Model Risk Management in the Age of AI

A Primer for Risk Managers

Table of Contents

1. Introduction

Ever since the Office of the Comptroller of the Currency ("OCC") released its Risk Bulletin 2000-16 twenty years ago, the US banking industry has developed and refined a formal framework and organizational functions to identify, quantify, mitigate, and report model-related risks across all stages of a model's life-cycle.[1] Since 2000, the first major refinement of this framework occurred with the OCC's and Federal Reserve's 2011 release of broader and more detailed Supervisory Guidance on Model Risk Management[2] that fleshed out a more holistic three lines of defense framework[3], provided guidance on broader model governance expectations beyond independent model validation, and dove a bit deeper into the more technical aspects of model risk identification.

While most Banks have successfully evolved their model risk management programs over time to be consistent with the 2011 supervisory guidance, there are important questions as to whether these programs remain "fit for purpose" given the rapidly-evolving modeling revolution spearheaded by the rise of Machine Learning ("ML") and Artificial Intelligence ("AI") methodologies, and the proliferation of technical tools and platforms used therein. On the one hand, some practitioners would argue that – as a principles-based framework – existing supervisory guidance and current industry frameworks are sufficient to manage the model risks associated with these new modeling technologies. Alternatively, others argue that AI / ML models are sufficiently different from the "traditional" Bank models that another evolution of the model risk management framework is needed.

As a long-time model risk management practitioner, I propose that the existing model risk management framework continues to be "fit for purpose" from a programmatic perspective; however, it is also true that the rise of these new models and technologies changes a Bank's model risk landscape in fundamental and important ways that should be recognized and incorporated into further program refinements. I have written this document not as a short-form article presenting high level perspectives but, rather, as a more in-depth (and, therefore, lengthier) primer on the topic to focus on specific areas of novel risks and associated recommendations for program enhancements. Additionally, this primer is written for an audience of risk managers at medium to large size banking institutions – which includes not only those professionals developing and validating AI / ML models, but also other stakeholders within the three lines of defense for whom these model risks are relevant – such as model owners, model users, risk officers and committee members, and internal audit.

[1] A model's life-cycle typically involves the following stages: model design, model development, model deployment, and on-going model monitoring.

[2] See "Supervisory Guidance on Model Risk Management", OCC Bulletin 2011-12 and Federal Reserve Supervisory Letter 11-7.

[3] The frameworks' first line of defense involves controls deployed by model developers and model users to reduce / mitigate model risks by those who build and use models, the second line of defense involves independent model validation and other governance activities typically performed by the Bank's independent risk function, and the third line of defense involves an assessment of overall design and operating effectiveness of this framework by Internal Audit.

This primer starts at a macro level with **Section 2** introducing the key theme of algorithmic complexity and highlighting how this fundamental trait of AI / ML models and supporting technologies impacts a Bank's aggregate inherent model risk level and associated governance costs. Next, **Section 3**, focuses on core model validation activities and presents a detailed analysis for risk managers of the unique risks or validation testing challenges presented by AI / ML models in the areas of conceptual soundness, data inputs, technical soundness, and outcomes analysis, as well as specific recommendations related thereto. Finally, **Section 4** presents perspectives and recommendations related to certain additional AI / ML risk and governance dimensions – such as fairness, explainability, operational risks, and vendor models.

2. Managing Algorithmic Complexity at the Enterprise Level

The ten years since federal bank regulators issued their 2011 Supervisory Guidance on Model Risk Management has seen a steady expansion in the exploration and, in certain cases, deployment of AI / ML models within many large U.S. banks – with initial use cases focusing on managing fraud and money laundering risks, automating certain operational processes, providing new channels of customer interaction via chatbots and voice assistants, optimizing customer revenues, and managing credit risks. However, we are still largely in the early days of AI / ML adoption for several reasons – including organizational frictions, risk and regulatory concerns, a shortage of skilled resources, and – in some cases – a simple lack of inertia. Nonetheless, the potential returns from successful AI adoption are too great for these challenges to long persist and, accordingly, not only do I expect to see an expansion of Banks' model inventories based on new AI / ML use cases going forward, but I also expect we will observe a growing transition of existing Bank models to newer AI / ML methodologies.

At the enterprise level, this growth and concentration of AI / ML methodologies will transform the Bank's aggregate model risk profile in new and fundamental ways. For example, **Figure 1** below illustrates just a small sample of the vast and growing number of available AI / ML algorithms, the different technical platforms supporting these algorithms, the proliferation of freely available open-source tools and training code, and the expansion of publicly available datasets and pre-trained models that create an immense "superstore" of model development configurations from which a developer might now choose. Clearly, such choice can confer benefits to the enterprise. However, unbridled choice can also significantly increase the dimensionality of the skills and expertise required within a Bank to challenge effectively a given model's risks and limitations (the "governance tax")[4], as well as increase the Bank's inherent model risk level as developers freely use algorithms, open source code bases,

[4] I note that some of these choices may also be redundant and therefore provide little to no incremental benefit to the Bank – yet significant governance and support costs. For example, by allowing developers to choose multiple AI / ML code bases and platforms that perform the same computations, the Bank will need to ensure the corresponding skills and expertise exist across these multiple code bases and platforms in one or more downstream areas – such as model risk management, IT, and perhaps internal audit – to provide support for effective challenge and technical maintenance. Accordingly, more centralized control over these configurations – in addition to managing inherent aggregate model risk levels – can also lead to potentially large cost efficiencies for the Bank.

external datasets, pre-trained models, etc. with which they may have limited experience, and of which they may not fully understand the associated risks and limitations.

Figure 1: A Sample of AI / ML Technologies and Platforms

This suggests the need for Banks to re-assess and, likely, transform their model risk policies and other "guardrails" to ensure aggregate model risk levels continue to be managed according to the Bank's inherent risk and governance tax appetites. This governance refinement would require developers to operate within a set of approved technical configurations (e.g., algorithms, code bases, platforms, and pre-trained models) consistent with the Bank's risk and cost appetites – with a formal mechanism by which proposed new configurations could be evaluated and added over time, as appropriate.[5] Model risk management would have approval authority over these guardrails to promote consistency across the enterprise, and to ensure that approved configurations match the skills and expertise the Bank possesses for effective challenge and technical maintenance, or to identify skill and expertise gaps that it will need to fill.

[5] For example, an algorithm and technology carve-out could be recognized for approved research initiatives that promote innovation; however, the outputs of these initiatives should undergo a governance and approval process prior to formal Bank adoption to ensure appropriate net benefit levels are present, and sufficient resources will be available for governance and technical maintenance.

3. Refining Model Validation Skills and Procedures

Predictive models developed using AI / ML algorithms are markedly different than more traditional, statistically-based models commonly used by Banks such as linear and logistic regressions (i.e., "traditional Bank models"). In what follows, I discuss in more detail what I see as the core differences between these model types – as well as the challenges these differences may present to traditional model validation testing activities in the core model risk areas of data inputs, conceptual soundness, technical soundness, and outcomes analysis. I also suggest specific ways that risk managers can address these challenges within existing model validation frameworks.

3.1 Data Inputs

There are two important dimensions of AI / ML data inputs I wish to highlight here. First, AI / ML methodologies are typically applied to unconventional, non-structured data types – such as images, video, text, documents, and sound – to create models for computer vision and natural language processing ("NLP") tasks. These data types are highly specialized and very different from the traditional structured numeric data that form the basis of nearly all traditional Bank models. Accordingly, specific skills and experience are required to work with them properly. For example, the following presents just a sample of the unique requirements associated with image and text data inputs.

<u>Examples of Image Data Pre-Processing:</u>

- *Resize all images to a common set of dimensions (e.g., height and width)* – this may seem like a very straightforward task. However, how does one determine the common size dimensions? Are there certain use cases where larger sizes are important despite the significant impact on model size and compute requirements? What if resizing causes a change in the aspect ratio for some images? When shrinking images, should we just crop the original image? If so, what part of the image should be in the crop? What method should be used to stretch images to a larger size?

- *Translate images into appropriate mathematical representations* – AI / ML algorithms operate on numerical representations of data. This means that images must be disaggregated into individual color channels (typically, red, green, and blue) and the individual pixels for each color channel must be expressed along a mathematical scale (e.g., 0 to 255) that reflects their intensity.

- *Rescale pixel values through a standardization process* – most AI / ML algorithms require data inputs to be scaled between the values of 0 and 1.

- *Augment the image data through the introduction of sample images that have been modified in different ways* – an important risk and limitation of computer vision models is that real-world images fed into the model during deployment may deviate from the "clean" set of images used for model training and validation. Accordingly, generating and adding augmented images to the model development process – for example, image rotation, image noise, image shifting, image flipping /

mirroring, etc. – is an important part of image data pre-processing.

- *Label the images consistent with the intended use case* – this can involve overall image categorization, labeling specific object(s) within the image, specifying the mathematical coordinates of the center(s) of specific object(s), creating object bounding boxes, etc.

<u>Examples of Text Data Pre-Processing:</u>

- *Standardize the text data for input to the algorithm* – natural text data (such as e-mails, blog posts, social media posts, written documents, etc.) are simply blocks of alphanumeric characters and, accordingly, require a number of treatments to become useful model development data. In particular,

  - *Segment large blocks of texts into appropriate "tokens" based on the intended use case* – for example, segment sentences into individual word tokens, or segment words into individual character tokens. For sequence-based NLP models – such as transformers – the use of start-of-sequence ("SoS") and end-of-sequence ("EoS") tokens is also required to identify the sentence or sequence structure.

  - *Remove text that is irrelevant to the model use case* – typically, this involves removing e-mail addresses, embedded URL addresses, image captions, punctuation, special characters, and other textual data that are irrelevant to the modeling objective. For certain use cases, this step also includes the removal of "stop words" – that is, commonly-used words like "the", "a", "an", "is", "are", etc. that are extremely frequent yet of limited to no modeling value.

  - *Standardize and normalize tokens* – for example, converting capital letters into lower-case letters throughout the set of tokens. Additionally, replacing similar versions of words with a consistent token – such as replacing "carrying" and "carries" with the token "carry". Both standardization and normalization promote more consistent treatment of words that only differ by their specific grammatical usage.

- *Translate text into appropriate mathematical representations* – typically the text is disaggregated into word tokens (or, in some cases, character tokens) and each token is assigned a unique numerical value. These numerical values can be determined by a "bag of words" approach in which each token is assigned a unique numerical value, but where the specific numerical values across words have no meaning. Alternatively, a more advanced "embedding" process can be leveraged where the tokens are pre-processed using a separate AI / ML model (frequently an open-source pre-trained embedding model) that transforms them into multi-dimensional numerical representations that capture semantic similarities or relationships across different words or sequences.[6]

Clearly, as these examples demonstrate, specific expertise is required by both model developers – as well as risk managers – to work effectively and accurately with the types of unstructured data typically employed in computer vision and NLP models. This strongly suggests that Banks should assess and remediate potential skill gaps in these areas across the three lines of defense to ensure quality model development as well as effective challenge.

In addition to these specialized data types, I note that AI / ML models also frequently involve the use of very large datasets to train models – due either to the increasing availability of ever-larger "Big Data" datasets available both internally and externally – or, in the cases of computer vision, NLP, and general deep learning models, the necessity of very large datasets to produce a meaningful and robust neural network. In many cases, to take advantage of the additional dimensional depth provided by externally-sourced financial, socioeconomic, geographic, behavioral, attitudinal, transactional, and other consumer or business attributes, or to obtain sufficient unstructured data to train computer vision or NLP models, Banks are supplementing their internal data with large datasets obtained from either open-source repositories[7] or procured commercially from third-party data aggregators.

From a model validation perspective, testing data inputs is a standard procedure. However, for AI / ML models, as the above examples show, such testing requires much more specialized knowledge and experience – as well as different tools and approaches. Accordingly, when evaluating the model risks associated with data inputs, risk managers should consider the following incremental focus areas:

- *Was appropriate and sufficient due diligence performed by the developer on all externally-procured datasets – whether open-source or commercial?*

  While the ease of access to large open-source datasets has its benefits, it also creates potentially significant risks to the Bank that should be properly assessed and mitigated. Specifically, the developer should research and document the provenance or source(s) of the data, obtain a sufficiently detailed understanding of each data field that is used by the Bank, understand what type of quality control procedures were performed on the data by the owner, perform reasonable independent quality control procedures to assess potential issues and to implement appropriate remedial actions, and support the appropriateness of the data for the intended modeling objective. Additionally, the Bank should ensure sufficient legal and compliance reviews of the data are performed to include data rights (i.e., does the Bank have the right to use the data in the manner intended) and applicable regulatory compliance considerations – such as whether the data complies with privacy and other consumer protection laws and regulations.

