Japan GDP Growth Forecasting with Natural Disasters

Aldina Zeqiri

Abstract

This project builds a reproducible machine learning pipeline to forecast Japan's annual GDP growth using country-year aggregates of natural disasters. The objective is predictive rather than causal: to predict GDP growth in year t, the strict forecast setup uses only information observable by the end of year $t-1$. GDP and GDP growth are sourced from the World Bank World Development Indicators (WDI), while disaster outcomes are aggregated from EM-DAT (counts, deaths, damages, magnitude). To reduce leakage risk, all preprocessing is executed inside scikit-learn pipelines and time order is preserved in both the holdout split and TimeSeriesSplit cross-validation.

Across models (Ridge, Random Forest, Histogram Gradient Boosting, and an MLP benchmark) and strong time-series baselines, the best strict-forecast performance in the long-sample "main" specification is achieved by Gradient Boosting with test RMSE ≈ 1.50 (vs. a rolling-mean baseline ≈ 2.02). However, restricted-window variants (start year 1992) can have near-zero or negative R^2, consistent with small samples and noisy annual aggregates. In nowcast diagnostics (upper bound that may use same-year disaster information), adding an oil-related control substantially improves restricted performance (e.g., Ridge test RMSE ≈ 1.59, $R^2 \approx 0.40$). The report highlights why these patterns occur, with figures showing heavy-tailed damages, cross-validation instability for unregularized linear regression, and large errors driven by rare structural breaks such as COVID.

1 Introduction

Natural disasters can affect macroeconomic outcomes through capital destruction, production interruptions, supply-chain propagation, and reconstruction dynamics. The empirical macro literature emphasizes that impacts are heterogeneous across countries, event types, and institutional environments, and that aggregate effects are difficult to estimate precisely [4, 5, 6]. Japan provides salient examples where large shocks plausibly transmit through networks and supply chains [7]. At the same time, climate-related extremes are an increasingly important source of macro risk in the long run [3], which motivates testing whether disaster databases contain usable predictive signals for macro forecasting.

This project frames the problem as a forecasting benchmark. The target is Japan's annual GDP growth in year t, with a one-year-ahead horizon. A strict timing rule is enforced: predictors for year t must be observable at $t-1$. This matters because many disaster outcomes (especially damages) are finalized during or after year t, and using contemporaneous measures can overstate real-time predictability.

The contribution is practical and methodological. The repository implements a reproducible pipeline, compares multiple models to strong baselines under time-series validation, and reports diagnostics that explain success and failure cases. The goal is not to claim "ML wins," but to show when models beat persistence baselines, when they collapse to mean-like predictions, and how small annual samples and structural breaks limit what is achievable.

2 Background

A large empirical literature studies disasters and growth. Survey evidence highlights that average effects vary widely and depend on exposure, preparedness, and economic structure [5, 6]. Some work finds persistent output losses after catastrophic events, especially when damages are large relative to economic size. For Japan, supply-chain propagation can amplify localized shocks [7]. These mechanisms suggest that disaster information *could* matter for forecasting, but they also imply that coarse annual aggregates might miss timing and sectoral transmission.

Forecasting with annual macro data is intrinsically difficult: sample sizes are small and rare crises dominate losses. Standard guidance in time-series forecasting stresses leakage-free evaluation, strong baselines, and careful split design [8, 9]. This motivates the project's architecture: (i) a strict ex-ante forecasting mode, (ii) transparent baselines (mean and rolling means), and (iii) both a chronological holdout and TimeSeriesSplit cross-validation for stability checks.

3 Design & Architecture

3.1 Repository structure (reproducibility-first)

The codebase separates raw inputs, feature construction, modeling, and artifacts:

- `src/data_loading.py` loads and standardizes spreadsheet inputs from `data/`.
- `src/features.py` builds the cleaned yearly master table by aggregating EM-DAT to year level and merging with WDI GDP/growth (plus optional macro/oil series).
- `src/models.py` constructs lag/rolling features, applies strict-vs-nowcast timing rules, trains baselines and ML models, runs time-series validation, and writes outputs to `results/`.
- `main.py` is the CLI entry point that runs benchmarks and produces the saved `results/` artifacts used by plots and dashboards.
- `tests/` contains unit/integration tests that enforce correctness and reproducibility (e.g., no shuffled years).
- Optional scripts generate artifacts: `scripts/make_report_figures.py` exports report-ready PNGs/CSVs, and `dashboard/build_dashboard.py` builds an HTML dashboard.

