NASA/TM-20220017137

A Robust Machine Learning Schema for Developing, Maintaining, and Disseminating Machine Learning Models

Brandon L. Hearley, Steven M. Arnold, and Joshua Stuckner
Glenn Research Center, Cleveland, Ohio

December 2022

NASA STI Program . . . in Profile

Since its founding, NASA has been dedicated to the advancement of aeronautics and space science. The NASA Scientific and Technical Information (STI) Program plays a key part in helping NASA maintain this important role.

The NASA STI Program operates under the auspices of the Agency Chief Information Officer. It collects, organizes, provides for archiving, and disseminates NASA's STI. The NASA STI Program provides access to the NASA Technical Report Server—Registered (NTRS Reg) and NASA Technical Report Server—Public (NTRS) thus providing one of the largest collections of aeronautical and space science STI in the world. Results are published in both non-NASA channels and by NASA in the NASA STI Report Series, which includes the following report types:

- TECHNICAL PUBLICATION. Reports of completed research or a major significant phase of research that present the results of NASA programs and include extensive data or theoretical analysis. Includes compilations of significant scientific and technical data and information deemed to be of continuing reference value. NASA counter-part of peer-reviewed formal professional papers, but has less stringent limitations on manuscript length and extent of graphic presentations.

- TECHNICAL MEMORANDUM. Scientific and technical findings that are preliminary or of specialized interest, e.g., "quick-release" reports, working papers, and bibliographies that contain minimal annotation. Does not contain extensive analysis.

- CONTRACTOR REPORT. Scientific and technical findings by NASA-sponsored contractors and grantees.

- CONFERENCE PUBLICATION. Collected papers from scientific and technical conferences, symposia, seminars, or other meetings sponsored or co-sponsored by NASA.

- SPECIAL PUBLICATION. Scientific, technical, or historical information from NASA programs, projects, and missions, often concerned with subjects having substantial public interest.

- TECHNICAL TRANSLATION. English-language translations of foreign scientific and technical material pertinent to NASA's mission.

For more information about the NASA STI program, see the following:

- Access the NASA STI program home page at http://www.sti.nasa.gov

- E-mail your question to help@sti.nasa.gov

- Fax your question to the NASA STI Information Desk at 757-864-6500

- Telephone the NASA STI Information Desk at 757-864-9658

- Write to:
  NASA STI Program
  Mail Stop 148
  NASA Langley Research Center
  Hampton, VA 23681-2199

NASA/TM-20220017137

A Robust Machine Learning Schema for Developing, Maintaining, and Disseminating Machine Learning Models

Brandon L. Hearley, Steven M. Arnold, and Joshua Stuckner
Glenn Research Center, Cleveland, Ohio

National Aeronautics and
Space Administration

Glenn Research Center
Cleveland, Ohio 44135

December 2022

Acknowledgments

The authors would like to acknowledge the support of the NASA Transformational Tools and Technologies (TTT) project within the Aeronautics Research Mission Directorate.

This work was sponsored by the
Transformative Aeronautics Concepts Program.

Trade names and trademarks are used in this report for identification only. Their usage does not constitute an official endorsement, either expressed or implied, by the National Aeronautics and Space Administration.

Level of Review: This material has been technically reviewed by technical management.

This report is available in electronic form at https://www.sti.nasa.gov/ and https://ntrs.nasa.gov/

A Robust Machine Learning Schema for Developing, Maintaining, and Disseminating Machine Learning Models

Brandon L. Hearley, Steven M. Arnold, and Joshua Stuckner
National Aeronautics and Space Administration
Glenn Research Center
Cleveland, Ohio 44135

Summary

Recent advances in the development of machine learning (ML) algorithms have enabled the creation of predictive models that can improve decision making, decrease computational cost, and improve efficiency in a variety of fields. As an organization begins to develop and implement such models, the data used in the training, validation, and testing of ML models, the model parameters, and the use cases or limitations of the models must be properly stored to ensure models are both fully traceable and used correctly. In the context of predicting material behavior, advances in computationally intense, physics-based modeling of material behavior at various length scales and the emergence of Integrated Computational Materials Engineering (ICME) have driven the need for developing data-driven surrogate models of the physics-based simulation tools using ML techniques. Surrogate model development allows for accurate material behavior prediction at a fraction of the cost of its physics-based counterpart, allowing for multiscale simulations of real-world applications, further enabling the ability to design fit-for-purpose materials for a reasonable computational investment. However, training such models requires extensive data, and thus, effective data management is necessary to reach the full potential that ML can offer to material design and ICME.

This paper proposes a generalized, robust schema that allows organizations to store both real (experimental) and virtual (simulation) data used to train ML models and the defining model parameters and architectures within the Granta MI Platform. The developed schema allows for various types of data inputs and outputs, including single point values, time-series data, and images that can be used in the prediction of material behavior, while following outlined best practices for effective data management. An effective schema for ML data and models can help prevent the recreation of virtual/real training data and surrogate models, help reduce the time to create new models similar to existing ones by offering a starting point in the hyperparameter determination stages, minimize resources devoted to verification and validation (V&V) and certification of models, and ensure that data and surrogate models are not misused due to full traceability of both the data and ML model. It also allows organizations access to models that have already been developed, such that they can be used in the design of new materials, enabling the overall goals of ICME.

1.0 Introduction

The desire to reduce the cost and time to produce new materials, coupled with the desire to effectively design new materials to a specific application for superior performance, has led to the emergence of ICME (Integrated Computational Materials Engineering) as a fast-growing discipline in material science and engineering. The goal of ICME is to replace expensive and time-consuming experimentation and testing with effective simulation techniques that can surpass the limitations associated with traditional design methods (Ref. 1). ICME is predicated on linking material models at multiple time steps and length/time scales and using information from each previous scale to determine the effective properties,

allowables, or response at the next higher scale, allowing the design of a material to be tailored to the performance of a specific engineering application (Ref. 2). Therefore, ICME is heavily dependent on available data, requiring material information at various length scales that can be used to build cost-efficient simulation tools, which can in turn outline the parameters available to effectively design application-driven materials.

The simulation tools that ICME depends on for its implementation typically are either calibrated from available experimental data or solved through physics-based models. Calibrated models from experimental data require extensive material testing, which is both expensive and impractical in the design stages for new fit-for-purpose materials. Physics-based models can address these shortcomings by providing a means of predicting material behavior without testing the physical material, thus lending itself to be a better tool for ICME. However, as the complexity of a simulation grows, particularly as the analysis moves down the various length scales, physics-based models can become too computationally expensive to provide accurate predictions in a realistic timeframe. The limitations associated with physics-based models have been recently addressed using machine learning (ML), in which a surrogate model for a physics-based model is developed by training on virtual data created by the physics-based model (Refs. 3 to 7). Though surrogate models have shown incredible promise in replacing physics-based modeling, allowing for faster and more complex multiscale analyses of materials, one requirement of training such models is a large data set. However, such models are not without their own unique limitations. Since surrogate models are "learning" to mimic a physics-based model and no actual physics are incorporated in their construction, the accuracy of the surrogate models is limited to that of the model(s) and/or data used for training and the bounds associated with the data used as inputs and outputs. As a result, surrogate models are more restricted in their application to real-world problems compared to their physics-based counterpart, which can normally be applied to various types of problems. Furthermore, the data sets for ML models can range anywhere from tens of thousands to hundreds of thousands of examples, and generally contain virtual data (created by the physics-based model to be replaced) and/or experimental data along with their respective metadata as well. Furthermore, within the context of material behavior, various types of inputs and outputs can be used or required for accurate predictions, and thus the data required to train different models can vary. For example, some simpler models may require inputs of constituent material properties and predict homogenized effective properties at the next highest length scale, in which case a simple Artificial Neural Network (ANN) will suffice. Other models may predict history dependent behavior, in which the inputs and outputs vary with time, requiring a Recurrent Neural Network (RNN), such as models that predict audio and text sequences, time-series forecasting, or time-dependent material properties. Furthermore, behavior could also be dependent on analyzing images of a material to understand the structure at lower length scales. In those cases, Convolutional Neural Networks (CNN) would be required. Due to the high variability that can exist between different machine learning models, the variability that can exist between the types of data used for training and the vast amount of data required to train a surrogate model, there is an inherent need for a robust means of storing data used to develop ML models that will enable an organization to effectively make use of surrogate model development in material design.

There are many existing databases currently available, both free and paid for, that offer the ability to store material information that can be used to develop ML models to aid material design. Many of the existing available databases, however, are either specific to certain subsets of materials, including specific databases for alloys (Refs. 8 and 9), nanomaterials (Refs. 10 and 11), and materials with crystalline structures (Refs. 12 and 13) be they real or virtual data but not both. Additionally, both material-specific and general material databases only give the ability to store material information, rather than provide both the means to store data and link that data to developed, data-driven models. Using such a database would

then require the user to manually link data to a given model and use some other database to store the model parameters and information. The Granta MI database platform looks to bridge this gap by allowing users to define not only the data present in the database, but also the *attributes* for a specific subset of data, denoted as *tables* in Granta, in each data *record*, thus allowing users to define both material and model information. The ability to link records across different tables gives users the means to directly connect their data to the models developed without relying on copying the data and storing it in another location.

In this paper, a proposed schema for storage of ML data and model parameters is presented in the context of a larger ICME schema. Rather than explain the development of an individual surrogate model, the goal of this paper is to provide a robust framework for storing various types of ML models, their associated data (albeit real or virtual), and the traceability between model development and parameters used such that users within an organization can understand the models' availability, their use, and their limitations. The ML schema developed fits into the existing Materials Database information management system used by NASA Glenn, presented in previous publications (Ref. 14). The current schema, shown in Figure 1, is designed to operate within the Granta MI system and addresses techniques for effectively storing generalized data that can be used for ML model development within the limitations of the Granta platform. With such a system in place for ML data, organizations can efficiently find existing characterized surrogate models that have been created, determine the traceability of any model previously developed, and find existing data that may be used to train further models, which can significantly reduce the time and cost associated with using surrogate models for material decisions.

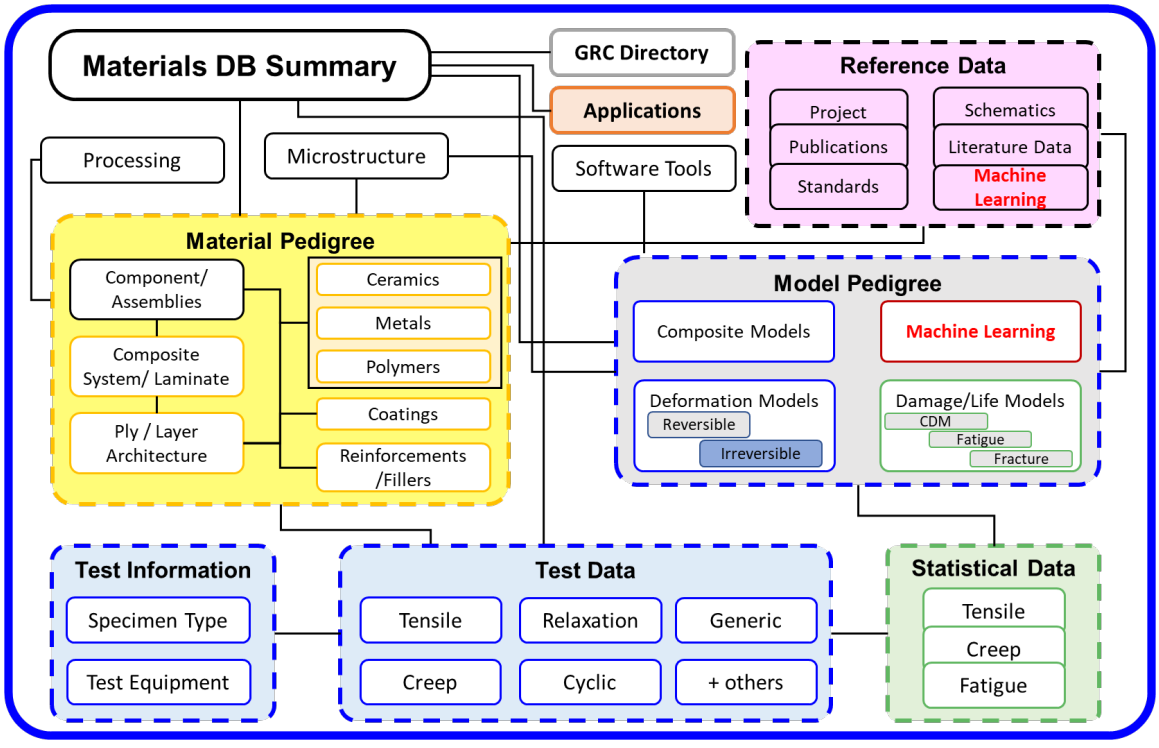


Figure 1.—NASA Glenn's ICME schema modified for machine learning.

