Advanced Programming 2025

Forecasting Dept dynamics using Machine Learning

Final Project Report

Yanik Bittel
yanik.bittel@unil.ch
Student ID: 21-200-175

January 11, 2026

Abstract

Sovereign debt crises are rare but extremely costly events, and early warning tools are useful for policy makers and investors. This project builds a supervised machine-learning framework to predict the probability of a sovereign debt crisis within different forward-looking horizons (3, 5, and 10 years). I merge annual macroeconomic indicators from the IMF World Economic Outlook with a global sovereign crisis database (Harvard / Laeven–Valencia style crisis dating), then construct a panel dataset and engineer lagged predictors capturing fiscal conditions, external imbalances, inflation, labour-market stress, and debt dynamics. To avoid look-ahead bias, models are evaluated with a chronological train–test split. I compare a Logistic Regression benchmark, a Random Forest, and an XGBoost classifier, using probability outputs and a decision threshold chosen to reduce missed crises. Results show that tree-based models generally outperform the linear baseline in identifying crisis periods, while maintaining reasonable discrimination. Feature-importance and SHAP-style analyses indicate that government debt-to-GDP and related fiscal variables are among the strongest risk drivers, with additional roles for growth and external balance measures depending on the horizon. Overall, the main contribution is a reproducible pipeline (data processing, modeling, evaluation, and interpretability) for multi-horizon sovereign crisis risk assessment.

Keywords: data science, Python, machine learning, sovereign dept forecasting,

Contents

1 Introduction

Sovereign debt crises are rare but highly disruptive events. When a government loses market access or is forced into debt restructuring, the macroeconomic fallout can be severe: financing conditions tighten abruptly, fiscal adjustment becomes unavoidable, and the contraction can spill over to the banking system and the broader economy. Because the policy tools available to prevent or manage such episodes (front-loaded consolidation, liquidity support, external assistance, or restructuring) are costly and politically contentious, there is strong demand for credible early-warning signals showing when debt vulnerabilities are building. At the same time, forecasting sovereign debt crises is intrinsically difficult. First, the outcome is infrequent, which complicates statistical learning and makes naive accuracy measures misleading. Second, crisis definitions and reporting practices differ across sources, so the "ground truth" label is imperfect. Third, many crises are triggered or amplified by shocks only partially captured by standard macroeconomic indicators (political instability, shifts in investor sentiment, commodity price collapses, or global financial tightening), which implies a natural upper bound on out-of-sample predictability based on macro data alone.

This project investigates whether a set of observable macro-financial indicators contains enough information to anticipate sovereign debt crises in advance, and how predictability changes across forecast horizons. I construct an annual country-year panel by combining IMF World Economic Outlook (WEO) macro variables with crisis events from the Laeven and Valencia database (accessed via Harvard Business School's Behavioral Finance and Financial Stability data portal). The empirical task is framed as a supervised classification problem: for each country-year t, estimate the probability that the country experiences a sovereign debt crisis within the next h years, focusing on horizons $h \in \{3, 5, 10\}$. As benchmark, I estimate a transparent logistic regression model, and then compare it to more flexible non-linear tree-based approaches (Random Forest and XGBoost) that can capture interactions and threshold effects that are plausible in sovereign risk dynamics. Because crises are rare, model performance is evaluated with out-of-sample metrics adapted to imbalanced classification, including ROC-AUC, PR-AUC (average precision), precision, recall, F1 score, and confusion matrices. In addition, the project emphasizes decision-relevant thresholding: rather than using a fixed 0.5 cutoff, I also select a classification threshold on a validation set to reduce false negatives (missing crises), which is often the more costly error in an early-warning context. The remainder of the report is organized as follows: Section 2 will fo through related work, Section 3 will talk about the actual coding done in this project, Section 4 reports the empirical results and compares model performance across horizons, Section 5 is left for discussion about results, and Section 6 concludes.

2 Literature Review / Related Work

Recent Machine Learning work on sovereign vulnerability differs mainly along three dimensions: the outcome being predicted (crisis events vs. spreads vs. ratings), the information set (macroeconomic fundamentals alone vs. enriched with political, market, and sentiment variables), and the preferred ML families (linear baseline vs. tree ensembles / boosting vs. neural models). Across these choices, a consistent message is that sovereign risk is driven by nonlinear dynamics and interactions, which makes flexible models attractive-provided evaluation is truly out of sample.

A first distiction concerns the dependent variable. Early-warning studies predict a binary crisis indicator, aiming to signal distress ahead of time; this is the approach in Alminos et al. (2021), who model sovereign debt crises in a long cross-country panel. By contrast, Belly et al. (2022) forecast sovereign bond spreads, a market price reflecting both fundamentals and time-varying risk premia, while Takawira (2024) predicts sovereign credit ratings, an ordinal assessment that embeds agency judgement and may adjust slowly. These outcomes are related but not interchangeable: spreads can react quickly to news and policy regimes, ratings can be persistend, and crisis indicators are rare and "lumpy". Consequently, strong predictive performance for discrete crisis events (Belly et al., 2022; Takawira, 2024), while crisis models face stronger class-imbalance constraints (Alaminos et al. 2021)

A second theme is the construction of the predictor set. Alaminos et al. (2021) combine standard macrovariables from the IMF and the WB with political and conflict/fragility measures sourced from the POLITY IV data. These include indicators such as the polity score (autocracy–democracy), regime durability, state fragility, and conflict intensity proxies. The implicit argument is that some sovereign debt crises are not purely the culmination of gradual macro deterioration; instead, political shocks (instability, conflict, regime change) can trigger sudden financing stress or amplify existing vulnerabilities. This contrasts with the spread-forecasting setting in Belly et al. (2022), where the dataset is deliberately high-dimensional and includes not only macro fundamentals but also financial-market variables, sentiment proxies (e.g., Google Trends), and euro-area-specific risk channels (including proxies for redenomination risk). Here, the conceptual rationale is that market pricing of sovereign risk is sensitive to high-frequency information and regime risk, so models benefit from broader signals than macro fundamentals alone. Takawira (2024) sits in between: the information set is primarily macroeconomic and the sample is narrower (single-country quarterly data), which is well-suited to rating prediction but limits generalizability and may partly explain very high classification performance when ratings move infrequently.

