Irving Fisher Committee on Central Bank Statistics

BIS

IFC Satellite Seminar on "Sustainability data issues and central banks' experience"

4 October 2025

Improving the timeliness of the ESCB carbon indicators with nowcasting techniques

Trond Husby, Michiel Nijhuis and Milan Karsten,
De Nederlandsche Bank,

Ignacio Felez de Torres,
Bank of Spain

Predicting the Present with the Python Package Oraculo

Imagine you're analyzing portfolio climate risk and need current-year company emissions data, but annual reports won't arrive for another 6-12 months. Or you're tracking unemployment trends and need this month's figures, but the official statistics won't be published for weeks. These data lags can severely hamper the timeliness of your analytics and decision-making.

The solution? **Nowcasting** - using proxy variables that are available in real-time to predict data that exists but hasn't been published yet. Essentially, you're predicting the present (Giannone et al., 2008).

In this blog post, we present **Oraculo**, a Python package, developed to make nowcasting straightforward and accessible. Whether you're forecasting economic indicators, emissions data, or any time-lagged series, Oraculo provides a unified framework for configuration, modeling, and evaluation.

- Note: Since a nowcast is technically a type of forecast, these terms are used interchangeably throughout.

What You'll Learn

In this post, we'll walk through:

1. **Configuration-driven modeling** - How YAML files define your entire nowcasting pipeline, making it easy to experiment with different model specifications
2. **Rapid experimentation** - How Oraculo's design facilitates exploratory modeling by providing instant feedback through error metrics and diagnostic visualizations
3. **Model evaluation** - Using RMAE and RME metrics to assess forecast accuracy
4. **Visual diagnostics** - Identifying error patterns through histograms and plots
5. **Feature importance** - Understanding which proxy variables drive predictions
6. **Benchmark comparison** - Why LOCF is the key baseline to beat
7. **Advanced techniques** - Normalization, lagged features, and hybrid modeling strategies

By the end, you'll see how Oraculo's exploratory framework enables systematic iteration - taking a basic model with 153% error and improving it to 37% through rapid experimentation.

Background: Climate Indicators at the ECB

Since 2020, the European Central Bank has been developing climate and nature-related statistical indicators to support its climate-related financial stability monitoring and monetary policy assessment (ECB, 2021; ECB, 2024). These experimental indicators track the carbon footprint of euro area financial institutions' portfolios, the climate transition risks in the banking sector, and the environmental impact of corporate bond holdings. The ECB publishes these indicators regularly - with recent releases in 2024 providing comprehensive data on greenhouse gas emissions associated with euro area financial sector exposures.

However, a critical challenge undermines the timeliness of these indicators: **company-level emissions data suffers from publication lags of often more than 12 months** (Busch et al., 2022). When the ECB publishes climate indicators in Q1 2026, the underlying emissions data typically reflects 2024 or even 2023 activity. This lag creates a significant blind spot for policymakers and financial supervisors who need to assess current climate risks, not historical ones. For monetary policy decisions and financial stability assessments that require real-time insights, year-old emissions data can miss emerging trends, sudden transitions, or the impact of recent policy interventions.

Oraculo addresses this timeliness gap by combining lagged emissions data with more timely proxy variables - economic indicators, energy consumption patterns, and sector trends - to generate nowcasts that fill the gap until official data becomes available. This approach enables the ECB to maintain more current climate risk assessments and respond more quickly to evolving environmental challenges in the financial system.

A Practical Example: Nowcasting Scope 2 Emissions

To illustrate Oraculo's capabilities, let's nowcast company-level Scope 2 emissions (indirect emissions from purchased electricity). We'll use:

- **Target variable**: Scope 2 emissions from ISS (a data provider that collects emissions from annual reports)
- **Proxy features**:
  - Aggregate GDP data from the World Bank (World Bank, 2024)
  - Oil consumption data from the U.S. Energy Information Administration (EIA, 2024)
  - Historical emissions patterns (lagged values)

The intuition is straightforward: emissions correlate with economic activity and energy consumption, both of which are available with shorter lags than company-reported emissions data.

Configuration-Driven Modeling

Oraculo uses YAML configuration files to define your entire nowcasting pipeline. This design philosophy makes experiments reproducible and facilitates rapid iteration - you can test different model specifications, feature combinations, and data transformations simply by editing a configuration file. No need to rewrite code or manage complex parameter dictionaries.

- *Key advantage for exploratory modeling*: Change your model type, add features, or adjust transformations in seconds, then immediately see the impact on forecast accuracy through Oraculo's built-in evaluation metrics and visualizations.

