Keynote - AI Assurance

Dr Keith Joiner & GPCAPT Randall McCutcheon, UNSW Canberra*

Never Stand Still

APAC Entrepreneur

https://apacentrepreneur.com/ten-ways-to-improve-your-people-management-skills/

Forbes, B. Marr (2020)

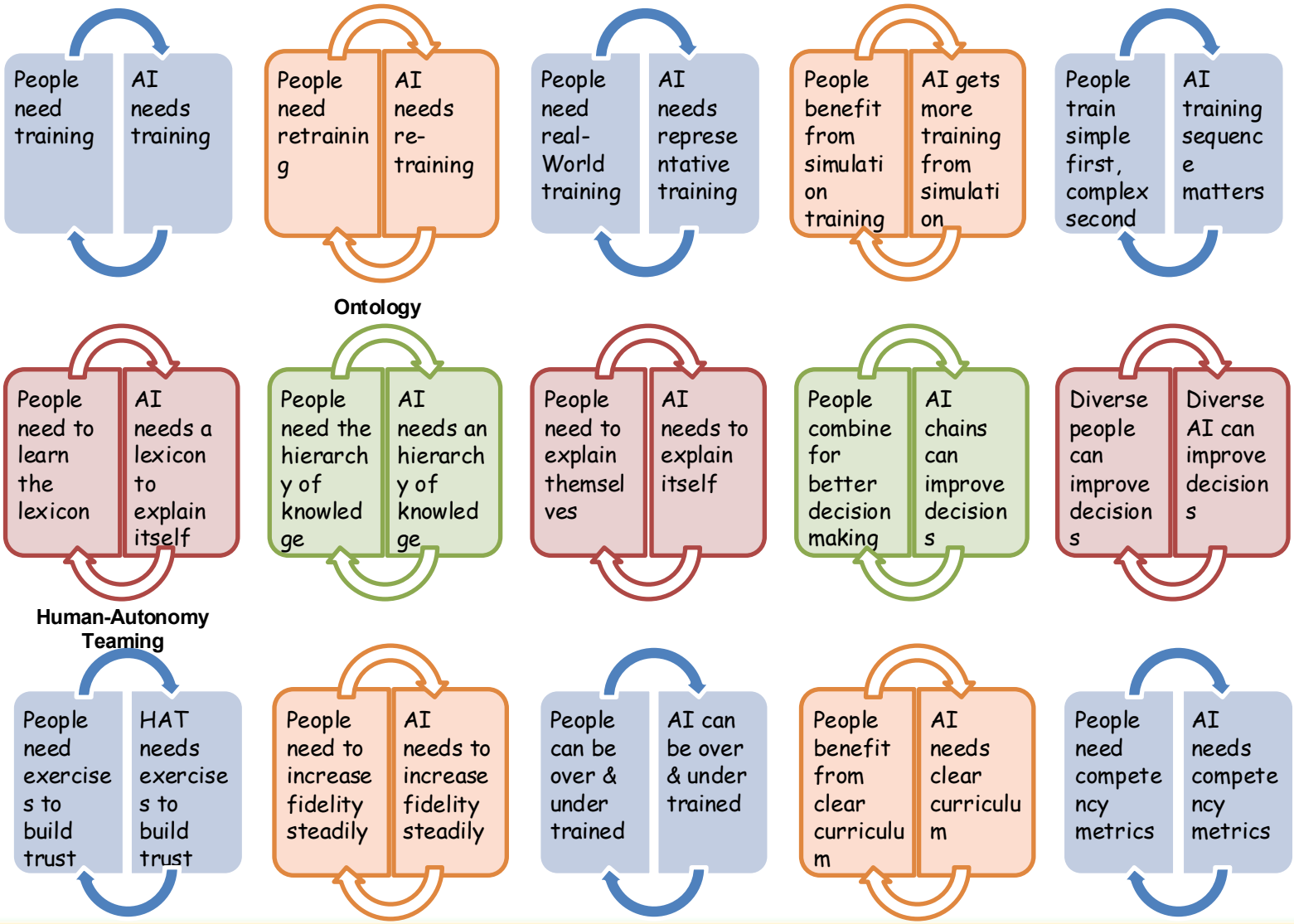
https://www.forbes.com/sites/bernardmarr/2020/08/03/3-important-ways-artificial-intelligence-will-transform-your-business-and-turbocharge-success/?sh=a49eb4c620fa

*https://research.unsw.edu.au/people/dr-keith-francis-joiner

ADFA
AUSTRALIAN DEFENCE FORCE ACADEMY

ProjectCHAT 2024 pitched:

"If you can manage people, you can manage AI"



Todays Keynote overview of our Masterclass:

Introduction - Why AI-Enabled systems need new assurance

What can we take from Human-Human assurance?

What can we take from Software assurance?

What have we found in AI assurance?

What happens when we map Human-Human to Software and AI assurance (RM)?

Conclusions (KJ)

- AI-enabled systems are prevalent.
- No IT developer or engineer today will avoid AI ML use unless directed, as:
  - it usually provides efficiency & superior performance
  - IT developers & new engineers quickly grasp the concepts
- Most risks with AI-enabled systems are readily managed when there are these controls for engineered systems:
  - System safety engineering (i.e., MIL-STD-882)
  - System security engineering (i.e., NIST SP 800-172 cybersecurity RMF)
  - Early validation & involvement of representative users
- Problems with AI-enabled systems are likely when development is:
  - not iterative
  - testing is not integrated (DT & OT)
  - multi-proprietary,
  - multi-generational,
  - multi-security, or
  - IT developers have an isolated approach or ‘grab bag’ of OTS

F. H. Ferreira, E. Y. Nakagawa, and R. P. dos Santos, ‘Reliability in Software-Intensive Systems: Challenges, Solutions, and Future Perspectives’ (paper presented at 47th Euromicro Conference on Software Engineering and Advanced Applications (SEAA), September, 2021).