A third common theme is method choice and model performance. Across crisis prediction, spread forecasting, and rating prediction, the strongest evidence tends to favor tree-based ensembles and boosting, particularly gradient boosting implementations such as XGBoost (Alaminos et al., 2021; Belly et al., 2022; Takawira, 2024). The economic reason is straightforward: sovereign vulnerability is often shaped by threshold effects (e.g., debt is especially dangerous when growth slows) and interactions (weak external buffers matter more under tight global financial conditions), which tree/boosting methods capture naturally. In Alaminos et al. (2021), ensemble methods frequently dominate because they can combine macro variables with political-shock indicators in a nonlinear way; in Belly et al. (2022), boosting performs well because spread dynamics are highly nonlinear and time varying; and in Takawira (2024), flexible classifiers outperform logistic regression when the mapping from macro indicators to rating categories is nonlinear. At the same time, all three contexts highlight the continuing role of logistic regression as a baseline: it is transparent, easy to interpret, and provides a disciplined benchmark for whether ML adds value beyond a simple parametric model (Alaminos et al., 2021; Takawira, 2024). The main strengths of this literature are; systematic benchmarking of ML against classical baselines, recognition that feature-rich datasets can matter either via political instability variables in crisis prediction or via broader market/sentiment inputs in spread forecasting and empirical evidence that boosting/ensemble methods often deliver gains in out of sample prediction (Alaminos et al., 2021) (Belly et al., 2022) (Takawira, 2024). The limitations are equally important for an economics oriented interpretation. First, crisis prediction is a rare event problem, so headline accuracy can be misleading without careful emphasis on recall/precision and false-alarm trade-offs (Alaminos et al., 2021). Second, spread forecasting is sensitive to monetary-policy regimes and risk premia, which can weaken the link between "predicting spreads" and "predicting crises," and can reduce external validity outside the euro-area setting (Belly et al., 2022). Third, rating prediction can show very high performance partly because ratings are persistent and incorporate qualitative information; this makes the exercise informative, but not a direct substitute for crisis prediction, and single-country designs can limit general conclusions (Takawira, 2024). Finally, richer ML models increase the risk of overfitting, so strict time respecting out of sample evaluation and transparent robustness checks are essential across applications.

Overall, the combined evidence supports using a transparent baseline (logit) while benchmarking tree/boosting models as primary candidates, and it motivates expanding the information set beyond macro fundamentals when feasible, either through political instability/conflict measures for crisis early, warning or through broader market/sentiment indicators for spread-based measures—while keeping the interpretation anchored in the economic differences between outcomes (Alaminos et al., 2021; Belly et al., 2022; Takawira, 2024).

3 Methodology

3.1 Data Description

The analysis is based on an annual country-year panel built by merging macroeconomic indicators from the IMF World Economic Outlook (WEO) with historical crisis events. The macro dataset is taken from `data/raw/IMF_WEO_dataset.csv`. In this file, each time series is identified by a `country_code` that contains the country code and the indicator code. In the data preparation step (`src/data_loader.py`), I split this code to recover `country_code` and `indicator`, keep a restricted set of indicators, reshape the data from wide year-columns into long format, and then pivot it back into a standard panel with one row per (`country_code`, `year`).

Crisis information comes from `data/raw/global_crisis_data.xlsx`. This file provides yearly binary indicators for several crisis categories. In the code, the relevant columns are renamed to consistent names and converted into numeric 0/1 variables (missing values are treated as zeros for the crisis flags). I keep the following crisis indicators in the cleaned panel: `external_default_1`, `external_default_2`, `domestic_default`, `currency_crisis`, and `inflation_crisis`. The main label used in this project is a sovereign debt crisis indicator, defined as

$$\begin{aligned}\texttt{sovereign\_crisis}_{i,t} = \mathbb{1}(&\texttt{external\_default\_1}_{i,t} = 1\\ &\text{or } \texttt{external\_default\_2}_{i,t} = 1\\ &\text{or } \texttt{domestic\_default}_{i,t} = 1)\end{aligned}$$

so it equals 1 whenever at least one of these default events occurs in year t for country i.

The final modelling dataset is obtained by an inner merge between the WEO macro panel and the crisis panel on (`country_code`, `year`). This is done in `build_and_save_panels()`, and the merged output is saved as `data/processed/merged_weo_crisis.csv`. After merging, the dataset contains 2527 country-year observations covering 69 countries over 36 years.

Regarding features, I focus on a small set of macro-financial indicators that are commonly discussed in the sovereign risk literature and also relatively available across countries. The retained WEO indicators are: government gross debt as % of GDP (`GGXWDG_NGDP`), interest payments as % of GDP (`GGXINT_NGDP`), net lending/borrowing of government as % of GDP (`GGXONLB_NGDP`), real GDP growth (`NGDP_RPCH`), inflation (`PCPIPCH`), unemployment rate (`LUR`), nominal GDP (`NGDPD`), population (`LP`), and the current account balance as % of GDP (`BCA_NGDPD`). In the modelling step, identifiers such as `country_code` and `year` are not used as predictors; the models learn only from the macro variables.