Let's start with a basic example.

```
name: "Oraculo crystal ball"
series_keys:
```

```
  - "issuerid"
date_key: "date"
forecast_interval:
  years: 1

model_configs:
  - name: "XGBoost Model Scope 2"
    model: "LGBM"  # LightGBM gradient boosting
    target: "scope2"
    train_start: 2018-01-01
    features:
      - oil_consumption
      - gdp

data_config:
  repository: "ISSRepository"
  path: path/to/repositories/iss_repository.py
  refresh: false
  merge:
    repository: "WorldbankRepository"
    path: path/to/repositories/worldbank_repository.py
    refresh: false
    join_keys: ["country", "date"]
    merge:
      repository: "EIARepository"
      path: path/to/repositories/eia_repository.py
      refresh: false
      join_keys: ["country", "date"]
```

Key components:

- **General settings**: Define the series identifier (`issuerid`), date column, and forecast horizon (1 year ahead)
- **Model configuration**: Specify the model type (LightGBM), target variable, training start date, and features
- **Data configuration**: Oraculo's unified data framework handles importing and merging from multiple sources. Each repository is a Python class that knows how to fetch and format data from a specific source (ISS, World Bank, EIA). The nested `merge` structure works sequentially: *start with ISS emissions data → merge in World Bank GDP by country/date → merge in EIA oil consumption by country/date*. This creates a single unified dataset with all features aligned. Note: you need to create repository classes for your data sources.

Running Your First Nowcast

With the configuration in place, running a model is straightforward. For training and evaluation, we'll create a forecast for a historical date (2023-01-01) where we have actual observations to compare against:

```
import datetime as dt
from oraculo.services.model_run_service import ModelRunService
from oraculo.config import config

# Load configuration
fc_basic = config.ForecastConfig.from_yaml(
    "./configs/blog_post_config_basic.yaml",
    "./repositories/template_credentials.yaml"
)

# Create model instance
mrs = ModelRunService()
fd = dt.date(2023, 1, 1)

# # Run model using forecast date and configuration
mrs.run_model(fd, fc_basic)
```

Evaluating Model Performance

Oraculo provides **instant feedback** on model performance to facilitate rapid iteration - a core feature for exploratory modeling. As soon as your model runs, you get immediate diagnostic information without writing additional evaluation code. This tight feedback loop enables you to quickly test hypotheses, identify issues, and refine your approach.

Let's check our basic model's accuracy using two key metrics:

- **Relative Mean Absolute Error (RMAE)**: Average absolute error as a percentage of actual values
- **Relative Mean Error (RME)**: Average error as a percentage (reveals bias)

```
# Create evaluation instance
evr_basic = mrs.return_evaluation_results()
evr_basic.summarise('scope2')
```

The results suggest our basic model isn't performing well:

- **RMAE = 153%**: On average, predictions are off by 153% of actual values
- **RME = +126%**: Indicates severe systematic overestimation

These metrics tell us we need to dig deeper to understand what's going wrong.

Visual Diagnostics: Understanding the Errors

For visual inspection of forecast error distributions, Oraculo provides built-in plotting functions:

```
evr_basic.plot_histogram()
```

