Ensuring Reliable AI in Real World Situations

Introduction

As Artificial Intelligence (AI) capabilities have grown over the past decade, we bear witness to a proliferation of real-life applications. Across industry, financial services, public services, security and critical infrastructure AI adoption is on the rise thanks to advances in stability, scalability and transparency. Time and again, Machine Learning (ML) accurately identifies patterns in complex data, presenting us with new opportunities to automate time-consuming tasks or even to push beyond the boundaries of human capabilities. Its inherent compatibility with other technologies such as cloud computing has ushered in unprecedented performance gains in all kinds of processes – and at scale. Implemented correctly, ML systems can have a truly transformational effect.

And yet, proper implementation remains a challenge. There are numerous factors to consider, from the algorithm/architecture to hyperparameters and, most importantly, the data used to train the model. Decisions based on poorly designed or executed AI may harm individuals "processed" by the AI and damage the reputation of the organization providing the AI system. The growing reliance on nascent AI technologies, especially in more critical or sensitive applications, raises concerns whether they are reliable and robust enough to handle less-than-perfect, real-world conditions beyond the safe confines of the lab. It is essential for us to understand how such systems can – and likely will eventually – fail in order to design better systems in the future: Systems that are less likely to fail. Systems that, if they were to fail, will do so within acceptable limits and without causing serious economic, physical or psychological harm.

According to most experts, Deep Neural Networks (DNNs) already surpass their human counterparts in performing specific tasks such as classifying images. They are also notoriously complex and difficult to decipher, the rationale behind their predictions a mystery to developers and consumers alike. This not only negatively impacts acceptance among the general public but also AI's potential to produce reliable outcomes. Researchers in the field of robustness have drawn attention to significant flaws in the way artificial neural networks make decisions that leave them susceptible to adversarial attack, misuse and technical failure. Bad actors do not even require full access to the internal architecture to manipulate a model[1, 2].

Instead, they can use techniques that introduce imperceptible perturbations into the model data to compromise the system and alter its behavior (predictions or decisions). In many ways, hacked models are even more dangerous than inoperable models: If manipulation goes unnoticed, the models will continue to operate, but in ways contrary to the developer's original intent. It is vital for designers to make their models resilient against (intentional) noise in the input data, also known as an adversarial attack[1, 2]. In the context of ML, we measure resilience in terms of the amount of noise necessary to change a model's decision on the input data.

Robustness is a far wider topic than resilience to adversarial attack. A model that is optimally trained to reproduce outcomes with training data but fails to generalize to other cases is termed "over-fit." For an AI model to be useful, it must be capable of generalizing beyond training data. Data has by far the largest impact on AI model performance – in terms of the quality, completeness and volume of a representative training dataset – and not just in the pre-launch phase, but also throughout the model's lifecycle. To evaluate the robustness and generalization of models, we point out common pitfalls and minimize risk through techniques such as regular retraining, anticipation of potential adversaries or cross-validation testing to combat over-fitting.

In the following sections, we examine typical challenges to model robustness and offer potential solution strategies. We conclude with an intuitive framework designed to proactively evaluate the robustness and reliability of models – addressing potential failures before they occur.

Where AI systems fall short

Despite their impressive abilities, AI systems are not sufficiently mature to autonomously manage critical applications in complex environments. The unique traits of AI may introduce new vulnerabilities beyond the classic variety common to any software. There are three distinct criteria for an AI system to meet before it may be considered ready to deploy in a real-world environment:

- **Reliability:** The prediction accuracy meets expectations consistently over time, avoiding too many oversights or false alarms.
- **Stability:** The model performs well both generally and under stress conditions, such as in edge cases. It is not overly sensitive to naturally occurring noise, targeted noise is covered by resilience.
- **Resilience:** Model behavior is not easily manipulated through exploitation of vulnerabilities (either within the code or the training data).