Finally, the project constructs forward-looking crisis targets to reflect an early-warning setting. For each country-year t, I created three horizon variables: `crisis_h3`, `crisis_h5`, and `crisis_h10`. Each target equals 1 if the country experiences at least one sovereign crisis in any of the following h years (i.e., $t+1,\dots,t+h$), and 0 otherwise. Technically, this is implemented by shifting `sovereign_crisis` forward within each country and taking the maximum across the future window.

A relevant data-quality issue is missing macroeconomic values, which is typical in cross-country datasets. The pipeline handles this with a simple two-step imputation. First, if a country has at least one observed value for a given variable, missing entries are filled with that country mean (country-mean imputation). Second, as a last-resort fallback, remaining missing values are filled with the global mean of the variable across the sample. This keeps most country-year observations available for training while avoiding dropping many rows due to incomplete macro series.

3.2 Approach

The project is set up as an early-warning *binary classification* problem. For each country-year observation (i, t), I observe a vector of macro-financial indicators $X_{i,t}$ and I aim to predict whether the country will enter a sovereign debt crisis within the next h years. In practice, I work with three horizons $h \in \{3, 5, 10\}$ and define the target as

$$y_{i,t}^{(h)} = \Big(\exists\, s \in \{t+1, \dots, t+h\} \text{ such that } \texttt{sovereign\_crisis}_{i,s} = 1\Big),$$

which corresponds to the variables `crisis_h3`, `crisis_h5`, and `crisis_h10` created in `src/data_loader.py`.

Data preprocessing

Before training the models, the code applies a few consistent preprocessing steps (see `src/models.py` and `src/data_loader.py`). First, all feature columns are forced to numeric format (non-numeric entries are coerced to missing). Columns that are entirely missing are dropped. Since missing values are common in cross-country macro data, the pipeline uses imputation: during panel construction, missing macro values are filled with country means when possible, and then remaining missing entries are filled with global means as a fallback. In the modelling stage, a simple mean imputer (`SimpleImputer(strategy="mean")`) is applied again after the train/test split, so training and test sets keep the same feature space and no rows are dropped due to remaining missing values. Identifier columns (`country_code`, `year`) and crisis helper columns are excluded from the feature matrix.

Train/test split and validation for thresholding

The main split is *chronological*, which is more realistic for prediction: the test set corresponds to the last $\approx 20\%$ of years in the sample (split by years, not by rows). For XGBoost, I also create a validation block (the years just before the test years) to choose a classification threshold. As a robustness check, I also run a *random* split with stratification (same test share), mainly to see if results are very sensitive to the split method.

Algorithms used

I compare three supervised models (implemented in `src/models.py`):

- **Logistic Regression (baseline).** This is the transparent benchmark. It estimates

$$\Pr\Big(y_{i,t}^{(h)} = 1 \mid X_{i,t}\Big) = \sigma\big(\beta_0 + X_{i,t}'\beta\big)\,,$$

  with $\sigma(\cdot)$ the logistic function. In the code I set `max_iter=2000` to avoid convergence issues.

- **Random Forest.** This is a tree-ensemble (bagging) model that can capture non-linearities and interactions. I use `n_estimators=300` and `class_weight="balanced"` to partly account for the fact that crises are rare.

- **XGBoost.** This is a gradient boosting tree model, often strong for tabular data. I use a standard configuration (`n_estimators=300`, `max_depth=6`, `learning_rate=0.1`, `subsample=0.8`, `colsample_bytree=0.8`) with objective `binary:logistic`. The model outputs probabilities and I convert them into class predictions using a chosen threshold.

Threshold choice (decision rule)

Because the policy cost of missing a crisis can be high, I do not rely only on the default 0.5 threshold. For XGBoost, the code selects a threshold on the validation set by scanning values in $[0.01, 0.99]$ and choosing the one that maximizes recall (equivalently, minimizes false negatives), with a tie-breaker that prefers a slightly higher threshold to reduce false positives. This makes the prediction rule more "early-warning" oriented, and it is reported together with the standard 0.5 threshold results.

Evaluation metrics

Model performance is evaluated out-of-sample using metrics adapted to imbalanced classification (see `src/evaluation.py`). I report accuracy, ROC-AUC, PR-AUC (average precision), precision, recall, F1 score, and balanced accuracy, together with confusion matrices. ROC curves are also plotted and saved to the results folders. Importantly, ROC-AUC and PR-AUC are computed from predicted probabilities (so they do not depend on the chosen threshold), while precision/recall and the confusion matrix depend directly on the threshold and therefore reflect the decision rule used.

3.3 Implementation

All scripts are implemented in **Python** and organized in a small, modular pipeline so that data construction, model training, and evaluation are separated and easy to reproduce. The main external libraries are `pandas` and `numpy` for data handling, `scikit-learn` for baseline models and metrics, `xgboost` for gradient boosting, `matplotlib` and `seaborn` for plots, and `joblib` to save trained models to disk. Paths are managed with `pathlib.Path`, so the code stays portable across machines and folders.

System architecture. The code base follows a simple structure with one entry point and three core modules:

- `main.py` is the entry script. It orchestrates the full workflow: build the processed panel, train the models for each horizon, evaluate results, save plots and model files, and finally run a small custom prediction block.
- `src/data_loader.py` contains the data engineering part: loading the raw WEO and crisis datasets, reshaping them into a country-year panel, imputing missing macro values, and creating the horizon targets.
- `src/models.py` contains the modelling utilities: feature selection, splitting functions (chronological or random), and training functions for Logistic Regression, Random Forest, and XGBoost (including a validation split for threshold selection).
- `src/evaluation.py` contains the evaluation routines: computation of classification metrics, ROC curves, confusion matrices, and the threshold-selection helper.

