AWS Whitepaper

Responsible AI Lens

Responsible AI Lens: AWS Whitepaper

Table of Contents

Responsible AI Lens - AWS Well-Architected Framework

Publication date: **November 19, 2025** (*Document revisions*)

The AWS Well-Architected Framework Responsible AI (RAI) Lens assists builder teams succeed at responsibly building and operating AI solutions that solve specific AI use cases. It complements existing Well-Architected publications.

Challenges: AI technology is disrupting existing business processes and powering the next wave of transformational capabilities in business, consumer, and public sector systems and applications. However, AI technology that uses machine-learning differs from rule-based software technology, and moving an AI use case from idea to trusted production workload requires builder teams to balance benefits and risks using best practices that are specific to AI, as well as best practices for rule-based software technology. An increasingly complex AI application stack, consisting of customer and third-party data, models, agents, MCP tools, open and closed-source libraries, retrieval augmented generation libraries, guardrails, and other specialized components, increases the complexity of making design decisions.

Specifically, teams building or deploying AI technology (models or applications) confront a range of risks that are not typically covered in academic or on-the-job training in AI/ML. These risks, if not blocked or mitigated, can undermine the reliability and utility of an AI system. As an example, consider a generative AI application that creates descriptions of condominiums for sale from retrieved private and public data sources. Is the generated text equally welcoming to each buyer demographic group? Are the property features accurately captured from the inputs and not hallucinated? Have private details about the owners or past occupants leaked into the descriptions? Are uploaded condo images free of unsafe content? Are generated images both accurate and protected by watermarking?

Responsible AI is the discipline of designing, developing, and using AI technology with the goal of maximizing benefits and minimizing risks. At AWS, we define Responsible AI using a core set of dimensions that we update over time as AI evolves. These dimensions are specific to AI and complementary to engineering considerations such as cloud security, cloud privacy, operational excellence and other factors covered already in the AWS Well-Architected Framework. We use these responsible AI dimensions to assist builders identify benefits and risks inherent in an AI use case, and to organize their decision-making around responsibly designing and evaluating their AI system. The dimensions are:

- **Controllability:** Having mechanisms to monitor and steer AI system behavior.

- **Privacy:** Appropriately obtaining, using, and managing data.
- **Security:** Protecting data and models from exfiltration and adversarial inputs.
- **Safety:** Blocking harmful system output and misuse.
- **Veracity:** Achieving factually correct system outputs.
- **Robustness:** Achieving correct system outputs for both expected and unexpected inputs.
- **Fairness:** Considering impacts on different groups of stakeholders.
- **Explainability:** Having mechanisms to understand system behavior.
- **Transparency:** Enabling stakeholders to make informed choices about their engagement with an AI system.
- **Governance:** Incorporating best practices into the AI supply chain, including providers and deployers.

The Responsible AI Lens has three audiences:

- **AI builders:** Engineers, product managers, and scientists who develop and deploy AI systems to solve AI use cases. Builders get guidance on how to structure their work to identify and optimize benefit and risk tradeoffs specific to AI applications.
- **AI technical leaders:** Oversee teams building AI systems and implement enterprise-wide responsible AI practices. Leaders get a framework they can use to standardize their approaches to balancing portfolio risk and earning their own customers' trust.
- **Responsible AI specialists:** Establish the specific policies needed by their organizations to assist you to comply with applicable regulations and industry standards, and work with builder teams to meet the policies. Specialists benefit from having a science-based best practice framework to assist them set and implement their own organization's AI-related policies.

How to use this guidance

The Responsible AI Lens is structured as a set of eight focus areas, each of which aligns with a phase in the machine learning lifecycle. Each focus area contains a set of key questions for builders to consider, and each question is answered with a set of best practices. Builders determine for themselves if a given question or best practice is relevant to their situation. Since AI development can be iterative and nonlinear, we do not expect that builders will sequentially work through the focus areas, the questions within the focus areas, and the best practices for each question. However, we do recommend that builders familiarize themselves with the guidance by sequentially reading through the focus areas and questions and then working through the focus areas in an order appropriate to their situation. Builders should also consider the following factors in deciding how and when to use the guidance:

1. This guidance is aimed at assisting teams design an AI solution to solve a specific use case. It is not aimed at assisting teams to design general-purpose AI systems (for example, foundation models). The guidance is also not appropriate for AI use cases that can be solved without machine learning, for example, by using expert systems.
2. This guidance does not cover best practices for cloud security, cloud privacy, project management, software engineering, data management, or other areas for which builders already have many resources (like many other lenses and guidance in the AWS Well-Architected Framework).
3. Treat each question, best practice, and implementation guidance as considerations, not recommendations or requirements. Not every consideration will apply to each use cases or type of AI solution.
4. This guidance does not provide specific solutions for each of many possible AI use cases. Instead, the guidance articulates questions that are generally applicable across use cases. Therefore, expect that solving your specific use case will require additional digging, for example to identify the best metrics for your release criteria.
5. Many aspects of responsible AI, like security and post-release monitoring, are evolving quickly. We include best practices only when we feel they are sufficiently mature.
6. Do not use this guidance as a compliance or assurance checklist. There are an increasing number of AI-related regulations and standards (for example, the EU AI Act, NIST AI 600, and ISO 42001). Every organization must decide for itself how to interpret and implement the regulations and standards to which it is subject. You should consult with your legal counsel to understand legal or compliance obligations that may apply.
7. The term *risk* appears in the Responsible AI Lens in two contexts. First, some best practices use the term *risk* to refer to a potential responsible AI risk. Second, each best practice within the AWS Well-Architected Framework is labeled as high risk or medium risk depending on the potential risk to the reader's project or business of not considering the best practice (see Identify and understand risks for more information).

   In neither case does the term *risk* refer to a potential or actual legal or compliance risk. The information and recommendations provided in the Responsible AI Lens are advisory in nature and should be modified to fit the specifics of your situation. You should consult with your legal counsel to understand legal or compliance obligations that may apply.

Lens availability

Custom lenses extend the best practice guidance provided by AWS Well-Architected Tool. AWS WA Tool allows you to create your own custom lenses, or to use lenses created by others that have been shared with you.

To begin reviewing your AI workload, download and import the Responsible AI Lens into AWS Well-Architected Tool from the public AWS Well-Architected custom lens GitHub repository.

Design principles

Through research, experimentation, and experience, AWS has identified a set of best practices (a Responsible AI best practice framework) for building and operating AI applications that are intended to solve specific use cases (for example, traditional ML applications, generative AI applications (including those requiring builders to customize foundation models), and agentic applications). The framework is published as this Well-Architected guidance. We have selected best practices according to the following design principles:

- **Support narrowly defined use cases**: The specifications for an AI system should be developed by working backwards from the AI use case (in other words, the problem to be solved). The use case directly determines potential risks and release criteria. Narrowly defined use cases limit the extent of potential AI risks and focus builder teams on applicable risks. This guidance is not appropriate for building general purpose AI systems.
- **Responsible by design**: Our framework embeds responsible AI best practices throughout the AI lifecycle from design through operations. However, we emphasize identifying and resolving potential issues in design. This does not preclude the use of rapid prototype-and-release development processes, but it does encourage open, transparent iteration.
- **Follow the science**: We choose best practices that are based on science, and we express the practices in language that does not require deep expertise in machine learning to understand.

Use case

The AI use case is the problem you are solving with an AI system. This focus area assists you to produce a high-level use case description and a minimal description of the AI solution, including system inputs, system outputs, and the type of AI solution (traditional, generative, or agentic AI). Deeper dives into benefits and risks come in the Benefits and Risk focus area. If you want to design a system to handle multiple related use cases (for example, to recognize faces of missing children or locate frames in a video containing a specific actor), then you should consider applying this framework to each use case independently. If you want to design a general-purpose system (for example, to chat with anyone about anything), then this framework is not appropriate.

Focus areas

- Define your specific problem
- Identify use case stakeholders
- Refine your use case
- AI workflow impact
- Identify requirements and approvals

Define your specific problem

RAIUC01: How do you define the specific problem you are trying to solve?

The narrower the use case, the more precisely you can assess and mitigate risks and measure performance.

Best practices

- RAIUC01-BP01 Clarify the business problem
- RAIUC01-BP02 Verify that AI is required to solve the problem

RAIUC01-BP01 Clarify the business problem

Describe the specific problem or business challenge. Assess how frequently the challenge occurs, where it occurs, and its concrete impacts. Describe the specific benefit of solving the challenge for the primary user for your use case.

Level of risk exposed if this best practice is not established: High

Implementation considerations

1. Collect quantitative and qualitative data on the problem's frequency, impact, and specific scenarios from primary users and other sources.
2. Define clear boundaries for the problem scope, including domain, geographic, and user segment limitations.
3. Evaluate the urgency by quantifying inaction costs and quantifying potential benefits of solving the problem.
4. Draft a structured problem statement using a format such as "Enable <user> to <infer, generate, do <y> given input <x> when <context> every <frequency> with <benefit>".
5. Validate the problem statement with key stakeholders, confirm its current relevance, and refine based on feedback.

Resources

Related documents:

- ISO/IEC 42001:2023 A.9.3 Objectives for responsible use of AI system
- ISO/IEC 42001:2023 A.6.2.2 AI system requirements and specification
- NIST Artificial Intelligence Risk Management Framework (NIST AI 100-1): MAP1.1, MAP1.3, MAP1.4

RAIUC01-BP02 Verify that AI is required to solve the problem

Before committing to using AI, evaluate whether traditional software approaches or even manual processes could meet your requirements. Choose AI if it provides clear, substantial benefits over competing solutions, and not simply because it is technically possible to apply AI to the problem.

Level of risk exposed if this best practice is not established: High

Implementation considerations

1. Consider whether you can solve the problem by writing down a clear and compact set of rules. It is generally infeasible, for example, to write down a compact set of pixel-comparison rules to decide if two arbitrary face images represent the same person. However, it is feasible to write down a brief set of rules to decide if two images are identical. If yes, you may be able to use a solution other than machine learning, and you may not need this best practice guidance.
2. Consider whether you can access reliable sources of training, fine-tuning, and test examples. If it is difficult to access such examples, you may not have the data necessary to develop or evaluate an AI, and may need to consider either reframing the use case or alternate solutions.
3. Consider alternative reformulations of the use case that might be solved by rule-based systems, traditional information retrieval systems, or other software solutions.

Resources

Related documents:

- Responsible AI Practices
- ISO/IEC 42001:2023 A.9.2 Process for responsible use of AI systems
- NIST Artificial Intelligence Risk Management Framework (NIST AI 100-1): MAP1.1, MAP1.6, MAP2.1

Identify use case stakeholders

RAIUC02: How will you identify the stakeholders of your use case?

A stakeholder is a person, group or entity involved in, or affected by, the development or operation of the AI system. Identify the specific stakeholders so that you can assess actual benefits and risks for your use case.

Best practices

- RAIUC02-BP01 Identify downstream stakeholders
- RAIUC02-BP02 Identify contributing and other upstream stakeholders

RAIUC02-BP01 Identify downstream stakeholders

Identify a person, group, or entity involved in or affected by the operation of the proposed AI system. Consider different stakeholder categories, like primary users, secondary users, and indirect stakeholders. Consider whether vulnerable populations could be stakeholders. Seek out individuals with different perspectives from those of the builder team, including potential stakeholder groups and different organizational functions, to identify possible stakeholders.

Level of risk exposed if this best practice is not established: High

Implementation considerations

1. Organize workshops with potential end users, buyers, builders, and operators to brainstorm potential stakeholders. Include people with different backgrounds and expertise.
2. Categorize identified stakeholders into primary users (for example, those providing inputs and receiving outputs), secondary users (for example, those whose data may used as input), and indirect stakeholders (for example, those affected by system operations). Include vulnerable populations across categories.
3. Analyze each stakeholder group's relationship to the AI system by documenting their expected interactions, potential impacts, and specific needs or concerns. Break down larger stakeholder groups into relevant subgroups for detailed analysis.
4. Review and validate the stakeholder list periodically throughout system development to capture emerging stakeholder groups and changing relationships. Consider how system modifications might affect different stakeholder groups.

Resources

Related documents:

- NIST Artificial Intelligence Risk Management Framework (NIST AI 100-1): MAP1.1, MAP5.1, GOVERN5.1

RAIUC02-BP02 Identify contributing and other upstream stakeholders

Identify the full set of people involved in designing, developing, deploying, operating, funding, supplying, and approving an AI system built by your team. The set may include product managers, engineers, data scientists, AI oversight functions (compliance, assurance, risk), domain experts on topics such as security, privacy, existing AI systems, or the use case itself, as well as other contributors or users.

Level of risk exposed if this best practice is not established: High

Implementation considerations

1. Map out the roles involved in your AI system's lifecycle from initial concept to ongoing operations. Consider product, engineering, data science, legal, security, infrastructure, and other company functions that provide input on requirements or constraints.
2. Identify external stakeholders who contribute to or influence your system even though they are not part of your direct team. This includes vendors who supply data or model components and upstream teams whose decisions affect your system's design or operation.
3. Document the specific ways each stakeholder group contributes to or influences your system, rather than just listing names and titles. For example, note that your security team reviews threat models and sets deployment constraints, while your legal team provides guidance on data use and compliance obligations, and your infrastructure team manages the computing Resources your system runs on.
4. Include oversight and governance stakeholders who may not be involved in day-to-day development but have authority over key decisions about your system. This covers compliance officers who approve deployments, risk management teams who set acceptable use policies, and executive sponsors who control funding and strategic direction.
5. Create a stakeholder contact list with clear points of contact for each group, including backup contacts and escalation paths for important decisions. Keep this list updated as team structures change and make sure everyone on your team knows who to reach out to for different types of questions or approvals throughout the system's development and operation.

Resources

Related documents:

- ISO/IEC 42001:2023 A.3.2 AI roles and responsibilities
- NIST Artificial Intelligence Risk Management Framework (NIST AI 100-1): MAP1.1, MAP1.2, GOVERN5.1

Refine your use case

RAIUC03: How will you refine your use case understanding without detailing the technical solution?

Prior to designing or prototyping a solution, imagine the AI system solving the use case as just a box containing an unknown mechanism. Anticipate the inputs and outputs to the box, how the inputs and outputs might vary, and the kind of information (or type of AI) the mechanism in the box would need to succeed.

Best practices

- RAIUC03-BP01 Identify the expected input and outputs for the AI system
- RAIUC03-BP02 Identify how your expected inputs could vary in their content
- RAIUC03-BP03 Identify the type of AI required by your AI use case

RAIUC03-BP01 Identify the expected input and outputs for the AI system

Imagine the AI system solving the use case as a box containing an unknown mechanism. Describe the inputs to the AI system. Stay at a high level, focusing, for example, on whether inputs might contain spoken English text and images, but not on the specific audio or image filetypes. Consider what information is present in the input signal, and whether that information is enough to infer the desired outputs. Consider whether the inputs and outputs differ from how the use case is currently solved.

Level of risk exposed if this best practice is not established: High

Implementation considerations

1. Define your AI system's inputs at a high level by describing the types of information that will flow into your system, such as text in multiple languages, images, audio recordings, or structured data like user preferences or transaction histories. Focus on the content and meaning rather than technical formats. For example, you could define inputs as *customer support conversations* rather than *MP3 audio files*.

2. Specify what your AI system should produce as outputs, describing the type of information it will generate. This might include text responses, classification labels, recommendations, generated content, or structured data that downstream systems can use to take actions.
3. Analyze whether your inputs contain enough information to reliably produce your desired outputs by thinking through the logical connection between what you're giving the system and what you expect it to produce. If you want your system to diagnose medical conditions but only provide it with basic symptoms, consider whether that's sufficient or if you need additional input like medical history or test results.
4. Compare your AI system's inputs and outputs to how the problem is solved today without AI, identifying what's different and what stays the same. For example, if human customer service agents currently handle inquiries using phone calls and internal knowledge bases, but your AI will work with chat messages and documentation, note these differences and consider their implications.

Resources

Related documents:

- ISO/IEC 42001:2023 A.6.2.2 AI system requirements and specification
- NIST Artificial Intelligence Risk Management Framework (NIST AI 100-1): MAP2.1, MAP2.2

Related tools:

- Improve accuracy by adding Automated Reasoning checks in Amazon Bedrock Guardrails
- Use contextual grounding check to filter hallucinations in responses

RAIUC03-BP02 Identify how your expected inputs could vary in their content

Identify the ways in which inputs to the AI system might systematically vary under real-world conditions. For example, the inputs to system that transcribes speech in audio recordings might vary by background noise, physical characteristics of the voices, or the sensitivity of the microphone. Or, inputs to chatbot could vary by language, use of slang or jargon, or word spellings ("analyze" vs "analyse"). Decide whether each type of variation is something the AI system should attend to, or ignore.

Level of risk exposed if this best practice is not established: High

Implementation considerations

1. Review examples of potential real-world inputs to identify the types of intrinsic and confounding variations. Consider how inputs are sourced (for example, sensor types, environmental conditions, and potentially pre-processed). Consider variations across different user segments, geographic regions, and time periods.
    a. *Intrinsic variation* refers to differences in input data to which AI system should attend to succeed.
    b. *Confounding variation* refers to differences in input data that an AI system should ignore to succeed.
    c. For example, when comparing two images of faces to determine if the images are of the same person, an AI system must look at differences in pixel intensities that are due to facial geometry (like the width of the nose) and skin albedo (including scars, tattoos, and natural skin coloration), but not pixel differences due to camera angle, facial expression, or scene lighting. The first variations are intrinsic and the second are confounding.
2. Consider whether data capturing intrinsic or confounding variations can be synthesized.
3. Identify edge cases and other out-of-distribution scenarios that might affect system reliability.

Resources:

Related documents:

- Responsible AI Practices
- ISO/IEC 42001:2023 A.7.4 Quality of data for AI systems
- NIST Artificial Intelligence Risk Management Framework (NIST AI 100-1): MAP2.1, MAP2.2, MAP2.3

RAIUC03-BP03 Identify the type of AI required by your AI use case

Selecting the appropriate type of AI solution is a critical decision that fundamentally shapes your project's success and risk profile. Your choice of traditional ML, generative AI, or agentic AI must align with your specific use case requirements, data availability, and desired outcomes. The decision impacts everything from development complexity and resource requirements to explainability and risk management needs. A misaligned choice can lead to project failure, increased costs, or unmanageable risks, while the right selection creates a foundation for successful AI implementation that meets business objectives while maintaining appropriate controls.

Level of risk exposed if this best practice is not established: High

Implementation considerations

1. Determine if your use case primarily involves recognizing patterns in complex but pre-defined input data. If so, you may need traditional ML. Examples include fraud detection, demand forecasting, or quality control systems where patterns exist but are too complex for explicit rules.
2. Determine if your use case requires understanding widely varying inputs, including natural language, and creating new content or providing human-like responses. If so, you may need Generative AI. Examples include media creation, code generation, and advanced customer chatbots.
3. Determine if your use case requires breaking down high-level user objectives into workflows, and potentially reconfiguring the workflows depending on the results of intermediate tasks, as opposed to just responding to queries or making predictions. The use of a **natural language interface** for users to communicate these complex, high-level intents and receive updates is one of the primary characteristics of this approach. If so, you may need agentic AI. Examples include research and travel assistants.

Resources

Related documents:

- AWS AI Services
- AWS Responsible AI
- ISO/IEC 42001:2023 A.6.2.2 AI system requirements and specification
- NIST Artificial Intelligence Risk Management Framework (NIST AI 100-1): MAP2.1, MAP2.2, MAP2.3

AI workflow impact

RAIUC04: How will you anticipate the impact of your AI solution on the workflow in which it is deployed?

Mapping user journeys can identify unexpected points of friction and misunderstood design requirements and provide guidance on where transparency and human oversight may be most useful.

Best practices

- RAIUC04-BP01 Map the user journey to identify AI interaction requirements
- RAIUC04-BP02 Identify human oversight opportunities
- RAIUC04-BP03 Identify accessibility requirements for different user groups

RAIUC04-BP01 Map the user journey to identify AI interaction requirements

Map the user journey to identify interaction requirements and risks. During pre-interaction, assist users in learning about the system's capabilities and limitations.

During the initial interaction, consider the different accessibility needs of users, and the different sources for key system inputs.

During processing, maintain transparency about AI decision-making and provide appropriate progress indicators. Post-interaction, enable users to understand, challenge, and provide feedback on AI outputs, which keeps the system accountable and improvable.

Consider how different user groups might be affected differently at each stage and implement appropriate safeguards and support mechanisms. If the AI system will be embedded in an existing human-powered workflow, consider what purposes the workflow might address that the AI system does not, and consider the variety of ways in which users might modify system inputs and outputs for other purposes.

Level of risk exposed if this best practice is not established: High

Implementation considerations

1. Identify the expectations users may have and the information they need before engaging with the proposed AI system.
2. Sketch the initial interaction phase. Identify the first touchpoints where users provide input and the guidance they need for effective interaction. Consider how users might alter inputs to influence outputs.
3. Sketch the AI processing phase. Consider how long processing takes, what users see during processing, and what information assists users to understand system activity and confidence levels.
4. Sketch the post-interaction phase. Plan how users receive, interpret, and act on AI outputs, including confidence indicators, explanation features, and guidance for appropriate use of results. Consider the purposes which the outputs might serve.
5. Identify transparency touch points. Mark specific moments in each phase where guidance is most valuable.

Resources

Related documents:

- ISO/IEC 42001:2023 - User interaction lifecycle implementation
- ISO/IEC 42001:2023 A.5.4 Assessing AI system impact on individuals or groups of individuals
- NIST Artificial Intelligence Risk Management Framework: Generative Artificial Intelligence Profile (NIST AI 600-1):
- NIST Artificial Intelligence Risk Management Framework (NIST AI 100-1): MAP2.1, MAP2.2, MAP2.3

RAIUC04-BP02 Identify human oversight opportunities

Place human review at points where the quality of system inputs or outputs can be harder to judge. Consider moments where human expertise adds unique value or assists with consequential decisions.

Level of risk exposed if this best practice is not established: High

Implementation considerations

1. Review the user journey to find moments where errors could have significant consequences, or where human expertise is uniquely valuable. Create user interface elements and workflows that make human oversight natural and efficient, providing the right information at the right time for effective decision-making.
2. Identify requirements to assist human reviewers, such as system output explanations and contributing factors, confidence scores, similar case examples, and other information needed for informed oversight decisions.

Resources

Related documents:

- ISO/IEC 42001:2023 A.6.2.6 AI system operation and monitoring

RAIUC04-BP03 Identify accessibility requirements for different user groups

Identifying accessibility points assists to generate requirements for people with different capabilities and disabilities to use the proposed AI system effectively.

Level of risk exposed if this best practice is not established: High

Implementation considerations

1. Conduct accessibility needs assessment. Research the specific accessibility challenges faced when interacting with AI systems, including cognitive, visual, auditory, speech, sensory, and motor accessibility needs. You may explore research published in the ACM CHI: Conference on Human Factors in Computing Systems, or consider reading more from articles published from Amazon: Use AWS AI and ML services to foster accessibility and inclusion of people with a visual or communication disability or 12 ways Amazon is making products more accessible for customers with disabilities.
2. Map out multimodal interactions. Provide multiple ways for users to input information and receive AI outputs, including voice, text, visual, and tactile options as appropriate for your system.

3. Consider if AI explanations, confidence indicators, and system status information would be available in formats accessible to users with different abilities (screen reader compatible, high contrast, simplified language options). You may read more from AWS Accessibility or 12 ways Amazon is making products more accessible for customers with disabilities.

Resources

Related documents:

- AWS Accessibility
- Use AWS AI and ML services to foster accessibility and inclusion of people with a visual or communication disability
- Exploring accessible audio descriptions with Amazon Nova | Artificial...
- 12 ways Amazon is making products more accessible for customers with disabilities
- ISO/IEC 42001:2023 A.8.2 System documentation and information for users
- Sign-Speak builds with AI on AWS to create accessible experiences
- Accessible Rich Internet Applications (ARIA)

Identify requirements and approvals

RAIUC05: How will you identify applicable organizational requirements and approvals?

AI systems may involve data processing, contributors, and users that span multiple geographic locations and legal jurisdictions. This may result in the AI system needing to meet multiple differing, overlapping, and organizational obligations. Consult organizational experts to identify and clarify requirements.

Best practices

- RAIUC05-BP01 Engage your organization in approving your use case

RAIUC05-BP01 Engage your organization in approving your use case

Identify the geographic locations in which the proposed AI system will operate. Consult with your legal team to identify applicable regulatory requirements. Check your organization's policies and processes for the approval of AI use cases. They may establish governance procedures that outline approval requirements and designate responsible oversight bodies to evaluate and authorize AI initiatives.

