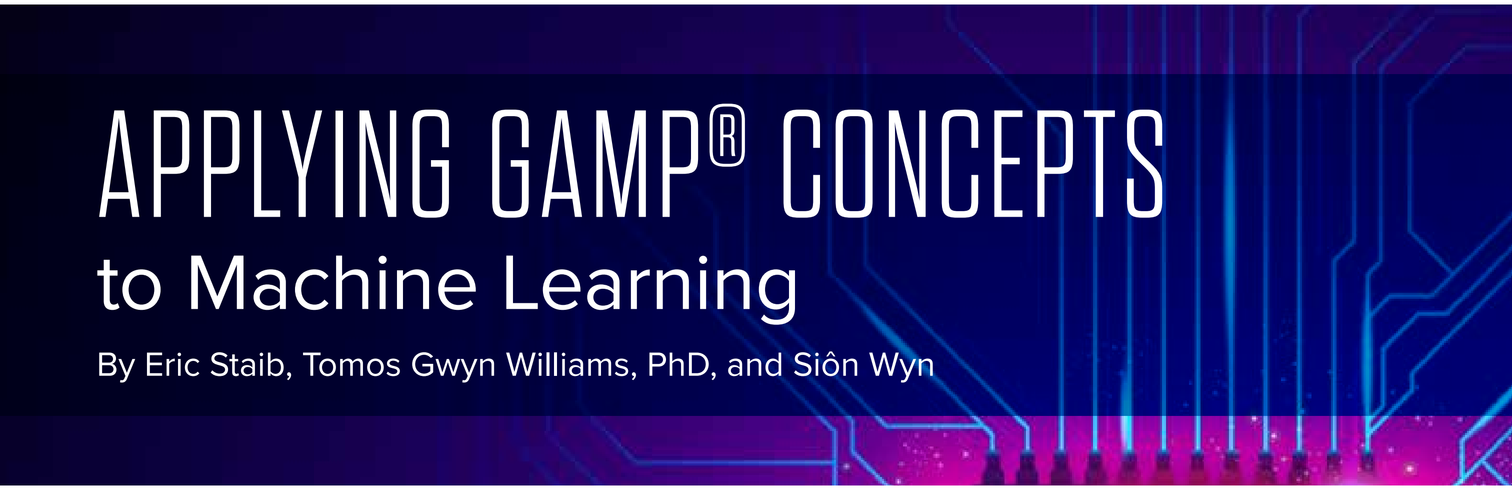


APPLYING GAMP® CONCEPTS to Machine Learning

By Eric Staib, Tomos Gwyn Williams, PhD, and Siôn Wyn

This article explores life-cycle activities for machine learning (ML) within regulated life sciences. It positions and contextualizes the life cycle and management of the ML subsystem or components within a wider system life cycle. It also gives general descriptions and guidance illustrated by a case study demonstrating a ML application to medical image recognition, or software as a medical device (SaMD) [1].

This article focuses on the ML component or subsystem embedded within the wider system, solution, or application. It is not intended to be a general primer or introduction to artificial intelligence (AI) or ML, nor an introduction to general computer validation and/or life-cycle activities.

ML is a subdiscipline of AI. An ML system builds a predictive model from input (i.e., training) data, and uses the learned model to make useful predictions from new, never-before-seen data.

For most systems that use ML, many aspects of the traditional computerized system life cycle, and compliance and validation approach, are still fully applicable (e.g., those related to specification and verification of user interface, reporting, security, access control, data integrity, and data life-cycle management).

The use of the term *ML component* is not intended to suggest that such a component is a single entity. In most cases, the ML component will typically consist of several subcomponents comprising a "pipeline" supporting a number of functional stages, such as input/data preparation or output/results filtering, and one or more central ML "engine" or model(s) connected together. In such cases, the term *ML subsystem* is the most appropriate. The authors strongly encourage the use of appropriate software automation and other tools to develop and manage both the ML subsystem and the broader, overarching system, solution, or application. This article also seeks to avoid the implication that new documentation deliverables are necessary, unless they are clearly required by regulations (for example, in some cases of SaMD where device requirements, user needs analysis, human factors evaluations, clinical trials, and regulatory submission need to be considered), or such deliverables are clearly beneficial to the reliability, maintainability, and/or quality of the operational system and its fitness for intended use. (See the sidebar for other key definitions.)

Operational ML subsystems provide different outputs as they evolve, but the verification and validation of the system should be kept updated in line with these changes. This must include appropriate change management, version control, and monitoring. In addition, some ML systems have stochastic elements (having a random probability distribution or pattern), which means that results will be different for identical inputs regardless of model training. Therefore, validation and verification must use a sufficiently large validation data set and calculating summary performance measures that are meaningful and representative of the overall system performance and robust to small output variations between successive runs.