2.0 Database Management Best Practices

When designing the schema for ML data and models, it is important to keep in mind the best practices previously defined during the development of the larger ICME schema (Refs. 2 and 15) as well as principles previously established for data management, such as the FAIR principles (Ref. 16). Particularly to ML models, the associated best practices that influence the schema design are capture, traceability, and accessibility.

2.1 Capture

The developed schema must be able to capture information fundamental to defining the ML model and the model pedigree, including the associated data used to train, validate, and test the model. The schema must be robust enough to handle different types of data (single point, time series, images, etc.). Furthermore, the data must be captured and organized such that it is easily findable and is described uniquely to be distinguishable from similar yet different data in the database.

2.2 Traceability

The model pedigree must have high traceability such that it is inherently clear what data is used to define an ML model. Traceability is particularly important for ML models, since the relation between inputs and outputs is not necessarily based on physics-based constitutive equations and can act as a "black box." Thus, it is essential to understand the full development of the model to know its limitations.

Traceability also encompasses the database principle that data should be defined in only one place in the database and viewed (copied) elsewhere. By following this best practice, if data needs to be changed or updated, the change need only occur once, allowing all viewed instances of the data to be automatically updated, thus ensuring the traceability of any associated pedigrees is maintained.

2.3 Accessibility

For various reasons, data may need to be controlled such that only those who are authorized to view it may see it within the Granta MI platform. In the current NASA GRC schema, an *attribute-based* access control is used, such that each attribute within a record is assigned a security control. This methodology allows the schema administrator to restrict access to specific attributes to some users while allowing them to see the remaining data. Within the context of ML, this feature is particularly important in which data used for training may be restricted, but the resultant model may be viewed (or vice versa).

Within the organization, or within the specified group who is granted access to the database, data should be easily accessible and usable. This includes both the relative ease of users to upload data to the database, view data in the database, and download or output data in machine-readable forms such that it can be used to drive decisions or, in the case of ML, create new surrogate models without repeating the data creation, which generally is the most consuming component of the development of ML models.

3.0 Overview of Granta MI

Granta MI is a database software system offered by ANSYS that allows users to capture, manage, and share material information within an organization (Ref. 17). Within Granta, users capture and store specific data by assigning different *attributes* to be populated within a given *record*, *folder*, or *table*. The collection of attributes and their organization (e.g., attributes are often collected into sections under specific header names) is called the layout. For flexibility and organization, Granta allows users to design different schema for different types of data. The highest level of organization in Granta is called a *table*, where all records within a table have the same layout. Records can further be organized into *folders* or *generic records*. Folders simply contain records and have no other associated data, similar to a computer directory tree. Generic records are a combination of a record and a folder—they have the schema attributes that can be populated for the generic record and can also contain different records within. An example of a Granta MI tree structure and an associated record/generic record are shown in Figure 2.

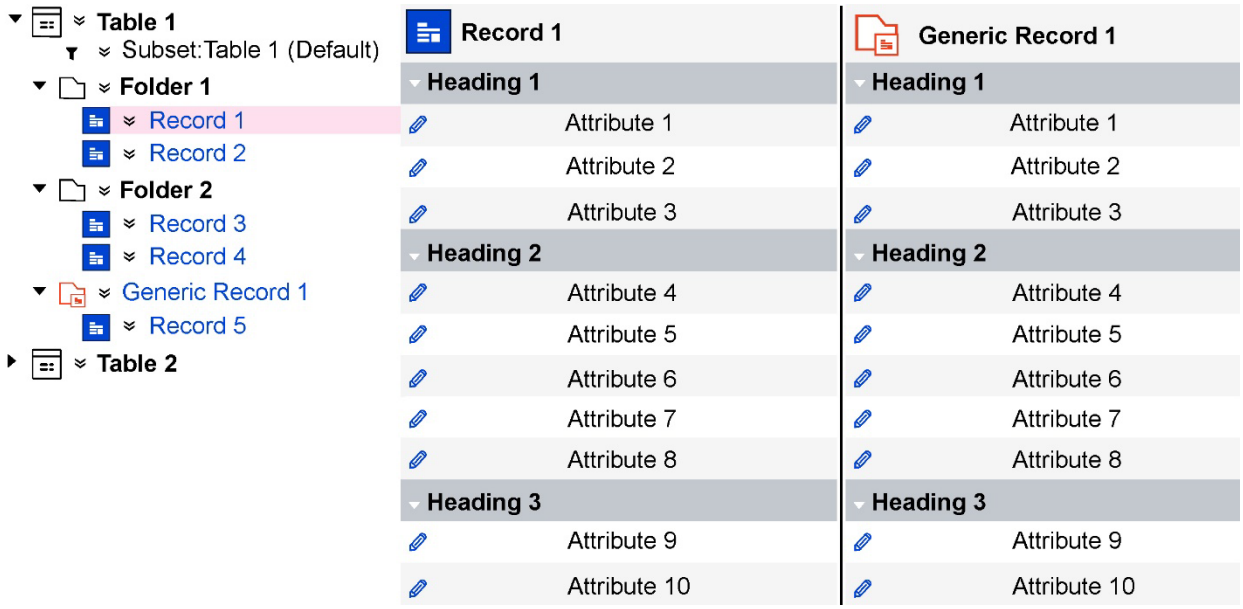
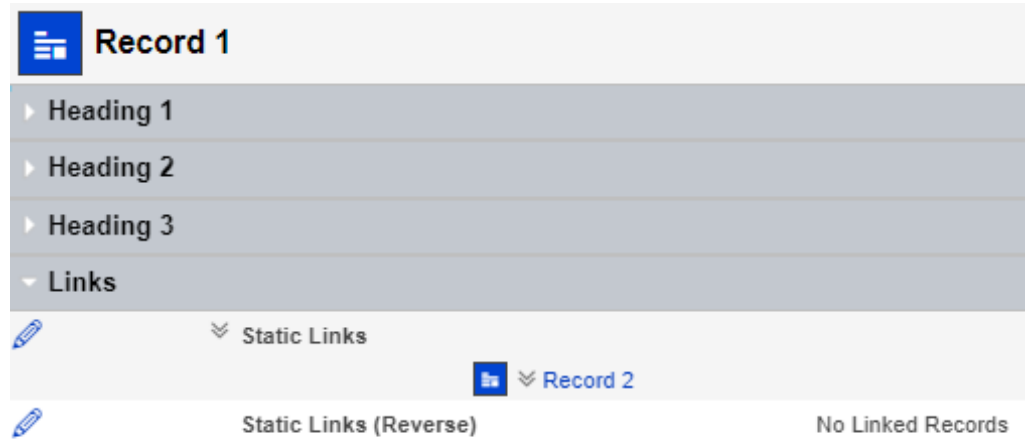


Figure 2.—Granta MI definitions.

Granta MI also offers the ability to link records together for traceability of data across different records. Links can either be *static*, where a record is explicitly defined by the user as linked to another record, or a *smart link*, in which the user can define attribute-based criteria to automatically determine the linking relationship between two records. Links in Granta can also be either one-way or two-way links. For instance, in the example in Figure 3, the user defined Record 2 as being linked to Record 1, shown in Figure 3(a). Clicking on the link brings the user to Record 2, shown in Figure 3(b). Although not explicitly defined, the reverse direction link is automatically populated in the record, showing that Record 2 is linked to Record 1 as well.



(a)

(b)

Figure 3.—Linking directionality in Granta MI. (a) Record 1. (b) Record 2.

Along with viewing direct links to different records in Granta, users can view data in another record within the same database, even if the two records are in different tables, through a tabular attribute type. Therefore, if the record gets updated or changed where the data is defined, the changes will automatically propagate to the record where the data is viewed. When viewing data in a tabular attribute, records are linked together by the "Linking Value," which is the value of a specified attribute that Granta MI uses to populate the viewed table. In Figure 4, data from Record 1 (Figure 4(a)) will be viewed in a record named "View of Record 1." When defining the link between the two records on the *back end*, or the window the user sees when editing the attribute properties (Figure 4(b)), the user enters the value of the defined linking attribute. As a result, when viewing the record on the *front end*, or the window the user sees when viewing the populated record (Figure 4(c)), the attributes in Record 1 are automatically pulled and displayed in the viewed record. The linked tabular attribute in Granta MI allows the user to view data from multiple records in one location (by selecting "Add a blank row" in Figure 4(b) and entering the linking value of a different record), giving users the ability to view large data sets in one record with guaranteed traceability.

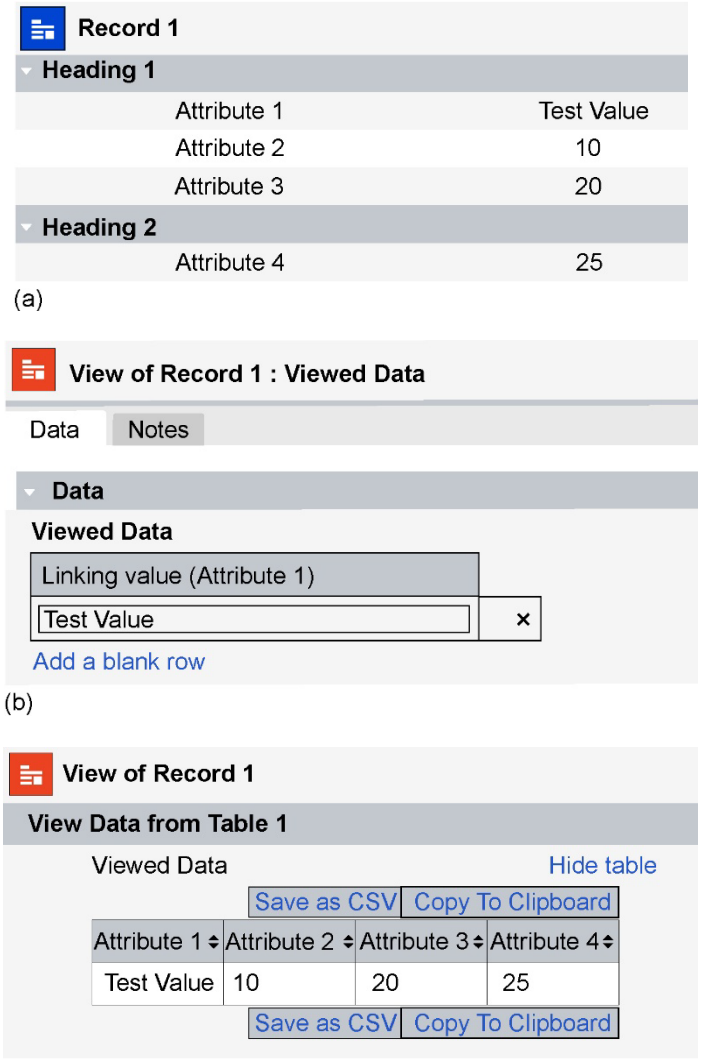


Figure 4.—Viewing linked data in Granta MI. (a) The record where data is *defined*. (b) The back end of the record where data is to be viewed. (c) The front end of the record where data is *viewed*.

4.0 Table Organization

To enable the best practices for database management outlined in Section 2.0, the data storage for ML is separated into two tables: Reference Data: Machine Learning, which contains all *virtual data* used to train, validate, and test the ML model, and Models: Machine Learning, which contains the model architecture, hyperparameters, and access. Real experimental data is placed as usual within one of the tables within the Test Data collection (see Figure 1). Within the greater ICME schema (Figure 1), the Reference Data: Machine Learning table is placed within the Reference Data collection (indicated in Figure 1 by dash-lined box) and the Models: Machine Learning table is placed within the Model Pedigree collection, thus showing the previously developed schema is robust enough to handle the addition of ML data.

