JUXT AI Radar

An Opinionated Guide to The AI Landscape 2025

JUXT | A GRID DYNAMICS COMPANY

An Opinionated Guide to The AI landscape from JUXT, a Grid Dynamics Company

Welcome to the first edition of the JUXT AI Radar, where we map the landscape of AI tools, technologies, frameworks, and practices based on our collective expertise and client experiences. Our committee of technology experts has carefully evaluated each entry based on real-world applications, industry trends, and practical utility. This radar represents our current viewpoint and will evolve as the rapidly changing AI ecosystem matures.

Radar Overview

Our radar is organized into four main categories, each containing technologies evaluated across four adoption levels:

Adopt: Technologies we recommend using now

Trial: Worth exploring for new projects

Assess: Keep under observation

Hold: Not recommended for new projects

Categories

Techniques
AI methodologies, approaches, and practices that shape how we build intelligent systems.
Adopt, Trial, Assess, Hold

Languages & Frameworks
Programming languages, libraries, and frameworks that power AI development.
Adopt, Trial, Assess, Hold

Tools
Software tools and utilities that enhance AI development workflows.
Adopt, Trial, Assess, Hold

Platforms
Infrastructure and platform services that support AI applications.
Adopt, Trial, Assess, Hold

Contributors

This radar represents our current viewpoint and will be updated regularly. We welcome feedback and suggestions from the community. Each technology entry includes detailed reasoning for its placement, helping you make informed decisions for your AI projects.

Henry Garner

Henry is a principal engineer at JUXT and leader of the AI Chapter. He's implemented AI systems in domains as diverse as education, financial services, and local government in roles spanning data scientist, software engineer and CTO. He's author of the book Clojure for Data Science and maintainer of the open source statistics library kixi.stats.

Ben Halton

Ben is an account manager at JUXT with extensive experience engineering and architecting complex systems. His career spans domains from risk systems in Tier 1 banks to retail recommendation engines and wine trading platforms. His interest in AI is especially in how it can enhance developer experience and productivity.

Denis Lobanov

Denis is a software engineer at JUXT whose technical experience spans developing Linux kernel modules for Satellite communications to distributed graph databases to web backends. He is currently focused on providing a platform that integrates, controls and secures LLM interaction.

Oliver Marshall

Oliver is a software engineer at JUXT who specialises in building data processing systems and backend infrastructure. He's recently worked on integrating cutting-edge database technology for a client and approaches AI technologies with healthy skepticism: hopeful about what's possible but focused on what actually works in practice.

Neale Swinnerton

Neale is a principal engineer at JUXT. He's spent his career in software development across many domains. He sees himself as an engineer more than a scientist, advising teams how to use pragmatic workflows to improve developer productivity and joy.

Chris Williams

Chris is a software engineer at JUXT with broad experience across industries and technologies. He has long seen automated testing as a superpower for building reliable systems, and now views large language models as the next tool for boosting productivity while supporting real learning.

Radar at a glance

Languages and Frameworks

ADOPT

1. PyTorch
2. dbt
3. MCP

TRIAL

4. AutoGen
5. A2A
6. DeepEval
7. LlamaIndex

ASSESS

8. Prolog
9. JAX
10. LangChain & LangGraph
11. PydanticAI
12. Smolagents
13. CrewAI

HOLD

14. TensorFlow
15. Keras
16. R
17. OpenCL

Techinques

ADOPT

18. Classical ML
19. RAG
20. LLM-as-a-judge
21. BERT variants
22. Few-shot prompting

TRIAL

23. Cross-encoderreranking
24. Chain of thought (CoT)
25. Model distillation & synthetic data
26. UMAP

ASSESS

27. Structured RAG
28. Hypothetical document embeddings (HyDE)
29. Fine-tuning with LoRA
30. Agentic tool use

HOLD

31. Word2Vec & GloVe
32. t-SNE
33. Zero-shot prompting
34. AI pull request review

Radar at a glance

Tools

ADOPT

35. Software engineering copilots
36. Provider-agnostic LLM facades
37. Notebooks

TRIAL

38. MLflow
39. Vector databases
40. Local model execution environments

ASSESS

41. AI application bootstrappers
42. Agentic computer use
43. Lakera

HOLD

44. Conversational data analysis

Platforms

ADOPT

45. Weights & Biases
46. Foundation models
47. Data pipeline orchestration tools
48. Cloud model hosting platforms

TRIAL

49. Production AI monitoring platforms
50. Open weight LLMs
51. AI-powered workflow automation platforms

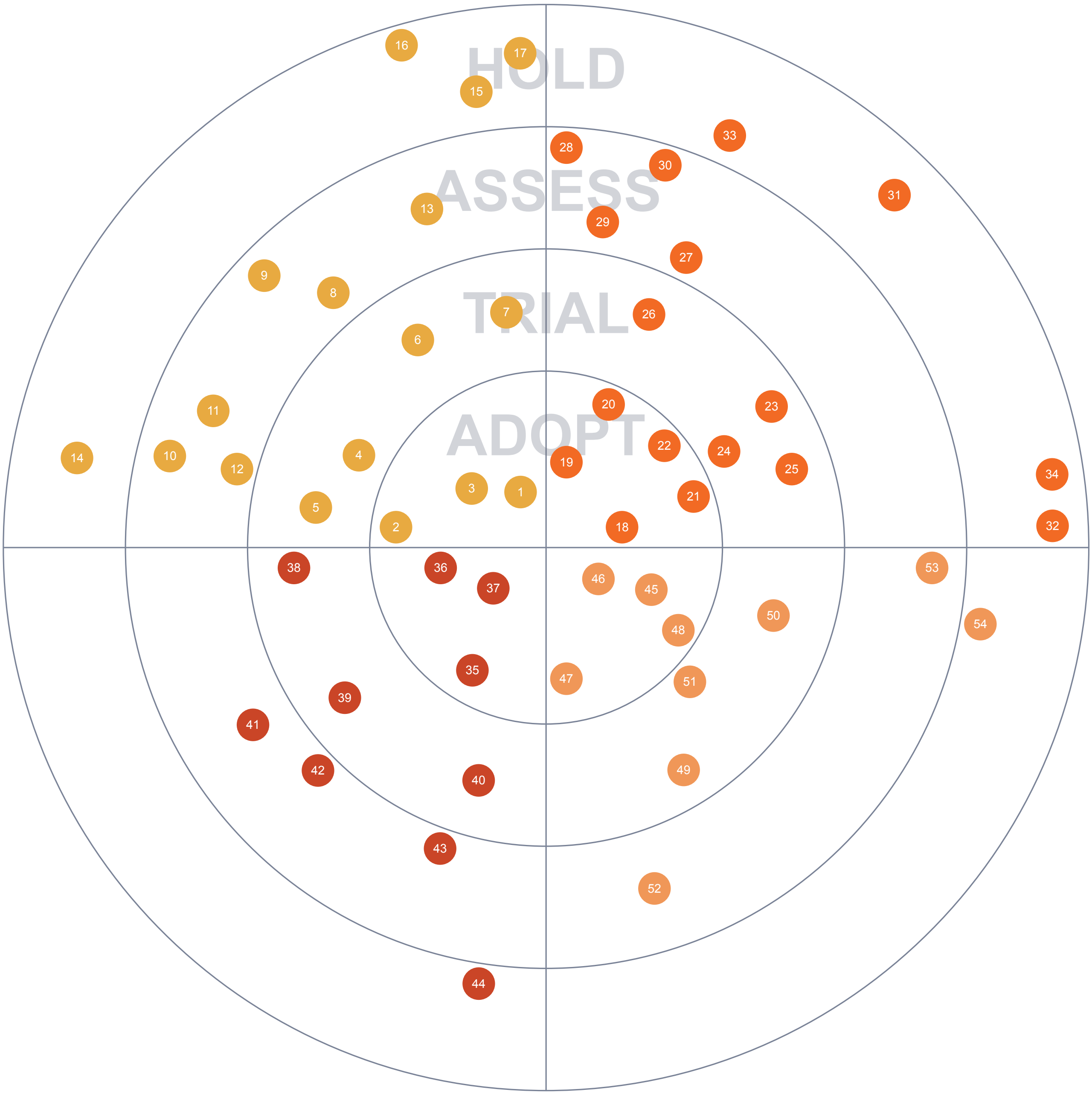
ASSESS

52. Galileo
53. Kubeflow

HOLD

54. Building against vendor-specific APIs

The Radar



HOLD
ASSESS
TRIAL
ADOPT
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54

Languages and Frameworks

Programming languages and frameworks form the backbone of AI development, providing the tools and abstractions needed to build intelligent systems. From established libraries to emerging frameworks, these technologies enable developers to create sophisticated AI applications efficiently.

ADOPT

1. PyTorch
2. dbt
3. MCP

TRIAL

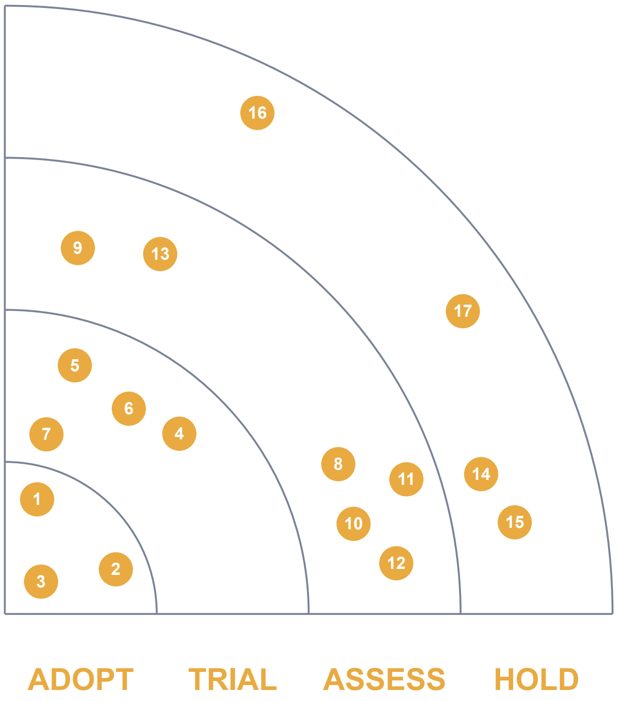
4. AutoGen
5. A2A
6. DeepEval
7. LlamaIndex

ASSESS

8. Prolog
9. JAX
10. LangChain & LangGraph
11. PydanticAI
12. Smolagents
13. CrewAI

HOLD

14. TensorFlow
15. Keras
16. R
17. OpenCL



Languages and Frameworks

Adopt

These languages and frameworks represent mature, well-supported technologies that are ready for production use. They offer excellent performance, extensive ecosystems, and proven track records in real-world applications.