The raw inputs are stored under `data/raw/`, the merged panel is saved under `data/processed/`, and all outputs are written under `results/`. The results folder is split into `results/chrono/` (main chronological experiment) and `results/random/` (robustness with random split).

Key code components. A first important component is the panel builder `build_and_save_panels()` in `src/data_loader.py`. It (i) filters the WEO dataset to the selected indicators, (ii) reshapes it into a clean country-year panel, (iii) merges it with crisis events, and (iv) constructs forward-looking crisis targets `crisis_h3`, `crisis_h5`, and `crisis_h10` by looking ahead within each country.

```python
def build_and_save_panels(base_dir: Path) -> Tuple[pd.DataFrame, pd.DataFrame]:
    raw_dir = base_dir / "data" / "raw"
    processed_dir = base_dir / "data" / "processed"
    processed_dir.mkdir(parents=True, exist_ok=True)
(...)
```

Listing 1: Panel construction function (beggining)

A second component is the splitting logic in `src/models.py`. The main experiment uses a *chronological* split (last ≈ 20% of years as test set), which is closer to a real forecasting setup. For XGBoost, the code additionally creates a validation block right before the test period, used only to tune the decision threshold. As a robustness check, the pipeline also supports a *random* split with stratification to keep the crisis share similar in train and test.

```python
def split_train_test(
    X: pd.DataFrame,
    y: pd.Series,
    years: pd.Series,
    split_method: str = "chronological",
    test_share_years: float = 0.2,) -> tuple[pd.DataFrame, pd.DataFrame, pd.
   Series, pd.Series]:
(...)
    if split_method == "random":
        return train_test_split(X, y, test_size=0.2, random_state=SEED,
   stratify=y)

    if split_method != "chronological":
        raise ValueError(f"split_method must be 'chronological' or 'random',
   got {split_method}")
(...)
```

Listing 2: Chronological split method

A third component is model training and persistence. Each algorithm is trained via a dedicated function (`train_logistic_regression`, `train_random_forest`, `train_xgboost`, and `train_xgboost_with_val`). Trained models are saved with `joblib.dump()` into the corresponding results subfolders (for example `results/chrono/logit_h3/logit_model_h3.joblib`), so results are reproducible without retraining.

Finally, evaluation is handled in `src/evaluation.py`. The functions compute ROC-AUC and PR-AUC from predicted probabilities, and also threshold-dependent metrics (precision, recall, F1, balanced accuracy) plus confusion matrices. ROC curves and confusion-matrix figures are saved as `.png` files in the same results directories. For XGBoost, the threshold is not fixed at 0.5: the helper `choose_threshold_min_fn()` scans thresholds and picks one that reduces false negatives on the validation set, which fits an early-warning objective.

User-facing prediction block. In `main.py`, I also include a small custom scenario prediction section. It asks the user to input macro values, aligns them to the model feature order, and prints two outputs: an "optimistic" classification using the standard 0.5 threshold and a more "pessimistic" classification using the validation-tuned threshold. To make it readable, the script uses a variable label dictionary to show human-friendly names for the input variables.

4 Results

4.1 Experimental setup

All experiments are executed through `main.py`, which acts as a single entry point for data construction, model training, evaluation, and saving outputs. Reproducibility is enforced through a fixed random seed (`SEED=42` in `src/models.py`), which stabilizes the random split, the Random

Forest bootstrap, and the XGBoost stochastic components. The code creates a results directory at `results/` and organizes outputs into `results/chrono/` for the main time-respecting evaluation and `results/random/` for the robustness check.

The modelling dataset is the merged country-year panel built from IMF WEO macro-financial series and crisis indicators. A sovereign crisis indicator is constructed from crisis subcategories (external default and domestic default) and forward-looking horizon targets are created as future windows. For a given horizon h, the target is defined as

$$y_{i,t}^{(h)} = \nVdash\Big(\exists s \in \{t+1,\ldots,t+h\} \text{ s.t. } \texttt{sovereign\_crisis}_{i,s} = 1\Big),$$

which corresponds in the code to `crisis_h3`, `crisis_h5`, and `crisis_h10`. Predictors are a restricted set of macro-financial indicators available across many countries in WEO, while identifiers such as `country_code` and `year` are excluded from the feature matrix. Missing values are handled with mean-based imputation: values are coerced to numeric, columns that are entirely missing are dropped, and a mean imputer (`SimpleImputer(strategy="mean")`) is applied after splitting so train and test sets remain aligned in feature space.

The main evaluation uses a chronological split by year, implemented in `split_train_test(...)`: the test set is defined as the last $\approx 20\%$ of years (not the last 20% of rows). This design mimics forecasting and reduces time leakage. For XGBoost, the code additionally creates a validation window right before the test years using `split_train_val_test(...)` to tune the classification threshold in a way consistent with early-warning objectives.

Hyperparameters are fixed and intentionally simple. Logistic Regression is fitted with `max_iter=2000`. Random Forest uses `n_estimators=300` and `class_weight="balanced"`. XGBoost is trained with `n_estimators=300`, `max_depth=6`, `learning_rate=0.1`, `subsample=0.8`, and `colsample_bytree=0.8`.

4.2 Performance evaluation

To interpret the results, it is useful to separate probability ranking quality and threshold-based classification performance. Ranking quality is measured by ROC-AUC and PR-AUC (average precision). Threshold-based measures are derived from the confusion matrix counts: true positives (TP), false positives (FP), true negatives (TN), and false negatives (FN). Precision and recall are

$$\text{Precision} = \frac{\text{TP}}{\text{TP}+\text{FP}}, \qquad \text{Recall} = \frac{\text{TP}}{\text{TP}+\text{FN}},$$

and the F1 score is

$$\text{F1} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}.$$

Balanced accuracy averages sensitivity and specificity and is less dominated by the majority class than raw accuracy.

