Australian Government
IP Australia

Invention Title:

DAI- Dataset Discovery: DATASET DISCOVERY IN **D**ATA **A**NALYTICS USING A**I**- BASED PROGRAMMING.

Name and address of patentees(s):

Dr. Surbhi Bhatia (Assistant Professor) Address: Department of Information Systems, College of Computer Science and Information Technology, King Faisal University, Al-ahsa 31982 P.O. Box 400, Saudi Arabia. E-mail: surbhibhatia1988@yahoo.com
Dr. Mohammad Tabrez Quasim Address: Department of Computer Science, College of Computing and Information Technology, University of Bisha, Bisha – 67714, Saudi Arabia. E-mail: tabrezquasim@gmail.com
Ms. Rashi Kohli (Senior Member IEEE) Address: 219 Harmony Blvd, Pooler, Georgia- 31322, USA. E-mail: rashikohli.amity@gmail.com
Dr. Kapal Dev, Affiliation: Trinity College Dublin, Ireland Address: 34 Westland Row, Dublin, Ireland. E-mail: kapal.dev.60@gmail.com
Dr. Anchal Garg (Associate Professor, IEEE Senior Member) Address: E3-301A, Department of Computer Science and Engineering, Amity University Uttar Pradesh Noida, India. E-mail: Agarg@amity.edu
Prof.(Dr.) Rakesh Kumar. E R (Manager Examination) Address: Texila American University, LOT 24-42, Plantation, Providence, East Bank Demerara, George Town, Guyana, South America. E-mail: rakeshkumarer@gmail.com
Dr. Meghna Chhabra (Associate Professor) Address: Faculty of Management Studies, Manav Rachna International Institute of Research and Studies, Faridabad, Haryana, India. E-mail: meghnachhabra28@gmail.com
Dr. Ankur Singh Bist (Associate Professor) Address: Graphic Era Hill University-Bhimtal Campus, Sattal Rd, Bhimtal, Uttarakhand-263156, India. E-mail: ankur1990bist@gmail.com
Mr. Revanth Madamala Address-1: KORE.ai Software Limited, Hyderabad, Telagana, India. Address-2: Flat No 204, Devi Residency, Balaji Nagar, Nellore, AP-524002, India. E-mail: revanthrex1001@gmail.com
Prof.(Dr.) Biplab Kumar Sarkar (Founder GEH Research LLP & Project Director TCS-G) Address-1: GEH Research, G-12, Lavelle Road, Bengaluru, Karnataka-560001, India. Address-2: GEH Research, A-19-21, Ihonbashihakozakichō, Chūō-Ku, Tōkyō-To-103-0015, Japan. Address-3: GEH Research, E-101 Kitchawan Rd, Yorktown Heights, NY 10598, United States. E-mail: dr.bksarkar2003@yahoo.in

Complete Specification: **Australian Government.**

FIELD OF THE INVENTION

My Invention "**DAI- Dataset Discovery**" is related (computer engineering) to dataset discovery in **d**ata **a**nalytics using A**I**- based programming and to techniques for discovering datasets for use in data analytics.

BACKGROUND OF THE INVENTION

Business intelligence (BI) may include processing and analysis of data for business purposes. Businesses typically accumulate large amounts of data, with different data created for different purposes and by different sources. Because potentially related data across a business entity may have different formatting and in many cases is not identified or indexed as being related, business opportunities may be missed. Further, manual location and organization of related data can be time consuming and inaccurate. Even if portions of data location and/or organization may be automated, a human typically reviews the data, making imprecise manual approximations and assumptions. Data science typically refers to the science that incorporates various disciplines including, but not limited to, data engineering, mathematics, statistics, computing, and domain-specific expertise. A data scientist thus is one who practices some or all aspects of data science in attempting to solve complex data problems. Data analytics is one aspect of data science. Conventional data analytics solutions are becoming more and more limited due to the increasing sizes and varieties of data sets that such solutions are applied against. For example, such limitations include the lack of ability to determine which datasets among the increasing sizes and varieties of data sets are relevant to solutions of any given complex data problems. Accordingly, improved data analytics techniques are needed that enable business users and data scientists to execute data analytics more easily and efficiently.