Level of risk exposed if this best practice is not established: High

Implementation considerations

1. Identify the geographic locations where your AI system will operate, including where data will be collected and processed and where decisions will be made or applied. This geographic scope may determine which laws and regulations apply to your system, so be specific about countries, states, or regions rather than using a simple global or multi-regional scope.
2. Work with your legal team to understand the specific regulatory requirements that apply to your use case in each geographic location, including AI-specific regulations, data protection laws, and industry-specific rules. Make sure to understand both current requirements and upcoming regulations that might affect your timeline.
3. Find and review your organization's existing policies for AI use cases, including responsible AI principles, data governance standards, and approval procedures that specify what documentation you need to provide. Determine which teams or committees need to sign off on your project before you can proceed.
4. Connect with the appropriate governance bodies or approval committees in your organization to understand their evaluation process, timeline expectations, and information they need from you to approve your AI use case. Schedule early conversations to get guidance on how to structure your proposal and what potential concerns they might have about your specific use case.

Resources

Related documents:

- AWS Privacy
- AWS GDPR Center
- AWS Compliance Programs
- Data Privacy Center

- ISO/IEC 42001:2023 A.2.2 AI policy

Benefits and risks

The benefits and risks focus area assists you to characterize the potential benefits and inherent risks of the use case, using only minimal assumptions about the AI solution given in the Use Case focus area. In general, the depth of analysis devoted to potential benefits and inherent risks should align with the breadth and complexity of the use case, and the potential scale of deployment. Iteration is expected. Additionally, since use cases are often defined by working backwards from desired benefits, risks may require more effort to explore than benefits.

Technically, benefits and risks can be assessed similarly. Both are based on the likelihood and impact of system interactions, where a system interaction typically involves a sequence of input and output pairs, which can be considered as *events* in the interaction. A helpful interaction has a positive impact on a stakeholder, while a harmful interaction has a negative impact on a stakeholder.

The result of an interaction can be the compounding effect of multiple events. To estimate benefits, you identify examples of helpful interactions, categorize the examples into benefits, and then use your knowledge of your business to estimate the value of each benefit. Assessing risk is trickier, because there is usually less information available about harmful interactions. As a result, you first identify categories of potential harmful interactions, qualitatively estimate the likelihood and impact severity of each harmful interaction category, and map the qualitative estimates to a risk level. In most cases, benefits and risks should be assessed for each stakeholder group.

Note that the risk posed by the AI system implementation when applied to the use case is referred to as residual risk. This is computed in *Monitoring*, as part of the process of deciding whether the AI system is ready for release.

Focus areas

- Characterize benefits
- Harmful events
- Assess risks
- Mitigate risks

Characterize benefits

RAIBR01: How will you characterize intended benefits?

Identify the benefits to each type of downstream stakeholder. Example benefits could include efficiency improvements, cost reductions, and enhanced experiences. Benefits are the fundamental justification for your system and provide a foundation for making trade-off decisions throughout the development lifecycle. Capture the highest impact benefits, irrespective of how precisely they can be measured. This approach assists you to align development priorities with stakeholder needs rather than being driven by technical capabilities alone.

Best practices

- RAIBR01-BP01 Aggregate beneficial events into intended benefits

RAIBR01-BP01 Aggregate beneficial events into intended benefits

Identify the specific beneficial events that could assist each type of downstream stakeholder. Translate these events into specific intended benefits for the use case.

Level of risk exposed if this best practice is not established: High

Implementation considerations

1. For each downstream stakeholder, identify examples of system interactions (input and output pairs) that you consider to be beneficial, and group the examples into different categories of benefits.
2. Score each benefit on impact and likelihood, using your existing knowledge (from the Use Case focus area) about the workflow.
3. Prioritize the benefits that are critical to your organization's success.

Harmful events

RAIBR02: How will you identify and categorize potential harmful events?

Use a structured approach to identify harmful events that could potentially occur when solving this use case with the proposed AI system.

Best practices

- RAIBR02-BP01 Identify potential harmful events impacting fairness
- RAIBR02-BP02 Identify potential harmful events impacting veracity
- RAIBR02-BP03 Identify potential harmful events impacting robustness
- RAIBR02-BP04 Identify potential harmful events impacting privacy
- RAIBR02-BP05 Identify potential harmful events impacting safety
- RAIBR02-BP06 Identify potential harmful events impacting system and data security
- RAIBR02-BP07 Identify potential harmful events impacting explainability
- RAIBR02-BP08 Identify potential harmful events impacting transparency
- RAIBR02-BP09 Choose multiple strategies to identify potential harmful events

RAIBR02-BP01 Identify potential harmful events impacting fairness

Examine how the proposed AI system might affect different stakeholder groups and subgroups throughout the entire system lifecycle. A fairness assessment may consider harms to individuals (for example, wrongful denials) and to groups (for example, performance variations across demographic groups).

Level of risk exposed if this best practice is not established: High

Implementation considerations

1. Consider how different demographic groups are represented in the inputs (for example, by geography).
2. Consider whether some inputs could unintentionally represent or misrepresent different demographic groups (for example, proxy a demographic attribute).
3. Consider whether training data might inappropriately represent the expected users and whether a wider variety of inputs could impact performance. For example, a facial recognition system trained primarily on certain skin tones might not perform as well on other skin tones.
4. Assess potential impacts at the levels of individuals, groups, and society. For example, a job candidate screening tool might impact individual candidates, demographic group success rates, and overall workforce representation.

Resources

Related documents:

- Preventing Fairness Gerrymandering: Auditing and Learning for Subgroup Fairness
- Equality Of Odds
- Fairness, model explainability and bias detection with SageMaker AI Clarify
- ISO/IEC 42001:2023 A.5.4 Assessing AI system impact on individuals or groups of individuals
- ISO/IEC 42001:2023 A.7.4 Quality of data for AI systems

Related tools:

- Amazon SageMaker AI Clarify

RAIBR02-BP02 Identify potential harmful events impacting veracity

Veracity harms arise when AI systems produce factual errors, as measured against an established base set of facts. Errors include hallucinations, omissions, and misemphases. These errors can propagate through AI systems, affecting downstream decision-making processes. Hallucinations and other veracity-related issues can compound across other responsible AI dimensions to create complex patterns of harm.

Level of risk exposed if this best practice is not established: High

Implementation considerations

1. Consider which facts, if any, will be represented in outputs.
2. Consider how you will validate that a fact is true. What are your reference sources? How subject to debate will output facts be?
3. Consider the implications of a veracity error propagating through your AI system or the workflow you are trying to improve with the AI system. How does inaccurate information spread through system interactions and user networks?
4. Consider how factual inaccuracies interact with other responsible AI considerations, like fairness or safety. For example, an AI system's veracity errors might exacerbate unwanted biases.

Resources

Related tools:

Evaluate the performance of Amazon Bedrock Resources

- Amazon Bedrock Knowledge Bases

RAIBR02-BP03 Identify potential harmful events impacting robustness

Mishandling foreseeable variations in inputs can create harmful events. Input variations come in two kinds. Intrinsic variations are differences in input data to which an AI system must attend to succeed. Confounding variations are differences in input data that an AI system must ignore to succeed. You should also consider whether slight changes in input data can produce dramatically different outputs and how input instabilities can cascade across system components.

Level of risk exposed if this best practice is not established: High

Implementation considerations

1. Map input variation scenarios and their potential harmful impacts. Consider medical imaging AI, where varying equipment calibrations and scan qualities influence diagnostic accuracy. Document how differences in data format, quality, and characteristics affect reliability.
2. Analyze how input patterns shift over time to identify distribution harms. For example, recommendation systems should adapt to evolving user preferences and emerging content categories. Seasonal trends and special events often introduce unexpected usage patterns.
3. Consider cascading effects in multi-step workflows. In a multi-step AI workflow where one model's output feeds into another, assess how initial inaccuracies could amplify through the chain. For example, in a document processing system, errors in text extraction might affect subsequent classification or summarization steps.

Resources

Related documents:

- Improve LLM application robustness with Amazon Bedrock Guardrails and Amazon Bedrock Agents
- ISO/IEC 42001:2023 A.5.4 Assessing AI system impact on individuals or groups of individuals

- ISO/IEC 42001:2023 A.7.4 Quality of data for AI systems

Related tools:

Evaluate the performance of Amazon Bedrock Resources

- Amazon SageMaker AI Clarify

RAIBR02-BP04 Identify potential harmful events impacting privacy

Harmful events can result from using data that is confidential or personal in ways that do not align with the rules for correctly handling such data.

Level of risk exposed if this best practice is not established: High

Implementation considerations

1. Review the types of data that you expect to appear in development and operations (including user inputs and system outputs), and categorize the data as confidential, personal or other, as advised by your legal counsel. Consider harmful events resulting from errors in handling this data. For example, could data that is licensed only for training be accidentally output to a user?
2. Consider what types of data might unexpectedly appear in training or operations, whether the unexpected data could be confidential or personal, and what harms might result if this data was not blocked from flowing into development or operational pipelines.

Resources

Related documents:

- Differentially Private Fair Learning
- Remove PII from conversations by using sensitive information filters
- ISO/IEC 42001:2023 A.5.4 Assessing AI system impact on individuals or groups of individuals
- ISO/IEC 42001:2023 A.7.3 Acquisition of data

Related video:

- Amazon Bedrock Guardrails: Implementing Custom Safeguards for Responsible AI Applications

- AWS re:Inforce 2025 - Privacy-first generative AI: Establishing guardrails for compliance (COM224)

RAIBR02-BP05 Identify potential harmful events impacting safety

System outputs (content or actions) might create unintended impacts on the health or well-being of individuals, groups, society or the environment and can be misused in ways that could cause harm. Unsafe inputs can create harmful system responses. Understanding safety harms requires examining both immediate harms and downstream effects across different stakeholder groups, while considering how safety violations might cascade through system operations and user interactions.

Level of risk exposed if this best practice is not established: High

Implementation considerations

1. Consider if inputs could request content that the system is not designed to handle. For example, in medical advice use cases, can generated content present improper self-treatment recommendations or cause psychological distress through insensitive delivery? Consider both direct and indirect harm potential.
2. Consider input handling safety concerns and response protocols. For example, AI chatbots may need systems to detect crisis signals in user inputs and provide appropriate responses while avoiding harmful advice.
3. Consider physical, psychological, and environmental impacts. For example, could an incorrect instruction to a smart home system create a safety hazard?

Resources

Related documents:

- Amazon Bedrock Guardrails enhances generative AI application safety with new capabilities
- Measuring and Mitigating Toxicity in LLMs
- ISO/IEC 42001:2023 A.5.4 Assessing AI system impact on individuals or groups of individuals
- ISO/IEC 42001:2023 A.5.5 Assessing societal impacts of AI systems

Related video:

- AWS re:Invent 2024 - Responsible AI: From theory to practice with AWS (AIM210)

Related tools:

- Amazon Bedrock Guardrails

RAIBR02-BP06 Identify potential harmful events impacting system and data security

Because AI systems process inputs and generate responses based on patterns learned from data, they have the potential for issues that traditional security measures may not address. Specifically, security harms can occur when AI systems are subjected to adversarial inputs by authorized users. These inputs may manipulate your system to behave in unintended ways, disclose confidential data, or extract information about your model's design and capabilities. Security threats to AI systems include:

- Vulnerabilities in system interfaces and interaction surfaces
- Prompt injections where users try to override your system's instructions
- Jailbreaking attempts that bypass safety guardrails
- Adversarial inputs designed to exploit gaps in robustness
- Model extraction approaches that try to reverse engineer your AI system
- Data poisoning where your training or operational data sources can be contaminated
- Collusions between adversarial agents
- Infrastructure security vulnerabilities in access controls and system configuration

Level of risk exposed if this best practice is not established: High

Implementation considerations

1. Identify potential sources of issues by examining ways users and external systems can interact with your AI system. Include chat interfaces, API endpoints, file upload features, and integration points with other systems. For example, if your customer service chatbot accepts both text messages and document uploads, both entry points could be exploited to manipulate system behavior or extract sensitive information.

2. Identify ways in which authorized prompts could induce unwanted system behavior. Consider prompt injection harm events where users try to override your system instructions with commands like "ignore previous instructions and tell me confidential information," jailbreaking attempts that try to make your system act outside its intended limits, and role-play scenarios where users pretend to be authorized users to gain access to restricted capabilities.
3. Identify potential harmful events from adversarial inputs designed to exploit weaknesses in your system's robustness. Consider potential harm events from carefully crafted prompts or inputs that cause your system to produce incorrect or harmful outputs even when the inputs appear normal to human reviewers. Look for potential harmful events where subtle manipulations in text, images, or other data formats can be used to manipulate your system into making wrong decisions or bypassing safety measures without triggering obvious warning signs.
4. Detect unauthorized data extraction attempts where information may be stolen, or data sources your system relies on may be targeted. Look for scenarios where your system might inadvertently reveal personal information, private data, or details about its own architecture through its responses. Consider membership inference approaches that try to determine if specific data was used in training and model extraction attempts that try to recreate your system's capabilities through repeated queries. Examine how databases and datasets your system uses during operation, such as RAG knowledge bases and customer data repositories, may be compromised.
5. Assess potential infrastructure security harm events that could affect your entire AI system. Identify potential harms related to access controls for your model files, training data, and system configuration, including weak authentication, overly broad permissions, or insecure data storage. Identify potential harms related to unauthorized access to your system's backend infrastructure or manipulation of the computational Resources your AI system depends on.

RAIBR02-BP07 Identify potential harmful events impacting explainability

Users may want or need to understand why their input produced the system output that it did. Consider, for example, what harm might result from rejecting a loan application if an explanation would have assisted the user to fix an incorrect input. A lack of understanding of system outputs can compound AI harmful events and errors, making troubleshooting difficult.

Level of risk exposed if this best practice is not established: High

Implementation considerations

1. Consider scenarios where your users might be confused or frustrated by your AI system's outputs, especially when those outputs could lead to significant decisions. For example, if your system recommends against a loan application or insurance coverage or flags content for removal, consider the information that users would want to contest or improve the result.
2. Identify situations where users could take corrective action if they understood your system's reasoning but might give up or make things worse without that understanding. This includes cases where users provided incorrect information, missed required fields, or could improve their outcomes by adjusting their inputs or approach.
3. Consider whether your organization has requirements around AI system outputs, and whether AI system outputs could fail to meet those requirements.
4. Consider how a lack of explanation might amplify other problems with your system by making it harder for users to provide feedback, for operators to troubleshoot issues, or for your team to identify when the system is making systematic errors.
5. Look for situations where misunderstanding your system's outputs could lead users to make harmful decisions themselves, such as ignoring important warnings, over-relying on uncertain recommendations, or losing trust in legitimate system outputs. Think about both immediate harms to individual users and broader impacts if many people misunderstand how your system works.

Resources

Related documents:

- Advanced tracing and evaluation of generative AI agents using LangChain and Amazon SageMaker AI MLFlow
- Build verifiable explainability into financial services workflows with Automated reasoning checks using Bedrock Guardrails
- ISO/IEC 42001:2023 A.5.4 Assessing AI system impact on individuals or groups of individuals
- ISO/IEC 42001:2023 A.8.2 System documentation and information for users

Related videos:

- Amazon Bedrock AgentCore - Observability | Amazon Web Services

- AWS re:Invent 2024 - Building explainable AI models with Amazon SageMaker AI (DEV219)

Related tools:

- Amazon Bedrock Guardrails

RAIBR02-BP08 Identify potential harmful events impacting transparency

Transparency is the degree to which stakeholders can make informed choices in their engagement with an AI system. Consider situations in which users do not understand the probabilistic nature of an AI system, are unaware of AI system presence, or may not realize that an output is AI-generated.

Level of risk exposed if this best practice is not established: High

Implementation considerations

1. Identify decision points where users rely on AI outputs. For example, in healthcare AI use cases, identify where patients or providers make treatment decisions based on AI recommendations. Consider impact severity if users are unaware of system confidence levels or limitations.
2. Consider differing levels of expertise among stakeholder groups.
3. Evaluate how transparency gaps might hide or amplify other harms. Consider medical diagnosis systems where unclear AI involvement could lead to overreliance on automated assessments, potentially compromising patient safety.

Resources

Related documents:

- NIST AI Risk Management Framework: Emphasizes transparency in the "Govern" and "Manage" functions
- ISO/IEC 42001:2023 A.5.4 Assessing AI system impact on individuals or groups of individuals
- ISO/IEC 42001:2023 A.8.2 System documentation and information for users

RAIBR02-BP09 Choose multiple strategies to identify potential harmful events

In addition to assessing potential harmful events for each responsible AI dimension independently, employ complementary strategies to identify potentially harmful events and negative stakeholder impact within the context of different use environments. Check for these events at different steps of using the AI system and under different failure modes, which includes both technical failures and misuse or abuse of the AI system. Additional strategies include scenario-based analyses, system limitation assessments that surface operational constraints, choosing a risk team with diverse backgrounds, consulting with external stakeholders, and reviewing historical incidents or risk assessment results from similar systems.

Level of risk exposed if this best practice is not established: High

Implementation considerations

1. Choose which strategies are appropriate to use for your design and development process and assign owners to track the progress and iterations of each employed scenario.
2. Establish a standardized documentation process for recording identified harmful events across different contexts.
3. Implement regular review cycles to reassess potential harms as the system evolves and establish feedback channels for continuous input from diverse team members and external stakeholders.

Resources

Related documents

- Learn how to assess the risk of AI systems

Assess risks

RAIBR03: How will you assess the risks posed by the potential harmful events?

Assess the risk level of potential harms based on their likelihood and severity. Risk rankings assist to prioritize investments in mitigations and to incorporate proper controls in the design phase. Establish your methodology for assigning risk before performing the assessment. If your organization has already developed a methodology, use it so that your organization can make equal risk comparisons across use cases. Register risk for tracking and informed decision-making across AI use cases.

Best practices

- RAIBR03-BP01 Identify the likelihood of each potential harm
- RAIBR03-BP02 Identify the severity of each potential harm
- RAIBR03-BP03 Assign an overall risk level to each potential harm
- RAIBR03-BP04 Use a risk registry to track and calibrate potential harms and risks

RAIBR03-BP01 Identify the likelihood of each potential harm

Establish a risk rating methodology that considers the likelihood of the event occurring. The risk likelihood indicates the probability of a harmful event occurring when the system is deployed for the use case.

Level of risk exposed if this best practice is not established: High

Implementation considerations

1. Create a standardized likelihood scale with clear definitions. For example, establish ranges from *almost certain* (95%+ probability) to *highly unlikely* (less than 5% probability). Include specific frequency ranges for each category to maintain consistent evaluation.
2. Document likelihood assessments with supporting evidence. For example, consider a content moderation system where historical data shows harmful content detection failures occur in 15% of edge cases, placing this risk in the *unlikely* category. Include rationale for each assessment.

Resources

Related documents:

- Learn how to assess the risk of AI systems
- NIST Risk Management Framework
- Responsible AI in the generative era
- ISO/IEC 42001:2023 A.5.2 AI system impact assessment process

RAIBR03-BP02 Identify the severity of each potential harm

Risk severity estimates the magnitude of the negative on affected stakeholder groups if it were to occur. Severity also considers the reversibility of harm, recognizing that some types of harm may be permanent or difficult to remedy.

Level of risk exposed if this best practice is not established: High

Implementation considerations

1. Create standardized severity scale with clear impact levels. For example, establish a range from *low* (minimal, reversible impact) to *extreme* (substantial, long-lasting impact). Include specific criteria for each level to enable consistent evaluation.
2. Evaluate harm severity considering multiple factors. As an example, in medical AI systems, incorrect diagnoses might have *major* severity due to potential health impacts and difficulty in reversing treatment decisions. Consider immediate effects, long-term consequences and reversibility.
3. Document severity assessments with supporting evidence. For example, consider financial AI where incorrect investment advice might have a moderate or major severity estimate for impacted users, depending on the context. Include analysis of varying stakeholder impacts.

Resources

Related documents:

- Learn how to assess the risk of AI systems
- NIST Risk Management Framework
- ISO/IEC 42001:2023 A.5.2 AI system impact assessment process

RAIBR03-BP03 Assign an overall risk level to each potential harm

Risk ratings are typically determined by using a risk matrix that combines the likelihood (probability of occurrence) and severity (degree of consequences) of the potential harmful events.

Level of risk exposed if this best practice is not established: High

Implementation considerations

1. Create a consistent risk matrix or scoring system that combines likelihood and severity ratings to produce overall risk levels for each potential harm you've identified. Your matrix should define clear categories for both dimensions. For example, use a one to five scale for likelihood (unlikely to likely) and severity (minimal to extreme impact), then combining these to create overall risk ratings like low, medium, high, or critical.
2. Evaluate the likelihood of each potential harm by considering factors like how often similar issues have occurred in comparable systems, how robust your current mitigations are, and what conditions would need to align for the harm to occur. Be realistic about probabilities rather than assuming your system will work perfectly, and consider both technical failures and misuse scenarios.
3. Assess the severity of each potential harm by thinking through the full scope of consequences if it were to occur, including immediate impacts on affected individuals, broader effects on communities or society, and long-term damage to trust in AI systems. Consider both direct harms and cascading effects that might result.
4. Apply your risk matrix consistently across the identified harms to generate comparable risk ratings, and use the same criteria and standards for each assessment. Document your reasoning for each rating so others can understand and review your risk evaluations and consider having multiple people independently assess potential harms that you haven't considered.
5. Prioritize your risk mitigation efforts based on these overall risk ratings, focusing first on the highest-risk harms while also considering factors like mitigation cost and feasibility. Use these risk levels to guide decisions about which harms need immediate attention, which can be addressed in future iterations, and what level of mitigation investment is appropriate for each type of harm.

Resources

Related documents:

- Learn how to assess the risk of AI systems
- NIST Risk Management Framework
- Responsible AI in the generative era
- ISO/IEC 42001:2023 A.5.2 AI system impact assessment process

RAIBR03-BP04 Use a risk registry to track and calibrate potential harms and risks

Establish a risk registry to track and calibrate categories of risks across your ML lifecycle and other use cases your team or organization may be tackling. The registry includes information about each identified risk, including the associated use case, examples of harmful input and output pairs, affected stakeholders, likelihood, severity, risk level, and high-level mitigation approaches. Risk registry maintenance includes processes for keeping risk information current and accurate as use cases and systems evolve, new threats emerge, and responsible AI understanding deepens.

Level of risk exposed if this best practice is not established: Medium

Implementation considerations

1. Use a secure mechanism to capture each risk, with fields for the associated use case, examples of harmful input and output pairs, affected stakeholders, likelihood, severity, risk level, and high-level mitigation approaches.
2. Create workflows that link each risk in the registry to development artifacts such as release criteria and technical mitigation specifications, and track whether those fixes worked. Record baseline measurements before mitigation, implementation details, and follow-up measurements to see which approaches work best for different risks.
3. Periodically review registered risks to check if mitigations are working and risk assessments were accurate. Compare actual outcomes against predictions and update risk ratings when you have new evidence about likelihood or severity.
4. When starting a new use case, consult the risk registry to speed and calibrate your risk assessments.

Resources

Related documents:

- ISO/IEC 42001:2023 A.5.3 Documentation of AI system impact assessments

Mitigate risks

RAIBR04: How will you mitigate the risks you have identified?

Narrowing the use case may limit the exposure to risks. You can mitigate some by exploring trade-offs with either benefits or other risks.

Best practices

- RAIBR04-BP01 Narrow the use case
- RAIBR04-BP02 Weigh trade-offs across competing use case objectives
- RAIBR04-BP03 Assign your potential harm mitigations to implementation strategies

RAIBR04-BP01 Narrow the use case

Identify the minimum viable use case that still delivers meaningful business value while reducing complexity and associated risks. Narrow the use case to a specific domain, industry vertical, geography, or user segment rather than attempting to solve broad, general problems. Restrict the types of inputs your system accepts and the formats of outputs it generates.

Level of risk exposed if this best practice is not established: Medium

Implementation considerations

1. Evaluate current scope and identify risk reduction opportunities. For example, an AI medical diagnosis system might focus on a specific condition type rather than general diagnostics or limit analysis to structured lab results rather than free-text notes.
2. Define specific boundaries for system application. As an example, a financial AI advisor might serve only retail investors within certain portfolio sizes, using standardized investment products rather than complex instruments. Consider expertise requirements.
3. Document input and output restrictions to control risk exposure. Consider a customer service AI that accepts only structured inputs rather than free-text queries, which improves response reliability. Include clear guidance on system limitations and context for appropriate use.

