

# 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: in the **strict forecast** setup, predicting GDP growth in year  $t$  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 (event counts, deaths, damages, magnitude). To prevent leakage, all preprocessing is executed inside scikit-learn pipelines, rolling features are computed on shifted series, and time order is preserved in both the chronological holdout split and TimeSeriesSplit cross-validation.

Across models (Ridge, Random Forest, Histogram Gradient Boosting, and an MLP benchmark) and strong baselines, the best strict-forecast performance in the long-sample **main** specification is achieved by Gradient Boosting with test RMSE  $\approx 1.50$  (vs. a rolling-mean baseline  $\approx 2.02$ ). Restricted-window variants (start year 1992) can have near-zero or negative  $R^2$ , consistent with small samples and noisy annual aggregates. The project also reports **nowcast** diagnostics (an upper bound that may use same-year information and, in **ex\_post** mode, a COVID dummy). In restricted nowcasts, adding oil-related controls substantially improves performance (e.g., Ridge test RMSE  $\approx 1.59$ ,  $R^2 \approx 0.40$ ). Figures explain these patterns via heavy-tailed damages, cross-validation instability for unregularized OLS, and large errors driven by rare structural breaks such as 2020.

## 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)

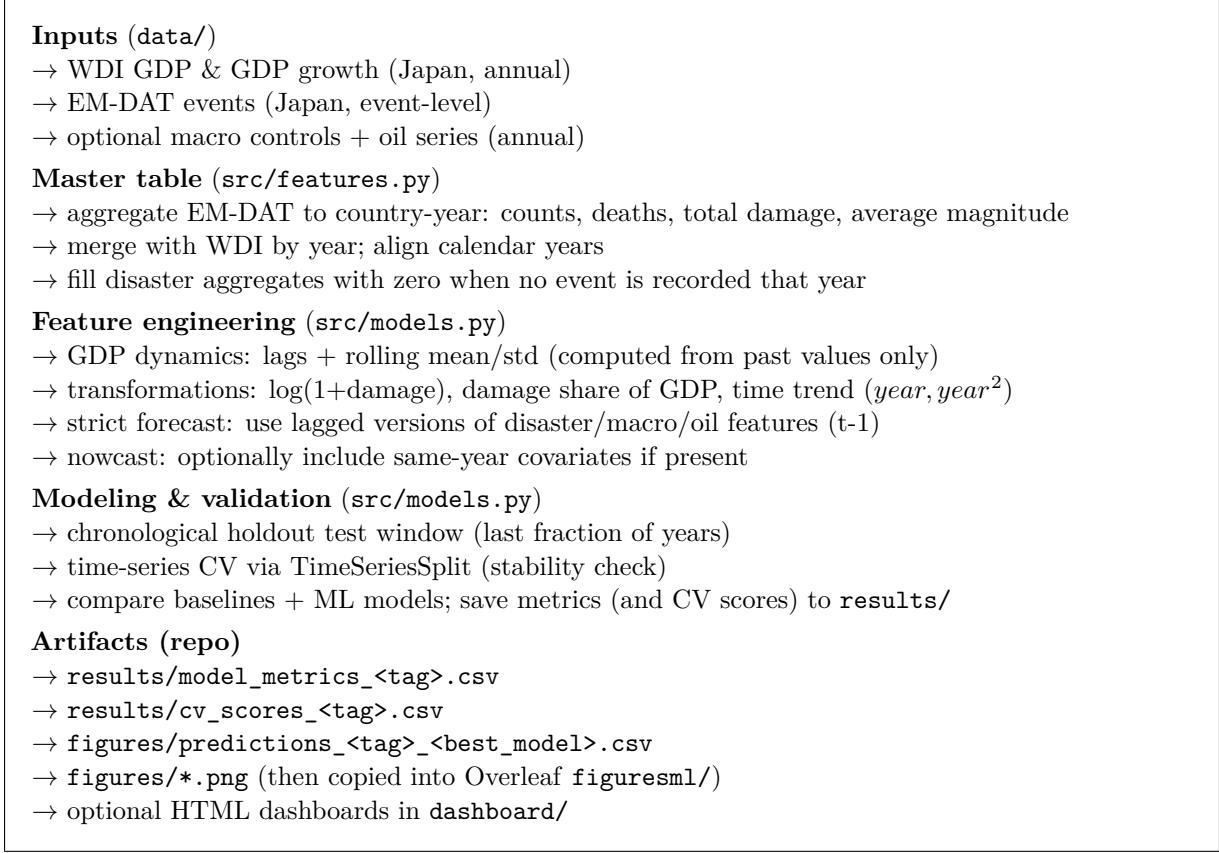
The codebase separates raw inputs, feature construction, modeling, and artifacts:

- `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 information (when present in the feature set) to approximate what could be achieved with contemporaneous data.

