E-commerce Delivery Delay Risk Prediction

Recap ...

Business Problem

Olist: e-commerce platform at scale (~9k partner retailers)

Faces **inconsistent** delivery performance, leading to a late-delivery rate that fluctuates based on operational conditions.

Solution

Late delivery prediction

Operationalize an end-to-end machine learning pipeline to **predict late deliveries**.

Key Users

Data Science Team (DS)

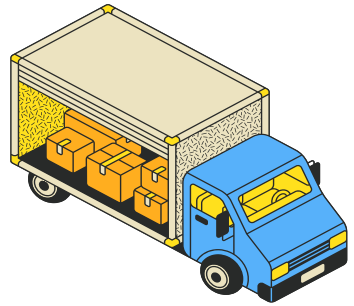
Owns and monitor the model and the entire pipeline

Logistics Team

Results of the model help them expedite orders that are at high risk of being delivered late

Customer Service Team

Results of the model help them determine which customers to proactively notify in terms of order delivery



Recap ...

Machine Learning Task

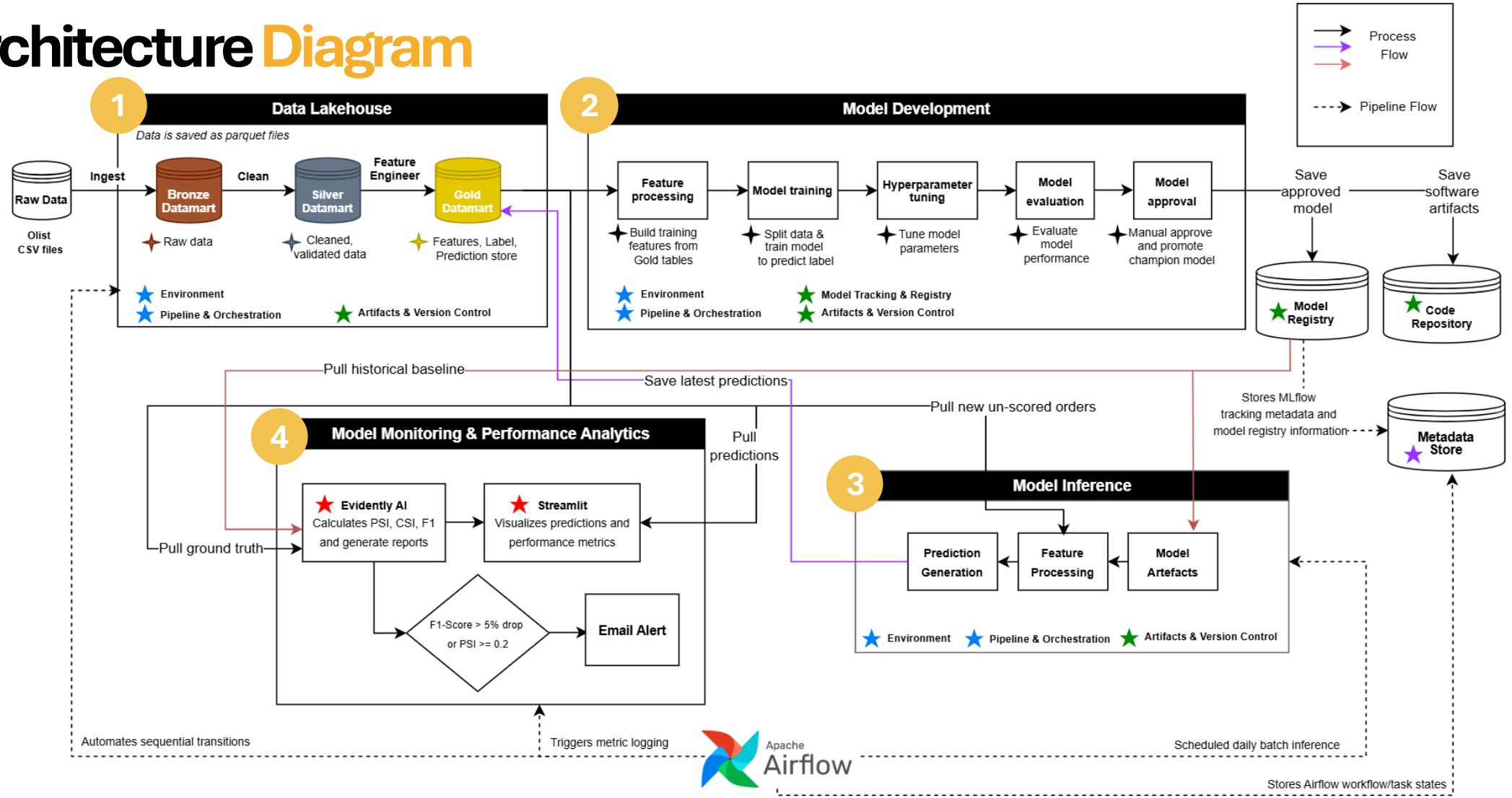
Supervised
Binary Classification

Given vector of features (X), predict late delivery status ($y \in \{0,1\}$) where 1 corresponds to late delivery while 0 corresponds to on-time delivery

Dataset

- Sourced from Kaggle and covers 100k orders from 2017 to 2018
- Consists of 8 different files:
  - orders
  - order_items
  - order_payments
  - order_reviews
  - products
  - customers
  - sellers
  - geolocation
- Features cover the end-to-end order lifecycle (order details, shipment details, reviews, etc.)

Architecture Diagram



Environment	Pipeline & Orchestration	Model Tracking & Registry	Metadata Store	Artifacts & Version Control	Model Monitoring	Observability Dashboard
docker	Apache Airflow	mlflow		GitHub	EVIDENTLY AI	Streamlit
Package scripts and dependencies into containers	Setup & trigger the end-to-end pipeline	Store experiment results and save model artifacts	Store metadata for Apache Airflow and mlflow	Manage project code, DAG files, and documentation	Calculate PSI, CSI, F1 and generate reports	Centralize monitoring of performance metrics and predictions

Data Lakehouse: Medallion Architecture

Bronze Layer

- **Creates immutable copy** of raw data
- Converts CSV files to compressed Parquet for efficient storage
- Append-only and idempotent, with lineage fields tracking when, where, and which run each record came from

Silver Layer

- Enforces schema, removes duplicates, validates keys, and flags every record with a data quality pass/fail
- **Handles domain-specific fixes** (e.g. conflicting geolocation data) and uses Delta Lake for tables with changing statuses
- Generates quality report for each table

Gold Layer

- **Builds training and inference tables** from a shared 29-feature pipeline with safeguards against target leakage
- Stores model predictions and joins them with features into an operational monitoring table that powers the fulfilment dashboard

