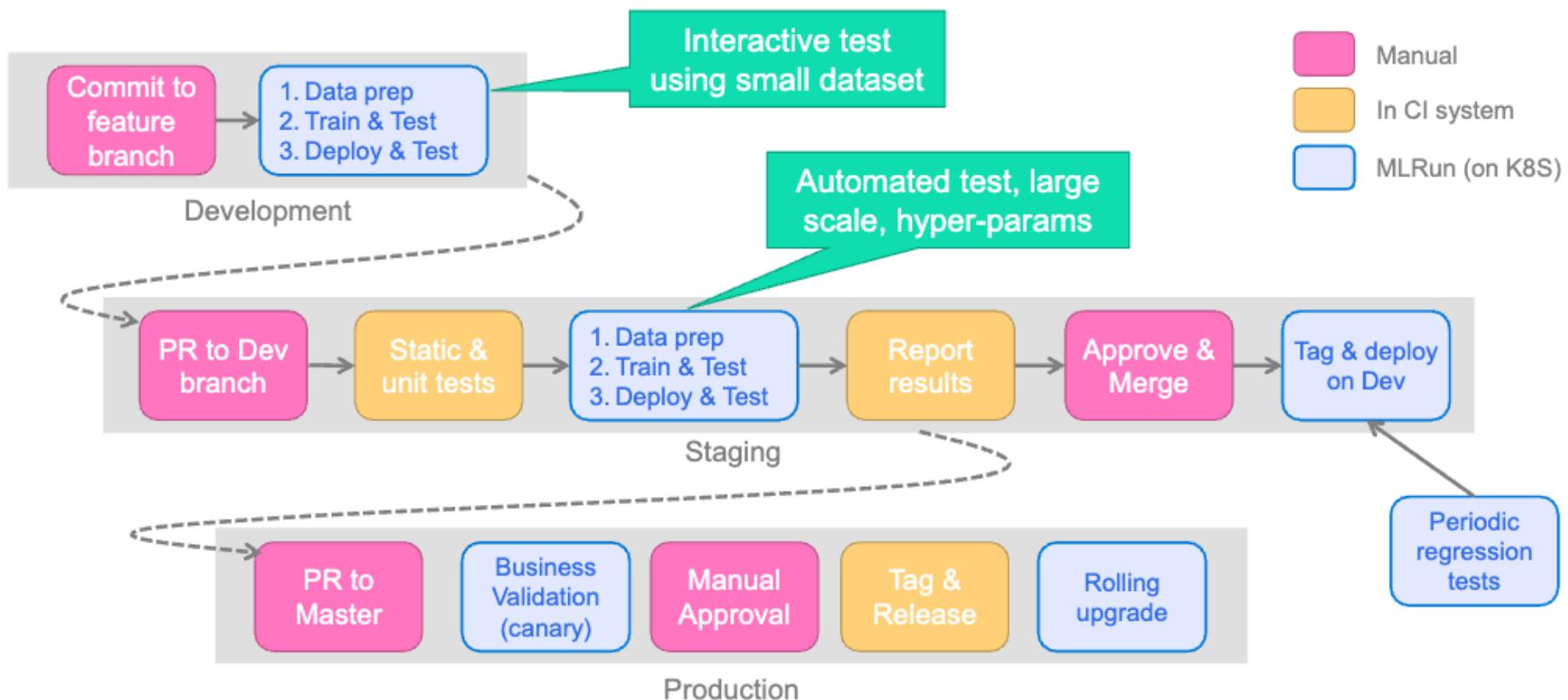


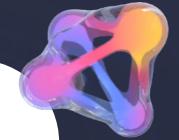
The image shows a screenshot of a GitHub Actions interface. At the top, there are tabs for "project.yaml", "workflow.yml", "/User", and "Untitled.ipynb". The "workflow.yml" tab is active. The code in the workflow file is as follows:

```
1 name: mlrun-project-workflow
2 on: [issue_comment]
3
4 jobs:
5   submit-project:
6     if: github.event.issue.pull_request != null && startsWith(github.event.comment.body, '/run')
7     runs-on: ubuntu-latest
8
9   steps:
10    - uses: actions/checkout@v2
11    - name: Set up Python 3.7
12      uses: actions/setup-python@v1
13      with:
14        python-version: '3.7'
15        architecture: 'x64'
16
17    - name: Install mlrun
18      run: python -m pip install pip install mlrun
19
20    - name: Submit pipeline
21      run: python -m mlrun project --r main ${CMD:5}
22      env:
23        CMD: ${{ github.event.comment.body}}
24        V3IO_USERNAME: ${{ secrets.V3IO_USERNAME }}
25        V3IO_API: ${{ secrets.V3IO_API }}
26        V3IO_ACCESS_KEY: ${{ secrets.V3IO_ACCESS_KEY }}
27        MLRUN_DBPATH: ${{ secrets.MLRUN_DBPATH }}
28        GITHUB_TOKEN: ${{ secrets.GITHUB_TOKEN }}
29        SLACK_WEBHOOK: ${{ secrets.SLACK_WEBHOOK }}
```