PyTorch

PyTorch has demonstrated consistent maturity and widespread adoption across both research and production environments, earning its place in our Adopt ring. We're seeing it emerge as the default choice for many machine learning teams, particularly those working on deep learning projects, thanks to its intuitive Python-first approach and dynamic computational graphs that make debugging and prototyping significantly easier.

The framework's robust ecosystem, exceptional documentation and strong community support make it a reliable choice for teams at any scale. While TensorFlow remains relevant, particularly in production deployments, PyTorch's seamless integration with popular machine learning tools, extensive pre-trained model repository and growing deployment options through TorchServe have addressed previous concerns about production readiness. The framework's adoption by major technology organisations and research institutions, coupled with its regular release cycle and stability, gives us confidence in recommending it as a default choice for new machine learning projects.

dbt

We've placed dbt (data build tool) in the Adopt ring because it has proven to be an essential framework for organising and managing the data transformations that feed AI systems. dbt brings software engineering best practices like version control, testing, and documentation to data transformation workflows, which is crucial when preparing data for AI model training and inference.

The reliability and maintainability of AI systems heavily depend on the quality of their input data, and dbt helps teams achieve this by making data transformations more transparent and trustworthy. We've seen teams successfully use dbt to create clean, well-documented data pipelines that connect data warehouses to AI applications, while maintaining the agility to quickly adapt to changing requirements. Its integration with modern data platforms and strong community support make it a solid choice for organisations building out their AI infrastructure.

MCP

Anthropic's Model Context Protocol (MCP) has rapidly gained adoption since its introduction, addressing the critical need for standardised integration between language models and external tools. We've placed MCP in the Adopt ring based on its practical utility and straightforward implementation process.

MCP solves the persistent problem of connecting AI models to organisational data and tools without requiring custom integration work for each connection. The protocol's popularity stems from how straightforward MCP servers are to create and deploy, our team has successfully built functional MCP servers within a matter of hours. This ease of implementation, combined with the growing ecosystem of community-created servers, significantly reduces development overhead.

For organisations evaluating MCP, the value proposition is clear: rather than building bespoke integrations between AI assistants and internal systems, teams can leverage existing MCP servers or create new ones following established patterns. The protocol handles context management and tool discovery effectively, enabling models to reason appropriately about available capabilities.

We recommend starting with existing MCP servers that match your requirements before building custom implementations. The protocol's design encourages reusability, meaning investments in MCP server development can benefit multiple AI applications across your organisation.

Trial

These languages and frameworks show promising potential with growing adoption and active development. While they may not yet have the same maturity as Adopt technologies, they offer innovative approaches and capabilities that make them worth exploring for forward-thinking teams.

AutoGen

We've placed AutoGen in the Trial ring based on its promising approach to orchestrating multiple AI agents for complex problem-solving. This Microsoft-developed framework enables developers to create systems where AI agents can collaborate, dividing tasks between specialised roles like coding, testing, and reviewing, similar to how human development teams operate. While still evolving, we've seen compelling early results from teams using AutoGen to build more sophisticated AI applications, particularly in scenarios requiring multi-step reasoning or specialised domain knowledge.

The framework's ability to handle interaction patterns between agents with built-in error handling and recovery shows particular promise for enterprise applications. However, we recommend carefully evaluating its fit for your specific use case, as the overhead of managing multiple agents may not be justified for simpler applications where a single large language model would suffice. We're also watching how the framework's approach to agent coordination evolves as the field matures.

A2A

Google's Agent2Agent (A2A) protocol addresses the emerging need for standardised communication between AI agents in multi-agent systems. Launched in April 2025 and now governed by the Linux Foundation, A2A enables agents from different providers to discover each other's capabilities, delegate tasks, and collaborate on complex workflows without requiring custom integration work.

The protocol complements rather than competes with Model Context Protocol. Whilst MCP focuses on connecting AI models to tools and data sources, A2A specifically handles agent-to-agent communication. This distinction becomes important as organisations move towards multi-agent architectures where specialised agents collaborate to accomplish complex tasks requiring diverse capabilities.

A2A's design centres around "Agent Cards" that advertise capabilities in JSON format, enabling dynamic task delegation between agents. The protocol supports various modalities including text, audio, and video streaming, with built-in security features for enterprise deployment. Industry backing from over 150 organisations, including major hyperscalers, technology providers, and consulting firms, suggests strong momentum for adoption.

We've placed A2A in Trial because whilst the protocol shows clear potential and has impressive industry support, it remains relatively new with limited production deployment patterns. Early implementations suggest promise for organisations building complex multi-agent systems, but teams should evaluate whether their use cases truly require agent-to-agent communication versus simpler architectures. For most organisations, starting with MCP for tool integration before exploring A2A for multi-agent scenarios represents a sensible progression path.

DeepEval

We've placed DeepEval in the Trial ring as it addresses a critical gap in AI application development: the systematic evaluation of Large Language Model outputs. While traditional software testing frameworks focus on deterministic outcomes, DeepEval provides a comprehensive toolkit for assessing the reliability, accuracy and consistency of AI-generated content.

The framework stands out for its practical approach to testing LLM applications, offering built-in metrics for evaluating responses across dimensions like relevance, toxicity and factual accuracy. What particularly impressed our committee was its ability to handle both unit and integration testing scenarios, making it valuable for teams building production-grade AI systems. However, we recommend starting with smaller, non-critical components first, as best practices around LLM testing are still emerging and the framework itself is relatively new to the ecosystem.

LlamaIndex

LlamaIndex, formerly known as GPT Index, is a framework that supports developers in connecting large language models with external data sources in a structured way. It provides tools to build indices, data structures that help LLMs access relevant information efficiently, thereby improving their ability to handle specific tasks requiring contextual or domain-specific data.

We consider LlamaIndex suitable for teams trialling methods to augment LLM performance, especially in data-centric applications. While its modular design and focus on customisation are appealing, its relative maturity as a toolkit means that teams may encounter challenges around documentation, setup, or adapting it to complex datasets. As with many emerging tools, its value depends on careful experimentation and matching it to the right problem space.

Assess

These languages and frameworks represent emerging or specialized technologies that may be worth considering for specific use cases. While they offer interesting capabilities, they require careful evaluation due to limited adoption, specialized requirements, or uncertain long-term viability.

Prolog

We've placed Prolog in the Assess ring of our languages quadrant due to its renewed relevance in AI development, particularly for adding structured logical reasoning capabilities to Large Language Model applications, and decoupling logic from procedure. Prolog (and logic programming in general) may offer significant value due to its ability to extract from and represent knowledge graphs, which have a well-studied symbiotic relationship with LLMs, allowing us to couple the versatility of LLMs with the ability to have a concrete expert knowledge base to prevent hallucinations, reify concrete rules, etc. This also can allow LLMs to produce consumable data for further engineering needs, and allows us to express preferences in our systems in unambiguous ways. The use of such expert systems alongside LLMs has been likened to Kahneman's system 1 and 2. Finally, the metaprogramming & dynamic capabilities of Prolog are extremely strong.

While Prolog has been around since the 1970s, we're seeing interesting experiments where developers combine its powerful symbolic reasoning with modern LLMs to create more robust and explainable AI systems, by leveraging Prolog as a reasoning agent. However there are challenges around performance, as well as some redundancy in knowledge graphs given the existence of semantic web languages such as RDF, OWL, SPARQL, etc. Prolog is also not the only language of its kind– there are many kinds of logic language, which are all fundamentally different from each other (E.G., some are used for induction as in SATs, some don't use the same kinds of logic), though this does not necessarily discount Prolog's utility. Since Prolog interoperates extremely well with most other programming languages, it can also be embedded within applications rather easily.

The renewed interest doesn't yet warrant a higher ring placement, as adoption patterns are still emerging and the tooling ecosystem needs maturation. However, we believe technical teams should assess Prolog's potential, especially for projects where transparent logical reasoning needs to be combined with LLM capabilities. Teams working on applications in regulated industries or those requiring auditable decision paths may find particular value in exploring this approach. At the very least, surveying Prolog provides insight into the possibilities of where historical findings might enrich the current space.

JAX

We've placed JAX in our Assess ring as we observe increasing interest in this ML framework that combines NumPy's familiar API with hardware acceleration and automatic differentiation. While TensorFlow and PyTorch remain dominant in the ML ecosystem, we're seeing JAX gain traction particularly in research settings and among teams working on custom ML architectures.

What interests us about JAX is its functional approach to ML computation and its ability to compile to multiple hardware targets through XLA (Accelerated Linear Algebra). The framework shows promise for projects requiring high-performance numerical computing, though we suggest careful evaluation of its relative immaturity in areas like deployment tooling and the smaller ecosystem of pre-built components compared to more established frameworks. We recommend teams experimenting with JAX do so on research projects or contained proofs-of-concept before considering broader adoption.

LangChain & LangGraph

We've placed LangChain and its companion LangGraph in the Assess ring as they represent an emerging approach to building applications with Large Language Models. These frameworks provide structured ways to compose AI capabilities into more complex applications, with LangChain focusing on general-purpose AI interactions and LangGraph extending this to handle more sophisticated multi-step processes.

While these tools have gained significant adoption and show promise in reducing boilerplate code when working with LLMs, we recommend careful evaluation before widespread use. The rapid pace of change in the underlying AI platforms means that some of LangChain's abstractions may become outdated or less relevant as the ecosystem evolves. We've observed teams successfully using these frameworks for prototypes and smaller production systems, but also encountering challenges when requirements grow more complex or when they need to debug unexpected behaviours. Consider starting with focused experiments that test whether these tools truly simplify your specific use case rather than assuming they're the right choice for all AI development.