3.2 Pipeline dataflow and run modes (forecast vs nowcast; strict vs ex_post)

Forecast vs nowcast. Forecast mode enforces the strictest timing rule: to predict gdp_growth_t, predictors must be known by $t-1$ (disaster and control variables are used as lagged features). Nowcast mode is a diagnostic upper bound: it may use same-year disaster aggregates to approximate what could be achieved with contemporaneous information.

Strict vs ex_post. In code, `-covid-mode strict` excludes an ex-post indicator. `-covid-mode ex_post` allows a simple COVID dummy (year $\geq$ 2020) as a robustness control that is not a genuine “forecastable” variable in real time. Ex_post results are therefore interpreted as a robustness/diagnostic variant, not the headline strict forecasting claim.

Inputs (`data/`)
→ WDI GDP & GDP growth (Japan, annual)
→ EM-DAT events (Japan, event-level)
→ optional macro controls + oil series (annual)

Master table (`src/features.py`)
→ aggregate EM-DAT to country-year: counts, deaths, total damage, average magnitude
→ merge with WDI by year; align calendar years
→ fill disaster aggregates with zero when no event is recorded that year

Feature engineering (`src/models.py`)
→ GDP dynamics: lags + rolling mean/std (computed from past values only)
→ transformations: log(1+damage), damage share of GDP, time trend ($year, year^2$)
→ strict forecast: use lagged versions of disaster/macro/oil features (t-1)
→ nowcast: optionally include same-year disaster features

Modeling & validation (`src/models.py`)
→ chronological holdout test window (last fraction of years)
→ time-series CV via TimeSeriesSplit (stability check)
→ compare baselines + ML models; save metrics (and CV scores) to `results/`

Artifacts
→ `results/model_metrics_<tag>.csv`
→ `results/cv_scores_<tag>.csv`
→ `figures/predictions_<tag>_<model>.csv`
→ optional HTML dashboards in `dashboard/`

Figure 1: End-to-end pipeline and dataflow used in the project.

3.3 Key design choices

First, the strict ex-ante rule is treated as a contract: to forecast year t, the pipeline only uses predictors available by $t-1$. This is implemented directly in `src/models.py` by constructing lagged versions of disaster and control variables and by computing rolling GDP statistics from shifted series.

Second, baselines are treated as serious competitors rather than placeholders. With small annual samples, simple persistence rules can be hard to beat. The project therefore includes mean and rolling-mean baselines so that any ML “gain” is interpreted relative to realistic alternatives rather than relative to a weak benchmark.

Third, multiple feature sets are used to separate signals. The “main” specification uses a long sample and includes GDP persistence plus lagged disaster aggregates. “Restricted” variants start in 1992 to test stability in more recent decades and to align with shorter-availability controls. Optional macro and oil variants test whether broad macro shocks add predictive value beyond disasters.

Fourth, model complexity is balanced against overfitting risk. Ridge regression provides a regularized linear benchmark [12]. Random Forest and Histogram Gradient Boosting allow nonlinearities

[10, 11]. An MLP is included as a high-capacity stress test; when it fits training extremely well but generalizes poorly, that is used as evidence of variance/overfitting rather than as a success.

3.4 Leakage prevention and validation protocol

Leakage prevention is enforced mechanically. Rolling features are computed from shifted GDP growth so that the rolling window ends at $t - 1$. Strict-mode covariates are shifted by one year. Preprocessing (imputation and scaling where relevant) is done inside scikit-learn pipelines so that each fold fits transformations on the training subset only.

Validation follows time-series best practice. A chronological holdout measures final out-of-sample performance. TimeSeriesSplit cross-validation provides a stability check across multiple historical folds [9]. Early folds are intentionally difficult because training windows are small; this explains why some models show unstable fold RMSE even when the final holdout comparison is stable.