The separation of the data and model parameters into two tables allows one to follow the best practice that data only be defined in one location within the database. Figure 5 shows two different example cases that illustrate this division. In the first (left) example, a ML model is trained from a combination of virtual (simulation) and real (experimental) data. The separation of data and model parameters allows data for a model to come from various sources albeit real or virtual, which enables greater flexibility when defining a model in Granta MI without repeating any preexisting data in the database. In the second (right) example, the same virtual data set is used to define two different surrogate models. In this case, the separation of data and model parameters prevents the virtual data from being repeated in the database in multiple locations. Not only does this help maintain the traceability of each model, but it also reduces the storage space necessary for capturing the full model pedigree of the two surrogates, which can be particularly large for ML models. Furthermore, the current schema best enables the ability to track/record model pedigree (e.g., idealization assumptions, mechanism, loading conditions, correlation data, etc.) and V&V results so that one can ensure proper transferability, usability, reusability. This use case and the full capabilities of the designed schema will be further demonstrated in an example in Section 4.0.

In general, it is assumed that data for training, validation, and testing is *defined* in the Reference Data: Machine Learning table and *viewed* in the Models: Machine Learning table. Thus, the individual attributes within each table must be designed to enable this relationship within the limitations of the Granta MI platform. A detailed description of the attributes, layout, and linking behavior for the two tables is described in the remainder of this section.

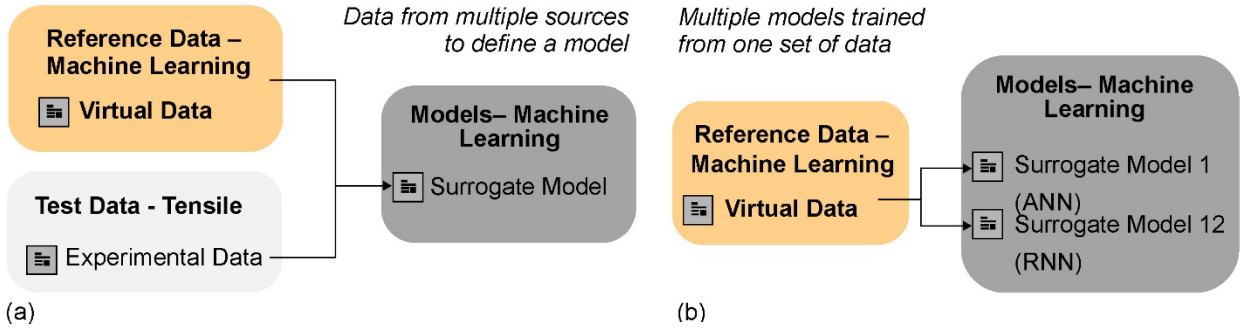


Figure 5.—Examples of the need for separating data and model parameters. (a) One model trained from multiple data sources. (b) Two models trained from a single data source.

4.1 Reference Data: Machine Learning

The purpose of the Reference Data: Machine Learning table is to allow for storage of any generalized data albeit literature or virtual such that it can be properly linked to any arbitrary number of surrogate ML models that use the data for training, testing, and validation. Figure 6 illustrates the current layout and attributes in the Reference Data: Machine Learning table.

The first header of the layout, Data Series Information, contains information associated with the purpose of the data, how it was created, who owns the data, and the current point of contact. The "Point of Contact" tabular attribute columns are defined in Table I. The first and second column define the column name and the data type for that column. The third column defines whether the attribute is linked (viewed from another table) or static (defined in the Reference Data: Machine Learning record). All linked values for the "Point of Contact" attribute are defined in the GRC Directory table in Granta MI, with the linking value as the person's name.

The final attribute in the first header of the layout contains a summary of all other data series information, with its columns defined in Table II. In the current schema, the project information is *defined* in a generic folder, and each member of the data set is placed into an individual record within that generic record. In the individual data records, rather than redefining the project information, it is *viewed* in the "Project Information Summary" attribute by linking to the generic folder's value for "Data ID."

Attribute	Type
Data Series Information	
Standard Test Description	LTXT
Data Ownership	DCT
Data Ownership (Other)	STXT
Distribution Category	DCT
Funding Organization	STXT
Project Code	STXT
Project Name	STXT
Project Notes	LTXT
Point of Contact	TABL
Original Data Filename	FILE
Project Information Summary	TABL
Data	
Data ID	STXT
Data Labels	TABL
Attribute 1	LTXT
Attribute 200	LTXT
Image 1	IMG
Image 50	IMG
Data	
Model Records	LINK
Software Tools Used	LINK

DATE	Date (DD/MM/YY)
DCT	Discrete text (specified choice)
FILE	Allows association of any file format
FUNC	Functional data (XY plot)
IMG	Allows association of any image formation
LTXT	Long text field
PNT	Point value
TABL	Tabular attribute
STXT	Short text field

Figure 6.—Reference data machine learning layout and attributes.

TABLE I.—DEFINITION OF THE “POINT OF CONTACT” TABULAR ATTRIBUTE

Column name	Type	Linked
Name	STXT	Yes
Email	STXT	Yes
Phone	STXT	Yes
Office	STXT	Yes
Mail Stop	STXT	Yes
Current Employment Status	DCT	Yes
Additional Information	LTXT	Yes

TABLE II.—DEFINITION OF THE “PROJECT INFORMATION SUMMARY” TABULAR ATTRIBUTE

Column name	Type	Linked
Summary Record	LINK	Yes
Name	LINK	Yes
Standard Test Description	LTXT	Yes
Data Ownership	DCT	Yes
Data Ownership (Other)	STXT	Yes
Distribution Category	DCT	Yes
Funding Organization	STXT	Yes
Project Code	STXT	Yes
Project Name	LTXT	Yes
Project Notes	STXT	Yes

TABLE III.—DEFINITION OF THE “DATA LABELS” TABULAR ATTRIBUTE

Column name	Type	Linked
Attribute Name	STXT	No
Label	STXT	No
Unit	STXT	No

The second layout header, Data, stores the actual data values. The “Data ID” attribute is used as a linking value to enable the viewing of the data in subsequent Model: Machine Learning records. Because data can take on many forms, including single point values, time-series values, or images, each data label that would be used as either an input or output for a surrogate ML model is stored in a generic long text attribute (“Attribute X”) or generic image attribute (“Image X”). The attributes themselves are defined using the “Data Labels” tabular attribute, in which the user enters the attribute or image number, its associated label, and the unit of that label. The “Data Labels” attribute columns are defined in Table III.

The third layout header (see Figure 6), Further Information, contains links in Granta MI to any associated records in the Models: Machine Learning table that views the data and links to any software tools used to create the virtual data set.

As stated previously, data sets are uploaded to the Reference Data: Machine Learning table as a generic record, which contains general information that applies to each example in the data set (i.e., attributes in the Data Series Information header and the "Data Labels" attribute), with individual records for each example in the data set contained within that generic record. Although this formulation results in potentially creating thousands of records for a single data set, it is necessary in order to follow the outlined best practices in Section 2.0 within the current limitations of the Granta MI platform. The alternative would be to replace "Attributes 1–200" and "Images 1–50" with a tabular attribute and allow the number of rows to be equal to the size of the data set. However, in the current configuration of Granta MI, it is not possible to link, or view, individual rows from a tabular attribute in another record. Thus, given the current system, by consolidating all of the data into a single record using a tabular attribute, the model records would lose the flexibility of being able to define training, validation, and test data and would ultimately require the data to be repeated in the database if it were to be used by more than one model. Thus, within the current limitation of Granta MI, the schema uses individual records for each data example and long text field/image attributes to enable this flexibility. A detailed example illustrating this use case and the need for the current schema is given in Section 4.0. Also, a feature request has been submitted to ANSYS to enhance the current tabular attribute capabilities to allow for a more robust implementation, explained in further detail in Section 7.0.

4.2 Models: Machine Learning

The purpose of the Models: Machine Learning table is to allow for storage of an established (i.e., developed, enhanced, tailored, etc.) surrogate model, including the creator of the model, all associated model parameters and hyperparameters, links to the data in the Reference Data: Machine Learning table used for training, testing, validation, and model performance, as shown in Figure 7.

Attribute	Type	Attribute	Type
Project Information		**Data Definition**	
Performing Organization	STXT	Data Variables Summary	TABL
Data Ownership	DCT	Data Assembly	TABL
Data Ownership (Other)	STXT	Training Data Set (View)	TABL
Distribution Category	DCT	Validation Data Set (View)	TABL
Funding Organization	STXT	Test Data Set (View)	TABL
Funding Organization	STXT	**Model Validation**	
Project Code	STXT	Training Loss	FUNC
Project Name	STXT	Validation Loss	FUNC
Project Notes	LTXT	Validation Predictions (Define)	TABL
General Model Information		Test Predictions (Define)	TABL
Model Name	STXT	Accuracy	PNT
Model Description	LTXT	Accuracy Metric	DCT
Source Code Location	HYP	Validation Notes	LTXT
Point of Contact	TABL	**Further Information**	
Version	STXT	References	LINK
Date	DATE	Data Information	LINK
Lbraries/Modules	TABL	Software Information	LINK
Model Notes	LTXT	Software Tools Implemented In	LINK
Model Architecture			
Architecture Type	STXT		
Architecture Description	TABL		
Architecture Notes	LTXT		

Figure 7.—Models: Machine learning layout and attributes.

Attributes within the Models: Machine Learning table are separated into six different sections with their own assigned headers. The first header, Project Information, defines the owner of the model and associated funding/project information for the model development. The second, General Model Information, provides information about the purpose of the model, programming languages used in development, and how a user can access the model, including the point of contact (see Table I) and the source code location. The third, Model Architecture, gives the hyperparameters and architecture of the neural network developed, so that the model could be reproduced using any ML programming package (e.g., TensorFlow, PyTorch, etc.). This allows both a means of providing one with the ability to create the model in a different ML software package and a starting point for development of a similar model that may use different training data or predict different output features. The columns for the tabular attribute "Architecture Description" are shown in Table IV.

The fourth layout heading, Data Definition, defines the data employed to create the model. It provides the training, validation, and test split of the data, and links each individual data record within the Reference Data: Machine Learning table to one of the three aforementioned categories, such that the data displayed in the "Training Data Set," "Validation Data Set," and "Test Data Set" tabular attributes are viewed from the associated linked record. Because each attribute is viewed from the Reference Data: Machine Learning record, column names for the three data set attributes take on the generic names described above. Therefore, there is an additional attribute in the model record (under this fourth heading), "Data Variables Summary," to allow for the definition of each of the attribute names and whether they are inputs, outputs, or not used in the model (see Table V). Note that in Table V, ideally the Attribute Name, Variable, and Unit would be viewed from the "Data Labels" attribute in the Reference Data: Machine Learning record. However, as described previously, it is not possible within the current configuration of Granta MI to view individual rows from a tabular attribute in another table. Therefore, in the current formulation, the data in the "Data Variables" tabular attribute is static and repeated, but the entries marked in bold with a "*" in Table V would be linked if the capabilities in Granta MI change to allow tabular row-wise linking.

TABLE IV.—DEFINITION OF THE "ARCHITECTURE DESCRIPTION" TABULAR ATTRIBUTE

Column name	Type	Linked
Label	STXT	No
Value	STXT	No

TABLE V.—DEFINITION OF THE "DATA VARIABLES" TABULAR ATTRIBUTE

Column name	Type	Linked
Attribute Name	STXT	No*[a]
Variable	STXT	No*
Unit	STXT	No*
Input/Output	DCT	No

[a]Entries marked with a "*" would be linked if the capabilities in Granta MI change to match the proposed feature request.

TABLE VI.—DEFINITION OF THE "DATA ASSEMBLY" TABULAR ATTRIBUTE

Column name	Type	Linked
Folder	LINK	Yes
Data Shuffle Equation	STXT	No
Data Shuffled	PNT	No
Training Split (%)	PNT	No
Validation Split (%)	PNT	No
Test Split (%)	PNT	No
Size	PNT	No

TABLE VII.—DEFINITION OF THE "TRAINING DATA SET," "VALIDATION DATA SET," AND "TEST DATA SET" TABULAR ATTRIBUTES

Column name	Type	Linked
Record Name	LINK	Yes
Data Source Type	DCT	No
Attribute 1	LTXT	Yes
⋮	⋮	⋮
Attribute 20	LTXT	Yes
Image 1	IMG	Yes
⋮	⋮	⋮
Image 50	IMG	Yes

The "Data Assembly" tabular attribute (in Figure 6 under the Data Definition header) is used to define how the data is split into training, validation, and test data. Each column is defined in Table VI.