PRIOR ART SEARCH

US20020107712A1 *2000-12-122002-08-08Lam Kathryn K. Methodology for creating and maintaining a scheme for categorizing electronic communications.
US20110288867A1 *2010-05-182011-11-24General Motors Llc Nametag confusability determination.
US20120330971A1 *2011-06-262012-12-27Itemize Llc Itemized receipt extraction using machine learning.
US20150213372A1 *2014-01-302015-07-30LinkedIn Corporation Systems and methods for email response prediction.
US20150381552A1 *2014-06-302015-12-31Ravi Kiran Holur Vijay Personalized delivery time optimization.
US20020198889A12001-04-262002-12-26International Business Machines Corporation Method and system for data mining automation in domain-specific analytic applications.
US20040128287A12002-12-202004-07-01International Business Machines Corporation Self tuning database retrieval optimization using regression functions.
US20080294583A12007-01-262008-11-27Herbert Dennis Hunt Similarity matching of a competitor's products.
US20100017870A12008-07-182010-01-21Agnik, Llc. Multi-agent, distributed, privacy-preserving data management and data mining techniques to detect cross-domain network attacks.
US20100088284A1 *2008-10-062010-04-08Sap Ag. Archiving system for massive databases using pre-computed data lifecycles.

OBJECTIVES OF THE INVENTION

1. The objective of the invention is to a Apparatuses, systems, methods, technology and AI- Based computer program products are presented for performing data analytics using machine learning.

2. The other objective of the invention is to the invented technology an unsupervised learning module is configured to assemble an unstructured data set into multiple versions of an organized data set.
3. The other objective of the invention is to a supervised learning module is configured to generate one or more machine learning ensembles based on each version of multiple versions of an organized data set and to determine which machine learning ensemble exhibits a highest predictive performance.
4. The other objective of the invention is to the invention also initial work package defines at least one hypothesis associated with a given data problem, and is generated with one or more phases of an automated data analytics lifecycle.
5. The other objective of the invention is to a plurality of datasets is identified. One or more datasets in the plurality of datasets that are relevant to the at least one hypothesis are discovered. The at least one hypothesis is tested using at least a portion of the one or more discovered datasets.
6. The other objective of the invention is to wherein the plurality of unsupervised learning techniques includes identifying relationships in data of the unstructured data set based on one or more queries submitted for the one or more structured data sources.
7. The other objective of the invention is to wherein the plurality of unsupervised learning techniques includes identifying relationships in data of the unstructured data set by identifying semantic distances between data in the unstructured data set.
8. The other objective of the invention is to wherein the plurality of unsupervised learning techniques includes using statistical data to determine a relationship between data in the unstructured data set.
9. The other objective of the invention is to wherein the plurality of unsupervised learning techniques includes identifying relationships in data of the unstructured data set based on an access frequency of data of the unstructured data set.

SUMMARY OF THE INVENTION

An apparatus is presented for performing data analytics using machine learning. In one embodiment, an extract module is configured to extract data from one or more structured data sources. A load module, in a further embodiment, is configured to load data into an unstructured data set. An unsupervised learning module, in certain embodiments, is configured to assemble an unstructured data set into an organized data set using a plurality of unsupervised learning techniques. The performing data analytics using machine learning is presented. In one embodiment, an unsupervised learning module is configured to assemble an unstructured data set into multiple versions of an organized data set. A supervised learning module, in certain embodiments, is configured to generate one or more machine learning ensembles based on each version of multiple versions of an organized data set and to determine which machine learning ensemble exhibits a highest predictive performance.

A method is presented for performing data analytics using machine learning. A method, in one embodiment, includes extracting data from one or more data sources. In a further embodiment, a method includes loading data into an unstructured data set having an unstructured format. A method, in certain embodiments, includes assembling an unstructured data set into an organized data set having a structured format. In another embodiment, a method includes generating one or more learned functions based on an organized data set. a method comprises the following steps. An initial work package is obtained. The initial work package defines at least one hypothesis associated with a given data problem, and is generated in accordance with one or more phases of an automated data analytics lifecycle. A plurality of datasets is identified. One or more datasets in the plurality of datasets that are

relevant to the at least one hypothesis are discovered. The at least one hypothesis is tested using at least a portion of the one or more discovered datasets.