NIST AI RMF (NIST AI 100-1, Jan 2023) (https://nvlpubs.nist.gov/nistpubs/ai/NIST.AI.100-1.pdf)

'*is a guidance designed to improve the robustness and reliability of AI by providing a systematic approach to managing risks*'

is built on four functions:

Govern - risk-aware organizational culture,

Map - contextualize AI systems within broader operational environment with impacts across technical, social, & ethical dimensions

Measure - risk assessment, promoting both quantitative and qualitative approaches to understand the likelihood and potential consequences

Manage - risk response, through a combination of technical controls and procedural safeguards

Dangers are in generational AI ML ignorance means:

- not 'risk-aware'
- limited social & ethical exploration when representative users are not involved (i.e. poor test scoping & OTS shortcuts)
- Don't understand metrics
- Don't know technical control & safeguard options

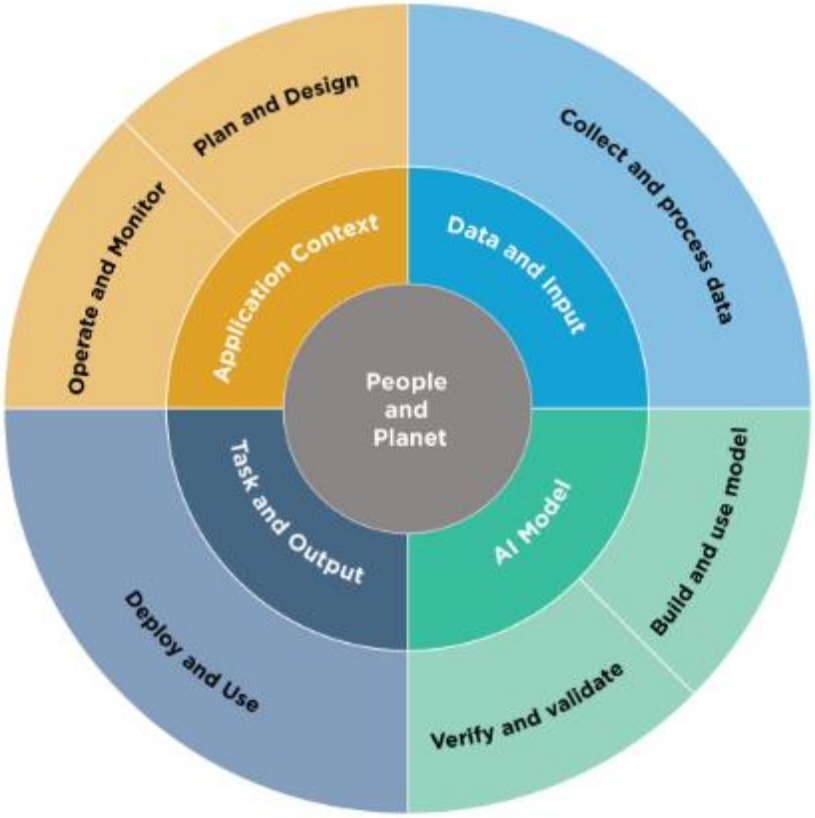
Best Practices from NIST AI RMF

'*include principles like*

- *ensuring data quality* [DQ],
- *fostering transparency in AI decision-making* [T], *and*
- *maintaining human oversight* [HAT].

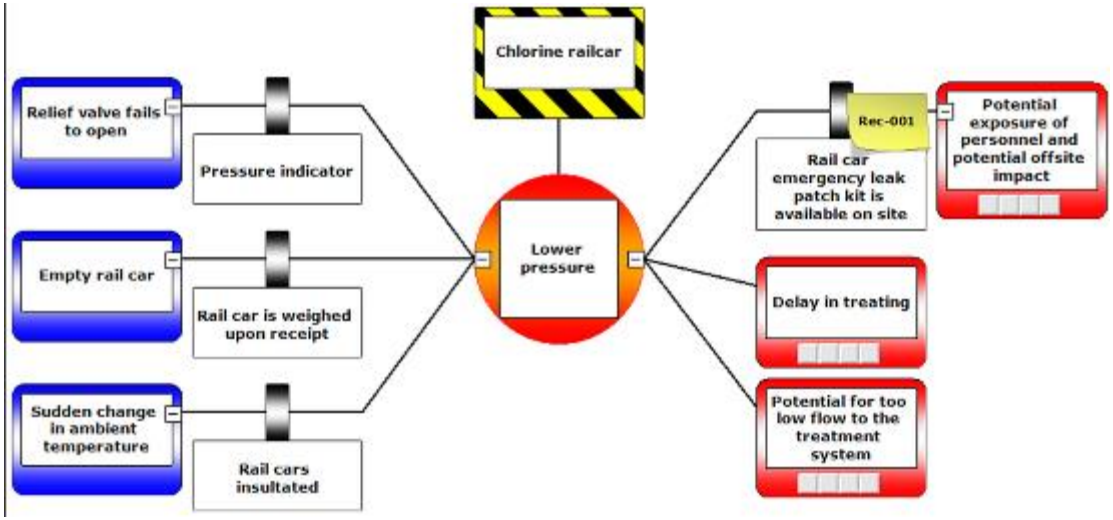
Best practices also

- *advocate for the inclusion of robust security measures,*
- *regular audits for bias and fairness, and*
- *adherence to privacy regulations.*'