Sample Quality Report

table	batch_date	metric	column	value
customers	6/3/2026	table		customers
customers	6/3/2026	row_count		96096
customers	6/3/2026	duplicate_rows		0
customers	6/3/2026	missing_values	customer_id	0
customers	6/3/2026	missing_values	customer_unique_id	0
customers	6/3/2026	missing_values	customer_zip_code_pr	0
customers	6/3/2026	missing_values	customer_city	0
customers	6/3/2026	missing_values	customer_state	0
customers	6/3/2026	missing_values	_ingest_date	0
customers	6/3/2026	missing_values	_batch_id	0
customers	6/3/2026	missing_values	_source_layer	0
customers	6/3/2026	missing_values	_source_file	0
customers	6/3/2026	missing_values	_ingested_at	0
customers	6/3/2026	missing_values	data_quality_remark	0
customers	6/3/2026	missing_values	data_quality_pass	0
customers	6/3/2026	data_types	customer_id	string
customers	6/3/2026	data_types	customer_unique_id	string
customers	6/3/2026	data_types	customer_zip_code_pr	string
customers	6/3/2026	data_types	customer_city	string
customers	6/3/2026	data_types	customer_state	string
customers	6/3/2026	data_types	_ingest_date	string
customers	6/3/2026	data_types	_batch_id	string
customers	6/3/2026	data_types	_source_layer	string
customers	6/3/2026	data_types	_source_file	string
customers	6/3/2026	data_types	_ingested_at	timestamp
customers	6/3/2026	data_types	data_quality_remark	string
customers	6/3/2026	data_types	data_quality_pass	boolean
customers	6/3/2026	distinct_counts	customer_id	96096
customers	6/3/2026	distinct_counts	customer_unique_id	96096
customers	6/3/2026	distinct_counts	customer_zip_code_pr	14982
customers	6/3/2026	distinct_counts	customer_city	4118
customers	6/3/2026	distinct_counts	customer_state	27
customers	6/3/2026	distinct_counts	_ingest_date	1
customers	6/3/2026	distinct_counts	_batch_id	1
customers	6/3/2026	distinct_counts	_source_layer	1
customers	6/3/2026	distinct_counts	_source_file	1
customers	6/3/2026	distinct_counts	_ingested_at	1
customers	6/3/2026	distinct_counts	data_quality_remark	1
customers	6/3/2026	distinct_counts	data_quality_pass	1

Model Development: Human-Governed Building & Deployment

Model Building & Calibration

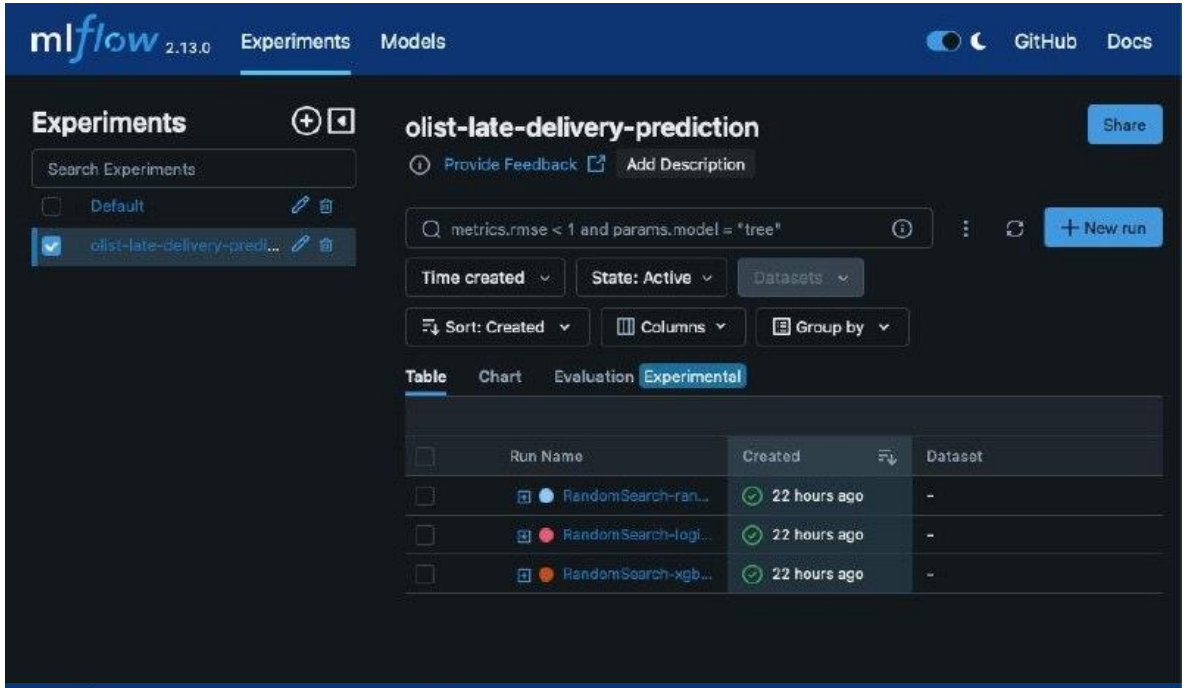
Aim: a fair, comparable set of calibrated model candidates, which is reproducible and ready for human review.