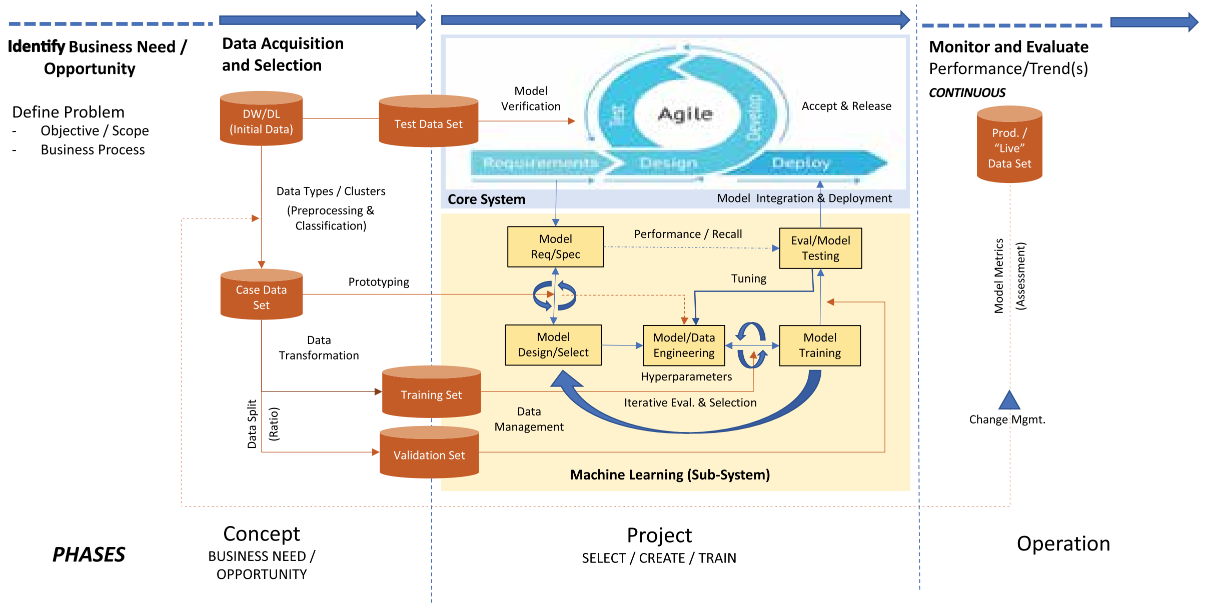
PREREQUISITES AND CONTEXT

There are many similarities in best practices between ML and more traditional algorithmic programming. Successful implementation of ML requires good business analysis and process understanding by data scientists, effective planning, and the application of good software development, engineering, and maintenance practices. The business case and intended use must be fully understood to best select the right data and data management must be supported by a mature data governance strategy.

Performance metrics are important in the design of any ML subsystem. They define what output(s) will be generated and how they will be evaluated against the required or expected results to determine the ML performance. These metrics drive the iterative training, evaluation, and improvement stages that are inherent within the development of all ML systems, as described in the project/production phase.

Another key aspect of ML development is the tight integration of data and metadata into the development process. The term

Figure 1: ML subsystem life cycle.



data-centric development is sometimes used to reflect this. As a result, data should be managed with utmost care, including controls for data acquisition, selection, classification, cleansing, and augmentation.

As with other software system development, ML development has business, technical, and project risk activities commensurate with the complexity and novelty of the system. Managing these risks require good process/business analysis, risk analysis, and cost/benefit analysis at all stages of development to recognize issues and decide whether to take mitigating or rectifying steps, or to terminate the project.

Development planning requires consideration of human factors or bias, privacy, security, and legal liability. This requires transparency and an understanding for the ability to reproduce outcomes, adequately interpret the results, and understand the applicability for how models will be applied.

The level of risk depends on the intended use. The extent, rigor, and documentation of validation and controls should take into account factors such as the level of human involvement, the significance of information to the healthcare decision (to treat or diagnose, to drive clinical management, or to inform clinical management), and the healthcare situation or condition, (critical, serious, or nonserious).

The ISPE *GAMP® Records and Data Integrity Good Practice Guide: Data Integrity by Design* [2], Appendix S1: Artificial Intelligence: Machine Learning, identifies the life cycle of data within a ML framework, emphasizing the link to both the GAMP data life cycle and GAMP system life cycle. Wider data integrity (DI) topics are also discussed in the guide.

ML SUBSYSTEM LIFE-CYCLE OVERVIEW

The following is an overview of the life-cycle model for the ML subsystem (see Figure 1). Phase terminology consistent with the GAMP 5 overall system life cycle is used including concept, project/production, and operation. A case study follows that presents the specific life-cycle activities for a SaMD product. For consistency with the *GAMP® Good Practice Guide: A Risk-Based Approach to Regulated Mobile Applications* [1], phase terminology includes project and production.

In the concept phase, the business need or opportunity is identified, clarified, and agreed upon. The specific problem to be solved is defined. The initial data is identified (it may be from a data warehouse or data lake), selected, and prepared as "case data." Prototyping allows the assessment and selection of suitable algorithms and hyperparameters, and preliminary hyperparameter values used to control the learning process. Examples are variables that determine the network structure, such as number of hidden units, and variables that determine how the network is trained, such as the learning rate. Data management begins in this phase when the case data is originally collected.

During the project/production phase, following a defined plan, the selected technologies and technical architecture are defined. Formal risk management activities commence, as well as other supporting activities, including project-based configuration and change management. Project/production phase activities for the ML subsystem are typically iterative and incremental rather than linear. These iterative activities include model design/selection, engineering, model training, testing, evaluation, and hyperparameter tuning.

Definitions

Artificial intelligence (AI): a system that displays intelligent behavior by analyzing its environment and taking actions (with some degree of autonomy) to achieve specific goals. AI-based systems can be purely software-based, acting in the virtual space, or can be embedded in hardware devices. As a scientific discipline, AI includes several approaches and techniques, such as machine learning (of which deep learning and reinforcement learning are specific examples), machine reasoning (which includes planning, scheduling, knowledge representation and reasoning, search, and optimization), and robotics (which includes control, perception, sensors, and actuators, as well as the integration of all other techniques into cyber-physical systems).

Machine learning (ML): a subdiscipline of AI and a program or system that builds (trains) a predictive model from input data (such as training data). The system uses the learned model to make useful predictions from new data drawn from the same distribution as the one used to train the model.

Deep learning: also known as deep structured learning or convolutional neural networks (CNNs), it is a part of a family of machine learning methods based on artificial neural networks with representation learning.

Random forest: a ML technique used to solve regression and classification problems.

Case data: data that is strategically selected to be unbiased and representative of the types of information to be processed by the ML, used for selection of training and validation samples/subsets.

Training data: a sample and/or subset of data, used for learning, to fit the model parameters of the model/classifier.

Validation data: a sample or subset of data used during model training and tuning to evaluate the model. The data provides evaluation independent of the training data, but not completely independent of the model training process. In data science and AI/ML, validation is used differently in GxP computerized systems.