While these criteria may be universally valid, deficiencies in the models are context and task-dependent, as in, for example, a time-series analysis for non-stationary data (e.g., analysis of the product prices). Common causes may contain/common causes are:

- **Unrepresentative data:** The data (and its labels) used to train the algorithm does not represent the real-world environment in which the model is designed to operate.
- **Annotation quality:** Human annotation brings a subjective interpretation to raw data (classification into types). Labelling can vary widely between human "taggers" who interpret raw data (objects in images) based on their own life experience. The resulting model predictions may be either erratic or consistently inaccurate.
- **Overfitting:** The model succeeds only in "predicting the past" (e.g., the data on which it was trained) and cannot generalize to accurately predict based on new data it encounters in operation.
- **Model decay:** Performance decreases over time as the operational environment evolves past the dynamics learned during development. In other words, the training data that was once representative can no longer predict outcomes in the world as it is now – a phenomenon known as covariate shift or data drift. Model decay may also occur when the relationship between the input features and target variables changes (e.g., due to seasonality) – a phenomenon known as concept shift.
- **Under-specification:** Even if there is consistency in the pipeline (i.e., training data, features, pre-processing, algorithm selection), models can behave erratically when there are several ways to achieve the same performance on the evaluation dataset[2]. Even an optimally tuned model may not be a suitable reflection of reality.

Evaluating the robustness of machine learning models

A thorough evaluation of an AI system demands that we not only consider model performance (e.g., prediction accuracy, bias, transparency, computational cost) but also the robustness criteria (reliability, stability, and resilience). These will impact how we choose suitable metrics and procedures, along with data types, model class and other specifications pertaining to the use case. Metrics play a fundamental role: The right metrics will provide a quantitative guide for optimization, the wrong metrics risk optimizing toward the goals not central to our objectives.

These evaluation procedures build on common validation practices (k-fold cross-validation), testing for imbalance, detection of spectrum bias (capturing necessary diversity and complexity) and out-of-sample validation (to limit overfitting), among others. We intentionally subject the model to stress situations to determine how well it will handle more ambiguous cases. Dealing with approximate situations is a strong selling point for AI models – unambiguous cases may be treated by simpler models often make incorrect predictions when they are overly sensitive to unfamiliar "edge cases", which conventional testing has failed to sufficiently address. To earn our trust, we need AI models to be flexible enough to handle edge cases as well as other imperfect situations.

Besides the sensitivity to edge cases, machine learning also introduces new attack vectors such as adversarial attacks, which are particularly important to consider. Bad actors can exploit these vulnerabilities in subsequent stages of the AI processing chain, posing multiple threats that could potentially add up to an aggregated risk of system failure. Many forms of attack require knowledge of the model parameters – so-called white box adversarial attacks. Conversely, black box adversarial attacks (requiring no inside knowledge) present another very real risk.[3] In a grey area between the black box and white box approach, researchers have found that some adversarial attacks may actually be transferable[4], i.e., malicious samples designed to attack a known model (white box) can also be effective against another, unknown (black box) targets[5, 6].

Comprehensive testing demands multiple datasets in order to accommodate edge cases that arise for different reasons: A representative example may have been overlooked in the training data, input data may have become corrupted or certain situations may be unrecognizable to the model.

To assess the risk of model decay, we measure degradation based on either covariate shift (data drift), concept shift or a mix of both. There are several established methodologies to detect model decay: statistical, window-based and ensemble-based.

Statistical methods of detecting changes in the data (covariate shifts) include variations of the sequential probability test (SPRT) to examine the following:

1. the logarithm of probability distribution ratio for the features,

2. alterations of the classic 3-sigma rule that measures to what extent the "signal" deviates from the expected trajectory, and

3. drift detection methods (DDM) through monitoring of probability of model misclassification or through the estimation of local density change using the nearest neighbor of each data point in the dataset.

A significant drawback to these methods is their failure to indicate where or when the data drift occurs. We can overcome this by identifying the differences between Gaussian Mixture Models (GMM) – applied to approximate datasets at multiple points in time. We calculate the difference between GMMs using statistical techniques such as the Jenson-Shannon (JS) distance, which indicate drift where we observe a large gap between the clusters.

A more advanced method of detecting concept drift is the Hierarchical Linear Four Rates (HLFR) framework[7], an improved form of DDM. There are also several other statistically based methods that do not require training sets to detect both concept and covariate shifts: the Population Stability Index (PSI), Kolmogorov-Smirnov statistics and the Kullback-Leibler (KL) or Jensen-Shannon (JS) divergence test.