A computer program product is provided which comprises a processor-readable storage medium having encoded therein executable code of one or more software programs. The one or more software programs when executed by one or more processing elements of a computing system implement steps of the above-described method. An apparatus comprises a memory and a processor operatively coupled to the memory and configured to perform steps of the above-described method. Advantageously, illustrative embodiments described herein enable business users and data scientists to leverage methodologies that catalog and describe datasets to support hypothesis tests within a work package that is created to automate a data analytics lifecycle. It is realized here that finding the appropriate datasets for a given analysis or experiment can be one of the most challenging aspects of a data science project. Illustrative embodiments overcome these and other challenges. These and other features and advantages of the present invention will become more readily apparent from the accompanying drawings and the following detailed description.

BRIEF DESCRIPTION OF THE DIAGRAM

FIG. 1: is a schematic block diagram a system for data intelligence.
FIG. 2A: is a schematic block diagram of a data intelligence module.
FIG. 2B: is a schematic block diagram an unsupervised learning module.
FIG. 3: is a schematic block diagram of a supervised learning module.
FIG. 4: is a schematic block diagram of a system for a machine learning factory.
FIG. 5: is a schematic block diagram of learned functions for a machine learning ensemble.
FIG. 6: is a schematic flow chart diagram of a method for a machine learning factory.
FIG. 7: is a schematic flow chart diagram of a method for a machine learning factory.
FIG. 8: is a schematic flow chart diagram of a method for directing data through a machine learning ensemble.
FIG. 9 is a schematic flow chart diagram of a method for data intel FIG. 10A illustrates cloud infrastructure and a data analytics lifecycle automation and provisioning system.
FIG. 10A: is a cloud infrastructure and a data analytics lifecycle automation and provisioning system.
FIG. 10B: is a more detailed view of the cloud infrastructure of FIG. 10A.
FIG. 11: is a processing platform on which the cloud infrastructure and the data analytics lifecycle automation and provisioning system of FIG. 10A are implemented.
FIG. 12: is a data analytics lifecycle automation and provisioning system.
FIG. 13: is a data analytics lifecycle automation and provisioning methodology.
FIG. 14: is a dataset discovery engine and methodology.

DESCRIPTION OF THE INVENTION

FIG. 1 depicts one embodiment of a system 100 for data intelligence. The system 100, in the depicted embodiment, includes a data intelligence module 102. The data intelligence module 102 may be in communication with several data sources 104, other data intelligence modules 102, or the like over a data network 106, over a local channel 108 such as a system bus, an application programming interface (API), or the like. A data source 104 may comprise an enterprise data source, a data storage device, a software application, a database, an input device, a document scanner, a user, a hardware computing device with a processor and memory, or another entity in communication with a data intelligence module 102. In general, the data intelligence module 102 is configured to extract data from one or more structured data sources 104. The extracted data may then be loaded into an unstructured data set. In certain embodiments, the data intelligence module 102 then uses one or more unsupervised learning techniques to identify relationship between data 110 in the unstructured data set and/or assemble the data into an organized data set. The organized data set may include all of that data 110 from the unstructured data set, or a subset of the data 110, such as one or more assembled instances. The resulting organized data set and/or identified relationships can then be used to create learned functions that may provide predictive results based on the data from the structured data sources.

Thus, in certain embodiments, the data intelligence module 102, instead of or in addition to an “Extract”, “Transform,” and “Load” (ETL) process, the data intelligence module 102 may use an “Extract,” “Load,” and “Learn” (ELL) process to assemble data and/or to provide business intelligence using machine learning. Thus, in certain embodiments, this process effectively eliminates the most time consuming step of “Transformation” within the traditional ETL process and then relying on unsupervised and/or supervised learning processes to assemble a meaningful instead of through the traditional use of manual, human intervention with its accompanying errors and bias. The data intelligence module 102 may be configured to identify one or more data sources 104 (e.g., an automated scan, based on user input, or the like) and to extract data 110 from the identified data sources 104 (e.g., “extract” the data 110). Instead of or in addition to transforming the data 110 into a rigid, structured format, in which certain metadata or other information associated with the data 110 and/or the data sources 104 may be lost, incorrect transformations may be made, or the like, the data intelligence module 102 may load the data 110 in an unstructured format and automatically determine relationships between the data 110 (e.g., “load” the data 110). The data intelligence module 102 may use machine learning, as described below, to identify relationships between data in an unstructured format, assemble the data into a structured format, evaluate the correctness of the identified relationships and assembled data, and/or provide machine learning functions to a user based on the extracted and loaded data 110 (e.g., in either a raw or pre-processed form), and/or evaluate the predictive performance of the machine learning functions (e.g., “learn” from the data 110).