# Dev -> Staging -> Prod

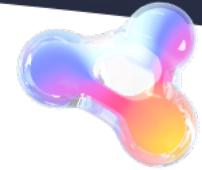




# Demo



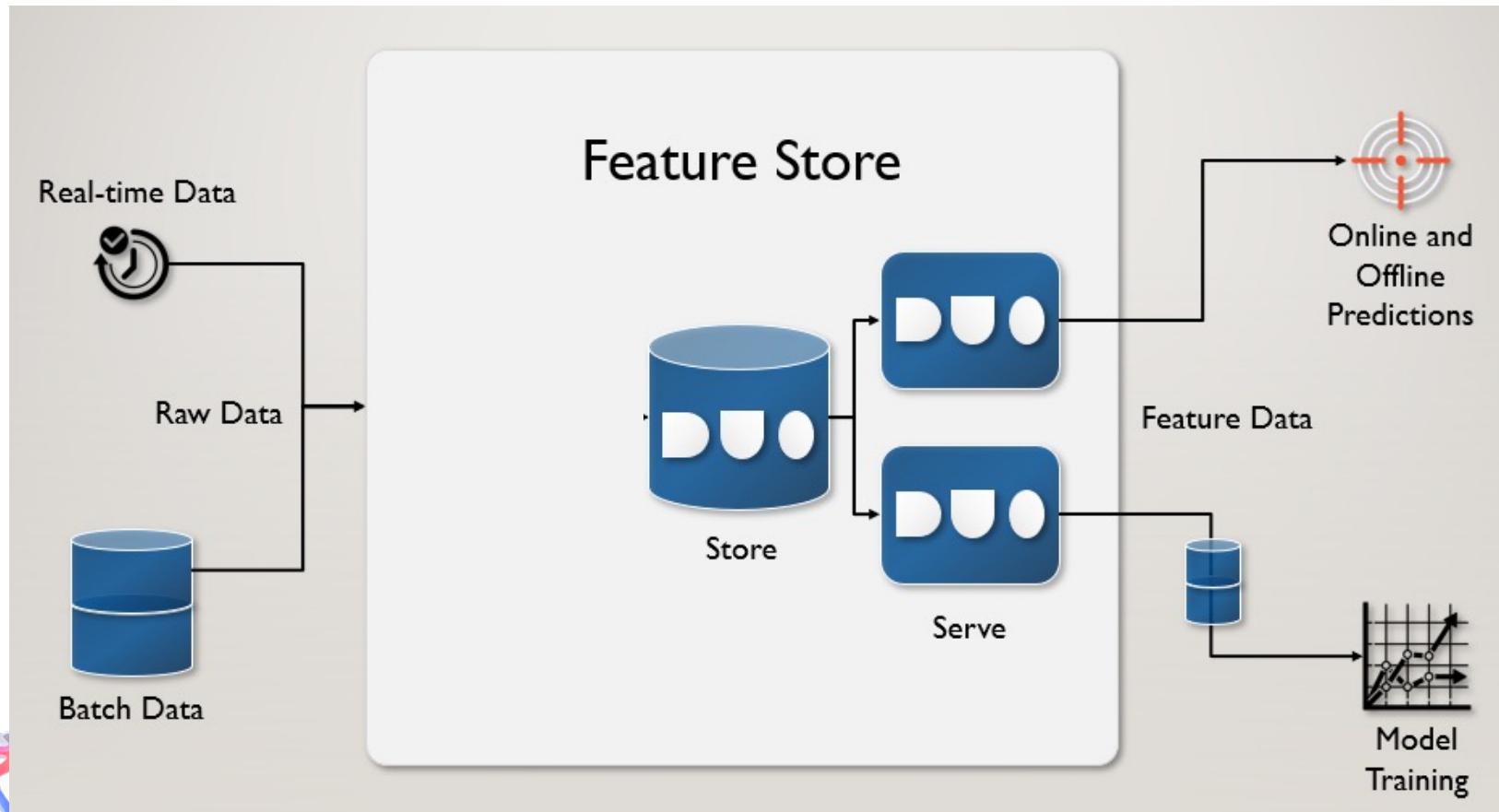
# Feature Store as a Hub for ML



- Feature store is a gift from one developer to another
- Allows users to re-use existing features from a central location for use in other models/projects/pipelines/teams
- Ingest, transform, join multiple data sources in batch and real-time
- Re-use transformations to mitigate train/serve skew
- Excellent for larger organizations with many models, may not be necessary for others