In Table VI, the folder column contains a link to the generic record for the data set, with the linking value equal to the "Data ID" attribute in the Reference Data: Machine Learning table. The remaining three tabular attributes ("Training Data Set," "Validation Data Set," "Test Data Set") contain the specific model inputs and outputs for the three respective categories. Each tabular attribute has the same column structure, as defined in Table VII.

Each linked column in the three tabular attributes is automatically populated by entering the "Data ID" from the associated individual data record, within the generic record, in the Reference Data: Machine Learning table, and each row represents a different example in that set. Additionally, the user can enter whether the data in that specific row was virtual or experimental in the "Data Source Type" column, giving the user the ability to easily identify the limitations associated with training the model.

The fifth layout heading, Model Validation, allows for the definition of the model performance. Because predictions are specific to each model, the values in the tabular attributes "Validation Predictions" and "Test Predictions" are defined in the model record and are not linked to another table in the database. A definition of the columns for the two tabular attributes is given in Table VIII.

The final layout heading, Further Information, provides links to any references, data information, or software information that may have been used in the development of the model. Additionally, completed ML models that contribute to developed software are also linked to the Software Tools table. A detailed example illustrating the flexibility of the schema is given in Section 4.0.

TABLE VIII.—DEFINITION OF THE "VALIDATION PREDICTIONS" AND "TEST PREDICTIONS" ATTRIBUTES

Column name	Type	Linked
Attribute 1	LTXT	No
⋮	⋮	⋮
Attribute 20	LTXT	No
Image 1	IMG	No
⋮	⋮	⋮
Image 50	IMG	No

5.0 Example

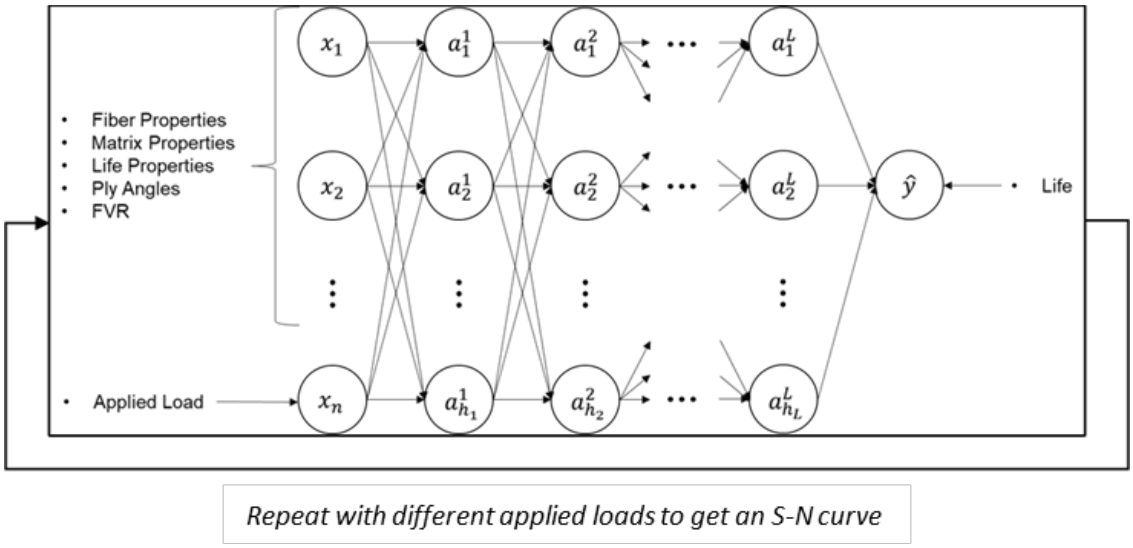
5.1 Problem Definition

To demonstrate the need for the current ML schema and its ability to store ML data and models while following the outlined best practices for database management, an example problem is herein described in detail. In this example, two different ML surrogate models are created to replace a physics-based tool that predicts the fatigue-life (S-N) curve of a polymer matrix composite given inputs of constituent mechanical properties, constituent life properties, laminate ply schedule, laminate volume fraction, and a vector of applied loads (Ref. 18). Virtual data is created using a physics-based micromechanics tool known as MAC/GMC (Refs. 18 and 19) and uploaded to the Reference Data*:* Machine Learning table in Granta MI. Two different surrogate models were trained from the same data set to reproduce the physics-based results:

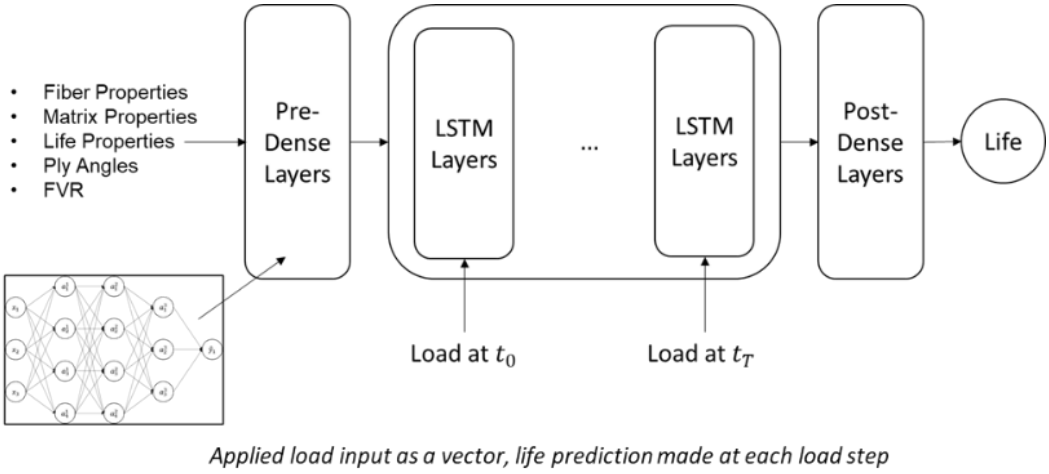
1. An artificial neural network (ANN) known as feed-forward Multi-layer Perceptron (MLP) (Ref. 20) that takes a single applied load and produces a single value of life at that load. To produce the full S-N curve, the ML model must be called at each applied load step (Figure 8(a)). This model exactly mimics the physics-based tool, in which each life calculated is independent of other applied loads.
2. A recurrent neural network (RNN) (Ref. 21) that takes the applied load as a vector and produces a vector of associated life values. To produce the full S-N curve, the ML model is called only once (Figure 8(b)). Unlike the actual physics-based tool, in which values of life are calculated at a given load independent of the solutions at other loads, the RNN formulation considers the relationship between neighboring points that is ultimately inherent to the physics-based solution.

5.2 Virtual Data

Both the ANN and RNN are trained from the same data set, and thus the virtual data is only uploaded once to the Reference Data: Machine Learning table. A generic record ("Fatigue Life Estimator Data") is created for the full data set, and individual records for each run case in the creation of the virtual data is created and placed within the generic record (see Figure 9). In the generic record, the project information, point of contact, and data ownership information is defined in the Data Series Information layout heading (Figure 10(a)). In the Data layout heading, a generic "Data ID" is defined (herein named "Fatigue Life") for linking the model records, as well as the "Data Labels" tabular attribute for defining the physical quantities each attribute represents (e.g., 21 attributes are shown in Figure 10(b)). Finally, in the Further Information layout heading, links to the two associated models in the Models: Machine Learning table are defined (see Figure 10(c)).



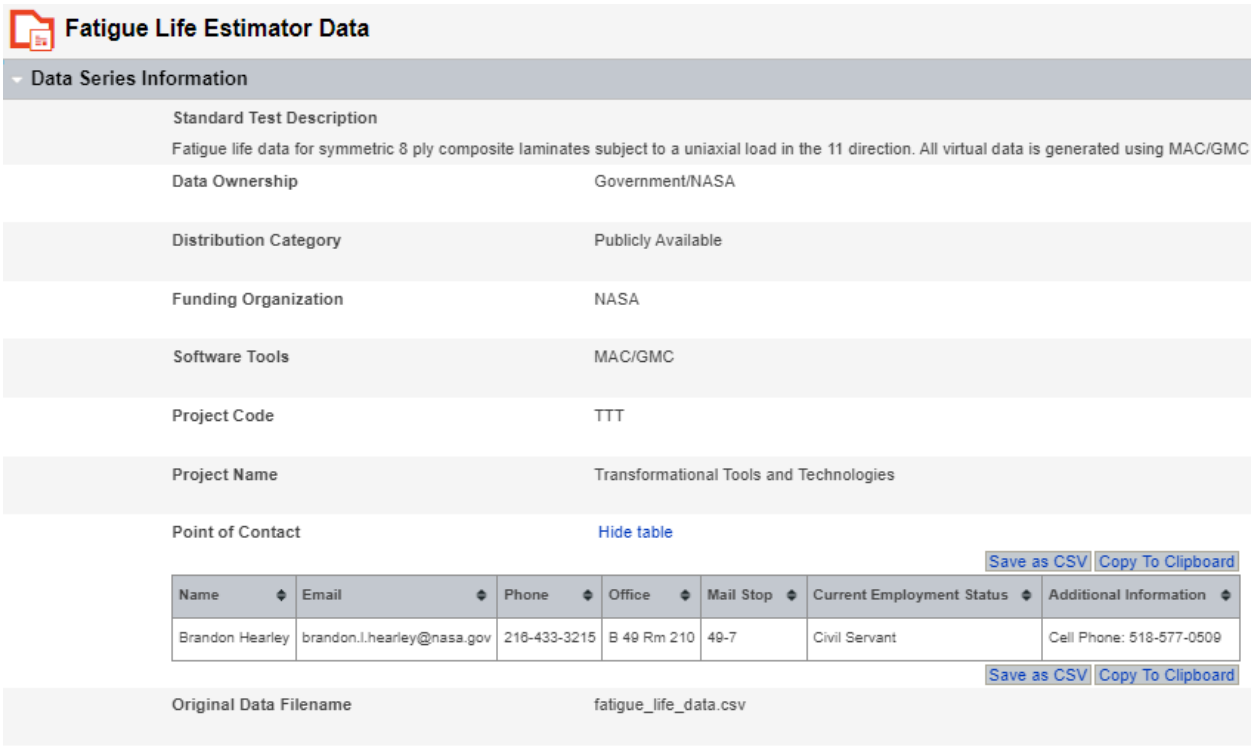
(a)



(b)

Figure 8.—The ML general architecture. (a) ANN model. (b) RNN model.

Figure 9.—Folder tree organization of the virtual data.



(a)

Figure 10.—Generic record attributes. (a) Data series information. (b) Data. (c) Further information.

Data

Data ID | Fatigue Life

Data Labels | Hide table

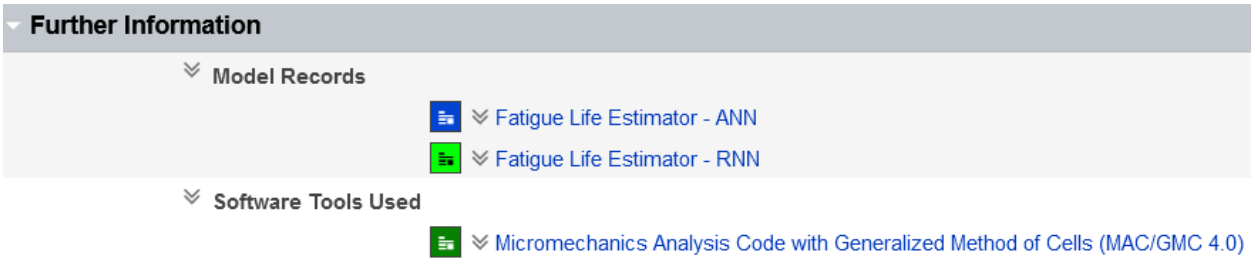
Save as CSV | Copy To Clipboard

Attribute Name	Label	Unit
Attribute 1	Fiber Longitudinal Modulus	MPa
Attribute 2	Fiber Transverse Modulus	MPa
Attribute 3	Fiber Longitudinal Poisson's Ratio	
Attribute 4	Fiber Transverse Poisson's Ratio	
Attribute 5	Fiber Longitudinal Shear Modulus	MPa
Attribute 6	Matrix Elastic Modulus	MPa
Attribute 7	Matrix Poisson's Ratio	
Attribute 8	Volume Fraction	
Attribute 9	Ply Angle 1	°
Attribute 10	Ply Angle 2	°
Attribute 11	Ply Angle 3	°
Attribute 12	Ply Angle 4	°
Attribute 13	SU11	MPa
Attribute 14	SU22	MPa
Attribute 15	Epsm1	
Attribute 16	Epsm2	
Attribute 17	Beta	
Attribute 18	SFL	MPa
Attribute 19	XML	MPa
Attribute 20	Applied Load	MPa
Attribute 21	Life	log(cycles)

Save as CSV | Copy To Clipboard

(b)

Figure 10.—Continued.