A key modelling choice is how the probability threshold is selected. Logistic Regression and Random Forest use the default sklearn decision rule (effectively a 0.5 cutoff). For XGBoost under the chronological evaluation, the threshold is tuned on the validation set using `choose_threshold_min_fn(...)`: thresholds from 0.01 to 0.99 are scanned and the one maximizing recall is selected (with ties resolved by choosing the highest threshold). This produces a conservative early-warning rule that reduces missed crises (FN), at the cost of more false alarms (FP).

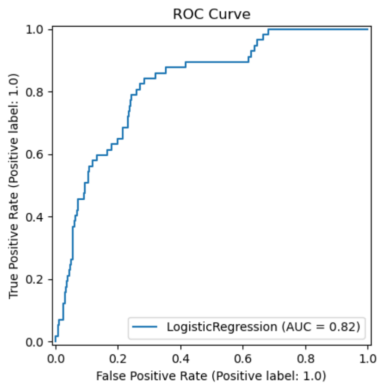
All metrics are written automatically into CSV files (`*_metrics.csv`) under `results/chrono/<model>_h<h>`
To avoid transcription errors, the table below is populated by reading these CSV outputs directly.

Table 1: Out-of-sample performance by model and horizon (chronological split).

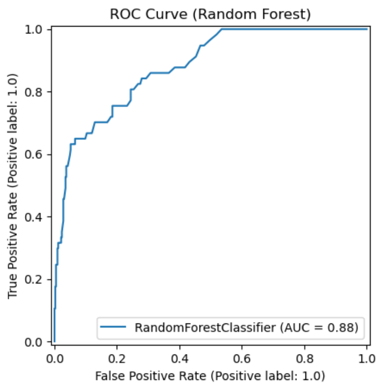
Horizon	Model	Thr	Acc	ROC-AUC	PR-AUC	Prec	Rec	F1	BalAcc
3	Logit	0.500	0.878	0.823	0.363	0.375	0.053	0.092	0.520
3	RF	0.500	0.903	0.879	0.603	0.857	0.211	0.338	0.603
3	XGB	0.010	0.760	0.799	0.453	0.275	0.632	0.383	0.704
5	Logit	0.500	0.870	0.814	0.367	0.400	0.066	0.113	0.526
5	RF	0.500	0.892	0.865	0.591	0.846	0.180	0.297	0.588
5	XGB	0.010	0.725	0.757	0.444	0.246	0.574	0.345	0.660
10	Logit	0.500	0.876	0.819	0.380	0.520	0.213	0.302	0.592
10	RF	0.500	0.907	0.881	0.648	0.900	0.295	0.444	0.645
10	XGB	0.010	0.706	0.765	0.453	0.235	0.590	0.336	0.656

4.3 Visualizations

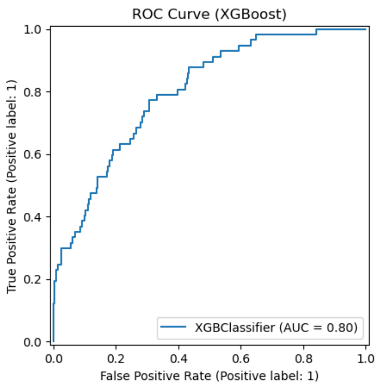
The results are complemented by visual diagnostics produced by the code. ROC curves compare ranking performance across models, while confusion matrices highlight the FP–FN trade-off at the chosen threshold. In addition, XGBoost is interpreted with SHAP values using `xgb_risk_drivers_report(...)`, which saves a bar plot of global importance (mean absolute SHAP values) and a SHAP beeswarm plot. These interpretability outputs are not causal, but they help to validate whether the learned patterns are economically plausible and stable across horizons.



(a) Logit (h=3)

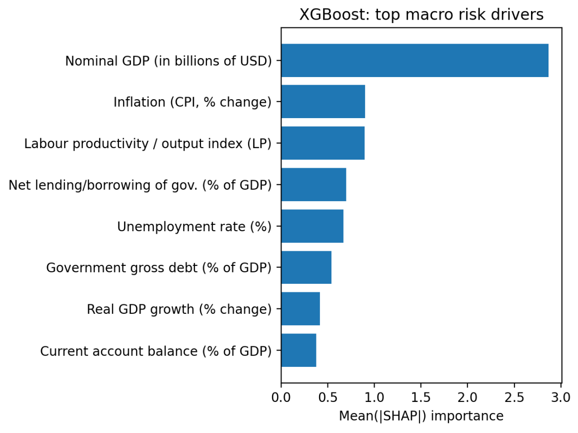


(b) RF (h=3)

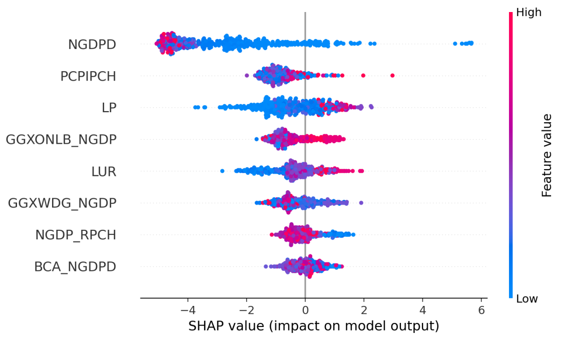


(c) XGB (h=3)

Figure 1: ROC curves for the within-3-years prediction task (chronological split).