3.5 Feature-set summary (what changes across runs)

Run configuration	Included feature groups (timing enforced by the run)
Main	Time trend + lagged GDP persistence (lags, rolling mean/std) + lagged disaster aggregates (counts, deaths, damages, magnitude, share)
Restricted	Same construction rules as Main, but evaluated on years $\geq$ 1992 to test stability on a later subsample
Restricted__macro	Restricted window + lagged macro controls (when available)
Restricted__oil	Restricted window + oil-related features (and optionally their changes)

Table 1: Conceptual summary of run configurations used by `main.py`.

3.6 Artifacts and outputs

The project produces machine-readable artifacts that are used consistently by the report, dashboards, and tests.

Run outputs (written by `main.py` / `models.py` into `results/`).

- `results/model_metrics_<tag>.csv`: train/test RMSE, MAE, R^2, and diagnostic counts (+ run metadata).
- `results/cv_scores_<tag>.csv`: fold-level RMSE for time-series CV stability plots.

Reporting artifacts (generated by `scripts/make_report_figures.py` into `figures/`).

- `figures/predictions_<tag>_<model>.csv`: year-by-year predictions used in plots (exported by the reporting script, not by `main.py` directly).

4 Implementation

4.1 Data sources and preprocessing

GDP and GDP growth come from WDI [1]. Disaster information comes from EM-DAT [2]. The project aggregates EM-DAT at the Japan-year level (counts, deaths, total damages, average magnitude), merges those aggregates with WDI by calendar year, and then constructs forecasting features in `src/models.py`.

Two implementation details matter for correctness. First, disaster years with no recorded events are treated as zeros in the merged master table. Second, damage values are heavy-tailed, so the model uses $\log(1 + \text{damage})$ and damage share of GDP to stabilize scale and reduce the influence of extreme outliers.

4.2 Feature engineering (how each feature is computed)

GDP persistence is represented by gdp_growth_{t-1} and gdp_growth_{t-2} plus rolling mean/std features computed from shifted series (so the window ends at $t - 1$). Disaster aggregates are transformed (log damage, damage share) and then lagged under strict forecasting. In nowcast mode, the pipeline may additionally use same-year disaster aggregates as a diagnostic upper bound.

The pipeline also includes a smooth time trend ($year$, $year^2$) because long-run macro drift can otherwise be misattributed to other predictors. Time is always known at forecasting time, so including it does not violate the ex-ante rule.

4.3 Missing values, scaling, and leakage-proof preprocessing

Feature availability differs by configuration because some macro/oil series start later. Rather than manually trimming datasets, the pipeline defines a consistent dataset contract: rows must have the target and essential GDP lag feature; remaining missing predictors are handled inside the sklearn pipeline via an imputer. Models that are sensitive to scaling (Ridge, MLP) use `StandardScaler` after imputation; tree models are evaluated with the same imputed inputs for comparability.

4.4 Model training and validation

All models are evaluated with a chronological holdout split (last fraction of years as test). TimeSeriesSplit cross-validation is used to assess stability across folds. Baselines are implemented as first-class models (mean, last-year, rolling mean) to keep comparisons honest. The benchmark code saves summary metrics and CV scores to `results/`. Per-year predictions used in plots are exported by `scripts/make_report_figures.py` to `figures/`, making figures and diagnostics reproducible.

4.5 Hyperparameter policy

Hyperparameters are conservative by default because annual time series provide limited observations. Over-tuning can overfit the validation procedure itself, especially when early folds have short training windows. Optional tuning flags exist for selected models; for the headline strict-forecast run we enable Random Forest tuning using train-only time-series CV, while keeping other choices conservative.

4.6 Testing and contracts (what the tests enforce)

The test suite is designed to catch silent mistakes that would invalidate the forecasting story (e.g., shuffled years, missing required columns, inconsistent output artifacts). Unit tests check that the master table contains required variables and that key transformations behave as intended. Integration tests run end-to-end and verify that outputs are produced deterministically.

5 Evaluation

5.1 Headline results with baselines

Table 2 reports two representative settings using the values produced by the saved `model_metrics` outputs.