Window-based detectors are highly efficient at dealing with sequential or even real-time streaming data. They all originate from the Hoeffding Tree algorithm for incrementally building decision trees, e.g., constructing a decision tree from streaming data without storing examples in memory to utilize data before and after the detected shift. They employ a sliding window of adaptive size (for example, a two-window approach in ADwin algorithms[8]).

Another large group of detectors is based on ensemble learning techniques designed to improve on simple window-based approaches. The ensemble approach combines drift detection algorithms that can distinguish between different types of shifts, for instance, abrupt changes vs. gradual drifts. The ensemble method integrates several base methods and combines their advantages to obtain better predictive performance. Typical fusion rules for ensemble models, such as majority voting or weighted voting, are not appropriate for drift detection as different base detectors usually find a drift at different time steps. For this reason, we find the best ensemble strategy for detecting concept drift is to acknowledge where any base detector triggers[9, 10].

Making machine learning models more robust

Machine learning models trained on data from the outside world can be corrupted by malicious attacks that, e.g., inject malicious points into the models' training sets. Common defense strategies against these attacks are:

- **Data sanitization**: cleansing data of potentially malicious content before it trains the model
- **Robust learning**: redesigning the learning routine to defend against malicious actions and unknown situations

Whereas data sanitization is self-explanatory, robust learning is rather more nuanced. We look at two different approaches in the following, which are both based on the security-by-design principle: adversarial training and regularization.

Adversarial training: Attack algorithms probe decision boundaries to determine the relative ease to corrupt a model. With adversarial training, however, models are less prone to attack algorithms as they are already exposed to adversarial samples during training. Our objective is to strengthen models against such adversarial attacks, learning from new and different examples to improve their ability to resist them. While sophisticated statistical methods do exist, adversarial training is clearly the most effective approach to building algorithmic defenses[11]. Adversarial training makes use of those adversarial examples produced by attack algorithms, which were successful in compromising a model during decision-boundary probing.

Regularization: The predictive power of ML in production suffers when the model does not generalize well and only remembers samples from training. Regularization offers a pathway to alleviate these shortcoming.

Regularization for NN refers to the L1 (Lasso) or L2 (Ridge) norm, where an additional term is added to the loss function to penalize a considerable number or substantial values of parameters. In terms of robustness, we adopt the more general definition of regularization as any technique that reduces overfitting. Regularization is applied by modifying hyperparameters, for example, by limiting the maximum depth of decision trees, triggering early stopping or prompting dropout layers in neural network or model pruning.

Our interest in these methods is not to evaluate failures post-mortem, but to build models that are inherently robust before they are deployed.

Two caveats:

1. Excessive use of regularization can degrade predictive power due to over-simplification of the model, a scenario known as under-fitting.

2. Any of these approaches could result in a general degradation of performance (inherent to statistical approaches).

Paradigm shift: reactive to proactive

Our interest in these methods is not to evaluate failures post-mortem, but to build models that are inherently robust before they are deployed. This avoids propagation of risk and unnecessary downstream costs. We propose a proactive strategy that integrates methods and associated metrics into a logical workflow. (We coded this methodology into a toolset to enable our ML engineers and auditors to proactively identify potential failure modes and resolve them in future model iterations.) Our analysis is based on the same robustness criteria outlined above, operationalizing them in the context of the AI model assessment methodology:

1. **Reliability of specification:** testing whether the model functions as expected and solves the intended problem

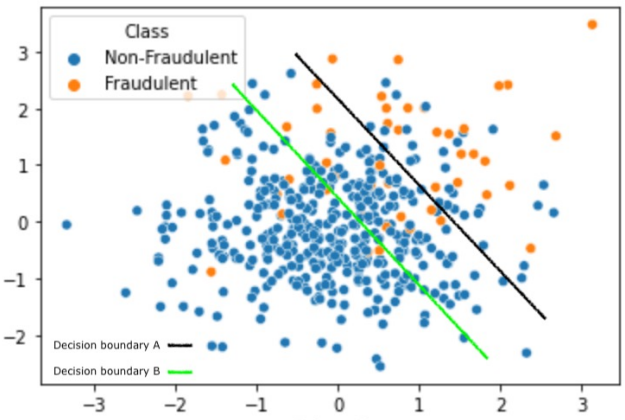
2. **Stability of features:** quantifying and categorizing the effect of feature space disruption on the model performance