The data intelligence module 102 assembles data 110 into an organized format using one or more unsupervised learning techniques. These unsupervised learning techniques can identify relationship between data elements in an unstructured format and use those relationships to provide join instructions and/or to join related data 110. Unsupervised learning is described in greater detail below with reference to FIGS. 2A through 2B. The data intelligence module 102 can use the organized data derived from the unsupervised learning techniques in supervised learning methods to generate one or more machine learning ensembles. These machine learning ensembles may be used to respond to analysis requests (e.g., processing collected and coordinated data using machine learning) and to provide machine learning results, such as a classification, a confidence metric, an inferred function, a regression function, an answer, a prediction, a recognized pattern, a rule, a recommendation, or other results. Supervised machine learning, as used herein, comprises one or more modules, computer executable program code, logic hardware, and/or other entities configured to learn from or train on input data, and to apply the learning or training to provide results or analysis for subsequent data. Supervised learning and generating machine learning ensembles or other machine learning program code is described in greater detail below with reference to FIG. 2A through FIG. 8.

The data intelligence module 102 may provide, access, or otherwise use predictive analytics. Predictive analytics is the study of past performance, or patterns, found in historical and transactional data to identify behavior and trends in unknown future events. This may be accomplished using a variety of techniques including statistics, modeling, machine learning, data mining, and others. One term for large, complex, historical data sets is Big Data. Examples of Big Data include web logs, social networks, blogs, system log files, call logs, customer data, user feedback, RFID and sensor data, social networks, Internet search indexing, call detail records, military surveillance, and complex data in astronomic, biogeochemical, genomics, and atmospheric sciences. These data sets may often be so large and complex that they are awkward and difficult to work with using traditional tools. A Data

Scientist typically must determine the optimal class of learning machines that would be the most applicable for a given data set, and rigorously test the selected hypothesis by first fine-tuning the learning machine parameters and second by evaluating results fed by trained data. The data intelligence module 102, in certain embodiments, generates machine learning ensembles or other machine learning program code for the client's 104, with little or no input from a Data Scientist or other expert, by generating a large number of learned functions from multiple different classes, evaluating, combining, and/or extending the learned functions, synthesizing selected learned functions, and organizing the synthesized learned functions into a machine learning ensemble. The data intelligence module 102, in one embodiment, services analysis requests for the client's 104 using the generated machine learning ensembles or other machine learning program code.

By generating a large number of learned functions, without regard to the effectiveness of the generated learned functions, without prior knowledge of the generated learned functions suitability, or the like, and evaluating the generated learned functions, in certain embodiments, the data intelligence module 102 may provide machine learning ensembles or other machine learning program code that are customized and finely tuned for a particular machine learning application, data from a specific client 104, or the like, without excessive intervention or fine-tuning. The data intelligence module 102, in a further embodiment, may generate and evaluate a large number of learned functions using parallel computing on multiple processors, such as a massively parallel processing (MPP) system or the like. Machine learning ensembles or other machine learning program code are described in greater detail below with regard to FIG. 2A, FIG. 2B, FIG. 3, FIG. 4, and FIG. 5.

The data intelligence module 102 may service machine learning requests to clients 104 locally, executing on the same host computing device as the data intelligence module 102, by providing an API to clients 104, receiving function calls from clients 104, providing a hardware command interface to clients 104, or otherwise providing a local channel 108 to clients 104. In a further embodiment, the data intelligence module 102 may service machine learning requests to clients 104 over a data network 106, such as a local area network (LAN), a wide area network (WAN) such as the Internet as a cloud service, a wireless network, a wired network, or another data network 106. FIG. 2A depicts an embodiment of the data intelligence module 102. In the depicted embodiment, the data intelligence module 102 includes an extract module 202, a load module 204, an unsupervised learning module 208, and a supervised learning module 206. The data intelligence module 102, in one embodiment, uses the extract module 202, the load module 204, the unsupervised learning module 208, and the supervised learning module 206 to perform an extract, load, and learn (ELL), effectively eliminating the need for manual data transformations and providing an additional learning function to derive new meaning and interpretations from extracted data sets.