PydanticAI

We've placed PydanticAI in the Assess ring of our Languages & Frameworks quadrant because it represents a promising approach to building AI applications that merits closer examination, while not yet being broadly proven in production environments.

PydanticAI brings the well-regarded developer experience of FastAPI to generative AI application development. Built by the team behind Pydantic (which has become a foundation for many AI frameworks including OpenAI SDK, Anthropic SDK, LangChain, and others), it offers a familiar, Python-centric approach to building LLM-powered applications. The framework provides important features like model-agnostic support across major LLM providers, structured responses through Pydantic validation, and a dependency injection system that facilitates testing.

What particularly interests us is how PydanticAI leverages existing Python patterns and best practices rather than introducing completely new paradigms. This could significantly lower the learning curve for developers working with AI. However, as a relatively new framework in a rapidly evolving space, we're placing it in Assess while we watch for broader adoption, community growth, and production-proven implementations across different use cases. Organisations with Python-based stacks and teams familiar with FastAPI or Pydantic should consider evaluating PydanticAI for their AI application development needs.

Smolagents

We've placed smolagents in the Assess ring of the Languages & Frameworks quadrant based on our evaluation of its current state and potential.

This lightweight agent framework takes a minimalist approach with its core codebase of under 1,000 lines. Early feedback suggests it can be effective for quickly prototyping agentic concepts before transitioning to more robust frameworks like AutoGen or LangGraph for production implementations. The framework's code-based agent approach, where agents execute actions as Python code snippets, appears to reduce the number of steps and LLM calls in certain scenarios, though this comes with inherent security considerations.

We've positioned smolagents in Assess rather than Trial for several reasons: it lacks extensive production validation, the security implications of code execution require careful evaluation, and while benchmark results with models like DeepSeek-R1 are interesting, we need to see more diverse real-world implementations. Teams exploring agent architectures should evaluate whether SmolaGents' approach aligns with their specific needs and security requirements, whilst recognising its limitations for production-grade systems.

CrewAI

We've placed CrewAI in the Assess ring of the Languages & Frameworks quadrant because it represents a promising approach to multi-agent orchestration that's gaining traction among developers building complex AI systems.

Crew.ai provides a framework for creating teams of specialised AI agents that work together to accomplish tasks through coordinated effort. Our team members report that it offers a well-structured approach to defining agent roles, communication patterns, and task delegation: addressing many of the challenges involved in building effective agentic systems. The framework's emphasis on human-in-the-loop integration, along with the ability to combine specialised agents with different capabilities, makes it particularly valuable for complex workflows where single-agent solutions fall short.

While Crew.ai shows significant promise and has already been used successfully in production environments, we've placed it in Assess rather than Trial because the multi-agent paradigm itself is still evolving. Organisations need to carefully evaluate whether the added complexity of managing multiple agents offers sufficient benefits over simpler approaches for their specific use cases. Teams should also be aware that best practices for agent collaboration are still emerging, and implementations may require considerable tuning and oversight to achieve reliable results.

Languages and Frameworks

Hold

These languages and frameworks are not recommended for new projects due to declining relevance, better alternatives, or limited long-term viability. While some may still have niche applications, they generally represent technologies that have been superseded by more effective solutions.

TensorFlow

We have placed TensorFlow in the Hold ring for several reasons. While TensorFlow remains a capable deep learning framework that helped popularise machine learning at scale, we're seeing teams struggle with its steep learning curve and complex deployment story compared to more modern alternatives. The framework's verbose syntax and intricate architecture often lead to longer development cycles, particularly for teams new to machine learning.

PyTorch has emerged as the clear community favourite for both research and production deployments, with a more intuitive programming model and better debugging capabilities. Additionally, with the rise of AI platforms that abstract away much of the underlying complexity, many teams no longer need to work directly with low-level frameworks like TensorFlow. For new projects, we recommend exploring higher-level tools or PyTorch unless there are compelling reasons to use TensorFlow, such as maintaining existing deployments or specific requirements around TensorFlow Extended (TFX) for ML pipelines.

Keras

We have placed Keras in the Hold ring primarily due to its transition from a standalone deep learning framework to becoming more tightly integrated with TensorFlow, along with the emergence of more modern alternatives that offer better developer experiences.

While Keras served as an excellent entry point for many developers into deep learning, providing an intuitive API that made neural networks more accessible, the landscape has evolved significantly. Frameworks like PyTorch have gained substantial momentum, offering clearer debugging, better documentation and a more Pythonic approach. Additionally, recent high-level frameworks such as Lightning and FastAI provide similar ease-of-use benefits while maintaining closer alignment with current best practices in deep learning development. For new projects, we recommend exploring these alternatives rather than investing in Keras-specific expertise.

R

Despite R's historical significance in data science and statistical computing, we've placed it in the Hold ring for new projects. While R remains capable for statistical analysis and data visualisation, we're seeing its adoption declining in favour of Python's more comprehensive ecosystem for machine learning and AI workflows.

The key factors driving this recommendation are the overwhelming industry preference for Python-based ML frameworks, the stronger integration of Python with modern AI platforms and tools, and the challenges of hiring R specialists in today's market. While R retains some advantages for specific statistical applications and academic research, we believe teams starting new AI initiatives will benefit from standardising on Python to maximise their access to cutting-edge AI libraries, tools, and talent.

Languages and Frameworks

OpenCL

We’ve placed OpenCL in the Hold ring of our Languages & Frameworks quadrant. While OpenCL (Open Computing Language) was groundbreaking when introduced as a standard for parallel programming across different types of processors, we believe teams should look to alternatives for new projects.

Despite its promise of write-once-run-anywhere code for GPUs, CPUs, and other accelerators, OpenCL has seen declining industry support and faces significant challenges. Major hardware vendors have shifted their focus to more specialised frameworks like CUDA for NVIDIA hardware, while newer alternatives such as SYCL and modern GPU compute frameworks offer better developer experiences with similar cross-platform benefits. The complexity of the OpenCL programming model, combined with inconsistent tooling support and a fragmented ecosystem, makes it increasingly difficult to justify for new development compared to more actively maintained alternatives.

Techniques

AI methodologies, approaches, and practices that shape how we build intelligent systems.

ADOPT

18. Classical ML
19. RAG
20. LLM-as-a-judge
21. BERT variants
22. Few-shot prompting

TRIAL

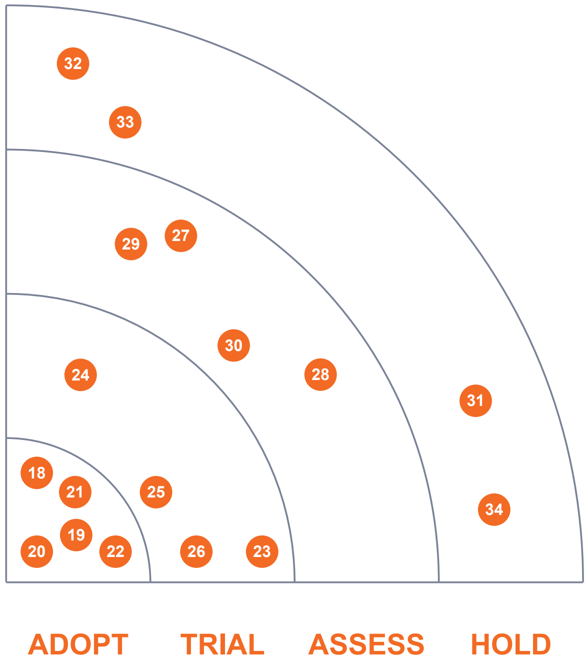
23. Cross-encoderreranking
24. Chain of thought (CoT)
25. Model distillation & synthetic data
26. UMAP

ASSESS

27. Structured RAG
28. Hypothetical document embeddings (HyDE)
29. Fine-tuning with LoRA
30. Agentic tool use

HOLD

31. Word2Vec & GloVe
32. t-SNE
33. Zero-shot prompting
34. AI pull request review



Techniques

Adopt

These techniques represent mature, well-supported approaches that are ready for production use. They offer excellent performance, extensive documentation, and proven track records in real-world applications.

Classical ML

We continue to see tremendous value in classical machine learning approaches like random forests, gradient boosting (XGBoost, LightGBM), linear/logistic regression and support vector machines for many business problems. While attention has shifted dramatically towards deep learning and large language models in the last couple of years, these traditional techniques often provide the best balance of explainability, computational efficiency, and performance for structured data problems.

The key advantages that keep classical ML firmly in our Adopt ring include faster training times, lower computing requirements, and easier deployment compared to deep learning approaches. However, it's important to recognise that realising these benefits requires both quality training data and staff with appropriate expertise. Unlike the recent wave of LLM-based solutions that have democratised AI capabilities for organisations without extensive data science teams, classical ML continues to demand specialised knowledge in feature engineering, model selection, and evaluation.

For organisations with the necessary data assets and technical capabilities, these methods work well even with the smaller datasets common in enterprise settings, often matching or exceeding the performance of more complex approaches while remaining more interpretable to stakeholders and easier to maintain. Their lower training costs, smaller carbon footprint, and built-in feature importance metrics provide practical advantages that directly translate to business value, particularly as organisations face increasing pressure to make their ML systems both cost-effective and environmentally sustainable.

RAG

Retrieval-Augmented Generation (RAG) is an AI approach that combines search and text generation to produce more accurate responses. The approach helps prevent confabulation, cases where AI models generate plausible but incorrect information, by grounding responses in real data.

We're placing RAG in the Adopt ring because it addresses key challenges in deploying AI systems in information retrieval contexts. The technique is particularly valuable when accuracy and traceability of information are crucial, such as in customer service, technical documentation, or compliance scenarios. While implementing RAG requires careful attention to document processing and embedding strategies, the widespread availability of tools and frameworks has significantly lowered the barriers to adoption. Teams should consider RAG as a foundational technique when building AI applications that need to leverage organisational knowledge.