Test data: a sample and/or subset of data excluded from all training, tuning, and validation activities, reserved to assess and evaluate the performance of a fully specified model/classifier.

Gold standard/"ground truth": a set of results that serves as the approved external criterion in which the model/classifier output is ultimately evaluated and/or compared against.

Data management is another key project/production phase activity, including the acquisition of new data, secure storage and handling, preparation (including labeling), and partitioning of data into training and validation data sets. During the model development stages, the training data set is used to train the model, and the validation data set is used to provide an unbiased evaluation of the model while tuning the model's hyperparameters. In certain scenarios, such as cross-validation experiments, specific data sets may fulfill the role of training and validation but not in the same iteration of the experiment. The test data set is excluded from all training and hyperparameter tuning activities; instead it is used to provide an unbiased evaluation of the final model within the overarching system. There is usually integration of the ML component into the wider computerized system and deployment into the target or other environment where acceptance and release activities are performed using the test data set.

In the operation phase, the system performance is monitored and evaluated. As new (live) data becomes available, further configuration/coding, tuning, training, testing, and evaluation are performed. There is likely to be a tight and iterative loop of alternating production and operation activities as the availability of new data and ongoing performance evaluation and quality checks lead to opportunities for improved performance, both proactive and reactive, or changing scope of use. This requires effective change and configuration management applied to all constituents of the ML system, such as code, the data, and models.

ML SUBSYSTEM LIFE-CYCLE PHASES

The following sections describe and discuss the typical activities conducted during the ML subsystem life cycle and are supported by an illustrative case study example [3] at the end of the article.

Concept Phase

The objective of this phase is to provide insight into the expected development cost and operational benefits of a ML subsystem. This should include a decision or rationale on why a ML solution shall be incorporated. This phase also provides opportunities to research and investigate which ML algorithms should be considered for development based on cost, development risks, and expected performance. This phase also include efforts in gathering initial case data and understanding the properties of that data.

Identify business need and opportunity

The business need is developed and analyzed, the overall process and workflow are defined and agreed upon, and how the proposed application will support the process is identified. This analysis will help determine constraints, such as availability of data, deployment hardware, legal liability, and regulatory and intellectual property (IP) factors. Detailed data-related factors such as source, structure, format, and segmentation should also be considered.

Problem definition

At this stage, the initial set of requirements may be specified. This initial "requirements specification" drives the development and defines the functionality required from the system and ML subsystem.

Nonfunctional requirements such as integration and deployment constraints should also be considered at this early stage to inform the choice of ML algorithms. Nonfunctional requirements include an initial set of performance metrics. These are a detailed description of the ML subsystem output and how these outputs will be compared to the defined expectations. This comparison will provide quantitative measures of how well the subsystem performs. These measurements drive the training, evaluation, and tuning of the ML subsystem models. The performance metrics may change during development, training, and retraining. Other nonfunctional requirements include deployment constraints, such as choice of hardware, and performance constraints such as speed and/or capacity.

Prototyping

ML projects can benefit significantly from deploying algorithms and techniques developed for and applied to other applications and use cases. The objective of this stage is to conduct research and initial prototyping to identify which algorithms and resources are most likely to result in successful delivery of the project.

The ML field has a varied and growing range of algorithms and model architectures to choose from, and within each algorithm there are numerous hyperparameters to tune. For a new system, it is unlikely that the choice of algorithm is so clear-cut that a decision can be made to fully specify the component and proceed to development at this stage. In order to decide which algorithm is most suitable and how it should be trained and evaluated, the candidates should be evaluated against the operational, performance, and, if relevant, regulatory requirements. These activities provide an early indication of the likely predictive performance of the model and how likely the system is to achieve that level of performance.

Data acquisition and selection

An initial set of data will be collected from the existing business activities, or need to be gathered, to provide a starting point for the prototyping. Once identified, this stage determines what is required to prepare the data for the training and evaluation of the models, including formatting, cleaning, and feature extraction (collectively referred to as data transformation). It is also likely that the data needs to be labeled to provide the training inputs from which the prototype subsystem will be evaluated. At this phase, it is not expected that the data be complete because subsequent stages will identify if there is a need for additional data and the plan for acquiring and labeling that data. It is, however, important to partition the case data into training and validation sets to avoid compromising future evaluations. Training data may include biased human decisions or reflect inequalities, or bias may be introduced by flawed data sampling, in which groups or classes are over- or underrepresented in the training data. Appropriate measures should be applied to control the risk of such bias.

Project/Production Phase

The output from this phase is an implementation of the ML subsystem integrated into the overarching IT system together with extensive performance evaluation measures. Integral to this is the development of the training and performance evaluation infrastructure that supports training, tuning, and evaluation of the models. Tools supporting model construction or data preparation may also be developed during this phase (such as tools that support labeling of the training data).

This phase follows an iterative approach where successive versions of the ML subsystem are specified, designed/selected, implemented, trained, tuned, and evaluated. The phase consists of a series of experiments that iteratively improves the design, implementation, and hyperparameter selection of the subsystem to optimize performance.

Project data management

Prior to the project/production phase kickoff, it must be determined if the case data fulfills the requirements of the project life cycle: for instance, that there is a sufficient amount of data to train the model and a data range that encompasses the expected real-world data. If this is not the case, additional data will be needed, which may require a separate data acquisition project. This phase also determines the appropriateness of the data for intended use, and prepares it for subsystem development. Activities include format specification, selection, and application tools for data annotation and clean up.

The extent and format required for the data is driven by the performance metrics previously obtained. For example, for the task of image analysis object localization, the performance metric is specified as the agreement between the ground truth and results predicted by the AI. The ground truth is the set of results that serves as the approved external criterion in which the model/classifier output is ultimately evaluated and/or compared against. To achieve this, the ground truth data and AI output must be in a comparable form that will enable that measurement to be made (for instance, by image segmentation). For a classification task, simple labeling of an image as containing a particular feature may be sufficient.