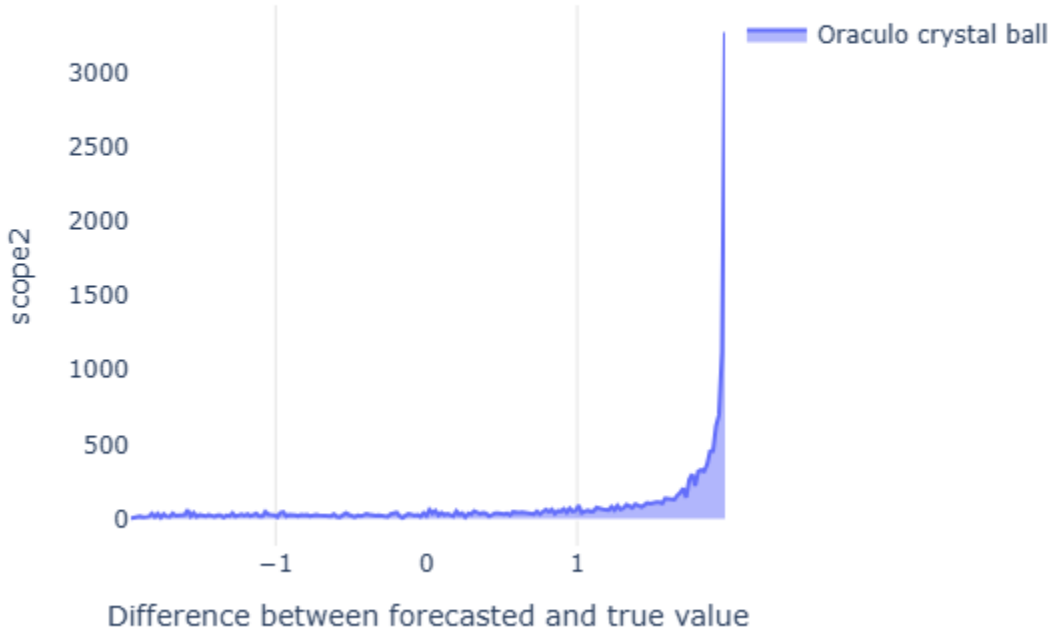


Figure 1: histogram of forecasting errors

The histogram visualization helps reveal the source of our positive bias: errors are concentrated in firms with large Scope 2 emissions. The model struggles with high-emitting companies, likely because:

1. These firms are outliers in the distribution
2. The relationship between features and emissions may be non-linear at high values
3. We haven't normalized the data to handle scale differences

This diagnostic insight will guide our improvements in the next section.

Feature Importance Analysis

For tree-based models, Oraculo can calculate permutation importance scores to understand which features drive predictions:

```
# Check feature importance
mdl = fc_basic.model_configs[0].name
evr_basic.model_permutation_importance(mdl)
```

The permutation importance results reveal interesting insights:

- **Oil consumption** (1.62) and **GDP** (1.58) have nearly equal importance
- Both scores > 1.0 indicate that permuting these features degrades model performance - they're genuinely contributing to predictions
- The similar importance suggests both economic activity and energy consumption are valuable signals for emissions nowcasting
- However, with only two features and no historical emissions data, the model lacks the persistence information that LOCF captures naturally

This analysis suggests we need to add lagged emissions as features to capture temporal patterns.

The LOCF Benchmark: A Reality Check

Last Observation Carried Forward (LOCF) is the most relevant — and often hardest to beat — benchmark for nowcasting. LOCF simply uses the most recent observed value as the prediction. It's naïve but surprisingly effective, especially for slowly changing series.

However, LOCF has important limitations that make it an imperfect solution (Makridakis et al., 2018):

- **Unrealistic assumption**: It assumes values remain exactly constant - emissions are identical to last year's, unemployment is unchanged from last month. This rarely reflects reality.
- **Trend bias**: When data exhibits trends (e.g., declining emissions due to decarbonization efforts, or rising unemployment during recessions), LOCF systematically lags behind, producing biased forecasts.
- **No forward-looking information**: LOCF ignores all available proxy variables that might signal changes (economic indicators, policy shifts, sector trends).
- **Poor performance during transitions**: During periods of structural change or volatility, LOCF fails to anticipate shifts that more sophisticated models might detect.

For these reasons, **while LOCF provides a crucial benchmark, the goal is to develop models that outperform it** by incorporating forward-looking information and capturing dynamics that simple persistence misses. This is especially important for policy and risk assessment applications where understanding current trends matters more than historical values.

Oraculo makes it easy to compare your model against LOCF:

```
mrs.compare_to_LOCF('scope2', fd, fc_basic)
```

The comparison reveals a stark difference in performance:

- **Basic model**: RMAE = 153%, RME = +126%

- **LOCF benchmark**: RMAE = 33%, RME = +8%
- *LOCF dramatically outperforms our basic model** by nearly 5x on RMAE. This is disappointing but informative - it tells us that our current feature set (just GDP and oil consumption) isn't capturing enough signal to beat the simple persistence baseline.

This is a common challenge in nowcasting: you need to add genuine predictive value beyond simple persistence. The key insight is that emissions tend to be sticky - last year's value is often a strong predictor of this year's value. Let's see if we can do better by incorporating this persistence information.

Leveraging Advanced Features of Oraculo

Our basic model underperforms LOCF, but this is where Oraculo's exploratory modeling capabilities shine. The framework makes it easy to systematically test improvements:

Improvements to try:

- **Switch model type**: Try OLS (Ordinary Least Squares) for better interpretability
- **Normalize variables**: Handle scale differences between features and target
- **Add lagged target values**: Include historical emissions as features (captures persistence)
- **Create first-differences**: Model changes rather than levels
- **Split the sample**: Use the model for stable companies (±50% year-over-year change), LOCF for volatile ones

With Oraculo's configuration-driven approach, each of these experiments requires only minor YAML edits - no code changes needed. You can rapidly iterate through different specifications and immediately see the impact on forecast accuracy.

