

System and Method for Protecting Sensitive Production Data for Machine Learning Model Generation

Field of the Invention

The present application relates to systems and methods for protecting sensitive data for Machine Learning (ML) model generation, and in particular populating a synthetic dataset in a development environment using a set of metadata associated with a real dataset of a production environment, processing the synthetic dataset in a development environment for training a machine learning model in the development environment, and deploying a validated development model to the production environment after correcting for source code vulnerabilities corresponding to the training of the machine learning model.

Background

In the field of developing and designing ML models, it is often necessary to conduct testing of the ML model within a development environment, before the model is deployed to a production environment for real-world use by end users. Production environments, such as a real-world website or a client business application, often contain sensitive or confidential information that could compromise the identity or security of a user. Moreover, it has become increasingly commonplace for developers located in various regions around the world to collaborate on the building of ML models. This has led to an increased sensitivity towards compliance with international and local laws concerning data privacy protections for cross-border collaborative ML projects. Notwithstanding these requirements, there remains a need to ensure comprehensive and effective testing and deployment of ML models despite the undesirability of making production data available outside of the production environment. Thus, it is desirable to provide a system and method for developing ML models which protects against the identification of sensitive production data during ML development or at least to provide the public with a useful choice.

Summary of the Invention

The present invention aims to provide new and useful systems and methods for protecting sensitive data for Machine Learning (ML) model generation, and in particular using a set of metadata associated with a real dataset of a production environment for populating a synthetic dataset in a development environment for the training of machine learning models in the development environment and the deployment of the machine learning models to the production environment.

In broad terms, the present invention proposes a method for protecting sensitive production data configured to receive a request associated with a generation of a machine learning

model in a development environment and to read a set of metadata associated with a real dataset of a production environment based on the request. One way of implementing this method is to transmit the set of metadata to the development environment for populating a synthetic dataset in the development environment and process the synthetic dataset in the development environment for training the machine learning model from the development environment, wherein the training of the machine learning model generates a plurality of source code. As it can be appreciated from the described embodiment, the method may scan the source code of a trained machine learning model for source code vulnerabilities and correct source code vulnerabilities corresponding to the trained machine learning model. The trained machine learning model may be deployed as a validated development model to the production environment after the correction of the source code vulnerabilities.

In one aspect, the set of metadata is provided on a graphical user interface (GUI) and arranged as a plurality of columns in a table that correspond to a classification of the real dataset. The set of metadata may be provided to a user of the development environment and the synthetic dataset is populated by the user.

In implementations, synthetic dataset can be populated without exposing the real dataset to a user and the generated synthetic dataset may be further ingested by a neural network of the machine learning model to train the machine learning model.

The correction of the source code vulnerabilities may include removal or replacement of source code. Optionally, the transmitted set of metadata may be presented to a user of the development environment via a graphical user interface. The set of metadata may further correspond to at least one characteristic of the real dataset.

In some implementations, a source code repository may maintain source code of the development environment and the production environment including logged changes to the source code associated with the machine learning model. The metadata may represent at least one of an identification, a description, and a classification of the real dataset.

In cases where a development workflow management module is configured to extract the synthetic data for preparing a training dataset and a validation dataset, the training dataset is used to train the machine learning model and the validation dataset is used to validate the learning accuracy of the trained machine learning model for the development environment.

After deployment of the validated development model, a production workflow management module may be configured to extract the real dataset for preparing a production training dataset and a production validation dataset, wherein the production training dataset is used to further train the validated development model and the production validation dataset is used to examine the learning accuracy of the validated development model for the production environment.

The development workflow management module can be further configured to encode the prepared training data based on a machine learning model and to conduct training of an artificial neural network. A graphic user interface (GUI) for display in the development environment can enable a plurality of users to use the synthetic data to develop the machine learning models through iterative model experiments until an acceptable version of the machine learning model is obtained.

The trained machine learning model can be validated for use in the production environment based on the determined compliance of the trained machine learning model to a set of configurations of the production environment. The source code vulnerabilities can include source code not generated by a customized software development kit (SDK), wherein source code generated by the customized SDK is executed in the development environment.

In cases where the source code vulnerabilities include native code or log writing functions of particular programming languages, wherein the native code or log writing functions corresponds to source code not generated by the customized SDK for displaying or recording of data, a trained machine learning model will not be deployed to the production environment. The log writing functions may correspond to a predetermined blacklist of functions stored in a testing server. The customized SDK may be a unified logging system configured to perform data logging and sensitive information monitoring of the machine learning models. The customized SDK uses a deployment function to determine a “match” in similarity between the source code of an application for training the machine learning model and a predetermined list of sensitive data, wherein a match in similarity is determined by a Regular Expression matching algorithm.