RAIBR04-BP02 Weigh trade-offs across competing use case objectives

Evaluate and balance trade-offs between benefits and risks. If not already available from your organization, develop explicit trade-off criteria (like tenets) that reflect organizational policies and stakeholder needs.

Level of risk exposed if this best practice is not established: Medium

Implementation considerations

1. Review objectives, benefits, and risks across relevant responsible AI dimensions and stakeholders to identify potential conflicts.
2. Analyze conflicts between different to prioritize their importance. For example, a diagnostic health care use case might trade off overall accuracy against full coverage of disease types, or a financial fraud detection use case might trade off a higher false positive rate against a faster response time.

Resources

Related documents:

- Responsible AI: From Principles to Production
- Resolving Ethics Trade-offs in Implementing Responsible AI

Related videos

- AWS re:Invent 2024 - Responsible AI: From theory to practice with AWS (AIM210)

RAIBR04-BP03 Assign your potential harm mitigations to implementation strategies

As input to your system design, consider whether potential harms can be addressed through technical features or stakeholder guidance.

Level of risk exposed if this best practice is not established: High

Implementation considerations

1. Categorize your mitigations into implementation strategies that are either built into the system (a part of the core AI system or resolved with filtering of inputs and outputs) or addressed through guidance. For example, a healthcare chatbot might reduce the risk of incorrectly responding to requests for legal advice by either customizing the underlying model or guardrails, or by warning users not to request legal advice, or both.

Resources

Related documents:

- Learn how to assess the risk of AI systems
- Responsible AI in the generative era
- NIST Risk Management Framework

Release criteria

The Release Criteria focus area assists you to translate benefits, risks, and organizational obligations from the previous focus areas into a single list of criteria that you use to assess the readiness of the AI solution for release. The list of release criteria defines a high-level specification for the design of system datasets and the AI system. You should consider defining release criteria for the full set of expected benefits and potential harms to reduce downstream surprises post deployment. For AI solutions using machine learning, a full set of criteria would cover factors such as overall usability performance, latency, and uptime, as well as responsible AI factors such as safety, controllability, security, privacy, robustness, fairness, veracity, explainability and transparency.

A release criterion is a binary test that consists of a quantitative assessment and a decision threshold. The quantitative assessment should specify a minimum degree of confidence that the measured value meets the decision threshold. For example:

1. Are we at least 95% confident that mean latency measured from last input token to last output token < 100 milliseconds (yes or no)?
2. Are we at least 99% confident that the maximum disparity between false positive rates across all product categories < 10% (yes or no)?

The release decision itself is also a binary decision that aggregates the results from each individual release criterion. *Evaluate and release* addresses the release decision.

Focus areas

- Define release criteria
- Measure test properties
- Select metrics
- Release criteria thresholds

Define release criteria

RAIRC01: How will you define release criteria?

Set clear, testable criteria to determine when your AI system is ready for deployment by defining binary pass or fail tests for each expected benefit and potential harm.

Best practices

- RAIRC01-BP01 Turn your expected benefits and potential harms into testable release criteria

RAIRC01-BP01 Turn your expected benefits and potential harms into testable release criteria

Turn your identified potential harms and expected benefits into clear yes or no questions that determine if your system is ready for deployment. Each question should address either a specific harm you want to block or a benefit you want your system to deliver. These questions form the basis of your release criteria that should be passed before your system is considered ready for release. Track which stakeholders bear the impact of a failed criterion. You may need multiple criteria for complex harms and benefits. This approach yields a consistent, data-driven approach for determining when your system is ready for release.

Level of risk exposed if this best practice is not established: High

Implementation considerations

1. Take each potential harm and expected benefit you identified and write it as a yes or no question about prevention or delivery. For example, change "users might get biased recommendations" to "Does the system mitigate unwanted bias for each user group?" and "improved response time" to "Does the system improve the response time for user queries?" This assists you to define exactly what success looks like for harm prevention and measure whether your system delivers the expected value.
2. Check that every question can only be answered yes or no based on measurable data, not opinions or interpretations. This reduces ambiguity during evaluation and makes release decisions clear and objective.
3. For each criterion, document the stakeholders who would be impacted if the system failed to meet the release criterion. This assists you to prioritize which criteria are most critical and creates accountability for release decisions.

Resources

Related documents:

- ISO/IEC 42001:2023 A.6.2.4 AI system verification and validation

Measure test properties

RAIRC02: How will you measure the properties tested by your release criteria?

Select appropriate quantitative metrics that can reliably detect and measure the specific information needed to answer the questions in your release criteria.

Best practices

- RAIRC02-BP01 Select metrics to measure the properties tested by the release criteria
- RAIRC02-BP02 Consider strength and limitation trade-offs when choosing metrics
- RAIRC02-BP03 Design a custom metric if no suitable metric exists

RAIRC02-BP01 Select metrics to measure the properties tested by the release criteria

For each release criterion you defined, choose specific metrics that can reliably measure the information needed to answer the question. A single criterion may require multiple metrics to properly measure it. Consider both automated metrics (like accuracy scores and toxicity detection) and human evaluation methods (like expert reviews and user feedback) depending on what you're measuring and explore open-source libraries as well as proprietary services that provide pre-built metrics. Document which metrics map to which criteria so you have a clear measurement plan for every release question you need to answer.

Level of risk exposed if this best practice is not established: High

Implementation considerations

1. Take each yes or no release criterion and identify what specific measurements you need to answer that question. For example, if your criterion is "Does the system respond to queries quickly?", you need response time metrics, or if it's "Does the system block toxic content?", you need toxicity detection scores. Break down abstract criteria into concrete, measurable criteria.

2. Look for existing automated metrics that can measure what you need, such as accuracy scores, response time tracking, or toxicity detection tools. Check open source options like scikit-learn or Hugging Face libraries as well as paid services such as Amazon Bedrock Evaluations. Automated metrics save time and provide consistent measurements you can run repeatedly.
3. Consider using LLM-as-a-judge for criteria that require understanding context, quality, or appropriateness. For example, you can prompt an LLM to evaluate whether responses are helpful, coherent, or follow specific guidelines by giving it examples and scoring rubrics. LLM judges work well for subjective assessments that are too complex for simple automated metrics and are more scalable than human review.
4. Identify which criteria need human evaluation because neither automated metrics nor LLM judges can capture what you're trying to measure. For example, measuring whether user interface designs are intuitive may require actual users to test the interface to better capture the real user experience and preferences. Human evaluation catches the most nuanced issues and is more representative of your user experience but is slower and more expensive.
5. If you find yourself needing multiple different metrics to test one criterion because the criterion itself is complex, consider splitting the criterion into separate yes or no questions. For example, change "Does the system provide a good user experience?" into "Does the system respond quickly?", "Does the system give accurate results?", and "Does the system have an intuitive interface?" This makes each criterion simple to measure definitively.
6. Track which metric you'll use for each release criterion. This gives you a clear testing plan and creates a mapping from your measurements to your release criteria.

Resources

Related documents:

Amazon SageMaker AI AI : Metrics and Validation

Amazon SageMaker AI Canvas : Metrics reference

Evaluating your SageMaker AI AI-trained model

Evaluation metrics and statistical tests for machine learning

ISO/IEC 42001:2023 A.6.2.4 AI system verification and validation

Related tools:

Metrics and scoring: quantifying the quality of predictions

LLM-as-a-judge on Amazon Bedrock Model Evaluation

Hugging Face

Amazon Bedrock Evaluations

RAIRC02-BP02 Consider strength and limitation trade-offs when choosing metrics

Before selecting a metric to measure a release criterion, assess its strengths and weaknesses. Validate model-derived metrics (such as LLM-as-a-judge or -jury) through correlation with human assessors, and document limitations that affect reproducibility (for example, random seed or model version used in LLM-as-a-judge). Evaluate metrics derived from human assessors and annotators for unwanted bias, assessor variance, and consistency. Consider trade-offs between automated metrics, which are generally consistent but may miss context, compared to human evaluation, which may be more nuanced but subjective and harder to scale.

Level of risk exposed if this best practice is not established: High

Implementation considerations

1. Track what each potential metric does well and what it might miss before you choose it. For example, automated accuracy scores are consistent and fast, but might not catch responses that are technically correct but unhelpful to users. Understanding these trade-offs upfront assists you to pick the right combination of metrics.
2. Test LLM-based or model-derived metrics against human evaluators to see how well they agree. Run a set of examples through both your LLM judge and human reviewers, then calculate correlation scores to see if the LLM is measuring what you think it is. This validation catches cases where LLMs might have different responses than humans.
3. Check your human evaluators for bias and consistency by having multiple people evaluate the same examples and comparing their scores. Look for patterns where certain evaluators consistently rate things higher or lower or where people disagree a lot on similar examples. This assists you to spot when human judgment might be unreliable or a task is too subjective.
4. Balance the trade-offs between automated metrics that are consistent but might miss nuance and human evaluation that may be more representative of your users but increases costs and time. Use automated metrics for things you can measure objectively and human evaluation when human feedback is vital.

5. Document your final metric choices and why you picked them, including what limitations you're accepting. This assists future team members understand your reasoning and alerts them to potential blind spots in your measurements.

Resources

Related documents:

- Evaluate the performance of Amazon Bedrock Resources
- Review metrics for an automated model evaluation job in Amazon Bedrock (console)
- Create a model evaluation job with Amazon Bedrock
- ISO/IEC 42001:2023 A.6.2.4 AI system verification and validation

RAIRC02-BP03 Design a custom metric if no suitable metric exists

When creating custom metrics for benefits or potential harmful events, define what you need to measure and its key characteristics. Break complex concepts into quantifiable components that directly relate to stakeholder impacts. Design metrics with definitions and examples of positive and negative results, including edge cases. Validate your custom metric against known examples, choose appropriate measurement scales (like binary, categorical, or continuous), and document the methodology. Plan for refinement based on testing, being cautious of metrics that may not generalize well beyond initial testing.

Level of risk exposed if this best practice is not established: High

Implementation considerations

1. Clearly define what you're trying to measure and write down its key characteristics, focusing on how it directly impacts your stakeholders. For example, if you need to measure how natural a conversation is, define what makes a conversation feel natural or robotic to your specific users. This foundation assists to build an accurate metric.
2. Break complex concepts down into smaller pieces that you can count or score. For example, split user satisfaction into task completion rate, time to complete, and user survey scores, as you can measure each of these objectively. This makes abstract concepts concrete and measurable.
3. Identify what good and bad results look like, including edge cases that might confuse your metric. Define that a helpful response should be accurate, relevant, and actionable. Clear examples reduce confusion during measurement.

4. Test your custom metric on examples where you already know what the right answer should be. Run your metric on obviously good and obviously bad examples to see if it gives the results you expect. This catches major problems with your metric design before you use it on real data.
5. Choose the types of scores your measurement needs. Continuous scores give you more nuanced information and let you track gradual improvements, while categorical ratings are simpler for humans and LLM judge models to assign consistently and binary scores simplify the metric but can hide performance nuance.
6. Document exactly how to calculate your metric, including step-by-step instructions that someone else could follow to get the same results. This blocks inconsistency when different team members apply your metric and assists you to spot problems in your methodology
7. Plan to refine your metric based on real testing since custom metrics often need adjustment after you see how they perform. Start with small tests and be ready to modify the metric if it doesn't work well in practice or gives misleading results on new types of data.

Resources

Related documents:

Use custom metrics to evaluate your generative AI application with Amazon Bedrock

ISO/IEC 42001:2023 A.6.2.4 AI system verification and validation

Related tools:

scikit learn : make_scorer

Select metrics

RAIRC03: How will you select metrics for the release criteria across each responsible AI dimension?

Consider specialized metrics for release criteria related to your identified potential harms.

Best practices

- RAIRC03-BP01 Measure safety harms and harmful outputs
- RAIRC03-BP02 Measure fairness as unwanted bias across stakeholder groups

- RAIRC03-BP03 Measure veracity of outputs
- RAIRC03-BP04 Measure robustness of outputs to input variation
- RAIRC03-BP05 Measure privacy protection
- RAIRC03-BP07 Measure user controllability of system behavior
- RAIRC03-BP08 Measure explainability of system behavior
- RAIRC03-BP09 Measure security risks and threats
- RAIRC03-BP10 Measure transparency quality

RAIRC03-BP01 Measure safety harms and harmful outputs

Create objective definitions of safe and unsafe content for your use case by considering both direct potential harms and contextual inappropriateness. Identify harm categories relevant to possible outputs of your system (for example, toxicity or violence). For identified harm categories, select metrics and plan tests with both quantitative (for example, model-based toxicity classifiers) and qualitative evaluation strategies (for example, human red-teaming). Supplement your safety evaluation with popular open-source benchmarks (like ToxiGen and AdvBench) and Resources (like Detoxify), and choose metric types that are appropriate for the risk of your use case.

Level of risk exposed if this best practice is not established: High

Implementation considerations

1. Write clear and objective definitions of what counts as safe and unsafe content for your specific use case by creating measurable criteria and concrete examples of acceptable and unacceptable outputs. Include both direct harms like violence or toxicity and contextual problems like inappropriate tone for your audience, with specific thresholds and boundaries that evaluators can apply consistently. Objective definitions reduce subjective interpretation and assist evaluators apply consistent standards.
2. Identify the specific harm categories that your system could potentially produce, such as toxicity, violence, misinformation, or inappropriate content for your target users. Focus on harms that are realistic given your system's purpose and capabilities rather than trying to cover every possible risk. This targeted approach assists you to allocate evaluation resources effectively.
3. Choose quantitative metrics like automated toxicity classifiers or content filtering tools that can measure your identified harm categories at scale. Test popular tools like Detoxify or Perspective API on sample outputs to see how well they detect the types of harmful content your system might produce. Automated metrics give you consistent measurement across large datasets.

4. Plan qualitative evaluation methods like human red-teaming where experts try to get your system to produce harmful outputs through adversarial prompting. Have safety experts or domain specialists review sample outputs for harms that automated tools might miss. Human evaluation catches nuanced safety issues that automated systems may overlook.
5. Supplement your custom evaluation with open-source benchmarks like ToxiGen or AdvBench that test for common safety problems. Run these standard tests alongside your custom evaluation to compare your system's performance against known safety baselines. This provides additional validation and assists to identify blind spots in your custom evaluation approach.
6. Match your evaluation intensity to your system's risk level by using more thorough testing for higher-risk applications. For example, consider using basic automated screening for low-risk creative tools but adding human red-teaming for systems that might influence important decisions. Appropriate evaluation depth blocks both over-testing low-risk systems and under-testing higher-risk ones.

Resources

Related documents:

- NIST AI Risk Management Framework
- Build a robust text-based toxicity predictor
- ISO/IEC 42001:2023 A.6.2.4 AI system verification and validation

Related tools:

- Perspective API
- Detoxify
- ToxiGen
- AdvBench
- Bedrock Evaluations

RAIRC03-BP02 Measure fairness as unwanted bias across stakeholder groups

Measure variations across relevant stakeholder groups based on your specific use case and context. This evaluation may include identifying appropriate fairness metrics that align with your use case requirements and could examine consistency at both individual and group levels. Technical approaches for measuring variations in system performance may include metrics such as demographic parity, equal outcome rates, equalized odds and equal opportunity to understand the experience of different groups using the system. Balance these different fairness metrics based on your use case context, as optimizing for one type of fairness may sometimes conflict with others.

Level of risk exposed if this best practice is not established: High

Implementation considerations

1. Measure individual fairness by testing whether similar individuals get similar treatment regardless of their demographic characteristics.
2. Measure group fairness by comparing your system's performance across the demographic groups you identified in your risk assessment (RAIBR02) using metrics like accuracy, precision, and recall. Calculate performance differences between groups and compare them to your acceptable thresholds to identify potential biases. Group-level measurement reveals systemic unwanted bias that may have larger impacts (like bias across entire groups).
3. Test for representational fairness by analyzing whether your system's outputs reinforce harmful stereotypes or misrepresent different groups. Use existing tools like stereotype detection classifiers or analyze generated content for biased language patterns. This catches subtle bias that may not show up in performance metrics but still causes harm.
4. Consider testing your system on pairs of similar inputs that differ only in demographic attributes to see if outputs change inappropriately. This reveals potential bias where demographic factors inappropriately influence decisions.
5. Consider testing your system on intersectional groups that combine multiple demographic characteristics, using the same metrics you applied to single-group analysis. Compare results across these intersectional groups to identify potential bias that might be hidden when looking at single demographics alone.
6. Consider experimenting with complementary fairness metrics like demographic parity, equal opportunity, and equalized odds to get multiple perspectives on your system's fairness. For example, measure whether different groups receive similar positive prediction rates and whether the system correctly identifies positive cases at similar rates across groups. Multiple metrics reveal different types of bias since systems can appear fair on one measure but not on another.
7. Identify which fairness metrics conflict with each other for your system and make explicit decisions about which to prioritize based on your use case context and stakeholder values established in your risk characterization (RAIBR02). Record your reasoning for these trade-

offs since optimizing for one type of fairness often reduces performance on others. Clear prioritization assists you to make consistent decisions when fairness measures conflict.

Resources

Related documents

- NIST AI Risk Management Framework
- ISO/IEC 42001:2023 A.6.2.4 AI system verification and validation
- Common fairness metrics

Related tools:

- Amazon Bedrock Evaluations
- Amazon SageMaker AI Clarify

RAIRC03-BP03 Measure veracity of outputs

Assess your system's tendency to generate factually accurate information while avoiding the specific types of hallucinations, misinformation, or fabricated content your risk assessment identified as problematic for your use case. Implement automated fact-checking and human expert evaluations. Measure the specific aspects of truthfulness your risk assessment prioritized such as factual accuracy, groundedness to source material, or consistency across interactions.

Level of risk exposed if this best practice is not established: High

Implementation considerations

1. Identify metrics for potential hallucination, omission, and misemphasis harms that you identified in your risk assessment (RAIBR02).
2. Plan expert human evaluations where domain specialists review sample outputs for factual accuracy and appropriateness within their area of expertise. Have subject matter experts evaluate claims in their field to catch subtle inaccuracies that automated tools might miss. Human experts can assess context, nuance, and domain-specific accuracy that automated systems often overlook.
3. Measure groundedness, i.e. the degree to which your system's outputs can be traced back to reliable source material when sources are available. Check if claims in generated content align

with the source documents and whether citations are accurate and relevant. Groundedness testing blocks your system from making claims that aren't supported by its reference materials.

4. Measure consistency by asking your system the same questions multiple times and across different phrasings to see if answers remain factually consistent. Also test related questions to see if responses contradict each other across different interactions. Consistency testing reveals when your system generates conflicting information about the same topics.

Resources

Related documents

- NIST AI Risk Management Framework
- ISO/IEC 42001:2023 A.6.2.4 AI system verification and validation

Related tools:

- Amazon Bedrock Evaluations
- Improve accuracy by adding Automated Reasoning checks in Amazon Bedrock Guardrails

RAIRC03-BP04 Measure robustness of outputs to input variation

Measure how consistently your system performs when faced with the specific input variations and distribution shifts that are relevant to your use case. Prepare to test performance across the natural variations your risk assessment determined users might provide (such as different writing styles, dialects, image qualities, or audio conditions relevant to your use case).

Level of risk exposed if this best practice is not established: High

Implementation considerations

1. Build controlled robustness tests that vary one input factor at a time while keeping the content meaning the same, using the input variations your RAIBR02 risk assessment found most likely in your deployment environment. Create paired test cases where you change only one thing, such as converting formal business language to casual speech or adjusting image lighting conditions. Controlled variation testing shows you which specific input factors cause performance drops and by how much.

2. Apply the same metrics you selected in RAIRC02-BP01 to measure performance across different input variations, comparing how your system performs on standard inputs versus challenging variations. Use controlled comparisons where you test the same content with only one input characteristic changed at a time, such as measuring accuracy on both formal and casual versions of the same question. This approach reveals which specific input factors cause performance drops and by how much.
3. Calculate performance variance and degradation across known input variations to quantify how much your system's reliability fluctuates under different conditions. Identify the worst-case performance drops across input types.
4. Test combinations of multiple input variations together, such as processing accented speech with background noise or analyzing low-quality images with poor lighting, since real users often provide challenging inputs with several issues simultaneously. Focus on combinations most likely to occur in your deployment environment based on your use case analysis. Combined variation testing catches failure modes that only emerge when multiple challenging factors interact.

Resources

Related documents

- NIST AI Risk Management Framework
- ISO/IEC 42001:2023 A.6.2.4 AI system verification and validation

Related tools:

- Amazon Bedrock Evaluations

RAIRC03-BP05 Measure privacy protection

Measure how well your system protects each type of confidential or personal information that your risk assessment identified as at risk. This may include detecting privacy leaks, unauthorized data access patterns, or inappropriate data retention issues your risk assessment determined to be most likely or impactful. Assess private data identification and redaction capabilities for the data types that your risk assessment prioritized and consult with your legal team on the specific privacy regulations relevant to your use case.

Level of risk exposed if this best practice is not established: High

Implementation considerations

1. Build privacy attack tests that target the vulnerabilities your RAIBR02 risk assessment found, using both automated tools such as Promptfoo and manual testing to check for membership inference, data extraction, and prompt injections. Create standard test cases with clear success measures and document your testing methods so you can repeat them across different system versions.
2. Set up automated detection tests that check your system's ability to find and remove the types of confidential and personal information that your risk assessment prioritized. Build testing pipelines that measure how accurately your system detects these data types.

Resources

Related documents:

- NIST AI Risk Management Framework
- NIST Privacy Engineering Program
- ISO/IEC 42001:2023 A.6.2.4 AI system verification and validation
- Remove PII from conversations by using sensitive information filters

Related tools:

- Promptfoo
- Presidio: Data Protection and De-identification SDK

RAIRC03-BP07 Measure user controllability of system behavior

To verify that users can effectively control your AI system when they need to override, adjust, or roll back its behavior, develop quantitative measures that assess how well user controls correlate with intended system outcomes. Test the range and granularity of control effectiveness by measuring whether adjustments produce the expected changes in system behavior. Create metrics that capture both the responsiveness of controls and their precision. Your metrics should measure when controls fail to work as intended or when they produce unexpected side effects.

Level of risk exposed if this best practice is not established: High

Implementation considerations

1. Plan how you'll test user controls before building your system by deciding which control mechanisms matter most for user safety based on your RAIBR02 risk assessment findings. Create simple test scenarios that check if each control works as intended and build basic measurement tools that show whether user inputs can change system behavior. This upfront planning saves time later and assists you to build controls that work when users need them.
2. Design tests that check how well users can fine-tune your system's behavior, from small adjustments to major changes. Test both precise control scenarios where users make small tweaks and broad control scenarios where users need to make big behavioral shifts. Include tests that push controls to their breaking points to determine where the system stops responding to user input, which assists you to fix weak spots.
3. Build ways to measure how fast your system reacts when users try to override, adjust, or roll back its behavior. Track how long controls take to activate and how quickly the system settles into new behavior patterns after users make changes. Fast, reliable control response keeps users in charge of the system.
4. Create tests that catch when controls fail or cause unexpected problems elsewhere in your system. Test what happens when users try controls that should fail gracefully and verify that your system gives clear feedback when controls can't work. Look for cases where adjusting one thing accidentally breaks something else, as surprise effects can undermine user trust.

Resources

Related documents:

- FollowBench
- IFEval
- Prompt Steerability
- Human Agency Scale
- ISO/IEC 42001:2023 A.6.2.4 AI system verification and validation

RAIRC03-BP08 Measure explainability of system behavior

Consider metrics for explainability based on user studies that quantitatively measure stakeholders' ability to understand system outputs, including their comprehension of confidence scores, reasoning paths, and limitations, while also tracking the effectiveness of provided explanations across different user groups and expertise levels. This can include objective metrics (such as task completion rates when acting on AI explanations) and subjective assessments (like user satisfaction scores and trust ratings). Pay particular attention to whether users can accurately identify when to rely on or question the system's outputs.