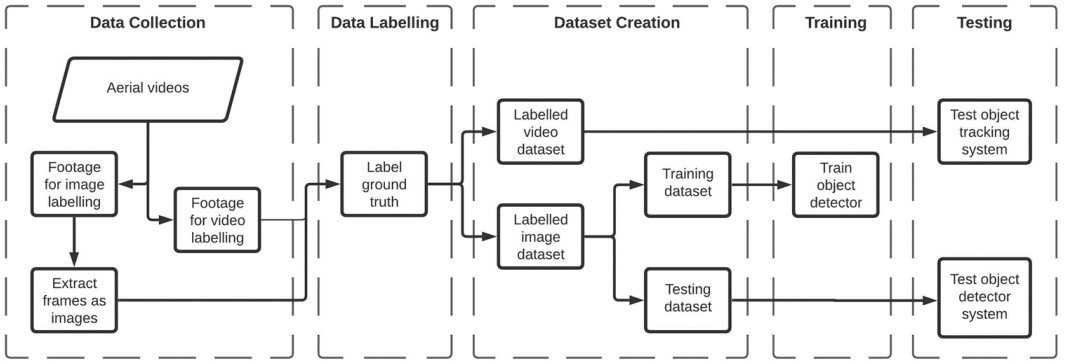


12 Key Assurance needs of AI – arguably all good PM & SE before AI:

1. Clear articulation of the User story in each AI ML instance [HAT]
2. Early Preliminary Hazard Analysis (PHA) to band safety criticality & provide mitigation strategies & options across the human-autonomy growth paths [HAT]



3. A flowchart developmental process [T] clearly showing representative user input opportunities, testing & iteration [HAT]

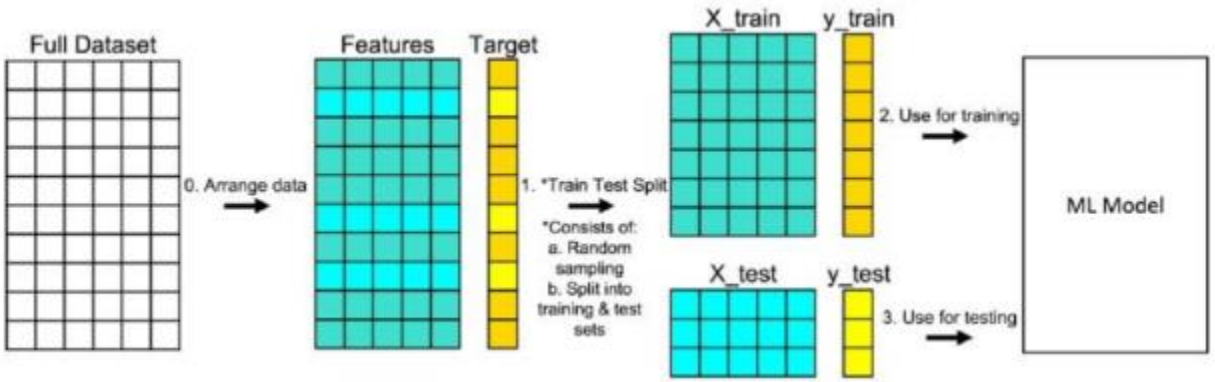


4. Documented and thorough research on architectural options due to the rate and spread of global development, leading to an appropriate evaluation for down-select [T]

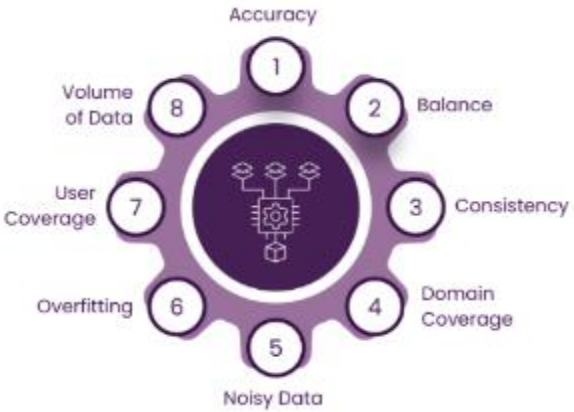
Linear regression	*K-nearest neighbor*
Naive Bayes	*Random forest K-Means clustering*
Neural networks (CNN, RNN, ANN, LSTM)	*Cluster analysis*
Convolutional neural networks	*Anomaly detection*
Logistic regression	*Apriori algorithm*
Support vector machines (SVM)	*Principal component analysis*

5. Human-autonomy trust contract is clearly articulated, with representative user input opportunities in the developmental flowchart to build that trust & related test metrics [HAT]

6. Quality assurance of the training & test data to ensure causal (factor) representativeness & appropriate class balance [DQ]