Setting	Best model	RMSE	R^2	MAE	Best baseline	Baseline RMSE
Forecast (strict, main)	Gradient Boosting	1.500	0.131	1.039	Roll-3 mean	2.016
Nowcast (ex_post, restricted_oil)	Ridge	1.592	0.401	1.416	Mean (train)	2.102

Table 2: Headline test-set results with baseline comparison.

On the long-sample main specification, Gradient Boosting improves test RMSE by about 25% relative to the rolling-mean baseline, suggesting incremental predictive content beyond persistence. In restricted nowcast diagnostics, adding oil features yields a large improvement and a substantially higher R^2 on the shorter evaluation window.

5.2 Restricted nowcast results (with and without oil)

Table 3 focuses on the restricted nowcast setting and quantifies the improvement from adding oil features. "Severe disaster" years are defined as years whose damage-share proxy exceeds the 85th percentile of the training distribution; because the restricted test window is short, the number of severe test years is small, so the severe-R^2 is not reported.

Setting	Model	RMSE test	R^2 test	RMSE severe	n severe
Nowcast (ex_post, restricted)	Ridge	2.012	0.043	2.390	3
Nowcast (ex_post, restricted_oil)	Ridge	1.592	0.401	2.043	3

Table 3: Restricted nowcast performance for Ridge with and without oil features.

5.3 Prediction diagnostics and model behavior (strict forecast main)

In the strict forecast main run, a small number of extreme years dominate the test error. The COVID recession year (2020) is the largest error: the model under-predicts the magnitude of the downturn, which is expected when training on a small annual dataset with few crisis observations.

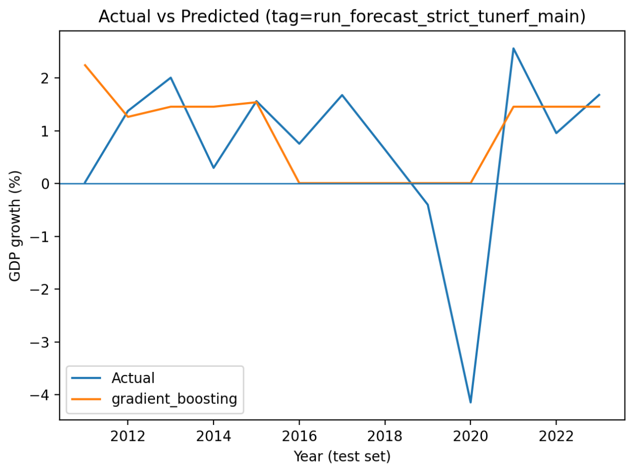

Figure 2: Strict forecast (main): Predicted vs actual GDP growth for Gradient Boosting. *Commentary.* The model tracks moderate fluctuations but underestimates rare structural breaks (notably 2020), which drive a large share of test loss.

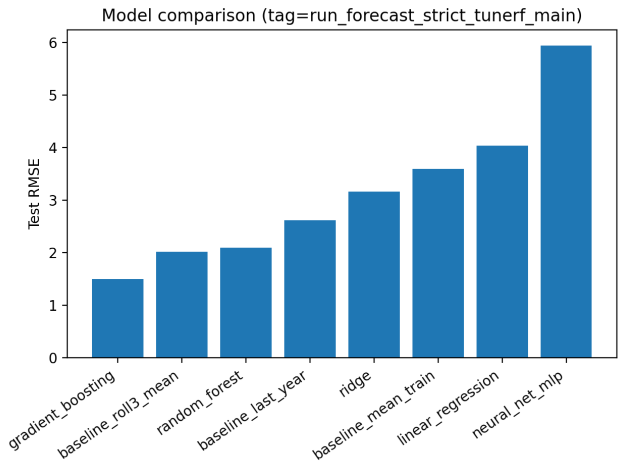

Figure 3: Model comparison (strict forecast, main): test RMSE across baselines and ML models. *Commentary.* Gradient Boosting is best on the holdout test window. Random Forest is competitive. Unregularized linear regression and the MLP generalize poorly in this small-sample setting.