This hybrid approach recognizes that different prediction strategies work better for different types of companies.

Advanced Configuration

Here's how to implement these improvements in the configuration file. **Save this as `blog_post_config.yaml`:**

```
name: "Oraculo improved model"
series_keys:
  - "issuerid"
date_key: "date"
forecast_interval:
  years: 1

model_configs:
  - name: "OLS Model Scope 2"
    model: "OLS"
    target: "scope2"
    train_start: 2018-01-01
    features:
```

```
      - oil_consumption
      - scope2_lag1
      - gdp
    split_filter:
      column: ["scope2_lag1", "scope2_lag1"]
      value: [0.5, 1.5]
      operator: [">", "<"]

  - name: "LOCF Model Scope 2"
    model: "LOCF"
    target: "scope2"
    train_start: 2021-01-01
    split_filter:
      column: [""]
      value: [null]
      operator: ["otherwise"]

data_config:
  repository: "ISSRepository"
  path: path/to/repositories/iss_repository.py
  refresh: false
  merge:
    repository: "WorldbankRepository"
    path: path/to/repositories/worldbank_repository.py
    refresh: false
    join_keys: ["country", "date"]
    merge:
      repository: "EIARepository"
      path: path/to/repositories/eia_repository.py
      refresh: false
      join_keys: ["country", "date"]

data_transformations:
  normalize_dict:
    scope2: "max"
    gdp: "first"
    oil_consumption: "mean"
  target_diff:
    - "scope2"
  simple_lags:
    scope2: [1, 2]
  differences:
    scope2: [[1, 2]]
```

What's new:

- **Two models**: OLS for stable companies, LOCF for volatile ones
- **Split filter**: Routes companies to different models based on year-over-year stability
- **Data transformations**: Normalizes variables, creates lags and differences
- **Lagged features**: `scope2_lag1` captures historical patterns

```
# Load improved configuration
fc = config.ForecastConfig.from_yaml(
    "./configs/blog_post_config.yaml",
    "./repositories/template_credentials.yaml"
)

# Run model using forecast date and configuration
mrs.run_model(fd, fc)

# Create evaluation instance
evr = mrs.return_evaluation_results()

# Plot mean prediction
evr.plot_mean_results('scope2', normalise=False)
```

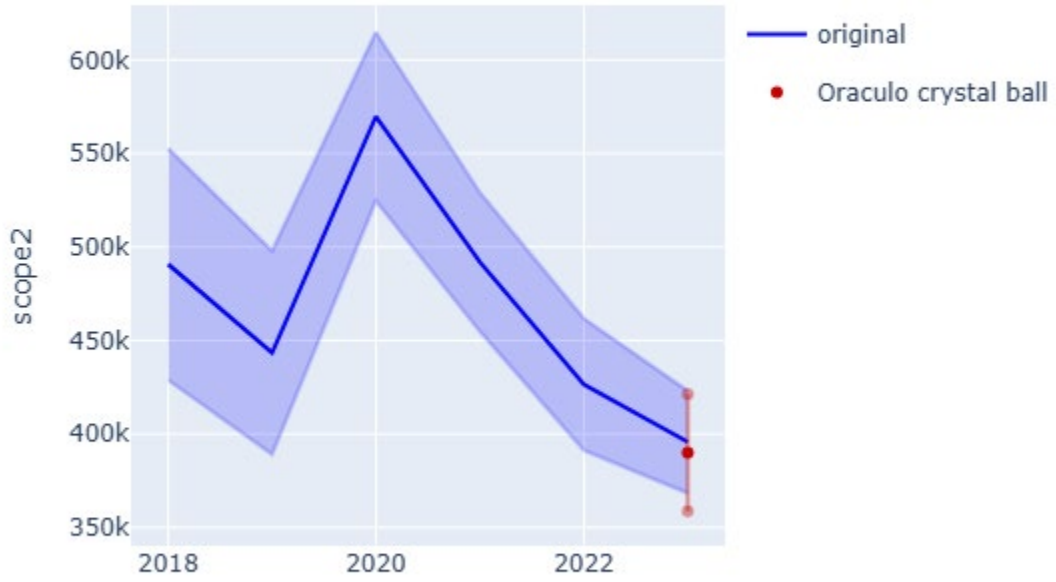


Figure 2: predicted and actual mean emissions +/- 2 standard errors

Results: Significant Improvement

The improvements made a substantial difference! Let's compare the metrics:

Metric	Basic Model	LOCF Baseline	Improved Model	Improvement vs Basic
RMAE	153%	33%	**37%**	**76% reduction**
RME	+126%	+8%	**-2%**	Nearly unbiased
Coverage	37,146 rows	17,589 rows	17,381 rows	Focused on stable firms

Key takeaways:

- The improved model achieves **37% RMAE** - slightly better than LOCF's 33% and dramatically better than the basic model's 153%
- **Bias nearly eliminated**: RME went from +126% overestimation to just -2% (slight underestimation)
- The hybrid approach works: by routing volatile companies to LOCF and stable companies to the OLS model, we leverage each method's strengths
- Normalization handled the scale issues with large emitters
- Adding lagged emissions captured the crucial persistence patterns that the basic model missed

The plot shows that predicted mean emissions closely track actual values, with the model capturing both the overall trend and year-over-year variations.

Conclusion

Oraculo demonstrates how configuration-driven nowcasting can transform lagged data into timely insights. The package's design philosophy centers on facilitating exploratory modeling through rapid experimentation:

Key features for exploratory modeling:

- **Configuration-driven**: Test different models, features, and transformations by editing YAML files - no code changes required
- **Instant feedback**: Immediate error metrics (RMAE, RME) and diagnostic plots after each model run
- **Visual diagnostics**: Built-in histograms, time series plots, and feature importance analysis
- **Easy iteration**: Tight feedback loops enable systematic hypothesis testing and refinement
- **Unified data framework**: Seamlessly merge multiple data sources
- **Flexible modeling**: Support for multiple model types (LGBM, OLS, LOCF, etc.)
- **Built-in transformations**: Normalization, lags, differences, and more
- **Hybrid strategies**: Route different subsets to different models

When to use Oraculo:

- You have time-lagged target variables
- Proxy variables are available with shorter lags
- You need to experiment with multiple modeling approaches
- You want reproducible, configuration-driven workflows
- You value rapid iteration and immediate feedback

Oraculo is not yet published as an open source package. If you are interested in using it, please get in touch with one of the authors.

While most forecasting focuses on predicting the future, nowcasting reminds us that sometimes the most valuable predictions are about the present we can't yet observe.

References

Nowcasting Methodology:

- Giannone, D., Reichlin, L., & Small, D. (2008). Nowcasting: The real-time informational content of macroeconomic data. *Journal of Monetary Economics*, 55(4), 665-676. https://doi.org/10.1016/j.jmoneco.2008.05.010

Forecasting Methods and Benchmarks:

- Makridakis, S., Spiliotis, E., & Assimakopoulos, V. (2018). Statistical and Machine Learning forecasting methods: Concerns and ways forward. *PLOS ONE*, 13(3), e0194889. https://doi.org/10.1371/journal.pone.0194889

ECB Climate Indicators:

- European Central Bank (2021). *ECB economy-wide climate stress test: Methodology and results*. Occasional Paper Series No. 281. https://www.ecb.europa.eu/pub/pdf/scpops/ecb.op281~05a7735b1c.en.pdf
- European Central Bank (2024). *Climate-related statistical indicators*. https://www.ecb.europa.eu/stats/ecb_statistics/climate/html/index.en.html

Emissions Data Timeliness:

- Busch, T., Johnson, M., & Pioch, T. (2022). Corporate carbon performance data: Quo vadis? *Journal of Industrial Ecology*, 26(1), 350-363. https://doi.org/10.1111/jiec.13008

Data Sources:

- ISS Corporate Solutions (2024). *ESG Data & Ratings*. https://www.issgovernance.com/esg/
- U.S. Energy Information Administration (2024). *International Energy Statistics*. https://www.eia.gov/international/data/world
- World Bank (2024). *World Development Indicators*. https://databank.worldbank.org/source/world-development-indicators

Improving the timeliness of the ESCB carbon indicators with nowcasting techniques

DeNederlandscheBank
EUROSYSTEEM

Husby, T.G. (Trond) (STAT_EDB)

ESCB context

- In line with Governing Council action plan and new roadmap, ECB, DNB and ESCB (all other Eurosystem NCBs) committed to produce **euro area statistical indicators**
- **DNB** co-chair of Expert Group on Climate Change and Statistics (EG CCS)
- Where possible, data is sourced from **public/ESCB data collections**
- The new experimental and analytical (= 'more experimental') indicators, like all similar data, come with **significant limitations**, to be used and analyzed with care!
- The second release has been **published April 2024** on the ECB and DNB website and is **expected to be updated in 2025Q4**