We're particularly interested in monitoring how this technique develops alongside others improving AI system reliability and truthfulness. For example, by augmenting the approach with Self-RAG to recognise when more evidence needs to be gathered, conflicting information verified, or responses refined for better accuracy. This 'self-criticism' mechanism has shown promising results in improving response quality and reducing hallucinations.

See also Cross-encoder reranking, Chain of thought, Structured RAG.

Techniques

LLM-as-a-Judge

We've placed LLM-as-a-judge in the Adopt ring because it has quickly proven itself to be one of the most practical and cost-effective techniques for evaluating AI system outputs. At first glance, it might seem like circular reasoning to have one LLM evaluate another LLM's work. However, the capabilities of today's strongest models are such that they can provide nuanced, multidimensional critique that simpler evaluation methods cannot match, except when using very constrained metrics like exact match or BLEU scores (Bilingual Evaluation Understudy, a method for automatically evaluating machine translations).

This technique has become widely adopted in both offline and online evaluation scenarios. In offline evaluation, it scales far better than human assessment, allowing teams to test thousands of outputs quickly during development and quality assurance workflows. In online scenarios, an LLM judge can evaluate another LLM's output in real-time in production, enabling dynamic workflow adjustments or user experience modifications based on quality assessments. This real-time evaluation approach serves as a foundation for more sophisticated agentic workflows, where multiple AI components collaborate to refine outputs before user delivery.

Recent research demonstrates that the current frontier models can provide judgements that correlate strongly with human preferences across many common evaluation dimensions. For best results, we recommend using a different LLM as the judge than the one being evaluated, and viewing this approach as an augmentation to, not replacement for, human evaluation. The strongest LLMs can identify nuanced issues in reasoning, factuality, and tone that would otherwise require substantial human review time, creating a more efficient evaluation pipeline whilst preserving critical human oversight for final quality assurance.

BERT variants

Bidirectional Encoder Representations from Transformers (BERT) revolutionised Natural Language Processing (NLP) by allowing AI models to process human language by looking at words in relation to their entire context, rather than just left-to-right or right-to-left. Think of it like a reader who can understand a word by looking at all the surrounding words for context, rather than reading sequentially. The original BERT spawned a family tree of variants, with ModernBERT representing the latest evolution. Released in late 2024, ModernBERT improves legacy BERT through architectural updates which shorten training times and improve accuracy.

BERT-style models serve fundamentally different purposes than generative models like GPT. While GPT models excel at generating text and conversational interactions, BERT models are optimised for understanding and analysis tasks such as classification, named entity recognition, and sentiment analysis. They're particularly valuable for creating semantic vector embeddings that capture text meaning in numerical form, making them essential components in Retrieval Augmented Generation (RAG) systems. In these pipelines, BERT embeddings help retrieve relevant information that is then fed as text to GPT models for generation: the models don't directly share embeddings, but rather work in complementary roles.

We particularly recommend DeBERTa for organisations starting new NLP projects. It handles word relationships more effectively using a disentangled attention mechanism and enhanced position encoding. DistilBERT is smaller and faster whilst retaining most of the model's performance, so it is particularly valuable for production deployments where latency requirements are strict or computing resources are limited, such as edge devices or high-throughput API services.

For organisations choosing between BERT and GPT models, consider your specific use case: BERT models require fewer computational resources for inference and excel at precise understanding tasks, while GPT models offer impressive out-of-the-box generation capabilities through accessible APIs. Many sophisticated AI applications today use both types in complementary roles, BERT for understanding and information retrieval, and GPT for generation based on that understanding.

There are options for specialised domains like biomedical (BioBERT) or financial text (FinBERT). While these can outperform general models in their niches, they often require significant expertise to use effectively and may need additional tuning for specific use cases.

Few-shot prompting

The technique of providing examples to guide an AI model's responses has proven consistently effective across different Large Language Models. By showing the model a few examples of desired input-output pairs, developers can achieve more reliable, consistent, and contextually appropriate responses without resorting to complex prompt engineering or fine-tuning.

The method's strength lies in its simplicity and portability across different AI platforms. Our team members report significantly improved results when moving from zero-shot (no examples) to few-shot approaches, particularly for tasks requiring specific formats, technical terminology, or domain expertise. While the optimal number of examples varies by use case, we typically see diminishing returns beyond 3-5 examples. The main trade-off to consider is token consumption, as each example uses up context window space that could be used for other content.

Trial

These techniques show promising potential with growing adoption and active development. While they may not yet have the same maturity as Adopt techniques, they offer innovative approaches and capabilities that make them worth exploring for forward-thinking teams.

Cross-encoder reranking

Cross-encoder reranking sits in our Trial ring as a promising enhancement for AI search and chat systems. It works alongside traditional embedding-based search (where documents and queries are converted into numbers that represent their meaning) by taking a closer look at the initial search results. While embedding search is fast and good at finding broadly relevant content, cross-encoder reranking excels at understanding subtle relevance signals by looking at the query and potential results together.

Most teams we've observed use this as a two-step process: first, a quick embedding search finds perhaps 50-100 potentially relevant items from their knowledge base. Then, cross-encoder reranking carefully sorts these candidates to bring the most relevant ones to the top. While this additional step does add some processing time, we're seeing it deliver meaningful improvements in result quality across various use cases.

The technique has shown consistent improvements across different domains and use cases, often reducing hallucinations in downstream LLM responses by ensuring higher quality context selection. Implementation has also become more straightforward with libraries like sentence-transformers providing ready-to-use models. However, teams should be mindful of the additional latency introduced by the reranking step and may need to tune the number of candidates passed to the reranker based on their specific performance requirements. The computational overhead is generally justified by the marked improvement in retrieval quality, making this a reliable enhancement to any RAG pipeline where response accuracy is a priority.

Chain of thought (CoT)

Chain of Thought (CoT) sits in our Trial ring as a proven technique for improving the reasoning capabilities of large language models, where they are required.

This technique involves prompting an AI model to show its step-by-step reasoning process rather than jumping straight to a conclusion. Think of it like asking a student to show their working when solving a problem, rather than just writing down the final answer. Essentially, CoT encourages models to explain their thought process in a structured manner, rather than jumping directly to a conclusion. This has shown to be especially useful in tasks that require complex reasoning, such as mathematical problem-solving or logical inference.

We've placed CoT in the Trial ring because it has shown promising results in improving the interpretability and accuracy of AI responses when faced with complex tasks. However, it's worth noting that CoT typically requires more tokens (and thus more cost) than direct prompting, and isn't always necessary for simple tasks. We recommend using it selectively where the complexity of the task warrants the additional computation and cost. Newer 'reasoning' models such as o1 and o3 are specifically built to work with CoT behind the scenes and have very impressive benchmarks at logic/coding tests at the cost of being quite slow and expensive.

We're keeping an eye on related techniques such as LLMs as Method Actors, which achieves similar goals by treating LLMs as actors requiring prompts and cues. However, we caution that this and similar techniques typically require longer, more carefully crafted prompts, which increases token usage and costs. We're also watching for evidence of whether they consistently outperform simpler prompting approaches in production environments.

Model distillation & synthetic data

We've placed Model Distillation in the Trial ring of our Techniques quadrant. Distillation involves training a smaller, more efficient model to mimic a larger one. A common emerging pattern we're seeing is using LLMs to generate synthetic training data for this smaller model. The larger LLM acts as a "teacher," creating diverse, high-quality examples that can help the "student" model learn the desired behaviour. For instance, a large model might generate thousands of question-answer pairs that are then used to train a more compact model for a specific domain.

This creates an interesting synergy: the large LLM's ability to generate varied, nuanced responses helps create richer training datasets than might otherwise be available, while distillation makes the resulting solutions more practical to deploy. This approach makes AI deployment more practical and cost-effective, especially for edge devices or resource-constrained environments. However, we're keeping it in trial as the process still requires considerable expertise to execute well. Teams need to carefully validate the quality of generated training data and ensure the distilled model maintains acceptable performance levels. There's also ongoing debate about potential amplification of biases or errors through this approach.

Be sure to check the licence of the model you're using for distillation. Llama forbids the use of its output to train other models. The launch of DeepSeek R1 in January 2025 brought distillation into popular consciousness, as it has been widely assumed that it represents a distillation of existing Foundation models.

Techniques

UMAP

UMAP (Uniform Manifold Approximation and Projection) enters our Trial ring as a promising dimensionality reduction technique that's gaining traction in the AI community. While t-SNE has been the go-to choice for visualising high-dimensional data, UMAP offers better preservation of global structure and runs significantly faster, making it particularly valuable for large-scale AI applications like exploring embedding spaces and analysing neural network activations.

We're seeing successful applications of UMAP across several AI projects, especially in combination with clustering algorithms for understanding large language model behaviours and exploring semantic relationships in vector spaces. However, we recommend starting with smaller, well-understood datasets when first adopting UMAP, as its parameters can be sensitive and require careful tuning to avoid misleading visualisations. The technique shows enough promise and maturity to warrant serious evaluation, though teams should be prepared to invest time in understanding its mathematical foundations to use it effectively.

The Python UMAP library provides extensive documentation and explanation. There are also libraries for Rust, Java, and R among others.

Assess

These techniques represent emerging or specialized approaches that may be worth considering for specific use cases. While they offer interesting capabilities, they require careful evaluation due to limited adoption, specialized requirements, or uncertain long-term viability.

Structured RAG

Structured RAG extends basic RAG by organising knowledge in a more formal way, rather than just as chunks of text. Think of it like the difference between a filing cabinet (basic RAG) and a well-designed database (structured RAG). Instead of just retrieving text fragments, structured RAG can work with specific fields, relationships, and hierarchies in your data. For example, in a product catalogue, it could separately track and retrieve product names, prices, specifications, and reviews, understanding how these elements relate to each other.

The key advantages we're seeing in real-world applications include more consistent outputs, better handling of complex queries, and reduced confabulation rates compared to traditional RAG approaches. While implementations can vary, successful patterns are emerging around using JSON schemas, XML structures, or database-like organisations for retrieved information.

However, implementing structured RAG requires more upfront work in data organisation and schema design than traditional RAG. Teams need to carefully consider their data structures and retrieval patterns. This additional complexity is why we've placed it in Assess rather than Trial: while the benefits are clear, implementation patterns are still evolving.

