OPEN-WEIGHT MODEL ADAPTATION

# Domain Adaptation and Post-Training of an Open-Weight Model for Financial Reasoning

A technical case study using expert data, agent environments, and outcome-based evaluation

BASE MODEL / KIMI K2.6  DOMAIN / FINANCIAL REASONING

| 81.9% | +14.5 pts | 3.1% |
| --- | --- | --- |
| Aggregate expert score | Improvement over base model | Critical-error rate |

AUTHORS / CONTACT

Lord Munjal
lord@alpheva.ai

Dinesh Bali
dinesh@alpheva.ai

# Contents

A complete account of the training system, evaluation design, results, limitations, and supporting appendices.

# Abstract

General-purpose language models have become increasingly capable at reasoning, tool use, and long-horizon task execution. Yet capability on professional work does not follow automatically from general intelligence.

Financial applications require a different level of reasoning. A useful model must interpret customer-specific context, identify the actual decision being made, perform accurate calculations, recognize missing information, reason across interacting domains, weigh trade-offs, use tools correctly, and communicate conclusions with appropriate uncertainty.

This case study describes how Alpheva AI worked with a U.S.-based financial vertical AI company to build a domain-specific post-training pipeline around an open-weight foundation model.

The engagement used Kimi K2.6 as the base model. Rather than training a foundation model from scratch, Alpheva focused on the layer that converts domain expertise and real-world financial work into model-training signals: expert-authored tasks, supervised demonstrations, preference judgments, realistic financial environments, agent trajectories, deterministic verifiers, domain-specific rubrics, expert adjudication, and outcome-based post-training.

The resulting system was organized around a closed improvement loop:

real financial work → expert task generation → demonstrations and preferences → supervised fine-tuning → agent rollouts → verification and grading → post-training → held-out evaluation → failure analysis → targeted expert data

This approach treats expert judgment as a training signal rather than merely a source of labels.

The evaluation framework covered financial correctness, contextual reasoning, decision quality, completeness, communication, and critical-error incidence. A separate agent evaluation measured end-to-end task completion, tool selection, intermediate calculations, handling of missing information, and outcome verification.

The central finding is that different training signals address different classes of model failure. Expert demonstrations improve contextual reasoning. Preference optimization improves alignment with professional judgment. Environment-based training improves multi-step execution. Deterministic verification provides stronger signals for objectively measurable outcomes.

The broader implication extends beyond financial AI: vertical AI companies can create proprietary model capability by owning the system that converts domain expertise and real-world workflows into training data, environments, evaluators, and post-training signals.

# 1. Client and Confidentiality

The client was a U.S.-based financial vertical AI company whose identity is withheld at its request.

The company develops AI systems that help consumers analyze and make decisions across areas including investments, taxes, debt, insurance, retirement, cash flow, and broader financial planning.

The client's application already incorporated customer financial information, documents, financial knowledge, retrieval systems, and language models.

The engagement focused on improving the model layer responsible for financial reasoning and multi-step financial workflows.

No customer-identifying information, proprietary product details, or confidential business information is disclosed in this case study.

# 2. The Problem

The client already had a functioning vertical AI application.

The application could answer many conventional financial questions.

The harder problem was what happened when the system had to perform actual financial reasoning.

Consider questions such as:

- Should a customer pay down a mortgage or invest excess cash?
- How should a change in income affect tax and retirement decisions?
- Is an investment allocation appropriate given the customer's liquidity requirements?
- How should several financial decisions be optimized together rather than independently?
- What information is missing before a recommendation can safely be made?
- Which trade-offs matter most to this particular customer?
- What should the customer actually do next?

These are not simply knowledge-retrieval problems.

A model may know the relevant tax rule, understand mortgage mathematics, and know the historical relationship between risk and return, yet still produce an inferior answer.

Failure can occur because the model:

- identifies the wrong decision;
- ignores material customer context;
- makes an unsupported assumption;
- misses an interaction between financial domains;
- performs an incorrect calculation;
- fails to ask for missing information;
- selects the wrong tool;
- loses state during a multi-step workflow;
- gives an answer that is technically defensible but poorly aligned with professional judgment;

- or communicates a conclusion with more certainty than the evidence warrants.

The objective was therefore not simply to make the model know more finance.

The objective was to make it reason more reliably about financial problems in context.

# 3. Research Question

The engagement was organized around one central question:

> To what extent, and through which training signals, can expert-generated data, realistic financial environments, and outcome-based post-training improve an open-weight model's performance on the financial reasoning and agentic workflows required by a vertical AI application?

Four hypotheses followed.

### H1. Expert demonstrations improve contextual financial reasoning

High-quality demonstrations should teach the model how professionals structure financial decisions, not merely which answers are correct.

### H2. Preference data improves alignment with professional judgment

Two financial answers can both be technically plausible while differing substantially in usefulness, completeness, risk awareness, and decision quality.

Expert preference data should help the model distinguish between these cases.

### H3. Environment-based training improves multi-step execution

Models trained on realistic workflows should become better at using tools, maintaining state, performing intermediate actions, and reaching defined outcomes.

### H4. Combining multiple training signals produces broader improvement

The combination of demonstrations, preferences, trajectories, verification, and outcome-based optimization should produce broader gains than supervised fine-tuning alone.

# 4. Base Model

The engagement used Kimi K2.6 as the starting model.