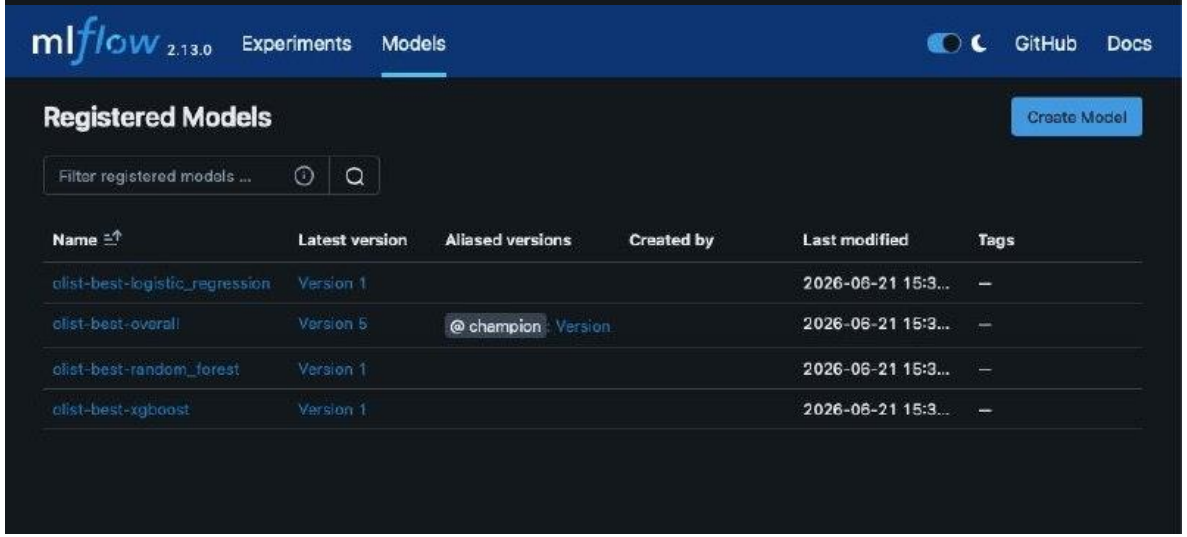
- **3 model families:** Logistic Regression, Random Forest, XGBoost; covering linear, ensemble and boosting algorithm.
- **Chronological split preventing data leakage:** train / validation / test / OOT
- **Feature Selection:** Boruta - fit once, reused across all tuning trials
- **Class imbalance handling:** balance class weights per model family
- **Hyperparameter tuning:** 20-iteration random search per model
- **Probability calibration:** sigmoid for LR, isotonic for tree models so predicted probabilities reflect **real world likelihood**.

Champion Selection & Deployment

Aim: automatic shortlisting of models but the promotion into production is based on human-decision.

- **Metric-based shortlisting:** best per family by validation F1 (balances precision/recall for imbalanced class), then compared across model families to identity best-overall
- **Human review in MLflow:** DS team inspects calibration curves, feature importance, and edge-case behaviour
- **Champion sign-off & deployment:** DS team manually promotes the approved model to the MLflow Model Registry (no DAG)
- **Human guardrail by design**