(a) Top drivers (bar)



(b) Top drivers (beeswarm)

Figure 2: XGBoost interpretability output using SHAP values for horizon $h = 3$.

4.4 Robustness with random split (within 3 years)

As a robustness check, the project repeats estimation for the within-3-years target (`crisis_h3`) using a stratified random 80/20 split, with outputs saved under `results/random/`. Random splits can be more optimistic than chronological splits in macro panels because the train and test sets are more homogeneous in time and share similar distributions. For this reason, the chronological results remain the main benchmark, while the random split is interpreted as an upper-bound reference.

5 Discussion

The results highlight three main points about modelling sovereign crises with a restricted macro-financial information set. First, evaluation must be designed for rare events. When the crisis class is small, headline accuracy can be misleading because a model can predict "no crisis" almost everywhere and still obtain a high accuracy. For this reason, the report emphasizes PR-AUC, recall, balanced accuracy, and confusion matrices. In practice, comparisons based on ROC-AUC and PR-AUC are the most robust because they evaluate the quality of probability rankings over all possible thresholds.

Second, the chronological split imposes a stricter and more realistic out-of-sample test than random splits. The model is trained on earlier years and evaluated on later years, which exposes it to time variation in global conditions and reporting regimes. The random split robustness check is still useful, but it can be optimistic because train and test observations are drawn from similar time periods and therefore share similar distributions. This difference is economically relevant in macro prediction because structural breaks and regime changes are common.

Third, the probability threshold is not a neutral choice in an early-warning setting. By tuning the XGBoost threshold on a validation window to maximize recall, the project adopts a conservative decision rule that reduces missed crises (false negatives). This comes at the cost of more false alarms (false positives) and therefore lower precision. This trade-off is expected and should be interpreted as a policy preference: in a realistic implementation, one would ideally specify a loss function that assigns costs to missed crises and false alarms and then choose the threshold accordingly. The present implementation is a transparent approximation that encodes the idea that missed crises are more costly than false alarms.

Interpretability results suggest that some predictive power may come from slow-moving structural proxies such as GDP level. This is not surprising in cross-country crisis prediction, but it points to a limitation of a macro-only feature set. Political instability, governance quality, and conflict shocks can trigger financing stress even before macro fundamentals deteriorate strongly in annual data. In that sense, the model can be improved by enriching the information set, and it is also possible that part of the observed "GDP importance" is capturing omitted institutional factors.

6 Conclusion and Future Work

This project builds an annual country-year dataset by merging IMF WEO macro-financial indicators with sovereign crisis events and formulates crisis prediction as a supervised classification problem for horizons $h \in \{3, 5, 10\}$. A transparent baseline (logistic regression) is compared to two non-linear ensemble methods (Random Forest and XGBoost). The evaluation is performed out-of-sample with a chronological split by year, complemented by a random-split robustness check for the within-3-years horizon. Performance is summarized using both threshold-free metrics (ROC-AUC, PR-AUC) and threshold-dependent metrics (precision, recall, F1, balanced accuracy), together with ROC curves and confusion matrices saved directly by the code. For XGBoost, the classification threshold is tuned on a validation window to maximize recall, reflecting an early-warning preference that prioritizes avoiding missed crises.

Future work can extend the project without changing the core pipeline. A first step is to enrich predictors with political and institutional risk variables (regime durability, conflict, governance), which may capture triggers not visible in annual macro data and may reduce the extent to which GDP proxies for omitted factors. A second extension is probability calibration (for example isotonic regression or Platt scaling) so that predicted probabilities can be interpreted more directly. A third step is to formalize threshold selection with an explicit cost-sensitive criterion rather than targeting recall only. Finally, additional validation schemes such as rolling-window evaluation or country-based splits could be used to stress-test generalization across time and across country groups, which matters for any early-warning application.

References

1. Alminos, D., Peláez, J. l., Salas, M. B., & Fernández-Gámez, M. A. (2021). *Sovereign dept and currency crises prediction models using machine learning techniques.* Symmetry, 13(4), 652.

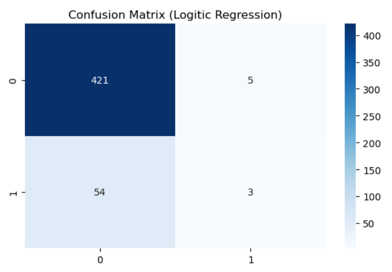
2. Belly, G., Boeckelmann, L., Caicedo Graciano, C. M., Di lorio, A., Isterfi, K., Siakoulis, V., & Stalla-Bourdillon, A. (2020). *Forecasting sovereign risk in the euro area via machine learning.* SSRN Working Paper.

3. Takawira, O. (2024). *Comparing the prediction performance of machine learning algorithms against the logistic regression model in forecasting sovereign dept ratings.* International Journal of Economics and Finance Studies, 16(01), 23-61.

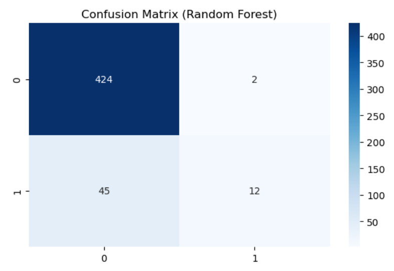
4. Harvard Business School, Behavioral Finance and Financial Stability. (n.d.). *Global Crises Data.* Available at: `https://www.hbs.edu/behavioral-finance-and-financial-stability/data/Pages/global.aspx`

5. International Monetary Fund. (n.d.). *World Economic Outlook (WEO) Database.* Available at: `https://www.imf.org/en/Publications/WEO/weo-database`