# Feature Store Architecture

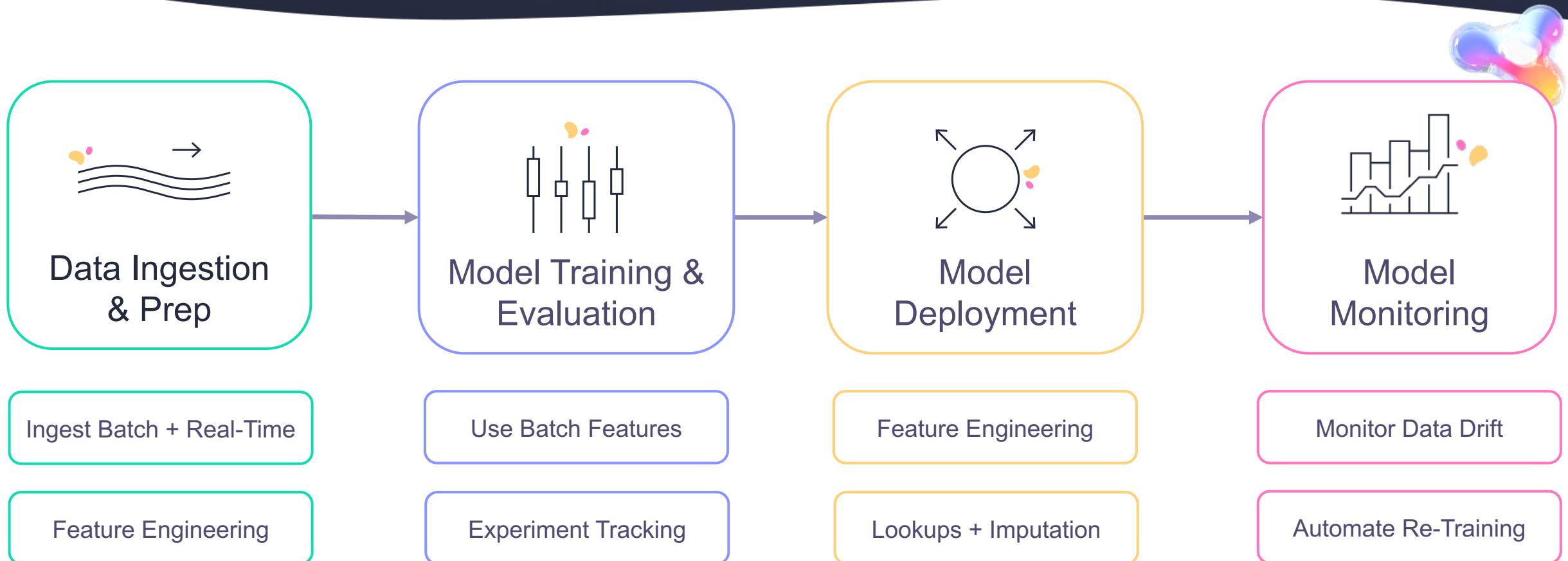


Experiment Tracking

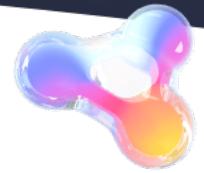
Model Serving

Model Monitoring

# Feature Store in Overall ML Workflow



# Feature Store Ingestion



```
import mlrun.feature_store as fstore

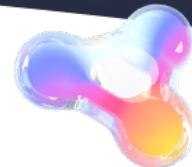
heart_disease_target_set = fstore.FeatureSet(
    name="heart_target",
    entities=[fstore.Entity("PATIENT_ID")],
    description="Heart disease targets via parquet"
)
```

```
resp = fstore.ingest(
    featureset=heart_disease_target_set,
    source=ParquetSource(
        path=f"{data_path}/heart_disease_target_sf.parquet"
    )
)
```

```
resp = fstore.ingest(
    featureset=heart_disease_categorical,
    source=CSVSource(
        path="s3://igz-us-sales/heart/heart_disease_categorical_sf.csv"
    )
)
```

```
fstore.ingest(
    featureset=heart_disease_continuous_set,
    run_config=spark_config,
    source=SnowflakeSource(
        name="snowflake_heart_disease",
        url = os.getenv("SF_URL"),
        user = os.getenv("SF_USER"),
        password = os.getenv("SF_PASSWORD"),
        database = os.getenv("SF_DATABASE"),
        schema = os.getenv("SF_SCHEMA"),
        warehouse = os.getenv("SF_WAREHOUSE"),
        query = "SELECT * FROM MLRUN_DEMO.HEART_DISEASE.CONTINUOUS"
    )
)
```

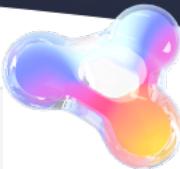
# Feature Store Retrieval



```
vector = fstore.FeatureVector(  
    name="heart-disease-vec",  
    features=[  
        "heart_categorical.AGE_mapped_toddler",  
        "heart_categorical.AGE_mapped_child",  
        "heart_categorical.AGE_mapped_adult",  
        "heart_categorical.AGE_mapped_elder",  
        "heart_continuous.*"  
    ],  
    label_feature="heart_target.TARGET",  
)
```



# Feature Store Retrieval (Training)



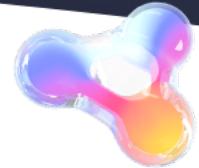
```
df = fstore.get_offline_features("heart-disease-vec").to_dataframe()  
df
```

	AGE_mapped_toddler	AGE_mapped_child	AGE_mapped_adult	AGE_mapped_elder	TRESTBPS	CHOL	RESTECG	THALACH	OLDPEAK	CA	TARGET	
0	0	0	0	1	0	125	212	1	168	1.0	2.0	0
1	0	0	0	1	0	140	203	0	155	3.1	0.0	0
2	0	0	0	0	1	145	174	1	125	2.6	0.0	0
3	0	0	0	1	0	148	203	1	161	0.0	1.0	0
4	0	0	0	1	0	138	294	1	106	1.9	3.0	0
...	...	...	...	...	...	...	...	...	...	...	...	
963	0	0	0	1	0	140	221	1	164	0.0	0.0	1
964	0	0	0	1	0	125	258	0	141	2.8	1.0	0
965	0	0	0	1	0	110	275	0	118	1.0	1.0	0
966	0	0	0	1	0	110	254	0	159	0.0	0.0	1
967	0	0	0	1	0	120	188	1	113	1.4	1.0	0

968 rows x 11 columns



# Feature Store Retrieval (Serving)

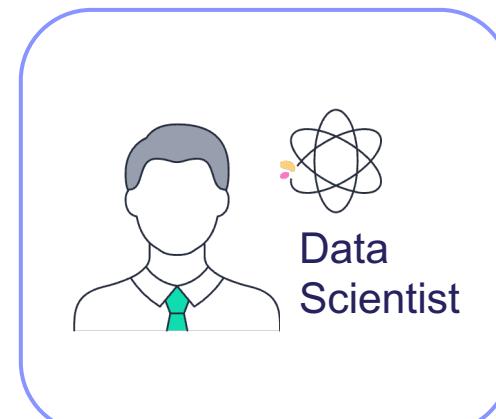
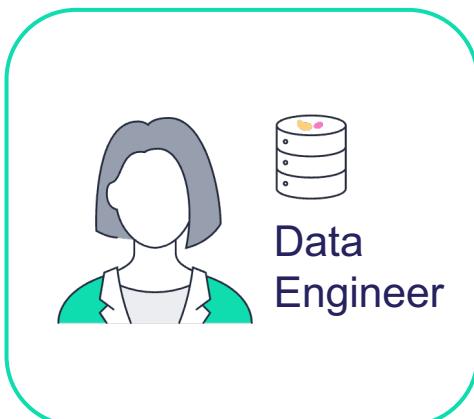


```
feature_service = fstore.get_online_feature_service("heart-disease-vec")
feature_service.get(
    [
        {"PATIENT_ID" : "e443544b-8d9e-4f6c-9623-e24b6139aae0"},
        {"PATIENT_ID" : "8227d3df-16ab-4452-8ea5-99472362d982"}
    ]
)

[{'AGE_mapped_toddler': 0,
 'AGE_mapped_child': 0,
 'AGE_mapped_adult': 1,
 'AGE_mapped_elder': 0,
 'TRESTBPS': 125.0,
 'CHOL': 212.0,
 'RESTECG': 1.0,
 'THALACH': 168.0,
 'OLDPEAK': 1.0,
 'CA': 2.0},
 {'AGE_mapped_toddler': 0,
 'AGE_mapped_child': 0,
 'AGE_mapped_adult': 1,
 'AGE_mapped_elder': 0,
 'TRESTBPS': 140.0,
 'CHOL': 203.0,
 'RESTECG': 0.0,
 'THALACH': 155.0,
 'OLDPEAK': 3.1,
 'CA': 0.0}]
```