3. **Resilience to edge-case vulnerabilities:** determining the effort required to undermine the model and deducing which samples are at risk through targeted attacks

Reliability of specification

The first step is to specify the risk tolerance of the industry model application. Regulated and high-risk applications, such as IRB risk models, usually require a strict quantitative model validation. Traditionally, we use performance metrics such as recall for classification or Root Mean Squared Error (RMSE) for regression to evaluate a model. A higher recall score or a lower RMSE would suggest a reliable model. Yet these metrics fail to address robustness. As illustrated in figure 1, a fraud detection model may have a high-performance metric score but generate little or no business value. This is because ML algorithms are designed to minimize errors (formulated by loss functions), which makes them more prone to "adapt" to the majority class of non-fraudulent activities.

Fig. 1 – Decision boundary example



Source: illustration

In the illustration above, the desired outcome of the ML algorithm is to produce decision boundary B (green line), where the number of misclassifications of fraudulent activities is low and the number of misclassifications of non-fraudulent activities is high. This produces a higher error rate, because the number of misclassified non-fraudulent activities exceeds that of the misclassified fraudulent activities. To reduce the total number of misclassified samples, the ML algorithms will instead produce decision boundary A, where the number of misclassified points from both classes are approximately equal. This, however, yields more misclassified fraudulent activities compared with boundary B, a more costly outcome. The chosen performance score, in this example, alone is sufficient to ascertain whether the model will provide relevant results in a real-world business context.

We evaluate model reliability on a more granular level by means of the following questions about prediction confidence[1,2]:

- When the predictions are correct, how under-confident is the model?
- When the predictions are wrong, how over-confident is the model?

Dedicated metrics answer these questions quantitatively and allow us to compare the confidence results with a benchmark model created for the task. The approach applies both to classification and to regression models. One of the advantages of AIQualify is that the user can specify what right or wrong is. Therefore, this is not "our" definition. At the time of this paper's publication, our extensive research did not identify any application (neither open source nor proprietary) that illustrates the under/over-confidence concept to regression problems. We hope to inspire future work in this new field with our toolset.