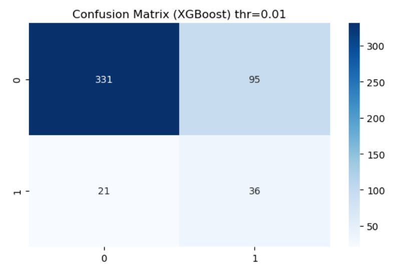
A Additional Figures



(a) Logit (h=3) Confusion matrix

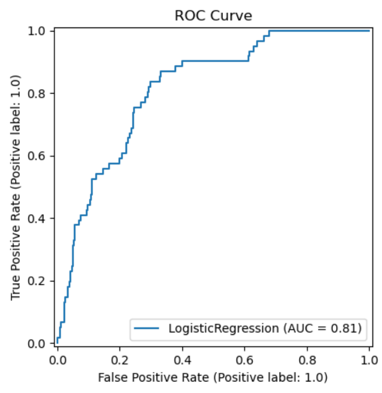


(b) Random Forest (h=3) Confusion matrix

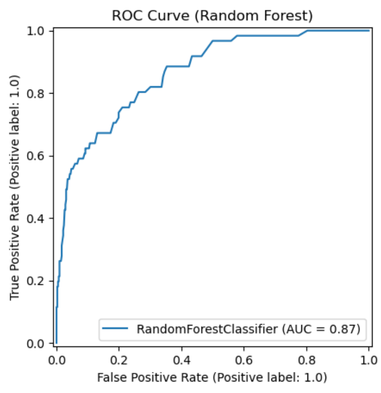


(c) XGBoost (h=3) Confusion matrix

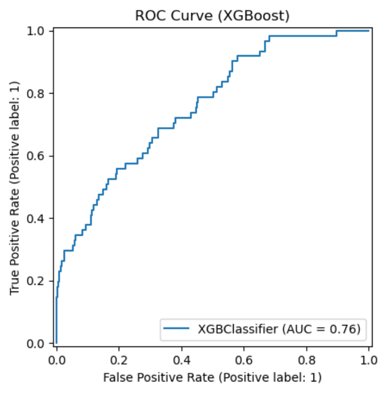
Figure 3: Confusion matrices for the chronological split (h=3).



(a) Logit (h=5) ROC

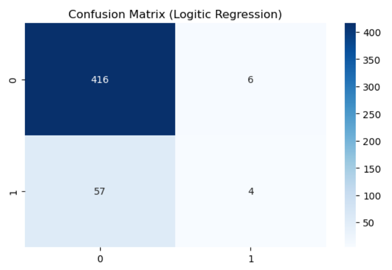


(b) Random Forest (h=5) ROC

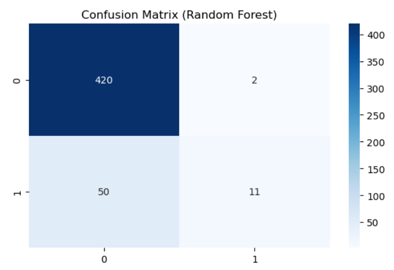


(c) XGBoost (h=5) ROC

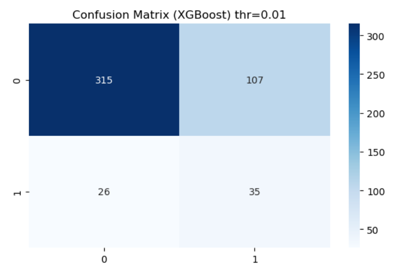
Figure 4: ROC curves for the chronological split (h=5).



(a) Logit (h=5) Confusion matrix

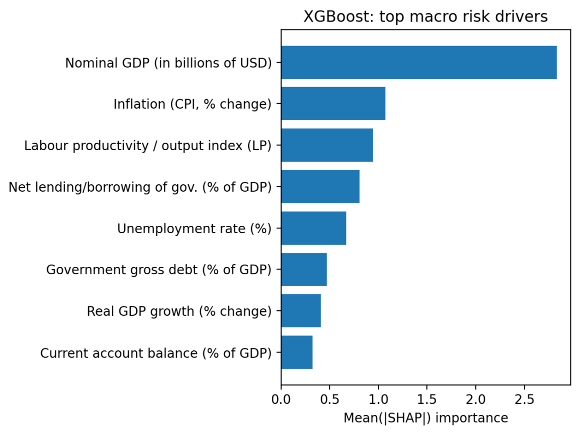


(b) Random Forest (h=5) Confusion matrix

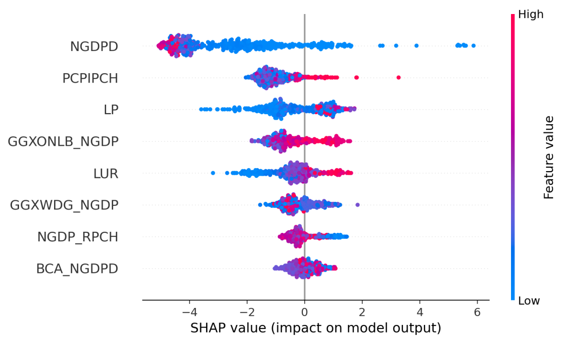


(c) XGBoost (h=5) Confusion matrix

Figure 5: Confusion matrices for the chronological split (h=5).