5.4 Exploratory data analysis (context for the forecasting difficulty)

EDA provides context for why the task is hard. GDP growth has rare extreme years, and disaster damages are heavy-tailed, which increases sensitivity to a small number of observations and elevates overfitting risk.

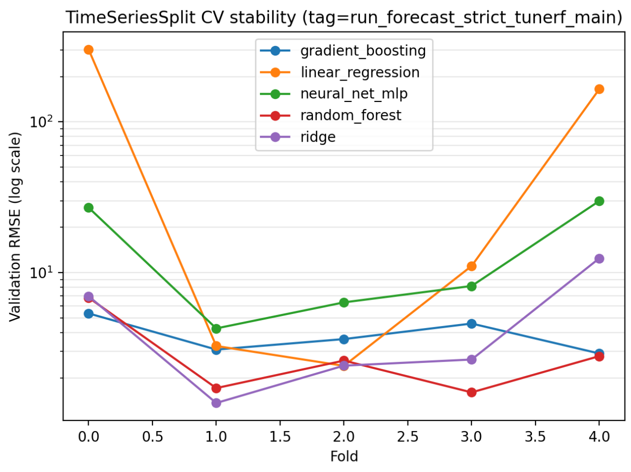

Figure 4: Time-series cross-validation stability (strict forecast, main).
Commentary. Fold RMSE can be volatile because early folds have short training windows. Linear regression shows extreme instability in some folds (note the log-scaled y-axis), while regularized and tree-based models are more stable.

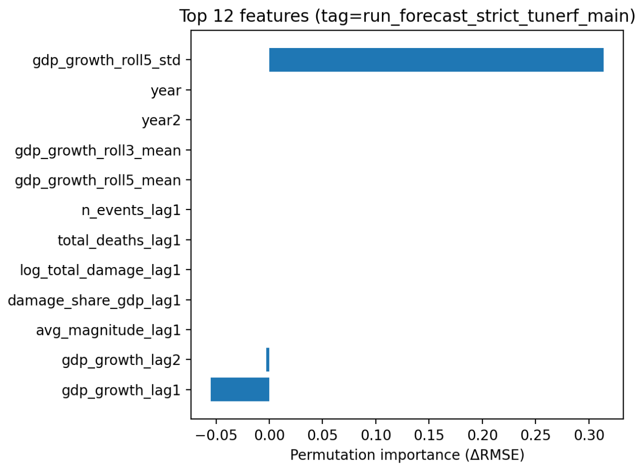

Figure 5: Permutation feature importance for the selected model (strict forecast, main).
Commentary. GDP persistence and volatility proxies (lags and rolling std) dominate predictive content. Disaster aggregates contribute limited incremental signal at annual frequency once GDP dynamics are included.

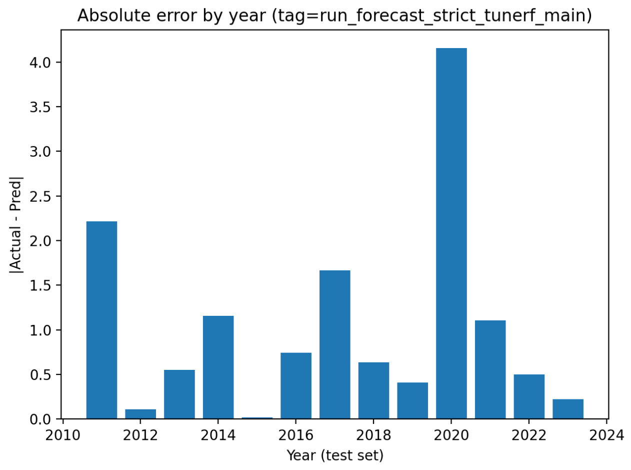

Figure 6: Absolute error by year (strict forecast, main).
Commentary. A few years account for most loss. The largest bar corresponds to the COVID recession year, consistent with "rare breaks dominate RMSE" in annual macro forecasting.