The model was selected because it provided a capable general reasoning and agentic foundation while allowing domain-specific adaptation.

The selection criteria were:

- Strong general reasoning capability.
- Strong tool-use and long-horizon execution capability.
- Availability of model weights suitable for domain adaptation.
- Sufficient baseline capability to focus the engagement on domain behavior rather than fundamental model limitations.

The objective was not to make the base model universally superior to frontier proprietary models.

The objective was narrower:

> Create a model that is materially better at a defined financial workload.

This distinction is important.

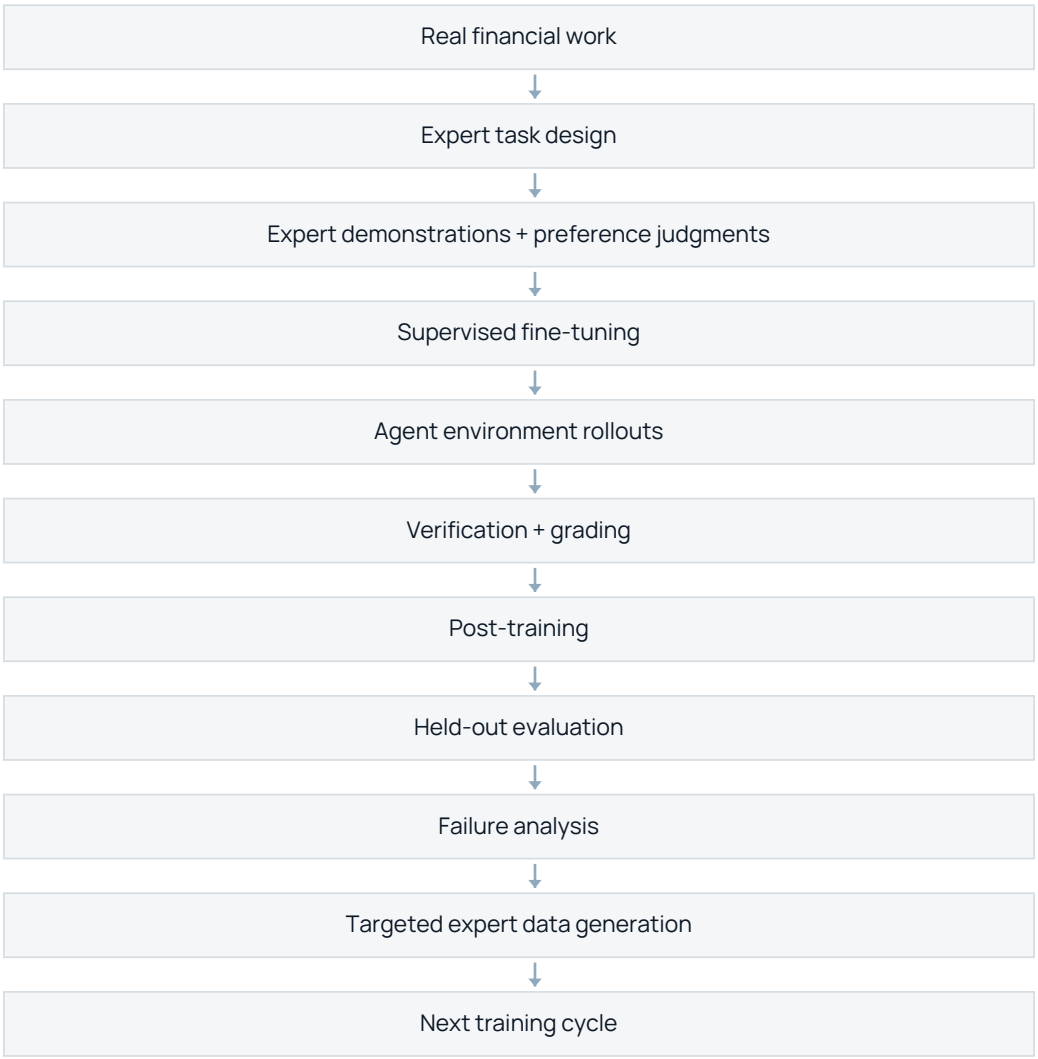
Vertical AI companies generally do not need to reproduce the entire capability frontier. They need to optimize model behavior around the work their customers actually perform.

# 5. From Expert Knowledge to Training Signal

The central design principle was that expert knowledge should not be treated as a static knowledge base.

Instead, expert work was converted into multiple forms of training signal.

The resulting pipeline was:




Each layer served a different purpose.

Demonstrations taught the model how to approach the work.

Preferences taught it which plausible behaviors experts preferred.

Environments exposed the model to the structure of multi-step work.

Verifiers distinguished objectively correct outcomes from plausible-looking responses.

Expert adjudication handled tasks where professional judgment could not be reduced to a deterministic rule.

Failure analysis determined what new data should be generated next.

This transformed the training process from a static dataset exercise into an iterative system.

# 6. Expert Task Generation

Alpheva generated a corpus of expert-reviewed financial tasks designed to resemble actual customer problems rather than isolated financial trivia.

The training configuration consisted of approximately:

- 8,400 expert-reviewed examples
- 42 financial professionals
- 6,700 pairwise preference judgments

The expert pool represented multiple areas of financial work, including:

- financial planning;
- investment management;
- tax;
- insurance;
- lending;
- retirement planning;
- and financial analysis.