In some implementations, the development workflow management module may monitor the source code executed by the development environment and the development workflow management module may provide machine learning models generated in the development environment to a testing cluster for testing of the machine learning models, wherein the testing cluster includes specific algorithms or techniques for performing the testing of the

generated machine learning models. The set of configurations of the production environment may include security and topology configurations associated with one or more specific production environments.

The invention may be expressed as a method, or alternatively as a system comprising one or more computers and one or more storage devices storing instructions that when executed by the one or more computers cause the one or more computers to perform the method. It may also be expressed as a computer program product, such as downloadable program instructions (software) or one or more non-transitory computer storage media storing instructions. When executed by one or more computers, the program instructions cause the one or more computers to perform the method.

Brief Description of the Drawings:

Embodiments of the invention will now be explained for the sake of example only, with reference to the following figures in which:

Figure 1 is a flow diagram of a process for protecting sensitive production data in a ML development environment, according to an embodiment of the present disclosure.

Figures 2A-2B are example table representations of metadata, real data, and user populated synthetic data corresponding to the metadata, according to an embodiment of the present disclosure.

Figure 3 is a flow diagram of a deployment function process of deploying trained machine learning models to a production environment, according to an embodiment of the present disclosure.

Figure 4 is an example system architecture illustrating the various elements and entities involved in the development and deployment of a ML project, according to an embodiment of the present disclosure.

Figure 5 is a block diagram illustrating an example computer system which can be used to perform the methods of Figures 1 and 3, and to implement the system of Figure 4.

Detailed Description of Embodiments of the Invention

Embodiments will now be discussed with reference to the accompanying FIG.s, which depict one or more exemplary embodiments. These embodiments are described in sufficient detail to enable those skilled in the art to practice the embodiments and it is to be understood that mechanical, logical, and other changes may be made without departing from the scope of the embodiments. Therefore, embodiments may be implemented in many different forms

and should not be construed as limited to the embodiments set forth herein, shown in the FIG.s, and/or described below.

Embodiments of the present invention relate to a new and useful system and method for protecting sensitive production data while building machine learning models in a development environment using synthetic user data.

Herein, the term “production environment” refers to settings where computer applications are deployed in systems for use in real-world environments (such as web-based interfaces) by end users. “Production data” or real data, refers to data of an application executing in a production environment, can include sensitive data, such as financial data and confidential information such as, but not limited to, client names, client contact information, bank account balances, and credit score information. The term “development environment” herein refers to settings where a software developer or tester can use computer applications to create compiled (e.g., executable) code, debug, maintain, or otherwise support computer programs and applications. For example, in a machine learning development environment, a developer may use compiled code to test, train and validate machine learning models for potential deployment in the production environment. “Development data” refers to the data used in the development environment by a newly developed software application such as a machine learning model for testing purposes. The term “metadata” herein refers to data that identifies one or more features, classifications, or other associations regarding the content of production data based on a request to develop or test a machine learning model in the development environment. “Synthetic data” herein describes data created by a user or developer of the development environment based on metadata provided from the production environment. For example, the synthetic data may correspond to a set of real user data randomly populated by a user of the development environment without knowledge of any personal or identification data identifying the real user data associated with the metadata which may be considered sensitive or personal to the individual user.

Figure 1 is a flow diagram of a process for protecting sensitive production data in a development environment, according to an embodiment of the present disclosure. Although process or method steps may be described in a sequential order, such processes, methods and techniques may be configured to work in alternate orders. In other words, any sequence or order of steps that may be described does not necessarily indicate a requirement that the steps to be performed in that order. The steps of processes described herein may be performed in any order practical. Further, some steps may be performed simultaneously.

Referring to the steps of the method 100, at step 105, the processor 522 is configured to receive a request associated with the generation of a ML model in the development environment. The received request may correspond to a request to receive one or more content related files associated with a software or machine learning application in the production environment. The request may be initiated by a user in the development environment and may correspond to the building or testing of a ML model for use in performing a task in the development environment.

