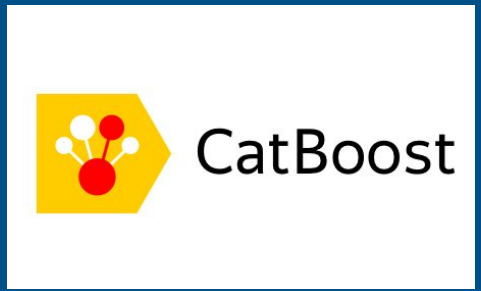
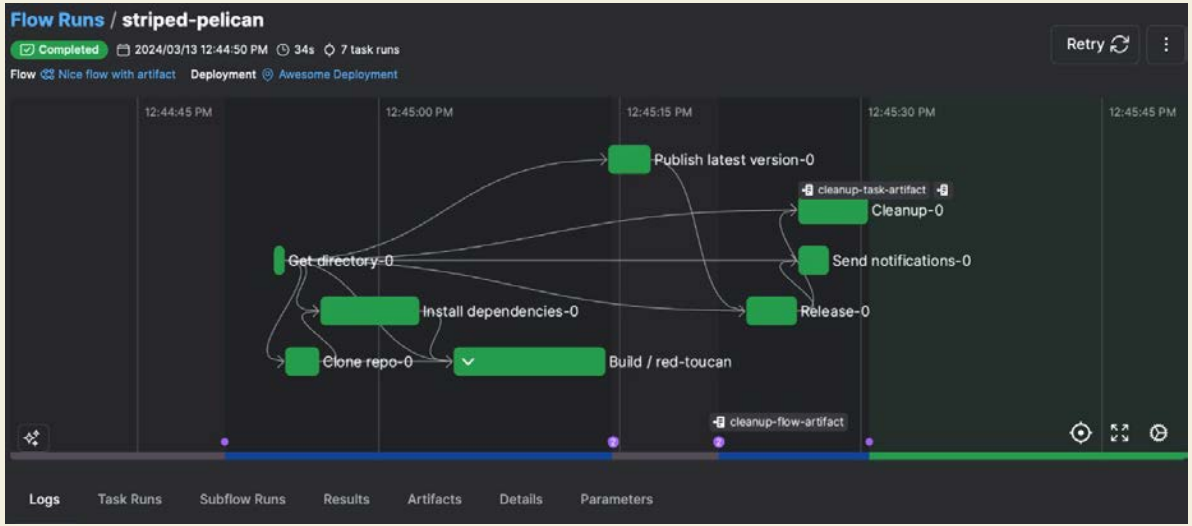
13.02.2026

Stop Failures Before They Happen: Gradient Boosting for Prefect Flows



Pavel Solikov

Prefect.io



- Orchestration
  - Flows
  - Tasks as units of work
  - States
  - Retries, caching, scheduling
- Why use it
  - Reliability + observability: logs, task timelines, retries, parameters
  - Runs are measurable and controllable

Problem statement

- What’s Failure in our case
  - SLA breach = total run time > SLA (sec)
- Prediction timing
  - On flow start time
- Possible actions
  - Abort / fail fast + explain why
  - Warn but continue
  - Reroute
  - Degrade (adjust parameters)

Idea

- Synthetic problem - portfolio risk computation
  - Generate return matrix -> compute stats (cheap) -> heavy simulation step (expensive)
- Feature sources
  - Flow params: n_assets, n_days, n_sims, missing_rate, outlier_rate
  - Code revision proxy: impl_version (categorical)
  - After Task 1: returns_missing_frac, returns_mean, returns_std, returns_kurtosis
  - Early runtime: t_first_task_s
- Timeline
  - T0: start -> features known
  - T1: after Task 1 -> stats + timing -> score P(SLA breach)
  - T2: heavy compute -> runtime realized -> SLA label assigned

Setup

- Overview
  - The Prefect flow executes for every run
  - Task 1 does data generation + stats
  - Heavy step runs real CPU work (covariance + simulation loops)
  - Nonlinear operations on input params

Modeling choice

- Primary model: CatBoostClassifier on the feature vector
  - Handles nonlinear interactions well
  - Native categorical support
  - Works great on tabular data
- Other models
  - Logistic Regression
  - Random Forest
  - XGBoost
  - HistGradientBoosting
  - 1-feature decision rule (sanity baseline)
- Evaluation
  - Holdout test split
  - ROC AUC
  - SHAP for per-run explanations

Results

- Dataset
  - 5000 runs
  - SLA fitted to get balanced 50/50 target
- Leaderboard (holdout)
  - **CatBoost: ROC AUC 0.875**
  - HistGradientBoosting: ROC AUC 0.866
  - XGBoost: ROC AUC 0.865
  - RandomForest: ROC AUC 0.865
  - Logistic: ROC AUC 0.848

Interpretability

- Model output
  - risk_score = 0.956 (before heavy task)
- Explanation (top SHAP drivers)
  - returns_kurtosis (+1.0230)
  - missing_rate (+0.9688)
  - n_sims (+0.7857)
  - returns_missing_frac (+0.7475)
  - t_first_task_s (+0.2613)

Demo

In production

Flow code is changing constantly

- Use code diff features as input for model
  - Categorize changes
    - COSMETIC_REFACTOR
    - INTERFACE_CHANGE
    - ALGORITHM_LOGIC_CHANGE
    - INFRA_DEPLOY_CHANGE
    - Etc
  - Use commit messages to infer intent
    - https://github.com/conventional-changelog/conventional-changelog
  - ChangeDistiller
    - https://bitbucket.org/sealuzh/tools-changedistiller/wiki/Home

Real world applications

- Use cases
  - Monte Carlo calculations
  - Large ETL
  - ML training pipelines
  - Scientific/HPC workflows
- Actions
  - Abort / reroute / degrade / warn with explanation
  - Allocate bigger workers only when needed
- Extension beyond SLA
  - Domain specific failures
  - Retry success