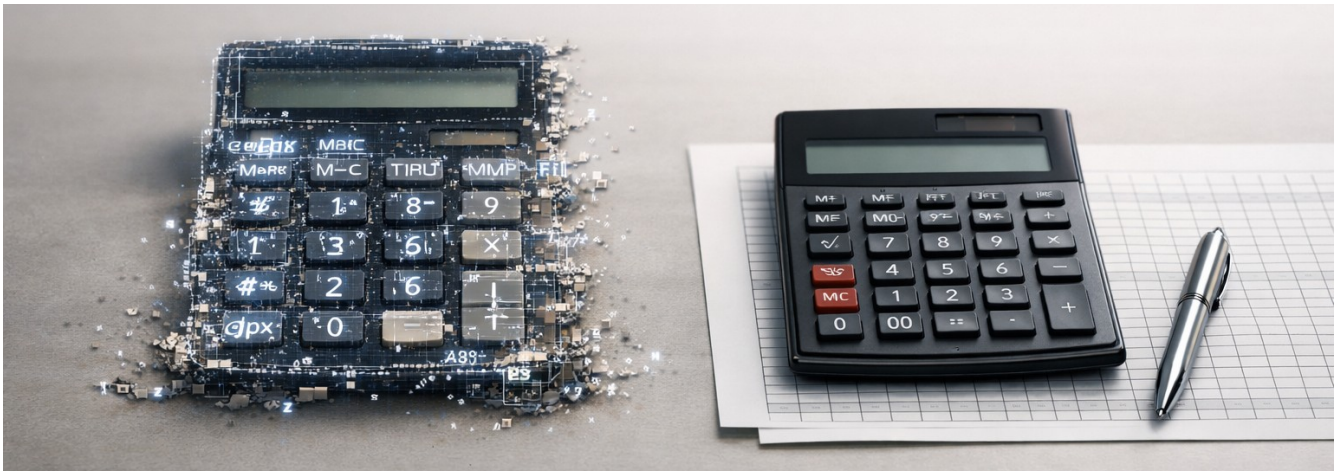
# Prediction Is Not Calculation

## Fluent, Confident, and Often Incorrect



There is a particular kind of wrong answer that modern AI produces exceptionally well: one that sounds right.

I was recently drawn into a spreadsheet summarization problem where genAI had been implemented as the obvious solution. The output was polished, coherent, and professional. It was also wrong. Not because the model malfunctioned, and not because AI is useless, but because we had mistaken one kind of capability for another. We had used a probabilistic language engine for a task that required exact arithmetic and deterministic results.

That kind of mistake is becoming easier to make. Once a system looks competent in one kind of task, it becomes easy to trust it in others.

***Tool choice still matters. And the reason people forget that isn't irrationality. It's success.***

Large language models have become so good at producing useful prose, plausible explanations, and even convincing step-by-step reasoning that it's easy to overgeneralize from what we see on the surface. If a system can explain algebra, comment on a balance sheet, and discuss the tradeoffs in a technical architecture, why shouldn't it total a column correctly or summarize a workbook with precision?

The problem is that these aren't all the same kinds of tasks. Language models are built to predict likely sequences, not to perform exact counting, arithmetic, or rule-based aggregation. They can often imitate the appearance of calculation surprisingly well. Sometimes they even get the result right. But getting it right *sometimes* isn't the same as being reliable.

When a business uses a probabilistic tool for work that demands exactness, it doesn't just risk an occasional wrong answer. It can create a more expensive workflow altogether, one that adds verification, rework, and management overhead to a task that should have been exact from the start.

## Prediction Isn't Calculation

A large language model doesn't approach math the way a calculator, spreadsheet, or database engine does. It doesn't "know" that 27 plus 38 must equal 65 in the rigid, rules-based sense that traditional software does. Instead, it generates the next most likely token in a sequence, based on patterns learned from an enormous body of *text*. That design makes it remarkably good at language. It doesn't make it trustworthy for precise computation.

Part of the problem is that the surface behavior is so persuasive. A modern model can explain algebra, describe accounting logic, walk through a spreadsheet formula, and present its answer in calm, polished prose. To a human reader, that often feels very close to reasoning. Sometimes it's useful. But a convincing explanation doesn't guarantee correct results.

The gap becomes especially obvious in tasks that demand precision rather than plausibility. Counting words in a paragraph, totaling values across multiple sheets, checking a summary against a source table, or performing multi-step arithmetic aren't merely language tasks with numbers inside them. They are exact tasks, which means they depend on consistency, repeatability, and rules that must be followed every time.