In step 110, metadata associated with real-world data in the production environment is read in response to the request associated with the development environment in step 105. The metadata may correspond to at least one characteristic or attribute of the real dataset. In some embodiments, the set of metadata may be predetermined by a user of the production environment, wherein the user of the production environment has determined the values and conditions for the description of the metadata based on the user's assessment of the real data that is grouped or listed in the production environment. Additionally, the metadata may be dynamically updated based on changes to the real data information in the production environment.

In step 115, the metadata is then provided to a user in the development environment for generating synthetic data. The description of the metadata from the production environment provides the user of the development environment necessary context for populating the development environment with synthetic data corresponding to the sensitive production data. For example, a metadata can provide a description of the type of user information located in an application in the production environment as "user_name, user_gender, user_device, user_device_id, and user_device region" which would allow an individual user to generate synthetic data that is similar or approximate to the real actual data of the application. Accordingly, the synthetic dataset would share the same metadata as the real dataset but contain different data records than the real dataset.

In step 120, the synthetic dataset is used as a test dataset for obtaining a training data set and a validation dataset and a machine learning model is further trained based on the training dataset and model validation is performed based on the validation dataset. In some embodiments, the synthetic dataset is first pre-processed into a format suitable for model estimation before training and validation is performed based on the synthetic dataset. Moreover, data cleansing (i.e. inspecting and cleaning of the data for missing values) may be performed on the synthetic dataset prior to model training and model validation.

In step 125, the source code of the trained and validated machine learning model of the development environment is scanned to determine potential source code vulnerabilities.

Such source code vulnerabilities may include a determination of whether the extracted source code of the machine learning application used for model training or validation

corresponds to the source code format of a custom Software Development Kit (SDK) used to create, test and output the components of the machine learning application for training the ML model. In some embodiments, the source code vulnerabilities may correspond to a determination of native code or log writing functions of particular programming languages, which do not correspond to the source code generated by the custom SDK for outputting or displaying components of the machine learning application. The custom SDK may further define components, graphs, and other visualizations corresponding to the development and deployment of a machine learning model in the development environment.

In step 130, the trained and validated machine learning model is deployed in the production

environment as a validated or deployed development model after correction of the source code vulnerabilities of step 125. The correction of the source code vulnerabilities may include removal or replacement of the source code to correspond to the format generated by the custom SDK. An example evaluation for source code vulnerabilities and related deployment of the machine learning model for production as provided in steps 125 and 130 is further described with respect to the steps of the deployment function and FIG. 3 below.

In step 135, following deployment of the machine learning model in the production environment, the deployed development model may be further trained and validated in the production environment based on additional training, validation and test data sets from the production environment. In step 135, the real dataset is used as a test dataset for obtaining a training dataset and validation dataset and the deployed model is further trained based on the training dataset generated from the real data. Moreover, model validation of the deployed model may be performed based on the validation dataset generated from the real data of the production environment. Similar to the development environment, in some embodiments, the real dataset may be first pre-processed into a format suitable for model estimation before training and validation of the real-world dataset is performed. Moreover, data cleansing (i.e. inspecting and cleaning of the data for missing values) may be performed on the real-world dataset prior to the model training and model validation.

Figures 2A-2B are table representations of metadata, real data, and user populated synthetic data corresponding to the metadata generated by the process 100 shown in FIG.

1. For example, in one implementation, a GUI designed to present reference table 2A which

contains metadata (e.g. user_name, user_gender, user_device, user_device_id, user_device region) and real data (e.g. Allen Chan, Male, Apple iPhone 14, Aslfdaslfask32, and Singapore corresponding to identification 98) may be provided in the production environment wherein the column metadata indicates an attribute, category or data type of the real data contained in each column. In this example, a user in the development environment, e.g. a software developer, may request production information corresponding to clients or personal ID numbers 98 and 135. Given the user in the development environment is not entitled to access real data table with respect to user ID numbers 98 and 135, only the metadata information (user_name, user_gender, user_device, user_device_id, user_device region) will be transmitted and presented to the development environment user as shown in FIG. 2B. This allows for non-removal and non-encryption of the sensitive real data while protecting the sensitive data of the production environment. As provided in FIG. 2B, based on only the received the metadata information, the user can create the corresponding table with synthetic data in the development environment. For example, the user may input Henry Rose, Male, Xiaomi Mi 11, Cabeeghjdsghd, and Singapore based on the metadata information about employee 98 presented to the user as shown in FIG. 2B. The table of FIG. 2B maybe stored in a server in the development environment.

Figure 3 is a flow diagram of a deployment function process for deploying trained machine learning models to a production environment, according to an embodiment of the present disclosure.