Model requirements specification

This stage may be considered a "tollgate" in the project/production phase, where information gained from the previous phase is documented and presented together with informed and detailed planning for the project/production phase. The objective is to provide information on the likely cost, risks, and benefits of the ML subsystem to inform a decision on whether or not to continue. The information presented during this stage provides confidence that the additional investment required in data acquisition, management, and development will deliver the business need.

The information and experience gained during the concept phase are utilized at the start of the project/production phase to specify and design the ML subsystem to as much certainty as possible and allow for the planning effort, including risk estimation, of its delivery. Activities in this stage include formulating the initial design of the subsystem by identifying the main components and how they will integrate to perform the analysis. Design decisions rely heavily on the practical experience gained in developing the prototype solutions in the previous phase. In addition, the specification of the subsystem is formed, which includes the format of the input and output data for the subsystem and the definition of performance metrics.

Planning involves detailed breakdowns of the development effort with estimates of timelines and associated risks. Risk analysis of the project can be performed during this stage to determine the items most likely to fail and provide for appropriate mitigating actions or alternative solutions to reduce risk. Planning also includes the specification of the development environment for the ML subsystem, which will have its own budgetary implications in the form of software licenses and computing and storage resources. The development operations and hardware infrastructure are set up to support the ML component training and evaluation. These may include code and data version-controlled repositories, applying any combination of local and cloud-based computation. This phase may use a research-focused language and platform, but should also take into consideration the end deployment requirements and platform to ensure that there are no subsequent technical or IP infringement issues.

Model design and selection

The baseline architecture of the ML model is chosen and designed during this stage. Knowledge gained from the prototyping phase is applied here to identify the single or small number of candidate algorithms identified as being most likely able to fulfill the model requirements, both functional and nonfunctional (such as performance). The requirements can be sufficiently broad to allow the selection of models across different ML algorithm classes. Data scientists should be wary of choosing too many candidate algorithms at this stage since the effort required to optimize each can be significant. If the number of candidate algorithms is greater than three, the scientists may wish to return to the prototyping stage to eliminate some to avoid excessive optimization.

The choice of the underlying ML algorithm leads to the set of hyperparameters for each model. Subsequent iterations of the development process refine the architecture driven by model test results.

Model/data engineering

This stage involves constructing the model architectures and the surrounding infrastructure for data input and evaluation that enables training and hyperparameter tuning of the models. Tasks include selecting, preparing, managing, and maintaining the data for training iterations and recording results to allow comparison between trials of different hyperparameters and results from different versions of the architecture. Once set up, the infrastructure is then employed to execute a series of trials in which the model hyperparameters are altered to determine the set of parameters that result in the best model performance.

Model training and hyperparameter optimization

This stage involves training a series of model instances by varying hyperparameter values (e.g., the number of hidden units or learning rate) and recording the results. Hyperparameter optimization may involve manual selection and altering the parameters after each iteration, or automated processes using exhaustive search or the more efficient Bayesian optimization of the hyperparameter space.

Most ML algorithms possess many hyperparameters and hence define a large hyperparameter space over which to optimize. However, applying knowledge of the algorithm and problem domain gained during the prototype phase allows data scientists to identify the subset of hyperparameters whose values can be predetermined and fixed, thus greatly reducing the parameter space. Though libraries and infrastructures exist that allow for automated hyperparameter tuning, data scientists are advised not to take a completely hands-off approach to hyperparameter tuning. Dividing the hyperparameter search space experiments into smaller regions by allowing only a subset of the hyperparameters to optimize for each experiment run can provide useful insights on the effect hyperparameters have on the model training and performance, leading to a more efficient tuning stage.

The output from this stage is a trained model using all the training data and an optimal, or near-optimal, set of hyperparameters. This is considered the best model given the existing fixed architecture and parameters evaluated using the validation data. The iteration of model design to model engineering to hyperparameter tuning to model training to model evaluation reveals insights into the performance of the latest and previous models. This yields further evidence as to how the model architecture and training options may be altered to improve the performance and then a redesign or selection of an alternative model may be performed and evaluated.

Evaluation and model testing

This is when the best-performing models from the previous training and selection iteration are subjected to the validation data. Excluded from the training of the previous iteration, the validation data is passed through the model(s) and the model's performance is evaluated. A key requirement for a fair comparison is to apply identical training and validation data sets to each candidate model. The results are compared to the gold standard labeling to produce a set of aggregate and indicative performance metrics, or scorecards, which inform on the current performance and drive the following iteration if required.

Many ML libraries incorporate the validation data evaluation into their training functions, thus automating much of this process. Data scientists, however, should be wary of relying on the quantitative measures for model evaluation. Visual qualitative evaluation of the validation results often leads to better insights on how the model is performing, allowing common error modes to be identified/addressed, and enabling crucial refinement of the performance metrics to provide better alignment with the required outputs. To this end, it is advisable to use expert and domain knowledge when hyperparameter tuning, rather than relying solely on fully optimized hyperparameter tuning functions provided by many development environments. In practice, this will involve a hybrid approach consisting of a series of tuning experiments where a subset of hyperparameters are tuned according to a performance metric, interleaved with manual interpretation and qualitative analysis of the results, to determine the next set of tuning experiments or to terminate the tuning activities.

A detailed description and evidence of the performance evaluation and comprehensive performance measures of the pre-released product is a data science-based expectation.

When target model performance is achieved and/or no further changes to architecture are identified, the best performing ML models are selected as the candidates for integration into the overarching IT system and deployed. This selection is based on not only the nonfunctional requirement of performance on the validation data set, but also on the criteria defined in the requirements, such as the ease of algorithm maintenance, ease of deployment in the target deployment environment, and other nonfunctional requirements such as runtime.

As with other software system development, ML development has business, technical, and project risk activities commensurate with the complexity and novelty of the system.