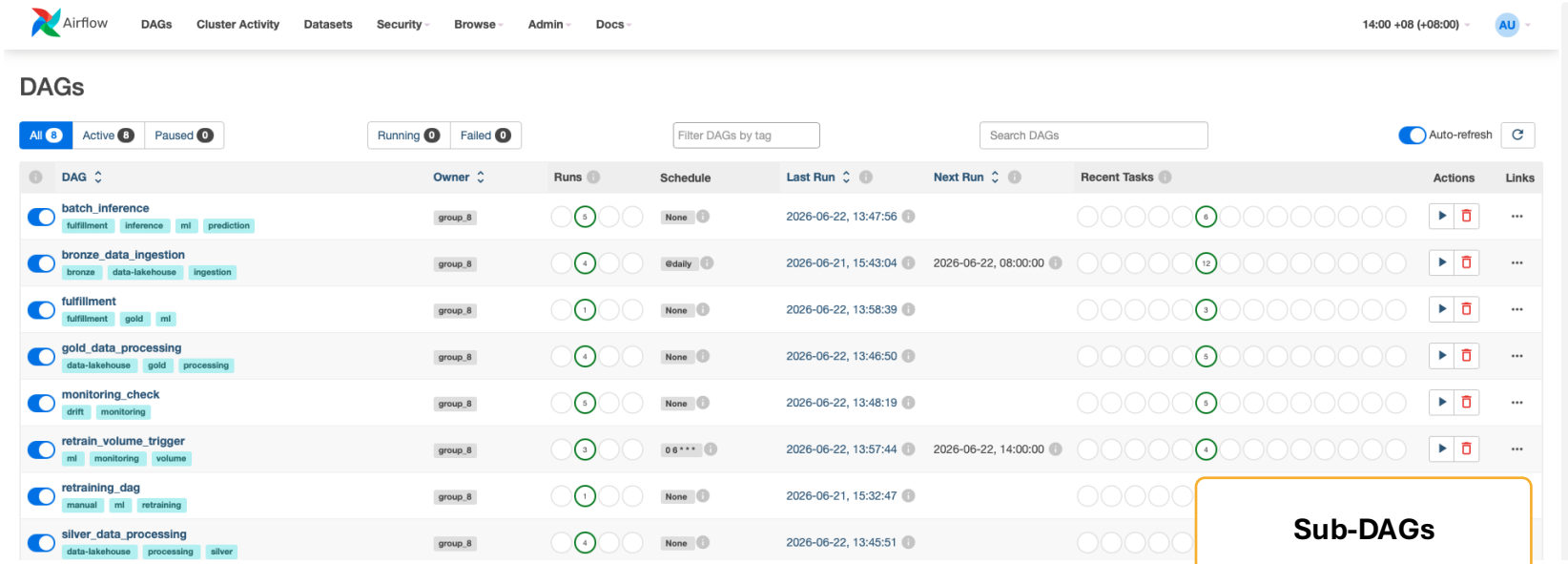




MLOps Workflow: Automated Orchestration

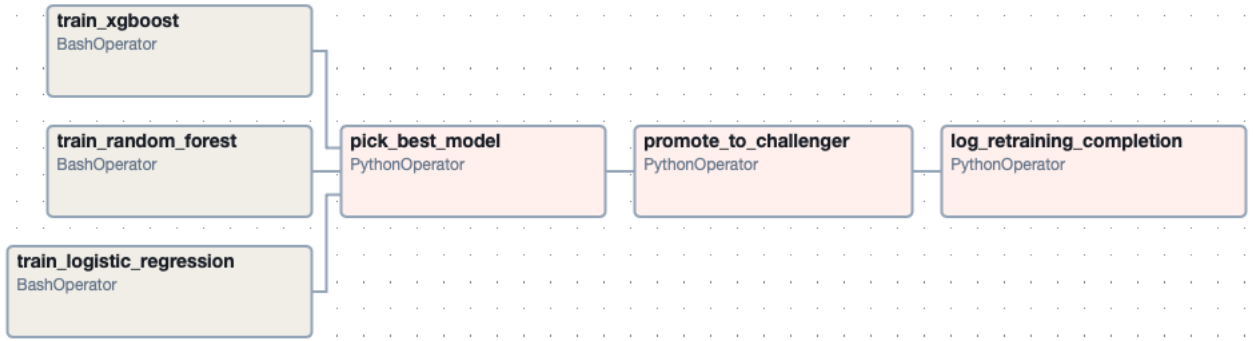
Orchestration

- **Apache Airflow DAG:** Data pipeline (Bronze →Silver → Gold) → Batch Inference → Monitoring
- **Failure isolation:** each stage can be re-run independently.
- **Champion-only inference:** same preprocessing as training, no training-serving skew.