In step 305, code developed for applications corresponding to the testing and development of machine learning models may be scanned and identification of user-related data associated with the applications may be determined.

In step 310, a pre-determined “blacklist” of logging functions or native data formats not corresponding to the custom source code generated by a custom SDK for outputting or displaying components in the development environment is accessed by the deployment function.

In step 315, a comparison is performed to determine if the scanned code of step 305 is compliant and corresponds to the custom SDK format for outputting/displaying data or if the code corresponds to code which is not implemented by the custom SDK for outputting data. Similarity or a “match” between the source code of the application and a predetermined list of noncompliant output functions or native data formats is determined by a Regular Expression (Regex) matching algorithm. If there is a match between source code and the

predetermined list of output functions or native data formats, then the developed machine learning model will not be deployed for production.

In step 320, the customized SDK of the development environment implements a built-in
5 logger function which provides a record of any sensitive data information that may have
been logged by a user during model testing and development. A developer may sanitize or
remove any logged sensitive data from non-debug versions of the application in accordance
with the developer's review of the SDK logging record. Following a determination, the
scanned code is compliant with custom SDK format and the code does not contain any
10 logged sensitive data, the validated ML model may be deployed in the production
environment.

Figure 4 is an example system architecture illustrating the various elements and entities
involved in the development and deployment of a ML project, according to an embodiment of
15 the present disclosure.

The system 400 and variants of it can be used to perform any of the methods described
herein. In the example, a production workflow management module 430 is configured to
receive one or more sets of production configurations 415. The production configurations
20 415 may include one or more parameters compatible with a production environment and/or
utilized by one or more ML applications of the production environment. In this example
implementation, the production configuration 415 may be included in a source code
repository 410. The source code repository 410 may include applications source code,
component source code, and infrastructure source code for ML testing or validation
25 performed in both the production and development environments.

The production workflow management module 430 may be further configured to manage
task commands in the production environment. For example, the production workflow
management module 430 may receive task commands associated with performing particular
30 tasks used in a ML model and determining an order of the task commands for execution in
the production environment. The production workflow management module 430 may
transmit information corresponding to the production environment such as ML training or
prediction data or other metrics to a customer or user in the production environment via a
production graphical user interface (GUI).

35 One or more snapshots, backups, or copies of a dataset from the production environment
are stored in production clusters 440 and the production clusters 440 may provide data

storage to one or more applications in the production environment. The dataset from the production environment stored in production clusters may include potentially sensitive data such as personally identified information and one or more datasets of metadata associated with the sensitive data.

5

The production clusters 440 may be configured to provide the testing clusters 435 access to the datasets of metadata without providing the testing clusters 435 with access to the real data stored in the production clusters which may include sensitive data. The testing clusters 435 function to perform metadata ingestion by processing the received metadata from the production clusters 440 into one or more formats that can be ingested by ML models and/or ML algorithms generated or used in the development environment.

10

In implementations, a development workflow management module 425 may be further configured to manage task commands in the development environment. For example, the development workflow management module 425 may receive task commands associated with performing particular tasks using a ML testing or model validation and determining an order of the task commands for execution in the development environment. The development workflow management module 425 may transmit information corresponding to the development environment to a development graphical user interface (GUI) 405 such that the GUI may display visualization information corresponding to one or more ML applications. The GUI 405 may include user selectable options or features related to a solution to be developed within the development environment. The development workflow management module 425 may extract development source code information from the source code repository 410 including source code for ML testing and validation performed in the development environment.

15

20

25

The system 400 further includes server 445 coupled to the development workflow management module 425 and the production workflow management module. The server 445 may be connected to a network or may be part of a data centre. A user of server 445 may submit a request to the testing clusters 435 to deploy a tested or validated ML model of the development environment to the production environment. The request via server 445 may also include a one or more production configuration or production environment parameters associated with the production environment.

30

35

Upon receipt of the request, the development workflow management module 425 may evaluate the request and identified candidate trained ML model for potential deployment. Upon scanning of the source code of the candidate ML model for source code vulnerabilities

and potential correction of the vulnerabilities, the candidate ML model may be deployed as a validated development model to the production environment. Upon deployment, the development workflow management module 425 transmits to the production workflow management module 430 of the production environment the particular workflow steps used in the training or development of the ML model in the development environment for potential regeneration of the workflow for the production environment. For example, the workflow steps may include actions or processes taken by a software developer or user of the development environment during ML code creation or modification or testing or debugging or profiling or deployment of the ML model.