**Strict vs ex\_post.** In code, `-covid-mode strict` excludes an ex-post indicator. `-covid-mode ex_post` allows a simple COVID dummy ( $\text{year} \geq 2020$ ) 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.



**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 Dataset coverage and evaluation windows (exact years)

To make results fully auditable, Table 1 reports the exact years and sample sizes used by the configurations cited in the Evaluation section. The holdout split is deterministic with `test_ratio = 0.2`.

| Run cited in report                  | Tag (CSV suffix)                                | Years used (N), Train window, Test window                   |
|--------------------------------------|---|---|
| Strict forecast (main, tuned RF run) | <code>run_forecast_strict_tunerf_main</code>    | 1962–2023 (62); Train: 1962–2010 (49); Test: 2011–2023 (13) |
| Nowcast (ex_post, restricted)        | <code>run_nowcast_ex_post_restricted</code>     | 1992–2023 (32); Train: 1992–2016 (25); Test: 2017–2023 (7)  |
| Nowcast (ex_post, restricted_oil)    | <code>run_nowcast_ex_post_restricted_oil</code> | 1992–2023 (32); Train: 1992–2016 (25); Test: 2017–2023 (7)  |

**Table 1:** Exact evaluation windows for the configurations cited in the report.

### 3.6 Feature-set summary (what changes across runs)

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

**Table 2:** Conceptual summary of run configurations used by `main.py`.

### 3.7 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 such as `start_year`, `test_ratio`, `n_train`, `n_test`, `n_features`).
- `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>_<best_model>.csv`: year-by-year predictions for the best model in that run (selected by lowest test RMSE).
- `figures/*.png`: report-ready figures (comparison, CV stability, feature importance, EDA).

For Overleaf, I copy the generated PNGs (and any CSVs needed) from repo `figures/` into the Overleaf folder `figuresml/`, which is why the report uses `\graphicspath{{figuresml/}}`.

## 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 covariates if they are part of the feature set (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 strict-forecast report run we enable Random Forest tuning using train-only time-series CV, while keeping other choices conservative (see Appendix B).

### 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 3 reports two representative settings using the values produced by saved `model_metrics` outputs: `results/model_metrics_run_forecast_strict_tunerf_main.csv` and `results/model_metrics_run_nowcast_ex_post_restricted_oil.csv`.

| Setting                           | Best model        | RMSE  | R <sup>2</sup> | 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 3:** Headline test-set results with baseline comparison (values from saved `results/model_metrics_*.csv`).

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)

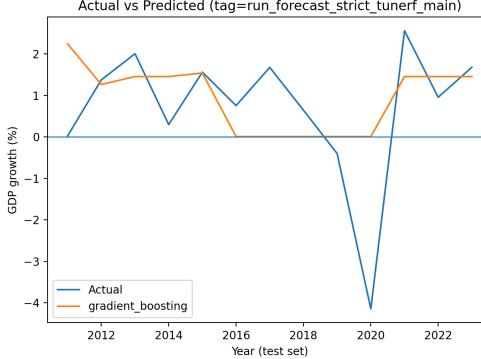
Table 4 focuses on restricted nowcasts and quantifies the improvement from adding oil features using: `results/model_metrics_run_nowcast_ex_post_restricted.csv` and `results/model_metrics_run_nowcast_ex_post_restricted_oil.csv`. “Severe disaster” years are defined as years whose damage-share proxy exceeds the 85th percentile of the training distribution; because the restricted test window is short, the number of severe test years is small.

| 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 4:** Restricted nowcast performance for Ridge with and without oil features (values from saved `results/model_metrics_*.csv`).

## 5.3 Prediction diagnostics and model behavior (strict forecast main)

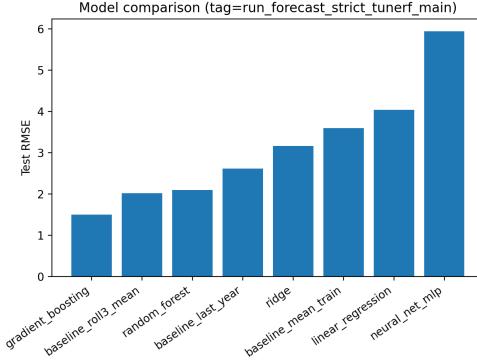
In the strict forecast main run, a small number of extreme years dominate the test error. The COVID recession year (2020) is the largest error: the model under-predicts the magnitude of the downturn, which is expected when training on a small annual dataset with few crisis observations.