Tasks were constructed around realistic customer states.

Depending on the task, the model could receive:

- customer profile;
- income;
- expenses;
- cash balances;
- investment holdings;
- retirement accounts;
- liabilities;
- debt terms;
- tax circumstances;
- insurance information;
- financial documents;
- stated objectives;
- constraints;
- and relevant external information.

Experts were asked to reason through each problem systematically.

They identified:

- The actual decision facing the customer.
- The facts that were material.

- Missing information.
- Required calculations.
- Relevant alternatives.
- Important trade-offs.
- The conclusion supported by the available evidence.
- How the conclusion should be communicated.

The resulting examples therefore captured more than answer correctness.

They captured the structure of professional financial judgment.

# 7. Training Data

The training corpus was divided into complementary data types.

## 7.1 Expert Demonstrations

Approximately 8,400 expert-reviewed examples formed the supervised training corpus.

The examples emphasized:

- contextual reasoning;
- calculation accuracy;
- explicit assumptions;
- trade-off analysis;
- appropriate uncertainty;
- decision-oriented communication;
- and recognition of missing information.

The objective was not to teach the model a fixed response template.

It was to expose the model to repeated examples of how professionals approach similar classes of decisions.

## 7.2 Preference Data

Approximately 6,700 pairwise expert preference judgments were generated.

Experts compared candidate responses and selected the response they would prefer to provide to a customer.

Preference dimensions included:

- financial correctness;
- contextual relevance;
- completeness;
- reasoning quality;
- actionability;
- risk awareness;
- uncertainty calibration;

- and communication.

This data addressed a limitation of ordinary accuracy metrics.

A response can contain no obvious factual error and still be a poor professional answer.

Preference optimization provided a mechanism for learning those distinctions.

## 7.3 Failure Taxonomy

Alpheva maintained a structured taxonomy of model failures.

Rather than recording only whether an answer was correct, failures were categorized according to their underlying cause.

Categories included:

- factual error;
- calculation error;
- reasoning error;
- context omission;
- unsupported assumption;
- incomplete trade-off analysis;
- tool-use error;
- failure to identify missing information;
- inappropriate confidence;
- and multi-step execution failure.

This taxonomy became an important part of the improvement loop.

A failure was not merely a negative example.

It became a specification for what additional expert data should be generated.

# 8. Financial Agent Environment

Static question-answer data is insufficient for many financial workflows.

A financial agent may need to inspect several sources, perform calculations, call tools, retrieve documents, update an intermediate state, and verify the resulting decision.

Alpheva therefore constructed a controlled financial task environment.

The environment represented the types of information and tools a financial AI agent might encounter, including:

- structured financial accounts;
- transaction histories;
- financial documents;
- tax information;
- investment information;
- calculators;
- APIs;
- and research tools.

Each task began from a defined state.

The model interacted with the environment through a sequence of actions.

A trajectory could therefore be represented as:

$$\text{state}_0 \rightarrow \text{observation} \rightarrow \text{action} \rightarrow \text{tool call} \rightarrow \text{state}_1 \rightarrow \text{observation} \rightarrow \text{action} \rightarrow \ldots \rightarrow \text{final outcome}$$

The complete trajectory was recorded.

This allowed the evaluation system to distinguish between process quality and outcome quality.

For example, an agent might reach a numerically correct answer after using an incorrect source.

Another agent might perform several correct intermediate steps but fail to complete the workflow.

These are different failures and should generate different training signals.

# 9. Verification and Grading

One of the most important design decisions was to avoid treating every financial task as either fully deterministic or fully subjective.

Alpheva used a three-layer evaluation architecture.

## 9.1 Deterministic Verification

Where the correct outcome could be computed or checked directly, deterministic verification was used.

Examples included:

- numerical calculations;
- account balances;
- debt amortization;
- portfolio allocation;
- constraint satisfaction;
- document extraction;
- and tool/API execution.

These checks provide a stronger training signal than a language model simply judging whether an answer appears plausible.

## 9.2 Rubric-Based Grading

Some financial reasoning tasks do not have one mathematically correct answer.

These tasks were evaluated against structured rubrics covering:

- completeness;
- contextual reasoning;
- financial judgment;
- trade-off analysis;
- assumptions;
- uncertainty;
- and actionability.

Rubrics converted expert expectations into explicit evaluation criteria.

## 9.3 Expert Adjudication

Where professional judgment could not be reliably captured through deterministic verification or automated grading, expert review remained the reference standard.

This was particularly important for ambiguous cases where multiple answers could reasonably be defended.

The objective was not to force subjective financial judgment into an artificial binary reward function.

Instead, the evaluation system used the strongest available verification mechanism for each class of task.

# 10. Grader Calibration and Reliability

A serious evaluation system cannot assume that its grader is correct simply because it produces a number.

The grading stack therefore incorporated calibration against expert judgment.

The evaluation protocol included:

- Independent expert review of a representative sample of model outputs.
- Comparison between automated grades and expert grades.
- Adjudication of disagreements.

- Periodic recalibration of grading rubrics.
- Measurement of inter-rater agreement for human-reviewed outputs.
- Separation of grader development data from the final held-out evaluation set.

For automated graders, the key question is not simply whether the grader correlates with the model score.

The relevant question is:

> Does the grader reliably distinguish the behaviors that domain experts actually care about?