Hypothetical document embeddings (HyDE)

We've found HyDE (Hypothetical Document Embeddings) to be an elegant solution to a common problem in search systems - their tendency to perform poorly when searching content that differs from their training data. HyDE works by first asking a large language model to imagine what an ideal document answering the user's query might look like. This 'hypothetical document' helps bridge the gap between how users naturally ask questions and how information is actually written in documents.

The system creates several of these imagined documents (typically five) to capture different ways the answer might be expressed. These are converted into numerical representations (embeddings) and averaged together. This averaged representation is then used to find real documents that are mathematically similar, which often leads to more relevant search results than traditional methods. The approach has proven particularly effective as part of larger systems, such as RAG (Retrieval Augmented Generation), where accurate document retrieval is crucial for generating reliable responses. Teams should evaluate HyDE particularly for cases where high-precision retrieval is crucial and the additional latency is acceptable.

See also: RAG, BERT

Fine-tuning with LoRA

We have placed Low-Rank Adaptation (LoRA) in the Assess ring. LoRA represents a significant advancement in making AI model customisation more practical and cost-effective. Rather than adjusting all parameters in a large language model (which can number in the billions), LoRA adds a small set of trainable parameters while keeping the original model unchanged. Think of it like teaching an expert to adapt to your specific needs without having to retrain their entire knowledge base. This approach typically reduces the computing resources needed for customisation by 3-4 orders of magnitude while maintaining most of the performance benefits of full fine-tuning.

The technique has proven its value across numerous enterprise applications, and robust tools like Lightning AI’s lit-gpt and axolotl have emerged to support implementation. However, we place it in the Assess ring rather than Trial because successfully applying LoRA still requires significant machine learning expertise and careful consideration of training data quality. Additionally, we caution organisations to view fine-tuning (including with LoRA) as a short-term investment rather than a long-term strategy. Fine-tuning typically ties you to a specific model architecture, and given the rapid pace of AI advancement, tomorrow’s general-purpose models may well outperform your carefully tuned older models with no customisation at all. Migrating fine-tuned weights between different model architectures is particularly challenging and requires a well-curated evaluation corpus. While LoRA is a valuable technique to have in your toolkit, it should only be deployed when the immediate business value clearly outweighs both the technical and opportunity costs.

Agentic tool use

We’ve placed agentic tool use in the Assess ring. This technique involves Large Language Models using external tools and APIs to augment their capabilities beyond pure language processing.

The ability of LLMs to use tools represents a significant advancement in AI system architecture. We’re seeing promising applications where LLMs act as orchestrators, calling specialised tools for tasks like web search, code execution, or API interactions. However, current implementations often struggle with reliability and can make unpredictable tool choices. While frameworks like LangChain, OpenAI’s Function Calling, and standards like Model Context Protocol have made tool use more accessible, organisations should carefully evaluate their specific use cases and implement robust validation mechanisms before deploying tool-using LLMs in production environments.

The decision to place this in Assess reflects both its potential and current limitations. Early adopters are reporting success with contained, well-defined tool sets, particularly in areas like web search and file operations. However, we must emphasise the substantial security risks associated with agentic tool use, especially in environments where malicious actors might attempt to manipulate these systems. It is only a matter of time before poorly secured implementations lead to significant security incidents, with potential for data breaches, unauthorised system access, or service disruption.

When implementing agentic tool use, several key aspects warrant consideration. Tool selection should be limited to essential, well-tested integrations with comprehensive input validation and output verification in place. Organisations must implement strict access controls, rate limiting, and continuous monitoring of tool usage patterns to detect potential misuse or exploitation attempts. All tool-using agents should operate within sandboxed environments with ‘principle of least privilege’ enforcement. Security considerations should be paramount in design decisions, with regular penetration testing to identify vulnerabilities before they can be exploited. Additionally, organisations should plan for graceful fallbacks when tools are unavailable or return unexpected results, ensuring system resilience even when tool interactions fail.

Hold

These techniques are not recommended for new projects due to declining relevance, better alternatives, or limited long-term viability. While some may still have niche applications, they generally represent approaches that have been superseded by more effective solutions.

Word2Vec & GloVe

We’ve placed both GloVe (Global Vectors for Word Representation) and Word2Vec (Word to Vector) in the Hold ring of our techniques quadrant. While these word embedding techniques were groundbreaking when introduced and served as fundamental building blocks for many NLP applications, they have been largely superseded by more advanced approaches.

These older embedding techniques, though computationally efficient, lack the contextual understanding that modern transformer-based models provide. Modern large language models and contextual embeddings like BERT produce more nuanced representations that capture word meaning based on surrounding context, rather than the static embeddings that GloVe and Word2Vec generate. For new projects, we recommend exploring more recent embedding techniques (see “BERT Variants” in our Adopt ring) unless you have very specific constraints around computational resources or model size that make these older approaches necessary.

t-SNE

We’ve placed t-SNE (t-distributed Stochastic Neighbor Embedding) in the Hold ring of our techniques quadrant. While t-SNE was groundbreaking when introduced for visualising high-dimensional data in lower dimensions, particularly for understanding the internal representations of neural networks, we’re seeing its limitations become more apparent in modern AI workflows.

The core issue is that t-SNE can be misleading when interpreting AI model behaviour, as it prioritises preserving local structure at the expense of global relationships. This can lead teams to draw incorrect conclusions about their models’ decision boundaries and feature representations. We’re increasingly recommending alternatives like UMAP (Uniform Manifold Approximation and Projection), which better preserves both local and global structure while offering superior computational performance. For projects requiring dimensionality reduction and visualisation of AI model internals, we suggest exploring these newer techniques rather than defaulting to t-SNE.

Zero-shot prompting

Zero-shot prompting – asking Large Language Models to perform tasks without examples or training – has been a quick way to get started with AI. However, we strongly recommend against using zero-shot prompts in production without appropriate guardrails and safety measures. We’ve heard of multiple incidents where unprotected prompts led to harmful, biased or inappropriate outputs, potentially exposing organisations to significant risks.

Our view is that zero-shot prompting should always be combined with input validation, output filtering and clear usage policies. While it can be valuable for prototyping and exploration, moving to few-shot prompting or fine-tuning with careful guardrails is a more robust approach for production systems. The current placement in “Hold” reflects our concern about organisations rushing to deploy unsafe prompt patterns rather than taking the time to implement proper controls.

AI pull request review

We’ve placed AI Pull Request Review in the Hold ring. Whilst AI tools can catch basic issues like style violations and potential bugs, they fall short in the crucial aspects of PR review that maintain code quality and team effectiveness. The key point is that PR review isn’t just about finding errors: it’s a vital knowledge-sharing mechanism where senior developers mentor juniors, architectural decisions are questioned and refined, and the team maintains a shared understanding of the codebase.

Based on our observations across multiple teams, AI review tools tend to focus on surface-level feedback while missing deeper architectural issues, implementation trade-offs, and business logic errors that human reviewers catch. More concerning is that teams who rely heavily on AI reviews often see a decline in collective code ownership and technical knowledge sharing.

The recent explosion of AI coding assistants has revealed that whilst they are sometimes helpful for tasks like code completion and refactoring, they struggle with higher-level software engineering decisions that require deep context and experience. As one tech lead noted in our research, “AI can tell you if your code follows patterns, but it can’t tell you if you’re using the right patterns in the first place.” Until AI systems can better understand architectural implications and business context, we recommend maintaining human-driven code reviews as a core practice.

Tools

Software tools and utilities that enhance AI development workflows, from coding assistants to data analysis platforms. These tools help developers build, test, and deploy AI applications more efficiently.

ADOPT

35. Software engineering copilots
36. Provider-agnostic LLM facades
37. Notebooks

TRIAL

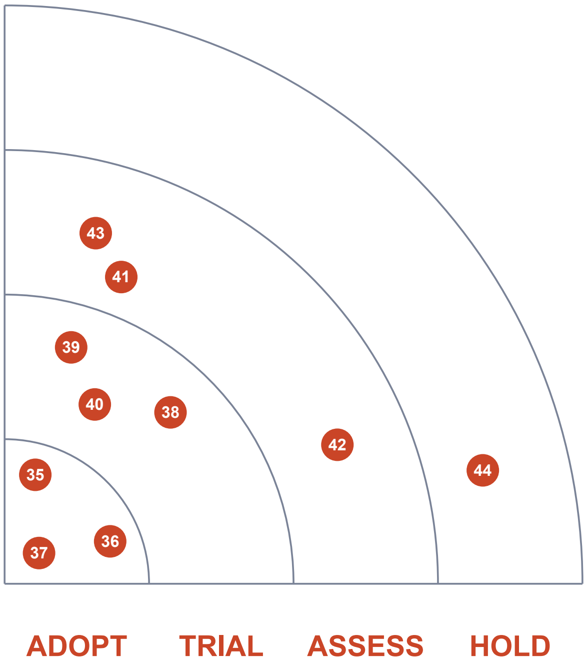
38. MLflow
39. Vector databases
40. Local model execution environments

ASSESS

41. AI application bootstrappers
42. Agentic computer use
43. Lakera

HOLD

44. Conversational data analysis



Tools

Adopt

These tools represent mature, well-supported technologies that are ready for production use. They offer excellent productivity gains, extensive documentation, and proven track records in real-world development workflows.

{/* data-change='up | down | new' */}

Software engineering copilots

AI-powered coding assistants have become essential development tools, spanning traditional IDE integrations like GitHub Copilot and Tabnine, standalone environments such as Cursor, Windsurf, and Zed, and command-line tools including Aider, Cline, Claude Code, and Gemini CLI. Cody focuses on enterprise-scale codebase understanding, Traycer emphasises upfront planning for complex tasks, and Kiro offers both open-ended coding and structured specification-driven development modes, whilst Warp reimagines the terminal experience with AI-enhanced command suggestions.

Two distinct approaches have emerged: free-form "vibe coding" and structured development methodologies. Kiro exemplifies this choice by offering both approaches: a conversational coding mode for rapid iteration and a dedicated specs mode where AI assists developers in drafting requirements, design decisions, and task breakdowns through three specification files before code generation. Cursor enables teams to codify standards through .cursorrules, embedding architectural patterns and guidelines directly into AI assistance.