Stability of features

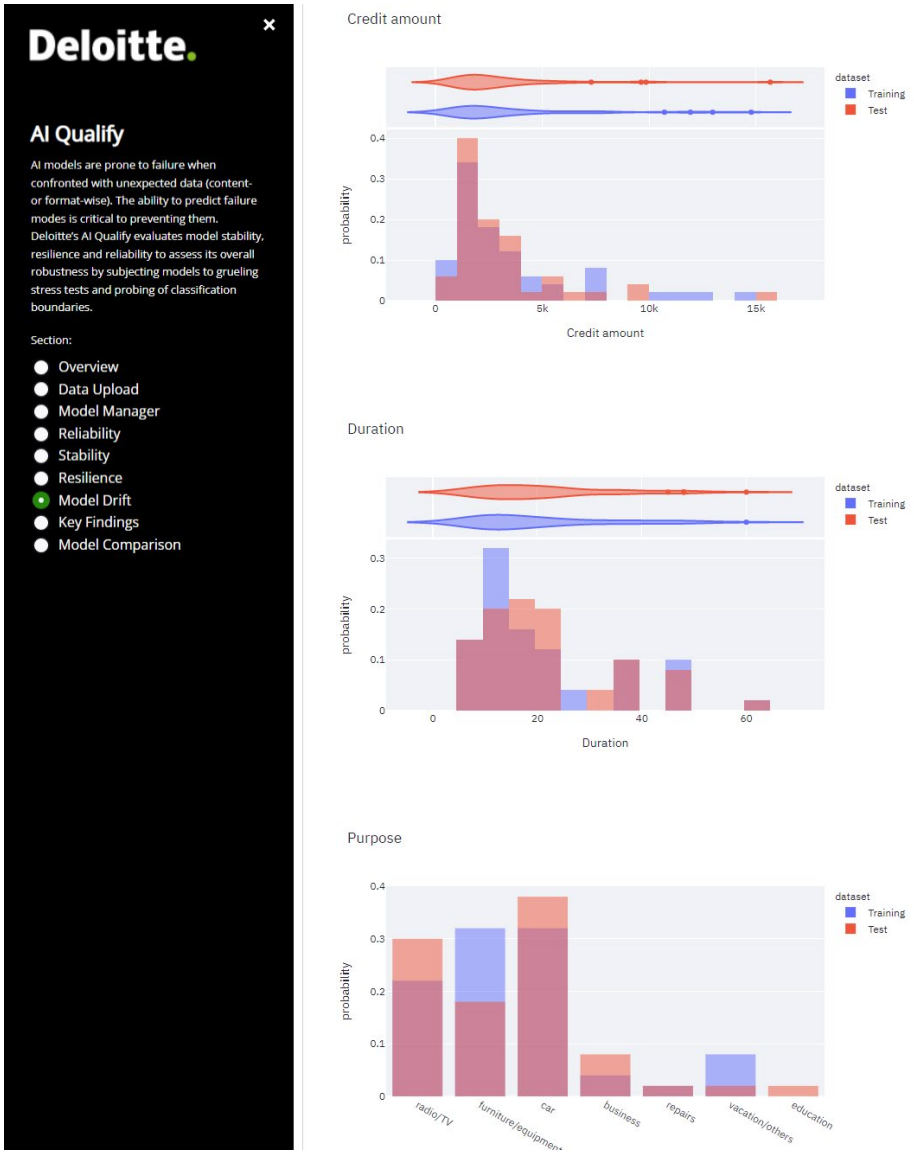
The second step assesses model stability through an augmented form of Monte Carlo Simulation. In addition to conventional Monte Carlo simulations, we explore the connection between feature importance and the inherent risk level of each feature. In the case of linear algorithms, the risk associated with a model feature is correlated to its importance, as high-ranking features tend to be the most disruptive. For more complex, non-linear algorithms such as neural networks or even tree-based algorithms, the relationship between feature importance and model disruption is not so straightforward. For example, we derive feature importance from the values of regression model coefficients in order to quantify their individual impacts on the final prediction result. There is no such directly attributable effect for tree-based models. It is still important to have a clear understanding of direct feature impact, because it gives us insight into the potential risk introduced by each feature. Our methodology quantifies the effect of the disruption and categorizes the features in line with the potential risks. This involves two levels of analysis:

1. **Individual disruption** – features perturbed independently

2. **Collective disruption** – a combination of features perturbed simultaneously

In the case of collective disruption, features that are not disruptive individually may become disruptive when combined with other variables. We must consider both individual and collective disruptions to ascertain which features (or their combinations) will introduce persistent change in the model's output in order to improve model performance.

Fig. 2 – AI Qualify User Interface



Source: AI Qualify

Resilience to edge-case vulnerabilities

The third and final step evaluates and quantifies the effort required to undermine the target model. We utilize adversarial attack algorithms to estimate a lower bound of "effort" required for successful attacks. Conversely, we determine a verified robust upper bound of tolerable input deviation. Combining these steps is central to our methodology. Adversarial attacks are highly sensitive to the initial (user-defined) threshold values, determining the amount of noise allowed in the attack algorithms. Threshold values drive the upper bound.

However, robustness verification itself does not reveal any underlying drivers (e.g., noise vectors) that are important for addressing robustness issues. We compensate for this by combining adversarial attacks with robustness verification to understand the target model's resilience in edge cases.

When we integrate a noise correction algorithm[13], we make sure it is compatible with tabular data that contains a mixture of numerical and categorical features, as commonly found in financial services databases.

To make a comprehensive analysis of model resilience, we evaluate the model at multiple levels:

- **Prediction resilience** – the "effort" required to undermine the decision in the model
- **Adversarial examples** – the adversaries produced by our methodology
- **Noise vectors** – the noise generated to produce adversarial examples
- **Adversarial outcome** – the results of testing, e.g., how the classification algorithm reacts to targeted noise

The resilience score indicates how easily a model can be compromised. For example, we should reject a loan application model with a very low resilience score for the class "application declined". Adversaries could easily hijack this model, changing its behavior to approve poor quality applications – potentially at great cost. We first pinpoint such vulnerabilities via adversarial examples, then utilize them to enhance robustness through adversarial training. The noise vectors and the adversarial outcome will also allow us to deduce which data samples are at risks:

- Data points with imperceptible (small values) noise vectors
- Data points with counter-intuitive noise vectors

Imperceptible noise vectors are those close to the decision boundary, requiring little perturbation to be undermined. Highlighting these data points is useful, because it shows how seemingly harmless data profiles can potentially break the model. Counter-intuitive noise vectors allow us to capture exceptions/edge cases to our models. Again, in the context of the loan application example, lowering a customer's income should not change the loan application outcome from "application declined" to "application approved."