# Feature Store in Team Workflows



Unified Interface for Data Ingestion (Batch + RT)

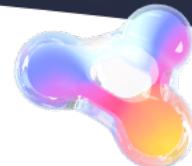
Don't Need to Provide Features

Retrieve Features from Snowflake with Python

Re-Use Features Across Models/Projects/Teams

Don't Need to Build Online Feature Infra

Drift-Detection + Re-Training



# Monitoring Drift + Automated Re-Training

- Models are not forever – it will decay over time
- Various types of drift, but the result is usually reduced model performance
- Automated re-training will keep your model up-to-date using fresh data and mitigate the performance decay over time



# Types of Drift (non-exhaustive)

## Covariate Shift / Data Drift / Virtual Drift

When the data underlying the model changed, but the model's performance did not

$P(X)$  changes but  $P(Y|X)$  remains the same

## Label Shift / Prior Drift

When the model's output or label distribution shifts

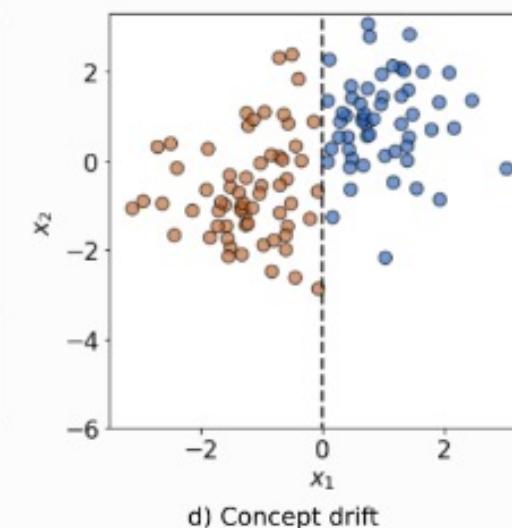
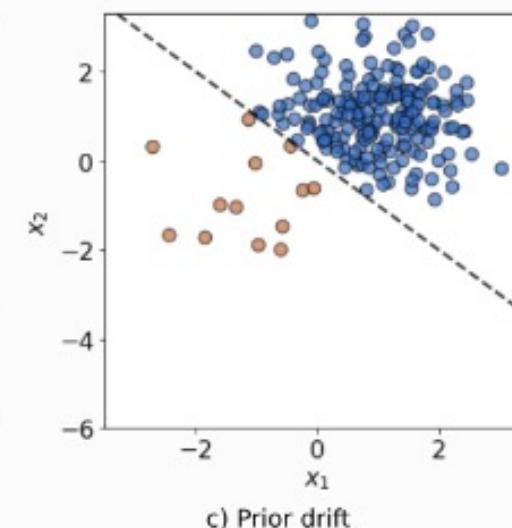
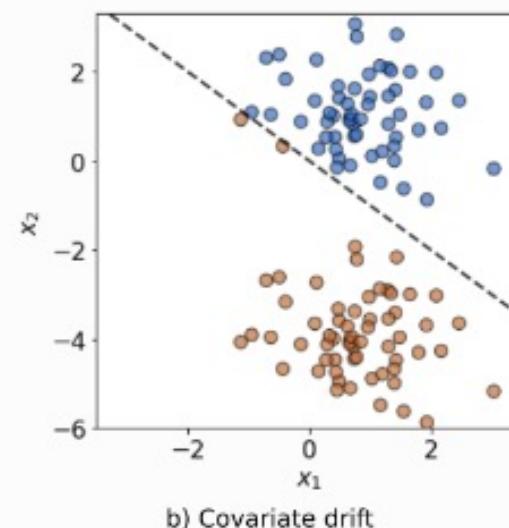
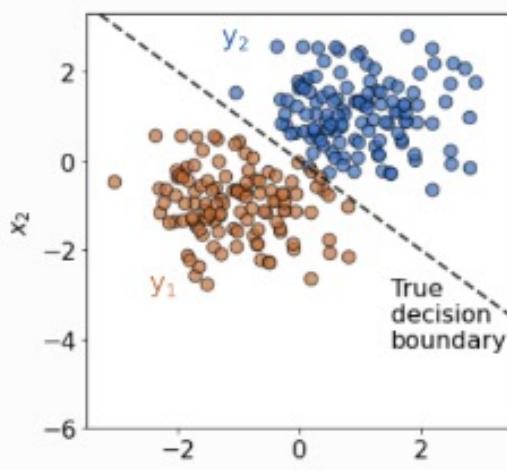
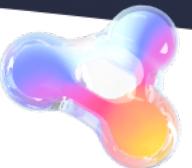
$P(Y)$  changes but  $P(X|Y)$  remains the same

## Concept Drift

When the meaning (statistical properties) of the input data has changed

$P(Y|X)$  changes but  $P(X)$  remains the same

# Types of Drift (non-exhaustive)



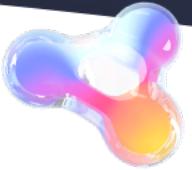
Seldon Alibi – What is Drift?