Usage patterns reveal that senior engineers derive greater value by leveraging AI for routine tasks whilst maintaining quality oversight. Junior developers frequently struggle to evaluate AI suggestions, occasionally accepting flawed implementations or overlooking edge cases. This suggests organisational training requirements around effective AI collaboration.

We've placed Software Engineering Copilots in the Adopt ring based on demonstrable productivity improvements, particularly for experienced developers. Teams report meaningful gains on routine coding tasks, though success correlates with careful workflow integration and rigorous code review practices.

Organisations should implement a "trust but verify" approach: utilise AI assistance for initial implementation whilst maintaining testing standards. The shift towards AI-augmented development appears permanent, making delayed adoption a competitive risk, though teams should remain adaptable as innovation continues across the ecosystem.

Provider-agnostic LLM facades

The LLM landscape evolves rapidly, making today's optimal choice potentially outdated within months. We recommend implementing a facade pattern between your application and LLM providers, rather than building directly against specific APIs. This approach reduces vendor lock-in and enables easier testing of alternative models as they emerge. When considering whether to write your own code, be sure to consider tools such as the lightweight AISuite, Simon Willison's LLM library and CLI tool, or heavyweight alternatives such as LangChain and LlamaIndex.

This recommendation reflects our team's experience seeing projects hampered by tight coupling to specific LLM providers, and the subsequent maintenance burden when transitioning to newer, more capable models.

Notebooks

We've placed Notebooks in the Adopt ring because they have become the de facto standard for data science and machine learning experimentation, prototyping, and documentation. The interactive nature of notebooks, combining code execution with rich text explanations and visualisations, makes them particularly valuable for AI/ML workflows where iterative exploration and clear documentation of model development are essential.

Widespread adoption across both industry and academia, plus an extensive plugin ecosystem and integration with popular AI frameworks, demonstrates their maturity as a method of interacting with code. We especially value how notebooks facilitate collaboration between technical and non-technical team members, as they can serve as living documents that combine business requirements, technical implementation, and results in a single, shareable format.

Jupyter notebooks are the most widely used, supporting multiple languages including Python, R and Julia. The cloud platforms provide their own implementations: Google Colab, AWS Sagemaker Notebooks, Azure Notebooks, Databricks Notebooks. And there are language specific notebooks, such as Pluto.jl for Julia, Clerk for Clojure, Polynote for Scala.

Trial

These tools show promising potential with growing adoption and active development. While they may not yet have the same maturity as Adopt tools, they offer innovative approaches and capabilities that make them worth exploring for forward-thinking teams.

MLflow

We have placed MLFlow in the Trial ring due to its potential as a lightweight and modular option for teams seeking to manage the machine learning lifecycle. Its open-source nature makes it an attractive alternative to the more monolithic cloud-based MLOps platforms provided by vendors like AWS, Microsoft and Google. A key advantage of MLFlow is its ability to avoid vendor lock-in, offering teams the flexibility to maintain control of their infrastructure and adapt workflows as their needs evolve.

That said, realising the benefits of MLFlow requires teams to have a certain level of technical expertise to configure and integrate it into their existing systems effectively. Unlike cloud-native behemoths such as SageMaker or Vertex AI, MLFlow does not provide an all-in-one, plug-and-play experience. Instead, it offers modular components that must be tailored to specific use cases. We recommend assessing MLFlow if your organisation values flexibility, has the technical proficiency to manage integrations, and prefers avoiding dependency on proprietary platforms early in your MLOps journey.

Vector databases

Vector databases have emerged as specialised tools for managing the high-dimensional data representations (embeddings) required by AI models. They enable efficient similarity search across text, images, and other content types. Prominent solutions include Pinecone, Qdrant, Milvus and Weaviate.

We've generally placed vector databases in the Trial ring, as they have proven valuable for specific use cases such as semantic search and recommendation systems. However, their adoption should be carefully evaluated based on individual requirements. Traditional databases may be sufficient for simpler operations and avoid the data consistency challenges of keeping embeddings synchronized with underlying content changes across databases. Alternative approaches, such as Timescale's PGAI vectorizer, bring vector embedding search directly into the Postgres database, ensuring embeddings remain synchronised with underlying content changes.

If a vector database is required for your use case, the choice of provider often depends on factors such as scale requirements, the need for real-time updates, and whether a managed or self-hosted solution is preferred. Pinecone leads in production readiness but comes with the costs of a managed service, while open-source alternatives like Qdrant and Milvus offer greater control but demand more operational expertise.

Local model execution environments

Tools like Ollama, LM Studio, and AnythingLLM provide accessible ways to run open weight models on local hardware. These environments enable rapid experimentation with open weight models from providers including Meta (Llama), Mistral, DeepSeek, Alibaba (Qwen), and OpenAI (gpt-oss) without API costs or sending data to external services. Many now support advanced capabilities including web search, tool calling via Model Context Protocol (MCP), and connections to commercial APIs for hybrid workflows.

These tools serve various evaluation needs: developers testing AI features during development, teams comparing model responses for specific use cases, and organisations exploring AI capabilities with sensitive data that cannot leave their infrastructure. The range spans from command-line interfaces like Ollama to graphical applications like LM Studio, accommodating different technical backgrounds and preferences.

We've placed these in Trial as they offer a valuable alternative approach to model evaluation alongside cloud-based testing. They're particularly useful for privacy-sensitive prototyping, offline development, and scenarios where extensive experimentation would be cost-prohibitive via APIs. Teams should consider these tools as one option among many for model evaluation, weighing their benefits against the overhead of local setup and maintenance.

Assess

These tools represent emerging or specialized technologies that may be worth considering for specific use cases. While they offer interesting capabilities, they require careful evaluation due to limited adoption, specialized requirements, or uncertain long-term viability.

AI application bootstrappers

We have placed AI Application Bootstrappers like V0, Bolt.new and Replit Agent in the Assess ring of our Tools quadrant. These tools represent an intriguing new approach to rapidly generating complete applications from prompts or designs. While they can dramatically accelerate the creation of demos and proofs of concept, their current limitations lead us to recommend careful assessment before adoption.

The primary value proposition is clear: the ability to go from concept to working prototype in hours instead of days or weeks. However, our experience shows that success with these tools correlates strongly with existing software engineering expertise. Senior developers can effectively use them as accelerators, understanding how to refactor the generated code, identify potential issues, and establish proper architectural boundaries. In contrast, junior developers or non-technical users often struggle with maintaining and evolving the generated codebase, finding themselves unable to effectively debug issues or make substantial modifications without creating cascading problems.

While these tools excel at creating initial implementations, the significant effort required to make applications production-ready still requires substantial engineering knowledge. We're particularly concerned about teams using bootstrapped code as a foundation for production systems without the expertise to properly evaluate and refactor the generated codebase. The tools are promising but should be approached with clear understanding of their current limitations and best used by teams with strong software engineering fundamentals.

Looking ahead, we expect these tools to mature and potentially move into the Trial ring as they develop better guardrails and more maintainable output. For now, we recommend assessing them primarily for simple prototyping and proof-of-concept work, while maintaining careful separation between bootstrapped demos and production codebases.

Agentic computer use

AI agents that directly interact with computer interfaces represent an intriguing development in AI tooling. OpenAI's Operator, integrated into ChatGPT as "agent mode," and Claude Computer Use can control web browsers and desktop applications through visual understanding and automated screen interactions. Development-focused agents like Devin take a different approach, working within integrated development environments and specialising in code repositories through programmatic tool interactions.

These systems process screen content through visual analysis, reasoning about current context and task requirements, then execute mouse clicks, keyboard inputs, and application navigation. While organisations express significant interest in deploying AI agents, early adopters are encountering reliability challenges, with success rates declining markedly as task complexity increases and agent workflows become more extended.

We've placed Agentic Computer Use in the Assess ring because whilst the technology demonstrates clear potential for specific use cases, practical implementation remains challenging. Early implementations show promise in constrained environments with well-defined boundaries, but teams report inconsistent results when scaling to more complex workflows or longer chains of automated activity.

For teams evaluating these tools, we recommend focusing on simple, isolated tasks with clear success criteria rather than complex multi-step workflows. Maintain human oversight for all critical operations and establish robust audit trails. The technology merits careful assessment, but organisations should approach deployment conservatively until reliability and control mechanisms mature further.

Lakera

Lakera is an AI safety and robustness platform designed to detect and mitigate risks in machine learning systems. It provides mechanisms for testing, analysis, and quality assurance to help developers identify weaknesses or vulnerabilities in AI/ML models prior to deployment. This makes it particularly appealing in contexts where reliability and safety are paramount, such as finance, healthcare, or any domain subject to compliance constraints.

We have placed Lakera in the Assess ring because while it addresses an important need for AI safety, the platform has several practical limitations that require careful evaluation. Currently, Lakera supports only text-based scanning, teams using multimodal AI systems with images, audio, or video will find gaps in coverage. Custom scanning capabilities for business-specific terms or PII detection rely on regex patterns rather than context-aware analysis, which can quickly hit limitations in complex scenarios.

Performance considerations vary significantly between deployment options. The SaaS offering may provide adequate performance for many use cases, but has text size limitations that require applications to handle chunking. Self-hosted deployments offer more control but require substantial GPU resources for acceptable performance. Additionally, Lakera's scanning is non-stateful, each prompt and response is scanned in isolation without awareness of the broader conversation context, and only 'user' and 'assistant' message types are recognised.

Given these constraints, Lakera may provide valuable safety assurance for straightforward text-based AI applications, but organisations should carefully assess whether its current capabilities align with their specific AI architectures and safety requirements. We recommend conducting thorough proof-of-concept testing that includes your specific modalities, custom requirements, and performance expectations before determining if Lakera fits your use case.

Hold

These tools are not recommended for new projects due to declining relevance, better alternatives, or limited long-term viability. While some may still have niche applications, they generally represent technologies that have been superseded by more effective solutions.