That's why large language models can be so impressive in discussion and still fragile in execution. They're very good at discussing the logic of a calculation while remaining less dependable at carrying it out exactly. They can mimic the form of step-by-step reasoning without having the same underlying guarantees that traditional computational tools provide. In other words, they can sound like a calculator without actually being one.

The distinction matters because business users are not buying prose for its own sake. They are often buying decisions, summaries, reconciliations, forecasts, and operational guidance. Once money, compliance, planning, or execution depends on the answer, sounding right is no longer good enough. In those cases, conventional analysis tools still matter.

They don't improvise totals or invent plausible estimates unless explicitly instructed to do so. They execute rules the same way every time, and in many business settings that consistency is exactly what makes them more valuable than a generative model.

None of this diminishes what LLMs are genuinely good for. They can explain a result beautifully. They can generate a formula, suggest a query, draft a narrative summary, or help a nontechnical user interact with a more exact system. But those are different roles. The moment we confuse "can talk about the math" with "should be trusted to perform the math," we move from intelligent use into expensive misuse.

That's the trap. The model's surface competence makes it easy to trust it beyond the role it should actually play. Once that happens, prediction begins to masquerade as calculation.

That distinction becomes even more important when the data is already structured.

## Why Structured Data Makes This Worse

Structured data changes the nature of the problem. Once information is already organized into rows, columns, tabs, headers, formulas, and named fields, much of the ambiguity that makes language models so useful has already been removed. The task is no longer to interpret messy prose or infer intent. It's to sort, filter, group, total, compare, reconcile, or summarize, all of which are rule-based operations. That's why moving structured data into a probabilistic system often adds uncertainty where none was needed. A spreadsheet already states its structure, a query already defines its logic, and a pivot table already groups and totals according to rules. Asking a language model to summarize that material directly means giving up those built-in guarantees and relying on inference from a *representation* of the structure rather than from the structure itself.

The problem extends beyond arithmetic alone. The risk isn't just that the model may add a column incorrectly. It might also drop rows, confuse headers, collapse distinct categories, misread relationships across tabs, or produce a summary that sounds perfectly plausible while quietly misrepresenting the underlying data. In other words, structured data invites a kind of error that's especially dangerous: not dramatic failure, but polished distortion.

Traditional tools handle this differently. Excel, SQL, Power BI, Pandas, and related platforms were built to perform consistent operations transparently and repeatably. They don't guess what belongs in a subtotal, or invent a grouping because it sounds reasonable in context. They apply explicit logic, and they apply it the same way *every time*. That's less glamorous than genAI, but in many business settings it's exactly what makes these tools valuable.

AI still has a role in structured analysis, but the stronger role usually sits beside the analysis rather than replacing it. Once the model is asked to summarize the data directly, the workflow gives up built-in guarantees and relies instead on inference. This is where avoidable uncertainty enters.

Used well, AI doesn't make the right tool less important. It helps organizations get more from the tool already in place.

## The Hidden Cost of the Wrong Tool

The cost of using the wrong tool isn't limited to the invoice for the tool itself, and that's especially true with genAI. The real cost appeared in what followed: checking, comparison, and another review cycle.

What emerges is a familiar pattern. When organizations use a probabilistic system for exact analysis, they often end up paying three times: once for the AI output, once for human validation, and once

again for correction or rework. In our spreadsheet case, that meant paying for a summary that Excel could have produced exactly, then paying a human to inspect the summary line by line, and then paying again in time and attention to repair what should never have been wrong in the first place.

In some ways, this is worse than obvious failure. An absurd answer gets rejected quickly. A polished, nearly correct answer is much more expensive, because it demands careful scrutiny. It slows people down precisely by appearing to save time. One of the most underappreciated costs in enterprise AI is that plausible wrongness is operationally stickier than visible error.

That cost isn't only financial. It is cognitive and organizational as well. Attention gets diverted into checking work that should have been exact from the start. Review loops expand. Managers become less certain about which outputs can be trusted and which need second looks. Some users become overconfident because the output sounds authoritative. Others swing the other way and begin distrusting the system entirely. Neither outcome is healthy. A tool that's misapplied doesn't just produce the occasional bad answer; it brings friction into the workflow around it.

That was the real lesson in the spreadsheet example. The work didn't disappear. It simply moved. Instead of spending effort on structured analysis from the start, the organization spent it later on verification, interpretation, and cleanup. What looked like acceleration was really cost shifted downstream.

The business case becomes clear here. The problem with misapplied AI isn't simply that it can be wrong. It's that it can create negative leverage. It promises speed, then adds checking. It promises convenience, then adds rework. It promises simplification, then adds another layer that has to be managed. Used in the right place, AI reduces friction. Used in the wrong place, it quietly multiplies it.