Model integration and deployment

During this stage, the ML algorithms and models are migrated from the development environment code, which supports fast prototyping and experimentation, to deployment target code that is more efficient and more suited to deployment environments and long-term maintenance. This process involves removing much of the code designed to support prototyping candidate algorithms and experimentation. This includes identifying the parameters and algorithm choices to be adopted and removing candidate algorithms that did not yield the desired properties or performance.

Key to this phase is modularization to isolate the inference module of the code from the remaining code. Inference modules are the components of the code relating to the forward passing of the test or previously unseen data as input through to the output of the ML subsystem. Inference refers to the forward pass execution of the subsystem, the module of the code that accepts the raw data as input and provides the output. This excludes any function relating to validating the output against the ground truth, or code involved with altering the model parameters or hyperparameters.

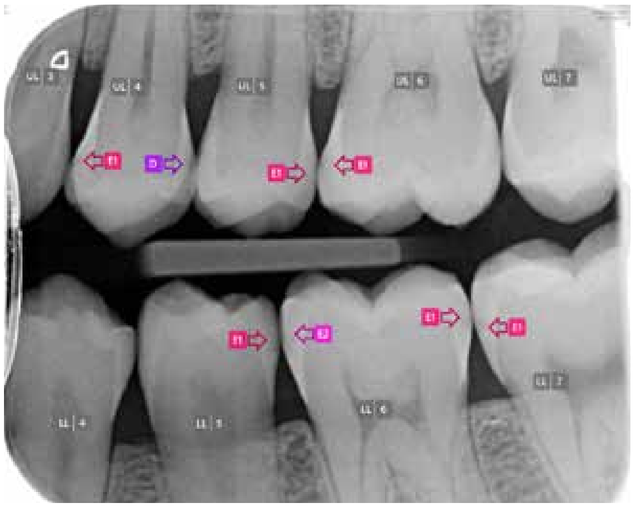
ML algorithms are typically developed in development environments tailored to support training, experimentation, and hyperparameter tuning. These environments are not always consistent with the deployment requirements, in which case porting the code and trained ML models to a runtime environment is required, along with the appropriate code review, verification, and testing. If necessary, the minimum amount of code that requires porting to the runtime environment is the inference portion. Integration also requires the specification and implementation of the interface between the ML subsystem and overarching IT system.

Similar to the inference, components of the pipeline performance evaluation exist in the training codebase. However, this needs to be implemented as a full pipeline performance evaluation, with the possibly to port it to a more suitable development and/or runtime environment.

Acceptance and release

The final infrastructure for release, maintenance, and performance verification of the ML subsystem is developed during this phase. Processes relating to the development, release, and maintenance of the subsystem is defined to specify if, how, and when the functions of developing ML algorithms are verified. Choices must be made as to whether the training and possibly tuning of the ML models are included in these processes. For example, it may be decided to run the complete model training, hyperparameter tuning, and model performance on the test data at regular instances to verify functions of the code. Alternatively, it may be deemed that the model training and hyperparameter tuning are not part of the core code or infrastructure, and are excluded from the verification process. At a minimum the process should specify, and appropriately document, how verification of the ML subsystem shall be performed. The execution of such processes should result in the release of the first version of the ML subsystem.

Figure 2: The product's graphical user interface [3].



Operation Phase

During this phase, the ML subsystem is continuously monitored and maintained. This may involve automation to alert if results deviate from predetermined limits, or may involve manual monitoring, or a combination. Performance monitoring may result in required changes affecting the subsystem. This is where the maintenance and performance evaluation processes need to be robust and sufficient to support the retraining and adoption of an alternative ML model(s). Such changes must be made in adherence with the organization's change management process, leveraging risk-based evaluation(s) considering the change's impact to current and future production data.

A typical request might be that the system is poor at generalizing to a specific subclass of input data. A typical solution is to acquire and integrate data of this subclass into the training dataset. The integration of additional training data must be systematic in that every change in the performance is measured, validated, and understood. For example, upon acquiring the additional data, a portion of it could be assigned to the training set and the rest excluded from all model training activities. Model training would proceed with the augmented training data set with the realization that the additional subclass of data may result in an overall drop in performance due to the inclusion of more challenging data. Once trained, tested, and tuned, performance of the revised model should be staged by initially executing the evaluation processes with the original model on the augmented test data set with an expectation that the performance may drop because of the increased challenge. Then, execution of the evaluation process with the revised model will take place with the expectation that the performance measures achieve the desired acceptable level.

It can be seen from this example that operation and maintenance of the system and ML subsystem are themselves iterative processes that follow the train-test-tune cycle of the original development effort, with appropriate management of new data through defined data governance and continued performance evaluation.

Case Study

This case study describes the development of an application for chair-side analysis of dental bitewing X-rays. Bitewings X-rays typically show both upper and lower teeth, including the root on the left or right side of the mouth. They are used as an aid to diagnose and monitor several conditions such as gum disease and cavities between teeth. The bitewing X-ray is taken by placing a sensor inside the mouth between the teeth and tongue, and pointing an X-ray source from the outside of the mouth. The sensor is then removed and digitally scanned to provide an image. Radiographic examinations can increase the number of carious lesions that are detected over those that would be detectable by clinical examination alone; this use is recommended by the UK Department of Health in the FGDP (UK) guideline document [4]. Nevertheless, systematic reviews have consistently reported poor diagnostic sensitivity of only 37% for radiographic detection of demineralization by dentists [5].

"The purpose of the product is to detect the early stages of tooth decay, known clinically as caries. Early caries are indicated by subtle changes in the appearance of the outer enamel surface of the tooth in bitewing x-rays. These small changes are challenging to detect, particularly given poor lighting conditions and time pressures present in a working dental practice. Not finding early-stage caries is a missed opportunity for using preventative treatments, such as interdental cleaning and resin infiltration, and is likely to lead to further decay and the need for restorative treatments such as drilling and infiltration" [6].