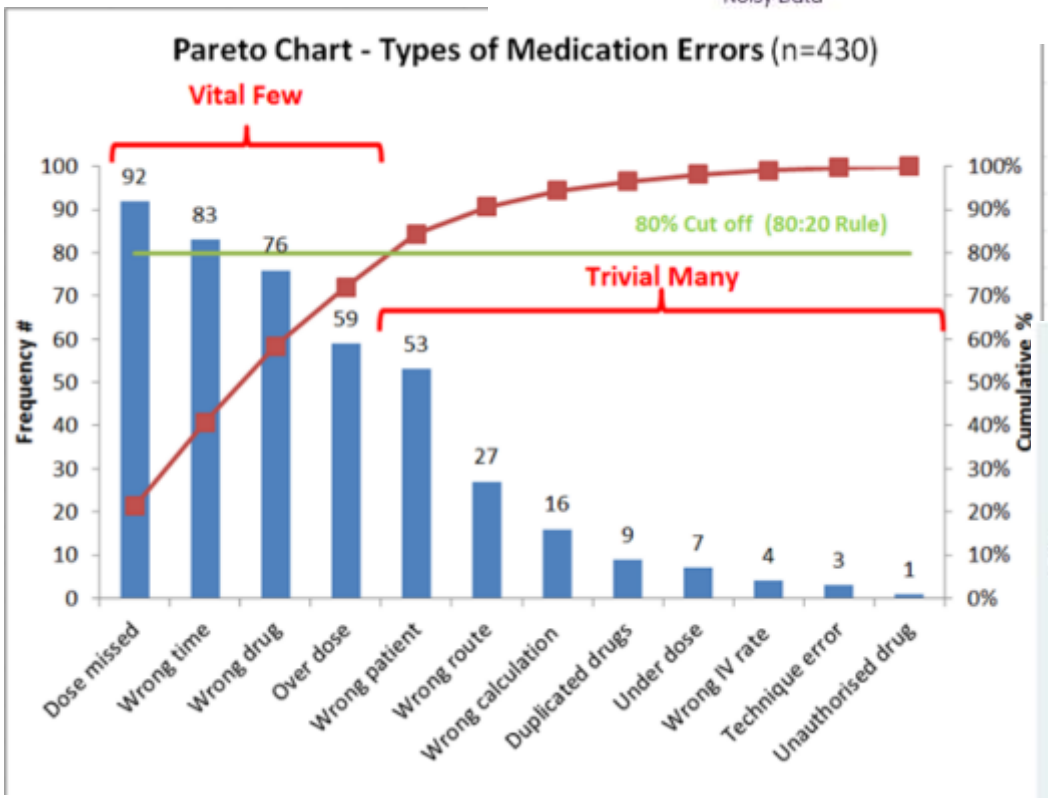
Note on Causal Factor Representativeness



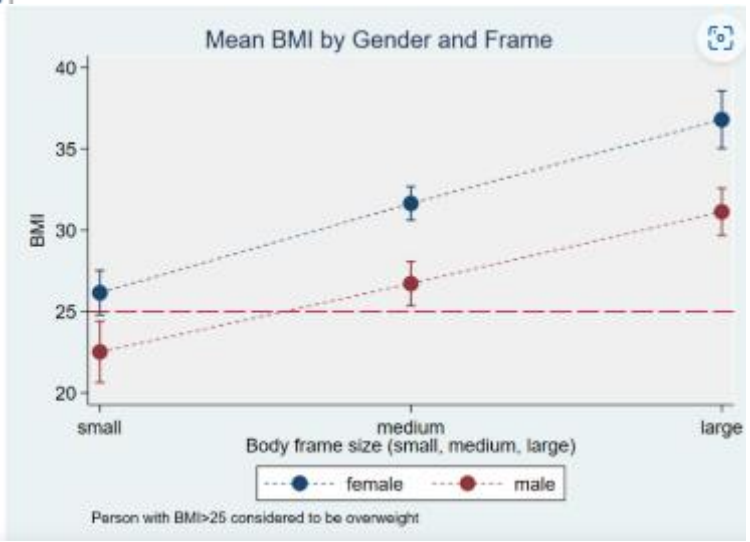
Know the domain

- Causal factors
- Pareto of most effect

  Direction of effect (marginal means plot)
- Characterise key factors in each training case
- Ensure training & test data are representative of factors

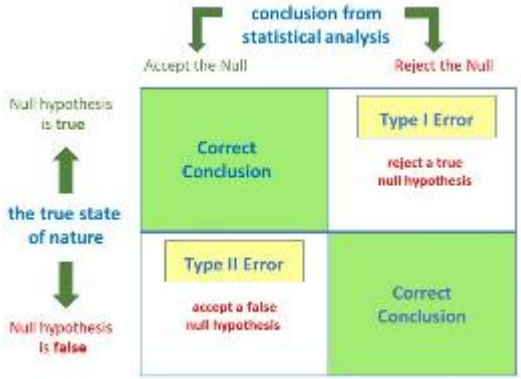


https://www.cec.health.nsw.gov.au/CEC-Academy/quality-improvement-tools/pareto-charts



https://www.theanalysisfactor.com/using-marginal-means-to-explain-an-interaction/

7. Comprehensive metric selection & prioritization, extending, where necessary, to an amalgamated objective function [T] & where there is a decision, the Type I and Type II errors both need to be checked (i.e., confusion matrix) [HAT]



		Predicted	
		Mismatch	Match
Actual	Mismatch	True Negative (TN)	False Positive (FP)
	Match	False Negative (FN)	True Positive (TP)

Recall Nanyonga (2025) NLP example last week

Models	Coherence Score	Perplexity
pLSA	0.7634	-4.6237
LDA	0.4394	-6.471
NMF	0.7987	2.0739
BERTopic's	0.264	-4.638

8. Thorough baseline performance known (i.e., causal factor representativeness, direction of effects), as often can get lost on how bad the current approach is & that's best justification [T]
9. Appropriate & documented sequence variation in training input (vary complexity order, random if you must) [DQ], [T]
10. Documented optimisation of hyperparameters (i.e., setting of the training) [T]