Sub-DAGs

Sample tasks per sub-DAG



Overall DAG

MLOps Workflow: Human-Governed Monitoring and Retraining

Monitoring

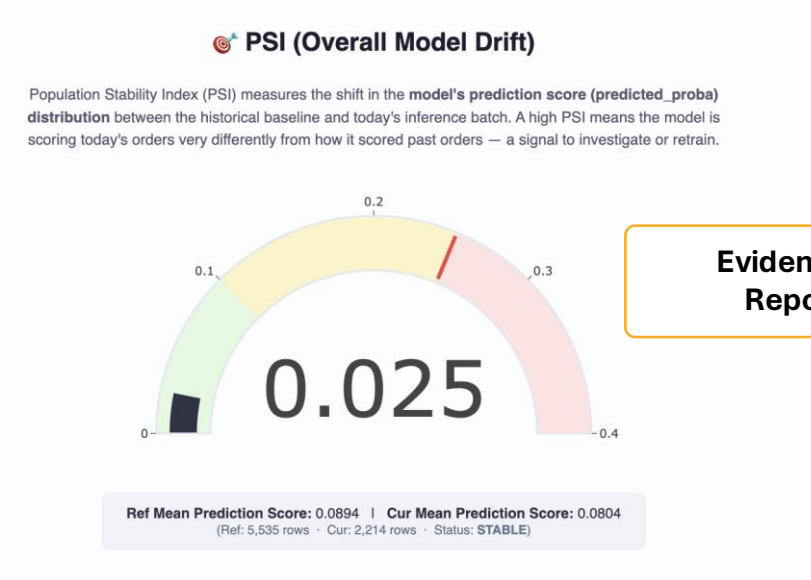
What is it for? Model and data monitoring so degradation is caught before it affect downstream users.

Who is it for? Data Science team to investigate and make decision.

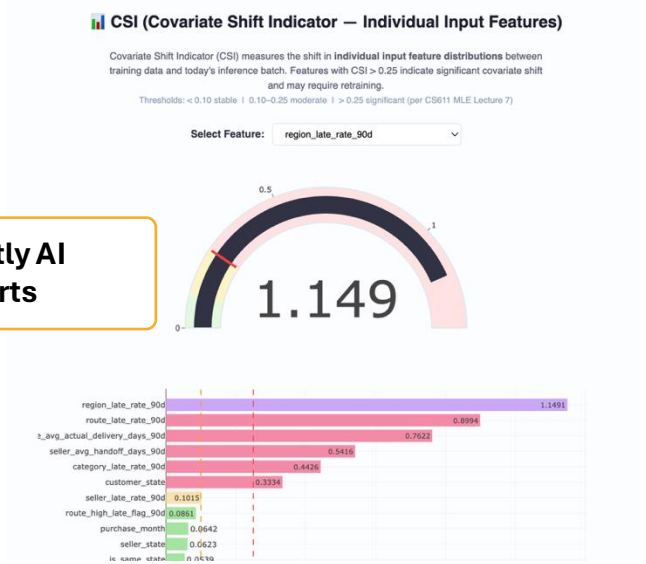
How does it work? Evidently report run automatically after each inference cycle and are surfaced in a Streamlit dashboard. Four signals are tracked, each with a defined threshold:

- **F1 7-day moving average drops > 5%** below validation baseline
- **Prediction Drift (PSI) ≥ 0.2**
- **Feature Drift (CSI) ≥ 0.2**
- **Data freshness check**: ≥ 10k new Gold rows since last retrain

If any threshold is breached, **alert email will be sent to Data Science team for further investigation.**