Conversational data analysis

Tools such as pandas-ai, tablegpt, promptql, and Julius enable natural language querying of databases and datasets, offering significant productivity benefits for knowledgeable data analysts. Modern database-specific Model Context Protocol (MCP) servers can provide substantial context to models, including schema understanding and data contents. Our experience with JUXT's own XTDB database revealed remarkable moments where models navigated complex table structures with apparent ease, demonstrating genuine potential for accelerating data analysis workflows.

For experienced analysts, these tools represent a meaningful productivity boost, rapidly converting natural language requests into draft queries that can be refined and optimised. However, our experience also reveals challenges: generated queries can be inefficient or occasionally incorrect despite appearing plausible. The technology sometimes struggles with nuanced requirements and may produce suboptimal approaches that experienced analysts would avoid. Uber's experience with their internal QueryGPT tool demonstrates both the potential and the complexity, highlighting the significant number of example queries and guardrails required to achieve reliable results.

We've placed conversational data analysis in the Hold ring not because the technology lacks value, but because successful deployment requires users capable of understanding and validating generated queries. These tools offer substantial benefits for data teams with appropriate expertise, but should be approached cautiously by those unable to review and debug AI-generated database queries.

For teams with strong analytical capabilities, these tools can meaningfully accelerate exploratory data analysis and routine query generation, treating AI output as sophisticated first drafts requiring expert review.

Platforms

Infrastructure and platform services that support AI applications, from model hosting to experiment tracking. These platforms provide the foundation for building, deploying, and managing AI systems at scale.

ADOPT

45. Weights & Biases

46. Foundation models

47. Data pipeline orchestration tools

48. Cloud model hosting platforms

TRIAL

49. Production AI monitoring platforms

50. Open weight LLMs

51. AI-powered workflow automation platforms

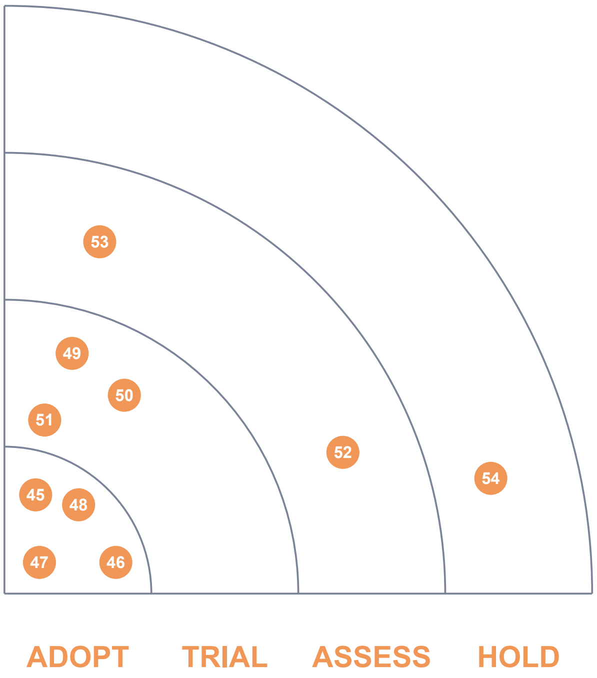
ASSESS

52. Galileo

53. Kubeflow

HOLD

54. Building agains tvendor-specific APIs



Platforms

Adopt

These languages and frameworks represent mature, well-supported technologies that are ready for production use. They offer excellent performance, extensive ecosystems, and proven track records in real-world applications.

Weights & Biases

Weights & Biases is a platform designed for tracking and visualising machine learning experiments. In recent projects, we've observed that it provides a robust solution for managing machine learning workflows, particularly when dealing with complex models and large datasets. Its user-friendly interface and integration capabilities with popular machine learning libraries make it accessible for teams looking to improve their model development processes.

We've seen how systems such as Weights & Biases can catalyse positive cultural changes in ML teams. By making experiment tracking very light touch, requiring just a few lines of code, they remove the friction that sometimes prevents teams from maintaining good measurement practices. When tracking experiments becomes a natural part of the workflow rather than an extra burden, teams tend to measure more, compare results more frequently, and generally make more data-driven decisions.

Collaboration features such as shared dashboards and reports amplify these benefits by making results and insights visible to the whole team. Rather than knowledge being siloed in individual notebooks or spreadsheets, experiments become shared assets that everyone can learn from. This visibility often leads to more discussion about results, faster knowledge sharing, and ultimately quicker iteration cycles as teams build upon each other's work rather than inadvertently duplicating efforts. However, it's important to note that tool adoption alone isn't enough, teams need to actively foster a culture that values measurement and experimentation for these benefits to fully materialise.

Foundation models

Foundation model providers continue to evolve at a rapid pace. Major players like OpenAI, Anthropic, Google, and Meta compete alongside emerging organisations such as DeepSeek, Alibaba, IBM and others. While industry benchmarks help compare these models, they tell only part of the story: different models excel in different areas, and benchmark results should be viewed as indicative rather than definitive.

A clear trend has emerged in how providers differentiate their offerings across three distinct tiers: smaller, faster models (e.g., Claude Haiku, DeepSeek Coder, Qwen Turbo) optimised for speed and cost; larger, more capable models (e.g., Claude Sonnet, DeepSeek V3, Qwen Max) balancing capabilities with reasonable response times; and specialised reasoning models (e.g., Claude Sonnet Extended, OpenAI o1, DeepSeek R1) designed for complex problem-solving. These reasoning models consume significantly more tokens and command higher per-token costs, but demonstrate remarkable capabilities in solving challenging logical puzzles, mathematics problems, and coding tasks.

We believe foundation models have evolved sufficiently to warrant adoption for many business applications. When paired with appropriate infrastructure (few-shot prompting, guardrails, retrieval-augmented generation, and evaluation frameworks), they offer compelling solutions to a wide range of problems. Our experience suggests there's no universal "best model". We recommend implementing your own benchmarking process focused on your specific use cases. When selecting a model, consider factors beyond raw performance, such as pricing, reliability, data privacy requirements, and whether on-premise deployment is needed. The recent emergence of high-quality open-source models with permissive licensing (such as DeepSeek's offerings) provides additional options for organisations with specific security or deployment requirements.

Platforms

Key considerations:

- Performance & capabilities (accuracy, speed, and domain-specific strengths)
- Total cost of ownership (API costs, compute resources, and integration)
- Deployment options & technical requirements (cloud, self-hosted, edge)
- Data privacy & compliance (regulatory, legal, and security implications)
- Integration & lifecycle management (context limitations, version control, updates)
- Vendor stability & support (roadmap alignment, documentation, community)

Foundation model providers feature comparison (September 2025)

Provider	Open Weights	Enterprise Focus	Reasoning Models	Edge Deployment	Long Context	Embedding API	Agentic Workflows
Alibaba	✓			✓	✓	✓	
Anthropic		✓	✓		✓		✓
AWS		✓	✓		✓		
Cohere	✓	✓	✓		✓	✓	
DeepSeek	✓		✓	✓			
Google		✓	✓		✓	✓	✓
IBM	✓	✓	✓	✓	✓		
Meta	✓			✓			
Mistral AI	✓	✓	✓	✓		✓	
OpenAI	✓	✓	✓	✓	✓	✓	✓
Stability AI	✓		✓	✓			
X	✓		✓		✓		✓

Platforms

Feature definitions

- Open Weights: Models whose weights are publicly available for download and customisation
- Enterprise Focus: Strong emphasis on governance, security, and enterprise integration
- Reasoning Models: Specialised models for complex reasoning tasks like mathematics or step-by-step problem solving
- Edge Deployment: Optimised for deployment on edge devices or resource-constrained environments
- Long Context: Support for context windows of 250K tokens or more
- Embedding API: Dedicated text embedding models and APIs for generating vector representations of text for semantic search and similarity tasks
- Agentic Workflows: Ability to autonomously plan, execute, and adapt multi-step tasks using tools and external services. Goes beyond basic function calling to include complex workflow orchestration, error handling, dynamic planning based on intermediate results, and completing entire business processes without human intervention at each step

Data pipeline orchestration tools

Data pipeline orchestration has become essential infrastructure for organisations managing complex data workflows, particularly those supporting AI and machine learning initiatives. Whilst transformation tools like dbt handle the “what” of data processing, orchestration platforms manage the “when,” “how,” and “monitoring” of entire pipelines. We’ve placed these tools in the Adopt ring because established organisations require systematic approaches to pipeline scheduling, dependency management, and failure recovery.

Apache Airflow represents the established approach, focusing on task-based workflows with broad integration support across cloud platforms. Its maturity and established ecosystem make it the de facto standard in many enterprises, though teams often find the learning curve steep. Prefect emphasises developer experience and dynamic workflow adaptation, allowing workflows to adapt to changing conditions with minimal code modification. Teams report faster development cycles, though fewer third-party integrations reflect the platform’s relative youth.

Dagster takes an asset-centric approach where data assets become first-class citizens, providing built-in lineage tracking, data quality monitoring, and metadata management. This modern architecture includes comprehensive developer tooling and observability, though the conceptual shift from task-based thinking requires adjustment.

The choice between platforms typically depends on organisational context rather than technical superiority. Established enterprises with diverse toolchains often gravitate towards Airflow’s ecosystem breadth, whilst teams prioritising developer velocity may prefer Prefect’s flexibility. Organisations with complex data lineage requirements increasingly consider Dagster’s asset-aware approach. We recommend evaluating these tools against your specific integration complexity, team expertise, and governance needs.

Platforms

Cloud model hosting platforms

The model hosting landscape has evolved far beyond simple API access, with distinct platforms serving different organisational needs from rapid prototyping to enterprise production deployments. Each platform's approach to custom model deployment varies significantly, as organisations increasingly require hosting for their own fine-tuned models alongside foundation model access. We've placed these platforms in the Adopt ring because cloud-based model hosting has become the de facto approach for most AI deployments, reducing operational overhead.

Enterprise production environments often gravitate towards established cloud providers such as AWS Bedrock, Google Vertex AI, and Azure OpenAI Service. These platforms provide fine-tuning capabilities with enterprise security features and integration with existing cloud infrastructure. Azure's hub-and-spoke architecture (separating model training from deployment environments) and Google's "Import Custom Model Weights" feature automate parts of custom model deployment, though the processes often require cloud platform expertise and lengthy setup procedures.