11. For Safety Critical or Operator Mission Critical AI-Enabled Systems undertake:

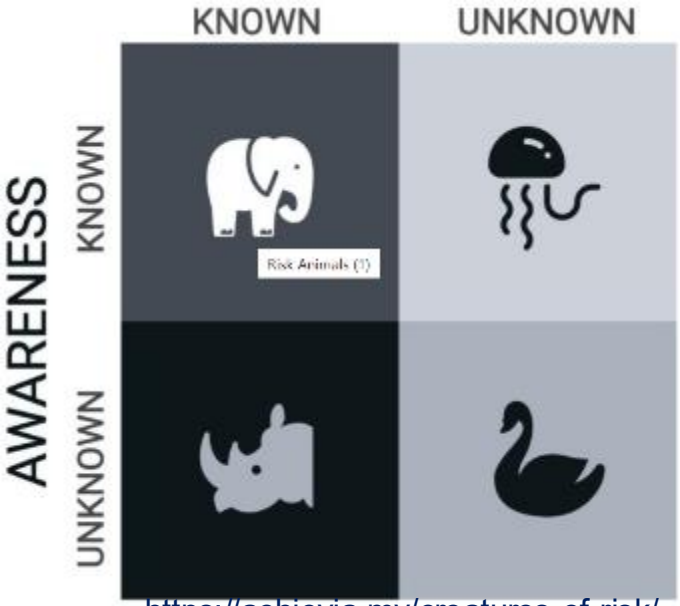
a) Explainable AI (XAI) to confirm algorithm triggers on main factors [T] & validate results with SMEs [HAT]

- To assert model transparency & trust by human users, deploy XAI to generate model explanations
- XAI is a technique applied in AI such that the results of a specific decision can be understood by humans.
- Various algorithms:
  - Local Interpretable Model-agnostic Explanation (LIME)
  - SHAP module to generate and visualize global explanations for each of the classifiers
- SHAP technique is a way of transforming “black box” AI models into transparent “gray box” models to enhance their trustworthiness by generating and visualizing global explanations for the model’s learned decision boundary.
- Works by assigning a numerical value (known as the shap_value) to each feature in the train set that defines its degree of importance to the model’s outputs

11. For Safety Critical or Operator Mission Critical AI-Enabled Systems undertake:

    b) Anti-fragility testing using Black Swan events & where necessary add exception training to additional outer layers or other mitigations [HAT]

- *"As we integrate machine learning (ML) into mission-critical systems in healthcare, finance, transportation, and social-scale infrastructure like power grids, a vital question arises about ensuring safety, security, and reliability despite myriad stressors (Hendrycks et al., 2021b). These manifest as natural or adversarial perturbations, aleatoric/epistemic uncertainties, distribution shifts (including domain shift, concept drift, nonstionarity, and out-of-distribution events). …*
- *there is no word for the exact opposite of fragile. Let us call it antifragile. Antifragility is beyond resilience or robustness. The resilient resists shocks and stays the same; the antifragile gets better. …*
- *Novel outliers routinely breach reactive defenses, raising concerns about their limitations against rare but impactful "black swan" events (Taleb, 2010). (Nair et al., 2022) present statistical arguments about why such long-tailed phenomena prove unexpectedly ubiquitous. …* [8 types, e.g.]
- ***Adversarial ML: Vaccination by attack**"*