Figure 5 is a block diagram showing the technical architecture 500 of a server which can perform some or all of the methods of Figures 1 and 3. The technical architecture includes a processor 522 (which may be referred to as a central processor unit or CPU) that is in communication with memory devices including secondary storage 524 (such as disk drives), read only memory (ROM) 526, random access memory (RAM) 528. The processor 522 may be implemented as one or more CPU chips. The technical architecture may further comprise input/output (I/O) devices 530, and network connectivity devices 532.

The secondary storage 524 is typically comprised of one or more disk drives or tape drives and is used for non-volatile storage of data and as an over-flow data storage device if RAM 528 is not large enough to hold all working data. Secondary storage 524 may be used to store programs which are loaded into RAM 528 when such programs are selected for execution.

In this embodiment, the secondary storage 524 has an order processing component 524a comprising non-transitory instructions operative by the processor 522 to perform various operations of the method of the present disclosure. The ROM 526 is used to store instructions and perhaps data which are read during program execution. The secondary storage 524, the RAM 528, and/or the ROM 526 may be referred to in some contexts as computer readable storage media and/or non-transitory computer readable media.

I/O devices 530 may include printers, video monitors, liquid crystal displays (LCDs), plasma displays, touch screen displays, keyboards, keypads, switches, dials, mice, track balls, voice recognizers, card readers, paper tape readers, or other well-known input devices.

The processor 522 executes instructions, codes, computer programs, scripts which it accesses from hard disk, floppy disk, optical disk (these various disk-based systems may all be considered secondary storage 524), flash drive, ROM 526, RAM 528, or the network connectivity devices 532. While only one processor 522 is shown, multiple processors may be present. Thus, while instructions may be discussed as executed by a processor, the

instructions may be executed simultaneously, serially, or otherwise executed by one or multiple processors.

Although the technical architecture is described with reference to a computer, it should be appreciated that the technical architecture may be formed by two or more computers in communication with each other that collaborate to perform a task. For example, but not by way of limitation, an application may be partitioned in such a way as to permit concurrent and/or parallel processing of the instructions of the application. Alternatively, the data processed by the application may be partitioned in such a way as to permit concurrent and/or parallel processing of different portions of a data set by the two or more computers. In an embodiment, virtualization software may be employed by the technical architecture 500 to provide the functionality of a number of servers that is not directly bound to the number of computers in the technical architecture 500. In an embodiment, the functionality disclosed above may be provided by executing the application and/or applications in a cloud computing environment. Cloud computing may comprise providing computing services via a network connection using dynamically scalable computing resources. A cloud computing environment may be established by an enterprise and/or may be hired on an as-needed basis from a third-party provider.

By programming and/or loading executable instructions onto the technical architecture, at least one of the CPU 522, the RAM 528, and the ROM 526 are changed, transforming the technical architecture in part into a specific purpose machine or apparatus having the novel functionality taught by the present disclosure. It is fundamental to the electrical engineering and software engineering arts that functionality that can be implemented by loading executable software into a computer can be converted to a hardware implementation by well-known design rules.

Whilst the foregoing description has described exemplary embodiments, it will be understood by those skilled in the art that many variations of the embodiment can be made within the scope and spirit of the present invention.

Claims:

1. A computer-implemented method for protecting sensitive production data for Machine Learning (ML) model generation, the method comprising:
 - 5 receiving a request associated with a generation of a machine learning model in a development environment;
reading a set of metadata associated with a real dataset of a production environment based on the request;
transmitting the set of metadata to the development environment for populating a synthetic
10 dataset in the development environment;
processing the synthetic dataset in the development environment for training the machine learning model in the development environment, wherein the training of the machine learning model generates a plurality of source code;
scanning the source code generated by a trained machine learning model for source code
15 vulnerabilities;
correcting source code vulnerabilities corresponding to the trained machine learning model;
and
deploying the trained machine learning model as a validated development model to the
production environment after the correction of the source code vulnerabilities.
20
2. The method according to claim 1, wherein the set of metadata is provided on a graphical user interface (GUI) and arranged as a plurality of columns in a table that correspond to a classification of the real dataset.
- 25 3. The method according to any one of the preceding claims, wherein the set of metadata is provided to a user of the development environment and the synthetic dataset is populated by the user.
4. The method according to any one of the preceding claims, wherein synthetic dataset is
30 populated without exposing the real dataset to a user.
5. The method according to any one of the preceding claims, wherein the generated synthetic dataset is further ingested by a neural network of the machine learning model to train the machine learning model.
35
6. The method according to any one of the preceding claims, wherein the correction of the source code vulnerabilities includes removal or replacement of source code.