(c)

Figure 10.—Concluded.

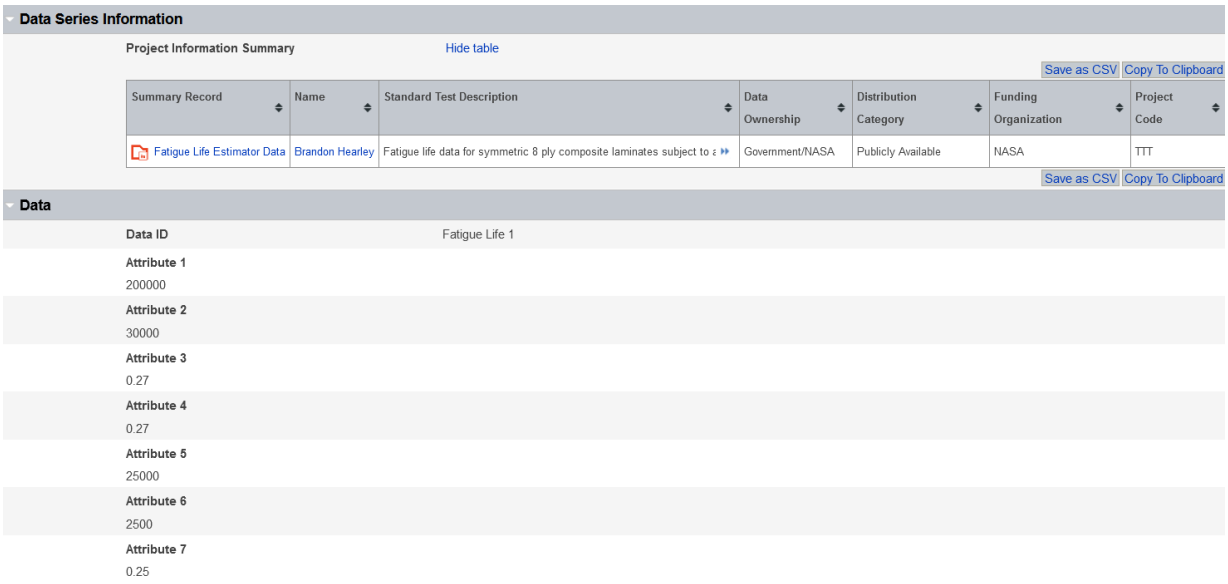


Figure 11.—Individual data record attributes.

Individual records are created for each of the different examples in the data set. Figure 11 shows an example of one of the populated records. Although all records will have the same attributes populated, their values will differ for each of the example records. In each of the records, only the "Data ID" and generic "Attribute X" are *defined*, since all other defining information is found in the parent generic record. The "Project Information Summary" tabular attribute is linked to the parent generic record and all entries in the tabular attribute are *viewed*.

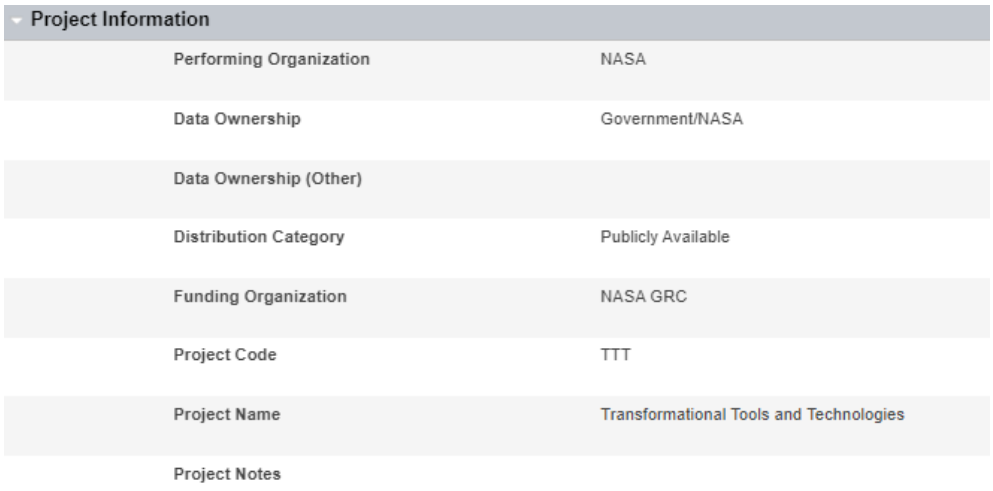
5.3 Model Data

Two different records are created in the Models: Machine Learning table: one for the ANN and one for the RNN. For clarity, figures relating to the ANN model will be outlined in blue and figures relating to the RNN model will be outlined in green. If the figure applies to both models, it will be outlined in black.

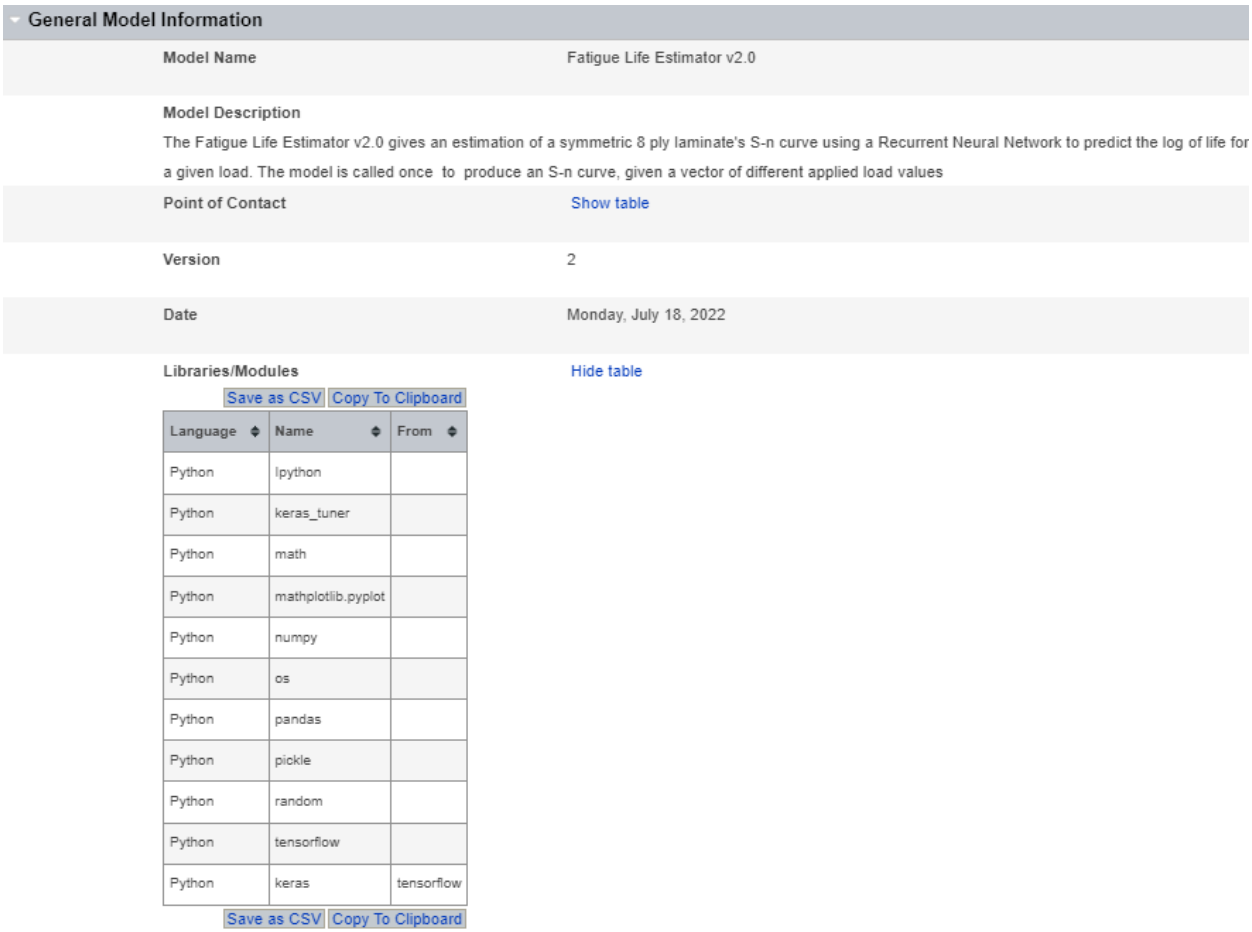
For both the ANN and RNN models, the Project Information (Figure 12(a)), and General Model Information (Figure 12(b)) are commonly defined. Note however, that in Figure 12(b), the "Model Description" would be different between the two models; Figure 12(b) shows the description for the ANN model.

Attributes under the Model Architecture layout heading are defined for both models in Figure 13. Due to the use of a tabular attribute for the "Architecture Description," the schema is able to define a model with a varying range of complexity with regard to its architecture.

Under the Data Definition layout heading, the "Data Variables Summary," "Data Notes," "Data Assembly," "Training Data Set," "Validation Data Set," and "Test Data Set" attributes are populated for both records (see Figure 14). The "Data Variables Summary" tabular attribute is the same for the two records and is a copy of the Data Labels tabular attribute in Figure 10(b) and is thus hidden for conciseness. The "Data Assembly" tabular attribute is the same for the two records except for the Data Shuffled column. For the ANN, the data was unshuffled (thus the value zero in Figure 14(a)) prior to being split into training, validation, and test data sets, whereas for the RNN, the data was shuffled with a random seed value of 42 (see Figure 14(b)). This difference manifests itself in the subsequent "Training Data Set," "Validation Data Set," and "Test Data Set" attributes. In the ANN, because the data is unshuffled, the "Training Data Set" Record Name entries appear in order (Figure 14(a)), while in the RNN the order is shuffled (Figure 14(b)). The remaining columns in both tabular attributes dynamically



(a)



(b)

Figure 12.—Model: Machine Learning Record example. (a) Project information. (b) General model information.

Model Architecture

Architecture Description — Artificial Neural Network

Architecture Type

Hide table

Save as CSV | Copy To Clipboard

Label	Value
Dense Layers	3
Dense Units	120.20.1
Dense Layer Activation	tanh,tanh,linear
Loss	Mean Square Error

Save as CSV | Copy To Clipboard

Architecture Notes

(a)

Model Architecture

Architecture Type

Architecture Description

Save as CSV | Copy To Clipboard

Label	Value
Pre-Dense Layers	1
Pre-Dense Units	100
Pre-Dense Activation	tanh
Pre-Dense Dropout Rate	0.1
LSTM Layers	2
LSTM Units	500
LSTM Activation	ReLu
Post-Dense Layers	50
Post-Dense Units	Value
Post-Dense Activation	ReLU. linear
Label	Mean Square Error
Learning Rate	1E-3.5

Save as CSV | Copy To Clipboard

Architecture Notes

(b)

Figure 13.—(a) ANN model architecture. (b) RNN model architecture.