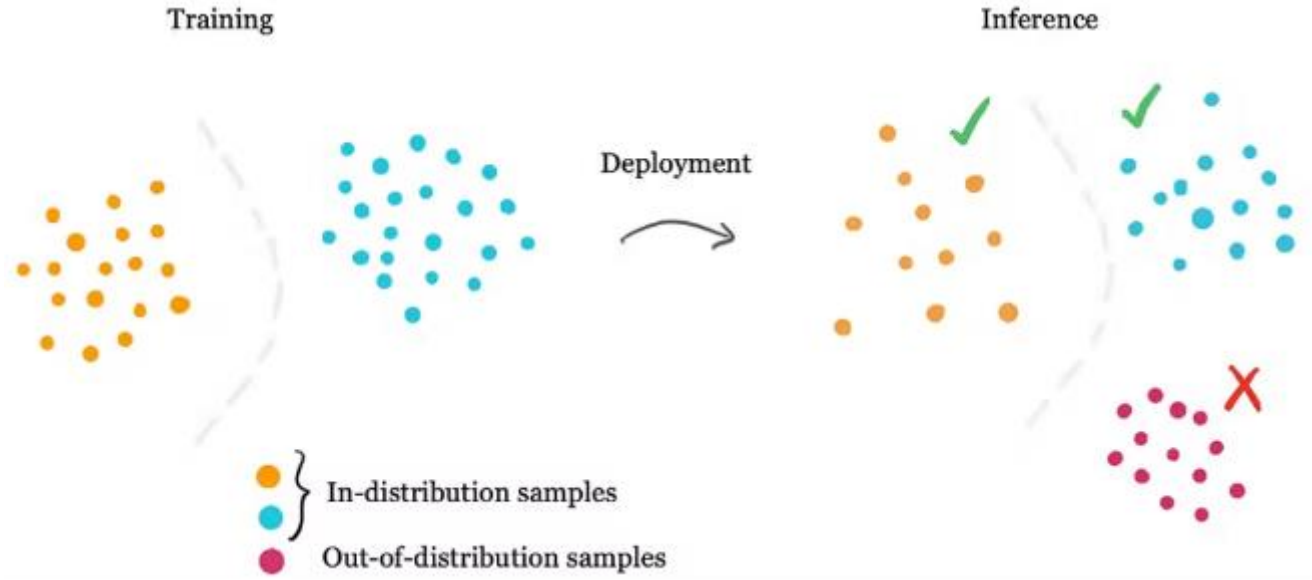


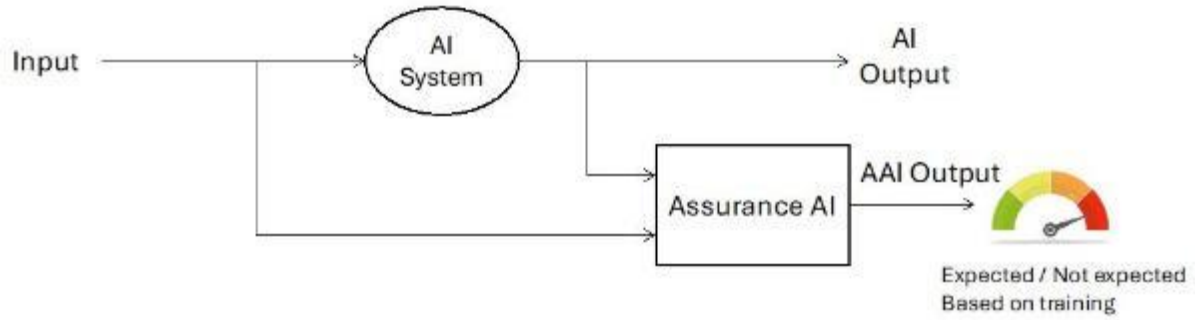
https://achievia.mv/creatures-of-risk/

"In contrast to the unpredictable nature of black swans, gray rhinos are probable events with high impact. We see these risks out there in the distance, but we don't clearly perceive their full dimensions."

Jin, M. (2024). Preparing for black swans: The antifragility imperative for machine learning. *arXiv preprint arXiv:2405.11397*.

12. For Safety Critical or Operator Mission Critical AI-Enabled Systems undertake Out of Distribution Detection or other AI monitoring AI with human alert [T], [HAT]





- *"When faced with out-of-distribution data, their activations can misfire, and their performance can drastically plummet, leading to unreliable or even hazardous outcomes in real-world applications.*
- *Despite their prowess and intricate loss function designs, models exhibit OOD brittleness primarily due to their training regimen. Their hyper-fine-tuning can make them less adaptable to unfamiliar inputs, emphasizing the need for novelty detection."*

https://encord.com/blog/what-is-out-of-distribution-ood-detection/

Suggested 20 Precepts of Human-Human Assurance

Anticipating Failure & Drift

- (H1) Look for and investigate small anomalies
- (H2) Watch for work practices drifting toward safety limits
- (H3) Treat supervisory assumptions as provisional and test them

Monitoring & Situational Awareness

- (H4) Stay closely attuned to what is actually happening
- (H5) Match supervision level to workload and system pressure
- (H6) Monitor interactions, not just individual components

Boundaries, Defences & Working Conditions

- (H7) Check that safeguards still work as intended
- (H8) Seek and fix upstream organisational contributors to error
- (H9) Make limits of safe operation clear and visible
- (H10) Resource work so safety isn't traded away by default

Communication, Reporting & Authority

- (H11) Create conditions where people raise issues early
- (H12) Ensure team members watch and back each other up
- (H13) Let those with expertise lead, regardless of rank
- (H14) Understand why people broke rules before punishing

Sensemaking & Decision-Making

- (H15) Resist simple stories about complex situations
- (H16) Treat supervision as continuous sensemaking
- (H17) Make safety–efficiency trade-offs explicit

Resilience & Adaptation

- (H18) Build capacity to absorb shocks and recover quickly
- (H19) Enable safe adaptation when procedures don't fit
- (H20) Follow through until risks are actually controlled

Suggested 20 Precepts of Software Assurance

Requirements Validity

- (S1) Derive software safety requirements from system hazard analysis.
- (S2) Treat software contributions as integral to the system safety process.
- (S3) Express hazardous software contributions concretely and verifiably.
- (S4) Use operational/incident data to refine software safety requirements.

Requirements Decomposition

- (S5) Preserve the intent of high-level safety requirements through decomposition.
- (S6) Validate decomposed requirements under realistic environmental and operational conditions.
- (S7) Ensure traceability captures rationale and safety intent, not just syntactic links.
- (S8) Re-evaluate safety intent when new design detail emerges.
- (S9) Structure software and specifications to support modular and compositional assurance.

Requirements Satisfaction

- (S10) Make software safety requirements clear, detailed, and verifiable.
- (S11) Use multiple, complementary verification and validation techniques.
- (S12) Target verification at hazard-relevant behaviours and failure modes.
- (S13) Explicitly account for limitations of each verification technique.

Hazardous Software Behaviour

- (S14) Analyse software design decisions for emergent hazardous behaviours.
- (S15) Prioritise systematic errors capable of generating hazardous behaviour.
- (S16) Apply greater rigour to safety-critical design elements.
- (S17) Continue identification and mitigation of hazardous software behaviour throughout the lifecycle.

Confidence (Cross-Cutting)

- (S18) Scale assurance rigour with the software's contribution to system risk.
- (S19) Assess appropriateness and trustworthiness of evidence, including assumptions.
- (S20) Combine multiple evidence types to achieve required confidence.

Drawn from Knight (2002), Woody et al. (2012) & Hawkins et al. (2013)

Suggested full 20 Precepts for AI Assurance

System risk, scoping & human role

- (A1) Risk led scoping & user story alignment
- (A2) Early hazard analysis & criticality banding
- (A3) Plan build test with integrated representative users
- (A4) Documented architecture selection & limits
- (A5) Human autonomy trust contract & escalation

Data, modelling & metrics discipline

- (A6) Data governance with provenance and suitability
- (A7) Causal factor representativeness
- (A8) Appropriate metrics & objective functions
- (A9) Established baselines & deltas
- (A10) Curriculum & hyper parameter discipline
- (A11) Explainability that is fit for assurance
- (A12) Adversarial & stress based robustness

Uncertainty, verification & stress testing

- (A13) Uncertainty & confidence in safety claims
- (A14) Shift, data drift & ODD compliance
- (A15) Integrated V&V strategy (testing + formal methods)
- (A16) Traceability of safety requirements, data, models & evidence