Once that pattern becomes visible, the next question is the important one: where should AI sit in the workflow so that it helps without undermining exactness?

## Where AI Adds Real Value

One of the most obvious roles is guidance. Many spreadsheet environments grow complicated over time. Data arrives from multiple tabs, different teams structure columns differently, formulas become uneven, and the person asked to produce the summary may not know the best analytical path. In that setting, AI can be extremely useful as a coach. It can suggest whether the job is best handled with a pivot table, slicers, Power Query, a cross-sheet lookup, or a Power BI model. It can explain those choices in plain language and help a less experienced user move more quickly toward a sound design.

AI is also valuable as a translator between business intent and technical execution. A user may know the question they need answered but not the mechanics required to answer it. They may know they want to compare quarterly sales by region across several sheets, isolate anomalies, and present the result clearly, but not know whether that calls for a pivot table, a data model, a DAX measure, or a

filtered visual. Generative AI can shorten that gap by helping turn an analytic goal into the formulas, queries, or reporting structures needed to achieve it.

Real productivity gains begin there. Instead of asking AI to replace structured analysis tools, organizations should use it to help people use those tools more effectively. The speed comes not from tolerating approximate answers, but from reducing the friction involved in getting to exact ones. In many environments, that's the better bargain. It preserves reliability while still giving the user a faster path to insight.

AI can also add value after the precision work is done. Once the pivot table has been built, the query has run, or the dashboard has produced verified results, the language model becomes useful again. It can draft an executive summary, explain trends in plain English, suggest ways of presenting the findings, identify questions worth investigating further, or tailor the explanation for different audiences. In that role, it is not inventing the numbers. It is helping people communicate and think with them.

AI is often strongest at interpretation, explanation, interface, and acceleration. It can generate formulas, draft queries, explain why a slicer behaves the way it does, or help someone who is rusty with spreadsheets move more quickly through a complicated analysis. Those are real productivity gains. What it shouldn't become, when the task demands exactness, is the authority of record for the results themselves.

Seen this way, the issue isn't whether AI should be in the workflow, but whether it's been assigned the right job. Used well, it acts as a guide to the right analysis tools, a translator between user intent and technical method, and an interpreter of verified output. Used poorly, it is asked to bypass those tools entirely and improvise where exactness was required all along.

A more mature view of responsible AI starts there. The goal isn't to use less AI, but to use it where it adds the most value. In that model, AI increases the value of the systems already in place.

## A Simple Rule for Leaders

Before assigning a task to generative AI, ask:

- **Does the answer need to be precise?**
  If yes, start with a rule-based tool.
- **Is the data already structured?**
  If it lives in spreadsheets, tables, databases, or dashboards, much of the interpretive work is already done.
- **Does a conventional tool already solve this well?**
  Conventional analysis and reporting tools, along with scripts, are often faster, cheaper, and more reliable for exact summarization.

- **Am I using AI to accelerate the right tool, or replace it?**
  AI adds more value when it helps build the pivot table, query, slicer, or report than when it improvises the answer itself.
- **Will someone still need to verify the result by hand?**
  If so, the apparent speed gain may be an illusion.
- **What exactly is AI adding here?**
  The real payoff is guidance, formula generation, explanation, interpretation, and clearer communication. Novelty alone is not value.
- **Who or what is the authority of record for the final answer?**
  For numeric calculations, that should usually be a deterministic system, not a language model.

If the task is exact and the data is structured, let AI help people use the right tool better. Don't ask it to replace the tool that was built for the job. That's the practical rule. The broader principle is simpler still.

## Responsible AI Is About Placement, Not Enthusiasm

The real lesson here is not that generative AI is flawed, or that older tools deserve loyalty for their own sake. It is that true value comes from placing each tool where its strengths actually apply.

Large language models are extraordinary at language, interpretation, translation, and guidance. They can help more people do advanced analysis well, help users move faster through unfamiliar tools, and turn verified results into clear explanations that decision-makers can act on. Those are real gains. But none of them erase the distinction between a system that predicts plausibly and a system that computes exactly.

For that reason, responsible AI use is ultimately a placement problem. It's not about resisting new technology, and it's not about forcing new technology into every available space. It's about knowing where AI adds value, where determinism still matters, and how to combine the two without confusing one for the other.

The organizations that get the most from AI won't be the ones that use it most enthusiastically. They'll be the ones that use it most intelligently, knowing when to let AI explain and accelerate, and when to let conventional tools do the heavy lifting.

Prediction isn't calculation. Fluency isn't precision. The best results still come from putting the right capability in the right place.