This distinction is critical when the grader itself becomes part of the training loop.

# 11. Held-Out Evaluation Set

A 1,200-task held-out evaluation set was defined before the final training cycle.

The evaluation distribution was designed to approximate the financial work performed by the application.

TABLE 01

| Category | Share |
|---|---|
| Investment and portfolio decisions | 25% |
| Tax | 20% |
| Debt and credit | 15% |
| Retirement | 15% |
| Insurance | 10% |
| Cash flow and personal finance | 10% |
| Cross-domain decisions | 5% |

The evaluation set included:

- factual questions;
- contextual reasoning;
- numerical problems;
- multi-step decisions;
- ambiguous questions;
- incomplete-information scenarios;
- and adversarial edge cases.

No evaluation example was used directly in training.

The held-out set was treated as a fixed measurement instrument rather than another source of optimization data.

# 12. Experimental Design

Four primary configurations were evaluated.

### Model A: Base Model

Kimi K2.6 with no financial-specific post-training.

### Model B: SFT

Kimi K2.6 fine-tuned on expert demonstrations.

### Model C: SFT + Preference Optimization

Supervised fine-tuning followed by preference-based post-training.

### Model D: Full Alpheva Pipeline

SFT + preference optimization + environment-generated trajectories + outcome-based reinforcement learning.

All configurations were evaluated against the same held-out evaluation set.

Where applicable, inference parameters, context limits, tool availability, and evaluation procedures were held constant.

This design isolates the incremental contribution of:

- expert demonstrations;
- preference data;
- environment-based training;
- and outcome-based optimization.

# 13. Evaluation Metrics

Performance was measured across five primary dimensions.

TABLE 02

| Dimension | Weight |
|---|---|
| Financial correctness | 30% |
| Contextual reasoning | 25% |
| Decision quality | 20% |
| Completeness | 15% |
| Communication | 10% |

A separate critical-error rate measured errors capable of materially changing a customer's financial decision or producing a materially incorrect financial action.

Human review was used to validate automated grading and identify systematic failures that automated evaluation could miss.

A stratified sample of approximately 240 outputs was included for independent expert review.

# 14. Results

## 14.1 Overall Performance

TABLE 03

| Metric | Base | SFT | SFT + Preference | Full Alpheva |
|---|---|---|---|---|
| Aggregate expert score | 67.4% | 75.8% | 79.6% | 81.9% |
| Financial correctness | 74.8% | 80.6% | 84.3% | 88.1% |
| Contextual reasoning | 61.2% | 72.1% | 77.4% | 82.7% |
| Completeness | 64.7% | 72.9% | 77.1% | 80.4% |
| Decision quality | 60.9% | 70.4% | 75.9% | 79.8% |
| Expert preference | 54.8% | 68.3% | 73.9% | 78.6% |
| Critical-error rate | 8.7% | 5.4% | 4.0% | 3.1% |

The pattern is more informative than any individual metric.

Supervised fine-tuning produced the first substantial improvement across the evaluation.

Preference optimization produced an additional increase in alignment with professional judgment.

The full pipeline produced the largest incremental gains in decision quality and tasks requiring multiple steps.

The results show that the training signals were complementary rather than interchangeable.

# 15. Agentic Task Results

A separate evaluation measured performance on multi-step financial workflows.

The evaluation contained 300 workflows requiring the model to use available information and tools to reach a defined outcome.

TABLE 04

| Metric | Base Kimi K2.6 | Full Alpheva |
|---|---|---|
| End-to-end task completion | 52.4% | 76.9% |
| Correct tool selection | 68.1% | 89.3% |
| Correct intermediate calculations | 73.6% | 91.5% |
| Appropriate handling of missing information | 49.7% | 78.4% |
| Outcome verification | 57.2% | 84.6% |

The most important improvement was not simply individual tool-call accuracy.

It was coherence across multiple steps.

The post-trained model was more likely to use intermediate results to inform subsequent actions, recognize when information was insufficient, and verify the final state rather than treating the workflow as a sequence of independent tool calls.

# 16. Ablation Analysis

The purpose of ablation is to determine whether improvement comes from the proposed methodology rather than simply adding more training.

Each ablation removed one major training signal while holding the rest of the pipeline as constant as possible.

## 16.1 Removing Expert Demonstrations

The largest degradation occurred on contextual financial reasoning.

The model retained broad financial knowledge but became more likely to produce generic responses.

Interpretation: expert demonstrations provide the primary mechanism for transferring domain-specific reasoning patterns.

## 16.2 Removing Preference Optimization

Financial correctness remained comparatively strong, while expert preference declined.

The model produced responses that were often reasonable but less aligned with how professionals prioritize information, explain trade-offs, and communicate uncertainty.

Interpretation: preference data contributes primarily to judgment alignment rather than basic financial knowledge.

## 16.3 Removing Environment Trajectories

Static question-answer performance remained comparatively stable, while multi-step task completion declined.

Tool use, state management, and long-horizon execution degraded disproportionately.

Interpretation: environment-generated trajectories provide training information that static demonstrations cannot fully reproduce.

## 16.4 Removing Deterministic Verification

Numerical and tool-based tasks became more difficult to optimize reliably.

Language-based graders can reward responses that appear strong despite failing objective checks.

Interpretation: where objective verification is possible, deterministic verification provides a higher-quality training signal than purely language-based evaluation.