The product deploys a series of algorithms to analyze bitewing for early decay and highlights areas that merit a closer look by the dentist arrows indicated regions where the AI has detected image biomarkers that are indicative of early caries. Control in the graphical user interface allow dentists to move, delete or add arrows [3].

The product is provided in multiple forms: as a stand-alone application, integrated into the dentist's existing image management software, or a web-hosted analysis service. Under the EU Medical Device Directive [7], it is registered as Software as a Medical Device of class 1 safety to be used by qualified dentist practitioners to aid in the diagnosis of early enamel-only caries. The product is developed and released according to ISO 13485 standards.

The business opportunities and health benefits for an early caries detector are in minimal-intervention or minimal-invasive dentistry. This is a pioneering approach to dentistry where early preventive actions are favored to preempt and minimize the use of traditional drill and fill treatments. Thus, instead of waiting until the caries or decay has penetrated deeper into the tooth to merit drilling, the disease is detected early when decay is limited to the outer enamel surface, so it can be repaired by noninvasive treatments such as high-fluoride toothpaste or a hygienist visit [8].

The product's functional requirements were formulated to describe an assistive tool that highlights evidence of decay for better-informed decisions on diagnosis and treatment paths. An assistive diagnosis function was chosen to dovetail into the dentist's existing work path. The assistive nature also had regulatory safety class implications in that the product can only be used by trained clinicians who are always given the final decision on diagnosis and treatment path. A key requirement for the product was that it provided clear indications to allow the clinician to make better-informed decisions but did not attempt to get in the way, otherwise interfere with, or replace the clinician's actions.

The product was also required to fit seamlessly into the dentist's clinical workflow. Once the need for a bitewing had been identified, the workflow involved acquiring a pair of bitewings, one for each side of the mouth, and analyzing the X-ray immediately on their chair-side computer so that patients could be informed of the chosen treatment path immediately. Note: Typically, dentists have only a short time to analyze each bitewing during which time they look for a range of conditions in addition to early caries. To this end, a fully automated analysis was required that highlighted regions of the bitewing that were early caries.

Nonfunctional requirements included working on a chair-side computer, usually a PC without specialist hardware or reliance on an internet connection. The analysis time also needed to be fast and not add to the dentist's current image analysis and clinical reporting time of 20 seconds.

Regarding caries detection performance, previous research demonstrated that general practice dentists detect approximately 40% of early caries [9, 10]. The performance objective was to increase detection rate without an unacceptable increase in false detections (i.e., false positives). False detections are undesirable, but since treatment paths are noninvasive, they were not harmful to patients.

A comprehensive search for relevant literature and published material on technologies relating to image analysis of dental bitewings was performed, with focus on evidence of implementation and performance. This yielded information on the application of deep learning algorithms to the analysis of dental images, but there were no published case data on bitewings. The lack of a significant published data set indicated the need for data acquisition to be conducted as part of the project, and thus a requirement that the ML algorithms used should not need a large sample size to generate a predictive model with suitable accuracy.

A task was undertaken to acquire an initial set of data which consisted of 130 bitewings collected from a single site (i.e., dental office/practice), selected to have higher prevalence of proximal caries. At the onset of the project, it was decided that for general dental practitioners to gain maximum benefit, the product would have to mimic the analysis of experts in dental bitewings. (Maxillofacial radiologists are the clinical experts in analyzing dental medical images and are adept at finding early enamel-only proximal caries in bitewings and distinguishing them from other pathologies or image artifacts.) To this end, five dento-maxillofacial radiologists were recruited; each one analyzed every image and recorded the location of proximal caries together with the severity of the caries on an internationally recognized four-point grading scale. Consolidation of the experts' analyses provided a single "gold standard" data set.

This case study describes the development of an application for chair-side analysis of dental bitewing X-rays.

Due to the challenge of finding small pathologies in X-rays, the ML subsystem was designed as a pipeline of algorithms to utilize the larger features of the images. This prototype formed the backbone of a simple, interactive product demo, which allowed business collaborators and potential customers to provide images of their own as inputs and evaluate the results. This provided valuable user feedback to guide subsequent development. It also demonstrated the need for the solution to be generic to images from all acquisition hardware, hence the need for the project to collect general practice data (which occurred during the project/production phase).

The prototype performance report contained detailed descriptions of the training and evaluation methods together with all experiments executed with quantitative performance measures and qualitative evaluations, including illustrations of the failure modes. This report also included a considered prediction of the performance of the product. (Intellectual property analysis determined the freedom to operate and identify novel IP to be considered for protection.)

Results from the prototype demonstrated that the algorithm performed poorly on X-rays collected from other sites. A subsequent ethically approved clinical data collection project was initiated to collect images from 10 general dental practices and have them annotated by a panel of maxillofacial radiologists. Once collected and annotated, a test set consisting of 20% of the images was selected by random selection, stratified over the sites that were excluded from all training and validation of the models and evaluation of the pipeline to provide the unbiased data set for evaluation of the final ML subsystem and end product.

Based on the prototype experiments, the ML subsystem design in the case study consisted of three distinct components: (a) pattern recognition tooth detector; (b) identification of a superset of candidate locations of caries by dynamic programming; and (c) deep learning model for classification of candidates as either early caries or other. Risk analysis of these components identified the final stage as being the highest risk due to the more innovative approach of using neural networks compared to pattern matching and the potential need for a large training data set to ensure sufficient detection performance. To this end, mitigating strategies were developed, such as identifying less data-hungry alternative classifiers or acquiring additional data.

The tooth detector and caries classifier both contained ML models, but for brevity we describe the planning for the final component: the deep learning classifier. The Python language and development environment was chosen to develop the deep learning classifier due to its support for rapid prototyping of ML models and availability of third party, convolutional neural network support libraries.