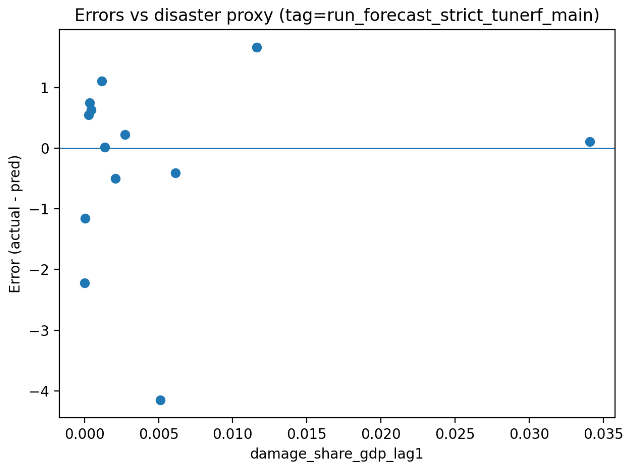

Figure 7: Errors vs disaster intensity proxy (strict forecast, main).
Commentary. There is no strong monotonic relationship between error size and the lagged disaster proxy, which is consistent with limited incremental predictive content from annual disaster aggregates.

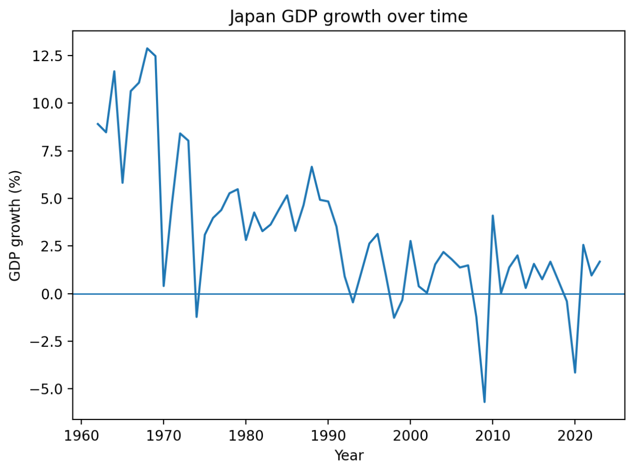

Figure 8: EDA: Japan annual GDP growth over time.
Commentary. The series features a few extreme downturns that dominate forecast error, so stability and baselines matter.

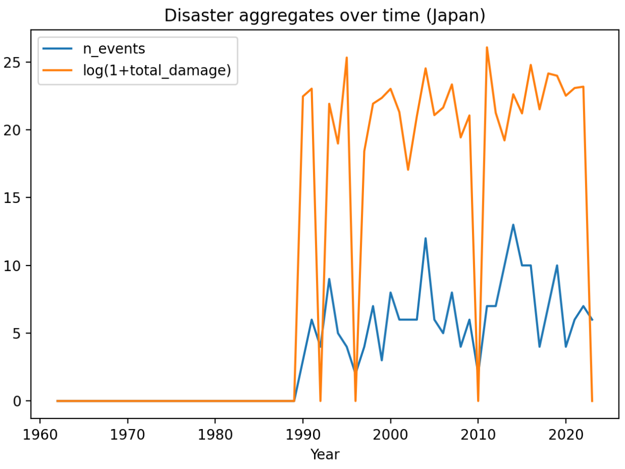

Figure 9: EDA: Disaster aggregates over time (annual Japan totals).
Commentary. Annual disaster measures are intermittent and can reflect both true event frequency and reporting/measurement changes, limiting stable signal.

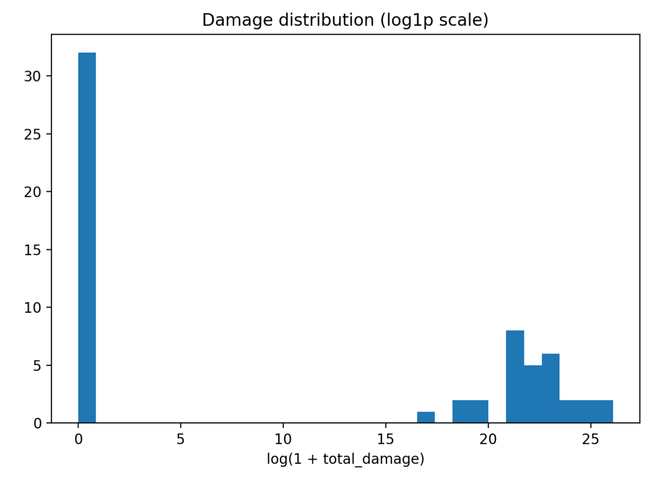

Figure 10: EDA: Distribution of reported damages (log1p scale).
Commentary. Reported damages are heavy-tailed; the log transform makes the mass of "small-to-medium" events visible and avoids the plot being dominated by a few extreme years.