# 17. Representative Behavioral Change

Consider the following customer question:

> "I have $250,000 in cash, a mortgage at 3.1%, $150,000 in retirement accounts, and expect to retire in eight years. Should I pay down the mortgage?"

A generic model can produce a technically reasonable discussion of mortgage rates, investment returns, and liquidity.

The domain-adapted model is trained to decompose the decision into the variables that actually determine the answer, including:

- liquidity requirements before retirement;
- opportunity cost of using cash to reduce low-cost debt;
- existing retirement exposure;
- tax implications;
- risk tolerance;
- investment horizon;
- emergency reserves;
- guaranteed debt reduction versus uncertain investment returns;
- and interactions between the customer's retirement timeline and cash requirements.

The important behavioral change is not verbosity.

It is decision structure.

A stronger financial model does not merely produce more financial facts.

It identifies the variables that determine the decision before producing the conclusion.

# 18. Error Analysis

Aggregate scores conceal important differences in model behavior.

Residual failures were therefore analyzed by underlying cause.

## 18.1 Cross-Domain Interactions

Some decisions require simultaneous reasoning across tax, investment, retirement, insurance, and cash flow.

The model can reason correctly within each domain yet fail to connect them.

These failures are particularly important because the value of a vertical financial system often lies precisely in connecting domains.

## 18.2 Ambiguous Customer Intent

Customers frequently ask literal questions that conceal a broader decision.

The model occasionally answers the literal question instead of identifying the decision behind it or requesting clarification.

This suggests a need for additional training around intent discovery and clarification behavior.

## 18.3 Long-Horizon Tool Use

Error probability tends to increase with workflow length.

Failures become more common when the model must combine multiple documents, calculations, APIs, and intermediate states.

These cases are particularly valuable for trajectory-based training because the failure is often procedural rather than informational.

## 18.4 Judgment Under Uncertainty

Financial decisions frequently involve uncertainty rather than a single objectively optimal answer.

A model can identify the relevant trade-offs yet still express the conclusion too definitively.

Training must therefore optimize not only for correctness but for calibrated confidence.

## 18.5 Rare Edge Cases

Residual critical failures tend to concentrate in unusual combinations of financial constraints.

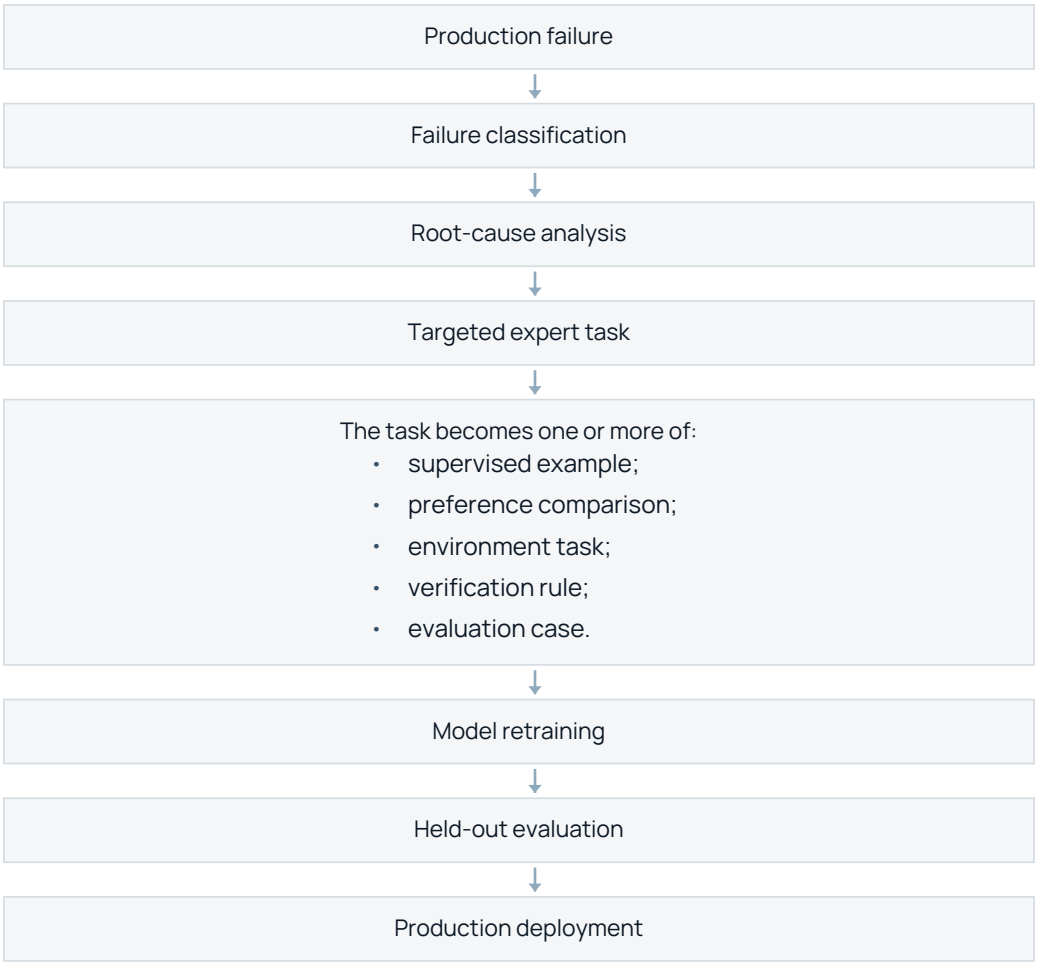
Examples include interactions among:

- tax rules;
- retirement accounts;
- investment decisions;
- liquidity requirements;
- and other customer constraints.