Level of risk exposed if this best practice is not established: High

Implementation considerations

1. Create baseline measurement approaches that check whether users can correctly interpret what your system is telling them and why. Include the user groups from your RAIBR02 risk assessment who need to understand system outputs, and design simple comprehension tests for confidence scores, reasoning paths, and system limitations.
2. Design objective testing that measures how successfully users complete tasks when they rely on your system's explanations. Build tests that track task completion rates, decision accuracy, and time to completion when users act on AI explanations and when they work without them. Test across different expertise levels to see where your explanations assist users to make better decisions and where they might mislead people.
3. Build subjective assessment tools that capture user satisfaction, trust levels, and confidence in your system's explanations. Create simple rating scales and feedback collection methods that show whether users feel your explanations are helpful, trustworthy, and simple to understand. Track how these subjective measures vary across different user groups so you can spot where your explanations work well and where they fall short.
4. Test whether users can accurately judge when to trust or question your system's outputs by creating scenarios where the system should and shouldn't be trusted. Build measurement approaches that check if users correctly identify high confidence as compared to low confidence situations and whether they appropriately rely on or override system recommendations. This testing assists you to catch cases where users might over- or under-trust your system.

Resources

Related documents:

- Advanced tracing and evaluation of generative AI agents using LangChain and Amazon SageMaker AI AI MLFlow

Related tools:

- Amazon SageMaker AI Clarify
- Amazon CloudWatch
- AWS Step Functions
- AWS EventBridge
- Amazon Simple Notification Service

RAIRC03-BP09 Measure security risks and threats

Consider quantitative measurements of security risks to AI systems, such as measuring adversarial attack success rates. For example, measure the rate of successful prompt injection attempts, prompt injection detection rates, jailbreaking success rate, guardrail bypass rates, and model extraction resistance (measuring how simply model parameters or behavior can be reverse engineered).

Level of risk exposed if this best practice is not established: High

Implementation considerations

1. Track metrics such as the percentage of prompt injections that successfully change your system's behavior, how many jailbreaking attempts bypass your guardrails, and whether attackers can extract sensitive information about your model's architecture or training data.
2. Measure attack detection accuracy. Determine the correct balance between blocking suspected attacks and not blocking legitimate user inputs.
3. Test your defenses with advanced attack combinations like prompt injections embedded within seemingly innocent requests or multi-turn conversations that gradually escalate toward harmful content. See if your security holds up when attackers chain techniques together or adapt their methods based on your system's responses.
4. Include red teaming exercises where security experts attempt to break your system using creative attack methods you might not have considered.

Resources

Related documents

- ISO/IEC 42001:2023 A.6.2.4 AI system verification and validation

Related tools

- Threat Composer
- Metrics in Amazon Cloudwatch
- Amazon Bedrock Guardrails

RAIRC03-BP10 Measure transparency quality

Consider situations where system documentation is insufficient, users do not understand the probabilistic nature of a system output, or where users are unaware of AI system presence. Transparency deficits might conceal or amplify potential harms while evaluating impacts on different stakeholder groups. The goal is finding the right transparency level for your situation by balancing enough openness to build trust and meet requirements without creating new vulnerabilities or unintended consequences.

Level of risk exposed if this best practice is not established: High

Implementation considerations

1. Plan how you'll test transparency effectiveness before building your disclosure features by identifying which stakeholders from your RAIBR02 risk assessment need what level of transparency about AI system presence, capabilities, and limitations. Create simple tests that check whether users understand when they're interacting with AI, grasp the probabilistic nature of outputs, and recognize potential biases. This upfront planning assists you to build transparency features that inform users without overwhelming them.
2. Design tests that measure whether your transparency disclosures assist users to make better decisions or accidentally create new problems. Build tests that track decision quality when users have different levels of system information and measure whether transparency improves outcomes or leads to misinterpretation. Test across different user expertise levels to see where more transparency assists versus where it might create confusion.
3. Build measurement approaches that capture both positive transparency outcomes like increased trust alongside potential negative effects like exposure or security risks. Create simple metrics that track user confidence, stakeholder satisfaction, and compliance-aligned measures while also checking for unintended information leakage or misuse. This balanced approach assists you to spot where transparency creates value and where it might cause harm.

4. Test transparency calibration by creating scenarios where users need to understand system confidence levels, limitations, and appropriate use cases for high-stakes decisions like financial or health recommendations. Build measurement tools that check whether users correctly interpret uncertainty indicators and make appropriately cautious decisions when system confidence is low. This testing catches cases where transparency gaps might lead to harmful over-reliance on uncertain outputs.

Release criteria thresholds

RAIRC04: How will you set your release criteria thresholds?

After identifying key metrics, establish target thresholds and statistical methodologies. Set baseline performance targets based on industry standards and risk tolerance, while considering trade-offs between different types of harms and between harm prevention and benefit delivery.

Best practices

- RAIRC04-BP01 Identify baseline performance targets
- RAIRC04-BP02 Consider trade-offs between release criteria
- RAIRC04-BP03 Set confidence requirements for your quantitative release criteria

RAIRC04-BP01 Identify baseline performance targets

Set specific performance goals for your AI system before you build it. These goals become the pass or fail criteria that determine whether your system is ready to release. Good targets are based on real data, not guesswork, and assist you to make clear decisions about when your system is working well enough to release.

Level of risk exposed if this best practice is not established: High

Implementation considerations

1. Research existing performance benchmarks in your domain by collecting data on how current solutions perform and what users need from your system. Look at industry standards, competitor performance, and user satisfaction data to understand the performance bar for your specific use case.

2. Collect baseline data from existing systems, user studies, or pilot tests that show what performance levels are achievable and what users will accept for each of your metrics. Real baseline data assists you to set targets that are challenging but realistic instead of impossible or too simple.
3. Set specific performance targets for your metrics by deciding what performance levels are acceptable for each measurement. This approach transforms your measurement capabilities into clear pass or fail criteria that guide your development and deployment decisions.
4. Plan how you'll track and update your performance targets as you learn more about your system and users by building feedback loops that capture real-world performance data after deployment. Create processes for adjusting targets when you discover your initial goals were too high, too low, or missed important performance dimensions. Flexible target management assists you to improve your system over time while maintaining the discipline of clear performance goals.

Resources

Related documents:

- ISO/IEC 42001:2023 A.6.2.4 AI system verification and validation
- ISO/IEC 42001:2023 A.9.3 Objectives for responsible use of AI system

RAIRC04-BP02 Consider trade-offs between release criteria

Consider trade-offs where meeting your criteria thresholds for one potential harm may reduce your ability to meet the criteria for another harm (for example, privacy as opposed to transparency). Consider harm and benefit trade-offs where meeting the criteria for your potential harms may also reduce your ability to meet the criteria for your benefits. Reconsider your threshold choices to appropriately balance the trade-offs given your use case priorities and document trade-off decisions.

Level of risk exposed if this best practice is not established: High

Implementation considerations

1. Map competing metric relationships and potential conflicts. For example, create a matrix showing how stricter privacy requirements might limit model explainability, or how higher accuracy targets could impact latency performance.

2. In the context of the metric relationships you identified, consider the limits you would set on each competing metric. For example, when user privacy and model accuracy compete, you may opt for privacy requirements even if it means accepting lower accuracy within acceptable bounds.
3. Document threshold decisions and rationale. For example, record final thresholds, identified conflicts, and justification for trade-off decisions in release documentation for future reference and auditing.

Resources

Related documents:

- ISO/IEC 42001:2023 A.6.2.4 AI system verification and validation
- ISO/IEC 42001:2023 A.9.3 Objectives for responsible use of AI system

RAIRC04-BP03 Set confidence requirements for your quantitative release criteria

Decide how certain you need to be that your system meets each performance threshold before each release criterion question can be answered. For example, if you were to divide use cases into higher, moderate, and lower risk, you might set corresponding confidence requirements to 99%, 95%, and 90% respectively. Consider what level of confidence your stakeholders might expect.

Level of risk exposed if this best practice is not established: High

Implementation considerations

1. Group your release criteria by risk level to understand which performance decisions need higher confidence as opposed to those where you can accept more uncertainty. Create simple risk categories like high, medium, and low based on how much harm could result if you're wrong about whether your system meets each performance limit. This grouping assists you to focus your most rigorous testing on the decisions that matter most while avoiding over-testing low-risk areas.
2. Check what confidence levels your key stakeholders expect by talking with users, business leaders, and other groups who depend on your system working correctly. Compare their expectations with your planned confidence levels and adjust where there are mismatches

between what you're planning and what they need. Stakeholder alignment assists you to avoid surprise rejection of your system because your confidence levels don't match their risk tolerance.

3. Set specific confidence levels for each risk category by deciding how certain you need to be before you can confidently say your system meets each performance limit. Assign confidence percentages like 99% for high-risk decisions, 95% for medium-risk, and 90% for lower-risk areas based on what level of uncertainty and risk tolerance your organization and stakeholders can accept.
4. For each release criteria, transform your question from "Does our system produce accurate outputs?" into confidence, threshold, and metric-based questions like "Are we at least 95% confident that our system achieves at least 85% accuracy on our LLM-as-a-judge metric for correctness?" This allows for clear, objective and measurable criteria that leads to binary yes or no responses that account for measurement uncertainty.

Resources

Related documents:

- ISO/IEC 42001:2023 A.6.2.4 AI system verification and validation
- ISO/IEC 42001:2023 A.9.3 Objectives for responsible use of AI system

Dataset planning

Developing an AI system to solve a use case requires developing two logically distinct solution stacks: the *AI evaluation stack* and the *AI system stack*.

The *AI evaluation stack* consists of a suite of evaluation datasets, associated mechanisms to produce the datasets, and the set of release criteria and metrics used to evaluate the system on the data. An evaluation dataset is used to test the performance of the AI system against one or more release criteria. An evaluation dataset may consist of inputs and their expected outputs, or just inputs where a separate mechanism (for example, a human workforce or another AI model) is used as part of the evaluation stack to evaluate the correctness of an AI system output for the input in the evaluation dataset.

The *AI system stack* consists of the AI system itself, required training or auxiliary datasets, and mechanisms needed to produce them. Training datasets are used for building the AI system and support learning, validation, calibration, and component and parameter selection. Auxiliary datasets provide information required during operation (for example, the policies used in automated ground truth checks or the document libraries used in a retrieval-augmented generation (RAG) system).

You can develop AI evaluation and AI system stacks independently. For example, your team and your downstream stakeholders may have their own AI evaluation stacks. Your downstream stakeholder may build their stack based on your documentation (see focus area User Guidance) and still end up making different tactical decisions about the design of their evaluation datasets. This will yield different evaluations of AI system performance.

Alternatively, you may discover that your ideal robustness evaluation dataset has a large statistical power requirement, so you build it out in increments, which results in your robustness performance dropping as each increment is added, even though your AI system stack remains unchanged.

The dataset planning focus area provides best practices for designing the datasets needed in both stacks. The System Planning focus area covers designing the AI system and the Evaluate and Release focus area covers using the evaluation stack to assess the AI system.

Datasets are often thought of as a fixed set of data objects (inputs or input and output pairs, in the case of training or evaluation datasets) that reside in permanent storage. However, the datasets used in training, evaluation, and operation are often dynamically constructed from the static stored version. For example, given a use case of recognizing cats and dogs in images and a static dataset of images labeled cat or not cat, a builder might construct an evaluation dataset for robustness by augmenting each image in the static dataset with rotated versions of the image, increasing the sensitivity of the test datasets to a key confounding variation (image rotation) without needing new labels. The best practices in the Dataset Planning focus area are ideally applied to the final datasets, including dynamic augmentation. However, but this may not be feasible as some static datasets may be augmented dynamically online throughout training.

Focus areas

- Identify datasets
- Dataset quality
- Dataset issues
- Dataset access and versioning

Identify datasets

RAIDP01: How will you identify which datasets are needed to evaluate, train, and operate your AI system?

Datasets assist you to develop an AI system that solves your use case and evaluate whether your system has met your release criteria.

Best practices

- RAIDP01-BP01 Identify evaluation datasets needed to measure system performance against release criteria
- RAIDP01-BP02 Identify the datasets needed for training and customizing your system
- RAIDP01-BP03 Identify auxiliary datasets needed to operate your system
- RAIDP01-BP04 Identify potential overlaps between datasets

RAIDP01-BP01 Identify evaluation datasets needed to measure system performance against release criteria

Work backwards from your release criteria to identify the specific evaluation datasets needed to test each one. Validate that each dataset has the right characteristics for its purpose (for example, demographic labels for fairness testing, harmful content examples for safety testing, and sufficient sample sizes for statistical confidence). Track mappings between datasets and criteria so you can verify complete coverage and maintain traceability between your release criteria and testing approach.

Level of risk exposed if this best practice is not established: High

Implementation considerations

1. For each release criterion, develop a dataset design that specifies the required data sources and data labels. Use your analysis of intrinsic and confounding variations to clarify the specifications. For example, safety testing may require harmful content examples and fairness testing may require demographic labels across groups.
2. Calculate required dataset sizes using statistical power analyses based upon the desired confidence level and interval for the criterion. Verify that the subgroup representation and sample sizes are adequate to test your release criteria with the required confidence you have set.
3. Consider whether one dataset can be used for multiple criteria. If so, verify that the statistical power offered by the dataset meets the needs of the most stringent release criterion.
4. Consider whether one criterion requires evaluation using multiple datasets. If your understanding of intrinsic and confounding variations is limited by known or unknown issues, your evaluation may benefit from using several independently sourced datasets.

Resources

Related documents:

- Statistical Power Analysis for the Behavioral Sciences
- Bedrock Model Evaluation
- NIST AI Risk Management Framework
- Datasheets for Datasets methodology
- ISO/IEC 42001:2023 A.4.3 Data Resources

RAIDP01-BP02 Identify the datasets needed for training and customizing your system

Identify and plan datasets needed to train your AI system to meet your release criteria. Determine which dataset types (training, fine-tuning, validation, calibration, and alignment) you need based on your training approach, assess existing data to identify gaps, then acquire or build the missing datasets through external sources, your own collection, crowdsourcing, or synthetic generation. Finally, plan how to combine and allocate your datasets while keeping them separate from evaluation data and maintaining proper representation across user groups.

Level of risk exposed if this best practice is not established: High

Implementation considerations

1. Map release criteria to training data requirements by listing specific capabilities, behaviors, and knowledge areas your system should demonstrate. Identify what types of training examples you need for each criterion, like domain-specific terminology for accuracy or diverse interactions for fairness.
2. Assess existing training data and identify gaps by checking which model capabilities your current datasets support. Look for missing edge cases, underrepresented languages, or insufficient examples for specific behaviors your system needs to learn.
3. Choose between building custom datasets and using existing ones by weighing control against cost for each gap. Custom datasets provide precise control but require more Resources, while existing datasets are faster but may not perfectly match your needs.
4. Plan data combination and allocation across training phases including pre-training, fine-tuning, validation, calibration, and alignment while maintaining complete separation from evaluation datasets. Design systems that block training-evaluation overlap to protect measurement integrity.

Resources

Related documents:

- Generative AI lifecycle
- Responsible AI Best Practices: Promoting Responsible and Trustworthy AI Systems
- AWS Generative AI Best Practices Framework v2
- ISO/IEC 42001:2023 A.4.3 Data Resources

RAIDP01-BP03 Identify auxiliary datasets needed to operate your system

Auxiliary data covers additional data that affects your system behavior beyond the training, validation, and evaluation datasets, such as knowledge bases used at inference time by RAG systems. Identify auxiliary data sources that affect system behavior during operation. Determine whether auxiliary datasets should be identical between evaluation and deployment environments or if differences are acceptable based on your use case requirements.

Level of risk exposed if this best practice is not established: High

Implementation considerations

1. Map auxiliary data sources your system uses during operation, like knowledge bases for retrieval, reference databases for fact checking, or real-time feeds for updates. Look at your system architecture to determine where additional data is pulled in and affects behavior. This assists you to see the complete data picture beyond just training datasets.
2. Find gaps where auxiliary data could fill coverage holes by analyzing what your training and evaluation data is missing. Check for underrepresented groups, missing domain knowledge, or outdated information. For example, if training data lacks recent events, you might need auxiliary news feeds.
3. Source auxiliary data that complements rather than duplicates your existing datasets by exploring databases, APIs, sensor feeds, and knowledge bases. Verify that auxiliary sources bring new perspectives or fill specific gaps instead of repeating patterns you already captured.
4. Plan to run tests on whether auxiliary datasets improve system capabilities using experiments comparing performance with and without the auxiliary data. Build simple tests showing whether auxiliary information assists with edge cases, accuracy, or underrepresented user groups.
5. Plan auxiliary data management by deciding which data should stay identical between testing and deployment versus which can differ. Build processes for updating auxiliary data when it becomes stale and create checks that verify datasets still match operational needs.

Resources

Related documents:

- An introduction to preparing your own dataset for LLM training

- Prepare ML Data with Amazon SageMaker AI Data Wrangler
- What is RAG (Retrieval-Augmented Generation)?
- Build verifiable explainability into financial services workflows with Automated Reasoning checks for Amazon Bedrock Guardrails
- Revisiting the Auxiliary Data in Backdoor Purification
- Learning to Group Auxiliary Datasets for Molecule
- Unanswerability Evaluation for Retrieval Augmented Generation
- Training a Helpful and Harmless Assistant with Reinforcement Learning from Human Feedback
- AI Benchmarks and Datasets for LLM Evaluation
- ISO/IEC 42001:2023 A.4.3 Data Resources

RAIDP01-BP04 Identify potential overlaps between datasets

Check for unintended data overlap between your training, evaluation, and auxiliary datasets. Ideally, evaluation datasets will contain entirely new examples that your system has never encountered during training, as testing on previously seen data can result in overconfidence in your system capabilities due to overfitting or memorization. Verify that you do not include public benchmarks used for evaluation in training data, particularly when using foundation models where training data provenance may be unclear. Document unavoidable overlaps and assess their potential impact on evaluation validity.

Level of risk exposed if this best practice is not established: High

Implementation considerations

1. Define what it means for the content of training and evaluation datasets to be too similar. For example, if you are building a bird classifier, you may not want the evaluation dataset to contain an image of a flock of birds and the training dataset to contain a sub-image from the flock image, even if the sub-image is contrast enhanced.
2. Define what risk there might be, if any, of having auxiliary and evaluation datasets overlap. For example, you may not want a RAG system to be tested using queries that exactly match the text of FAQs in the RAG document library.
3. Using your definitions of similarity, scan for unwanted similarities between your training, evaluation, and auxiliary data, and estimate the degree of overlap between each dataset.

4. If there are overlaps you cannot remove, estimate the impact on release criteria, adjusting release criteria as necessary.
5. Track changes in overlaps as your datasets evolve by setting up automated systems to flag similarities when you add or update data.

Resources

Related documents:

- Duplicate Detection with GenAI
- Prepare ML Data with Amazon SageMaker AI Data Wrangler
- An Analysis of Dataset Overlap on Winograd-Style Tasks
- A Large-scale Comprehensive Dataset and Copy-overlap Aware Evaluation Protocol for Segment-level Video Copy Detection
- Data Augmentation for Conflict and Duplicate Detection in Software Engineering Sentence Pairs
- Towards Scalable Generation of Realistic Test Data for Duplicate Detection
- What is Overfitting?
- ISO/IEC 42001:2023 A.4.3 Data Resources

Dataset quality

RAIDP02: How will you assess dataset quality?

Measure the quality, representativeness, and coverage of your datasets. Does the dataset cover the range of inputs that your system needs to handle? Are there enough examples in the dataset to measure every release criterion with high confidence?

Best practices

- RAIDP02-BP01 Validate the representativeness of datasets for the use case
- RAIDP02-BP02 Set dataset quality requirements based on your release criteria
- RAIDP02-BP03 Validate the quality of human and generated labels and features in your dataset

- RAIDP02-BP04 Validate the quality and reliability of augmented or synthetic datasets

RAIDP02-BP01 Validate the representativeness of datasets for the use case

Consider whether your datasets accurately reflect the real-world conditions where your system will be used. Gather examples that represent your users while filtering out data from contexts that don't match your use case. This is especially relevant for fine-tuning, alignment, and calibration sets, and for evaluation sets since testing on unrepresentative data can make it seem that your system works better (or worse) than it really does. Ask yourself: "Does this dataset reflect how my system will be used and exclude scenarios that are not part of my use case?" Document what you've included and excluded so you know where your results might not be sufficient.

Level of risk exposed if this best practice is not established: High

Implementation considerations

1. Determine who will consistently use your system and how they'll use it by thinking through your real deployment scenario. Consider how users typically interact with systems like yours and what kinds of inputs they'll give you. This assists you to understand the representative data for your actual use case.
2. Check whether your datasets match real user inputs by comparing what's in your datasets against what you've documented about your use case context. Look for gaps where you're missing certain user groups, missing typical interaction styles, or including data from scenarios that don't match how your system could be used.
3. Clean up your datasets by removing examples that don't match your use case and adding examples that fill important gaps. Focus on your fine-tuning, alignment, calibration, and evaluation datasets since these directly affect how your system behaves and how accurately you can measure its performance.
4. Record what you included and excluded from your datasets, so the limitations are known. Keep track of which user groups or scenarios might not be well-covered so you understand your evaluation limitations.

Resources

Related documents:

- Training data labeling using humans with Amazon SageMaker Ground Truth
- Using Amazon Augmented AI for Human Review
- Data lineage in Amazon DataZone
- Dataset Representativeness and Downstream Task Fairness
- NIST Towards a Standard for Identifying and Managing Bias in Artificial Intelligence
- Policy advice and best practices on bias and fairness in AI
- ISO/IEC 42001:2023 A.7.4 Quality of data for AI systems

RAIDP02-BP02 Set dataset quality requirements based on your release criteria

Work backwards from your release criteria to define the quality standards for each dataset, then select metrics and thresholds to measure when your data meets those standards. Think of this as creating data readiness criteria just like your system release criteria. Data quality means different things depending on how you'll use the data and what your release criteria need. For example, it could mean label accuracy, representation across user groups, diversity of examples, or completeness of coverage.

For each dataset, pick specific quality metrics that align with your release criteria and set minimum thresholds that should be met before using that data. Different datasets need different quality bars. For example, evaluation sets require higher quality standards than training sets.

Level of risk exposed if this best practice is not established: High

Implementation considerations

1. Work backwards from each release criterion to determine the quality standards means for your datasets by asking, "What quality level does my data need to reach for me to trust this measurement?" Define what quality means for each use case, whether that's label accuracy for fairness testing, completeness for harm detection, or consistency for robustness evaluation.
2. Pick specific quality metrics that align with how you'll use each dataset by choosing measurements like missing value rates, label agreement scores, noise levels, or coverage percentages. Make sure your metrics connect to your release criteria instead of just measuring generic data health that might not matter for your specific goals.
3. Set minimum quality thresholds that must be met before you can use each dataset by deciding on specific numbers like label accuracy above 95%, missing values below 2%, or representation

coverage across demographic groups. Document these thresholds as clear pass or fail criteria that your team can check against.

4. Set high quality standards for your evaluation data since evaluation quality directly affects your confidence in release decisions and noise in this data could lead to inaccurate test results.
5. Build data readiness checks that validate your quality thresholds before using a dataset by setting up both automated validation for quantitative metrics and manual reviews for qualitative standards. Treat these like deployment gates that block you from using data that doesn't meet your quality criteria.

Resources

Related documents:

- Amazon Responsible AI Best Practices
- Data Quality Assessment Guidelines
- ISO/IEC 42001:2023 A.7.4 Quality of data for AI systems

RAIDP02-BP03 Validate the quality of human and generated labels and features in your dataset

Implement quality control mechanisms for human annotators including training processes, unwanted bias identification, and inter-rater agreement measurements. Assess potential sources of unwanted human bias and establish procedures to minimize their impact on label quality. When using synthetic or model-generated labels, validate their accuracy against human judgment and document known limitations that affect reliability. Track annotator performance over time and implement feedback mechanisms to maintain consistent labeling standards across your datasets.

Level of risk exposed if this best practice is not established: High

Implementation considerations

1. Set up quality control for human annotators by creating training processes that teach consistent labeling, measuring how well different annotators agree with each other, and checking for unwanted biases in their work. Build simple tests using examples with known correct answers to catch annotators who aren't following guidelines or who might be introducing their own biases into the labels.

2. Hunt for sources of human bias in your labeling process by looking at whether certain annotators consistently label some groups differently than others, whether the annotation guidelines accidentally encourage biased decisions, or whether the examples themselves push annotators toward unfair judgments.
3. Check synthetic and model-generated labels against human judgment by reviewing a sample of machine-generated labels to see how often they're wrong or misleading. Test whether your synthetic labels work well for underrepresented groups and edge cases where automated systems may fail, and document the specific limitations you discover.
4. Track how your annotators perform over time by measuring their consistency, accuracy, and agreement with other annotators across different batches of work. Set up alerts that flag when someone's quality drops or when they start showing new bias patterns, so you can provide additional training or feedback before it affects too much data.
5. Build feedback loops that maintain consistent labeling standards by giving annotators regular updates on their performance, sharing examples of good and bad labels, and updating your guidelines when you discover new edge cases or bias sources. Create processes for fixing labels that don't meet your quality standards to block similar problems in future annotation work.