6 Discussion

6.1 Interpreting what the models are really learning

The strict-forecast main results show that the best model mainly leverages GDP persistence and volatility structure (lags and rolling statistics). This is not a weakness: in annual macro data, persistence is often the dominant predictable component. The feature-importance plot supports this directly, with rolling volatility and lagged GDP features contributing far more than disaster aggregates.

This pattern is consistent with the disasters-and-growth literature, which argues that aggregate disaster impacts are heterogeneous and can be hard to summarize in a stable country-year mapping [5, 6]. Annual aggregates may blur timing (within-year), omit sectoral exposure, and miss policy response, all of which matter for macro outcomes. As a result, disasters can be economically meaningful yet still provide limited *incremental* forecasting power in this particular setup.

6.2 What the diagnostics reveal (and why they matter)

Several diagnostics explain the observed performance differences:

- **Linear regression instability in CV.** The CV stability figure shows extreme fold RMSE spikes for unregularized linear regression. With correlated predictors (lags, rolling means, trends) and small samples, OLS can be numerically unstable. Ridge improves stability by shrinking coefficients.
- **MLP overfitting.** The MLP achieves near-zero training RMSE but poor test RMSE, which is classic high-variance behavior in small annual datasets. In this context it is a useful stress test: it confirms that model capacity alone does not create signal.
- **Rare breaks dominate loss.** The year-by-year error plot shows that the largest losses come from rare structural breaks (especially 2020). A model trained on decades of "normal" years cannot reliably anticipate unprecedented shocks using only lagged aggregates.

6.3 Why the oil feature helps in restricted nowcasts

In the restricted nowcast diagnostics, adding oil features materially improves fit. This is coherent with the economic context: Japan is sensitive to global energy price shocks, and oil variables can proxy broad macro conditions that are not captured by disaster aggregates alone. The improvement does not imply disasters are unimportant; it implies that, for annual GDP growth forecasting, broad macro shocks can be more consistently informative than coarse disaster totals.

6.4 Limitations (stated as constraints, not excuses)

The primary limitation is sample size, which is especially severe in restricted variants and when additional controls reduce availability. Small samples amplify the influence of a handful of years and make cross-validation folds unstable. Another limitation is measurement: EM-DAT damage reporting is heavy-tailed and incomplete, so even with log transforms, noise remains. Finally, the project uses country-year aggregates, which omit sectoral and geographic heterogeneity; richer exposure measures and higher-frequency data would likely be needed to capture disaster transmission more reliably.

7 Conclusion

This project delivers a reproducible ML pipeline to forecast Japan's annual GDP growth using lagged disaster aggregates and optional controls, with leakage-free time-series validation. In the long-sample strict-forecast main configuration, Gradient Boosting improves meaningfully over rolling baselines, indicating incremental predictive content beyond simple persistence rules. However, disaster aggregates add limited incremental signal once GDP dynamics are included, and flexible models can overfit under small-sample constraints.

Restricted nowcast diagnostics show that adding oil-related information improves performance substantially, suggesting that broad macro shocks can be more predictive for annual growth than coarse disaster totals. Future work would benefit from higher-frequency outcomes, richer exposure measures (sectoral or regional), and explicit handling of structural breaks.

References

[1] World Bank. *World Development Indicators (WDI).*

[2] CRED / UCLouvain. *EM-DAT: The International Disaster Database*. Dataset used for disaster aggregates.

[3] IPCC. *Climate Change 2021: The Physical Science Basis (AR6 WG1)*. Cambridge University Press, 2021.