[6] For example, Word2vec and GloVe for context-independent word-based embeddings, ELMO for context-sensitive word-based embeddings, and BERT for context-sensitive sub-word based embeddings.

[7] For example, the CIFAR and ImageNet libraries for images and the Wikipedia library and OpenWebText for text.

For commercially-procured datasets, similar due diligence procedures apply; however, the risk profile may be more complex as the data may have been aggregated from numerous sources, some of the data fields may be estimates produced by underlying models, other fields may be qualitative in nature based on consumer surveys and/or related segmentation analyses, and the vendor may be reluctant to provide certain information regarding the data due to its purported proprietary nature. In such cases, the risks and limitations of the data – relative to these due diligence expectations – should be documented and subject to appropriate risk governance.[8]

- *Was the data pre-processed appropriately and consistently with the requirements of the model architecture and the associated data type?*

  As noted above, AI models developed using image, video, text, or other unstructured data require specialized pre-processing. Risk managers should ensure that all appropriate and necessary steps in the data pipelines were implemented completely and accurately – including scaling, normalizations, augmentations, labeling, tokenization, embeddings, etc. Finally, I note that developers may employ certain types of data pre-processing to reduce the dimensionality of traditional structured data inputs prior to model training – such as principal components analysis ("PCA") – and all such pre-processing techniques should similarly be assessed for technical soundness, execution accuracy, and accurate flow through to production data pipelines by appropriately skilled personnel.

3.2 Conceptual Soundness

In general, the primary objective of AI / ML model development for supervised learning tasks is the maximization of a model's *overall predictive accuracy* on use cases involving either classification (i.e., the likelihood of one or more discrete events occurring or being true) or regression (i.e., predicting a continuous value of a certain numerical attribute) – with, in many cases, minimal attention devoted to the conceptual soundness of the underlying mathematical framework created by the algorithm to generate this predictive performance.[9] This de-emphasis of conceptual soundness is contrary to typical practice for traditional Bank econometric models wherein the statistical validity, statistical significance, and intuitive nature of each attribute's estimated relationship are typically considered as important as the model's overall predictive accuracy.

This dichotomy raises the question as to whether AI / ML models – because of their "black box" nature – require a conceptual soundness assessment under traditional model validation testing guidelines. I

[8] These limitations are also important to the Bank's ability to assess potential disparate impact or demographic bias in the models built from the data. See the **Section 4.4** for further details.

[9] As discussed in **Section 4.3,** increasing research efforts are being devoted to address the issue of model explainability / interpretability due to the "black box" nature of most AI / ML models. However, while interesting progress is being made, particularly in the area of credit decision models where explainability is a regulatory requirement, this is still a relatively nascent area.

address this question below by first highlighting the fundamental difference between traditional Bank "econometric models" and AI / ML black box models – specifically, the importance of variable causality in the former. I then summarize the challenges associated with a conceptual soundness assessment of AI / ML models and provide my recommendations for risk managers.

Econometric models are considered to be *"causal" models* – that is, the model's predictive variables (and functional forms) tend to be specified by the developer based on underlying scientific, behavioral, or economic theories applicable to these relationships. Accordingly, the empirical confirmation of these theories via the estimated functional forms, the values of the associated attribute weights, and the statistical significance of such weights tends to provide greater confidence in the model's overall robustness – separately and distinctly from the model's historical predictive accuracy – particularly for those predictions that may be truly out-of-sample or out-of-time. For example, within an econometric credit stress testing model, how macroeconomic factors specifically influence forecasts of the Bank's future credit losses may be very important to the use case and, therefore, the lack of conceptual soundness of such relationships could prove to be a significant model weakness regardless of the model's overall predictive performance. In this respect, conceptual soundness is a fundamental property of econometric models by their very nature.

On the other hand, proponents of AI. / ML models argue that, for some use cases, the de-emphasis of individual relationships relative to overall predictive accuracy may be warranted – particularly for certain low-risk use cases where the model's performance has been shown to be robust and stable across appropriate validation and testing samples, and where knowledge or intuition behind observed predictive variability is not needed. For example, how a model specifically leverages the pixels in an image to classify its contents may not be considered relevant to a user for a given use case – and likely has no clear intuitive foundation for which the benefit of a conceptual soundness evaluation would outweigh the associated costs.[10]

Furthermore, within many AI / ML models, it is very difficult to disentangle the estimated relationship of each individual attribute with the target variable for a number of reasons. First, as stated earlier, AI / ML models tend to be applied to very large datasets with dozens if not hundreds of individual data attributes – making evaluation of each individual estimated relationship potentially quite time-consuming. Second, AI / ML models tend to rely more heavily on complex, non-linear relationships between the data attributes and the target variable (i.e., the attribute the model is designed to predict) in order to provide increased predictive performance. Such "high dimensional" relationships among the data attributes tend to impede model explainability due to their complicated nature. Finally, a powerful feature of some AI / ML algorithms – such as neural networks – is their ability to perform

[10] I note that the foundation of this statement is currently the subject of significant active research. Specifically, novel approaches toward computer vision model explainability are beginning to emerge and I expect – with time – the costs associated with this validation activity will make a conceptual soundness assessment more feasible and potentially more valuable to users – to the extent that such activities reveal relevant model weaknesses and limitations that were previously unknown.

endogenous feature engineering. That is, unlike econometric models where the developer specifies the set of potential predictive attributes and their associated functional forms that will be considered during model development (collectively, "features"), some AI / ML algorithms actually derive these features as part of the model training process and "let the data speak for itself". Accordingly, because the developer no longer drives the feature specification, and because the model features are embedded within the model's complex mathematical architecture, there is no direct transparency of the model features – they must be inferred through time-consuming reverse-engineering.[11]

Overall, while AI / ML models certainly have their challenges when it comes to the evaluation of conceptual soundness, the lack of such testing certainly raises a model's risk profile along a number of dimensions (i.e., robustness, security, and fairness). Accordingly, risk managers should consider the following:

- *Is the explainability / interpretability of the selected model methodology consistent with the business requirements, the applicable use case, and the relative risk level of the model?*

  The above considerations suggest that: (1) the upfront choice of model methodology (e.g., econometric vs. AI / ML model) should consider the relative risk of the model for the specific use case and the corresponding need for a conceptually sound, explainable set of estimated relationships – in addition to requirements associated with overall predictive accuracy, and (2) these considerations should be embedded within appropriate Bank model development guidelines and subject to applicable governance and oversight.[12] For example, for bank stress testing purposes, it is likely that a model whose estimated macroeconomic sensitivities are consistent with applicable economic theory, are robust and stable across historical economic regimes, are intuitive relative to business / industry experience, and meet conventional measures of statistical validity are critically important requirements. In such a case, a black-box AI / ML model may not be the best choice of model methodology – even if such models produce higher rates of overall historical predictive accuracy.

  But what if such a high degree of attribute-level transparency is not considered critical for the use case? Does that mean that AI / ML models, under such conditions, do not require a conceptual soundness assessment to evaluate key drivers of model predictions? No. While such assessments

[11] While it is true that the data inputs are known, these are not necessarily the model's predictive features. Indeed, some of the data inputs may have very low contributions to the model's predictive performance, while other data inputs may contribute only indirectly through their combination with one or more other data inputs. Because the "higher level" features that ultimately drive the model's predictions may be quite complex combinations of multiple data inputs, they can be very difficult to interpret – thereby adding to the model's risk profile above and beyond the lack of conceptual soundness. For example, as discussed further in **Section 4**, such features could contain a hidden backdoor designed to manipulate the model's output, or could represent close proxies to a customer's race, ethnicity, gender, or age that may expose the Bank to fair lending risk.

[12] Indeed, the proposed model methodology should be evaluated and approved by the first line of defense during the initial model design stage before substantive development work has been undertaken.

may be challenging, they have the potential to uncover important and unexpected model risks and limitations of which model developers and users should be aware. Indeed, initial research into model explainability methodologies has revealed that some black box models may suffer from spurious predictive attributes, redundant data inputs, and other weaknesses and limitations that would not necessarily be discovered through other validation testing procedures (such as outcomes analysis).

This all suggests the following recommendations. First, if an econometric model is not required due to the specific use case and business requirements, then model developers should strive to include the minimum amount of complexity necessary into their AI / ML model to meet the model's business and technical requirements to minimize the technical challenges of a conceptual soundness assessment. Second, developers and validators should leverage the ever-growing set of tools and methodologies designed to aid model explainability to assess the presence of important model risks and limitations that may be hidden within the black box. And, finally, developers should consider whether an interpretable model architecture may be the right approach to address this issue and avail themselves of the evolving model architectures in this space. I discuss model explainability and interpretability further in **Section 4.3**.

3.3 Technical Soundness

From a technical perspective, AI / ML models are typically more computationally complex to develop than traditional Bank models as illustrated in **Figure 2** below:

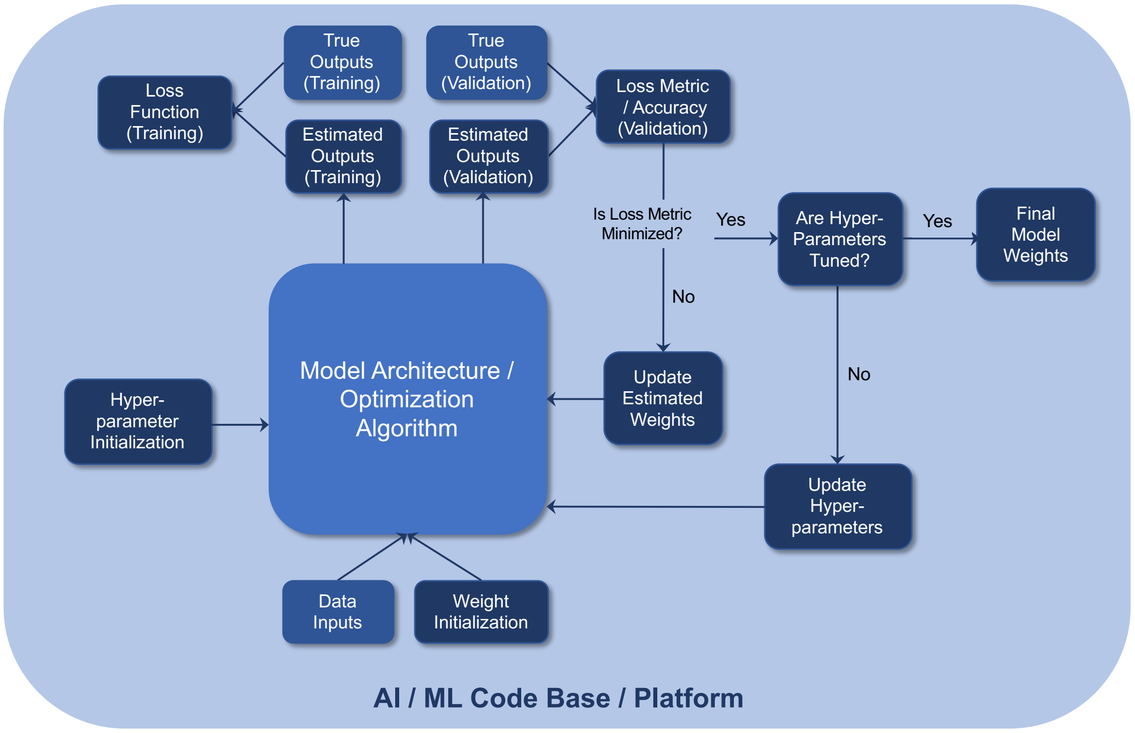
True Outputs (Training)
True Outputs (Validation)
Loss Function (Training)
Loss Metric / Accuracy (Validation)
Estimated Outputs (Training)
Estimated Outputs (Validation)
Is Loss Metric Minimized?
Yes
Are Hyper-Parameters Tuned?
Yes
Final Model Weights
No
No
Model Architecture / Optimization Algorithm
Hyper-parameter Initialization
Update Estimated Weights
Update Hyper-parameters
Data Inputs
Weight Initialization
AI / ML Code Base / Platform

Figure 2: Example of AI / ML Model Development Process

In general, the developer selects a specific AI / ML **model architecture** to produce model outputs (i.e., estimates or predictions) that are aligned to the business use case – given an available set of **data inputs** that are considered relevant to such predictions. The **weights** represent the unknown numerical values by which the data inputs are mathematically combined via the model architecture to produce the model output. These weights are "learned" by the computer (i.e., the "machine") through an iterative search process – the **optimization algorithm** – applied to the set of training data inputs. The objective of this search is to find the specific set of weights that minimizes a specific **loss function / loss metric** – that is, produces minimum predictive error or, conversely, maximum predictive accuracy on a separate set of validation data inputs to ensure that the model's predictive performance generalizes to data not used to train directly the model weights (i.e., "unseen" data). Once this optimization process ends, the model can be further enhanced through the tuning of certain **hyperparameters** – this is, certain features of the model architecture or optimization algorithm that may refine the model's specific mathematical structure or impact the execution of the iterative search process in manners that yield even greater predictive performance. Such hyperparameter tuning can either occur manually or via automated search and optimization methods. Once the hyperparameters are tuned, the process generates the final set of model weights and the model's performance is measured on a dataset completely unseen during model training and hyperparameter tuning (i.e., test data). All these steps are executed using one or more analytical **code bases** on a specific **AI / ML platform**.