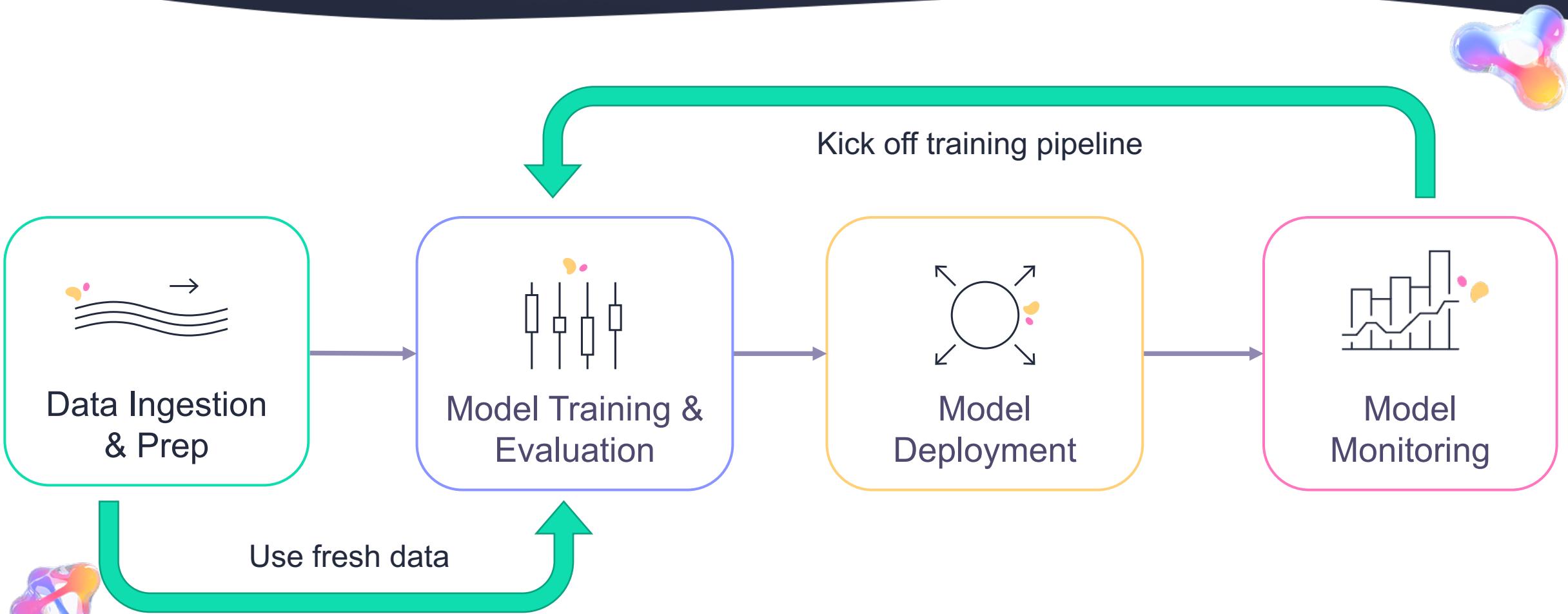
# Types of Drift Detection Algorithms

Detector	Tabular	Image	Time Series	Text	Categorical Features	Online	Feature Level
Kolmogorov-Smirnov	✓	✓		✓	✓		✓
Cramér-von Mises	✓	✓				✓	✓
Fisher's Exact Test	✓				✓	✓	✓
Least-Squares Density Difference	✓	✓		✓	✓	✓	
Maximum Mean Discrepancy (MMD)	✓	✓		✓	✓	✓	
Learned Kernel MMD	✓	✓	✓	✓	✓		
Context-aware MMD	✓	✓	✓	✓	✓		
Chi-Squared	✓				✓		✓
Mixed-type tabular	✓				✓		✓
Classifier	✓	✓	✓	✓	✓		
Spot-the-diff	✓	✓	✓	✓	✓		✓
Classifier Uncertainty	✓	✓	✓	✓	✓		
Regressor Uncertainty	✓	✓	✓	✓	✓		

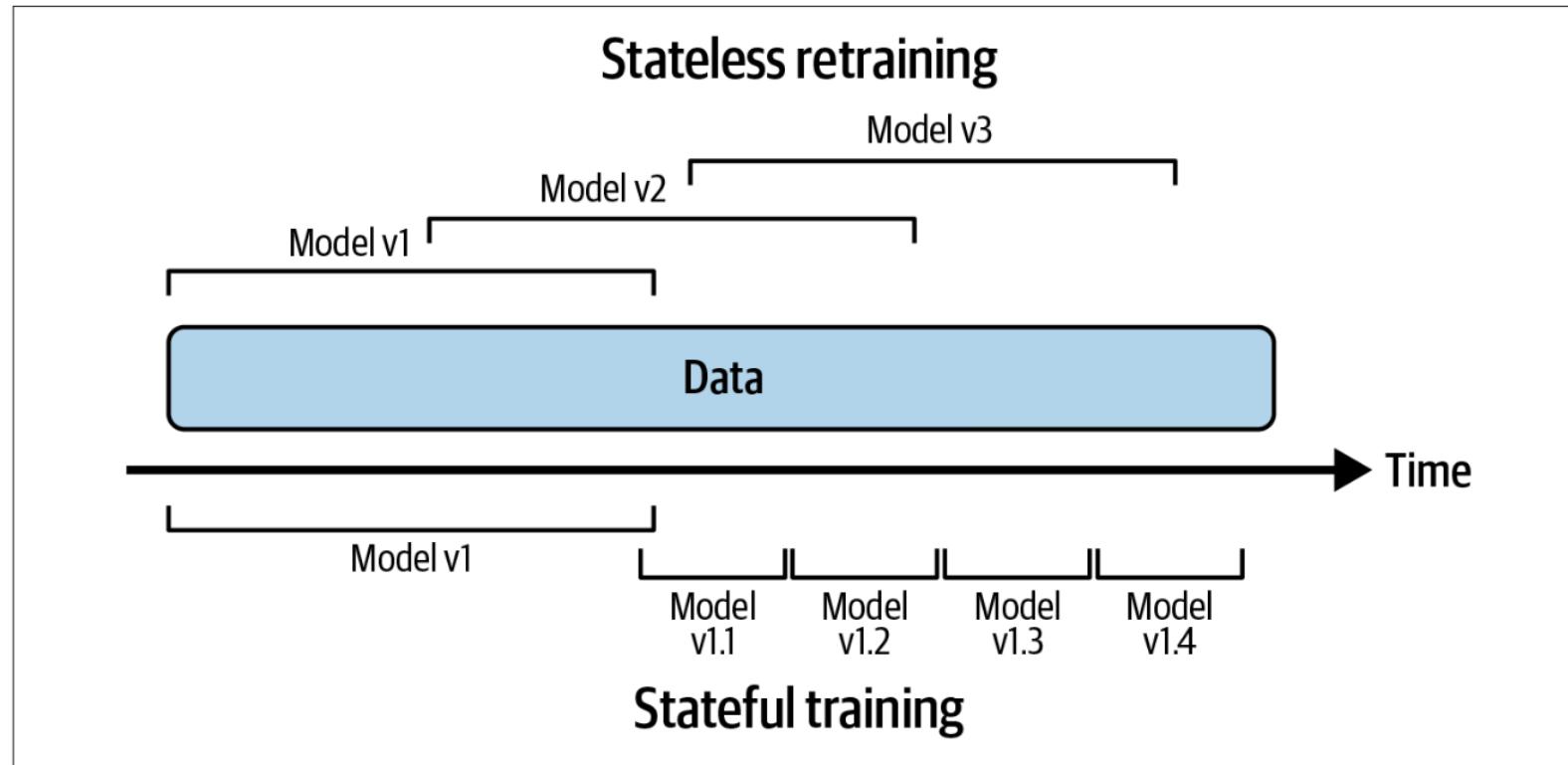
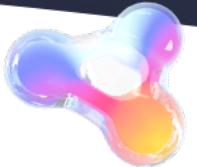
Most drift detection algorithms are for tabular data, however some apply to unstructured data too