Data Assembly Hide table

Save as CSV Copy To Clipboard

Folder	Data Shuffle Equation	Data Shuffled	Training Split (%)	Validation Split (%)	Test Split (%)	Size
Fatigue Life Estimator Data		0	80	10	10	100

Save as CSV Copy To Clipboard

Training Data Set (View) Hide table

Record Name	Data Source Type	Attribute 1	Attribute 2	Attribute 3	Attribute 4	Attribute 5	Attribute 6	Attribute 7	Attribute 8
Fatigue Life Estimation 1	Virtual	200000	30000	0.27	0.27	25000	2500	0.25	0.4
Fatigue Life Estimation 2	Virtual	200000	30000	0.27	0.27	25000	2500	0.25	0.4
Fatigue Life Estimation 3	Virtual	200000	30000	0.27	0.27	25000	2500	0.25	0.4
Fatigue Life Estimation 4	Virtual	200000	30000	0.27	0.27	25000	2500	0.25	0.4
Fatigue Life Estimation 5	Virtual	200000	30000	0.27	0.27	25000	2500	0.25	0.4
Fatigue Life Estimation 6	Virtual	200000	30000	0.27	0.27	25000	2500	0.25	0.4
Fatigue Life Estimation 7	Virtual	200000	30000	0.27	0.27	25000	2500	0.25	0.4
Fatigue Life Estimation 8	Virtual	200000	30000	0.27	0.27	25000	2500	0.25	0.4
Fatigue Life Estimation 9	Virtual	200000	30000	0.27	0.27	25000	2500	0.25	0.4
Fatigue Life Estimation 10	Virtual	200000	30000	0.27	0.27	25000	2500	0.25	0.4

(a)

Data Assembly Hide table

Save as CSV Copy To Clipboard

Folder	Data Shuffle Equation	Data Shuffled	Training Split (%)	Validation Split (%)	Test Split (%)	Size
Fatigue Life Estimator Data		42	80	10	10	100

Save as CSV Copy To Clipboard

Training Data Set (View) Hide table

Record Name	Data Source Type	Attribute 1	Attribute 2	Attribute 3	Attribute 4	Attribute 5	Attribute 6	Attribute 7	Attribute 8
Fatigue Life Estimation 56	Virtual	400000	50000	0.34	0.34	60000	2500	0.25	0.4
Fatigue Life Estimation 45	Virtual	400000	50000	0.34	0.34	60000	2500	0.25	0.4
Fatigue Life Estimation 100	Virtual	700000	70000	0.4	0.4	100000	2500	0.25	0.4
Fatigue Life Estimation 20	Virtual	200000	30000	0.27	0.27	25000	2500	0.25	0.4
Fatigue Life Estimation 92	Virtual	700000	70000	0.4	0.4	100000	2500	0.25	0.4
Fatigue Life Estimation 32	Virtual	200000	30000	0.27	0.27	25000	2500	0.25	0.4
Fatigue Life Estimation 18	Virtual	200000	30000	0.27	0.27	25000	2500	0.25	0.4
Fatigue Life Estimation 19	Virtual	200000	30000	0.27	0.27	25000	2500	0.25	0.4
Fatigue Life Estimation 93	Virtual	700000	70000	0.4	0.4	100000	2500	0.25	0.4
Fatigue Life Estimation 6	Virtual	200000	30000	0.27	0.27	25000	2500	0.25	0.4

(b)

Figure 14.—Data definition. (a) ANN model. (b) RNN model.

copy the associated attributes in the defined record, such that changes made in the Reference Data: Machine Learning records would automatically update in the Models: Machine Learning records. Within the current limitations of the Granta MI platform, the only way to define the virtual data once while simultaneously showing that data in two different model records can only be enabled by the current schema, particularly if it is desired to have the same example in different categories (i.e., training, validation, and test) for the two different models. Note that the "Validation Data Set" and "Test Data Set" attributes take a similar form to the "Training Data Set" attribute, and thus are not shown.

The Model Validation layout heading for both model records contain the same populated attributes, although because the models are different and the attribute values are model specific, the individual values differ between the two (Figure 15). For this example, the "Training Loss" and "Validation Loss" functional attributes are defined as a function of the number of epochs completed during training, shown

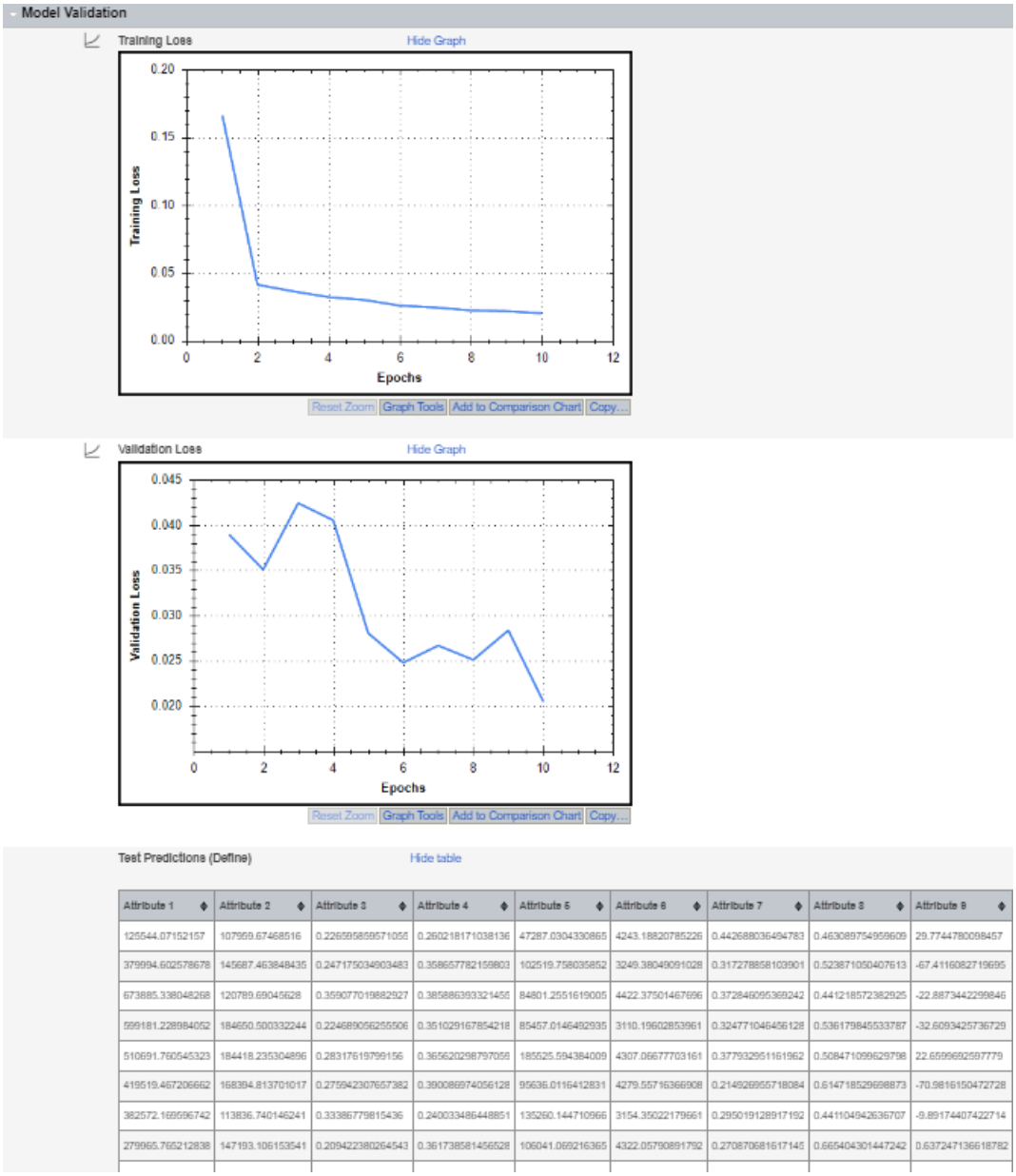


Figure 15.—Model validation attributes.

for the ANN model. For the “Test Predictions” tabular attribute, the inputs and model predictions are given for each example in the test set. Because the predictions are specific to the model and are not the data set used for training, there is no linking to the Reference Data: Machine Learning table for this attribute, and all values in the tabular attribute are defined rather than viewed.

The example problem shown demonstrate the need to separate the data sets used for model training and the parameters that define the models themselves in order to follow the outlined best practices for database management. Furthermore, it shows the need to create individual records for each example in a data set given the limitation of the current Granta MI platform. Without this overarching schema, it would be impossible to define multiple ML models from the same data set, or to train a ML model from multiple data sets, without repeating information somewhere in the database, thus violating one of the basic principles of database management. The example further shows the capability to capture all relevant information regarding the traceability and pedigree of an ML model such that the model can be easily found, used, and reproduced by members within an organization.

6.0 Importing Machine Learning Data

One of the more important aspects of effective database design and management has to do with minimizing the burden of utilization regarding data capture: that is, the ease of users to upload data they have on their hard drive or local machines to the database. Without effective tools that facilitate this process, adoption of the database by users within an organization can be difficult, leading to incomplete access to information across the organization. In the Granta MI Viewer software, users can only input data into a record manually, attribute by attribute or, in the case of tabular attributes, cell by cell. This method of uploading data is extremely time consuming and, particularly in the case of ML, where thousands of records need to be uploaded for a single model, is infeasible. To facilitate the ease of data upload, Granta MI offers a Python SDK connection point, which allows users to develop custom Python code that can effectively write records to the database automatically. Thus, for the Reference Data: Machine Learning and Models: Machine Learning tables a Python Graphical User Interface (GUI) was written using the tkinter Python module (Ref. 22).

Loading the Machine Learning Importer tool presents the user with the home screen (Figure 16), which allows the user to upload either ML data or a ML model individually, as well as both the data and associated model together. For demonstration purposes in this paper, the third option will be shown, as the other two are a subset of the combination upload. When continuing by selecting an option, the user is then displayed a table where each row matches the attributes in the Data Series Information layout heading in the Reference Data: Machine Learning table (Figure 17). Note that for discrete (DCT) type attributes, the GUI also presents the user with the same discrete options. There is also an option to read an .xlsx or .csv file and automatically populate the table for convenience. The Record Name row of the table is the name given to the generic (parent) record, and each child record will be the combination of the name entered and the row index in the data file (e.g., “Reference Data: Machine Learning Demo 1”).

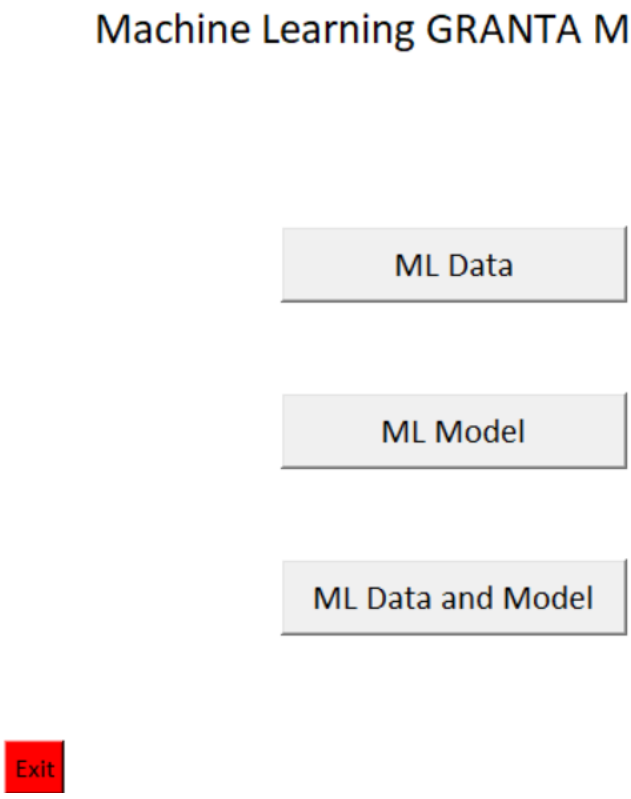


Figure 16.—Machine learning importer home screen.