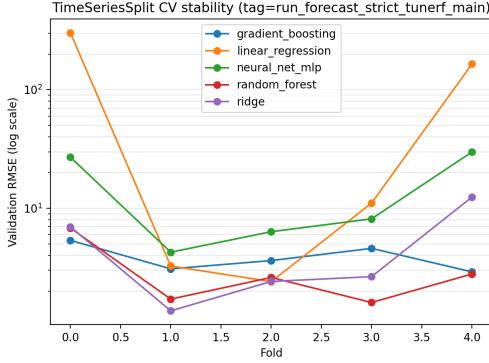
**Figure 2:** Strict forecast (main): Predicted vs actual GDP growth for Gradient Boosting.  
*Commentary.* The model tracks moderate fluctuations but underestimates rare structural breaks (notably 2020), which drive a large share of test loss.

## 5.4 Exploratory data analysis (context for the forecasting difficulty)

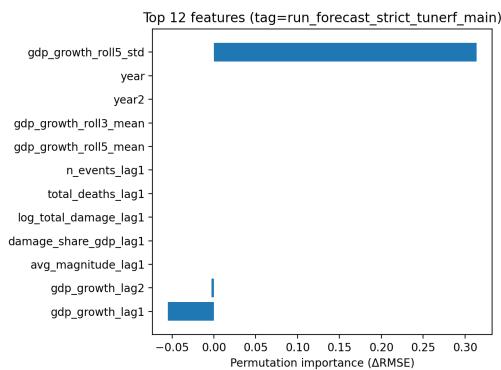
EDA provides context for why the task is hard. GDP growth has rare extreme years, and disaster damages are heavy-tailed, which increases sensitivity to a small number of observations and elevates overfitting risk.



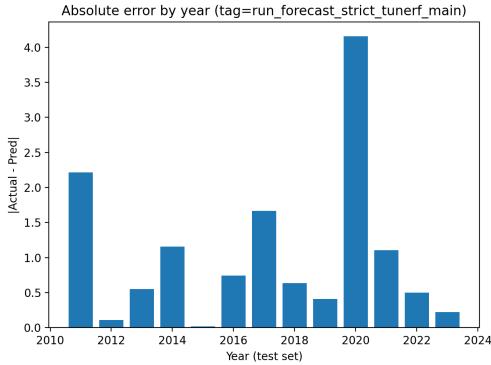
**Figure 3:** Model comparison (strict forecast, main): test RMSE across baselines and ML models.  
*Commentary.* Gradient Boosting is best on the holdout test window. Random Forest is competitive. Unregularized linear regression and the MLP generalize poorly in this small-sample setting.



**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.

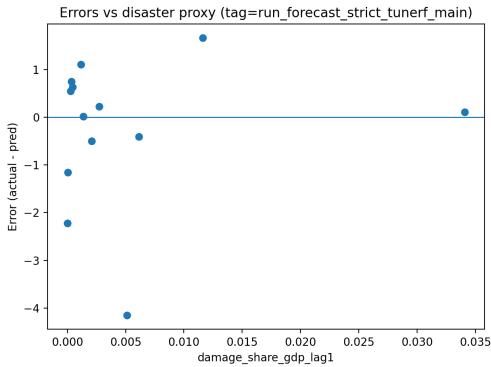


**Figure 5:** Permutation feature importance for the selected model (strict forecast, main).  
*Commentary.* GDP persistence and volatility proxies (lags and rolling std) dominate predictive content. Disaster aggregates contribute limited incremental signal at annual frequency once GDP dynamics are included.



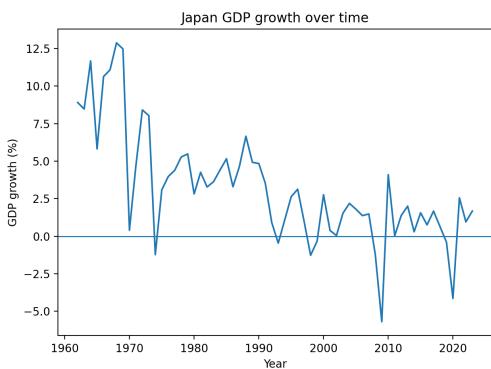
**Figure 6:** Absolute error by year (strict forecast, main).

*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.