This technical framework has several features that differ from those associated with traditional Bank models and, accordingly, model validation testing procedures and resources will need to be augmented by risk managers to address effectively these specialized features – as detailed further below.

3.3.1 Model Architecture

This represents the specific mathematical manner in which data inputs are transformed into model outputs (i.e., for supervised learning problems, either predictions of continuous values such as prices, or classification probabilities such as likelihood of future default). There are many different types of model architectures in the AI / ML world – including Support Vector Machines ("SVMs"), Random Forests, and Neural Networks – each of which takes a different mathematical approach to generating predictions from the data inputs and, therefore, each creating a unique optimization or "learning" problem to solve.

Since not all AI / ML architectures are applicable to every use case, it is important for risk managers to have a sufficient understanding of the broad base of architectures currently used in practice – as well as the important features, risks, and limitations of such architectures – in order to assess effectively model risk levels. For example, some architectures like neural networks may be difficult to train and require large amounts of data to truly unlock their predictive power. The use of such an approach by a relatively inexperienced developer, on a relatively small dataset, or for a use case that requires full transparency of estimated relationships should be recognized by the risk manager as potential red flags on which to focus validation testing.

Additionally, in the areas of computer vision and NLP, there exist a number of open-source pre-trained AI model architectures that can be leveraged via “transfer learning”. That is, given the significant training data requirements, compute resources, and technical proficiency required to build deep neural network architectures for computer vision applications (e.g., object detection) or NLP use cases (e.g., voice assistants), the bulk of such research typically occurs within large technology companies such as Google and Facebook who frequently provide public access to these models. While these pre-trained models may not match up exactly with the Bank’s intended uses, they can be fine-tuned by developers relatively easily for such purposes.[13] However, these benefits are not without risks and limitations and, accordingly, developers and risk managers should perform similar due diligence on pre-trained models as would be required for traditional vendor models – although I note that, unlike most vendor models, pre-trained models are much more transparent and testable due to their open-source nature.

In general, when evaluating the model risks associated with architecture, risk managers should consider the following focus areas:

- *Is the selected architecture aligned with the intended business use case(s) of the model?*

  Well specified business and technical requirements are crucial to assess the appropriateness (as well as the risks and limitations) of the selected model architecture. For example, is the model output an estimated value (regression) or a probability / likelihood (classification)? For NLP use cases, does the model need the ability to process variable length data inputs? Does the use case involve question answering, sentiment analysis, or named entity recognition? For computer vision, does the model need the ability to detect multiple objects within an image? For forecasting use cases, does the user require the ability to simulate accurately the effects of one or more independent variables? Given the variability in the strengths and weaknesses of various AI / ML architectures, it is imperative that developers and risk managers possess the breadth and depth of knowledge and experience: (1) to align appropriate architectures with the intended use case and the user’s business / technical requirements and (2) to identify potential risks and limitations (and significant inefficiencies) of selected architectures for potential remediation.

- *Is the selected architecture consistent with expected / needed predictive accuracy for each applicable use case?*

  There is a saying that you do not need a sledgehammer to crack a nut, and this sentiment applies very well to AI / ML modeling. While the more advanced model architectures such as deep neural

[13] The value in these pre-trained AI models lies in the lower- to mid-layers of their architectures where more general image features – such as edges and motifs – are created from a large corpus of general image training data. These pre-trained layers can then be used as accelerators to a Bank’s custom AI model architecture in which such layers are used to train higher-level image features used to predict the training data associated with the Bank’s specific use case. This transfer learning also applies to NLP use cases where pre-trained “embedding” models (i.e., complex mathematical representations of words incorporating semantic context) can be used as accelerators for customized Bank NLP use cases.

networks and complex ensembles are certainly powerful and academically interesting, they are likely inefficient from a cost and governance perspective for many use cases. For example, if a Naïve Bayes ML architecture generates 94% predictive accuracy on a low-stakes use case, then the incremental development, deployment, and governance costs of a deep neural network that generates 96% predictive accuracy on that same use case may not be justified. Alternatively, for a high-stakes use case where relatively small predictive errors can have large business impacts, more complex and costly model architectures may be justified. Going back to the theme in **Section 2**, Banks should proactively manage algorithmic complexity as part of its overall management of aggregate model risk and model governance costs.

- *Does the selected architecture meet user expectations for explainability and transparency?*

  As I discussed in **Section 3.2** above, many AI / ML architectures are highly complex with little to no transparency into how the various data inputs drive specific predictions. Accordingly, when users or use cases require a high degree of model transparency / explainability, risk managers should assess: (1) if the model architecture selected by the developer truly supports such requirements, and (2) whether the "explanations" of model behaviors offered by the developers are, in fact, complete, robust, and accurate. I discuss the area of model explainability / interpretability further in **Section 4.3**.

- *Is the selected architecture consistent with the technical model deployment specifications?*

  Given the high degree of compute variability and resource needs across different AI / ML model architectures, it is important for developers to consider the user's deployment needs and specifications within the overall model design process – for example: Is the selected architecture sufficiently and efficiently scalable for intended throughput? Will its complexity or memory requirements compromise its ability to meet required latency or throughput specifications?[14] Will all required data inputs be available in a production environment? Do the model's data input requirements and on-going production / maintenance costs align with the business's cost expectations?

- *Is the selected architecture consistent with the available training dataset size?*

  In general, more complex models tend to be characterized by a larger number of weights and higher model dimensionality – thereby requiring a greater number of training records for model development. For example, using a deep neural network architecture for a relatively small training dataset creates a significant risk of over-fitting (as well as a risk of under-identification – see **Section 3.3.3** below). Similarly, using a k-Nearest Neighbor architecture for a very high dimensional dataset (i.e., a large number of data inputs) may lead to poor model results due to

[14] Overly-complex models that take longer to generate and serve predictions – even by a second or less – can adversely impact business goals such as customer conversion rates, customer experience metrics, and operational efficiencies.

the greater distance between "neighbors" under such dimensionality. Risk managers should ensure that either the training dataset is an appropriate size for the selected model architecture, or that the model architecture is "right-sized" for the available training dataset size.

- *Are the assumptions underlying the architecture satisfied by the current model?*

  Certain AI / ML model architectures are based on specific assumptions or have other technical requirements. For example, the K-Means clustering algorithm assumes that clusters have spherical shapes, decision tree models assume that categories or "bins" of variable values are mutually exclusive, and Naïve Bayes models assume that predictive attributes are independent of each other. Risk managers should ensure that all relevant assumptions or requirements of a specific architecture are met or, if not explicitly met, that appropriate risk mitigants are in place.

- *Has the selected architecture been subject to sufficient independent professional scrutiny and evaluation to support its use outside of a research environment?*

  The AI / ML field is rapidly changing with the emergence of new architectures, new twists on existing architectures, and a healthy competitive environment across the industry in being on the leading edge of the field in both research and implementation. While these developments are exciting, and there is a natural desire to employ cutting-edge models, they also come with elevated risks due to the lack of a robust history of successful implementation. Accordingly, appropriate risk mitigation protocols should be established around the deployment of new innovative model architectures (ideally, at the model design or selection stage). This could involve requiring them: (1) to complete successfully an independent review in order to be added to an approved library of model architectures, or (2) be subject to inherent risk ratings that would guide required risk mitigants – such as limitations on use and/or enhanced model performance monitoring. Ultimately, Banks should ensure they have appropriate protocols to manage the risks (and costs) of AI / ML algorithmic complexity as discussed previously in **Section 2**.

- *If pre-trained models are leveraged, has appropriate and sufficient due diligence been performed by the developer?*

  While the ease of access to powerful pre-trained models – particularly for image recognition and natural language processing – has its benefits, it also creates potentially significant risks to the Bank that should be properly assessed and mitigated. Specifically, the developer should research and document the provenance or source(s) of the pre-trained model, obtain a sufficiently detailed understanding of the model architecture, the data used to build the model, the technical soundness of the model, the technical and data requirements to run the model, and the performance of the model on objective test datasets. Additionally, the Bank should support why the pre-trained model is appropriate for the intended business use case(s), how the pre-trained model will be fine-tuned or modified for the Bank's specific uses, and how this "hybrid" model will be internally validated for such uses. Finally, the Bank should ensure sufficient legal and compliance reviews of pre-trained models including intellectual property rights (i.e., does the Bank have the right to use the pre-trained model in the manner intended, and would the Bank have to provide the owner of the pre-trained models rights to any derivative works created by the Bank from the pre-trained model) as well as appropriate security reviews.[15]

3.3.2 Data Inputs

Section 3.1 covered the specialized risks associated with AI / ML data inputs; therefore, here I will only address the additional risks associated with specific dataset usage of which risk managers should be aware:

- *Is there sufficient training, validation, and test data relative to the typical requirements of the chosen model architecture?*

  As discussed previously, in general, the more complex the model architecture selected, the more data tends to be required. For example, if the developer selected a deep neural network architecture will millions of weights to be estimated (see **Section 3.3.3** for more information on weights), a very large dataset would be required to ensure reasonably robust estimates of such weights. Alternatively, if the developer is using a simpler logistic regression architecture with 20 predictive variables, then the dataset size requirements would be much less due to the significantly lower number of estimated weights and the greater density of the development data.[16] Indeed, risk managers should pay particular attention to models where the number of training records is low relative to the number of parameters or weights to estimate. Such situations may result in the "under-identification" of optimal weight estimates – which means that there may not exist a single optimal solution to the optimization problem but, rather, many solutions that yield the same loss value (see **Section 3.3.4**). In such cases, which of the multiple optimal solutions is selected can be a random outcome and, yet, have significant impacts on the model's predictive performance on unseen data.[17]

- *In cases where the developer is using very large numbers of data inputs (i.e., a large number of variables), how is the developer mitigating the risks of redundancy[18], noise, and over-fitting?*

[15] As discussed further in **Section 4.2**, unscrupulous developers or hackers may seed certain AI / ML model architectures and weights with hidden functionality (i.e., a "key") that permits manipulation of the model's predictions.

[16] In this context, "density" refers to the number of dataset records relative to the number of weight estimates. In general, a "denser" dataset can provide benefits to model training since the weight estimates tend to be based on a larger number of data points – all else equal.

[17] For a more in-depth discussion, see D'Amour, A., et al., "Underspecification Presents Challenges for Credibility in Modern Machine Learning", arXiv pre-print arXiv:2011.03395, 2020.

[18] See **Section 3.3.3** for a further discussion of variable redundancy and its associated costs.

It is becoming increasingly common to hear about AI / ML models that are using hundreds or thousands of variables to generate their predictions. To a certain extent, there is an impression among some practitioners that the ability to include more and more data inputs is a key benefit of machine learning as it provides the algorithm with a much broader information set from which to discover predictive features and therefore create better (i.e., more predictive) models. What's typically left unsaid, however, are the risks and costs to the Bank of such large data input requirements. In particular, many of these inputs may provide no meaningful incremental predictive power to the model, there may be many redundancies across similar inputs, and the presence of these "low value" inputs typically creates excessive "dimensionality" of the data that can lead to over-fitting – as well as excessive data acquisition and governance costs that can harm project ROI.

With respect to the excessive dimensionality point, consider a dataset with two inputs where the first input can take on 4 different values and the second input can take on 10 different values. Theoretically, there are 4 x 10 = 40 different input profiles that could exist across the dataset. If we had 10,000 records in our dataset, then – on average – we would have 250 records per input profile (assuming they were uniformly distributed). If we added another data input to our dataset that also takes on 10 different values, then the potential number of input profiles (i.e., the dimensionality of the data) increases to 4 x 10 x 10 = 400 – yielding an average of 25 records per input profile. As this example demonstrates, the more data inputs we use, the more granular the dataset input profiles become, and the more dissimilar individual records can be from each other.[19] This can lead to a situation where the dimensionality becomes so large – relative to the number of records – that rather than adding useful information, we risk adding noise because we have a meaningful number of input profiles that may be very sparsely populated. Such excessive dimensionality may result in the over-fitting of the training data as well as less reliable model predictions on new unseen data that may fall into the sparse profile regions.