Measuring robustness effectively

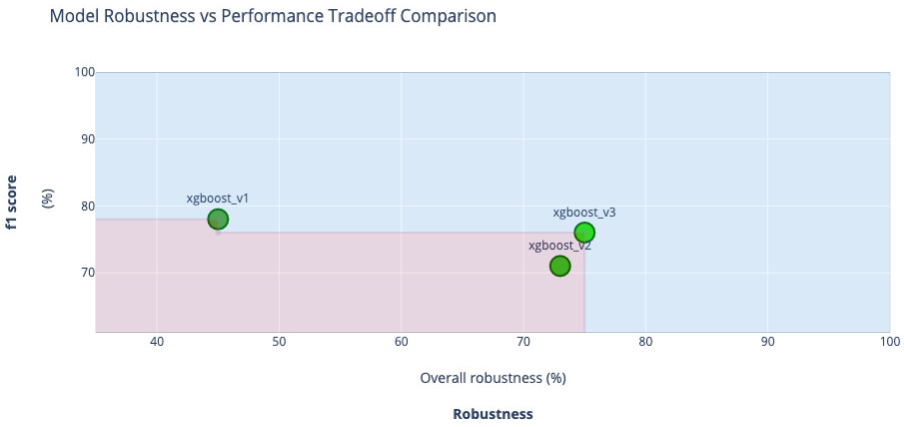
The most effective means to evaluate the robustness of ML models is by carefully combining complementary proactive and reactive strategies. We mapped the testing procedure in a workflow tool ("AI Qualify") that helps users navigate the evaluation process, ensuring a consistent approach across model types and iterations. Our tool is based on the same robustness criteria introduced earlier: reliability, stability and resilience.

We evaluate each of these from multiple angles and then consolidate them into a single score* for each criterion. The individual "criterion scores" aggregate into an overall robustness score, which allows for quick comparison between competing models (champion-challengers) across different iterations of the same model. The topic-specific scores and underlying metrics invite users to delve into the detail, identifying the nature of the robustness problem and determining whether remediation actions is needed.

* The metric ranges from 0 to 100 where higher is better

"AI Qualify" adopts the reactive approach to drift detection – for both covariate shift and concept shift. The usual performance metrics (e.g., recall, precision, F1 score) can only capture model performance in a quantitative manner. Adding the robustness metrics provides a more holistic view. We plot successive builds of models under investigation along a performance/robustness plane. This is a convenient, intuitive way to visualize the trade-offs between the two characteristics, providing developers with quick feedback to model optimization efforts, in particular the effect of remediation actions (if any) across multiple iterations.

Fig. 3 – Model Robustness vs. Performance



Source: AI Qualify

Conclusion

Robust AI models have never been more important, as AI goes mainstream and is deployed in ever more mission-critical applications. Based on our research and analysis, we conclude that:

Robustness is more than just a defense against adversarial attacks. More often than not, there is not a lone culprit for fragility in ML models. With many factors at play, it would be difficult to presume one is more significant than the other. We can observe this clearly along the entire model pipeline:

- **Data exploration** – risk of non-representative training data
- **Training and validation** – risk of overfitting and under-specification
- **Monitoring** – risk of model decay

The fact that all of these stages are interconnected suggests that there is no silver bullet, no one-size-fits-all solution for robustness.

Both proactive and reactive approaches are critical to success. To craft ML models that are truly robust and reliable, we blend an ex-ante with an ex-post approach. Detecting (and monitoring) concept shift is fundamentally after-the-fact, yet it is also critically important to alert users about inevitable model decay. A uniquely ex-post approach puts us in a sub-optimal "wait-and-see" posture that can be costly to business – especially in high-risk industries. A pre-emptive approach promises to alert users before risks become a problem, yet it cannot detect risks such as model decay. Current proactive measures such as robustness verification are insufficient, even when paired with reactive monitoring.