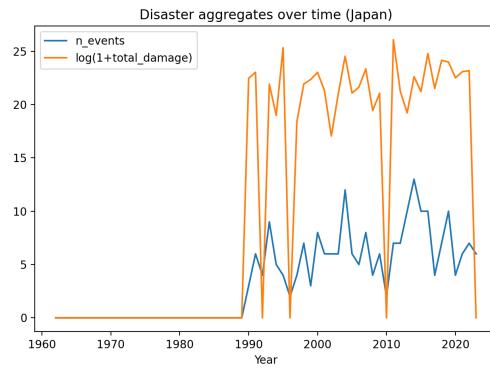
**Figure 7:** Errors vs disaster intensity proxy (strict forecast, main).

*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.



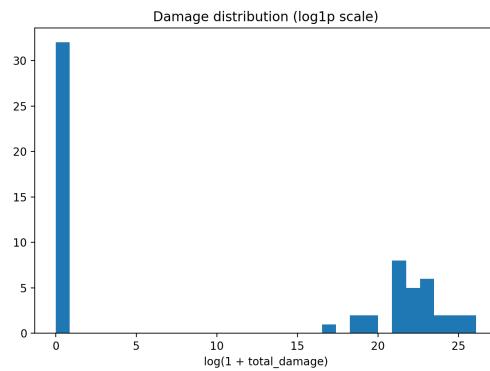
**Figure 8:** EDA: Japan annual GDP growth over time.

*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 main limitations are structural rather than implementation-specific. Annual data imply small samples, especially in restricted windows, which increases variance and makes time-series CV unstable. Disaster measures (in particular reported damages) are noisy and heavy-tailed, and annual aggregation blurs within-year timing and sectoral transmission. Finally, rare structural breaks (e.g., 2020) dominate forecast errors and cannot be learned reliably from a handful of crisis observations. To mitigate methodological risks, the pipeline enforces strict timing ( $t-1$  information only), computes rolling features on shifted series, and fits all preprocessing inside sklearn pipelines to prevent leakage.

## 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.

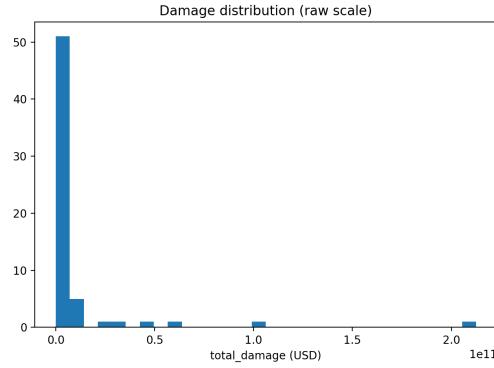
## References

- [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, hyperparameter details, 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 Hyperparameters and tuning grids (as implemented)

This appendix summarizes the defaults and the optional tuning grids implemented in `src/models.py`. These are included for transparency; the headline results use conservative choices to reduce overfitting risk in small annual samples.

| Model                       | Defaults / tuning grid used in code   |
|-----------------------------|---|
| Ridge                       | Default pipeline uses <code>Ridge(alpha=5.0, random_state=42)</code> with imputation + standardization.   |
| Random Forest               | Default pipeline uses <code>n_estimators=600, max_depth=5, min_samples_leaf=2</code> . If <code>-tune-rf</code> : grid over <code>n_estimators {200,400,600}, max_depth {None,5,10}, min_samples_split {2,5}, min_samples_leaf {1,2}</code> with TimeSeriesSplit CV.  |
| Histogram Gradient Boosting | Default uses <code>HistGradientBoostingRegressor(learning_rate=0.05, max_depth=3, max_iter=600, l2_regularization=0.1, early_stopping=False, random_state=42)</code> . If <code>-tune-gb</code> : grid over <code>learning_rate {0.03,0.05,0.1}, max_depth {2,3,4}, max_iter {300,600,1200}, l2_regularization {0.0,0.1,1.0}</code> . |
| MLP                         | Benchmark uses <code>hidden_layer_sizes=(32,16), alpha=1e-3, max_iter=5000</code> , with <code>early_stopping=False</code> , plus imputation + standardization.   |

**Table 5:** Hyperparameter policy matching the implementation in `src/models.py`.

## C Reproducibility (optional)

(A) Strict forecast main run used for the report figures (tag: `run_forecast_strict_tunerf_main`).

```
python3 main.py --mode forecast --covid-mode strict --only-main --tune-rf
```

(B) Full strict forecast suite (all forecast configurations).

```
python3 main.py --mode forecast --covid-mode strict --tune-rf
```

(C) Nowcast `ex_post` suite (needed for the nowcast tables).

```
python3 main.py --mode nowcast --covid-mode ex_post
```

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 test predictions exported by the reporting script.
- `figures/*.png`: report-ready figures exported by the reporting script.
- `dashboard/*.html`: interactive dashboards.

## D 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.