The data format was specified for each component. Focusing on the final component, the input was a set of candidate locations along the proximal tooth edges generated by the preceding component as indicative of caries, with the output specified as a probability of measure for whether a candidate is classified as having caries. These candidate locations and confidence measures could be evaluated by comparing them to the expert gold standard of caries annotations. The model's output was classified as true positives if they resided on the same proximal edge as an expert's caries annotation within a distance tolerance. This allowed for the construction of a receiver operator characteristic (ROC) curve accumulated for all candidate locations and for all potential threshold values, with the area under the curve (AUC) used as the principal performance metric for evaluating and comparing model performances.

For the task of object detection of early caries on the proximal edges of teeth, two candidate machine learning algorithms were identified: (a) random forest classification and (b) deep learning classifier.

Both required input data as a set of candidate locations along the surface edge. Ground truth data consisted of a classification if each point was non-carious or carious with a subsequent subclassification of carious regions as being enamel caries (i.e., caries that had only penetrated the outer enamel of the tooth) or dentine caries (i.e., caries that had progressed further into the tooth dentine). The performance metric was determined as the area under the ROC curve for the classification of enamel caries.

Model development followed the regular ML life cycle of iterative training, testing, evaluation, and hyperparameter tuning. Five-fold cross-validation stratified over the image source sites was used to divide the data into training and validation data sets. The approach for hyperparameter tuning involving careful recording of hyperparameters and results for each iteration was to manually identify the changes that had a positive effect on the performance and tweak the parameters accordingly for the next iteration, comparing performance metrics and performing qualitative evaluation of the results.

As the experiments progressed, it became evident that the deep learning classifier solution offered superior performance to the random forest classification and focus turned to choosing the best hyperparameters for that model.

The product specification required running the product on a regular PC without internet connection or bespoke hardware or additional software. To fulfill this nonfunctional requirement, the inference module of the ML subcomponent developed in Python was ported to C++ components of pipeline augmented with .net API. It excludes all modules related to evaluation, model training, and hyperparameter tuning. Deep learning models were ported from Python (TensorFlow) to ONNX format and runtime inference code written in C# using Microsoft ML support libraries. All design, maintenance, and release activities of the product were audited against ISO 13485 standards.

For the product, in addition to the inference module, code to validate the performance of the runtime inference was ported to the build and test pipeline environments. This enabled automated testing and validation of the model performance during the code build cycle. A design decision was made to develop a fully integrated performance evaluation reporting into the testing and release process. The interface was augmented with the ability to supply test images and compare the results with the gold standard annotations. Selenium UI testing was used to drive the tests which were integrated and automated in the testing components of the release pipeline. Model training and hyperparameter tuning functions were not considered core to the product and were excluded from the software maintenance and performance validation processes.

To provide additional validation of the product's performance, an ethically approved clinical study to investigate whether the ability of dentists to detect enamel-only proximal caries was enhanced using the product. The study reported that dentists using the product found 75.8% of the early caries compared with only a 44.3% detection rate for dentists using the bitewing X-ray image without AI assistance, a statistically significant increase in sensitivity of 71% [3].

During the study and subsequent early adopter usage, the need to train and educate users to gain maximum benefit from the product became evident. This type of interactive AI system, where the clinician is an integral part of the AI workflow loop, requires the human to examine the regions of interest suggested by the AI system but not blindly accept them as truth. Instead, users applied their clinical knowledge and judgment when making diagnostic decisions. Training material was produced to present users with the specificity and sensitivity performance measures of the product and discussions on how to best interpret the output to enable dentists to make better diagnosis and treatment decisions for their patients.

CONCLUSION

This article explored life-cycle activities for ML components or subsystems in regulated life sciences using an example of a SaMD product. It has illustrated the life cycle and management of the ML subsystem or components within a wider system or application life cycle. Such usage of ML is occurring throughout the pharmaceutical life cycle from drug discovery and clinical development, to post-licensure product surveillance and real-world data analytics.

References

1. International Society for Pharmaceutical Engineering. *GAMP® Good Practice Guide: Regulated Mobile Applications*. North Bethesda, MD: International Society for Pharmaceutical Engineering, 2014. https://ispe.org/publications/guidance-documents/gamp-good-practice-guide-regulated-mobile-applications

2. International Society for Pharmaceutical Engineering. *GAMP Records and Data Integrity Good Practice Guide: Data Integrity by Design*. North Bethesda, MD: International Society for Pharmaceutical Engineering, 2020. https://ispe.org/publications/guidance-documents/gamp-rdi-good-practice-guide-data-integrity-design

3. Devlin, H., T. Williams, J. Graham, and M. Ashley. "The ADEPT Study: A Comparative Study of Dentists' Ability to Detect Enamel-Only Proximal Caries in Bitewing Radiographs With and Without the Use of AssistDent Artificial Intelligence Software." *British Dental Journal* 231, (2021): 481–85. doi:10.1038/s41415-021-3526-6

4. Horner, K. and, K. A. Eaton (Eds.). *Selection Criteria for Dental Radiography, 3rd ed*. London: Faculty of General Dental Practice (UK), 2013.

5. Mejàre, I., H. G. Gröndahl, K. Carlstedt, A. C. Grever, and E. Ottosson. "Accuracy at Radiography and Probing for the Diagnosis of Proximal Caries." *Scandinavian Journal of Dental Research* 93 (1985): 178–84.