These failures become inputs to the next generation of expert tasks.

# 19. The Training Flywheel

The most strategically important component of the system is the feedback loop between production behavior and model training.




The newly identified failure is then measured again.

This creates a compounding data asset.

The objective is not simply to increase the number of training examples.

It is to increase the proportion of examples that address known weaknesses in model behavior.

Over time, the training system becomes increasingly specific to the work performed by the application.

That specificity is difficult to reproduce using generic public datasets alone.

# 20. Why This Matters for Vertical AI

There are two broad approaches to building a vertical AI product.

## Approach 1: Model as a Commodity

The company relies primarily on a frontier API and differentiates through:

- prompts;
- retrieval;
- product experience;
- workflow integration;
- and application UX.

This can produce a strong product, but much of the underlying model capability remains outside the company's control.

## Approach 2: Model Behavior as Part of the Product

The company develops proprietary:

- expert datasets;
- task environments;
- trajectories;
- evaluators;
- reward signals;
- failure taxonomies;
- and post-training pipelines.

The second approach creates a tighter connection between the application's domain expertise and the behavior of the model itself.

The objective is not to create a model that is universally better.

It is to create a model that is better at the company's own work.

That distinction becomes increasingly important as foundation models become more capable and increasingly interchangeable at the application layer.

# 21. What Alpheva Actually Contributed

The value of the engagement was not simply the creation of a dataset.

Alpheva's contribution was the construction of the training system around the model.

That system included:

Expert Data

Converting professional financial judgment into structured demonstrations and preference data.

Task Environments

Recreating the state, tools, documents, APIs, and constraints involved in real financial workflows.

Trajectory Generation

Capturing not only final answers but the sequence of actions leading to those answers.

Verification

Using deterministic checks wherever outcomes could be objectively measured.

Evaluation

Combining automated grading, structured rubrics, and expert adjudication.

Failure Analysis

Converting model failures into structured categories.

Targeted Data Generation

Using those categories to determine what experts should generate next.

Post-Training

Using the resulting signals to improve model behavior.

The core value proposition is:

> Alpheva converts expert work into the data, environments, and evaluation signals required to train AI systems on professional tasks.

# 22. Limitations

## 22.1 Expert Data Is Expensive

High-quality financial expertise cannot be generated at the cost of generic labeling.

The economics depend on selecting tasks where expert judgment produces substantial incremental training value.

## 22.2 Professional Judgment Is Not Always Deterministic

Qualified professionals can reasonably disagree.

A robust training system must preserve legitimate disagreement rather than artificially collapsing every problem into a single “correct” answer.

## 22.3 Automated Graders Can Fail

An automated grader can systematically reward an undesirable behavior if its rubric is poorly specified.

Grader calibration against expert judgment is therefore essential.

## 22.4 Benchmark Improvement Is Not Production Proof

Improvement on a held-out benchmark does not guarantee equivalent improvement across every real customer interaction.

Production monitoring remains necessary.

## 22.5 Distribution Shift

Financial products, tax rules, market conditions, customer behavior, and regulations change.

Training and evaluation systems therefore require continual updating.

## 22.6 Specialization Can Create Trade-Offs

Optimizing heavily for financial workflows may affect performance on unrelated tasks.

The appropriate optimization target depends on the application.

# 23. Conclusion

The starting point for this engagement was not a weak model.

It was a capable general-purpose open-weight model.

The differentiating work was the layer built around it.

Alpheva transformed financial expertise and real-world workflows into:

- supervised training examples;
- expert preferences;
- realistic task environments;
- agent trajectories;
- deterministic verifiers;
- domain-specific rubrics;
- expert evaluations;
- failure taxonomies;
- and outcome-based training signals.

The model was therefore evaluated not merely on whether it could produce plausible financial text, but on whether it could perform the underlying work.

The evaluation showed substantial improvement across both static financial reasoning and multi-step agentic workflows:

- Aggregate expert score: 67.4% → 81.9%
- Financial correctness: 74.8% → 88.1%
- Contextual reasoning: 61.2% → 82.7%
- Decision quality: 60.9% → 79.8%
- Expert preference: 54.8% → 78.6%
- Critical-error rate: 8.7% → 3.1%
- Multi-step task completion: 52.4% → 76.9%

The ablation results provide the more important insight.

Different training signals improve different dimensions of behavior.

Expert demonstrations transfer domain reasoning patterns.

Preference optimization transfers professional judgment.

Environment trajectories teach multi-step execution.

Deterministic verification anchors training to objectively measurable outcomes.

The combination creates a training system that is substantially more specialized to the work the application needs to perform.

A vertical AI company's defensibility does not necessarily require building a foundation model.

It can come from owning the system that teaches a model how to perform a valuable class of professional work.

The application provides the workflow.

The experts provide the judgment.

The environments provide the work.

The evaluators define success.

The failures identify what remains unsolved.

And the post-training loop converts those signals into increasingly specialized model behavior.

For financial AI, the objective is therefore not simply to build a model that knows finance.

It is to build a model that can perform financial work reliably.