To create a strong robustness solution, we must adopt a comprehensive approach based on three fundamental perspectives:

- **Reliability** of specification
- **Stability** of features
- **Resilience** to edge-case vulnerabilities

Properly applying and interpreting the many perspectives, methods, metrics, and tests discussed in this paper is a challenge in its own right. For this reason we organize them into workflows to ensure a sound, consistent assessment. "AI Qualify" gives developers and auditors the analytical tools they need to interpret results, devise remedies and monitor performance. It also helps document the validation process, which is particularly important for regulated industries such as banking and insurance.

We all have high expectations for AI. Technological advances and accumulation of success stories across domains give us good reason to do so. In many respects, however, current AI systems have a long way to go to earn our trust, especially in critical applications.

Arguably the most crucial component to developing or maintaining an AI system is data, the foundation for modern AI. The ability of AI algorithms to extract rules from data give them unprecedented predictive capabilities, yet also pose additional challenges. All models – traditional or machine learning – can go astray with faulty production data. The very nature of machine learning AI, deducing a function from training data, introduces a new class of risk. If AI systems are built on faulty training data to begin with, they can fail even if they are later fed high quality production. AI systems trained only prior to launch can lose accuracy over time, becoming disconnected with the realities of an ever evolving world. At the other end of the spectrum, models that automatically update can be hijacked to behave in ways unintended by their developers. A system that is robust will guard against both of these risks and still remain a powerful predictive tool – whether as a decision aid or an automation enabler. This is what makes the difference between good AI and great AI.

References

[1] Jianbo Chen, M. I. (2019). HopSkipJumpAttack: A Query-Efficient Decision Based Attack. CoRR.

[2] Ian J. Goodfellow, J. S. (2015). Explaining and Harnessing Adversarial Examples.

[3] Aleksander Madry, A. M. (2019). Towards Deep Learning Models Resistant to Adversarial Attacks. arXiv.

[4] Mahmood, K., Gurevin, D., van Dijk, M., & Nguyen, P. H. (2021). Beware the black-box: On the robustness of recent defenses to adversarial examples. Entropy, 23(10), 1359.

[5] Papernot, Nicolas & McDaniel, Patrick & Goodfellow, Ian & Jha, Somesh & Celik, Z. Berkay & Swami, Ananthram (2017). Practical Black-Box Attacks against Machine Learning.

[6] Carlini, Nicholas & Wagner, David (2017). Towards Evaluating the Robustness of Neural Networks.

[7] Yu, S. a. (2019). Concept drift detection and adaptation with hierarchical hypothesis testing. Journal of the Franklin Institute, 3187-3215.

[8] Bifet, Albert & Gavaldà, Ricard (2007). Learning from Time-Changing Data with Adaptive Windowing. Proceedings of the 7th SIAM International Conference on Data Mining.

[9] Brzezinski, Dariusz & Stefanowski, Jerzy (2014). Reacting to Different Types of Concept Drift: The Accuracy Updated Ensemble Algorithm. Neural Networks and Learning Systems, IEEE Transactions on. 25. 81-94.

[10] Du, Lei & Song, Qinbao & Zhu, Lei & Zhu, Xiaoyan (2014). A Selective Detector Ensemble for Concept Drift Detection. The Computer Journal.

[11] Athalye, A., & Carlini, N. (2018). On the Robustness of the CVPR 2018 White-Box Adversarial Example Defenses.

[12] Wong, A. a. (2020). How Much Can We Really Trust You? Towards Simple, Interpretable Trust Quantification Metrics for Deep Neural Networks. arXiv preprint arXiv:2009.05835.

[13] Cartella, F. a. (2021). Adversarial Attacks for Tabular Data: Application to Fraud Detection and Imbalanced Data. arXiv preprint arXiv:2101.08030.

Contacts

David Thogmartin
Director
AI & Data Analytics | AI Institute | aiStudio
dthogmartin@deloitte.de

Patrick Spitzer
Senior Consultant | Risk Advisory
AI & Data Analytics
pspitzer@deloitte.de

Jonas Körner
Consultant | Risk Advisory
AI & Data Analytics
jkoerner@deloitte.de

www.deloitte.com/de/aistudio

Deloitte.

Issue 06/2022