

From prompt to prototype in 7 minutes

19 March 2026, 14:07

Mihail Mihaylov
Wim Codenie

How a GenAI coding assistant built a forecasting tool

In a previous article, we showed how time series foundation models make forecasting far easier for industrial companies. These Generative AI models can produce reliable forecasts without training, tuning or specialist expertise. They are open-source and run on a standard laptop. The traditional cost barrier to accurate forecasting is therefore rapidly disappearing.

The other half of the story

However, forecasting models are only part of the challenge.

Even when the model itself is ready to use, someone still needs to build the surrounding software. Data must be loaded, the model must be called, results must be processed and visualised, and edge cases must be handled.

Traditionally this work required developers. In many projects, entire software teams were needed to build forecasting pipelines and applications. This raises a practical question.

If Generative AI has lowered the cost of forecasting models, could it also reduce the cost of building the software around them?

Enter GenAI coding assistants

A new generation of tools is beginning to answer that question.

Coding assistants such as Claude Code and Codex can generate software directly from natural language instructions. Instead of writing code manually, users describe what they want to build. The assistant then generates the application in a fraction of the time.

These systems do more than autocomplete code. They can structure projects, manage dependencies and adapt the architecture as requirements evolve. They do not replace developers, but they drastically shorten the path from idea to working prototype.

For companies exploring forecasting, this changes the process. Instead of requiring both data scientists and software engineers, a single domain expert with the right tools can often build an initial proof of concept.

What we built and how fast

At Sirris, we experienced this shift first-hand.

Over three months, we used Claude Code to build a comprehensive forecasting simulator. This was not a toy project. The simulator supports multiple time series foundation models, both local and cloud-based, along with several datasets, performance metrics, configurable parameters, visualisations and export functions.

3 months of work. A serious research tool.

Then we ran an experiment.

We asked Claude Code to analyse the simulator and produce a specification for a stripped-down version: a minimal proof-of-concept forecaster that reads a daily Excel file with historical time series data and generates a forecast for the next day using the best-performing foundation model from our benchmarks.

Claude produced the specification.

Then we did something deliberate. We opened a completely new session with the assistant. No knowledge of the simulator. A clean start.

We gave it the specification in one prompt and asked it to build the forecaster.

Seven minutes later, we had a working prototype.

Not a sketch. Not pseudocode. A complete application that loads data, calls the model and produces forecasts. With one additional prompt, it could even be prepared for integration into a production pipeline.

What took 3 months was done in 7 minutes?

To be clear, the seven-minute prototype is not equivalent to the three-month simulator. The simulator is a comprehensive research tool with many features. The prototype is deliberately minimal and focuses on one task.

But that is exactly the point. Most companies exploring forecasting do not need a full research platform. They need a proof-of-concept that answers a simple question: does this work for our data?

Until recently, answering that question required weeks of development and a dedicated team. Today it can often be done in minutes.

The three months of work were not wasted. They informed the specification: which model to use, how to structure the pipeline and which parameters matter. Domain expertise still drives the quality of the result. The coding assistant simply removes the implementation bottleneck.

What this means in practice

Consider the typical journey of an industrial company exploring forecasting. First, management must approve the investment. Then a data science team analyses the data, selects modelling approaches and builds pipelines. Months can pass before the first result appears.

Now consider the alternative.

A domain expert with access to a coding assistant describes the problem. Within minutes or hours, a working prototype runs on their machine. If the results look promising, the solution can be refined. If not, the experiment costs an afternoon instead of a quarter.

Proper engineering is still required when moving to production. But the exploration phase shrinks from months to days. Companies can test multiple forecasting ideas instead of committing to a single large project.


Range Slider

© Freepik

Two things have changed

Open-source time series foundation models now deliver forecasting accuracy comparable to expensive industrial solutions, without training or machine learning expertise. At the same time, coding assistants can turn a natural language description into a working forecasting prototype within minutes.

Together, these advances dramatically lower the cost of forecasting. Problems that were once too expensive to explore, such as forecasting energy use for a single building or predicting demand for niche products, suddenly become viable.

If you have time series data, you can start forecasting much sooner than you might think.

Explore the full series

This article is part of a three-part series on Generative AI for time series forecasting:

Intro: Your operations generate data. Are you forecasting with it?

Article 1: Generative AI makes forecasting accessible

Article 2: From prompt to prototype in 7 minutes

Authors



Mihail Mihaylov



Wim Codenie