7. The method according to any one of the preceding claims, wherein the transmitted set of metadata is presented to a user of the development environment via a graphical user interface.

5

8. The method according to any one of the preceding claims, wherein the set of metadata corresponds to at least one characteristic of the real dataset.

9. The method according to any one of the preceding claims, wherein a source code repository maintains source code of the development environment and the production environment including logged changes to the source code associated with the generation or training of the machine learning model.

10. The method according to any one of the preceding claims, wherein the metadata represents at least one of an identification, a description, and a classification of the real dataset.

11. The method according to any one of the preceding claims, wherein a development workflow management module is configured to extract the synthetic data for preparing a training dataset and a validation dataset, wherein the training dataset is used to train the machine learning model and the validation dataset is used to validate the learning accuracy of the trained machine learning model for the development environment.

12. The method according to any one of the preceding claims, wherein after deployment of the validated development model, a production workflow management module is configured to extract the real dataset for preparing a production training dataset and a production validation dataset, wherein the production training dataset is used to further train the validated development model and the production validation dataset is used to examine the learning accuracy of the validated development model for the production environment.

30

13. The method according to any one of claims 11 or 12, wherein the development workflow management module is further configured to encode the prepared training data based on a machine learning model and to conduct training of an artificial neural network.

14. The method according to any one of the preceding claims, wherein a graphic user interface (GUI) for display in the development environment enables a plurality of users to

35

use the synthetic data to develop the machine learning models through iterative model experiments until an acceptable version of the machine learning model is obtained.

15. The method according to any one of the preceding claims, wherein the trained machine learning model is validated for use in the production environment based on the determined compliance of the trained machine learning model to a set of configurations of the production environment.

16. The method according to any one of the preceding claims, wherein the source code vulnerabilities include a determination that a source code of machine learning application used to train or validate the machine learning model does not correspond to the source code format of a customized Software Development Kit (SDK) used to output components of the machine learning application for training the ML model.

17. The method according to claim 16, wherein a trained machine learning model will not be deployed to the production environment if the source code vulnerabilities include native code or log writing functions of particular programming languages, wherein the native code or log writing functions corresponds to source code not generated by the customized SDK for displaying or recording of data.

18. The method according to claim 17, wherein the log writing functions correspond to a predetermined blacklist of functions stored in a testing server.

19. The method according to any one of claims 16 to 18, wherein the customized SDK is a unified logging system configured to perform data logging and sensitive information monitoring of the machine learning models.

20. The method according to any one of claims 16 to 19, wherein the customized SDK uses a deployment function to review a match in similarity between the source code of an application for training the machine learning model and a predetermined list of sensitive data, wherein a match in similarity is determined by a Regular Expression matching algorithm.

21. The method according to any one of claims 11 to 20, wherein the development workflow management module monitors the source code executed by the development environment.

22. The method according to any one of claims 11 to 21, wherein the development workflow management module provides one or more the machine learning models generated in the development environment to a testing cluster for testing of the machine learning models, wherein the testing cluster includes specific algorithms or techniques for performing the testing of the generated machine learning models.

23. The method according to any one of claims 15 to 22, wherein the set of configurations of the production environment include security and topology configurations associated with one or more specific production environments.

24. A system for protecting sensitive production data for Machine Learning (ML) model generation, the system comprising:

a data storage device storing instructions for protecting sensitive production data; and a processor and a memory coupled to the processor, wherein the processor executes

instructions including:

receive a request associated with a generation of a machine learning model in a development environment;

read a set of metadata associated with a real dataset of a production environment based on the request;

transmit the set of metadata to the development environment for populating a synthetic dataset in the development environment;

process the synthetic dataset in the development environment for training the machine learning model in the development environment, wherein the training of the machine learning model generates a plurality of source code;

scan the source code generated by a trained machine learning model for source code vulnerabilities;

correct source code vulnerabilities corresponding to the trained machine learning model; and deploy the trained machine learning model as a validated development model to the production environment after the correction of the source code vulnerabilities.

25. One or more non-transitory computer storage media storing instructions that when executed by one or more computers cause the one or more computers to perform the operations of any one of the preceding claims.