The extract module 202 is configured to gather, collect, or otherwise extract data from one or more data sources 104. Additionally, in certain embodiments, prior to extracting data, the extract module 202 may identify the data sources 104 from which it will or may extract data 110. For example, the extract module 202 may automatically scan data sources 104 to which the extract module 202 has access to identify available data sources. In another example, the extract module 202 receives manual user input that identifies one or more data sources 104 and/or specific data within the one or more data sources 104 to be extracted. In yet another example, the extract module 202 identifies data sources 104 based on one or more declared business objectives of the data intelligence module 102. The objectives may be received manually or automatically deduced. In certain embodiments, the extract module 202 is configured to extract data from the running data source 104 that is not solely dedicated to providing data to the data intelligence module 102.

The extract module 202 may extract data from its native, structured sources 104. In certain embodiments, the data sources 104 from which data is extracted by the extract module 202 are structured data sources or data sources that primarily include structured data. Structured data includes data with a predictable structure or data model (a description of the objects represented by the data and/or a description of the object's properties and relationships) or is organized in a predefined manner. Conversely, unstructured data is data that does not have a predefined data model (a description of the objects represented by the data and/or a description of the object's properties and relationships) or is not organized in a predefined manner. Semi-structured data is a form of structured data that does not conform with the formal structure of data models associated with relational databases or other forms of data tables, but nonetheless contains tags or other markers to separate semantic elements and enforce hierarchies of records and fields within the data. The extract module 202 can extract various types of data that may be used by the unsupervised learning module 208 and the supervised learning module 206. Non-limiting examples of data 110 that the can extract include, spreadsheets and spreadsheet data, documents, emails, text files, database files, log files, transaction records, purchase orders, metadata, executable code, schema information or definitions, structured query language (SQL) statements, predictive byte code, executable code (with its data manipulation and reporting instructions, such as SQL code), data definition instructions, and other types of data 110. The extract module 202, in a further embodiment, may extract or mine specific feature sets from a data set 110 based on the data set 110's relationships and/or relevance to a declared business goal, as determined by the supervised learning module 206 or the like.

The load module 204 may load the data 110 into an unstructured data set, including a Big Data set. In certain embodiments, the load module 204 may load the data 110 into a relational database management system such as a binary large object (BLOB). The load module 204 may also load the data 110 into an unstructured or semi-structured solution, such as or as an Apache Hadoop or other like solution. The load module 204 can maintain at least a portion or all of the data's original information (e.g., metadata, context, formatting). As such the load module 204 can load data in an unstructured or semi-structured format. By loading data into a large data set of unstructured and/or semi-structured data, the unsupervised learning module 208 and/or the supervised learning module 206 may be able to discover relationships through machine learning as opposed to using manual, human labor. In certain embodiments, the unsupervised learning module may create substantially comprehensive instances to form an organized data set using joins, cross products, or the like, as described herein.

The load module 204 may cooperate with the unsupervised learning module 208 to assemble or restructure the unstructured data set. In some embodiments, the unsupervised learning module 208 is a subcomponent of the load module 204. In other embodiments, these are separate modules, as shown in FIG. 2A. For simplicity of discussion, the following description will refer to these modules as separate module with separate functions, though as mentioned in some embodiments these modules ma shares some or all of their functions. The unsupervised learning module 208 may assemble the unstructured data set, which was loaded by the load module 204, into an organized data set. As mentioned, the organized data set may include all or just some of the data from the unstructured data set. The organized data set may include one or more instances formed by formed by an organizing data elements of the data set using joins, cross products, and other unsupervised learning techniques, as described herein. The organized data set can be a combined, data warehouse, which comprises multiple data marts. The load module 204 and/or the unsupervised learning module 208 can suggest, define, or create data marts, identifying the constituent parts or the like, by combining and analyzing features of disparate tables or other data sources 104. The process of assembling an organized data set may include defining relationships (e.g., connections, distances, and/or confidences) between data elements of the unstructured data set using a plurality of unsupervised learning techniques. In general, unsupervised learning techniques attempt to discover structure in unstructured or semi-structured data. Examples of unsupervised learning techniques described with reference to FIG. 2B.