Evidently AI Reports



Retraining

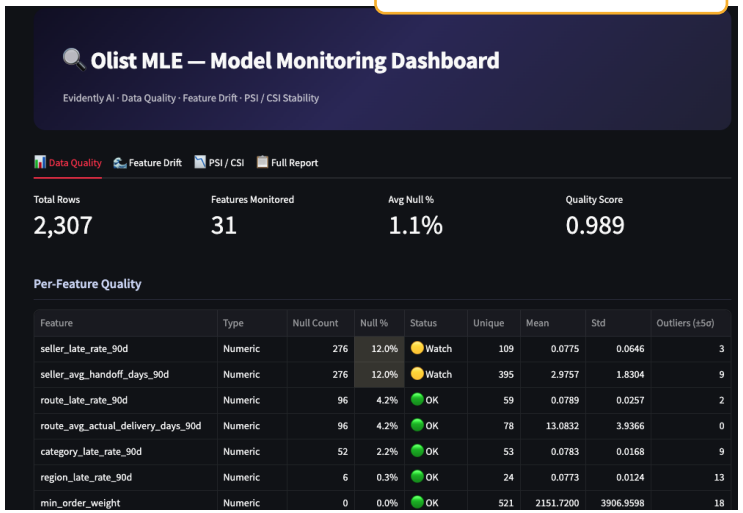
Principle: investigate first, resolve root cause. Retraining is the last resort, not the default response to every alert.

Why human-governed? Not every alert is real drift (e.g. temporary data quality). Automating retraining on every alert would be noisy and costly.

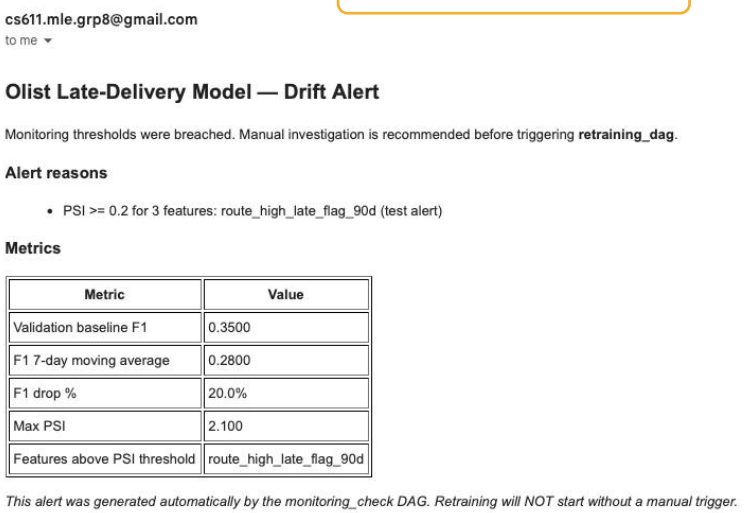
Retraining trigger: DS team initiated via Airflow, which will run all 3 models in parallel for retraining.

After retraining: champion swap still requires manual approval, same governance as initial deployment.

Streamlit Dashboard



Email Alert



Downstream Usage: Fulfillment Dashboard

What is it for?