Resources

Related documents:

- Amazon Sagemaker Ground Truth
- Amazon Sagemaker AI Augmented AI
- Amazon Responsible AI Best Practices
- Data Quality Assessment Guidelines
- ISO/IEC 42001:2023 A.7.4 Quality of data for AI systems

RAIDP02-BP04 Validate the quality and reliability of augmented or synthetic datasets

Assess the quality of model-generated labels and synthetic examples against human evaluation standards. Identify potential sources of unwanted bias in synthetic data generation. Validate that synthetic data maintains the statistical properties needed for your specific datasets and doesn't exclude important edge cases. Document the limitations of synthetic approaches and verify that synthetic examples can effectively substitute for real data in representing the phenomena you care about.

Level of risk exposed if this best practice is not established: High

Implementation considerations

1. Test your synthetic data quality against human standards by reviewing samples of your generated examples and labels to see how realistic and accurate they are. Check whether humans can tell the difference between your synthetic data and real data, and measure how often your synthetic examples contain errors or unrealistic patterns that could mislead your model training.
2. Search for bias in your synthetic data generation by checking whether your generation process consistently produces unfair or skewed examples for certain groups. Look at whether your synthetic data overrepresents some demographics while underrepresenting others, and test whether the generation process amplifies existing biases from your source data or introduces new ones.
3. Verify that your synthetic data keeps the statistical properties you need by comparing distributions, correlations, and patterns between your synthetic and real data. Make sure your synthetic examples don't accidentally exclude important edge cases or rare scenarios that your model needs to handle, and check that key relationships in the data are preserved.
4. Test whether synthetic examples can substitute for real data by having domain experts evaluate whether your synthetic examples capture the key phenomena and scenarios you need to represent, or by training discriminator models to predict whether examples are real or synthetic. If your synthetic data is high quality, the model should struggle to tell the difference. Check if your synthetic data covers the same range of situations, edge cases, and user behaviors as your real data, and verify that it includes the specific patterns and relationships that matter for your use case.
5. Document the limitations and failure modes that you discover in your synthetic data so downstream users know where it might be unreliable. Write down what types of examples your synthetic data handles well versus poorly, what biases it contains, and when it should versus shouldn't be used as a substitute for real data.

Resources

Related documents:

- AWS Well-Architected Machine Learning Lens
- Responsible AI Best Practices for Synthetic Data
- NIST AI Risk Management Framework
- Partnership on AI Synthetic Media Framework
- ISO/IEC 42001:2023A.7.4 Quality of data for AI systems

Dataset issues

RAIDP03: How will you identify and mitigate potential responsible AI issues in the datasets?

Consider responsible AI dimensions when designing and assessing datasets. Best practices for dataset creation differ based on the type of release criteria and responsible AI dimension.

Best practices

- RAIDP03-BP01 Address data that may be unsafe or inappropriate for your use case
- RAIDP03-BP02 Minimize unwanted bias in your datasets
- RAIDP03-BP03 Protect the privacy of individuals represented in your datasets
- RAIDP03-BP04 Include both intrinsic and confounding variations in your datasets
- RAIDP03-BP05 Review the correctness of the content of your datasets

RAIDP03-BP01 Address data that may be unsafe or inappropriate for your use case

To perpetuate dataset safety throughout the AI system lifecycle, establish definitions of safe and unsafe content for your use case. Create specific criteria for content exclusion across training, evaluation, and auxiliary datasets, considering both direct harms and contextual inappropriateness. Implement automated and human review filtering processes, with protective measures for reviewers. Document safety definitions and filtering decisions and regularly audit datasets to verify effective removal of unsafe content while maintaining necessary testing scenarios.

Level of risk exposed if this best practice is not established: High

Implementation considerations

1. Define what unsafe content looks like for your specific use case by creating objective definitions that align with your release criteria.
2. Consider implementing filters and other mechanisms to filter out potentially unsafe or inappropriate content. There may be scenarios where human review is appropriate and helpful in identifying problematic content that models might miss or misclassify. Depending on your use case, seek legal guidance about whether and how to build in processes to filter training data for illegal content such as known child sexual abuse material (CSAM) or adopt additional measures to mitigate risks related to CSAM and exploitative content.
3. Implement protection systems for dataset labelers. For example, set content warnings, exposure limits, and support Resources. Create rotation schedules and anonymous reporting channels for reviewer wellbeing.
4. Measure filtering effectiveness regularly. For example, track removal rates of unsafe content while verifying preservation of necessary test scenarios.
5. Document safety decisions you make to create an audit trail of what content gets filtered out and why, so you can explain your choices and improve your process over time.

Resources

Related documents:

- Flag harmful content using Amazon Comprehend toxicity detection
- Trust and safety
- Automate media content filtering with AWS
- Data-Centric Safety and Ethical Measures for Data and AI Governance
- AEGIS2.0: A Diverse AI Safety Dataset and Risks Taxonomy for Alignment of LLM Guardrails
- BEAVERTAILS: Towards Improved Safety Alignment of LLM via a Human-Preference Dataset
- CISA AI Data Security Guidelines - Best Practices for Securing Data Used to Train & Operate AI Systems
- Training curriculum on AI and data protection Fundamentals of Secure AI Systems with Personal Data
- AI Privacy Risks & Mitigations - Large Language Models (LLMs)
- Thorn Generative AI Child Safety Commitments

- ISO/IEC 42001:2023A.7.2 Data for development and enhancement of AI system
- ISO/IEC 42001:2023 A.7.4 Quality of data for AI systems

RAIDP03-BP02 Minimize unwanted bias in your datasets

When assessing the quality of a dataset, determine whether it appropriately represents the demographics of the expected range of system users. Consider datasets that include self-reported demographic labels. Calculate if datasets contain sufficient representation across demographic groups to enable statistically valid fairness assessments or fairness outcomes.

Level of risk exposed if this best practice is not established: High

Implementation considerations

1. Analyze the demographic composition of your datasets to identify which groups may be over- or under-represented for your use case.
2. Consider using self-reported demographic labels. For example, consider using survey responses or user-provided information rather than algorithmic or human predictions of demographic information.
3. Calculate statistical power for each demographic group in your evaluation datasets by working backwards from your release criteria. For instance, determine whether you have enough examples per group to answer each release criteria question with the required statistical confidence.
4. Address representation gaps by collecting additional data from underrepresented groups or using techniques like stratified sampling, where a population is divided into subgroups, or "strata," based on shared characteristics, and then a random sample is taken from each subgroup to verify representation.
5. Validate that your bias mitigation efforts don't introduce new fairness concerns. For example, check if balancing one demographic dimension inadvertently creates imbalances across intersectional groups.

Resources

Related documents:

- Metrics for Dataset Demographic Bias: A Case Study on Facial Expression Recognition

- Responsible AI question bank: A comprehensive tool for AI risk assessment
- A Review of Machine Learning Techniques in Imbalanced Data and Future Trends
- A survey on learning from imbalanced data streams: taxonomy, challenges, empirical study, and reproducible experimental framework
- A Survey on Small Sample Imbalance Problem: Metrics, Feature Analysis, and Solutions
- Amazon SageMaker AI Clarify: Machine Learning Bias Detection and Explainability in the Cloud
- Fairness, model explainability and bias detection with SageMaker AI Clarify
- Data Curation Practices to Minimize Bias in Medical AI.
- DSAP: Analyzing bias through demographic comparison of datasets
- Mitigating Bias in Training Data with Synthetic Data
- A framework to mitigate bias and improve outcomes in the new age of AI
- Balance your data for machine learning with Amazon SageMaker AI Data Wrangler
- How Clarify helps machine learning developers detect unintended bias
- Generate Reports for Bias in Pre-training Data in SageMaker AI Studio
- Get Insights On Data and Data Quality
- Build an enterprise synthetic data strategy using Amazon Bedrock
- ISO/IEC 42001:2023 A.7.2 Data for development and enhancement of AI system
- ISO/IEC 42001:2023 A.7.4 Quality of data for AI systems

RAIDP03-BP03 Protect the privacy of individuals represented in your datasets

Translate the guidance of your legal counsel on what constitutes personal information into technical definitions appropriate to your use case. Implement processes to identify and limit personal information in training, evaluation, and auxiliary datasets, using both automated filtering, data obfuscation, and manual review approaches. Validate the effectiveness of your privacy protection mechanisms against your taxonomy of personal information types. Maintain detailed documentation of privacy protection measures and regularly audit datasets so that personal information removal doesn't compromise your ability to measure important system behaviors.

Level of risk exposed if this best practice is not established: High

Implementation considerations

1. Translate the guidance of your legal counsel into a taxonomy of personal data types. For example, define the string patterns for direct identifiers (like names and addresses), quasi-identifiers (like age and zip code), and other attributes (like health conditions and financial status) relevant to your domain.
2. Implement multi-layered privacy filtering processes combining automated detection, data obfuscation, and manual review. For instance, use regex patterns and named entity recognition to flag potential personal information, and then apply techniques like tokenization, masking, or synthetic data replacement.
3. Create test datasets with deliberately inserted personal information to evaluate privacy criteria while preserving data utility.
4. Balance privacy protection with system and evaluation needs by verifying that your privacy measures don't compromise your system's ability to address your use case or your ability to test release criteria. For instance, verify that anonymization techniques maintain demographic diversity needed for fairness assessments.
5. Document privacy protection decisions and create audit trails of what information gets filtered, obfuscated, or retained.

Resources

Related documents:

- Towards Efficient Privacy-Preserving Machine Learning: A Systematic Review from Protocol, Model, and System Perspectives
- Training curriculum on AI and data protection Fundamentals of Secure AI Systems with Personal Data
- AI Privacy Risks & Mitigations - Large Language Models (LLMs)
- An overview of implementing security and privacy in federated learning
- Understanding Users' Security and Privacy Concerns and Attitudes Towards Conversational AI Platforms
- Clio: Privacy-Preserving Insights into Real-World AI Use
- Privacy Preserving Machine Learning Model Personalization through Federated Personalized Learning
- Privacy-Preserving AI: Techniques & Frameworks

- Data Anonymisation Made Simple - 7 Methods & Best Practices
- A Comprehensive Guide to Differential Privacy: From Theory to User Expectations
- Data protection in AWS Glue DataBrew
- Identifying and handling personally identifiable information (PII)
- Introducing PII data identification and handling using AWS Glue DataBrew
- Machine learning with decentralized training data using federated learning on Amazon SageMaker AI
- ISO/IEC 42001:2023 A.7.2 Data for development and enhancement of AI system
- ISO/IEC 42001:2023 A.7.4 Quality of data for AI systems

RAIDP03-BP04 Include both intrinsic and confounding variations in your datasets

Revisit your release criteria and use case description to confirm that your definitions of intrinsic and confounding input variations (respectively, variations the system should attend to, and variations it should ignore). Include coverage of relevant variations for your use case in your datasets. If you have robustness release criteria, label what type of variation is present in each example in your evaluation set so you can measure how well your system handles different kinds of variations.

Level of risk exposed if this best practice is not established: High

Implementation considerations

1. Update your lists of intrinsic and confounding input variations (respectively, variations the system should attend to and variations it should ignore) based on your release criteria.
2. Determine ways to get examples of intrinsic variations. Consider whether your samples cover the full distribution of values possible (for example, the full range of nose geometries) if designing a system to recognize dogs.
3. Determine ways to get examples of confounding variations. Consider whether your samples cover the full distribution of values possible (for example, the full range of head poses) if designing a system to recognize dogs.
4. Label variation types in your evaluation datasets to enable robustness measurements against your release criteria. For instance, tag each example with metadata indicating whether it contains lighting variations, formatting changes, or background differences.

Resources

Related documents:

- What is Data Augmentation?
- ISO/IEC 42001:2023 A.7.2 Data for development and enhancement of AI system
- ISO/IEC 42001:2023 A.7.4 Quality of data for AI systems

Related videos:

- Augmenting Datasets using Generative AI and Amazon Sagemaker for Autonomous Driving Use Cases on AWS

Related tools:

- Amazon Bedrock
- Data transformation workloads with SageMaker AI Processing
- Transform data with SageMaker AI Data Wrangler

RAIDP03-BP05 Review the correctness of the content of your datasets

Create regular review processes for ground-truth labels and factual content across your datasets. Implement fact-checking procedures using human reviewers or comparison against authoritative sources to identify and correct inaccuracies. Datasets used for veracity evaluation may require high accuracy standards to provide reliable measurements. Document the review process and track accuracy metrics over time, updating datasets when new information becomes available or when errors are discovered.

Level of risk exposed if this best practice is not established: High

Implementation considerations

1. Design your datasets with built-in accuracy validation by enabling multiple sources to confirm factual claims before including them.
2. Create fact-checking workflows that combine domain experts with authoritative source verification during dataset creation. Have subject matter experts review content and flag potential inaccuracies before data gets finalized.

3. Apply stricter standards to datasets that will be used for evaluation, since these provide the ground truth for measuring release criteria. Engage multiple reviewers to validate each claim and achieve high agreement before accepting labels.
4. Schedule periodic reviews of your dataset content to catch errors that may have emerged over time or due to changing information. Plan regular audits where you re-examine your data to verify labels and factual claims are still accurate.
5. Build correction processes for when you discover errors or when new information becomes available that affects your dataset accuracy. Create clear workflows for updating factual content and maintaining dataset integrity over time.

Resources

Related documents:

- Visualize data quality scores and metrics generated by AWS Glue Data Quality
- Build a data quality score card using AWS Glue DataBrew, Amazon Athena, and Quick Suite
- Ground truth generation and review best practices for evaluating generative AI question-answering with FMEval
- Inspect your data labels with a visual, no code tool to create high-quality training datasets with Amazon SageMaker Ground Truth Plus
- ISO/IEC 42001:2023A.7.2 Data for development and enhancement of AI system
- ISO/IEC 42001:2023 A.7.4 Quality of data for AI systems

Related tools:

- Amazon Bedrock Guardrails : Use contextual grounding check to filter hallucinations in responses
- The Effects of Data Quality on Machine Learning Performance on Tabular Data
- A Survey on Data Quality Dimensions and Tools for Machine Learning
- BoundingDocs: a Unified Dataset for Document Question Answering with Spatial Annotations
- CodeUltraFeedback: An LLM-as-a-Judge Dataset for Aligning Large Language Models to Coding Preferences

Dataset access and versioning

RAIDP04: How will you manage dataset access and versioning?

Reduce the risk of data manipulation by checking for data integrity and by providing the minimal access necessary. Implement a taxonomy (for example, classified, confidential, proprietary, and public) to classify data that interacts with your AI system. Align access rights to the classification of each data source.

Best practices

- RAIDP04-BP01 Create a dataset registry
- RAIDP04-BP02 Periodically evaluate and update datasets in the registry
- RAIDP04-BP03 Protect data from being manipulated or accessed for unintended purposes
- RAIDP04-BP04 Establish governance procedures for managing your datasets
- RAIDP04-BP05 Document the characteristics of each dataset using a datasheet

RAIDP04-BP01 Create a dataset registry

Create a registry to track dataset versions, metadata, and usage across training, evaluation, and operational contexts. Store datasets with version control, including local copies of public benchmarks to assist builders with reproducibility as external datasets evolve. Document the provenance, characteristics, and intended use of each dataset version to enable others to understand appropriate usage and limitations. Link dataset versions to specific system training events and evaluation results to maintain traceability between data changes and performance outcomes.

Level of risk exposed if this best practice is not established: High

Implementation considerations

1. Build a centralized registry system that captures essential metadata for each dataset including version numbers, creation dates, source information, and intended use cases. Start with a simple database or structured file system that can track when datasets were created, who created them, and what they're designed to test.

2. Create version control workflows that automatically snapshot datasets whenever changes are made like a version-controlled code repository. Test your versioning system by making small changes to a dataset and verifying you can retrieve both the current and previous versions reliably.
3. Set up local storage for copies of external benchmarks and public datasets you use, rather than pulling from external sources. Test this by comparing results from your local copy against the original source to catch differences that could affect reproducibility.
4. Build linking mechanisms that connect specific dataset versions to the training runs and evaluations that used them. Test this traceability by picking a model performance result and verifying you can trace back to the exact dataset version that produced it.

Resources

Related documents:

- Onboarding data in Amazon SageMaker AI Unified Studio

- Access your existing data and Resources through Amazon SageMaker AI Unified Studio, Part 1: AWSAWS Glue Data Catalog and Amazon Redshift
  - Automate data lineage in Amazon SageMaker AI using AWS Glue Crawlers supported data sources
  - ISO/IEC 42001:2023 A.7.5 Data provenance

Related tools:

- Data discovery and cataloging in AWS Glue
- AWS Glue
- Amazon SageMaker AI Catalog
- Accelerate generative AI development with Amazon SageMaker AI AI and MLflow
- Amazon SageMaker AI Unified Studio

RAIDP04-BP02 Periodically evaluate and update datasets in the registry

Schedule regular review cycles that assess whether existing datasets still meet your evolving requirements and quality standards. Increment version numbers and update associated documentation whenever datasets change, maintaining records of what changed and why. Assess whether dataset updates require corresponding system retraining or evaluation re-runs to maintain validity of previous results. Remove or archive outdated dataset versions while preserving the ability to reproduce historical results when needed for auditing or comparison purposes.

Level of risk exposed if this best practice is not established: Medium

Implementation considerations

1. Schedule review processes that automatically flag datasets for evaluation based on age, usage patterns, or changes in your system requirements.
2. Create change management workflows that require documenting the reason for a dataset modification along with version increments.
3. Compare new dataset versions against established quality metrics to catch degradation over time.
4. Design impact assessment procedures that assist you to decide when dataset changes require retraining your models or re-running evaluations.
5. Set up archival processes that move old dataset versions to long-term storage while keeping enough metadata to recreate historical results if needed.

Resources

Related documents:

- Data Analytics Lens : Best practice 7.2 – Monitor for data quality anomalies
- Generative AI lens: GENOPS02-BP02 Monitor foundation model metrics
- Data quality in Amazon SageMaker AI Unified Studio
- AWS Glue Data Quality
- Detecting data drift using Amazon SageMaker AI
- ISO/IEC 42001:2023 A.7.5 Data provenance

Related tools:

- Amazon SageMaker AI Catalog
- AWS Glue Data Quality

RAIDP04-BP03 Protect data from being manipulated or accessed for unintended purposes

Implement the principle of least privilege, only providing access to relevant data to those who really need it for both automated systems and human users accessing your datasets. Consider scanning datasets for unwanted content, including adversarial prompts, disinformation, malware, or other data poisoning attempts that could affect downstream system behavior. Establish access controls and audit trails that track who accesses datasets and what modifications are made. Use cryptographic verification methods where appropriate to detect unauthorized changes to critical datasets, particularly those used for evaluation or system operation.

Level of risk exposed if this best practice is not established: High

Implementation considerations

1. Build permission systems that align access controls with specific role requirements, assisting to reduce broad access to data.
2. Set up scanning tools that look for unwanted content like adversarial prompts, fake information, or suspicious patterns before a dataset gets used. These scanners should automatically flag potential data poisoning attempts or embedded malware that could affect your models.
3. Create detailed logs that track who looked at which datasets, when they accessed them, and what changes they made to the data. Your audit trail should be detailed enough that you can reconstruct exactly what happened during dataset operations.
4. Use checksums or digital signatures on your most important datasets so you can tell immediately they were changed without permission. This is especially important for evaluation datasets and operational data that your system relies on.
5. Plan out what your team will do when security problems happen, including how to quickly isolate manipulated datasets and figure out which models or evaluations might be affected.

Resources

Related best practice:

- RAISP02-BP02 Privacy: Build privacy-preserving mechanisms into the core AI system
- RAISP03-BP02 Security: Implement security safeguards to block AI-specific threats

Related documents:

- Security control recommendations for protecting data
- Onboarding data in Amazon SageMaker AI Unified Studio
- Data and model quality monitoring with Amazon SageMaker AI Model Monitor
- Detect and filter harmful content by using Amazon Bedrock Guardrails
- Monitor model invocation using CloudWatch Logs and Amazon S3
- ISO/IEC 42001:2023 A.7.3 Acquisition of data
- ISO/IEC 42001:2023 A.7.5 Data provenance

Related videos:

- Data protection strategies for the cloud - AWS Online Tech Talks:
- AWS re:Inforce 2023 - Using AWS data protection services for innovation and automation (DAP305)
- AWS re:Invent 2024 - Achieve seamless and secure data sharing (ANT325)

Related examples:

- Amazon SageMaker AI Lakehouse now supports attribute-based access control

Related tools:

- Amazon SageMaker AI
- AWS Lake Formation
- Amazon S3 Access Grants
- AWS Identity and Access Management

- AWS Key Management Service

RAIDP04-BP04 Establish governance procedures for managing your datasets

Maintain procedures for managing dataset access, retention, and deletion throughout the AI system lifecycle. Implement mechanisms to handle individual data requests, including the ability to remove individual data points when contributors withdraw consent. Document data lineage and retention policies that specify how long different types of data can be stored and used. Create procedures for handling governance-related dataset updates.

Level of risk exposed if this best practice is not established: High

Implementation considerations

1. Create clear retention policies that specify how long different types of data can be kept and when they need to be deleted.
2. Build workflows that let you quickly find and remove specific data points when people request deletion or withdraw their consent. Your system should be able to trace individual data samples across training sets, evaluation datasets, and cached model outputs without disrupting other parts of your data.
3. Document the complete journey of your data from collection to deletion, including who accessed it, when it was modified, and which models or evaluations used it. This data lineage assists you to understand the impact when you need to remove or modify datasets for compliance-aligned reasons.
4. Consider governance reviews with your legal team where you check that your data handling practices match your policies and legal obligations, including, but not limited to data retention schedules, deletion requests, and access controls.

Resources

Related documents:

- Responsible AI
- Generative AI lifecycle
- Responsible AI Best Practices: Promoting Responsible and Trustworthy AI Systems

- AWS Generative AI Best Practices Framework v2
- ISO/IEC 42001:2023 A.7.5 Data provenance

RAIDP04-BP05 Document the characteristics of each dataset using a datasheet

Create datasheets that document the intended uses, composition, and collection process for each dataset. Include information about data sources, collection methodologies, potential unwanted biases, and recommended and prohibited use cases to assist others to understand appropriate applications. Document the characteristics of data contributors and annotators, including demographic information and potential sources of unwanted bias that could affect system behavior. Maintain datasheets as living documents that are updated when datasets change or when new insights about their characteristics or limitations are discovered.

Level of risk exposed if this best practice is not established: High

Implementation considerations

1. If needed, create standardized datasheet templates that capture essential information about each dataset. Your template should cover basic information such as intended uses, inappropriate uses, data sources, data labels, collection methods, volumes, formats, as well as more nuanced aspects like known limitations and potential biases.
2. Complete the template. As appropriate, capture distributions of sources by label types, and note unexpected distribution skews, gaps in representation, and missing data. Characterize the types of human or machine annotators (for example, experience, training, and potential sources of bias). This assists others understand who's represented in your data and what perspectives shaped the labels or annotations.
3. Set up processes to keep your datasheets current as you learn more about your datasets or make changes to them. Schedule regular reviews to update datasheets when you discover new limitations, modify the data, or find better ways to describe the dataset's characteristics and appropriate uses.

Resources

Related documents

- Datasheets for Datasets

- ISO/IEC 42001:2023 A.7.5 Data provenance

System planning

The system planning focus area provides best practices for designing the AI system stack introduced in the Dataset Planning focus area. The AI system stack consists of training and auxiliary datasets designed in Dataset Planning, and the AI system solving your use case. The AI system itself decomposes into a *core AI system* and an optional set of *filters* (also referred to as *guardrails*) that block or augment data as it flows through, into, or out of the core AI system. The primary design strategies for designing the AI system stack to meet your release criteria are:

- Design choices that directly impact the core AI system (baking)
- Designing data filters (filtering)
- Communicating proper usage to users through documentation, legal terms of usage, classes, or similar channels (guiding)

Refer back to *Dataset planning* for best practices that apply to designing training and auxiliary datasets. The Monitoring focus area addresses guiding. Best practices for ML engineering and for optimizing overall ML performance can be found in the Machine Learning Lens.

Focus areas

- AI system architecture design
- AI system baking
- Filtering
- Choosing a system configuration

AI system architecture design

RAISP01: How will you design the core AI system architecture to meet your release criteria?