3.3.3 Weights

This term refers to a set of numerical values that are applied to the data inputs to generate the model outputs (i.e., the predictions). For example, in the simplest example, the model output $\hat{Y}$ is a linear function of the data input X as follows:

$$\hat{Y} = a + b \cdot X$$

where a and b, collectively, are the fundamental weight values for which the optimization problem is

[19] This example focused on discrete or categorical variables (i.e., those that take on a limited number of values). The presence of continuous variables increases the risks even faster due to the significantly greater number of potential profiles they create.

solving.[20] The machine learning process will iteratively update these estimated weights until it finds the specific values that minimize the loss function (i.e., the lowest predictive error between the model's estimated target values, $\hat{Y}$, and the true target values, Y – see **Section 3.3.4** below for further details on the optimization problem's loss function).

While an evaluation of model weights has always been a part of traditional Bank model validation procedures, AI / ML models create a new set of considerations that risk managers should consider:

- *Were weight values initialized appropriately (where applicable)?*

  Certain model architectures, such as neural networks, require the developer to specify a starting (or initial) set of weights to seed the optimization algorithm. Such weight initializations can have a significant impact on the speed and success of the optimization process – with different initialization approaches being associated with different model architecture features.[21] While a grossly incorrect initial set of weights could likely produce a failed training outcome (that, hopefully, would not be submitted for model validation), knowledge of weight initialization best practices is still important to help validators diagnose instability in the optimization process (for example, during the validator's model replication or testing) and to provide useful recommendations to model development teams on training speed and efficiency enhancements.

- *Does the final set of weights represent a unique solution to the optimization problem?*

  As will be discussed further below in **Section 3.3.5**, stochastic optimization algorithms are typically used to find the optimal set of weights for many AI / ML model architectures. However, under certain conditions, the randomness in the optimization process may yield different optimal weight solutions; that is, if we were to re-train the model multiple times on the same data inputs, but with a different initial random seed, we may get somewhat different sets of final weights. This is particularly likely when the model architecture is "under-identified" which can occur when the number of weights to be estimated exceeds the number of data points in our training dataset.[22]

  The relevant risk here is that – instead of a single optimal set of weights – there may, in fact, be multiple sets of weights that generate the same or very similar predictive error. While, on the one

[20] Technically, b is called the weight and a is called the bias term. However, I will refer to them collectively as "weights" for convenience and without sacrificing narrative accuracy.

[21] This is primarily encountered in neural network architectures where the initialization approach depends on the type of activation function used within the network structure and whether other architectural features are present – such as batch normalization.

[22] This is not an issue with the stochastic optimization algorithm itself; rather, it is the interaction of the specific mathematical structure of the chosen model architecture (typically driven by the user-specified hyperparameters – see **Section 3.3.6**) with the size of the training dataset that results in a non-unique solution for the weights – i.e., multiple sets of weight values may correspond to the minimum predictive loss value. This risk ties into the discussion in **Section 3.3.2** and Footnote 17.

hand, one might view this as inconsequential since model performance is the same, I note that this is only true for the training data. In fact, it may be the case that each set of weights generates very different predictive errors on a true out-of-sample or out-of-time dataset (aka a "drifted" dataset) and, therefore, which set of weights is ultimately selected becomes much more important. For risk managers, this highlights the importance of dataset size relative to model complexity, as well as model robustness testing (see **Section 3.3.5**) and appropriately designed outcomes analyses (see **Section 3.4**) to assess the risk of multiple optimal solutions and the corresponding stability of the model's final weights on true out-of-sample data.

- *Can the weights help in the assessment of the model's conceptual soundness or explainability?*

  In the simple example above, there is a linear relationship between the data input X and the model's predictive output $\hat{Y}$ with the weights a and b governing the nature of that relationship (i.e., its direction and strength). We can easily assess the conceptual soundness of this relationship through the estimated value of b – is it positive or negative? How quantitatively sensitive is $\hat{Y}$ to a given change in X (i.e., b in this example)? Are both results consistent with business intuition, economic theory, and similar relationships estimated in other industry models?

  I note, however, that this conceptual soundness assessment becomes much more challenging when the number of model weights grows larger and when estimated relationships between model inputs and outputs incorporate non-linearities (i.e., when the sensitivity of $\hat{Y}$ to a given change in X is also dependent on the values of other data inputs). As noted previously in **Section 3.2**, it is common for many AI / ML model architectures to rely on complex non-linear interactions among data inputs, and to leverage large numbers of such data inputs – both of which impede the transparency and explainability of the model.[23] Additionally, unlike traditional econometric models where one can directly assess the statistical significance of estimated relationships and, therefore, prune the model of irrelevant factors, such a practice is largely absent from most AI / ML model development processes due to a lack of statistical significance measurements in most model architectures as well as current model development behaviors that tend not to prioritize this activity.[24]

  To address this risk, risk managers should leverage a growing inventory of tools and techniques to gain further insight to key model drivers and to help identify irrelevant data inputs that could be pruned. For example, some ML architectures have hyperparameters (see **Section 3.3.6** below) that will automatically implement certain pruning strategies to remove irrelevant or redundant model weights. Additionally, some deep neural network model architectures can be decomposed using linear modeling methods to provide risk managers important insights into the nature and

[23] In fact, the number of weights used in state-of-the-art NLP models is now in the billions.

[24] While the presence of irrelevant factors does not, in general, impair the model's overall predictive performance in the presence of appropriate validation set testing, they do create indirect costs for the Bank – such as incremental data acquisition and governance costs.

statistical significance of variable-specific relationships, variable importance levels, regions of over-fitting or redundancy, and opportunities for model simplification.[25] Finally, there are methodologies and code libraries available to visualize the key drivers of image-based convolutional neural networks and AI models developed for natural language processing.

For Banks for whom AI / ML models are expanding in usage, it is important that both model developers and model risk managers build up new capabilities and resources to address these traditional model risk issues in a more complex analytical environment.

3.3.4 Loss Function / Loss Metric

The loss function represents the objective of the optimization or learning process – that is, how do we determine which set of weights is the "best" or "optimal" solution? Typically for AI / ML models, this objective is defined using a loss function where "loss" represents a mathematical measure of the model's predictive error – such as mean-squared error for continuous outcomes (e.g., estimated prices or values) or cross-entropy loss for classification problems (e.g., an event likelihood such as a loan default). Accordingly, the optimization algorithm seeks to find the set of weights that minimizes the loss function (or, equivalently, maximizes predictive accuracy).

As shown in **Figure 2**, a loss function / loss metric is applied to both the model training data[26] – that is, the data inputs used directly by the optimization algorithm to derive the weights, as well as the validation data which is used to evaluate whether the model's predictive performance – for a given set of weights – is appropriately *generalizable* to similar "unseen" data that was not used for model training. The latter represents a separate randomly-selected portion of the model development data that is withheld from the training algorithm and, since such data is "unseen" by the model, it provides a preferred data sample over which to evaluate the model's predictive error / accuracy. It is also the dataset that is used to tune the hyperparameters (discussed in **Section 3.3.6** below).[27]

In traditional Bank model development platforms for linear regression, logistic regression, and time series analysis, the loss function is typically standardized and lies out of reach from the model developer – that is, it is embedded into the underlying source code of the model estimation platform. Accordingly, there tends to be very low risk that the developer is specifying an inappropriate loss function for the selected model architecture, or is using a novel loss metric that may inadvertently suppress important

[25] See, for example, Sudjianto, Agus, et. al., "Unwrapping The Black Box of Deep ReLU Networks: Interpretability, Diagnostics, and Simplification", arXiv pre-print arXiv: 2011.04041, 2020.

[26] Technically, a loss *function* is applied to the training data and is used in the learning process. A loss *metric* is used to measure model performance on the validation and test datasets. The former has more desirable mathematical properties that facilitate the optimization process, while the latter is more intuitive, practical, and more easily interpreted by users.

[27] A third dataset (not shown in **Figure 2**) is the "Test set". Because the hyperparameters are tuned using the validation dataset, the test set represents a final "unseen" dataset used to measure objectively the final model's predictive error / accuracy.

model risks and limitations.

For many AI / ML model architectures, however, the code bases and platforms typically permit full developer access to the loss function and loss metrics – with different pre-populated choices from which to select, as well as the ability to create and use completely custom loss functions and metrics – the former of which may be employed to achieve a more complex mix of optimization objectives such as low predictive error and few predictive variables, or low predictive error and no demographic bias. Given this flexibility available to the developer, and the important impacts it has on the final model results, risk managers should consider:

- *Is the loss function and loss metric used clearly documented by the developer?*
- *Is the loss function and loss metric used appropriate for the business and technical specifications governing the model development effort?*
- *Has the loss function and loss metric used been implemented accurately?*
- *In cases where the loss function has multiple objectives, have their individual impacts been measured*[28]*, and have these impacts – along with all other relevant risks and limitations – been clearly communicated to model users?*

3.3.5 Optimization Algorithm

This represents the computer program or algorithm by which potential solutions to the model architecture (i.e., the model weights) are iteratively generated, evaluated for optimality (i.e., lowest measured loss), and revised accordingly. Fundamentally, this is really what the term "machine learning" refers to – it is the process whereby the machine (i.e., the computer) learns the best solution to the optimization problem through an iterative "test and revise" computational process.

Unlike traditional Bank models such as linear and logistic regression whose architectures are standardized and whose solutions are easily, quickly, and directly computed, many AI / ML models have more complex architectures whose solutions require a more computationally-intensive iterative search or "learning" process to solve. For example, **Figures 3 and 4** below illustrate one of the primary challenges of solving a complex AI / ML model's optimization problem. In both figures, the vertical axis represents the training loss (or predictive error) associated with various potential solutions

[28] This is particularly important when the objectives are *competing* and, therefore, a tradeoff exists between the optimization of both objectives. For example, it may be the case that the second objective (no demographic bias) can only be achieved by accepting a higher than optimum level of overall predictive error. In such cases, the developer should quantify how much overall predictive accuracy was sacrificed to meet the secondary objective. Even better, in the presence of competing objectives, the developer should measure and communicate the *trade-off curve* to users so they can understand the incremental costs of achieving a certain improvement in one objective at the cost of the other. Good model governance would ensure that appropriate stakeholders are involved in selecting the specific mix of the two objectives (i.e., the specific point on the trade-off curve) to which to calibrate the final model weights.

to the optimization problem (i.e., the estimated weights which are captured on the horizontal axis). For traditional models such as linear and logistic regression, the loss functions are fairly standardized and have the smooth shape illustrated in **Figure 3**. With this type of loss function, you can see that the solution to the optimization problem is quite easy to derive, well-defined, and unique. However, for complex AI / ML architectures, the loss function may be much more complex – as illustrated in **Figure 4**.

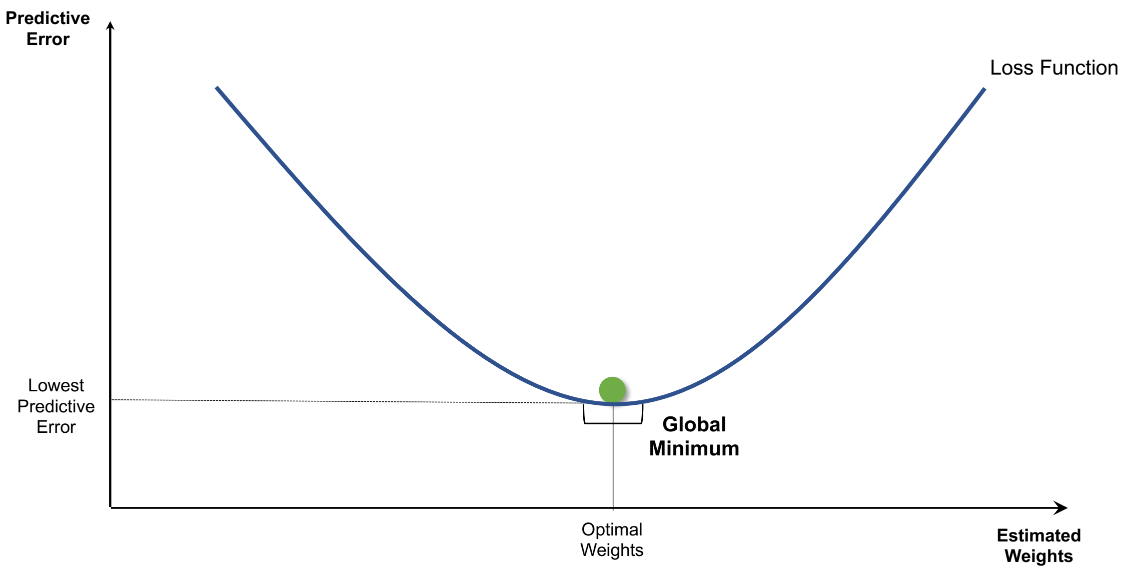

Figure 3: Illustration of Loss Function and Optimal Weights for Traditional Models