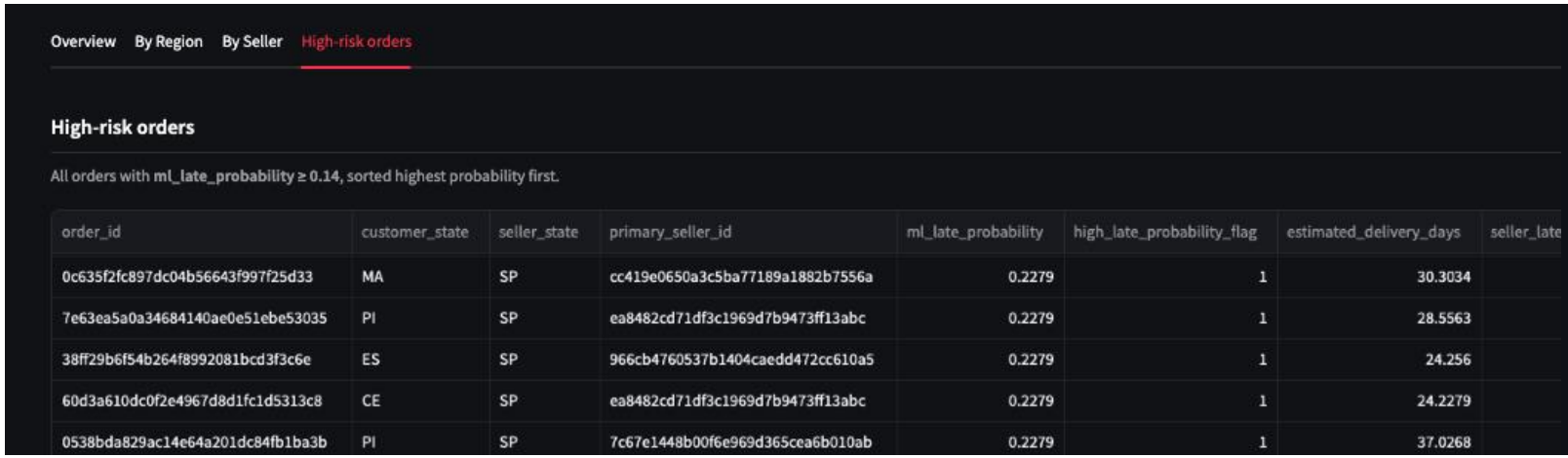
Forward-looking view of late-delivery risk at dispatch time, enabling proactive intervention before customer impact.

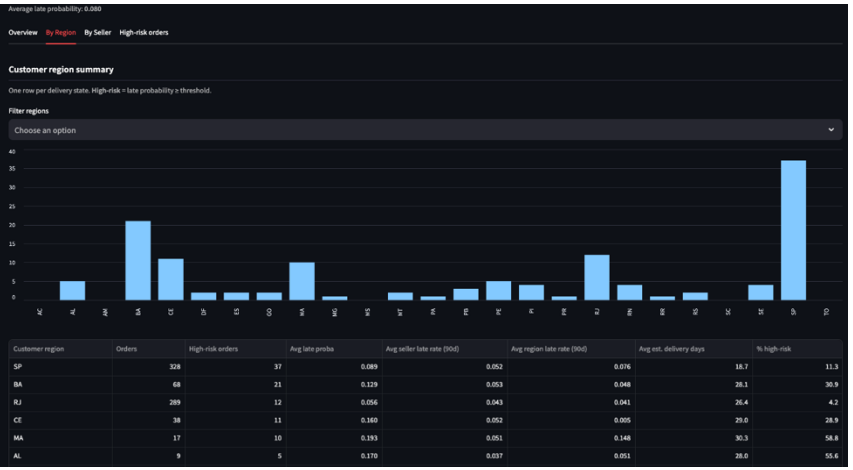
Who is it for?

Logistics and Customer Service Teams to expedite and reroute orders, and notify customers.

How it works?

Powered by Gold ml_fulfillment_monitoring datamart. It is auto-refreshed during each inference cycle.





Limitations & Future Works

1. Manual mode retrain and swap loop

Today:
Retraining and Champion model swap require human triggers, but these functions have already been implemented.

Limitation:
Slower response as relying on DS team investigation.

Future:
Steady state: alert → auto-retrain → challenger → human approval → champion swap.

2. On-premise architecture

Today:
Pipeline runs on a single Docker host.

Limitation:
No central store for predictions, MLflow runs, or monitoring artefacts.

Future:
Provision cloud infrastructure and container registry.

3. Model enhancement

Today:
Orders scored once, at "shipped" status.

Limitation:
No re-prediction as the order progresses.

Future:
Re-score at each stage of the fulfilment lifecycle, progressively refining risk estimates as more signal arrives over time.