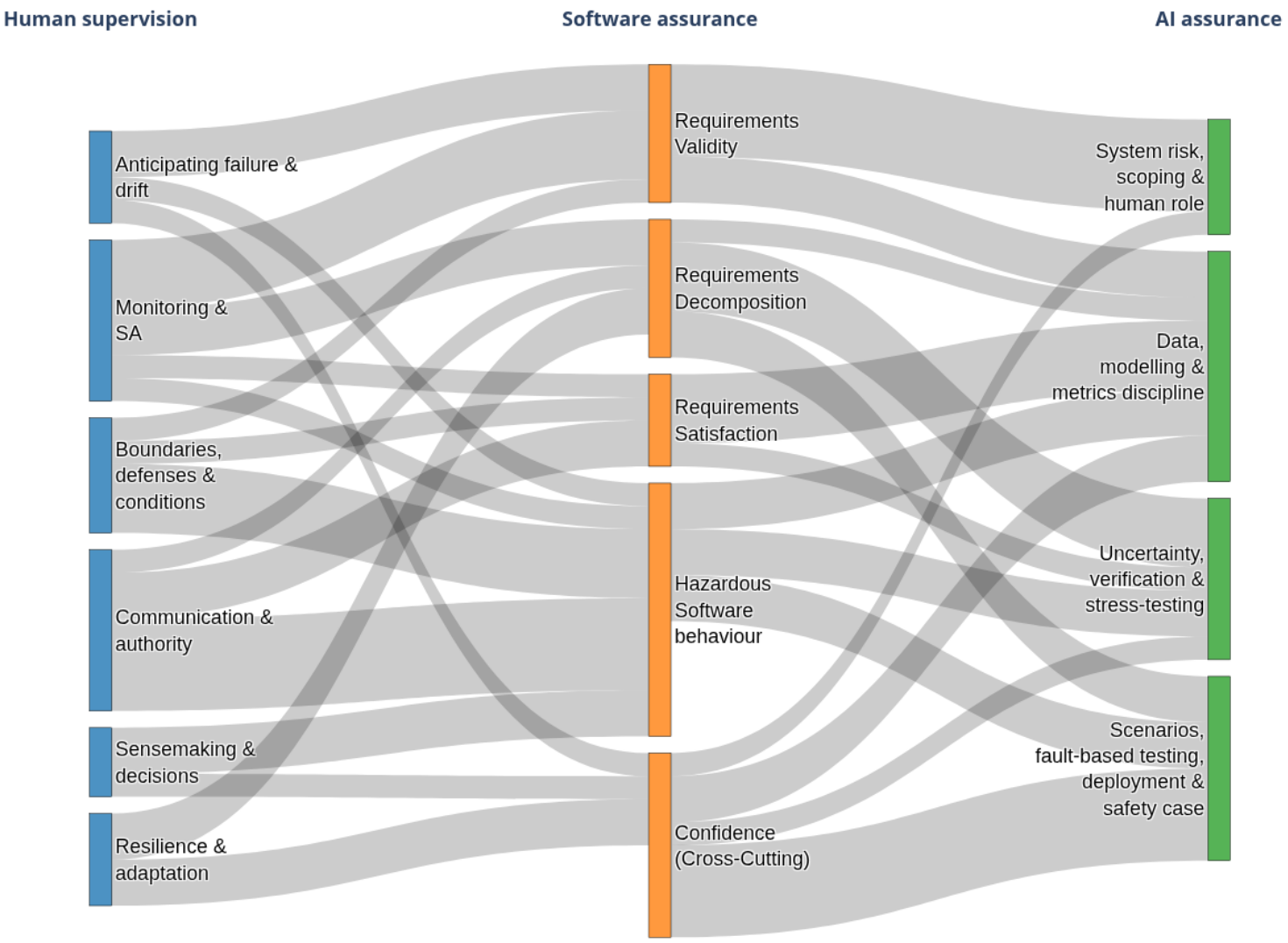
Scenarios, fault based testing, deployment & safety case

- (A17) Scenario-based and ODD-driven testing & simulation
- (A18) Runtime monitoring, safety indicators & recovery actions
- (A19) Change management, retraining triggers & re-qualification
- (A20) Structured safety case & evidence completeness

Theme	Principle	Software Assurance Principle	Theme
Anticipating failure & drift	Look for and investigate small anomalies	Derive software safety requirements from system hazard analysis.	Requirements Validity
	Watch for work practices drifting toward safety limits	Treat software contributions as integral to the system safety process.	
	Treat assumptions as provisional and test them	Express hazardous software contributions concretely and verifiably.	
Monitoring & SA	Stay closely attuned to what is actually happening	Use operational/incident data to refine software safety requirements.	
	Match supervision level to workload and pressure	Preserve the intent of high-level safety requirements through decomposition.	Requirements Decomposition
	Monitor interactions, not just individual components	Validate decomposed requirements under realistic environmental and operational conditions.	
Boundaries, defenses & conditions	Check that safeguards still work as intended	Ensure traceability captures rationale and safety intent, not just syntactic links.	
	Seek and fix upstream organisational contributors to error	Re-evaluate safety intent when new design detail emerges.	
	Make limits of safe operation clear and visible	Structure software and specifications to support modular and compositional assurance.	
	Resource work so safety isn't traded away by default	Make software safety requirements clear, detailed, and verifiable.	Requirements Satisfaction
Communication & authority	Create conditions where people raise issues early	Use multiple, complementary verification and validation techniques.	
	Ensure team members watch and back each other	Target verification at hazard-relevant behaviours and failure modes.	
	Let those with expertise lead, regardless of rank	Explicitly account for limitations of each verification technique.	
	Understand why people broke rules before you punish	Analyse software design decisions for emergent hazardous behaviours.	Hazardous Software behaviour
Sensemaking & decisions	Resist simple stories about complex situations	Prioritise systematic errors capable of generating hazardous behaviour.	
	Treat supervision as continuous sensemaking, not periodic checks	Apply greater rigour to safety-critical design elements, and explicitly assess the assurance and trustworthiness of dependencies.	
	Make safety–efficiency trade-offs explicit and deliberate	Continue identification and mitigation of hazardous software behaviour throughout the lifecycle.	
Resilience & adaptation	Build capacity to absorb shocks and recover quickly	Scale assurance rigour with the software's contribution to system risk.	Confidence (Cross-Cutting)
	Enable safe adaptation when procedures don't fit	Assess appropriateness and trustworthiness of evidence, including assumptions.	
	Follow through until risks are actually controlled	Combine multiple evidence types to achieve required confidence.	