[4] Ilan Noy. "The macroeconomic consequences of disasters." *Journal of Development Economics*, 88(2), 221–231, 2009.

[5] Eduardo Cavallo and Ilan Noy. "Natural Disasters and the Economy—A Survey." *International Review of Environmental and Resource Economics*, 5(1), 63–102, 2011.

[6] Gabriel Felbermayr and Jasmin Gröschl. "Naturally negative: The growth effects of natural disasters." *Journal of Development Economics*, 111, 92–106, 2014.

[7] Vasco M. Carvalho, Makoto Nirei, Yukiko U. Saito, and Alireza Tahbaz-Salehi. "Supply Chain Disruptions: Evidence from the Great East Japan Earthquake." *Quarterly Journal of Economics*, 136(2), 1255–1321, 2021.

[8] Rob J. Hyndman and George Athanasopoulos. *Forecasting: Principles and Practice* (3rd ed.), 2021.

[9] Christoph Bergmeir, Rob J. Hyndman, and Bonsoo Koo. "A note on the validity of cross-validation for evaluating autoregressive time series prediction." *Computational Statistics & Data Analysis*, 2018.

[10] Leo Breiman. "Random Forests." *Machine Learning*, 45, 5–32, 2001.

[11] Jerome H. Friedman. "Greedy function approximation: A gradient boosting machine." *Annals of Statistics*, 29(5), 1189–1232, 2001.

[12] Arthur E. Hoerl and Robert W. Kennard. "Ridge Regression: Biased Estimation for Nonorthogonal Problems." *Technometrics*, 12(1), 55–67, 1970.

Appendix (supplementary material)

The remainder of this document provides supplementary material (reproducibility notes, AI tool disclosure, and extra figures).

A Extra EDA figure (raw damage scale)

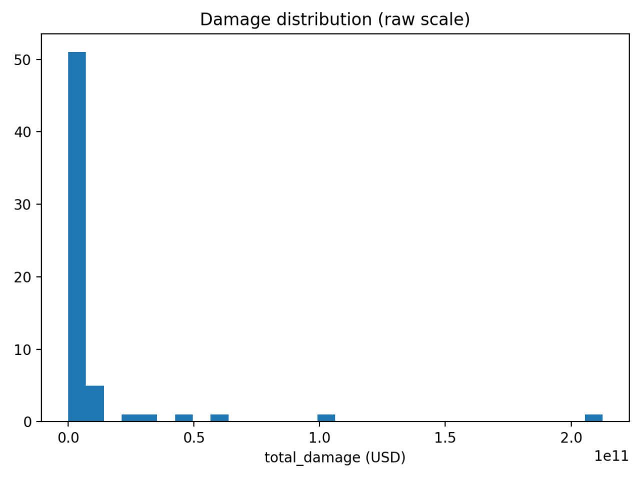

Figure 11: EDA: Distribution of reported damages (raw scale).
Commentary. On the raw USD scale, a small number of extreme observations dominate the histogram. This is why log transforms are used for modeling and for readable EDA plots.

B Reproducibility (optional)

Minimum run (safe for graders).

```
python3 main.py --only-main --tune-rf
```

Run all configurations.

```
python3 main.py
```

Build report figures (copy PNGs into Overleaf folder figuresml/).

```
python3 scripts/make_report_figures.py --tag run_forecast_strict_tunerf_main
```

Build dashboard (optional).

```
python3 dashboard/build_dashboard.py --tag run_forecast_strict_tunerf_main
```

Key outputs.

- `results/model_metrics_*.csv`: summary metrics per run.
- `results/cv_scores_*.csv`: CV fold RMSE per run.
- `figures/predictions_*.csv`: year-by-year predictions.
- `dashboard/*.html`: interactive dashboards.

C AI tool usage disclosure (optional)

I used AI tools as assistance **as needed** during development:

- **ChatGPT**: debugging help, pipeline improvements (time-series validation, artifacts), occasional small code snippets, and English writing edits (clarity, structure).
- **GitHub Copilot**: auto-completion and small syntax adjustments.

I take full responsibility for the final code, results, and interpretation.