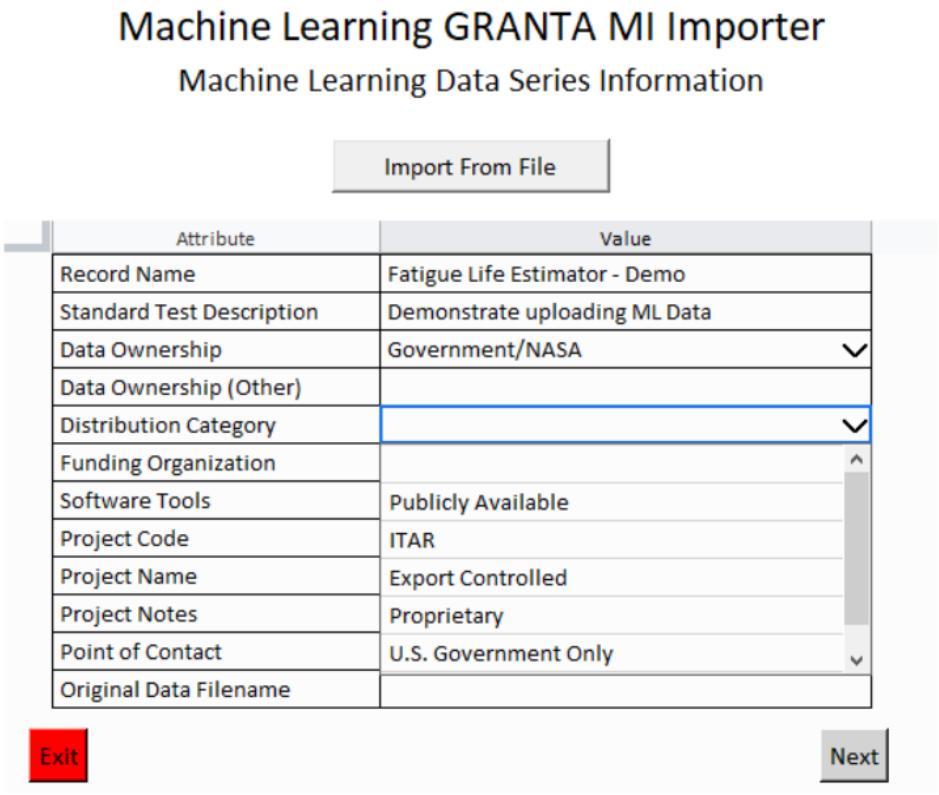


Figure 17.—Importing data series information for a Reference Data: Machine Learning record.

The next screen asks the user to upload the file containing all data used to train, validate, and test the model. The importer assumes that each row is a given example in the data set and that the first row in the file contains the data labels. Once a .xlsx or .csv file is selected by the user, the table shown in Figure 18 will be automatically populated. The Labels column will be populated with values from the first row and if those values contain opened and closed parenthesis ("( )"), any text within will be assumed as a Unit and populated in the adjacent column. By default, all parameters are assumed to be inputs in the final column, so the user must also change any rows that are outputs, or predictions, from the model in the GUI using the drop-down menu in each cell. Note that for simplicity the same table is used when uploading either a Reference Data: Machine Learning or Models: Machine Learning record individually, as well as in the combined case. For Reference Data: Machine Learning records, the Input/Output column data is not used, since the relationship between inputs and outputs is dependent on the ML model developed, and not necessarily the same as the software tool used to create the virtual data.

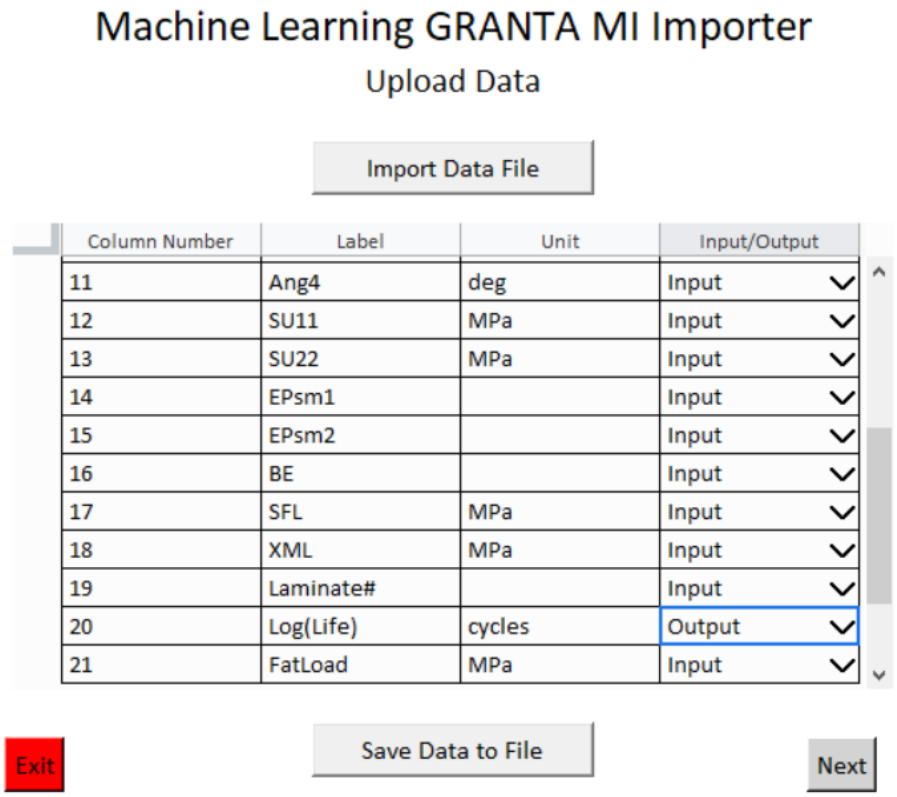


Figure 18.—Reading the data file and defining parameters.

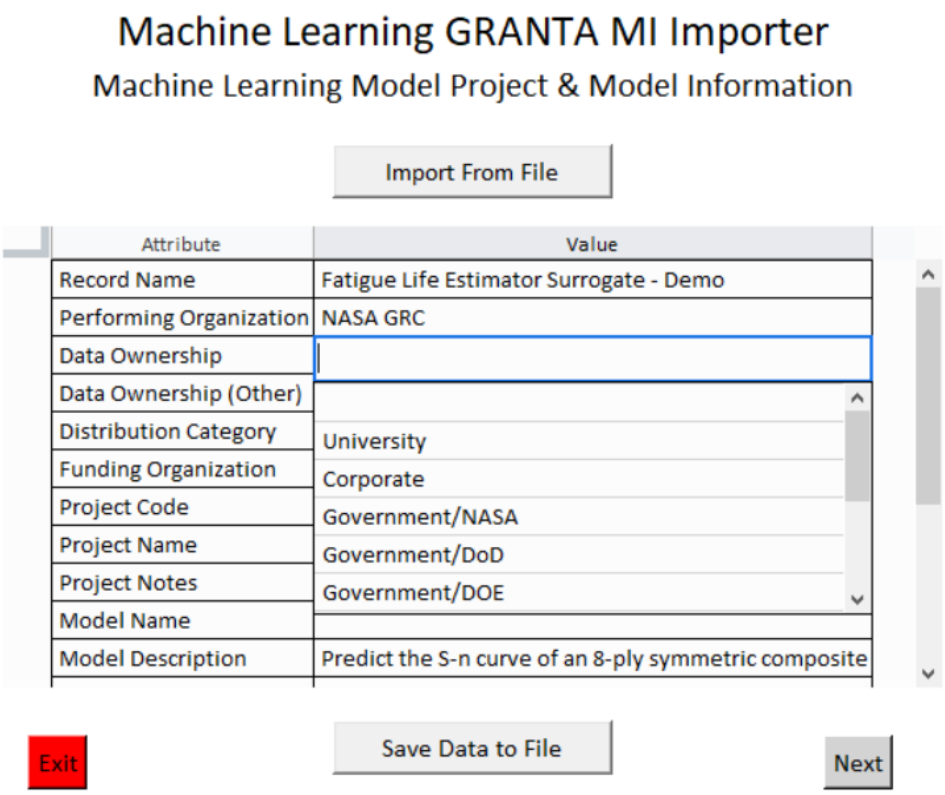


Figure 19.—Importing project and general model information for a Models: Machine Learning record.

Once the data has been uploaded and the different columns defined in the import tool, the next window presents the user with a table with attributes matching those present in the Project Information and General Model Information layout headings in a Models: Machine Learning record (Figure 19). Again, any attribute of discrete type has the same list options in the GUI to prevent invalid information.

The next screen presents the user with a table that matches the Data Assembly attribute in the Data Definition layout heading, allowing the user to specify the percentage split of training, validation, and test data (Figure 20). Similar to the Data Assembly attribute, the user also specifies the random seed value or equation used to shuffle the data prior to training. On the back end of the GUI, the code will take the index of the stored data and shuffle according to the user definition. The index and generic record name from the Reference Data: Machine Learning table are used by the importer to write the Models: Machine Learning record, such that the data is properly viewed and not redefined. Once the training, validation, and test data split have been defined, the importer will then ask the user to supply a .xlsx or .csv file containing the model test predictions (Figure 21). The importer assumes that each row in the supplied file corresponds to the row in the test data after it was shuffled. There are also options for the user to supply .xlsx or .csv files for the training and validation curves, as well as a value for accuracy.

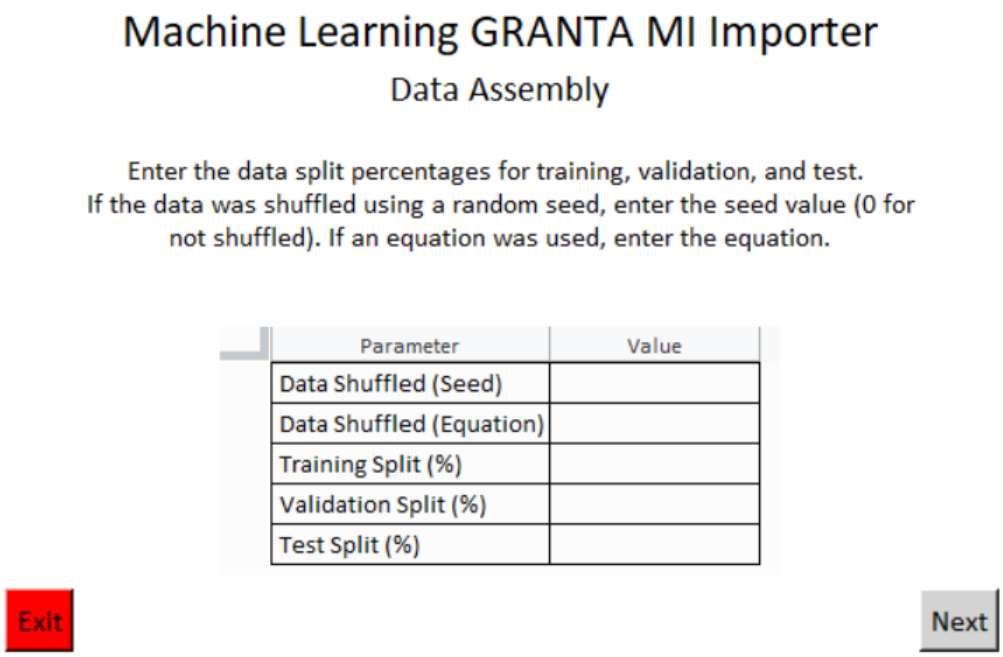


Figure 20.—Data assembly definition in the importer.

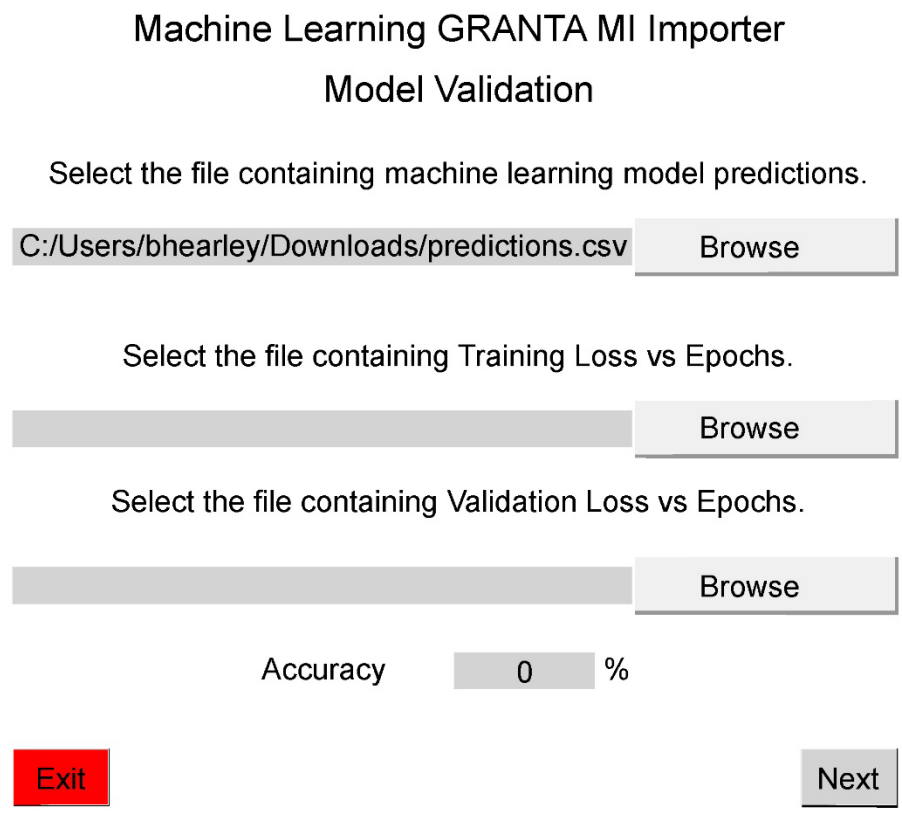
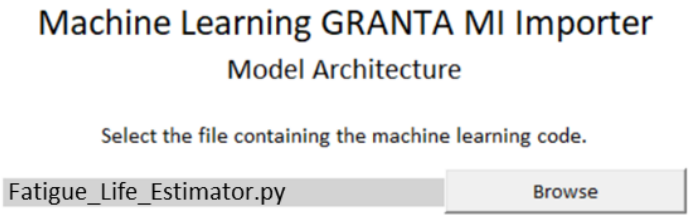


Figure 21.—Importing the model validation.