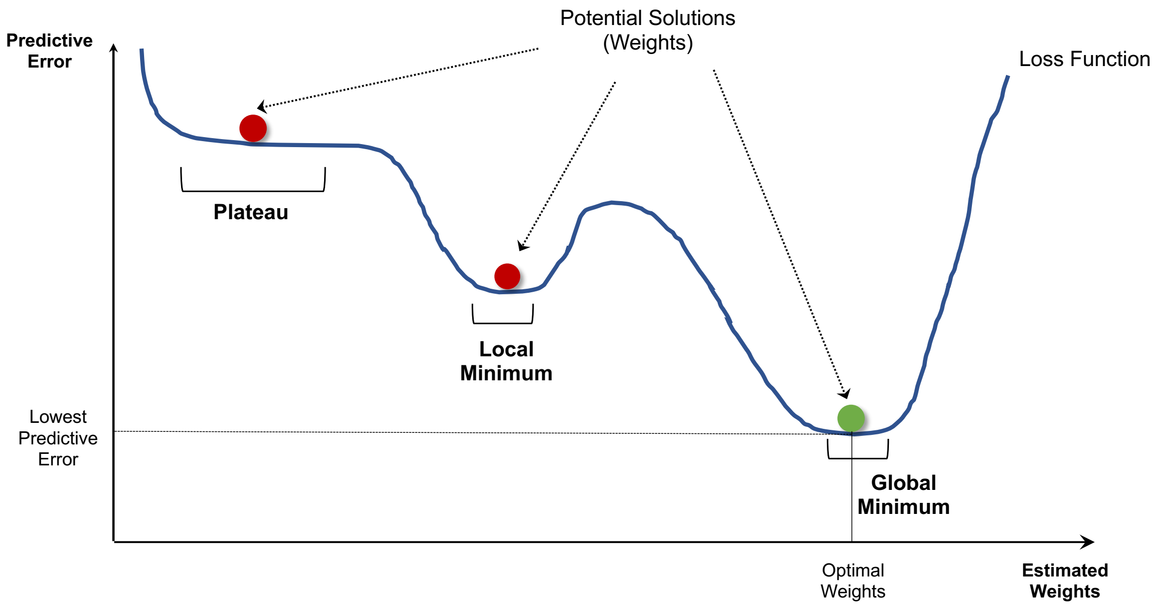

Figure 4: Illustration of Loss Function and Potential Solutions for Complex AI / ML Models

Here you can see that the loss function is not universally smooth and has complex features such as plateaus (relatively flat areas of the loss function where relatively large changes in the weight values have little effect on the estimated predictive error) and local minima (localized areas of the loss function that may appear to be the optimal solution since estimated predictive errors increase as the weights increase and decrease around this potential solution). In a machine learning context where an iterative search algorithm is used to find the optimal solution, these loss function complexities – which are generally unknown to the model developer ahead of time – may result in a final solution that is not the global optimum. Instead, the algorithm may get stuck on a plateau and assume it has found the optimum since it can no longer reduce estimated predictive error even for relatively large weight changes, or the algorithm may assume that a local minimum is really the global minimum since deviations away from this point in both directions lead to higher estimated predictive errors.

In addition to a more complex loss function, the machine learning process is further complicated through the introduction of a random component to the optimization algorithm. That is, unlike traditional Bank model estimation processes that either ingest the entire model training dataset or rely on a fixed random sample of this dataset, most AI / ML algorithms iterate over randomly-selected subsets of the model training data during the optimization process to estimate the model weights. By introducing this random component to the optimization process, not only is the amount of time required for model training reduced (particularly for very large training datasets), but the random noise introduced into the search process through the use of randomly-selected "mini-batches" of training data can also help avoid the two types of "traps" noted above (i.e., plateaus and local minima) by causing the algorithm to jump out of or past these traps.[29] I note, however, that such randomness comes at a cost as it can impede model validation and transparency by preventing model reproducibility and auditability. Specifically, without use and retention of a specific random "seed" for model training, it may be impossible to replicate and validate the developer's estimation results. This is a unique risk for AI / ML models (vs. traditional Bank models), but one whose mitigation is easily addressed.

In general, when evaluating the model risks associated with the optimization algorithm, risk managers should consider the following focus areas:

- *Has the loss function been configured appropriately for the selected model architecture, the model outcome, and applicable business specifications?*

  As noted in **Section 3.3.4** above, is the loss function aligned to the model outcome and consistent with loss functions typically used for this model architecture? If an atypical loss function is used,

[29] Training time is decreased primarily because the use of mini-batches reduces the number of redundant observations used for training. For example, if 20% of the training data is redundant (i.e., it is identical to other observations in the training dataset), then such data is not impactful to the algorithm's solution – but simply extends the computational time required to reach that solution. Additionally, the use of mini-batches permits more efficient use of specialized AI / ML hardware such as GPUs (graphics processing units) – thereby further decreasing computational time.

has the developer sufficiently documented its structure, supported its use both conceptually and technically, and implemented it in a technically accurate manner?

- *Has the optimization algorithm been configured appropriately for the selected model architecture, the model outcome, and applicable business specifications?*

  Besides the loss function, other relevant features of the optimization algorithm to assess include the size of the mini-batches, the number of epochs used for model training, and the learning rate. As noted previously, AI / ML optimization algorithms typically leverage random subsets of the training data (i.e., mini-batches) to learn the optimal weights. The size of these mini-batches should be consistent with applicable professional guidance and benchmarks – which may vary based on the model architecture as well as certain features of the training data. An "epoch" refers to a complete pass of the optimization algorithm through all of the training data (e.g., a training dataset containing 100,000 observations with a mini-batch size of 100 will, in general, complete an epoch after processing 1,000 mini-batches). In general, an optimization algorithm requires several epochs of training in order to find the optimal solution – however, the required number of epochs in not known ahead of time and, therefore, must be explored as part of the model training process. The "learning rate" represents the magnitude by which the estimated weights are updated during each iteration of the optimization algorithm (typically for each mini-batch). As will be discussed further in **Section 3.3.6** below, the learning rate is a critical hyperparameter in model training.

- *Is there sufficient proof of convergence to a global optimum solution?*

  As illustrated in **Figure 4** above, certain AI / ML architectures may possess loss functions with complex mathematical features (e.g., plateaus and local minima) that can impede the algorithm's search for the globally-optimal weight solution. Additionally, optimization algorithms typically continue to iterate through the training data, and revise the estimated weights, until either it reaches the end of the pre-specified number of training epochs or it triggers pre-specified "early stopping" criteria (such as failure to reduce the training loss by a certain amount over the past X epochs). Both hyperparameters can have a significant impact on whether the algorithm converges to the global optimum – for example, specifying too few epochs or an easily-attainable early stopping condition can result in premature termination of the learning process and sub-optimal weight values. Accordingly, risk managers should assess whether the developer has performed sufficient hyperparameter tuning and validation testing to provide confidence that the final set of weights are, indeed, globally optimal.

- *Are the model results reproducible?*

  As noted previously, the developer should employ and document the specific random seeds used in the optimization algorithm to promote reproducibility and auditability of the model estimation results. Along with other technical documentation associated with the model architecture, loss function, hyperparameters, and final model weights, risk managers can then use these seed values to confirm the model estimation results and to perform additional model validation testing activities.

- *Has there been sufficient testing of potential over- or under-fitting to the training data via validation and test samples?*

One of the key risks of AI / ML models is the potential to either over-fit or under-fit the training data. Over-fitting occurs when the model captures nuances of the training data that are merely statistical noise and, therefore, provide a false sense of model predictive ability on unseen data. For example, **Figure 5** illustrates over-fitting with respect to an AI / ML model designed to classify one group from another (i.e., red from blue in this case) using two data inputs (X_1 and X_2). The smooth black line represents a typical robust non-linear solution which, while not perfect in completely separating the two groups, leads to high predictive accuracy and should be generalizable on unseen data. Alternatively, the erratic green line represents an over-fit solution – where the model develops a highly complex solution in order to fit all of the training data. In this case, it is unlikely that this solution will generalize well on unseen data since it was fit to the specific statistical noise in the training dataset – thereby providing a false sense of better predictive performance than the first solution. Such over-fitting occurs when the developer adopts an overly-complex model architecture or when training time is excessive.

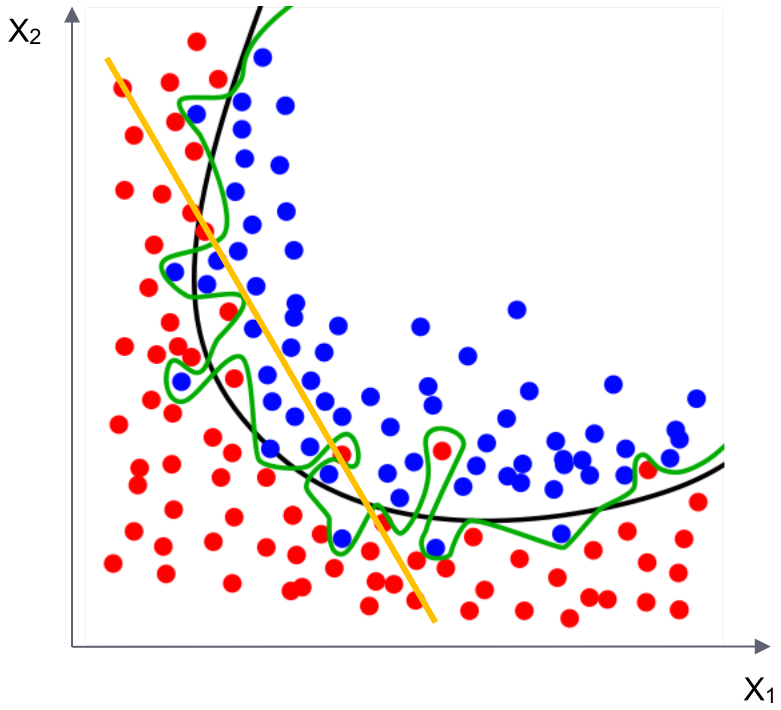

Figure 5: Example of Over-fitting (Green) and Under-fitting (Orange)[30]

Finally, the orange line in **Figure 5** is an example of under-fitting whereby the solution is too

[30] Image credit: https://commons.wikimedia.org/wiki/File:Overfitting.svg. Orange line and axis labels added by author.

simple (in this case a simple linear relationship) that fails to capture the presence of systematic non-linearities in the relationship between X_1 and X_2 relative to the two outcomes. In this case, the model will exhibit relatively low predictive performance on the training dataset and perhaps even lower performance on unseen data. Such under-fitting can occur when the developer adopts a too-simplistic model architecture or when there is insufficient training data to capture effectively the systematic non-linearities.

Typically, developers should use a separate validation sub-set of the training dataset to evaluate how a given model solution performs on unseen data, and many AI / ML platforms produce visualizations of certain model performance metrics on both the training data and validation data that permit relatively clear diagnosis of these potential issues.[31]

- *If multiple optimization objectives were specified, was there appropriate governance over the specification of these objectives, as well as over the specific point on the trade-off curve that was selected?*

  As noted previously, in the presence of multiple competing objectives in the loss function, it is important that appropriate transparency and governance was provided to arrive at the final solution.

3.3.6 Hyperparameters

This term represents certain developer-specified settings or values related to the optimization algorithm that impact the ultimate solution obtained. For example, a common hyperparameter for certain AI / ML models is the "learning rate" – a value that impacts the magnitude by which the optimization algorithm updates estimated weight values during each iteration in order to achieve a further reduction in the predictive error. This is illustrated in **Figure 6** below.

[31] The final model performance metrics – after hyperparameter tuning – that are generated on the test sub-set of data can also be leveraged to identify potential over-fitting or under-fitting issues.