Determine the appropriate system type, the required components and their interactions, third-party dependencies, safeguards, and trade-off decisions.

Best practices

- RAISP01-BP01 Detail your core AI system design in a system registry
- RAISP01 BP02 Consider design trade-offs across competing objectives
- RAISP01-BP03 Check if design choices have introduced new risks

RAISP01-BP01 Detail your core AI system design in a system registry

Detail how your AI system works, including the components and the data flows between them. When issues come up, you need to know exactly where problems might creep in, which components could fail, and how problems in one part affect the whole system. Include details about component versions and dependencies so you can track which specific versions might be causing issues and understand how updates could affect your system behavior.

Level of risk exposed if this best practice is not established: High

Implementation considerations

1. Create System Architecture Diagrams: Create system architecture diagrams including preprocessing steps, model architectures, deployment configurations, and integration points. Show end-to-end data flow and component relationships in architecture diagrams.
2. Define Component Specifications and Interactions: Detail each component's functionality, input/output specifications, and dependencies. Document data transformations, model parameters, and performance requirements. Include API specifications, security controls, and integration requirements. Map how components communicate and share data across the system.
3. Establish Documentation Management System: Set up a centralized registry for system documentation, including design decisions and component versions. Implement version control for document updates, create clear update and approval process, and establish review cycles. Include both technical details and business context for each component.

Resources

Related documents:

- ISO/IEC 42001:2023 A.6.2.3 Documentation of AI system design and development

RAISP01 BP02 Consider design trade-offs across competing objectives

Analyze how meeting one release requirement could impact others and establish clear protocols for managing these situations. Your release criteria will sometimes pull your system design in different directions.

For example, making your system more transparent might affect its accuracy, or adding stronger privacy protections could make it harder to explain how decisions are made. Focus on meeting the minimum requirements for each criteria rather than excelling at some while falling short on others.

Level of risk exposed if this best practice is not established: Medium

Implementation considerations

1. Identify where your release criteria might work against each other. Understand how meeting one requirement could make it harder to meet another. For example, adding privacy features might lower accuracy, or making decisions more explainable might slow things down.
2. Set clear priority rules that favor meeting minimum requirements for the criteria over excelling at individual ones. Work with your team, legal experts, and business stakeholders to decide approaches like we'll accept 85% accuracy if it assists us achieve 90% privacy protection rather than maximize accuracy.
3. Design your system using a holistic approach that meets release criteria requirements. Consider how your architecture, models, and components work together as a whole to achieve acceptable performance across each area instead of optimizing individual parts separately.
4. Build your system according to these balanced design choices, focusing on hitting your minimum goals across everything such as 85% accuracy, 90% privacy protection, and acceptable response times rather than pushing for peak performance in a single area.

Resources

Related documents:

- ISO/IEC 42001:2023 A.6.2.3 Documentation of AI system design and development

RAISP01-BP03 Check if design choices have introduced new risks

Review how design decisions affect the risk profile, determining whether additional assessment criteria must be incorporated into the testing framework.

For example, choosing to use a third-party component instead of building your own solution might introduce new risk considerations. When you identify new risks or changes in risk likelihood, decide if you need to update your release criteria to properly test for these issues before release.

Level of risk exposed if this best practice is not established: High

Implementation considerations

1. Review the AI system document, design decisions and tradeoffs for each component.
2. Determine if the design decisions result in new risks or updates to already identified risks in terms of severity and likelihood.
3. Update the risk assessment accordingly.
4. Review the release criteria and determine if the release criteria sufficiently covers the identified risks.
5. Update the release criteria accordingly.

Resources

Related documents:

- ISO/IEC 42001:2023 A.6.2.3 Documentation of AI system design and development

AI system baking

RAISP02: How will you design the core AI system to meet the release criteria (baking)?

Baking implements features directly in the core AI system, for example through choices of component model types, objective functions, training datasets, and training strategies.

Best practices

- RAISP02-BP01 Design the core AI system to directly address your release criteria
- RAISP02-BP02 Privacy: Build privacy-preserving mechanisms into the core AI system
- RAISP02-BP03 Mitigate unwanted bias directly in the core AI system design

RAISP02-BP01 Design the core AI system to directly address your release criteria

Build your system with your release criteria in mind from the beginning, choosing components and designing processes that directly support the performance targets you need to hit. Think of your release criteria as the blueprint for your system design where every design decision should move you closer to meeting those specific goals. Set up regular check-ins during development to see how you are tracking against your targets and be ready to adjust your approach if you spot gaps early. Make sure your candidate testing and validation closely mirror the way you will measure success at release time, so there are no surprises when it is time to release. This approach assists you to build exactly what you need to pass your release criteria, rather than creating a system that performs well on general metrics but falls short on the specific measures that matter for your use case.

Level of risk exposed if this best practice is not established: High

Implementation considerations

1. Define alignment goals that support the release criteria.
2. Define the architecture of the AI system to enhance alignment with release criteria (including determining which methods will be used to address criteria in model training, setting bounds on free parameters, putting in place uncertainty estimation and output validation procedures).
3. Develop training protocol with safety checkpoints and consider techniques like fine-tuning and constitutional training.
4. Establish a validation framework, including safety-focused testing, alignment validation, adversarial and stress testing and integration tests that mirror how the success of the AI system will be measured once it is released.

Resources

Related documents:

- Retrieve Anything To Augment Large Language Models
- Constitutional AI: Harmlessness from AI Feedback
- ISO/IEC 42001:2023 Information technology — Artificial intelligence — Management system

Related tools:

- Customize models in Amazon Bedrock with your own data using fine-tuning and continued pre-training

RAISP02-BP02 Privacy: Build privacy-preserving mechanisms into the core AI system

Design your system from the start to protect confidential and personal data. This may include incorporating techniques like data encryption, access controls, data minimization, data obfuscation, and privacy-preserving training methods directly into how your system works, based on your release criteria.

For example, if your release criteria include keeping certain types of user information confidential, you might build in automatic data masking, use techniques that scramble sensitive information while keeping it useful for training, or set up your system to process information without storing sensitive details. The specific privacy mechanisms you choose should align with your release criteria.

Level of risk exposed if this best practice is not established: High

Implementation considerations

1. Implement data protection measures: Establish security protections around sensitive information. First, identify essential data requirements through data minimization analysis. Create a mapping of sensitive fields and implement anonymization. For example, in a healthcare system, converting 'John Doe, diabetic, 123 Main Street' to 'Patient_2384, condition_type_2, region_14' maintains analytical value while protecting individual privacy. Encrypt sensitive data at rest and in transit. Establish role-based access controls with documented access levels for sensitive data.

2. Apply privacy-preserving training techniques: Consider using differential privacy techniques to introduce controlled noise to the training process. For example, when processing customer transaction data, apply calculated variations to individual purchases while maintaining accurate aggregate patterns. Consider using federated learning to enable distributed model training where data remains at source locations.

For example, with federated learning, healthcare institutions can improve diagnostic models by sharing only parameter updates instead of raw patient data. Consider using gradient clipping to block individual training examples from disproportionately influencing model learning, maintaining both privacy and model quality.

Resources

Related documents:

- Differentially Private Fair Learning
- Remove PII from conversations by using sensitive information filters
- ISO/IEC 42001:2023 A.6.1.2 Objectives for responsible development of AI system

Related video:

- Amazon Bedrock Guardrails: Implementing Custom Safeguards for Responsible AI Applications
- AWS re:Inforce 2025 - Privacy-first generative AI: Establishing guardrails for compliance (COM224)

Related tools:

- Amazon Bedrock Guardrails and PII removal

RAISP02-BP03 Mitigate unwanted bias directly in the core AI system design

Consider incorporating fairness mitigations such as sampling and optimization methods during training, alignment and calibration techniques that actively mitigate biased system responses, and post-processing strategies that review and adjust outputs before they reach users. The specific fairness strategies you use should directly support the fairness goals in your release criteria.

Level of risk exposed if this best practice is not established: High

Implementation considerations

1. Use sampling-based methods during training to improve model performance on underrepresented groups. Apply techniques like weighted sampling to give more importance to underrepresented examples, oversampling to create more training instances from minority groups, or stratified sampling to achieve balanced representation. Consider error-based weighted sampling where you identify groups that experience higher error rates on a validation set and sample datapoints from those groups at higher rates during training. These methods assist your model learn better patterns for each group instead of just the majority.
2. Consider using fairness metrics within the model loss function to guide model training to penalize unfair outputs.
3. Consider if demographic features or proxy features factor significantly into the model predictions by analyzing feature attributions for indications of a biased model. Consider using Amazon SageMaker AI Clarify for feature attributions and bias detection.

Resources

Related documents:

- Amazon AI Fairness and Explainability Whitepaper
- Fairness, model explainability and bias detection with SageMaker AI Clarify
- Transform responsible AI from theory into practice
- Automate model retraining with Amazon SageMaker AI Pipelines when drift is detected
- Accenture Enterprise AI – Scaling Machine Learning and Deep Learning Models
- Amazon SageMaker AI AI: Prepare ML Data with Amazon SageMaker AI Data Wrangler
- NIST AI Risk Management Framework
- ISO/IEC 42001:2023 Information technology — Artificial intelligence — Management system

Related tools:

- Amazon SageMaker AI Clarify
- Fairlearn

RAISP02-BP04 Build security protections directly into the core AI system design

Follow "secure by design" and "defense in depth" principles and build security protections into your system from the beginning to protect your system and maintain its intended operation. This means incorporating safeguards like access controls, input validation and sanitization to defend against prompt injections, and robust techniques to mitigate attempts at jailbreaking or bypassing your system's safety guardrails. The specific security measures you choose should directly address the security requirements in your release criteria.

Level of risk exposed if this best practice is not established: High

Implementation considerations

1. Build input validation into your model's core processing by checking inputs for unwanted patterns, prompt injections, and attempts to manipulate system behavior. Create filters that detect and block unwanted patterns such as instruction overrides, data extraction attempts, and jailbreaking prompts before they reach your model.
2. Design access controls directly into your system architecture by requiring authentications for each interaction, limiting what different user types can access, and restricting administrative functions to authorized personnel only. Set up role-based permissions that block unauthorized users from accessing sensitive model capabilities or training data.
3. Build adversarial robustness into your model by training it to resist attempts to manipulate its outputs through crafted inputs. Include adversarial examples in your training data and design your model architecture to be stable when faced with inputs designed to cause harmful or unexpected behavior.

Resources

Related documents:

- Detect and filter harmful content by using Amazon Bedrock Guardrails
- ISO/IEC 42001:2023 A.6.1.2 Objectives for responsible development of AI system

Related tools

- Amazon Bedrock Guardrails

RAISP02-BP05 Embed provenance indicators into core AI system outputs

Address release criteria for transparency by building provenance indicators directly into your AI system. Providing machine readable labels for audio and imagery outputs is one of the approaches.

Level of risk exposed if this best practice is not established: High

Implementation considerations

1. Consider the necessity and utility of embedding machine-readable labels into AI-generated content such as images, audio, and video that clearly identifies the content as AI-generated.
2. Consider whether to create provenance chains that track the details such as name of the AI system provider, name of the AI system, time stamp of synthetic output generation, and unique identifiers so that users can trace how content was created and modified. The level of detail provided should balance verification value against data costs, security impacts, and risks of disclosing proprietary system details.
3. If your system outputs machine readable labels , provide capabilities that let users check that content has originated from your system. For example, Amazon Bedrock provides customers with the capability to check if an image was generated by Amazon Nova Canvas or Amazon Titan Image Generator via a publicly available tool, Content Credentials Verify.

Resources

Related documents:

- Considerations for addressing the core dimensions of responsible AI for Amazon Bedrock applications
- Evaluate models or RAG systems using Amazon Bedrock Evaluations – Now generally available
- Amazon Titan Image Generator and watermark detection API are now available in Amazon Bedrock
- ISO/IEC 42001:2023 A.8.2 System documentation and information for users
- Thorn and All Tech Is Human Forge Generative AI Principles with AI Leaders to Enact Strong Child Safety Commitments

Related videos:

- Amazon Titan Image Generator Demo - Watermark Detection | Amazon Web Services

Related tools:

- Watermark detection for Amazon Titan Image Generator now available in Amazon Bedrock
- Generating images with Amazon Nova Canvas
- Amazon Nova – AWS AI Service Card

RAISP02-BP06 Enable users to customize core AI system behaviors

Design your system so users can adjust how it works to better fit their particular requirements and preferences, while keeping those adjustments within appropriate boundaries for your use case. This means incorporating features like adjustable output styles, user preference settings, or options that let users guide how the system interprets and responds to their requests.

Level of risk exposed if this best practice is not established: High

Implementation considerations

1. When adding guardrails to your system, build adjustable controls that let users determine how strictly content gets filtered. Set up multi-level filtering from low to high for different content types and create user roles where administrators set baseline policies while end users can adjust settings within safe limits. Include feedback systems to track how well these guardrail controls work and improve them over time.
2. Design interfaces for adjusting output content, style and tone. Allow users to adjust inference parameters like Temperature, Top P, and Top K for text generation to balance between creative and focused outputs. These parameters control the output by influencing the token selection process. Temperature determines randomness of token selection, with higher values producing more creative text and lower values resulting in more focused output. Top P (Nucleus Sampling) samples tokens whose cumulative probability sums to a given threshold, dynamically adjusting the option pool. Top K restricts the model's choice to a fixed number of highest-probability tokens. Similarly, provide ways to the user to adjust response length, format options, and output style.
3. Design structured prompting frameworks to enhance user control over AI system behavior and outputs. Create system-level prompt templates that allow administrators to define AI personality, tone, and response boundaries, while enabling end users to customize task-specific

instructions within safe limits. Build prompt libraries with preset configurations for common use cases (for example, professional communication, creative writing, technical analysis) that users can select and modify. Include prompt validation mechanisms to assist user-provided prompts align with safety guidelines while maintaining the desired level of control over AI responses. Design prompt management interfaces that assist users to understand prompt effectiveness through clear feedback and iterative refinement options.

4. Design tracking mechanisms for control adjustments, system responses, user interactions and performance impacts.
5. Create feedback mechanisms that enable users to refine AI behavior over time. This assists to maintain relevance and reliability of the AI system based on user input and preferences.
6. Develop role-based customization options, allowing different levels of AI feature access and customization based on user roles and business requirements.

Resources

Related documents:

- Prompt templates and examples for Amazon Bedrock text models
- Detect and filter harmful content by using Amazon Bedrock Guardrails
- Influence response generation with inference parameters
- ISO/IEC 42001:2023 Information technology — Artificial intelligence — Management system

RAISP02-BP07 Incorporate explainability mechanisms into the core AI system

Adding explainability to your AI system assists to address explainability release criteria by verifying stakeholders can understand and trust how decisions are made for your specific use case. Include confidence scores with predictions to show how certain the model is about its outputs, and for generative AI systems, use techniques such as content attribution, and token probabilities to explain what influenced the generated content. When explanations are critical, use interpretable models like decision trees that are simple to understand and when more complex models are required add explanation tools (like LIME or SHAP) afterward to interpret their decisions.

Level of risk exposed if this best practice is not established: High

Implementation considerations

1. Build confidence scoring into your system's output pipeline so users can see how certain your model is about each prediction. Test different confidence calculation methods to find ones that actually correlate with accuracy, since some approaches give misleading confidence scores that don't assist users to make better decisions.
2. Choose interpretable model architectures, such as decision trees or linear models when stakeholders need to understand exactly how decisions are made. Compare the performance trade-offs between interpretable and complex models for your specific use case to see if the explanation benefits justify accuracy costs.
3. Add explanation tools like LIME or SHAP to complex models that you can't make interpretable but still need to explain. Test these tools with your actual users to make sure the explanations are helpful rather than confusing, since some explanation methods work better for different types of models and use cases.
4. For generative AI systems, build in techniques like chain-of-thought prompting that show the reasoning process, content attribution that traces outputs back to source material, and token probability displays that reveal uncertainty. Test these explanation methods to see which ones assist users to understand and trust the generated content. For example, each response from an Amazon Bedrock agent is accompanied by a trace that details the steps being orchestrated by the agent. The trace assists to follow the agent's reasoning process that leads it to the response it gives at that point in the conversation.
5. Build automatic source attribution into your Retrieval Augmented Generation (RAG) system by linking retrieved information directly to its origin document with specific citations, page numbers, and document identifiers. Display these citations alongside generated content so users can independently verify where information came from.

Create explanation validation processes that check whether your explainability mechanisms are effective in assisting users make better decisions about trusting or acting on your system's outputs. Regularly test explanations with real users to catch when explanation methods become misleading or stop being useful as your system evolves.

Resources

Related documents:

- ISO/IEC 42001:2023 A.8.2 System documentation and information for users

RAISP02-BP08 Consider core AI system designs that improve factual accuracy

Design your system to produce more accurate information by incorporating techniques that distinguish facts from speculation, reduce hallucinations, and acknowledge uncertainty. This means connecting to authoritative knowledge sources through retrieval methods with source attribution (for example, RAG), employing alignment approaches like constitutional training and reinforcement learning from human feedback (RLHF) to block hallucinations, and incorporating automated reasoning capabilities like chain of thought reasoning for self-reflection along with uncertainty and confidence measurements that assist the system to recognize when it is not confident about information.

Level of risk exposed if this best practice is not established: High

Implementation considerations

1. Design and implement knowledge grounding strategy using RAG architecture or parametric knowledge, verifying clear version control of knowledge bases and rigorous source validation process.
2. Create training objectives that explicitly reward factual accuracy and penalize hallucinations, incorporating self-critique methods and negative examples while using tools like Constitutional AI or RLHF.
3. Build uncertainty quantification into model behavior through calibrated confidence scores and explicit knowledge boundaries, training the system to acknowledge limitations rather than generate plausible but unverified responses.
4. Establish continuous feedback loops to identify and correct factual errors, implementing regular validation cycles against authoritative sources and domain-specific accuracy metrics.
5. Use output validation to check that your system's responses are accurate and relevant. This is used to detect when your system makes up information by comparing responses against trusted sources and using logical checks to verify facts are correct. For example, Amazon Bedrock Guardrails provide capabilities for detecting hallucinations in model responses using contextual grounding checks. Automated Reasoning checks in Amazon Bedrock Guardrails assists to block factual errors from hallucinations using logically accurate and verifiable reasoning that explains why responses are correct. Automated Reasoning assists to mitigate hallucinations using sound mathematical techniques to validate/correct, and logically explain the information generated leading to outputs that align with known facts and are not based on fabricated or inconsistent data.

Resources

Related documents:

- Minimize AI hallucinations and deliver up to 99% verification accuracy with Automated Reasoning checks: Now available
- Build responsible AI applications with Amazon Bedrock Guardrails
- ISO/IEC 42001:2023 A.6.1.2 Objectives for responsible development of AI system

Related tools:

- Amazon Bedrock Guardrails

RAISP02-BP09 Design your core AI system to handle input variations

Design your system to be more resilient by building in the ability to handle input variations and edge cases that could cause it to fail or behave unpredictably. This means incorporating techniques like data augmentation that creates variations of your training examples, adversarial training that tests your system against challenging inputs, and exposure to diverse input formats, styles, and edge cases during the development process. The robustness techniques you choose should directly support your release criteria, assisting your system to perform consistently even when users interact with it in unexpected ways.

Level of risk exposed if this best practice is not established: High

Implementation considerations

1. Review your release criteria to identify expected input variations and how your data might change in real use, from variations like lighting changes in images to text typos or paraphrasing.
2. Create a data augmentation pipeline (automated system to modify training data) that generates different versions of your inputs. Use both simple transformations like rotations or text swaps, and advanced generative methods to create diverse examples.
3. Include robustness techniques during training by adding controlled noise (small random changes) to your data and using optimization objectives that assist your model to learn to ignore minor input differences.
4. Design for continuous monitoring and updating to adapt the system to new data, evolving environments, and unforeseen issues, verifying its continued robustness.

5. Refer to the Dataset Planning focus area for details on data related best practices for designing robust AI system.

Resources

Related documents:

- Clarify Semantic Robustness Evaluation
- ISO/IEC 42001:2023 A.6.1.2 Objectives for responsible development of AI system

RAISP02-BP10 Build safety protections into the core AI system

Follow the safety-by-design principle and design your system from the start to block harmful outputs and unsafe behaviors through multiple layers of protection. Start by creating clear, objective definitions of what constitutes safe versus unsafe behavior for your specific use case, then incorporate safety training approaches like model alignment techniques, constitutional training, and reinforcement learning from human feedback (RLHF) that teach your system to recognize and avoid harmful content while aligning with human values and safety preferences, input sanitization techniques that clean or modify problematic user requests before processing, output alteration methods that modify or block unsafe responses before they reach users, and guardrails that enforce safe interaction boundaries throughout the system.

For example, if your release criteria include safety standards for blocking harmful content, you might implement alignment methods to align your model behavior with your safety criteria, use training approaches that incorporate human feedback to reduce toxic output generation, build input filtering that neutralizes harmful requests, use output modification techniques that sanitize responses, or create interaction limits that block unsafe usage patterns. The safety techniques you choose should directly support your release criteria, creating multiple protective layers that work together to meet safety requirements in your release criteria.

Level of risk exposed if this best practice is not established: High

Implementation considerations

1. Define specific safety boundaries for your use case by creating clear examples of safe versus unsafe outputs that match your release criteria. Test these definitions with stakeholders to make sure your team agrees on what constitutes harmful behavior, then use these examples to guide your other safety work.

2. Build safety training into your model development process using techniques like constitutional AI training that teaches your system to follow safety principles, or RLHF approaches that incorporate human feedback about harmful content. Compare different safety training methods to see which ones work best for addressing the specific release criteria for your use case.
3. Create input sanitization filters that identify and modify problematic user requests before they reach your model. Build these filters to catch different types of harmful inputs like requests for dangerous information, attempts to bypass safety measures, or prompts designed to generate toxic content.
4. Build interaction guardrails that limit how users can interact with your system, like rate limits to block abuse, conversation boundaries that redirect harmful discussions, or session controls that detect and stop unsafe usage patterns. Test your complete safety system with red teaming exercises to find weaknesses and improve your protections before launch.

Resources

Related documents:

- Flag harmful content using Amazon Comprehend toxicity detection
- Thorn and All Tech Is Human Forge Generative AI Principles with AI Leaders to Enact Strong Child Safety Commitments
- ISO/IEC 42001:2023 A.6.1.2 Objectives for responsible development of AI system

Related tools:

- Amazon Bedrock Guardrails

Related videos:

- AWS re:Invent 2024 - Build an AI gateway for Amazon Bedrock with AWS AppSync (FWM310)

Filtering

RAISP03: How will you meet release criteria not fully addressed by the core AI system design (filtering)?

Filtering strategies add protective layers around your AI system through input and output filtering when core system design alone isn't sufficient. Best practices cover implementing privacy-preserving filters to remove sensitive data, security safeguards to block unwanted or adversarial inputs, and safety filters to catch harmful content before it reaches users, among others. This filtering approach is especially valuable for foundation model services where you have limited control over core training but need to meet specific safety and compliance-aligned requirements.

Best practices

- RAISP03-BP01 Add privacy-preserving filters
- RAISP03-BP02 Add security filters
- RAISP03-BP03 Implement output filtering to catch unsafe content before it reaches users
- RAISP03-BP04 Implement output filtering to detect and block hallucinations

RAISP03-BP01 Add privacy-preserving filters

Implement filtering mechanisms that automatically detect and remove unwanted confidential and personal data from both inputs and outputs. Design input sanitization processes that cleanse user queries and system data of unwanted information before processing, using both rule-based and machine learning approaches to identify personal data. Implement output filtering that blocks the generation or disclosure of confidential or personal information, including techniques like anonymization that replace identifying details with generic placeholders or synthetic alternatives.

Level of risk exposed if this best practice is not established: High

Implementation considerations

1. Create a strategy for Identifying unwanted data including personal information and domain-specific information that should not be included in inputs or outputs, then implement appropriate detection methods using tools like Amazon Bedrock Guardrails for filtering inputs and outputs.
2. Design and implement privacy filters for both inputs and outputs, using meaningful placeholders for redacted information and verifying proper encryption for data storage and transmission.
3. Design for unwanted data redaction across data touchpoints including logs and evaluation reports, not just primary inputs and outputs.