# Drift Detected Should Start Re-Training



# Stateful Training vs Stateless Re-Training



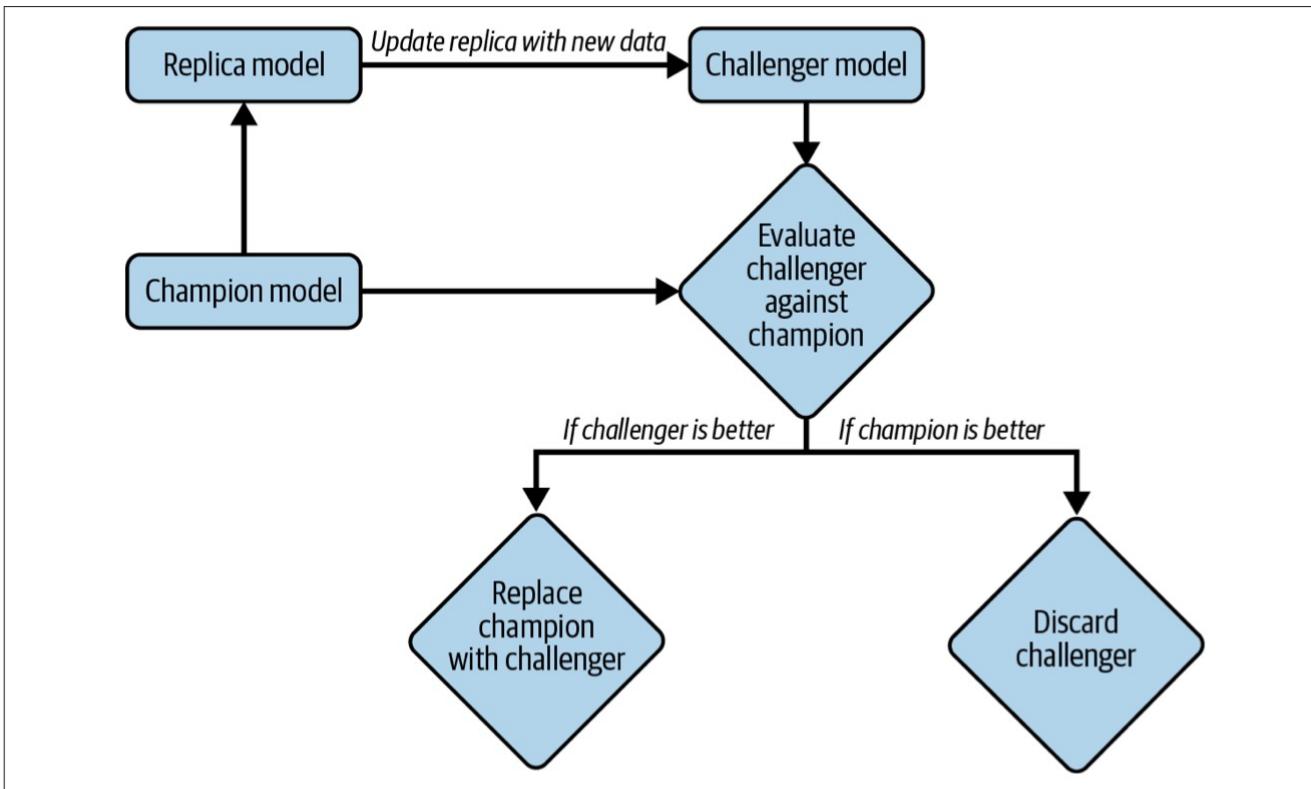
Stateful training allows you to update your model with less data

If you fine-tune your model from yesterday's checkpoint, you only need to use data from the last day