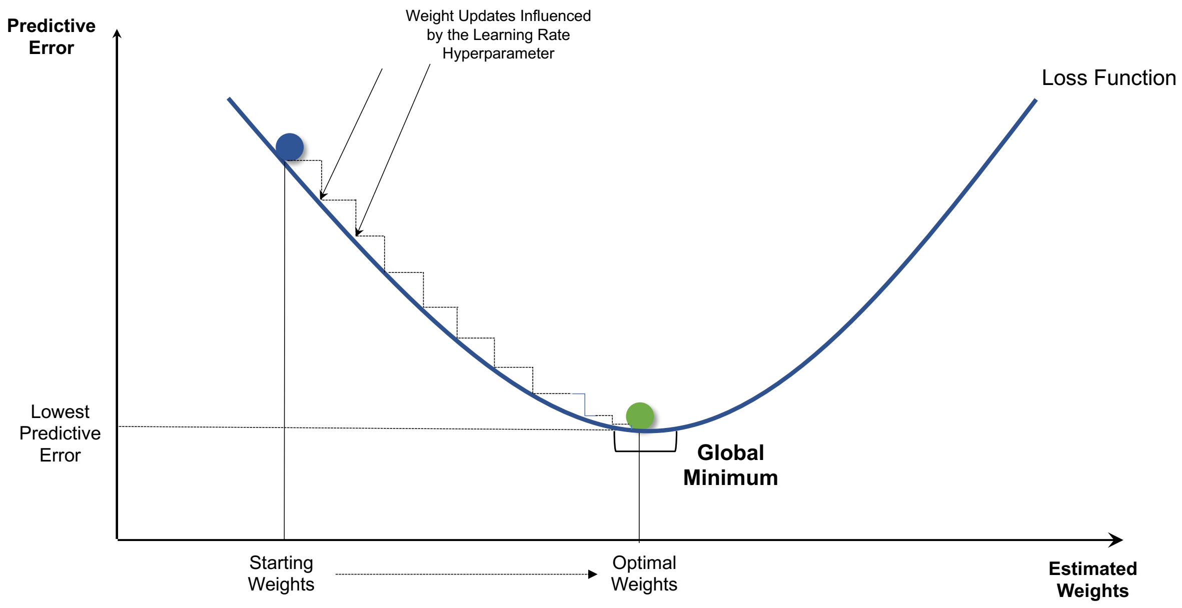

Figure 6: Example of Learning Rate Hyperparameter

Here, the blue ball represents the initial starting weights and the series of steps represents the machine "learning" process. Specifically, since the slope of the loss function is negative at the initial weight point, the algorithm knows that it needs to increase the estimated weight values in order to decrease the loss function value and, therefore, get closer to its global minimum (the green ball). But what is unknown is the amount by which to increase the weight value. That is where the learning rate comes into play. The learning rate – which is a hyperparameter value set by the developer – determines the size of each step down the loss function. Small learning rate values will move steadily toward the solution but will take longer from a compute standpoint – creating risks of over-fitting or, for complex model architectures, getting stuck on a plateau or in a local minimum. Alternatively, larger learning rate values will move more quickly to the solution, but risk overshooting the minimum, bouncing around the minimum, or even diverging (i.e., moving up the loss curve) if too large in value. Testing different learning rate values, or using more dynamic learning rate optimizers, are choices made by the developer that will impact the quality of the solution derived by the algorithm.

Other examples of hyperparameters are:

- For the **Random Forest Architecture** – the number of decision trees in the forest, the maximum depth of each decision tree, and the minimum size of each leaf – among others.

- For the **Support Vector Machine ("SVM") Architecture** – the "C" hyperparameter that reflects the relative tradeoff between the estimated margin width and the associated degree of training data misclassifications.

- For the **Neural Network Architecture** – the mini-batch size, the number of hidden layers, the number of neurons per layer, the activation functions, the weight initialization method, and many others.

For each hyperparameter, the developer typically selects an initial value at the start of the model training process and then, to finalize the model, explores the impact of different hyperparameter values on the validation loss metric – generally settling on the hyperparameter value that generates the lowest predictive error / highest predictive accuracy.[32] Besides this manual trial and error process, there are also certain automated search processes that find more efficiently the best set of hyperparameter values (e.g., the configuration that minimizes the loss metric on the validation sample).[33]

In general, when evaluating the model risks associated with hyperparameters, risk managers should consider the following focus areas:

- *Have the developers fully and appropriately documented the hyperparameters employed – along with sufficient rationale for their values – particularly if one or more values differ materially from typical professional practice, approved internal ranges, or established benchmarks?*

  It is common for AI / ML code bases to have "default values" of many hyperparameters pre-specified for ease of use. While these default values tend to be set at "typical values", it would not be prudent for a developer to simply use such values for this reason. Each modeling exercise can have its own unique features and objectives that may cause such default values to be sub-optimal from either a predictive accuracy perspective or by creating excessive algorithmic complexity – although they may represent appropriate starting points of the hyperparameter tuning process. An alternative to documenting the rationale for all hyperparameter values is for the associated analytical group to maintain such documentation for rarely changed hyperparameters. In this case, the developer would still document the complete set of hyperparameter values used; however, (1) only those hyperparameter values that differ from the group's documented standard values would need a specific documented rationale by the developer and (2) the developer should formally acknowledge that she reviewed and concurred that the group's standard values were appropriate for the specific model that she developed.[34]

[32] There is a risk of over-fitting the hyperparameters to the validation dataset, and the developer should mitigate against this risk. I note that the test dataset may indicate the presence of such over-fitting if its loss metric value is materially lower than the loss metric value on the validation dataset.

[33] Because hyperparameter tuning requires the core model architecture to be re-trained multiple times on the development data (using different hyperparameter values each time), this process is very compute-intensive and, potentially, costly for models based on large datasets (e.g., deep neural network models). Model risk management can provide additional value-add to the Bank by promulgating best practices in hyperparameter tuning across the Bank's modeling groups.

[34] I note that this recommendation relies on an appropriate governance structure within the analytical group to specify the population of these hyperparameter values for each AI / ML model architecture, document the group's approved standard values and the rationale for such, make this documentation freely available to group members and the

- *Are the final hyperparameter values supported by sufficient analytical rigor and documented analytical evidence?*

  Given the background discussed above, particularly, (1) the important impacts hyperparameter values can have on the final algorithmic solutions and (2) the need to explore sufficiently the range of potential hyperparameter values to determine the best or acceptable values – risk managers should ensure that the methodology or approach used by the developer to arrive at the final hyperparameter values is technically sound and clearly documented, the basis for selecting the final hyperparameter values is clear (e.g., highest predictive accuracy on the validation sample), and appropriate and sufficient analytical evidence has been retained to support the developer's methods and choices.

- *Has the developer measured the final model's predictive performance (after hyperparameter tuning) on a completely separate unseen "test" data sample?*

  As noted previously, the validation data sample is used to tune the hyperparameters. Accordingly, a third unseen data sample (the "test" data sample) is required to provide an objective measure of the final model's predictive performance. Risk managers should ensure that such a sample was used, it was appropriately configured for this purpose, final model performance metrics were accurately calculated on it, and that it is representative of the production data to which the final model will eventually be applied.

3.3.7 AI / ML Code Base & Platform

This represents the software environment and computational platform used by the model developer to perform the model development process. Currently, there are many popular software platforms and tools used for AI / ML – including Python, R, TensorFlow / Keras, PyTorch, fastai, H20.ai, DataRobot, AWS SageMaker, and many others. Many of these are available free as open source code / tools with frequently released updates and expansions.

The proliferation of, and easy access to, these tools can quickly create unneeded complexity and elevated governance costs for Banks if no guardrails are in place. At a minimum, the use of these tools should be subject to the Bank's overall policies on open source software, and strong consideration should be given to the maintenance of an approved library of tools for use across the Bank that promotes consistency, mitigates the risk of using open source tools that may not have the same degree of industry testing, validation, and security as other more mainstream tools, reduces the need for specialized training and resources to govern many overlapping tools, increases organizational depth of expertise, and reduces key person dependencies associated with more esoteric open source tools.

Bank's model risk management function, periodically update this information as needed, and consider an annual re-certification of the document to ensure its on-going completeness, accuracy, and validity.

3.4 Outcomes Analysis

Outcomes analysis is an important and standard part of model validation testing, and typically involves assessments of model performance in the following areas:

- *In-sample model predictive performance* – evaluation of the loss metric (typically a measure of predictive accuracy) on the model training data to assess the model's predictive performance on "seen" data.

- *Out-of-sample predictive performance* – evaluation of the loss metric on a separate randomly-selected "unseen" validation dataset that was not used for model training (although, technically, was used for hyperparameter tuning). I note that a validation dataset can either be a fixed randomly-selected subset of the training data (i.e., a "hold-out" dataset) or – through a process known as cross-validation – it can be a dynamic random subset of the training data (i.e., *k-fold cross-validation*).

- *Out-of-time or Test sample predictive performance* – evaluation of the loss metric on a separate "unseen" validation dataset that was not used for either model training or hyperparameter tuning. This is the purest and most objective type of "unseen" dataset on which to assess the model's predictive performance.[35]

- *Sensitivity analyses* – evaluation of the change in the model's predictive output based on certain changes in the values of one or more data inputs.

- *Stress testing* – evaluation of the model's predictive performance and output values as a result of extreme or unexpected data input values (i.e., "edge cases").

While AI / ML models do not change the need for outcomes analyses, there are a few nuances of such testing of which risk managers should be aware.

- *Is there sufficient evidence that the final model generalizes reasonably well on unseen data?*

  While, technically, AI / ML models can be both under-fit and over-fit to the training data, there is a greater tendency toward the latter due to the increasing use of high dimensional data inputs ("Big Data"), complex model architectures (e.g., deep neural networks with thousands of weights), and the never-ending quest for greater predictive accuracy. Furthermore, traditional diagnostic tools such as variable statistical significance, conceptual soundness assessments of estimated relationships, and multicollinearity tests that help to prune manually the dimensionality of models are less common outside of traditional regression-based model architectures. Accordingly, for AI

[35] For models whose predictions involve a time component – such as forecasting – the test dataset is not typically random. Rather, it covers a contiguous time period either after (or before) the end (the start) of the training / validation data.

/ ML models, validation and test set model performance assessments become the predominant means to assess the risk of model over- (or under-) fitting.[36]

Figure 7 below illustrates how these datasets are typically used to assess model performance.

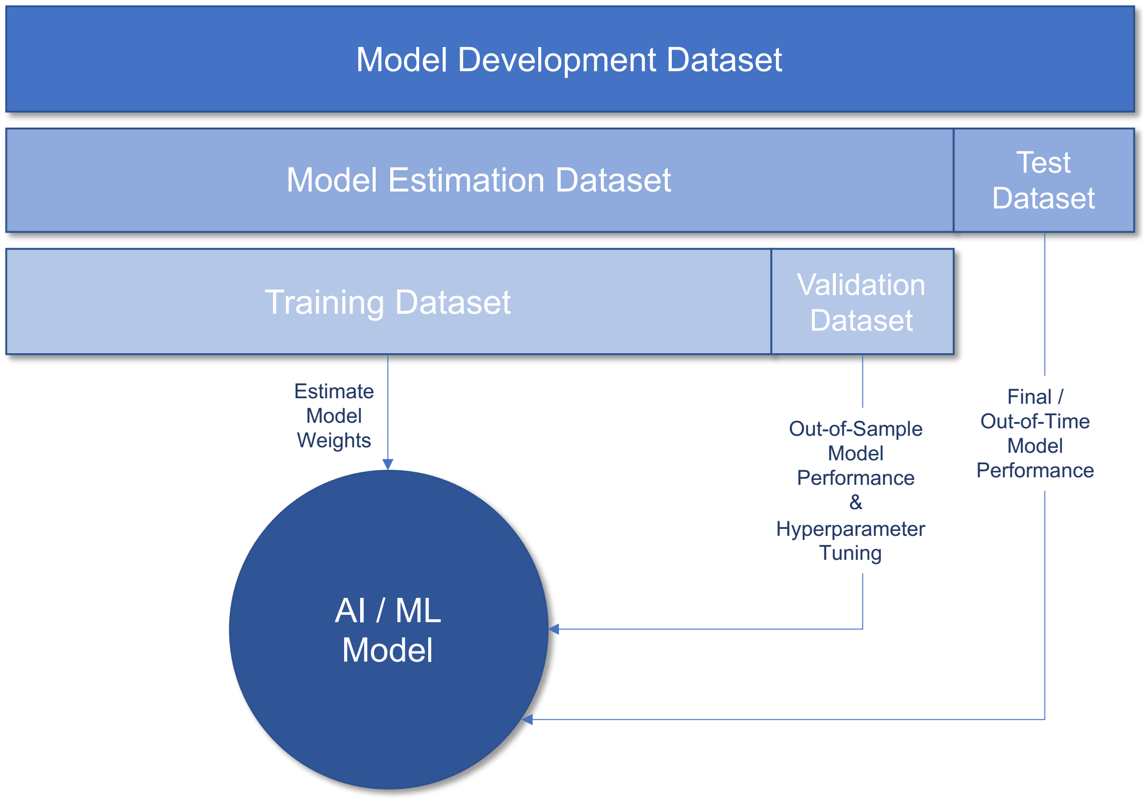
Model Development Dataset
Model Estimation Dataset
Test Dataset
Training Dataset
Validation Dataset
Estimate Model Weights
Out-of-Sample Model Performance & Hyperparameter Tuning
Final / Out-of-Time Model Performance
AI / ML Model

Figure 7: The roles of training, validation, and test datasets

The training dataset (also known as the "in-sample" dataset) is used to estimate the weights of the model architecture, is the primary workhorse dataset for model estimation, and is – accordingly – the largest of the three datasets. The validation dataset is excluded from estimation of the model weights and is therefore used to produce an unbiased measure of the model's predictive performance as part of the model development process. However, since the validation dataset is used indirectly to tune the hyperparameters, the test dataset is needed to provide the final unbiased measure of model predictive performance.