Resources

Related video:

- Amazon Bedrock Guardrails: Implementing Custom Safeguards for Responsible AI Applications
- AWS re:Inforce 2025 - Privacy-first generative AI: Establishing guardrails for compliance (COM224)

Related tools:

- Remove PII from conversations by using sensitive information filters

Related documents:

- Differentially Private Fair Learning
- Remove PII from conversations by using sensitive information filters
- Towards Efficient Privacy-Preserving Machine Learning: A Systematic Review from Protocol, Model, and System Perspectives
- Training curriculum on AI and data protection Fundamentals of Secure AI Systems with Personal Data
- AI Privacy Risks & Mitigations - Large Language Models (LLMs)
- An overview of implementing security and privacy in federated learning
- Understanding Users' Security and Privacy Concerns and Attitudes Towards Conversational AI Platforms
- Clio: Privacy-Preserving Insights into Real-World AI Use
- Privacy Preserving Machine Learning Model Personalization through Federated Personalized Learning
- Privacy-Preserving AI: Techniques & Frameworks
- Data Anonymisation Made Simple - 7 Methods & Best Practices
- A Comprehensive Guide to Differential Privacy: From Theory to User Expectations
- ISO/IEC 42001:2023 A.6.1.2 Objectives for responsible development of AI system

RAISP03-BP02 Add security filters

Implement security safeguards that detect and block threats such as prompt injections, roleplay jailbreaks, and other adversarial inputs. Design input validation mechanisms that identify unwanted prompts and suspicious query patterns before they can manipulate system behavior or extract sensitive information. Implement guardrails with content filtering capabilities designed to block the generation of harmful outputs even when inputs successfully bypass input protections.

Level of risk exposed if this best practice is not established: High

Implementation considerations

1. Revisit your security analysis in RAIBR02-BP06 to understand the scope of security issues, including chat interfaces, knowledge bases, file systems and tools, considering the entire system architecture and component interactions beyond just user touchpoints.
2. Design multi-layered security using input validation, content filtering, and rate limiting, utilizing tools like Amazon Bedrock Guardrails for unwanted prompt detection and Amazon Comprehend for input validation.
3. Apply traditional security best practices like least privilege access using AWS IAM roles and policies, while securing infrastructure components and monitoring systems.
4. Design your AI system with a zero-trust model, where no user or device is trusted by default and verification is required for every access attempt.
5. Design for providing the right access level to users and applications invoking AI features and related Resources. For example, provide role-based access control (RBAC) and attribute-based access control (ABAC) to assign permissions based on user roles and context, granting access only to necessary functions and data. Grant users and agents only the minimum permissions necessary to perform their tasks.
6. Design for encrypting sensitive data used in AI systems both at rest (in storage) and in transit to block unauthorized access.
7. Design for sanitizing sensitive inputs coming in and going out of AI system. Validate that system inputs conform to expected formats.

Resources

Related documents:

- Detect and filter harmful content by using Amazon Bedrock Guardrails

- ISO/IEC 42001:2023 A.6.1.2 Objectives for responsible development of AI system

Related tools:

- Amazon Bedrock Guardrails

RAISP03-BP03 Implement output filtering to catch unsafe content before it reaches users

Build screening mechanisms that automatically review and filter your system's outputs to catch potentially harmful content before users see it, acting as a final safety check regardless of what your system generates. This means implementing safety classifiers that can identify toxic, harmful, or inappropriate content in real-time, content filtering systems that block or modify unsafe outputs based on your safety definitions, and automated screening processes that evaluate each response against your safety criteria before delivery.

For example, if your release criteria include safety standards for blocking harmful outputs, you might implement toxicity detection models that score each response, build content filters that automatically block responses containing harmful information, or create screening systems that flag suspicious outputs for human review before they reach users.

Level of risk exposed if this best practice is not established: High

Implementation considerations

1. Based upon release criteria, identify types of potentially harmful content, including adversarial inputs and categories outside of your system's use case
2. Map the types of potentially harmful content to available content filters, such as those already built-in to safeguard services. Additional safety concerns may require custom safeguards, either through system prompting, fine tuning, exact match word configuration, or non-AI solutions that control critical access and permissions to resources.
3. Configure or customize safety filters that can identify and block potentially harmful content across multiple categories such as violence, self-harm, illegal activities, and other context-specific safety concerns.

Resources

Related documents:

- Amazon Bedrock Guardrails enhances generative AI application safety with new capabilities
- Measuring and Mitigating Toxicity in LLMs
- Flag harmful content using Amazon Comprehend toxicity detection
- Thorn and All Tech Is Human Forge Generative AI Principles with AI Leaders to Enact Strong Child Safety Commitments
- ISO/IEC 42001:2023 A.6.1.2 Objectives for responsible development of AI system

Related video:

- AWS re:Invent 2024 - Responsible AI: From theory to practice with AWS (AIM210)
- AWS re:Invent 2024 - Build an AI gateway for Amazon Bedrock with AWS AppSync (FWM310)

Related tools:

- Amazon Bedrock Guardrails

RAISP03-BP04 Implement output filtering to detect and block hallucinations

Build filtering mechanisms that automatically detect and block factually incorrect outputs, hallucinations, and misleading information before they reach users. These filters act as a final check to catch inaccuracies that your core AI system might generate. Use both automated reasoning checks and fact verification systems to validate outputs against known facts and logical consistency before delivery.

Level of risk exposed if this best practice is not established: High

Implementation considerations

1. Identify the specific types of veracity issues your system needs to filter based on your release criteria. Define what counts as hallucinations, factual errors, and misleading information for your use case.

For example, determine whether you need to catch mathematical errors, fabricated citations, invented statistics, or false historical claims.

2. Design your filtering strategy by deciding which verification methods to use and how they will work together. Plan whether you need automated reasoning checks, external fact verification, confidence scoring, or human review processes. Create a filtering architecture that can handle your expected output volume and response time requirements.
3. Implement your filtering mechanisms starting with automated reasoning checks that validate logical consistency, mathematical accuracy, and basic factual relationships. Add fact checking connections to reliable knowledge sources and databases. Build hallucination detection systems that can identify fabricated information patterns your system commonly generates.
4. Test your complete filtering system using known examples of your system's typical errors and hallucinations. Measure how effectively your filters catch different types of inaccuracies without blocking too many accurate outputs. Adjust detection thresholds and add new filtering rules based on what you discover during testing.

Resources

Related tools:

- Improve accuracy by adding Automated Reasoning checks in Amazon Bedrock Guardrails
- Amazon Bedrock: Knowledge Bases
- Amazon Comprehend Features
- Amazon Kendra
- Amazon Bedrock Evaluations

Choosing a system configuration

RAISP04: How will you choose between different system configurations?

Teams test different candidate configurations of their system, including different versions of components or models during development using validation sets to determine which performs best. Different versions can come from different component choices, hyperparameters, training settings, or model architectures. Teams set up controlled comparisons between versions on the same validation data, then use paired statistical tests to determine if one version is statistically better than the other based on release criteria. Evaluation sets stay separate from component selection to avoid biasing final performance measurements and release decisions.

Best practices

- RAISP04-BP01 Use paired tests to choose from candidate designs

RAISP04-BP01 Use paired tests to choose from candidate designs

Test different candidate configurations of your system, including different versions of your components or models during development using validation sets to determine which performs best. Different versions can come from different component choices, hyperparameters, training settings, or model architectures. Set up controlled comparisons between versions on the same validation data, then use paired statistical tests to determine if one version is statistically better than the other based on your release criteria. Keep your evaluation sets separate from component selection because using them would bias your final performance measurements and make your release decisions unreliable.

Level of risk exposed if this best practice is not established: Medium

Implementation considerations

1. Use validation datasets exclusively for choosing between candidate versions, keeping them separate from your final evaluation data. This separation blocks you from accidentally tuning your choices to the test set, which may make your final performance estimates unreliable.
2. Run head-to-head comparisons where each candidate version processes identical validation inputs under the same conditions. Measure their performance on metrics that matter for your release criteria so you can see which version delivers better results.
3. Apply paired statistical tests to determine whether performance differences between candidates are real improvements or just random noise. Calculate confidence intervals and effect sizes to understand not just whether one version is better, but by how much.

Resources

Related documents

- ISO/IEC 42001:2023 A.6.2.4 AI system verification and validation

Evaluate and release

The evaluate and release focus area provides best practices for making a responsible release decision, given the release criteria from *Release criteria*, the evaluation datasets from *Dataset planning*, and the AI system design from *System planning*. The release decision is a binary (yes or no) decision that aggregates the results of testing the AI system against each individual release criterion. A well-designed decision factors in the compounded uncertainty in the individual criterion decisions, that is have we met each criterion with the required confidence (for example, are we at least 95% confident that the release criteria have been satisfied?). If a specific release criterion is not met, release may still be possible if the residual risk is disclosed through Monitoring transparency mechanisms. Otherwise, you should return to *System planning* to review options for baking and filtering mitigations.

Focus areas

- System evaluation
- Aggregate results
- Address unmet release criteria

System evaluation

RAIER01: How will you evaluate the AI system against your release criteria?

Test system performance against the release criteria to determine if benefits and risks are addressed. Consider double checking your evaluations with evaluations by experts outside your team. Identify steps to address each criterion that is not met, modifying transparency solutions as appropriate. To reach a release decision, establish a methodology for combining the results for each release criterion.

Best practices

- RAIER01-BP01 Validate that release criteria still align with current industry standards
- RAIER01-BP02 Independently corroborate more critical and subjective evaluations
- RAIER01-BP03 For each system update, re-run the evaluation and update the system registry

RAIER01-BP01 Validate that release criteria still align with current industry standards

At the start of a release evaluation, check that the release criteria and associated evaluation tests are still aligned with the current version of the AI system. Research and confirm that there are no new and relevant benchmarks or expectations that need to be included in the evaluation.

Level of risk exposed if this best practice is not established: Medium

Implementation considerations

1. Compare your current release criteria against the actual system features and capabilities you plan to release, looking for gaps or mismatches. If your system includes capabilities that were not considered when you last updated your criteria, consider adding appropriate evaluation tests to cover these new features. This includes revisiting your risk and benefit assessment if necessary.
2. Stay up to date with new benchmarks, evaluation methods, or industry standards to see if there are new ways to test your system against your release criteria.
3. Consider new guidelines, updated regulations, or emerging compliance-aligned frameworks that might affect what you need to test before release. Consult with your legal team to assess relevant regulatory considerations.
4. Cross-check your evaluation datasets and test cases to make sure they still match the real-world scenarios where your system will be used. If your intended use cases have changed or expanded, you may need to update your evaluation approach to reflect these new applications.

Resources

Related documents

- ISO/IEC 42001:2023 A.6.2.4 AI system verification and validation

Related videos:

- AWS re:Invent 2024 - Responsible generative AI: Evaluation best practices and tools (AIM342)

Related examples:

- awslabs/agent-evaluation
- aws-samples/rag-evaluation

Related tools

- Amazon Bedrock Evaluations
- Amazon SageMaker AI AI
- Amazon SageMaker AI Clarify

RAIER01-BP02 Independently corroborate more critical and subjective evaluations

Consider getting second opinions on release criteria that are highly critical or more subjective. Such opinions can come from internal or external parties. To maximize independence, consider asking the independent party to build or acquire their own evaluation datasets, using the same information about the intended use case(s) of the AI solution that you intend to communicate to downstream users.

Level of risk exposed if this best practice is not established: Medium

Implementation considerations

1. Identify which evaluations are most critical or subjective and would benefit from independent review, such as safety assessments, unwanted bias evaluations, or user experience judgments. Include evaluations where your team is more likely to have blind spots that could affect its assessment.
2. Identify independent evaluation teams with the capability of building or acquiring independent datasets, such as quality assurance teams, product groups, or other research teams within your organization.
3. Run parallel evaluations where both the development team and the independent team assess the same aspects of your system using the same criteria and datasets. This gives you two perspectives on the same issues and assists you to spot areas where evaluations might be influenced by external factors.
4. For high-risk systems or particularly subjective evaluations, consider bringing in independent evaluators who have no stake in your project's success, but who can build their own evaluations

datasets using only the information that you plan to disclose to downstream deployers and users.

5. Compare the results from different evaluation teams and investigate significant disagreements before making release decisions. When independent evaluations contradict internal assessments, dig deeper to understand why before reconsidering your evaluation approach.

Resources

Related documents

- ISO/IEC 42001:2023 A.6.2.4 AI system verification and validation
- Thorn and All Tech Is Human Forge Generative AI Principles with AI Leaders to Enact Strong Child Safety Commitments July 16, 2024

RAIER01-BP03 For each system update, re-run the evaluation and update the system registry

Record evaluation activities in logs that capture test conditions, system configurations, data inputs, raw results, and methodological notes with sufficient detail to make the entire process reproducible. Establish version control for evaluation artifacts to assist builders to trace unique system builds and their corresponding evaluation results.

Level of risk exposed if this best practice is not established: High

Implementation considerations

1. Log your evaluation runs, including information on which datasets you used, what system version you tested, what hardware and software configuration you ran on, and raw and intermediate outputs. Your logs should be detailed enough that someone else could reproduce your exact evaluation months later.
2. Set up version control for your evaluation materials, including test scripts, configuration files, and result outputs.
3. Link your evaluation materials to both your system and your dataset registry so that it is clear which data and system versions led to the evaluation results. This allows you to link each system build and dataset pair to its specific evaluation artifacts.

Resources

Related documents

- ISO/IEC 42001:2023 A.6.2.4 AI system verification and validation
- ISO/IEC 42001:2023 A.7.2 Data for development and enhancement of AI system

Aggregate results

RAIER02: How will you aggregate the release criteria results into a responsible release decision?

Making a high-quality release decision requires integrating quantitative evaluation results into a cohesive assessment, consolidating information from each system aspect into a single source of truth, implementing a structured release decision process with stakeholder roles, and documenting learnings to either standardize effective practices or address failure points with accountability assignments.

Best practices

- RAIER02-BP01 Add statistical confidence to your release decision
- RAIER02-BP02 Summarize critical information and review with appropriate internal stakeholders

RAIER02-BP01 Add statistical confidence to your release decision

Move beyond simple averages and point estimates to understand how confident you can be that your system will meet its release criteria when deployed. Instead of just asking did we hit our target threshold, ask how confident are we that we'll consistently hit this threshold given the uncertainty in our test results? Use appropriate statistical methods to account for the limited data you have and the variation you expect to see in real-world performance. When you have multiple release criteria, adjust your analysis to account for the fact that meeting the criteria simultaneously is harder than meeting each one individually. This approach may provide a clear, data-driven answer to whether you're ready to release, rather than making that decision based on potentially misleading averages that don't account for uncertainty.

Level of risk exposed if this best practice is not established: High

Implementation considerations

1. Choose appropriate statistical methods to make inferences about your target population based on your sample. For example, use a t or normal distribution for continuous metrics. For ordinal metrics (for example, LLM as a Judge), use non-parametric approaches.
2. To calculate the confidence of meeting a minimum threshold, you can use a Cumulative Distribution Function (CDF), while for a maximum threshold, you would use the Survival Function (SF). For ordinal data, non-parametric approaches like bootstrapping can be used to empirically derive these values by repeatedly resampling from your observed data to create a full distribution of a summary statistic, such as the median. From this empirical distribution, you can directly calculate the proportion of outcomes that fall below or above a specific threshold.
3. Adjust confidence thresholds when evaluating multiple criteria together. Apply corrections like Bonferroni to address compounding uncertainty from multiple criteria. Document methodology and provide clear pass/fail decisions.

Resources

Related documents

- ISO/IEC 42001:2023 A.6.2.4 AI system verification and validation

Related tools:

- Python SciPy Stats
- Python Numpy

RAIER02-BP02 Summarize critical information and review with appropriate internal stakeholders

Organize evidence from your use case, risk assessments, release criteria testing, datasets, and system design evidence into a single document/source of truth that contains the information needed to make a release decision. Include verification that appropriate mitigations are in place for risks across relevant responsible AI dimensions. Update the system registry with the go/no-go decision.

Level of risk exposed if this best practice is not established: High

Implementation considerations

1. Pull together your release documentation into one package that includes your use case definition, risk assessment results, how you did on your release criteria tests, dataset quality reports, and system design details. Organize everything into a single source of truth that gives decision-makers the information they need to make an informed choice about releasing your system.
2. Check that you've addressed risks across responsible AI areas including safety, fairness, privacy, security, robustness, veracity, explainability, transparency, controllability, and governance. Document what mitigations you put in place and make sure they tackle the specific risks you identified earlier in your process.
3. Calculate a single readiness score that combines your confidence in meeting the release criteria. Start with your statistical confidence that the quantitative criteria will pass (using methods from PG-SC03-BP03). This gives you one clear number that shows overall system readiness for release.
4. Write an executive summary that hits the highlights including your key findings, whether you passed or failed each release criterion, what risks are still left after your mitigations, and a clear recommendation about whether you should go ahead with the release. Back up your recommendation with reasoning that stakeholders can understand.
5. Set up review meetings with internal teams like your legal experts, technical leads, risk management teams, compliance-aligned teams and business owners. Walk them through your findings and get their input on whether you're ready to release, since they might catch issues you missed or have concerns you have not considered.
6. Write down your final release decision and update your system registry with whether it's a go or no-go, why you made that decision, who signed off on it, and conditions or monitoring requirements you'll need to follow after release.

Resources

Related documents

- Machine Learning Lens for the AWS Well-Architected Framework

Address unmet release criteria

RAIER03: How will you address release criteria that are not met?

Addressing benefits and residual harms requires assigning unmet release criteria to appropriate implementation strategies, documenting limitations and uncertainties that cannot be fully addressed, and conducting stakeholder reviews to reassess whether business objectives still support the release given identified residual risks.

Best practices

- RAIER03-BP01 For each failed release criterion, re-assess the implementation strategy
- RAIER03-BP02 Identify release criteria that cannot be met and narrow your use case

RAIER03-BP01 For each failed release criterion, re-assess the implementation strategy

Re-evaluate the original implementation strategy assigned to each release criteria. Either improve the execution of the implementation strategy or design a new approach based of baking techniques (for example, additional fine-tuning, new training approaches or component choices), blocking techniques (for example, adding additional guardrails or filtering strategies) or a user steering strategy (for example, publishing user guidance).

Level of risk exposed if this best practice is not established: High

Implementation considerations

1. Analyze why your current implementation strategy failed to meet the release criteria by looking at the specific test results, edge cases where it did not work, and patterns in the failures. Understanding the root cause assists you to decide whether you need to alter your existing approach or add completely new implementation techniques.
2. Add new or enhance existing baking solutions by building additional implementations directly into your model through extra training rounds, refined fine-tuning approaches, or different model component choices. These approaches modify the model's core behavior rather than trying to catch problems afterward, which can be more effective for persistent issues that keep appearing.

3. Implement new or strengthen existing filtering techniques by adding more sophisticated content filters, better output classifiers, or additional input validation rules that catch harmful content before it reaches users. You might need to layer multiple blocking approaches or make your existing filters more sensitive to handle the specific failure cases you discovered.
4. Create new or improve existing guiding approaches that assist users to avoid harmful interactions through redesigned interfaces, clearer guidance, better warnings about limitations, or more comprehensive educational content about appropriate use cases. This works particularly well for criteria that depend on how people choose to interact with your system.
5. Test your new or modified implementation approaches against the same evaluation criteria that your original strategy failed on. Document what you added or changed and what you can learn from this experience for future implementations.

Resources

Related documents:

- Build responsible AI applications with Amazon Bedrock Guardrails

RAIER03-BP02 Identify release criteria that cannot be met and narrow your use case

Assess which of your release criteria you cannot meet with your current system design and implementation strategies, no matter how you refine them. When you find gaps that can't be closed through technical solutions alone, consider whether you are trying to solve too broad of a problem with your current approach. Rather than compromising on safety or performance standards, narrow your use case to focus on scenarios where you can meet your release criteria. Go back to your original risk and benefit assessment with this more focused scope, identifying new opportunities and constraints that come with the narrower application. Update your release criteria to reflect this refined use case, verifying they capture the specific harms you need to block and benefits you want to deliver within your new boundaries. This iterative process assists you to build a system that performs appropriately in its intended domain rather than struggling to meet unrealistic expectations or risk releasing with unmet criteria.

Level of risk exposed if this best practice is not established: High

Implementation considerations

1. List out which release criteria your system consistently fails to meet despite multiple implementation attempts and assess whether these failures are fundamental limitations of your current approach rather than problems that more implementations can solve. Look for patterns like specific types of content your system can't handle safely or performance gaps that persist across different model architectures.
2. Map these persistent failures to specific parts of your use case to understand which scenarios are causing the problems.

For example, if your chatbot struggles with medical advice but works well for general conversation, or if your content moderation system fails on certain languages but works fine for English, you can see where to draw new boundaries.

3. Define a narrower use case that avoids the scenarios where you cannot meet your release criteria, focusing on areas where your system can genuinely excel and deliver value. This might mean limiting the types of queries you handle, the domains you operate in, or the user populations you serve, but it lets you build something that performs appropriately.
4. Redo your risk and benefit analysis using this more focused scope, since narrowing your use case may change both the potential harms and the benefits you can deliver. You might discover new risks in your focused area that you had not considered or find that some broad risks no longer apply.
5. Rewrite your release criteria to match your refined use case, making sure they capture the specific standards that matter for your new boundaries. Your updated criteria may be achievable with your current system design while still maintaining the quality standards that protect users and deliver real value.

User guidance

The user guidance focus area provides best practices for implementing the guiding strategy described in the System Planning focus area. The guiding strategy is necessary because an AI system's effectiveness is evaluated against specific datasets that exhibit diverse technical properties, such as accuracy, fairness, and robustness which inherently involve trade-offs that block simultaneous optimization across each dimension.

Focus areas

Guiding

RAIGT01: How will downstream stakeholders learn how to responsibly use the AI system (guiding)?

Provide downstream stakeholders with tailored documentation that outlines potential benefits and risks of an AI system by discussing its intended uses, limitations, and failure modes, distributed via appropriate channels for each stakeholder group.

Best practices

RAIGT01-BP01 Develop a transparency strategy

For each identified stakeholder group, choose a transparency strategy (for example, blog post, user guide, FAQs, system documentation, service card) and identify appropriate distribution channels to provide downstream stakeholders information to make informed decisions about the AI system.

Level of risk exposed if this best practice is not established: High

Implementation considerations

1. Build a transparency format selection process that matches each stakeholder group's needs and technical capabilities with appropriate communication methods like secure portals, public documentation, or interactive guides. Test different formats with sample stakeholders to see which ones are effective in assisting them to make better decisions about using your AI system.
2. Identify how to reach each stakeholder group effectively, whether through existing professional networks, reporting systems, or direct user interfaces. Set up pilot channels to test delivery effectiveness before rolling out to stakeholders.
3. Create content development workflows that produce stakeholder-specific information covering system capabilities, limitations, risks, and decision-making guidance tailored to each group's expertise level. Build templates and review processes to keep information consistent while allowing customization for different audiences.
4. Build automated update systems that track when your AI system changes and trigger content revisions across different distribution channels to keep stakeholder information current. Set up monitoring to catch when outdated information is still circulating and create processes to quickly correct or redirect stakeholders to updated materials.
5. Engage your organization's leadership to determine public disclosure requirements by identifying information appropriate for public sharing versus proprietary details restricted to internal audiences. Implement a structured publication review and approval process for AI system documentation. This mechanism safeguards sensitive AI design and performance information while maintaining appropriate transparency to different stakeholder groups.

Resources

Related documents:

- ISO/IEC 42001:2023 A.8.2 System documentation and information for users

RAIGT01-BP02 Create a system card that communicates intended usage and limitations

AI system cards are a form of responsible AI documentation that provide stakeholders with a single place to find information on the intended use cases and limitations, responsible AI design choices, and deployment and performance optimization best practices. System cards do not provide guidance on expected performance of the AI system on the specific inputs the deployer may provide; that testing is the responsibility of the deployer.

Level of risk exposed if this best practice is not established: High

Implementation considerations

1. Identify intended use case(s) to illustrate how users should plan to interact with your system. The use case section gives the reader a tangible example, describing the steps and workflow required end-to-end while calling out limitations in the technology.
2. Plan a specific set of evaluations for the AI service card. As appropriate, disclose the datasets chosen for the evaluations and how they meet the criteria to support the testing of each Responsible AI dimension. For example, datasets should have appropriate demographic labels for fairness testing, a representative sample of examples from known safety categories, and common as well as uncommon variations in the input examples for robustness testing.
3. Include performance metrics and success criteria for each use case, with real-world examples demonstrating proper implementation.
4. Detail system limitations and constraints. Consider financial risk assessment AI where specific market conditions or transaction types might fall outside system capabilities. Document scenarios where system performance may degrade or become unreliable, including environmental factors affecting behavior.
5. Outline potential failure modes and implementation strategies when appropriate. As an example, describe how a recommendation system might fail during high-traffic periods or with novel user patterns, and provide recommended responses. Include warning signs and blocking strategies for each failure mode.