Theme	Software Assurance Principle
Requirements Validity	Derive software safety requirements from system hazard analysis.
	Treat software contributions as integral to the system safety process.
	Express hazardous software contributions concretely and verifiably.
	Use operational/incident data to refine software safety requirements.
Requirements Decomposition	Preserve the intent of high-level safety requirements through decomposition.
	Validate decomposed requirements under realistic environmental and operational conditions.
	Ensure traceability captures rationale and safety intent, not just syntactic links.
	Re-evaluate safety intent when new design detail emerges.
	Structure software and specifications to support modular and compositional assurance.
Requirements Satisfaction	Make software safety requirements clear, detailed, and verifiable.
	Use multiple, complementary verification and validation techniques.
	Target verification at hazard-relevant behaviours and failure modes.
	Explicitly account for limitations of each verification technique.
Hazardous Software Behaviour	Analyse software design decisions for emergent hazardous behaviours.
	Prioritise systematic errors capable of generating hazardous behaviour.
	Apply greater rigour to safety-critical design elements, and explicitly assess the assurance and trustworthiness of dependencies.
	Continue identification and mitigation of hazardous software behaviour throughout the lifecycle.
Confidence (Cross-Cutting)	Scale assurance rigour with the software's contribution to system risk.
	Assess appropriateness and trustworthiness of evidence, including assumptions.
	Combine multiple evidence types to achieve required confidence.

AI Assurance Principle	Theme
Risk-led scoping & user-story alignment (& safety-requirements alignment)	System risk, scoping & human role
Early hazard analysis & criticality banding	
Plan-build-test with integrated representative users	
Documented architecture selection & limits	
Human-autonomy trust contract & escalation	
Data governance with provenance and suitability	Data, modelling & metrics discipline
Causal (operational) factor representativeness	
Appropriate metrics & objective functions	
Established baselines & deltas	
Curriculum (Reproducible training) & hyper-parameter discipline	
Explainability that is fit-for-assurance	
Adversarial & stress-based robustness	
Uncertainty & confidence in safety claims	Uncertainty, verification & stress-testing
Shift, data drift & ODD compliance	
Integrated V&V strategy (testing + formal methods)	
Traceability of safety requirements, data, models & evidence	
Scenario-based and ODD-driven testing & simulation	Scenarios, fault-based testing, deployment & safety case
Runtime monitoring, safety indicators & recovery actions	
Change management, retraining triggers & re-qualification	
Structured safety case & evidence completeness	



Sankey Diagram

Conclusions

- Great AI RMFs exist

  We should simply adopt one & expect compliance…

- Four functions: Govern, Map, Measure, Manage

  Focus on data quality [DQ], transparency [T], human oversight[HAT]

- AI ML awareness is critical to understanding risks

- System safety & security engineering frameworks are key to managing AI-enabled systems

  Enables audit & robust testing tailored to latest techniques

- 20 suggested principles to measure awareness training & test competency

  Can trainees understand them?
  Can they apply them to assurance deliverables & cooperative test events?

References

Weick, K. E. & Sutcliffe, K. M. (2016), *Managing the unexpected : sustained performance in a complex world, 3rd Ed.*, Wiley, Hoboken, New Jersey.

Reason, J. T. (1997), *Managing the risks of organizational accidents*, Ashgate, Aldershot, Hants, England.

Rasmussen, J. (1997), Risk management in a dynamic society: a modelling problem, *Safety Science*, 27(2), pp. 183–213.

Knight, J. (2002), Safety critical systems: challenges and directions, Proceedings of the *24th International Conference on Software Engineering*. ICSE 2002, DOI: 10.1145/581339.581406.

Woody, C.; Mead, N. & Shoemaker, D. (2012), Foundations for Software Assurance, 45th Hawaii *International Conference on System Sciences*, DOI: 10.1109/hicss.2012.287.

Hawkins, R.; Habli, I. & Kelly, T. (2013), The principles of software safety assurance. In 31st *International System Safety Conference* (pp. 12-16). Boston, Massachusetts USA.

Ashmore, R.; Calinescu, R. & Paterson, C. (2021), Assuring the Machine Learning Lifecycle, *ACM Computing Surveys*, vol. 54, DOI: 10.1145/3453444.

Neto, A. V. S.; Camargo, J.; Almeida, J. R. & Cugnasca, P. (2022), Safety Assurance of Artificial Intelligence-Based Systems: A Systematic Literature Review on the State of the Art and Guidelines for Future Work, *IEEE Access*, vol. 10, DOI: 10.1109/access.2022.3229233.

Paterson, C.; Hawkins, R.; Picardi, C.; Jia, Y.; Calinescu, R. & Habli, I. (2025), Safety assurance of Machine Learning for autonomous systems, *Reliability Engineering & System Safety*, DOI: 10.1016/j.ress.2025.111311.