The final window in the import tool asks the user to supply the ML code used to train the model (Figure 22(a)). An additional feature built into the import tool is the ability to digest ML code as static text and automatically extract the libraries used and model architecture by reading keywords expected for a given ML package. Currently the import tool supports reading Python codes that use Keras with TensorFlow for ML, due to its widespread use at NASA GRC for developing surrogate models. Figure 22(b) shows an example of how the import tool determines the model architecture from the supplied ML code. By first finding ".Sequential," the code knows to look for ".add" for a new addition to the model. By knowing the layer types available in Keras (in this case "Dense" and "LSTM") and their associated options, the code can recognize layers, hidden units, and activation functions for the model.



(a)

(b)

Figure 22.—(a) User input for ML code digestion. (b) Example of code digestion procedure.

The keyword ".compile" indicates the end of the model building, and knowledge of the options supply within the importer with the loss function and learning rate. The code digestion feature for a Models: Machine Learning record has been shown to work well for various architectures but is currently limited to Keras models using TensorFlow. As more users adopt the ML schema presented, additional functionality can be added for different ML programming packages to help incentivize users uploading their data and models to the database.

All data uploaded through the import tool GUI is saved to local Python libraries with matching attribute names to the above schema. Using the Python SDK provided by Granta MI, records can then be written to the database automatically by matching the attribute names to the locally populated libraries. The import tool gives users the means to upload their large data sets to the database in a time-efficient manner, creating the individual data records necessary for best database management practices that would otherwise be infeasible to do manually. It also provides some additional utility features, such as the automatic code digestion and ability to save and import programmatic information, in an attempt to reduce the burden on the user and increase the likelihood of user adoption.

7.0 Current Challenges and Proposed Improvements to Granta MI

As stated previously, the current Reference Data: Machine Learning table schema is not optimal, in that it requires the creation of individual records for each example in the data set to enable database management best practices within the current confines of the Granta MI platform. Although the proposed solution works with this limitation, as a data set grows the feasibility of the proposed schema decreases and could thus deter users from using Granta MI for storing ML data. For example, when previously attempting to upload the full data set for the example problem, which contained ~10,000 total examples, the Remote Import Tool for Granta gave a timeout error and the data failed to upload. As a result, the data had to be uploaded in smaller batches, which adds time and opens up the possibility of errors to the data management process. In addition to challenges in uploading the data, once the data is in the database, the large number of records associated with a single data set results in longer load times when using Granta MI Viewer. This load time is further exacerbated when an organization uses *attribute-based* access control, in which the MI Viewer software must check through each attribute, in each record, to determine what data can be displayed to the user.

To address the above issues, the authors have proposed to ANSYS Granta an enhancement to the tabular attribute that would allow an ML data set to be stored in a single record without sacrificing the flexibility of the existing schema. In the proposed change to the tabular attribute, when linking from one tabular attribute to another, the user would specify both a linking value and a linked row. Figure 23 gives a simple demonstration of how the proposed changes would enable the desired feature for the ML schema. Instead of defining individual attributes in the multiple data records, each of the previous attributes would now be a column in the Data attribute and each example in the data set would be a row (see gray box—Data Record—in Figure 23). For the model record, the Training Data, Validation Data, and Test Data attributes would be defined as linked tabular attributes with the same columns as the "Data" attribute. When the user is defining the Model Record attributes (back end, see light yellow, Figure 23), the user would define the linking value and a linked row as opposed to just a linking value. The Granta MI software would then search for all records in the Data Table that contain the linking value in the "Data ID" attribute (see gray box Figure 23) and would display all values found for the specified column name (i.e., Value 1, Value 2) in the specified linked row. The resultant model record (orange box, Figure 23) would then display to the user (front end) the defined partition of the "Data" tabular attribute in the three tabular attributes shown.

Data Record

Data ID	Data Set	
Data		
Row	Value 1	Value 2
1	100	50
2	200	100
3	300	150
4	400	200
5	500	250
6	600	300
7	700	350
8	800	400
9	900	450
10	1000	500

Model Record (back end)

Training Data	
Linking Value	Linked Row
Data Set	1
Data Set	2
Data Set	4
Data Set	6
Data Set	8
Data Set	10
Validation Data	
Linking Value	Linked Row
Data Set	3
Data Set	5
Test Data	
Linking Value	Linked Row
Data Set	7
Data Set	9

Model Record (front end)

Training Data		
Row	Value 1	Value 2
1	100	50
2	200	100
3	400	200
4	600	300
5	800	400
6	1000	500
Validation Data		
Linking Value	Value 1	Value 2
Data Set	300	150
Data Set	500	250
Test Data		
Linking Value	Value 1	Value 2
Data Set	700	350
Data Set	900	450

Figure 23.—Proposed enhancement to the tabular attribute in Granta MI.

8.0 Conclusions

Advances in modeling techniques and machine learning (ML) have enabled the ability to quickly and efficiently mimic physics-based models at various length scales, further enabling the prospect of ICME and allowing for the design of fit-for-purpose materials. As these abilities grow further, and the problems to be solved become more complex, the time and cost traditionally associated with either experimental testing or computational runtime of physics-based models will transfer to the creation of virtual data and training time for surrogate models. Therefore, for any organization to efficiently embrace the advantages ML can offer, effective data management is paramount in maintaining traceability, preventing the repetition of re-creating existing virtual data, re-creating existing surrogate models, or creating new surrogate models without first referencing any existing similar models that have been developed. Another advantage is the avoidance of misuse of ML models and/or the understanding of imposed limitations on the models due to training data/procedure. The proposed schema for ML data and models looks to address these potential problems by offering a generalized, robust means of storing/retrieving models and their associated data and metadata within the ANSYS Granta MI Database platform, such that existing model pedigree information can be easily found, understood, and used, or not used, for future model development or application in the design of materials while following previously defined best practices for database management. Furthermore, the associated tools developed allow users to easily and efficiently interact with both models and data and import/export existing data into the database, thereby increasing the likelihood of adoption of the proposed ML schema. Also presented are the challenges that the current schema faces subject to the current restrictions imposed by the Granta MI tabular attribute, as well as the proposed solution to enable a more efficient means of storing such data. Surrogate models developed using ML are essential for truly implementing ICME, and the management of the data needed for such models is thus necessary in helping an organization capitalize on the capabilities that such tools can enable.

References

1. M. Yuan, S. Paradiso, B. Meredig, and S. Niezgoda, "Machine Learning-Based Reduce Order Crystal Plasticity Modeling for ICME Applications," *Integrating Materials and Manufacturing Innovation,* vol. 7, pp. 214–230, 2018.
2. S. Arnold, F. Holland Jr., and B. Bednarcyk, "Robust Informatics Infrastructure Required for ICME: Combining Virtual and Experimental Data," in *AIAA SciTech Forum*, National Harbor, MD, 13–17 January 2014.
3. Y. Al-Assaf and H.A. Kadi, "Fatigue life prediction of composite materials using polynomial classifiers and recurrent neural networks," *Composite Structures,* vol. 77, pp. 561–569, 2007.
4. M. Piekenbrock, J. Stuckner, S. Arnold, and T. Ricks. "Multiscale Analysis of Composites Using Surrogate Modeling and Information Optimal Designs," in *AIAA SciTech Forum*, Orlando, FL, 6–10 January 2020.
5. S. Khandelwal, S. Basu, and A. Patra, "A Machine Learning-based surrogate modeling framework for predicting the history-dependent deformation of dual phase microstructuers," *Materials Today Communications,* vol. 29, pp. 1–14, 2021.
6. R. Liu, Y. Yabansu, Z. Yang, A. Choudhary, S. Kalidindi, and A. Agrawal, "Context Aware Machine Learning Approaches for Modeling Elastic Localization in Three-Dimensional Composite Microstructures," *Integrating Materials and Manufacturing Innovation*, vol. 6, pp. 160–171, 2017.
7. B. Hearley, J. Stuckner, E. Pineda, and S. Murman, "Predicting Unreinforced Fabric Mechanical Behavior With Recurrent Neural Networks," *NASA Technical Memorandum,* NASA/TM-20210023708, pp. 1–26, 2022.
8. "ASM Alloy Center Database," ASM. [Online]. Available: mio.asminternational.org/ac
9. "High Performance Alloys Database (HPAD)," CINDAS LLC, 2022. [Online]. Available: https://cindasdata.com/products/hpad
10. "nanoHUB," NCN, 2022. [Online]. Available: https://nanohub.org/
11. "Nanomaterial Registry," Nanomaterial Registry, 2022. [Online]. Available: http://www.nanomaterialregistry.org/
12. N. Day and P. Murray-Rust, "Crystallography Open Database," CrystalEye, 2022. [Online]. Available: http://www.crystallography.net/cod/
13. "The Cambridge Crystallographic Data Centre (CCDC)," University of Cambridge, 2022. [Online]. Available: https://www.ccdc.cam.ac.uk/
14. S. Arnold, F. Holland, T. Gabb, M. Nathal, and T. Wing, "The Coming ICME Data Tsunami and What Can Be Done," in *54th AIAA/ASME/ASCE/AHS/ACS Structures, Structural Dynamics, Materials Conference*, Boston, MA, 23–27 April 2013.
15. S. Arnold, F. Holland, B. Bednarcyk, and E. Pineda, "Combining Material and Model Pedigree is Foundational to Making ICME a Reality," *Integrating Materials and Manufacturing Innovation, IMMI,* no. 4:4, DOI 10.1186/s40192-015-0031-2, 2015.
16. M.D. Wilkinson et al., "The FAIR Guiding Principles for scientific data management and stewardship," *Scientific Data*, vol. 15, no. 3, 2016.
17. "Granta MI: Materials Gateway for ANSYS Workbench," ANSYS, 2022. [Online]. Available: https://www.grantadesign.com/industry/support/granta-mi/gateway/notice/
18. S.M. Arnold, S.K. Mital, P.L. Murthy, and B. Hearley, "Use of Artificial Neural Nets to Predict Stiffness and Fatigue Lives of Polymer Matrix Composite Laminates," in *NAFEMS Regional Conference Americas*, Indianapolis, IN, June 21–23, 2022.

19. B.A. Bednarcyk and S.M. Arnold, “MAC/GMC 4.0 User’s Manual, Volume 2: Keywords Manual,” *NASA Technical Memorandum*, TM 2002-212077/Vol. 2, 2002.
20. M.W. Gardner and S.R. Dorling, “Artificial Neural Networks (The Multilayer Perpetron)—A Review of Applications in the Atmospheric Sciences,” *Atmospheric Environment*, vol. 32, no. 14/15, pp. 2627–2636, 1998.
21. Z. Lipton, D. Kale, C. Elkan, and R. Wetzel, “Learning to Diagnose with LSTM Recurrent Neural Networks,” in *International Conference on Learning Representations*, San Juan, PR, 2016.
22. “Graphical User Interfaces with Tk,” Python Software Foundation, 17 October 2022. [Online]. Available: https://docs.python.org/3/library/tk.html