Green bonds

Carbon indicators

Physical risk

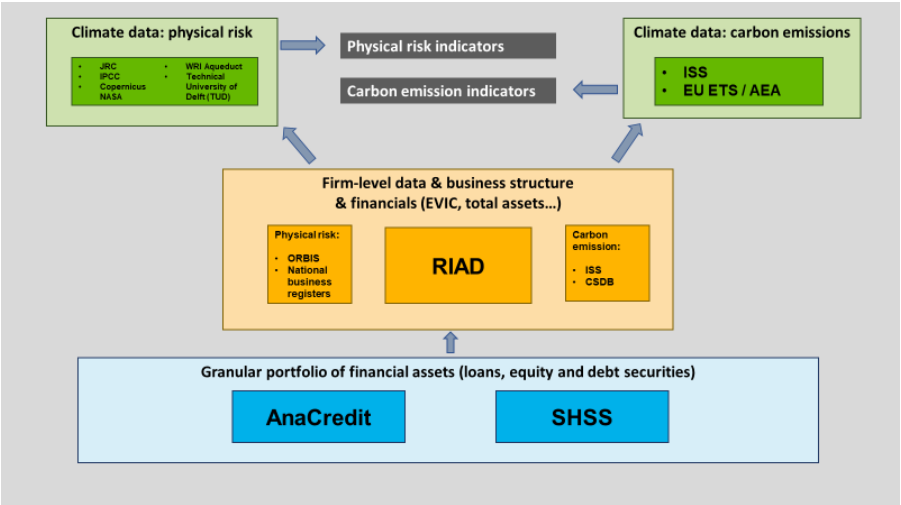
Four carbon indicators and their underlying data sources

Financed emissions | Carbon intensity | Weighted average carbon intensity | Carbon footprint

These four indicators show the amount (and share) of carbon emissions that can be attributed to financial institutions via their securities and loan portfolios.

→ Provide insights into the financial sector's monetary contribution to high-emitting economic activities, either as a ratio of carbon over revenue or the associated exposure to transition risks/financed absolute carbon emissions.

In words:
"If your portfolio holds 10% of a company's total value, you are responsible for 10% of its total (financed) emissions. This is summed for all companies in a portfolio to obtain the Financed Emissions indicator."

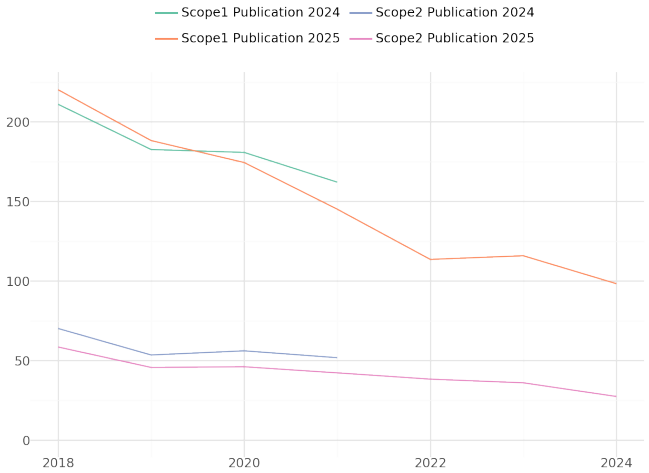


- **Carbon emission indicators** combine corporate securities and loan portfolios of financial institutions with firm-level data and climate data to calculate indicators at the aggregate level. Micro data are also provided to ESCB internal users for carbon and PR.
- **Physical risk (PR) indicators** capture financial system exposures to companies located in areas susceptible to natural disasters (such as flooding, windstorms, wildfires or droughts) and chronic physical risks (heat and water stress).