6. Yu, O. Y., W. Y. Lam, A. W. Wong, D. Duangthip, and C. H. Chu. "Nonrestorative Management of Dental Caries." *Dentistry Journal (Basel)* 9, no. 10 (2021):121. doi:10.3390/dj9100121

7. European Union. Regulation (EU) 2017/745 of the European Parliament and of the Council of 5 April 2017 on Medical Devices, Amending Directive 2001/83/EC, Regulation (EC) No 178/2002 and Regulation (EC) No 1223/2009 and Repealing Council Directives 90/385/EEC and 93/42/EEC. (5 May 2017). https://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:32017R0745

8. Banerjee, A., J. E. Frencken, F. Schwendicke, and N. P. T. Innes. "Contemporary Operative Caries Management: Consensus Recommendations on Minimally Invasive Caries Removal." *British Dental Journal* 223, no. 3 (2017): 215–22. doi:10.1038/sj.bdj.2017.672

9. Mejàre, I., H. G. Gröndahl, K. Carlstedt, A. C. Grever, E. Ottosson. "Accuracy at Radiography and Probing for the Diagnosis of Proximal Caries." *Scandinavian Journal of Dental Research* 93, no. 2 (1985): 178–84. doi:10.1111/j.1600-0722.1985.tb01328.x

10. Machiulskiene, V., B. Nyvad, and V. Baelum. "Comparison of Diagnostic Yields of Clinical and Radiographic Caries Examinations in Children of Different Age." *European Journal of Paediatric Dentistry* 5, no. 3 (2004): 157–62. https://pubmed.ncbi.nlm.nih.gov/15471524/

Bibliography

International Society for Pharmaceutical Engineering. *GAMP® 5: A Risk-Based Approach to Compliant GxP Computerized Systems, 2nd Edition*. ISPE: North Bethesda, Maryland (2022). https://ispe.org/publications/guidance-documents/gamp-5-guide-2nd-edition

US Food and Drug Administration. "Artificial Intelligence/Machine Learning (AI/ML)-Based Software as a Medical Device (SaMD) Action Plan." (22 September 2021). https://www.fda.gov/medical-devices/software-medical-device-samd/artificial-intelligence-and-machine-learning-software-medical-device

US Food and Drug Administration. "Proposed Regulatory Framework for Modifications to Artificial Intelligence/Machine Learning (AI/ML)-Based Software as a Medical Device (SaMD), Discussion Paper and Request for Feedback." https://www.fda.gov/files/medical%20devices/published/US-FDA-Artificial-Intelligence-and-Machine-Learning-Discussion-Paper.pdf

International Medical Device Regulators Forum. "Software as a Medical Device: Possible Framework for Risk Categorization and Corresponding Considerations." IMDRF Software as a Medical Device (SaMD) Working Group (18 September 2014). https://www.imdrf.org/sites/default/files/docs/imdrf/final/technical/imdrf-tech-140918-samd-framework-risk-categorization-141013.pdf

BSI, Association for the Advancement of Medical Instrumentation, and Medicines and Healthcare Products Regulatory Agency. "The Emergence of Artificial Intelligence and Machine Learning Algorithms in Healthcare: Recommendations to Support Governance and Regulation." Position paper (2019). https://www.bsigroup.com/globalassets/localfiles/en-gb/about-bsi/nsb/innovation/mhra-ai-paper-2019.pdf

US Food and Drug Administration. "Good Machine Learning Practice for Medical Device Development: Guiding Principles." (October 2021). https://www.fda.gov/media/153486/download

About the authors

Eric Staib is the Vice President of Compliance and Quality Management at Signant Health. He has more than 23 years of pharmaceutical industry experience in various GXP areas, including direct experience and leadership for quality systems development/management, software quality engineering, information technology, and computer systems validation. He holds a BS in biology from James Madison University, an MS in quality assurance and regulatory affairs from Temple University, a graduate certificate in project management from Lehigh University, and an MBA in pharmaceutical management from Drexel University. Eric was a previously Chair of the GAMP Americas Steering Committee for ISPE, and currently chairs a Software Automation and Artificial Intelligence Special Interest Group (SIG). He has presented at numerous industry conferences in addition to having published and contributed to several concept papers, magazine and journal articles, and good practice guides. Eric has been an ISPE member since 2001.

Tomos Gwyn Williams, PhD, is the Chief Technical Officer of Manchester Imaging Ltd., and leads research and development activities. Manchester Imaging specializes in the research and development of machine learning algorithms and their deployment in software as medical device products for both in-house and as third-party providers for other companies. Tom has over 20 years of experience working in artificial intelligence and machine learning spanning both academic and commercial institutions. He specializes in inventing computer vision solutions that aid clinicians in diagnosis and treatment planning and has developed successful solutions for a range of medical conditions including dental decay, nonmelanoma skin cancers, facial paralysis, and osteoarthritis. Tom has an engineering degree from Imperial College, London, and a PhD in AI from the University of Wales, Aberystwyth. He has been an ISPE member since 2020.

Siôn Wyn, Director, Conformity Ltd., is an acknowledged expert in computer system validation and compliance, data integrity, electronic records and signatures, and international regulations in this field. He assisted the FDA as a consultant with its reexamination of 21 CFR Part 11 and was a member of the core team that produced the FDA Guidance on 21 CFR Part 11 Scope and Application. He received the FDA Group Recognition Award for work on Part 11. Wyn is the Editor of ISPE's *GAMP® 5 Guide: A Risk-Based Approach to Compliant GxP Computerized Systems*, *2nd Edition*, co-lead of the ISPE *GAMP® Guide: Records and Data Integrity*, and is a member of the ISPE GAMP Global Steering Committee, GAMP Editorial Board, and the GAMP Europe Steering Committee. Wyn received the 2006 ISPE Professional Achievement Award, which honors an ISPE member who has made a significant contribution to the pharmaceutical industry. He received the ISPE UK Fellow Award in 2016. He is a Technical Consultant to ISPE.

EI ASSOCIATES

ARCHITECTURE ENGINEERING CONSTRUCTION

SINGLE SOURCE RESPONSIBILITY

From Benchtop to Distribution...

We Understand Your GxP Facilities

- Research
- Laboratories
- Process Scale-Up
- Finishing
- Pilot Plants
- Process Manufacturing: Batch / Continuous
- Sterile & Non-Sterile Manufacturing
- Clean Rooms
- Packaging Suites
- Warehouse Facilities
- Support Areas
- Plant Utility Systems
- API

8 Ridgedale Avenue, Cedar Knolls, NJ 07927 ph.:973-775-7777

Contact: guy_cipriano@eiassociates.com – ext. 378

www.eiassociates.com