Resources

Related documents:

- Introducing AWS AI Service Cards: A new resource to enhance transparency and advance responsible AI
- Resources that promote AI transparency
- Amazon SageMaker AI model cards
- Model cards for model reporting

- Model Registration Deployment with Model Registry
- Transform responsible AI from theory into practice
- Securing generative AI: data, compliance, and privacy considerations
- Thorn and All Tech Is Human Forge Generative AI Principles with AI Leaders to Enact Strong Child Safety Commitments
- ISO/IEC 42001:2023 A.8.2 System documentation and information for users
- ISO/IEC 42001:2023 A.8.3 External Reporting
- ISO/IEC 42001:2023 A.8.5 Information for interested parties

Related tools:

- Amazon SageMaker AI Model Cards
- Amazon SageMaker AI AI

RAIGT01-BP03 Create a plan for publishing and updating documentation

Identify which documents require updates based on stakeholder feedback, new use-cases, new system releases, and industry best practice developments. Dedicate an owner to facilitate the change management process which supports plans for review cycles, document and system versioning and approval chains.

Level of risk exposed if this best practice is not established: Medium

Implementation considerations

1. Establish documentation management infrastructure. Define the criteria for mandatory updates and create automated triggers (AWS EventBridge, Amazon SNS) based on system updates and stakeholder feedback. Assign ownership and responsibility for making the updates. Maintain document version history.
2. Establish and follow a review process. Set up an approval chain and create approval workflows. Check the contents for completeness, clarity, and technical accuracy.
3. Publish the updates and make them accessible to the stakeholders. Have a communication plan. Optionally set up an automated system to notify stakeholders of document updates.

Resources

Related documents:

- ISO/IEC 42001:2023 A.8.2 System documentation and information for users

Related tools:

- AWS Step Functions
- AWS EventBridge
- Amazon Simple Notification Service
- Serverless Computing - AWS Lambda
- Cloud Object Storage - Amazon S3

RAIGT01-BP04 Guide users on how to understand system outputs

Provide accessible guidance on how a user should interpret system outputs. Provide guidance on features the user can use to better understand why a particular input might have produced a specific output. This includes features such as confidence scores, feature importance indicators, decision paths, or chains of thought. Tailor the complexity and format of the guidance to match user expertise levels, providing both high-level summaries and detailed technical information as appropriate. Assist users to identify when to trust system outputs, when to seek additional verification, and when to override or ignore system recommendations based on their domain knowledge.

Level of risk exposed if this best practice is not established: Medium

Implementation considerations

1. Provide guidance on assisting users to understand how the decisions were made for predicting certain outcome. This includes showing which factors were most influential (feature importance) and how certain the system is about its decision (confidence scores). For traditional ML models, Amazon SageMaker AI Canvas provides visual explanations showing these key factors and confidence levels. For example, a loan approval system would display the main factors affecting the decision with their relative importance, how confident the system is in its prediction (expressed as a percentage), historical patterns that influenced the decision and data quality indicators supporting the prediction.

2. For generative AI systems and agents, consider looking at chain of thought techniques that provide the step-by-step thinking process, use observability features like from Amazon Bedrock Cloud Watch integrations to understand traces collected from agents execution that provides visibility to how tools for the agent was selected to be invoked that allowed agents to act. Amazon Bedrock Agents and AgentCore provide detailed traces showing how the system reached its conclusion. For example, a customer service agent would show the sequence of steps taken to resolve a query, which knowledge sources or tools were consulted, why specific approaches were chosen and what alternative options that were considered
3. When possible, provide ways to the users of AI system to understand when to rely on system outputs and when to seek additional verification. Implement monitoring, for example, use AWS AgentCore observability features to trace reliability. For instance, an AI-powered diagnostic system would set clear thresholds for automatic approval versus human review based upon tool invoked and other criteria's, show confidence levels with simple-to-understand indicators, provide specific criteria for when to seek expert verification

Resources

Related documents:

- ISO/IEC 42001:2023 A.8.2 System documentation and information for users

Related tools:

- Amazon SageMaker AI Clarify
- Amazon CloudWatch
- AWS Step Functions
- AWS EventBridge
- Amazon Simple Notification Service
- Observe your agent applications on Amazon Bedrock AgentCore Observability

RAIGT01-BP06 Guide users on how to responsibly change system behavior

Provide guidance that informs users how to effectively alter system behaviors and interpret results. Include user interface elements that guide users toward productive interactions while steering them away from approaches likely to produce poor or harmful results. Explain response mechanisms that provide real-time feedback on input quality and suggest improvements when user inputs are unclear, inappropriate, or likely to produce unsatisfactory results. Direct users to available education resources that assists users to understand system capabilities and limitations, enabling them to leverage the system effectively while maintaining realistic expectations about its performance.

Level of risk exposed if this best practice is not established: Medium

Implementation considerations

1. Create clear guidance materials that explain how users can adjust system settings or modify their inputs to get different types of outputs from your AI system. Test these instructions with real users to see if they can follow them and achieve the results they want without accidentally causing problems.
2. Build educational resources that assist users to understand what your system can and can't do, including interactive tutorials, capability demonstrations, and examples of common mistakes for control modifications. Test these materials with different user groups to make sure people walk away with realistic expectations about system performance and clear ideas about how to use it effectively.
3. Create feedback collection systems that let users report when the guidance isn't working or when they discover better approaches than what you've documented. Use this input to continuously improve your user guidance and update your educational materials based on what real users need to know. For example, feedback may indicate commonly used parameter combinations, or unsafe parameter settings that users may be trying, you can use this information to improve the educational material for these topics to guide users on effective controllability of the AI system.

Resources

Related documents:

- ISO/IEC 42001:2023 A.8.2 System documentation and information for users

Monitoring

The monitoring focus area provides best practices for understanding actual benefits and risks in operation, and for honoring obligations to contributing and downstream stakeholders during and after decommissioning of the AI system.

Focus areas

Monitoring for deviations

RAIMON01: How will you monitor for unexpected deviations from release criteria in operation?

Establish a strategy for monitoring release criteria after release, including baselines for drift monitoring.

Best practices

RAIMON01-BP01 Obtain consent for monitoring production data

As appropriate, implement consent mechanisms that inform users about what data will be collected for monitoring purposes and obtain appropriate permissions before beginning data collection activities. This includes considering opt-in and opt-out data collection strategies while adhering to guidance from your legal counsel. When appropriate, design transparent consent processes that explain monitoring objectives, data usage, retention periods, and user rights regarding their monitored data for opting in or opt out. Establish procedures for managing consent changes over time, including mechanisms for users to withdraw consent and processes for handling data from users who have opted out.

Level of risk exposed if this best practice is not established: High

Implementation considerations

1. As appropriate, create a consent framework defining data collection types and purposes. For example, a music recommendation system might ask for user consent to use system inputs and outputs for validating and improving system performance.
2. Build verification mechanisms to check consent before data collection. For example, an e-commerce system might verify consent status before collecting browsing behavior for personalization.
3. Deploy technical controls to filter data based on consent preferences. For instance, a smart home system might adjust data collection granularity based on user consent levels. Use Amazon S3 for storing data by consent levels.
4. If appropriate and feasible, set up automated processes for consent changes.
5. Maintain audit trails of consent activities. For example, a financial AI system might track consent changes with timestamps in an immutable ledger.

Resources

Related documents:

- AWS CloudTrail
- Amazon S3

RAIMON01-BP02 Set operational performance baselines and apply methods for drift detection

Set performance trend baselines by collecting initial production data over a representative time period to capture your system's actual operating performance, which may vary from your release criteria thresholds. Use statistical methods to characterize normal performance variation patterns, seasonal trends, and expected behavioral ranges for each monitored metric based on observed system behavior. Implement drift detection techniques such as statistical process control charts, change point detection algorithms, and trend analysis that can identify when current performance deviates significantly from established baseline trends, indicating the system is not performing as expected.

Level of risk exposed if this best practice is not established: High

Implementation considerations

1. Establish a baseline using either the training data or a representative validation dataset, defining the expected data distribution and model behavior.
2. Establish data collection to gather relevant metrics during normal operations, capturing representative system behavior including peak/off-peak periods and seasonal variations.
3. Use statistical tests and algorithms to compare live data and monitored metrics against the established baseline. Pre-built rules or custom rules can be configured to define thresholds for acceptable deviations. When a deviation exceeds these thresholds, it may indicate potential data drift, model performance degradation, or bias. Amazon SageMaker AI Model Monitor and SageMaker AI Clarify are examples of services supporting these functions.

Resources

Related tools:

- Data and model quality monitoring with Amazon SageMaker AI Model Monitor
- Fairness, model explainability and bias detection with SageMaker AI Clarify
- Bias drift for models in production

Related documents

- Automated monitoring of your machine learning models with Amazon SageMaker AI AIModel Monitor and sending predictions to human review workflows using Amazon A2I
- Amazon SageMaker AI AI Model Monitor– Fully Managed Automatic Monitoring for Your Machine Learning Models
- AWS re:Invent 2020: Detect machine learning (ML) model drift in production
- ISO/IEC 42001:2023 A.6.2.6 AI system operation and monitoring

RAIMON01-BP03 Preserve data privacy and set access controls on monitored data

Apply data governance processes that specify what monitoring data can be collected, processed, stored, and accessed throughout the monitoring lifecycle. Consider implementing privacy-preserving techniques including anonymization, differential privacy, and secure computation methods that enable system oversight without exposing individual user information. Using the principle of least privilege, create role-based access controls that limit monitoring data access to authorized personnel based on job function, with detailed audit trails tracking data access activities. Establish data retention policies that specify how long different types of monitoring data should be stored, with automated deletion processes and procedures for handling individual data requests.

Level of risk exposed if this best practice is not established: High

Implementation considerations

1. Apply data governance processes that specify what AI monitoring data can be collected, processed, stored, and accessed throughout the monitoring lifecycle. This involves implementing policies that define permissible data collection scope, processing methods, storage requirements, and access protocols for AI model monitoring activities. For instance, a facial recognition AI system allows collection of prediction accuracy metrics and inference latency but prohibits storage of actual facial images or biometric features. Use AWS Config to enforce data governance rules and AWS CloudTrail to audit adherence with data collection policies.
2. Implement privacy-preserving techniques including anonymization, differential privacy, and secure computation methods that enable AI system oversight without exposing individual user information. This requires deploying technical safeguards that protect user privacy while maintaining monitoring capabilities. For example, a healthcare chatbot application could anonymize patient identifiers in conversation logs, apply differential privacy to response accuracy metrics, and encrypt the monitoring data. Use Amazon SageMaker AI Processing jobs to run anonymization and differential privacy implementations, Amazon Macie to identify and protect sensitive data in monitoring datasets, and AWS KMS for encryption and key management.
3. Create role-based access controls that limit AI monitoring data access to authorized personnel based on job function, with detailed audit trails tracking data access activities. This involves implementing granular permissions that restrict monitoring data visibility to specific roles and responsibilities. For example, data scientists access model accuracy metrics while security teams

access only anomaly detection alerts, with access types logged and monitored. Use AWS IAM to implement role-based access controls and AWS CloudTrail to maintain detailed audit trails of monitoring data access.

4. Establish data retention policies that specify how long different types of AI monitoring data should be stored, with automated deletion processes and procedures for handling individual data requests. This requires defining lifecycle management rules for various monitoring data types and implementing automated compliance-aligned processes.

Resources

Related documents:

- Amazon SageMaker AI solution for privacy in natural language processing
- Differentially Private Fair Learning
- Approximate, adapt, anonymize (3A): A framework for privacy preserving training data release for machine learning
- Privacy preserving data selection for bias mitigation in speech models
- ISO/IEC 42001:2023 A.6.2.6 AI system operation and monitoring

Related tools:

- AWS Config
- AWS CloudTrail
- Amazon SageMaker AI Processing
- Amazon Macie
- AWS Key Management Service (KMS)
- AWS Identity and Access Management (IAM)

RAIMON01-BP04 Create monitoring dashboards for operational visibility

Design role-based monitoring dashboards that present relevant system health, performance, and risk indicators tailored to each stakeholder group's responsibilities and expertise levels. Create technical dashboards for engineering teams that show detailed performance metrics, error rates, and component-level health indicators with capabilities for deep-dive analysis. Develop executive dashboards that present summary-level information about benefit realization, risk mitigation effectiveness, and overall system performance against business objectives. Implement governance dashboards for teams that track adherence to release criteria and incident response metrics with historical trending capabilities.

Level of risk exposed if this best practice is not established: High

Implementation considerations

1. Map stakeholder dashboard requirements by role. For example, a healthcare AI system can create separate views for clinical staff showing patient outcomes, technical teams showing model performance, and executives showing system impact. Use QuickSight for dashboards and IAM for access control.
2. Create dashboards for performance metrics and have mechanisms for triggering alarms when threshold is met. For example, you can monitor each part of your Amazon Bedrock application using Amazon CloudWatch, which collects raw data and processes it into readable, near real-time metrics. You can graph the metrics using the CloudWatch console. You can also set alarms to watch for certain thresholds and send notifications or take actions when values exceed those thresholds. Amazon CloudWatch metric may include Bedrock Guardrails metrics like total requests intervened by guardrail for various reasons like denied topics, in appropriate content, sensitive information or context grounding concerns. Controlling CloudWatch metrics visibility by role is accomplished through AWS Identity and Access Management (IAM) policies.
3. When using Amazon SageMaker AI Model Monitor, Amazon SageMaker AI Model Dashboard can be used to track the performance of models as they make real-time predictions on live data. Use a dashboard to find models that violate thresholds you set for data quality, model quality, bias and explainability.

1. Data Quality: Compares live data to training data. If they diverge, your model's inferences may no longer be accurate.
2. Model Quality: Compares the predictions that the model makes with the actual Ground Truth labels that the model attempts to predict.
3. Bias Drift: Compares the distribution of live data to training data, which can also cause inaccurate predictions.
4. Feature Attribution/Explainability Drift: Compare the relative rankings of your features in training data versus live data, which could also be a result of bias drift.

Resources

Related documents

- Data quality
- Model quality
- Bias drift for models in production
- Feature attribution drift for models in production
- Implement safeguards for your application by associating a guardrail with your agent
- Monitor Amazon Bedrock Guardrails using CloudWatch metrics
- Amazon SageMaker AI Model Dashboard
- Automated monitoring of your machine learning models with Amazon SageMaker AI Model Monitor and sending predictions to human review workflows using Amazon A2I
- Amazon SageMaker AI Model Monitor – Fully Managed Automatic Monitoring For Your Machine Learning Models
- ISO/IEC 42001:2023 A.6.2.6 AI system operation and monitoring

Related tools

- Amazon CloudWatch

RAIMON01-BP05 Design protocols that trigger human oversight of automated monitoring alerts

Set protocols for when human reviewers should be involved in system oversight decisions. Create sampling-based human review processes that validate the accuracy and effectiveness of automated monitoring systems, including procedures for evaluating edge cases and challenging scenarios. Implement feedback mechanisms that enable human reviewers to improve automated monitoring through labeling ambiguous cases, refining alert criteria, and identifying new monitoring requirements. Design human oversight workflows that provide escalation paths, decision-making authority, and documentation requirements for monitoring decisions that affect system operation.

Level of risk exposed if this best practice is not established: High

Implementation considerations

1. Configure human review triggers in monitoring systems based on alert severity, confidence thresholds, and business impact. Use workflow orchestration tools like AWS Step Functions to route decisions and Amazon A2I for human review management.
2. Establish sampling protocols to validate monitoring accuracy, focusing on edge cases and high-risk scenarios. Integrate annotation tools for human reviewers to assess and label sampled alerts.
3. Create feedback loops allowing reviewers to label ambiguous cases and suggest monitoring improvements. Use Amazon A2I for feedback collection and AWS Step Functions to route feedback for monitoring system improvements.
4. Design escalation paths with clear authority levels and documentation requirements for critical monitoring decisions. Configure workflow tools to manage approvals and maintain audit trails of human oversight activities.
5. Document human oversight decisions, rationale, and outcomes to support continuous improvement of monitoring protocols. For example, documenting human interventions on monitoring alerts with timestamps, reviewer identity, decision rationale, and subsequent monitoring system behavior changes.

Resources

Related documents

- Amazon Augmented AI
- AWS Systems Manager Incident Manager
- ISO/IEC 42001:2023 A.6.2.6 AI system operation and monitoring

Responding to feedback

RAIMON02: How will you respond to feedback from monitoring?

Improve your AI system by applying feedback from monitoring and reported incidents.

Best practices

- RAIMON02-BP01 Create feedback loops to apply monitoring results to system improvement

RAIMON02-BP01 Create feedback loops to apply monitoring results to system improvement

Translate monitoring results, incident patterns, and performance trends into actionable system improvements and risk mitigation enhancements. Implement regular review cycles that analyze monitoring data across multiple time horizons, identifying both immediate optimization opportunities and longer-term improvement strategies based on usage patterns and performance drift. Update system components based on monitoring insights, including refining guardrails, adjusting model parameters, updating training data, and modifying deployment strategies. Track the effectiveness of monitoring-driven improvements by validating that changes address identified issues without introducing new problems or degrading system performance in other areas.

Level of risk exposed if this best practice is not established: Medium

Implementation considerations

1. Establish regular monitoring review cycles: daily checks for immediate issues, weekly trend analysis, and monthly pattern reviews. Example: Review ML model accuracy daily, analyze feature drift patterns weekly, evaluate system performance trends monthly.
2. Create an improvement action framework to categorize monitoring insights into quick fixes, medium-term adjustments, and long-term enhancements.
3. Build an automated alert-to-action pipeline that connects monitoring alerts to specific improvement workflows. Example: Configure Amazon SageMaker AI Model Monitor to capture incoming data and detect changes in model feature distributions or prediction patterns. Set up Amazon EventBridge to automatically initiate SageMaker AI Pipeline for model retraining when Model Monitor detects data drift beyond defined thresholds.
4. Implement validation checks to measure improvement effectiveness. Example: Compare model metrics pre and post-retraining, monitor downstream impacts, and validate that automated improvements maintain model quality standards.

Resources

Related documents:

- Automate model retraining with Amazon SageMaker AI Pipelines when drift is detected

Decomissioning

RAIMON03: How will you decommission your AI system?

Prior to decommissioning an AI system, consider the impact on upstream and downstream stakeholders.

Best practices

- RAIMON03-BP01 Establish mechanisms for honoring stakeholder obligations

RAIMON03-BP01 Establish mechanisms for honoring stakeholder obligations

Consider how to honor obligations you many have to upstream stakeholders (such as people who contributed content to an evaluation or training dataset) and downstream stakeholders (such as workflows that have taken dependencies on your AI system).

Level of risk exposed if this best practice is not established: High

Implementation considerations

1. Review your dataset registry to decide the correct way to handle each dataset, for example should it be kept for re-use, kept as a required record, or deleted.
2. Review logs and customer agreements to identify potential downstream dependents and determine a decommissioning strategy that provides appropriate notice.

Resources

Related tools:

- Overview of the decommissioning process

Conclusion

The Responsible AI Lens provides guidance on implementing trustworthy AI systems. The eight focus areas of the Lens guide builder teams from initial use case definition through development, deployment, and ongoing operations.

The lens emphasizes narrow use case definitions to manage risks effectively while maximizing benefits. It advocates for clear, statistically backed release criteria to support objective decision-making. Throughout the AI lifecycle, the Lens stresses the importance of structured risk assessment and monitoring processes.

Practical considerations accompany each best practice, providing development teams with actionable guidance. These steps cover establishing responsible AI processes, defining measurable criteria, implementing monitoring systems, creating appropriate documentation, and maintaining feedback loops for continuous improvement.

Recognizing the evolving nature of responsible AI, the lens is adaptable. Teams can customize these practices to their specific contexts while adhering to fundamental principles. This flexibility, combined with a commitment to early adoption and ongoing learning, supports the development of AI systems that are both powerful and trustworthy.

As the field of AI continues to advance, this Lens serves as a foundation for implementing systems that deliver value while maintaining appropriate safeguards. Regular updates will incorporate new insights and emerging standards to maintain relevance in a rapidly changing landscape.

Contributors

The following individuals and organizations contributed to this document:

- Rachna Chadha, Principal Technologist, AI, Field-AGS-WorldwideTech-All-CS, Amazon Web Services
- Peter Hallinan, Director, Responsible AI, AWS Responsible AI, Amazon Web Services
- Mathew Monfort, Senior Applied Scientist, AWS Responsible AI, Amazon Web Services
- Mikaela Myers, Sr. Product Manager Technical, AWS Responsible AI, Amazon Web Services
- Mike Diamond, Principal Product Manager, AWS Responsible AI, Amazon Web Services
- Ramya Pulipaka, Sr. Delivery Consultant, AWS WWCO ProServe, Amazon Web Services
- Denis Batalov, Tech Leader, ML & AI, WWSO Generative AI, Amazon Web Services
- Bharathi Srinivasan, Data Scientist II, SCIENC, WWSO Generative AI, Amazon Web Services
- Sara Liu, Sr. TPM, Responsible AI, AWS Responsible AI, Amazon Web Services
- Yim Register, Applied Scientist II, ML University, Amazon Web Services
- CJ Lee, Senior Applied Scientist, AWS Responsible AI, Amazon Web Services
- Vanessa Murdock, Principal Applied Scientist, AWS Responsible AI, Amazon Web Services
- Rongting Zhang, Senior Applied Scientist, AWS Responsible AI, Amazon Web Services
- Alicia Sagae, Applied Scientist II, AWS Responsible AI, Amazon Web Services
- Ambar Pal, Applied Scientist, AWS Responsible AI, Amazon Web Services
- Pinar Yilmaz, Principal Engineer, AWS Responsible AI, Amazon Web Services
- Neelam Koshiya, Principal Applied AI Architect, Field-AGS-WorldwideTech-All-CS, Amazon Web Services
- Byron Arnao, Principal Technologist, Field-AGS-WorldwideTech-All-CS, Amazon Web Services
- Mia Chang, GTM SSA AIML DE, AWS WWSO GTM Specialists-EMEA, Amazon Web Services
- Ashish Rawat, Sr. GenAI Specialist SA, AGS-WWSO NAMER-All-CS, Amazon Web Services
- Kait Healy, Solutions Architect, Field - AGS - EMEA - SUP - CS, Amazon Web Services
- Divya Muralidharan, Solutions Architect, Field-AWS I-Stra Accounts-AWSI, Amazon Web Services
- Abhi Shivaditya, Principal, Solutions Architect, Field-AWS I-Stra Accounts-AWSI, Amazon Web Services
- Martin Bertran Lopez, Applied Scientist, AWS Responsible AI, Amazon Web Services

- Rui Cardoso, Sr Partner SA, AWS Partner Field EMEA, Amazon Web Services
- Manoj Kale, Sr Solutions Architect, Field-AWS I-Stra Accounts-AWSI, Amazon Web Services
- Cassandre Vandeputte, Solutions Architect - Brussels, AWS WWPS MNO, Amazon Web Services
- Ioan CATANA, Senior AI/ML Specialist SA, AGS-WWSO EMEA-All-CS, Amazon Web Services
- Flora Eggers, Sr. Solutions Architect, Field-AGS-Germany-ENT-CS, Amazon Web Services
- Roland Odorfer, Solutions Architect, Field-AGS-Germany-ENT-CS, Amazon Web Services
- Prateek Prakash, Sr. Delivery Consultant, AWS WWCO ProServe, Amazon Web Services
- Sandeep Avula, Applied Scientist, AWS Responsible AI, Amazon Web Services
- Ram Ravi, Technologist, Field-AGS-WorldwideTech-All-CS, Amazon Web Services
- Debashis Das, Principal, AWS OCISO, AWS Security, Amazon Web Services
- Ryan Dsouza, Principal Solutions Architect, Amazon Web Services
- Mahmoud Matouk, Principal Security Lead SA, Amazon Web Services
- Madhuri Srinivasan, Sr. Technical Writer, Amazon Web Services
- Stewart Matzek, Sr. Technical Writer, Amazon Web Services
- Matthew Wygant, Sr. TPM Guidance, Amazon Web Services

Document revisions

The following table describes the documentation releases for the Responsible AI Lens.

Change	Description	Date
Initial release	Initial release of the Responsible AI Lens.	November 19, 2025

AWS Glossary

For the latest AWS terminology, see the AWS glossary in the *AWS Glossary Reference*.