Notes: AnaCredit: Analytical credit datasets. SHSS: Securities Holdings Statistics by Sector. RIAD: Register of Institutions and Affiliates Data. ISS is a commercial data provider offering carbon emission information at company level. CSDB: Centralised Securities Database. EU ETS denotes the European Emission Trading System and AEA the Eurostat Air Emissions Accounts. JRC: Joint Research Centre. IPCC: Intergovernmental Panel on Climate Change. WRI: World Research Institute. EVIC: Enterprise value including cash

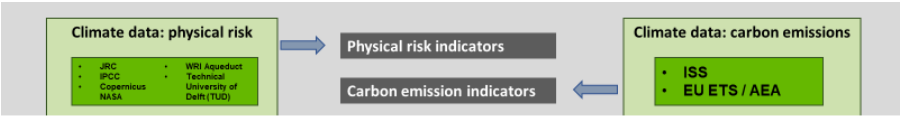
Timeliness of 2024 publication was hampered by availability of counterparty data

Weighted average carbon intensity



Source: ECB/EG CCS. The results of the 2025 publication are preliminary and only used to indicate the improvement in timeliness

Zooming in on data on emission and carbon intensity (emissions/revenue) of counterparties from ISS



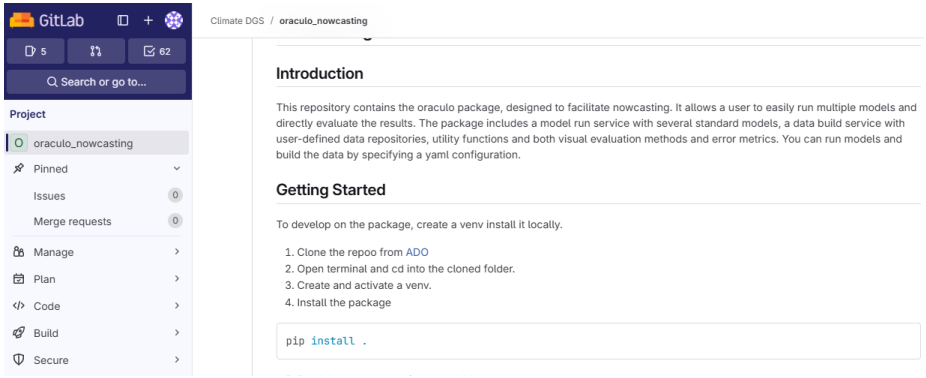
- At the time of the 2024 publication of the indicators the most recent counterparty data covered the bookkeeping year 2021
- However, timely data from other sources, such as those on the portfolios of financial assets, were available
- **Improving timeliness a strong user request!**
- For the current (2025) publication, EG CCS members teamed up with the DNB Data Science Hub to develop a python package for **nowcasting**: *predicting the present or the very near future of a variable or system, typically using real-time data*.
- The package was used to predict granular level *emissions* and *financial information* of ISS counterparties, allowing the inclusion of preliminary data points of the year 2024 the 2025 publication of the indicators

ORACULO: a python Package for nowcasting

ORACULO:

Package for nowcasting & forecasting of short timeseries

- Application of various models; from simple heuristics to machine learning
- Unified method of importing data
- Plotting and evaluation of forecast results
- Logging of model runs

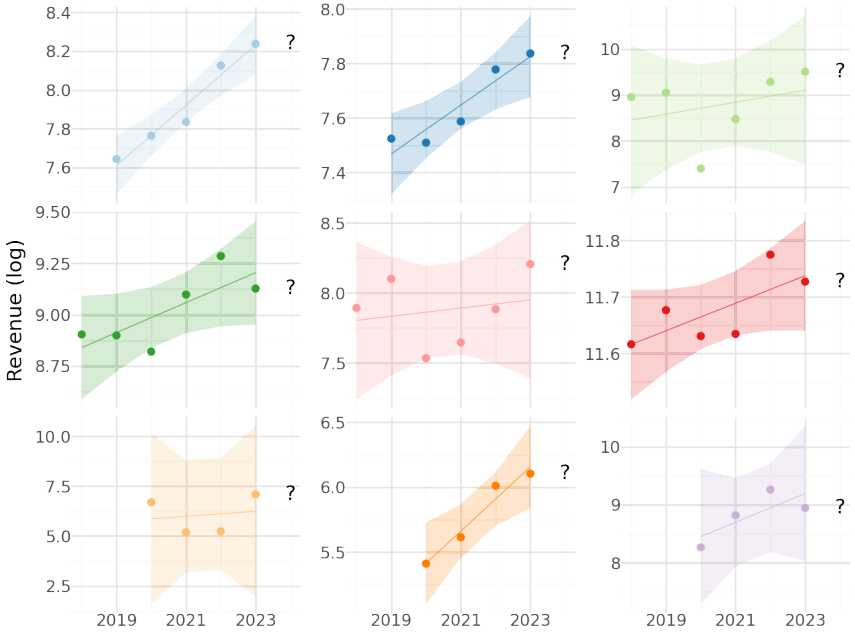


DataScience Hub

DeNederlandscheBank
EUROSYSTEEM

BANCO DE ESPAÑA
Eurosistema

Producing a better prediction than last observation carried forward



- Last observation carried forward (LOCF) give very accurate, but potentially biased, predictions
- General strategy: use LOCF as benchmark but also main method for entities with short time series or missing values