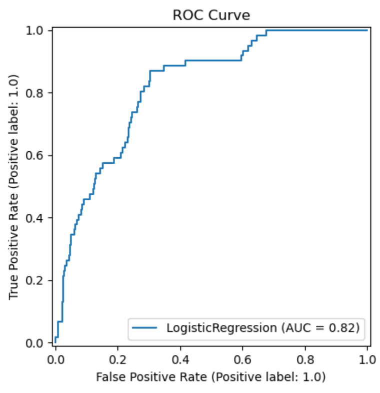


(a) XGBoost (h=5) Top drivers (bar)

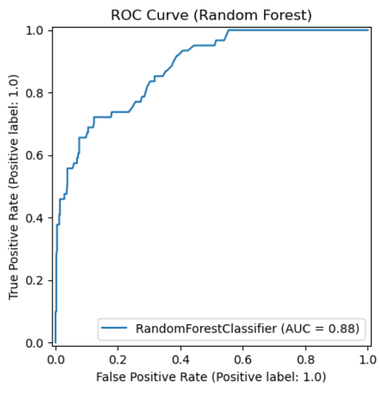


(b) XGBoost (h=5) Top drivers (beeswarm)

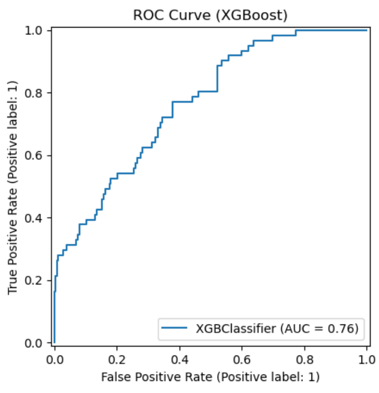
Figure 6: XGBoost risk-driver visualisations for the chronological split (h=5).



(a) Logit (h=10) ROC

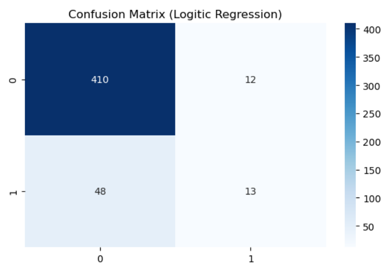


(b) Random Forest (h=10) ROC

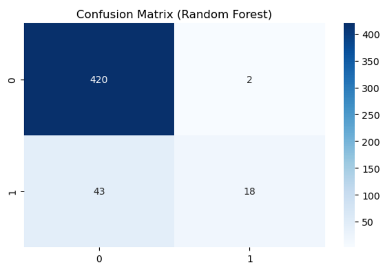


(c) XGBoost (h=10) ROC

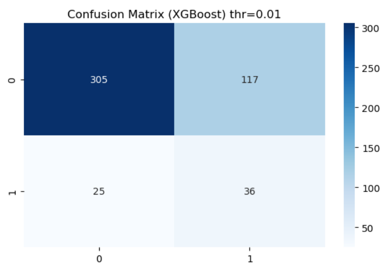
Figure 7: ROC curves for the chronological split (h=10).



(a) Logit (h=10) Confusion matrix

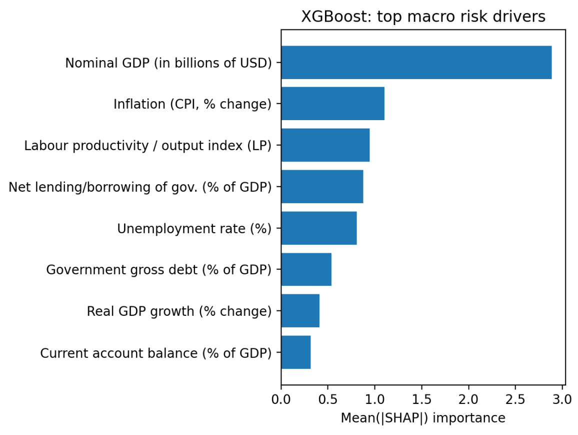


(b) Random Forest (h=10) Confusion matrix

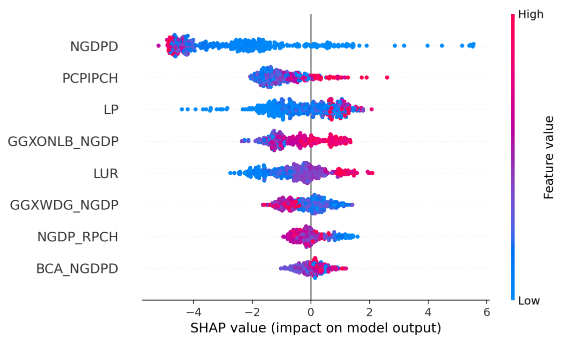


(c) XGBoost (h=10) Confusion matrix

Figure 8: Confusion matrices for the chronological split (h=10).



(a) XGBoost (h=10) Top drivers (bar)



(b) XGBoost (h=10) Top drivers (beeswarm)

Figure 9: XGBoost risk-driver visualisations for the chronological split (h=10).

B Code Repository

GitHub Repository: `https://github.com/Yanik08/Dept_Health_Analysis.git`

B.1 Repository Structure

The repository is organised to follow a clear and reproducible machine-learning pipeline. The `data/` folder contains the raw and processed datasets. The `src/` folder includes all core Python scripts, separated by functionality: data loading and preprocessing, model training, and model evaluation. The `results/` folder stores model outputs, performance metrics, and figures generated during the analysis. The main execution logic is handled by `main.py`, while `environment.yml` specifies the required software environment to ensure reproducibility.

B.2 Installation Instructions

To run the project, the repository must first be cloned locally. The required Python environment can be created using Conda with the provided `environment.yml` file. After creating and activating the environment, all dependencies needed for data processing, model training, and evaluation are installed automatically. The project is fully implemented in Python and does not require additional configuration beyond the Conda environment.

B.3 How to Reproduce the Results

All results can be reproduced by running the main script. Executing `main.py` automatically constructs the dataset, trains all models, and evaluates their performance using predefined settings. Trained models, evaluation metrics, and figures are saved in the `results/` directory. Since the analysis relies on chronological data splits and fixed random seeds where applicable, the results are fully reproducible.