# Appendix A. Training Configuration

TABLE 05

| Component | Configuration |
|---|---|
| Base model | Kimi K2.6 |
| Architecture | Mixture of Experts |
| Total parameters | ~1T |
| Activated parameters | ~32B |
| Expert contributors | 42 |
| Expert-reviewed SFT examples | 8,400 |
| Preference judgments | 6,700 |
| Generated agent trajectories | 52,000 |
| Retained post-training trajectories/examples | 11,500 |
| Held-out evaluation tasks | 1,200 |
| Agent evaluation tasks | 300 |
| Human-adjudicated evaluation responses | 240 |

# Appendix B. Evaluation Dimensions

## Financial Correctness

Accuracy of financial facts, calculations, relationships, constraints, and conclusions.

## Contextual Reasoning

Appropriate use of customer-specific circumstances and material facts.

## Completeness

Coverage of material considerations required to address the task.

## Decision Quality

Ability to connect evidence, constraints, alternatives, and trade-offs to an actionable conclusion.

## Communication

Clarity, concision, usability, appropriate qualification, and calibrated uncertainty.

## Critical Error

An error capable of materially changing the customer's financial decision or producing a materially incorrect financial action.

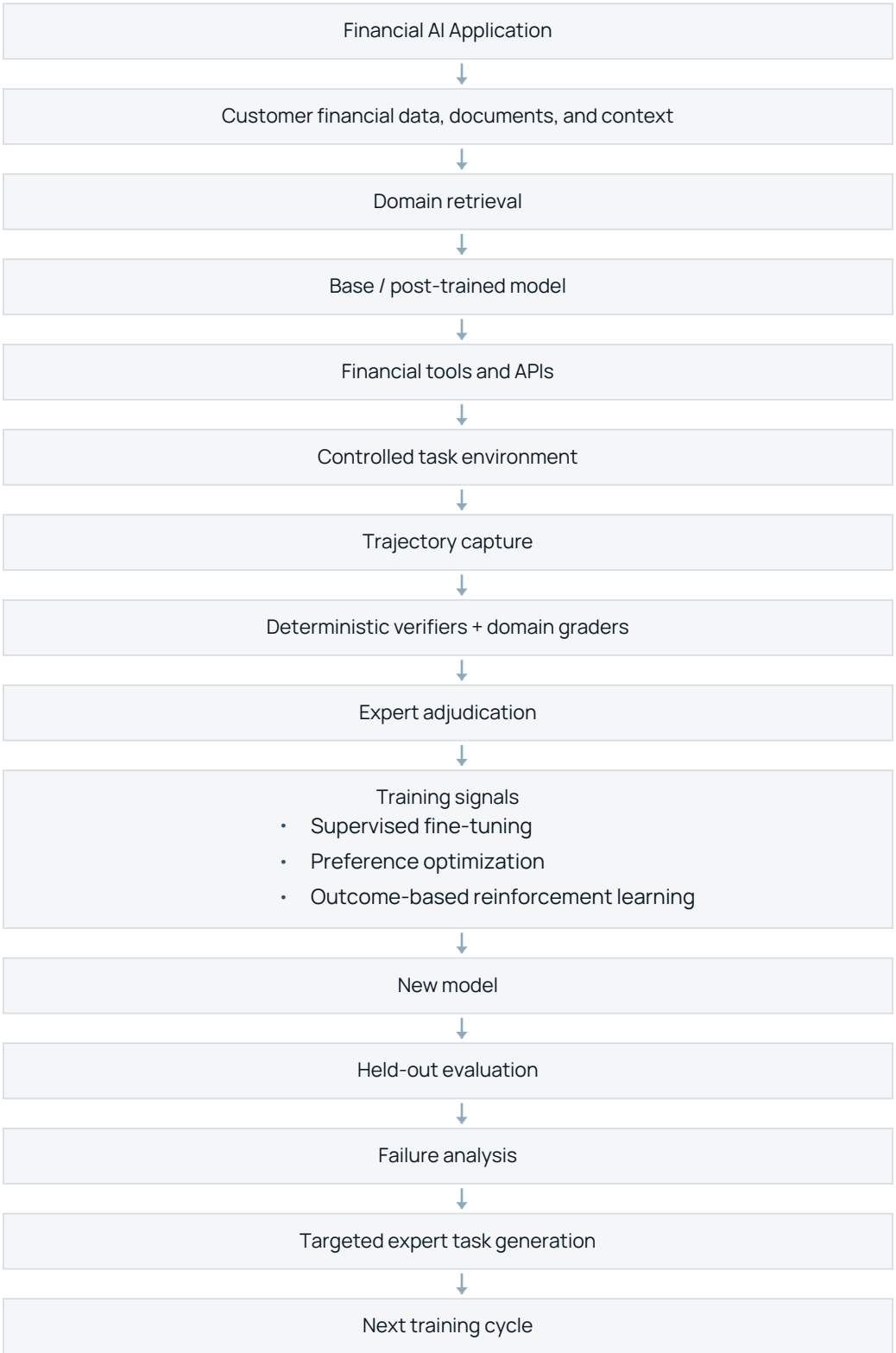
# Appendix C. Experimental Matrix

TABLE 06

| Configuration | Expert SFT | Preference | Environment Trajectories | Outcome-Based Optimization |
|---|---|---|---|---|
| Base | No | No | No | No |
| SFT | Yes | No | No | No |
| SFT + Preference | Yes | Yes | No | No |
| Full Alpheva | Yes | Yes | Yes | Yes |

The purpose of this matrix is to isolate the incremental contribution of each training signal.