- The LHS figure shows log revenue (log) along with a trend line for 9 randomly selected issuerids from the ISS dataset
- In many cases there is an upwards trend in revenue, for example due to inflation
- For some entities we can improve predictions using additional features
- Credit rating agencies publish ratings using the latest available information. Ratings from several agencies have recently been added to CSDB
- The credit ratings can help us nudge the predictions in the right direction

Nowcast of revenue

Prediction model:

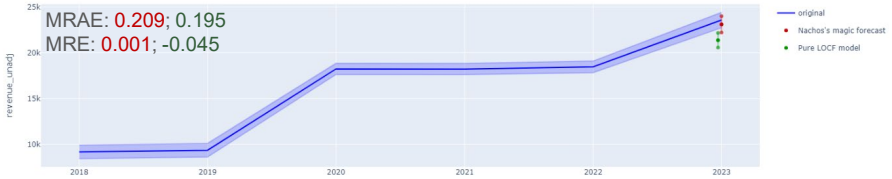
$$\widehat{\text{revenue}}_{i,t} = f\left(\text{revenue}_{i,t-1}, \Delta\text{credit rating}_{i,t}\right)$$

- Features: lag of revenue and change in credit rating
- Credit ratings are average of 5 ratings agencies, and average over a year. Rationale: agencies may have early information about the development of revenue
- Model fitted with XGBoost, results with random forest are similar
- Predictions for companies with credit ratings (about 1/3), LOCF used for companies without credit ratings data

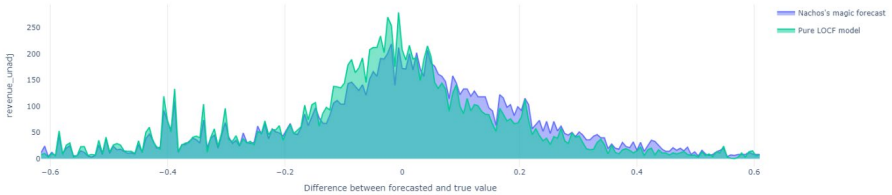
- Results show
  - Model with credit ratings reduces bias: LOCF of revenue ignores upwards trend due to inflation
  - The reduction of bias impacts the trajectory of the indicators
  - Predicted trend in scope1 WACI is more in line with expected values than that of LOCF

Main take away: Credit ratings improve the nowcast of revenue by reducing bias.

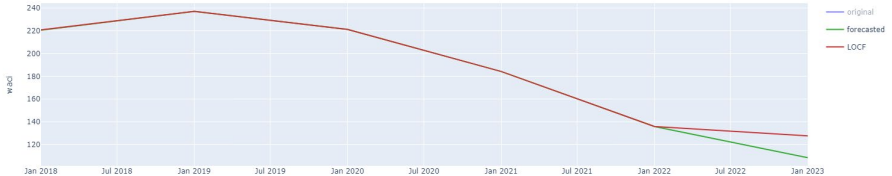
Forecast year 2023



Distribution of prediction errors



Scope1 WACI EA S122



Nowcast of scope1 and scope2 emissions

Prediction model:

$$\Delta\widehat{\text{emissions}}_{i,t}^{scope} = f\left(\text{emissions}_{i,t-1}^{scope}, \text{oil consumption}_{country,t}, \text{EVIC}_{i,t}\right)$$

- Predict *change* in emissions (scope1/2) for company *i*
- Features: lag emissions; oil consumption in country of HQ; EVIC
- Estimated using OLS, using data from 2020 with filter for extreme (50%) year-on-year changes, LOCF for the rest

- Results show
  - Predictions in 2023 are in general more accurate than those in 2022
  - The prediction model and LOCF perform similar in terms of formal error metrics (relative mean absolute error, relative mean error)
  - The figures show that model predicts the mean(s) of scope1/2 better than LOCF

Main take-away: Prediction accuracy of scope1 are comparable to LOCF, predictions of scope2 are more accurate than LOCF.

*Figures show mean of predictions plus/minus 2*SE of the mean*

Questions?

Thank you for listening!

Feedback? Interested in using, or contributing to, the package? Please contact t.g.husby@dnb.nl

Oraculo team: Trond Husby, Milan Karsten, Michiel Nijhuis (DNB), Ignacio Felez de Torres (BdE)