Figure 9-2. Stateless retraining versus stateful training



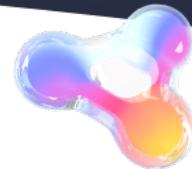
# Model Evaluation/Deployment Strategy



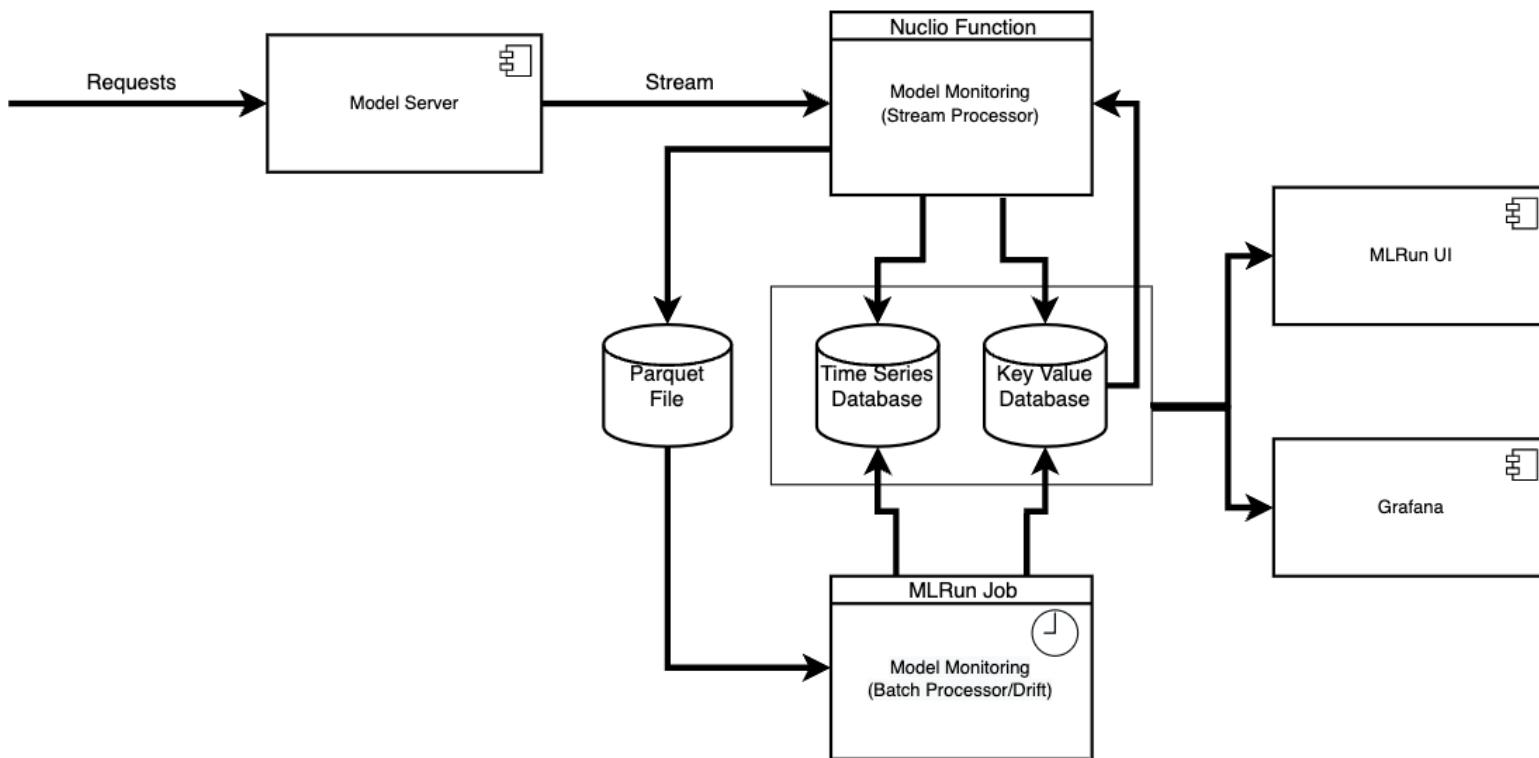
Once you have a new model, compare to the existing one to decide whether to deploy

Can also do more advanced strategies like shadow or canary deployment

Figure 9-1. A simplification of how continual learning might work in production. In reality, the process of handling the failed challenger is a lot more sophisticated than simply discarding it.



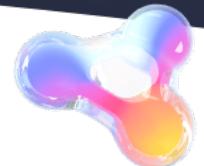
# Drift Detection in Iguazio - Architecture



Backend architecture  
automatically  
deployed with model  
serving service

MLRun - Model monitoring overview

# Drift Detection in Iguazio – Statistical Tests



**Total Variation Distance (TVD)** — The statistical difference between the actual predictions and the model's trained predictions.

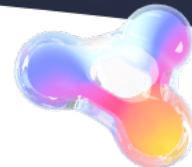
**Hellinger Distance** — A type of f-divergence that quantifies the similarity between the actual predictions, and the model's trained predictions.

**Kullback–Leibler Divergence (KLD)** — The measure of how the probability distribution of actual predictions is different from the second model's trained reference probability distribution.

Drift calculated on hourly basis using these tests



# Drift Detection in Iguazio - Dashboards



Models    Model Endpoints (Beta)