For smaller datasets, rather than having a single fixed validation dataset (which may reduce the training dataset to an unacceptably small size), the developer may instead use a cross-validation process – such as *k-fold cross validation* – in which the model development dataset is randomly split into k equal segments (or "folds") after randomly excluding the test dataset. Thereafter, the model training process is executed k times – with each execution sequentially using one of the k

[36] As noted above, for models with a time component to their predictions, the test set may be referred to as an out-of-time dataset.

folds as the validation dataset and the remaining *k-1* folds as the training dataset. At the end of the *k* executions, each observation in the dataset will have been used once in a validation dataset and *k-1* times in a model training dataset. Out-of-sample model performance is then measured by averaging the loss metric on each of the *k* validation datasets. **Figure 8** below illustrates this process where k=5. In this example, after the test dataset is excluded, the remaining data is split randomly into 5 equal size groups. Then the model is estimated 5 times – each time using four of the folds as training data and the remaining fold as the validation data. Once the model has been fully tuned, the final model is generated and its performance is measured using the test data.

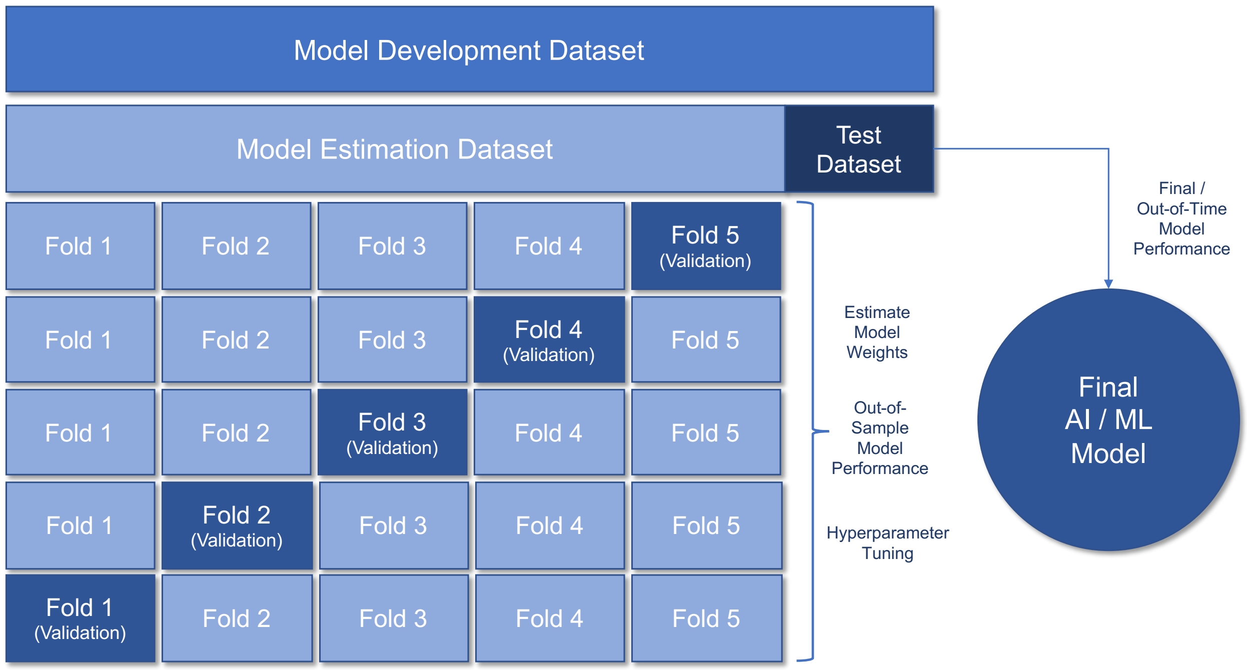

Figure 8: Illustration of k-fold cross validation

To assess the risk of over- or under-fitting, the developer should be comparing the model's loss metrics on the training, validation, and test datasets – looking for disparities in performance across the different samples. This assessment should occur not only on the final model, but also during the model development process such as when exploring different levels of model complexity or different hyperparameter values.

- *Has the developer used appropriate model performance metrics?*

Risk managers should confirm that model performance is measured using widely-accepted metrics that are appropriate for the specific prediction task(s) for which the model will be used. For example, a metric that calculates predictive accuracy may not be relevant to a model that will be used solely for rank-ordering. Additionally, more advanced accuracy metrics may be needed for NLP models, multilabel prediction models, object detection models, etc. Indeed, a best practice would be for such metrics to be defined and managed at the overall Bank group level to promote consistency across developers in both selection and execution, and for this inventory of approved metrics to be subject to appropriate on-going governance – involving periodic updates subject to formal change management processes, as well as annual re-certifications.

- *Has the developer measured the model's incremental predictive performance relative to an appropriate benchmark?*

  While developing deep neural networks can be "cool" and intellectually-rewarding, they – and many other complex model architectures – typically come with much higher development and governance costs (e.g., large data requirements, significant compute charges, potentially greater model development and validation labor costs, potentially greater explainability and documentation costs, and typically higher model maintenance costs) as well as higher inherent risks due to increased technical complexity and lower transparency. However, while such complex model architectures usually result in greater predictive accuracy, developers should be asked to justify the increased risks, direct development costs, and governance costs through an objective measurement of the model's *incremental* predictive accuracy over an appropriate benchmark model with a lower risk and cost profile.

  Ideally, developers should start the model development process with a simpler appropriate model architecture – such as linear regression, logistic regression, time series model, Naïve Bayes, etc. – to create a baseline level of model performance, and to assess whether the simple model's results may be sufficient to meet the user's business and technical specifications (which also typically include a desired time-to-market). Should more complex model architectures be explored and recommended by the developer, a clear and objective comparison should be provided to users outlining the incremental costs and risks of the more complex model vs. those of the simpler model. This discipline will not only help the Bank manage its overall model risk profile more proactively, but it will also be a means to increase efficiency in model development and governance spending – thereby increasing use-case ROIs.

- *Has the developer performed appropriate robustness / stress testing of model performance – particularly for computer vision and NLP models?*

  Stress testing is different than the previous two types of model performance assessment in that it seeks to explore how the model's estimates behave under more extreme data input values (e.g., "edge case" data input values that may not be observed historically and, therefore, are not contained in either the training, validation, or test datasets). Such testing helps to define the limits of reasonable model performance and is a crucial part of model risk mitigation. While stress testing is already a standard procedure in Bank model validation, AI / ML models create unique dimensions of such testing of which risk managers should be aware.

  First, given the reliance of many AI / ML models on high dimensional non-linearities, models can behave unexpectedly particularly for: (1) edge-case data attribute values subject to significant non-linear amplification (or shrinkage) due to the model's specific architecture and (2) edge-case data attribute values whose estimated relationship with the target variable is subject to significant extrapolation (and, therefore, uncertainty) due to its distance from training data values. In both cases, appropriate model testing should be performed to determine the boundaries of these edge cases such that model users can be informed of the corresponding risks and limitations and, accordingly, implement appropriate risk mitigations for data input values that approach these boundaries.

Second, in the areas of computer vision and NLP, stress / robustness testing is needed to mitigate the risks of potential domain shift in production data – that is, variations in input images or text inputs that differ from those used for training / validation / testing. For example, for a computer vision model, you can think of this testing as entailing various augmentations of input images – such as image sizes, aspect ratios, sharpness, coloring, rotations, offsets, zoom levels, and many other image characteristics – that the model may not have encountered during training and, accordingly, evaluating under what conditions the model may not perform well. As with the prior example above involving non-linearities, any areas of poor performance should be clearly communicated to model users for appropriate mitigation.

4. Other AI / ML Risk & Governance Issues

In this final section, I provide some perspectives on additional risk and governance topics related to AI / ML models. The intent here is to highlight these important topic areas and provide a summary of key considerations for risk managers. However, for a much more comprehensive understanding of the issues and the current state of corresponding risk mitigations, I encourage risk managers to research these areas more deeply.

4.1 Vendor Models

Vendor AI / ML models – while similar to the pre-trained AI / ML models discussed previously in **Section 3.3.1** – typically carry even greater inherent risks to Banks due to a lack of transparency into the model architecture, technical underpinnings, and even – for some vendor models – the data inputs due to the vendor's protection of its proprietary intellectual property. In my experience, vendors may only share with their customers some high-level model information – in many cases, marketing white papers or other sales-related materials that typically provide little of the information needed to assess conceptual and technical soundness. If the model ingests certain data inputs provided by the Bank, then knowledge of such inputs is obviously shared; however, vendors typically do not share specific information on the data inputs *they* may provide to the model (e.g., proprietary data owned by the vendor or external data licensed by the vendor from one or more third-parties), nor do they share information explaining how the Bank's data inputs are specifically used by the model to produce the predictive outputs.

While this lack of transparency of vendor models was explicitly addressed by the federal bank regulatory agencies in their 2011 supervisory guidance on model risk management, and such guidance is equally applicable to today's AI / ML models, there are two nuances of the AI / ML vendor model risk profile that I would like to highlight here:

1) *Compliance Risks* – as discussed in **Section 3.1**, AI / ML model architectures facilitate the use of ever-larger "Big Data" datasets to take advantage of the additional dimensional depth provided by externally-sourced financial, socioeconomic, geographic, behavioral, attitudinal, transactional, and other consumer or business attributes. Typically, vendors aggregate this data from numerous third-party sources, with some of the data fields estimated from underlying models (e.g., consumer income or wealth estimates), and other fields derived from consumer surveys and/or related segmentation analyses. Without sufficient knowledge of these data inputs, Banks face heightened legal and compliance risks – for example, some of this data may violate applicable consumer privacy laws and regulations, the Bank – as the model user – may not possess the applicable data usage rights from the third-party data owners, and one or more data fields may expose the Bank to potential fair lending or other related consumer protection risks (discussed further in **Section 4.4** below).

2) *On-going Monitoring* – as discussed in **Section 3.4**, model outcomes analysis is a critical component of AI / ML model validation, in general, and is even more important for vendor models due to their lack of transparency. Indeed, outcomes analysis typically forms the primary basis of model validation for vendor models. However, for outcomes analysis to provide valuable feedback to the Bank on model quality, the model must be stable so we can observe its predictions over a period of time and compare those predictions to actual realized outcomes. This model stability requirement tends to be a challenge for some AI / ML models since frequent model re-trainings by the developers on new data are common-place.[37] For a Bank, this challenge is even greater for vendor models since they lie outside of a Bank's control and, again, there is little transparency to such model updates. In such cases, the vendor's on-going model backtesting results may be a misleading indicator of model quality and performance as they may be based on an ever-changing set of model weights, hyperparameters, or model architecture.

Risk managers should review the Bank's vendor model policy or guidelines to ensure they reflect the full set of relevant risks associated with AI / ML models along with appropriate requirements designed to mitigate these risks to levels consistent with the Bank's risk appetite. It is also recommended that such requirements be clearly communicated to potential vendors during the Bank's initial due diligence phase and incorporated into vendor contracts. To the extent that vendors cannot, or will not, comply with one or more of these requirements, Banks may consider limited exceptions; however, such exceptions should be escalated through appropriate governance channels with full disclosure of the

[37] Indeed, it can be so common-place that it is actually an automated process.

corresponding risks associated with the exception – as well as the proposed controls or mitigants designed to manage these risks to an acceptable level.

4.2 Operational Risks

As discussed in **Section 2**, the growth and concentration of AI / ML methodologies will transform the Bank's risk profile in new and fundamental ways. For example, the vast and growing number of available AI / ML algorithms, the different technical platforms supporting these algorithms, the proliferation of freely available open-source tools and training code, and the expansion of publicly available datasets and pre-trained models all create an immense "superstore" of model development configurations from which a developer might now choose. While such choice may provide many benefits to the enterprise, it also increases the Bank's inherent risks as developers freely use algorithms, open source code bases, external datasets, pre-trained models, etc. with which they may have limited experience, and of which they may not fully understand the associated risks and limitations.

This evolution of analytical technology is being coupled with a growth in the deployment of AI / ML-based models and systems throughout the enterprise – for example, in robotic process automation ("RPA"), intelligent process automation ("IPA"), credit risk management, financial reporting, customer relationship management, revenue optimization, robo-advisors, voice assistants, chatbots, and a growing pipeline of additional use cases. Together, these trends work together to increase the level, and concentrations, of operational risk across the enterprise in ways that risk managers need to acknowledge, assess / measure, and manage.