Performance-critical applications are increasingly considering specialised providers such as Fireworks AI and Together AI, which focus specifically on inference optimisation and support deployment of custom fine-tuned models. These platforms offer API-based deployment workflows, with Together AI supporting trillion-parameter model training and Fireworks providing fine-tuning services. However, teams must evaluate whether simplified deployment compensates for reduced ecosystem integration compared to major cloud providers.

Development teams and startups often favour platforms such as Replicate, Modal, and Hugging Face Inference Endpoints, which emphasise deployment ease alongside flexible pricing. Hugging Face supports deployment of 60,000+ models with minimal configuration, whilst Replicate's Cog packaging system and Modal's Python-decorator approach reduce deployment steps. These platforms offer direct paths from trained model to production API, though enterprise governance features remain limited.

The choice between platforms reflects both organisational priorities and deployment complexity tolerance. Teams requiring sophisticated fine-tuning workflows with enterprise compliance often find major cloud providers necessary despite steeper learning curves. Performance-focused organisations benefit from specialised platforms that balance custom model support with optimisation capabilities. Development teams prioritising rapid iteration prefer platforms with simplified deployment processes, accepting more limited enterprise tooling.

Trial

These platforms show promising potential with growing adoption and active development. While they may not yet have the same maturity as Adopt platforms, they offer innovative approaches and capabilities that make them worth exploring for forward-thinking teams.

Production AI monitoring platforms

Whilst experiment tracking tools like Weights & Biases and MLflow excel at managing the development lifecycle, a distinct category of platforms has emerged to monitor AI systems in production. These tools detect drift, performance degradation, and unexpected behaviour in deployed models, issues that only surface when models encounter real-world data at scale. We've placed these platforms in the Trial ring as organisations continue establishing best practices for production AI monitoring.

Arize AI provides unified observability across traditional ML models and LLM applications, continuously tracking feature and embedding drift from training through to production. The platform helps catch production issues before customer impact, though careful configuration is needed to avoid alert fatigue. Evidently AI offers both an open-source library and cloud platform, with over 100 metrics covering data quality, drift, and bias monitoring. Its flexibility appeals to technical teams, though setup requires more effort than managed alternatives.

WhyLabs takes a privacy-preserving approach, monitoring through statistical profiles rather than raw data access. This enables massive scale monitoring whilst maintaining data security, particularly valuable for regulated industries. The platform claims superior drift detection accuracy, though teams must weigh privacy benefits against reduced debugging visibility.

Whilst there are many approaches to production AI monitoring, from custom metrics to manual spot checks, these platforms deserve consideration from teams hosting models in production. They integrate with existing SRE workflows through standard alerting channels (PagerDuty, Slack, email) and provide dashboards that fit alongside traditional application monitoring. The key benefit is proactive detection: organisations learn about performance degradation or prediction errors before customer impact, rather than discovering issues through support tickets. For teams already practising observability for their applications, adding AI-specific monitoring represents a natural extension of existing operational practices.

Open weight LLMs

2024 was the year when open weight LLMs (which are sometimes incorrectly referred to as ‘open source’) from companies such as Meta and Deepseek reached maturity, with some even surpassing flagship frontier models on certain tasks. We’ve placed open weight LLMs in the Trial ring because they allow organisations to benefit from AI capabilities while maintaining control over their data and deployment. These models have demonstrated impressive performance, particularly in specialised domains when fine-tuned on specific tasks.

The key benefits include reduced operational costs compared to API-based services, full control over model deployment and customisation, and the ability to run models in air-gapped environments where data privacy is paramount. However, we’ve kept them in Trial because organisations need considerable ML engineering expertise to deploy and maintain these models effectively, and the total cost of ownership isn’t always lower than API-based alternatives when accounting for computational resources and engineering time.

For certain use cases, the simplicity of a pay-per-use API integration outweighs the benefits and greater control of hosting an open source LLM. Additionally, implementing appropriate security controls, prompt injection protection, and data governance poses significant challenges.

AI-powered workflow automation platforms

Visual workflow automation platforms have become increasingly capable orchestrators of AI-powered business processes, allowing teams to build automated workflows through drag-and-drop interfaces rather than traditional coding. We’ve placed these platforms in the Trial ring because whilst they represent a maturing approach to democratising AI automation across organisations, the choice of platform depends heavily on specific technical and organisational requirements.

Prominent platforms in this space include Zapier, n8n, Microsoft Power Automate, and Make.com. Each serves different organisational needs and technical constraints. Zapier focuses on connecting thousands of SaaS applications with AI capabilities, positioning itself towards business users seeking rapid automation deployment. n8n distinguishes itself through flexibility for technical teams, offering self-hosting options, open-source licensing, and extensive customisation through HTTP nodes and JavaScript code injection. Microsoft Power Automate leverages native Office 365 integration and enterprise-grade governance features, whilst Make.com emphasises sophisticated visual workflow design with AI agent functionality.

These platforms attempt to bridge the gap between technical and business teams around AI automation. They allow organisations to prototype AI-enhanced workflows, connect disparate systems, and scale automation efforts without building custom integration layers. We've observed common use cases including lead qualification using LLM analysis, automated content generation and distribution, customer support ticket routing and responses, and data processing pipelines that incorporate AI models for classification or enrichment tasks.

When evaluating these platforms, teams should consider their organisation's technical capability, data sovereignty requirements, integration ecosystem needs, and long-term scalability plans. Self-hosted solutions like n8n offer maximum control and customisation but require technical expertise, whilst SaaS offerings like Zapier reduce operational overhead but may have cost implications at scale. Teams should also assess the platforms' capability for error recovery, monitoring, and debugging of AI-enhanced workflows, as AI components can fail in less predictable ways than traditional integrations.

Assess

These platforms represent emerging or specialized services that may be worth considering for specific use cases. While they offer interesting capabilities, they require careful evaluation due to limited adoption, specialized requirements, or uncertain long-term viability.

Galileo

We've placed Galileo in the Assess ring of the Platforms radiant because it represents an interesting approach to evaluating and improving AI model performance. It deserves attention but requires careful consideration before being adopted more broadly.

Galileo offers a comprehensive platform spanning both development evaluation and production monitoring of AI systems. During development, it provides tools for measuring and refining model performance, with specialised capabilities for AI agent evaluation and comprehensive testing frameworks. In production, the platform offers real-time monitoring with low-latency guardrails and hallucination detection. Our committee has noted that teams using the platform report better insights into how their AI systems perform across different scenarios and edge cases, from initial development through to production deployment.

We recommend assessing this platform, particularly if your organisation is developing custom models or fine-tuning existing ones, as the insights it provides could significantly improve model quality. However, we've stopped short of recommending it for trial by all teams, as its value varies depending on your level of AI maturity and your specific use cases. Organisations with simpler AI implementations, or those primarily using out-of-the-box models, may find less immediate benefit. The platform is likely to offer the most value to organisations that are actively developing or fine-tuning models, or deploying AI in high-stakes environments where consistent performance is critical. Teams should also consider whether they have the technical resources required to act effectively on the insights the platform provides.

Kubeflow

We've placed Kubeflow in the Assess ring of our Platforms quadrant. This open-source machine learning platform, built on Kubernetes, offers a comprehensive solution for managing ML workflows, but it requires careful evaluation before widespread adoption.

Kubeflow is gaining traction among data science and MLOps teams looking to standardise their machine learning workflows. Its strength lies in combining Kubernetes' orchestration capabilities with ML-specific tools: Pipelines for workflow automation, Katib for hyperparameter tuning, and KFServing for model deployment. This integrated approach helps bridge the gap between data scientists and operations teams, addressing one of the core challenges in operationalising ML models.

However, several factors keep Kubeflow in our Assess ring. First, implementing Kubeflow demands significant expertise in both Kubernetes and ML engineering, a specialised skill set that remains relatively uncommon. Second, while the platform is maturing, we've observed that many organisations struggle with its complexity during initial setup and ongoing maintenance. Teams often report a steep learning curve before realising tangible benefits.

Organisations with established ML practices and existing Kubernetes expertise should consider assessing Kubeflow, particularly if they're facing challenges with ML model deployment, experiment reproducibility or resource utilisation. The platform is especially suited to enterprises managing multiple ML models in production that require systematic oversight across their lifecycle. Smaller teams, or those earlier in their ML journey, may want to explore simpler alternatives first or consider managed options like Vertex AI Pipelines, which abstract away some of the infrastructure complexity.

Hold

These platforms are not recommended for new projects due to declining relevance, better alternatives, or limited long-term viability. While some may still have niche applications, they generally represent approaches that have been superseded by more effective solutions.

Building against vendor-specific APIs

We've placed "Building against vendor-specific APIs" in the Hold ring of the Platforms quadrant because tightly coupling your applications to vendor-specific LLM APIs poses significant business risks in this rapidly evolving landscape.

The foundation model ecosystem is changing at breakneck speed, with model capabilities, pricing and even entire companies shifting dramatically from month to month. Organisations that build directly against OpenAI, Anthropic or other proprietary APIs often find themselves locked in, facing painful migrations when a better or more cost-effective model emerges. We've seen teams invest substantial engineering effort into rewriting API integrations after discovering their chosen vendor has been outperformed or has significantly increased its pricing.

Instead, we recommend using abstraction libraries that provide a common interface to multiple LLM providers. Libraries such as AISuite or Simon Willison's LLM CLI let you switch between different models with minimal code changes, sometimes just a configuration update. These libraries handle the nuances of different vendor APIs, managing context windows, token limitations and provider-specific parameters behind a consistent interface. This approach preserves your flexibility to take advantage of new capabilities or improved pricing as the market evolves, while significantly reducing the engineering effort required to switch between models.

These abstractions do add some complexity and may occasionally limit access to vendor-specific features, but in our view, the protection against vendor lock-in far outweighs these drawbacks in most cases. As the foundation model market continues to consolidate, maintaining the flexibility to adapt quickly will be crucial for both cost management and staying competitive.

Thank you