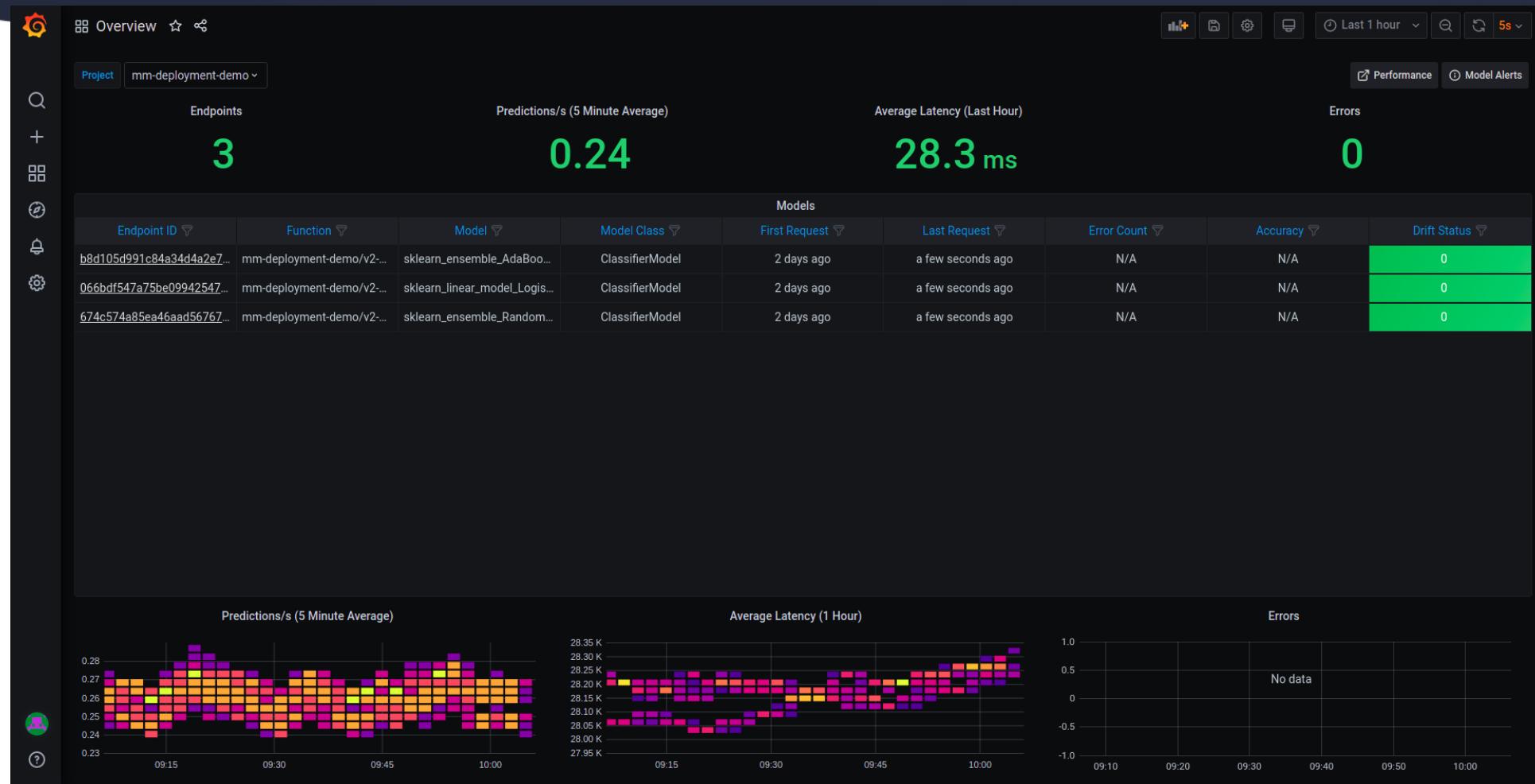
Labels: key1,value1,...

Name

Name	sklearn_ensemble_RandomForestClassifier fb36f09b988eff41ea8bc52422edd2ffe0fce77	⋮	X									
sklearn_ensemble_Ran...												
sklearn_ensemble_Ad...												
sklearn_linear_model_L...	<a href="#">Overview</a> <a href="#">Drift Analysis</a> <a href="#">Features Analysis</a>											
Feature	Expected	Mean	Std	Min	Max	TVD	Hellinger	KLD	Histogram			
sepal_length_cm	5.84	5.82	0.83	0.78	4.30	4.30	7.90	7.60	0.41	0.43	2.00	
sepal_width_cm	3.06	3.05	0.44	0.47	2	2.40	4.40	4.40	0.37	0.43	2.09	
petal_length_cm	3.76	3.59	1.77	1.67	1	1.10	6.90	6.60	0.25	0.31	1.06	
petal_width_cm	1.20	1.07	0.76	0.66	0.10	0.10	2.50	2.10	0.57	0.53	2.91	



# Drift Detection in Iguazio - Dashboards



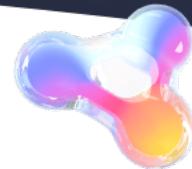
[MLRun - Model monitoring overview](#)

# Drift Detection in Iguazio - Dashboards



MLRun - Model monitoring overview

# Drift Detection in Iguazio – Re-Training Trigger



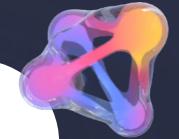
- Every drift calculation is posted in a stream
- Process will watch drift stream for events denoting possible or probable drift -> kick off CI/CD pipeline

```
> 2022-09-01 20:00:07,860 [info] Drift status: {'endpoint_id': 'd2891403ac07d5c6cdc672130a061eac29a6adef', 'drift_status': 'NO_DRIFT', 'drift_measure': 0.05010798456956764}
> 2022-09-01 20:00:08,098 [info] Done updating drift measures v3io:///projects/heart-disease-classifier/model-endpoints/parquet/key=d2891403ac07d5c6cdc672130a061eac29a6adef/year=2022/month=09/day=01/hour=19
```



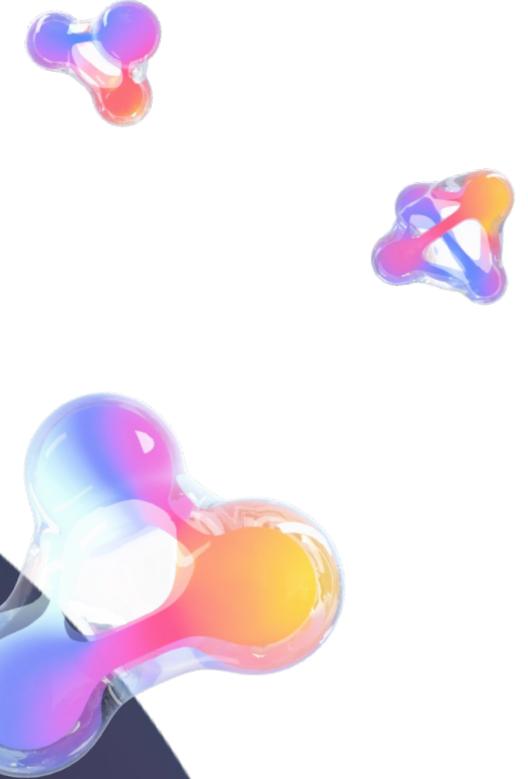
# Check out MLRun for Open Source MLOps

---



- [MLRun Website](#)
- [MLRun GitHub](#)
- [MLRun Docs](#)
- [MLRun Function Marketplace](#)
- [MLOps Live Slack Community](#)





# Iguazio and MLOps

---

Nick Schenone – DATASCI 290