Another important risk area related to the practice of leveraging pre-trained models, or due to the general lack of transparency of most AI / ML models, is *adversarial attacks*. Much can be written about this area; however, I will focus only on the primary concept – certain AI / ML models can be engineered, through direct manipulation of the model's mathematical structure, or through "poisoning" the training data, into creating a "backdoor" that can be exploited to manipulate the model's outputs. That is, once implemented by the developer or a hacker, anyone with knowledge of the "key" – that is a certain data input configuration – can obtain a pre-determined outcome from the model. For example, consider a computer vision model that is used to identify the faces of company employees to provide security access to certain office areas. The underlying AI model could be manipulated by the developer to always provide access to an individual displaying a certain visual cue – such as a unique pair of eyeglasses or a hat with a certain phrase – regardless of whether that individual is an authorized employee, thereby circumventing the security control. Because this backdoor is embedded within the complex mathematical architecture of the AI model, it would be very difficult to discover – although AI researchers are beginning to develop tools to assist with such detection.

Overall, the open source nature of many AI / ML technologies, the rise of a knowledge-sharing culture across the AI / ML community, the prevalence of freely-available pre-trained models for high-value AI tasks – such as computer vision and NLP, and the lack of transparency into many AI / ML models all combine to increase the risk of potential security breaches of AI / ML model deployments – allowing bad actors to seed hidden backdoors into models and model-based processes, thereby exposing the Bank to potentially serious operational risks.

4.3 Explainability and Interpretability

Previously, **Section 3.2** introduced some of the challenges and risks associated with the lack of transparency into how AI / ML models work – in particular, how specifically are the model's data inputs translated into model output? What is the directional relationship, and quantitative sensitivity, of the model's output to a given change in each data input's value? How important is each data input to the model's predictive performance? Not only is this a challenge for models designed for standard Bank use cases – such as loss forecasting, financial reporting, fraud detection, etc. – but it is even more so for AI models used for more novel computer vision and NLP use cases. Here, what does it mean to have model transparency when the data inputs are either pixels or individual words, or individual characters?

In response to these challenges, a fair amount of active research has been devoted along two related paths – the first is focused on developing analyses and tools to *explain* the behavior of AI / ML models *after* they have been developed. That is, how can we leverage various model artifacts to infer how the model is behaving along certain dimensions – including computer vision and NLP models. Alternatively, the second path is focused on developing new AI / ML model architectures that are foundationally *interpretable* – that is, the model architecture is specifically designed to provide direct transparency into the model's behaviors. Both of these research paths have yielded promising results and, in fact, have created an interesting debate within the AI / ML community as to whether it is better to simply migrate on the front end to interpretable AI / ML model architectures for all use cases where transparency is deemed important, instead of maintaining existing "black box" model architectures and inferring their behaviors on the back-end from a growing set of explainability tools.

For risk managers, the key takeaways are the following. First, the Bank does not have to settle for a black box if it chooses to leverage AI / ML model architectures, and it should consider adopting policies or guidelines around model explainability / interpretability for use cases for which such transparency is deemed crucial.[38] Second, interpretable models may come at a cost of predictive performance due to their use of constraints during the model training process. Accordingly, quantifying the trade-off between transparency and predictive performance may be useful for model users in cases where the need for transparency may not be as critical. Third, the results from explainability tools may also be estimates and, as a result, risk managers should ensure appropriate effective challenge of such tools and estimates prior to the reliance on their outputs.

[38] I remain skeptical that we will see vendors share the results of these tools / methods as applied to their models due to their continued concern over trade secret protection.

4.4 Fairness

Fair lending has long been a hallmark of Bank compliance risk management with significant resources devoted to *ex post* testing of Bank lending outcomes – such as credit decisions, loan pricing, product choice, geographical lending patterns, and loan servicing – to detect potential evidence of statistically significant, unexplained lending disparities between certain protected class groups and corresponding control groups.[39] Such *ex post* testing is typically performed on a regular basis (e.g., quarterly or annually depending on lending volumes), employs statistical methods to focus on disparities in lending outcomes that cannot be explained by differences in legitimate borrower qualifications (such as debt-to-income levels and credit history) or loan transaction attributes (such as loan-to-value ratios and loan purpose), and generally focuses on potential *disparate treatment* in the Bank's lending processes – that is, the systematic application of discretion by Bank personnel that results in relatively unfavorable lending outcomes for one or more protected class groups relative to their respective control groups. This testing is designed to comply with federal fair lending laws and regulations such as the Equal Credit Opportunity Act ("ECOA") as implemented by the Federal Reserve's Regulation B and the Fair Housing Act ("FHA").[40]

Over the past decade, however, certain trends have converged in government fair lending enforcement activities, consumer lending technology, and Banks' use of alternative data to alter traditional Bank fair lending compliance testing in important ways. Specifically,

- Federal bank regulatory and enforcement agencies – including the Consumer Financial Protection Bureau ("CFPB") and the Department of Justice's Civil Right Division ("DOJ") – have increasingly enforced federal fair lending laws and regulations under the *disparate impact* theory of discrimination. Importantly, unlike the traditional disparate treatment theory that is based on a pattern or practice of discrimination driven by the inconsistent application of lender discretion to similarly-situated loan applicants (such as inconsistency in manual credit underwriting decisions, credit exceptions, and pricing exceptions), disparate impact discrimination can be driven by the consistent application of lending policies or procedures to all applicants, but where: (1) the policy or procedure has a differential adverse impact on one or more protected class groups (relative to the corresponding control groups), (2) the policy or procedure cannot be justified based on business necessity, or (3) even if the policy or procedure is a business necessity, there may be an alternative policy or procedure that is similarly effective, but has a less discriminatory impact. In general, because disparate impact discrimination focuses on the Bank's lending policy or procedure, rather

[39] In general, protected class groups for which Banks have data to formally test include race, ethnicity, sex, marital status, and age.

[40] I am focusing my comments here to federal bank regulatory and enforcement agencies; however, the state attorneys general and community activist groups may also launch investigations into Bank fair lending compliance. Additionally, in my discussion of federal fair lending laws and regulations below, I present information from a compliance practitioner's perspective, not from a legal perspective. You are strongly encouraged to consult appropriate legal counsel for technically precise interpretations and recommendations.

than the disparate discretionary actions of Bank personnel, it sweeps the Bank's lending-related models and automated lending tools into the realm of fair lending compliance risks.[41]

- As discussed previously, the expansion of AI / ML models throughout the enterprise changes the Bank's aggregate model risk profile in fundamental ways. One of those changes is the growing algorithmic complexity of Bank models as evidenced by: (1) an expansion in the number of data attributes used to predict a specific modeled outcome, (2) the inclusion of high-dimensional non-linear relationships among the data attributes to improve model predictive performance, and (3) a general lack of transparency into how each data attribute specifically affects the model outcome. From a fair lending perspective, all of these complexities increase the Bank's risk. The larger number of data attributes increases the mechanisms by which disparate impact may occur. The high-dimensional non-linear relationships and general lack of model transparency complicate our ability to understand exactly how each attribute affects the modeled outcome and, therefore, how each attribute may or may not contribute to measured lending outcome disparities across protected class and control groups. And the complex non-linear interactions across attributes increases the potential for the model to create indirect "proxies" of protected class or control group membership as predictive factors – even though such group membership is not a direct input to the model. As noted in **Section 4.1**, these fair lending risks are even more elevated for vendor models due to the even greater lack of transparency into the model architecture and data inputs.

- Finally, the rise of "Big Data" or alternative data – particularly in the consumer and small business spaces – raises the fair lending risk profile of AI / ML models not only because of the increase in predictive attributes through which potential disparate impact can occur, but also because – by their very nature – alternative data typically is only indirectly or, in some cases questionably related, to customer credit performance. For example, the magazine subscriptions a customer may have, or the retail stores a customer frequents, are not attributes that have a logical direct connection with customer credit performance – even if the AI / ML model identifies them as predictive attributes.[42] This lack of a logical connection to customer credit performance raises the risk that such attributes, if deemed to have a potential disparate impact, would not satisfy the "business necessity" standard that may mitigate such risk. Once again, vendor models that rely on third-party data inputs tend to present an elevated risk in this area due to the lack of transparency into the specific nature of these inputs.

[41] Along with this focus on the disparate impact theory of discrimination, the industry has also seen the adoption by federal bank regulatory and enforcement agencies of statistical proxies of an applicant's race, ethnicity, and gender that permits an expansion of fair lending testing to non-mortgage credit products – such as auto loans, credit cards, personal loans, etc. – for which Banks do not collect regulatorily-mandated demographic data from its customers.

[42] Within this context, I am excluding from the term "alternative data" more direct measures of customer financial health such as the customer's bank account cash flow data or alternative credit scores based on, for example, the customer's utility or mobile phone payment histories. It is generally agreed that these "alternative" attributes have a more direct relationship with customer credit performance and, as a result, present a much lower risk of not satisfying the "business necessity" standard of a disparate impact risk assessment.

Overall, these three trends have converged to place an ever-growing focus by federal bank regulators, enforcement agencies, and other important stakeholders on the potential disparate impact risks of the AI / ML models used by Banks in their lending processes. For risk managers, this has the following implications:

- There is now an important intersection between model risk management and fair lending compliance. In particular, there is a need to supplement the Bank's model inventory – in conjunction with the Bank's compliance risk management function – to identify those models whose specific use(s) expose the Bank to fair lending risk. As well, the specific types or sources of fair lending risk that apply to such models should be listed – such as potential bias in underwriting and / or potential bias in pricing, etc. – commensurate with the model's specific applications within the Bank's overall lending process. For these models, it is important that model risk managers keep the Bank's compliance risk management function apprised of new models or changes to existing models so that appropriate fair lending risk assessments can be performed in a timely manner. Additionally, model performance metrics that exceed established model performance thresholds, or any other material model risk issue identified via on-going monitoring, should be timely communicated to compliance risk management for assessment. In some cases, poor predictive performance of a model used in the Bank's lending processes may trigger a fair lending compliance issue.

- The Bank should establish a fair lending testing and monitoring program and process for the AI / ML models identified above – including a well-defined corrective action process that would be triggered should such testing identify actionable fair lending risks. Typically, this activity is the domain of the Bank's compliance risk management function and can be performed by the same personnel who perform the Bank's *ex post* statistically-based fair lending testing on lending outcomes. However, should such resources not be available within the compliance function, then this activity could also be performed by the Bank's model risk management personnel – conditional on: (1) the Bank's compliance risk management function still owning responsibility for program design and management, (2) the Bank's compliance risk management function playing a lead role in the design of the program – including key elements of the testing, monitoring, and corrective action processes, and (3) the Bank's model risk management group having sufficient resources and training to execute the program in an effective and timely manner. I note that this program should not only pertain to fair lending testing prior to the model's use in production, but also address on-going fair lending monitoring of the model's performance – as well as model changes and changes to model use. The program should also apply to any vendor models that the Bank may choose to use.

- The Bank's overall fair lending risk profile associated with its AI / ML model use – along with key performance and risk indicators – should be aggregated and reported periodically to appropriate Bank governance committees including, as appropriate, Bank Board committees. This would typically occur within the context of the Bank's overall compliance risk management reporting.

I note that my analysis above relates specifically to potential discrimination in the area of consumer or small business lending. However, the concept of AI bias is much broader than just the lending context and encompasses potential adverse impacts across nearly all use cases. For example, facial recognition systems – based on computer vision AI models – may be used as part of a Bank's physical security infrastructure. However, such systems have been shown to have more difficulty recognizing faces of individuals of certain races or ethnicities. This may lead to more Bank employees in those racial / ethnic groups being denied facility access and, therefore, being subject to a manual security screening process. As another example, customer voice assistants – based on NLP models – may have difficulty understanding individuals with certain accents, thereby systemically directing individuals within certain ethnic groups to a less convenient customer service channel. Such biases are typically driven by biases within the data used to train and test the models – such as the facial images or speech samples that may underrepresent the impacted racial / ethnic groups; however, these biases may also occur due to insufficient diversity in the model development and deployment teams. Without such diversity, the development process may fail to consider modeling structures that may increase model predictive performance along multiple diversity dimensions, and the testing process may fail to include important diversity performance focal points.

*   *   *

About the Author

Drawing on his 25-year experience as a Big 4 financial services analytics / AI consulting leader, Ric currently provides strategic assistance to his clients in the areas of traditional and AI/ML-based predictive modeling, model risk management and governance, fair lending analytics, and algorithmic fairness.

Ric earned a PhD in Economics from the University of Rochester and a B.S. in Finance and Economics, *summa cum laude*, from the State University of New York College at Oswego.

Richard R. Pace
aimodelrisk@gmail.com