# Appendix D. Agent Environment Architecture



# Appendix E. Core Training Loop

The complete system can be summarized as:

1. Observe

Capture real financial workflows and model behavior.

2. Diagnose

Identify where the model fails and classify the failure.

3. Decompose

Determine whether the failure is factual, computational, contextual, procedural, judgmental, or tool-related.

4. Generate

Create targeted expert tasks corresponding to the failure.

5. Encode

Convert the task into demonstrations, preferences, trajectories, verification rules, or evaluation cases.

6. Train

Apply supervised fine-tuning, preference optimization, and outcome-based post-training.

7. Evaluate

Measure performance on fixed held-out tasks and targeted failure cases.

8. Repeat

Feed residual failures back into the expert-data generation process.

The resulting system is not a one-time training run.

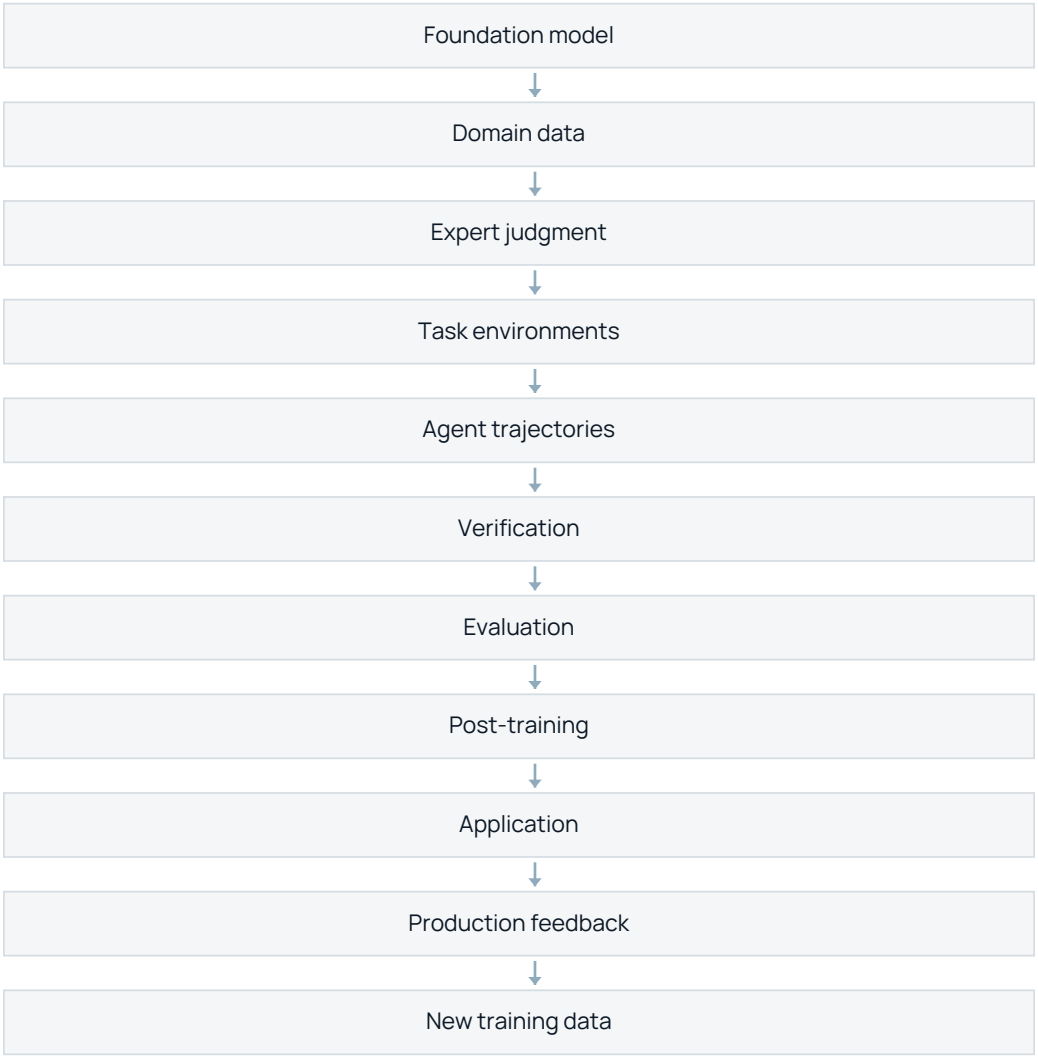
It is a continuously improving domain-specific model development loop.

# Appendix F. The Strategic Thesis

The conventional vertical AI stack is often described as:

Foundation model → retrieval → application → workflow

The emerging vertical AI stack is broader:




The competitive asset increasingly becomes the loop itself.

The company that owns the workflow can observe where the model fails.

The company that owns the experts can understand why it fails.

The company that owns the environments can reproduce the failure.

The company that owns the evaluators can measure the failure.

And the company that owns the post-training pipeline can turn the failure into improved model behavior.

That is the foundation of a vertical AI training moat.

## Final Takeaway

> The next generation of vertical AI will not be differentiated only by the applications they build. It will also be differentiated by the training systems they own.

> Alpheva builds those systems.