Optionally, the unsupervised learning module 208 may provide output results (e.g., probabilities, connections, distances, instances, or the like) that inform or populate a probabilistic graph database, a metadata layer for a probabilistic graph database, or the like. The unsupervised learning module 208 may populate other data structures, displays, visualizations, or the like with output results. In some embodiments, the data intelligence module 102 includes or communicates with a visualization module (not shown) for displaying results from the unsupervised learning module 208 and/or the supervised learning module 206. In certain embodiments, the unsupervised learning module 208 can identify or receive target concepts or business objectives that identifies what type of predictions are needed or request by a data intelligence module 102 and/or end user. This may involve requesting manual input from a user or identifying a known objective/concept. Non-limiting examples of business objective may include identify what types of products customers in a given zip code purchase, identifying which department of a company has the highest efficiency or overhead, or identifying what type of product a target demographic is likely to purchase next year. The unsupervised learning module 208 can configure unsupervised learning techniques to identify relationships among data element of the data set that relate to the target concept or business objective. For example, the unsupervised learning module 208 can configure a clustering algorithm (an unsupervised learning technique) to cluster around concepts related to the target concept or business objective.

The unsupervised learning module 208 may use supervised learning, such as one or more machine learning ensembles 222 *a-c* or other predictive programs, to provide feedback to the unsupervised learning module 208. Since the data used by the unsupervised learning module 208 is generally unlabeled, it may be difficult to evaluate the accuracy of the structuring of the resulting organized data set with the unsupervised learning module 208 alone. Accordingly, in certain embodiments, the supervised learning module 206 can evaluate the accuracy of the structure of the organized data set. The supervised learning module 206 can use machine learning to generate one or more learned functions and/or machine learning ensembles 222 *a-c* based on the organized data set. The supervised learning module 206 can then evaluate the predictive performance of the one or more learned functions and/or machine learning ensembles 222 *a-c* to provide an evaluation of the structuring of the organized data set. A detailed description of the general operation of the supervised learning module 206 is provided below with reference to FIGS. 3 to 8.

Given the large amount of processing power and time that may be required by the unsupervised learning module 208 to develop the organized data set, in certain embodiments, the unsupervised learning module 208 is configured to assemble a subset or sample of the unstructured data set into an organized, trial data set. This organized, trial data set may be developed faster since it can have required less processing power and time to process with the unsupervised learning techniques. Additionally, or alternatively, in some embodiments, the unsupervised learning module 208 may be configured to perform abbreviated or partial analysis when developing the organized, trial data set, in order to expedite the development process. The organized, trial data set may be input into the supervised learning module 206 for evaluation, as previously described. Based on the results of the evaluation, the supervised learning module 206 and/or another module of the data intelligence module 102 may assess the accuracy of the organized, trial data set. The unsupervised learning module 208 is configured to assemble the unstructured data set into multiple versions of an organized data set. For instance, the unsupervised learning module 208 can assemble tens, hundreds, or thousands of versions of organized data sets. Each version can be assembled using a unique combination of unsupervised learning techniques and thus each version may identify different relationships between data elements of the data set. Additionally, or alternatively, the unsupervised learning module 208 can assemble two or more versions of organized data sets using the same combination of unsupervised learning techniques, but by varying the parameters, key concepts, or business objectives used by the unsupervised learning techniques. As such each version of the organized data sets may be substantially different. Furthermore, the unsupervised learning module 208 can assemble each of these versions of the organized data set based only on a subset or sample of the unstructured data set, as previously described, such that each version is an organized, trial data set. By assembling a large number of data sets in this way without regard to accuracy, the probability that an accurate data set is developed increases.

To evaluate these versions of the organized data sets, the supervised learning module 206 can be configured to generate one or more machine learning ensemble based on each of the multiple versions of the structured data set. Each of these machine learning ensembles 222 *a-c* can be evaluating by the supervised learning module 206, which can then determine which version exhibits the highest predictive performance. Predictive performance may indicate which machine learning ensemble can predict unknown values with the highest degree of accuracy. These predictions may be evaluated using test data, as discussed herein. The data intelligence module 102 may use the machine learning ensemble with the highest predictive performance to provide predictive functionality to the user. Unused data sets may be discarded. The results of these evaluations may also be utilized by the supervised learning module 206 and/or another module of the data intelligence module 102 to identify which unique combination of unsupervised learning techniques was used to assemble the version of the organized data set that exhibited the highest predictive performance. In instances where the organized data set that exhibited the highest predictive performance is a trial data set, as previously described, the unsupervised learning module 208 can assemble a more complete data set using the same unique combination unsupervised learning techniques used to develop the trial data set, but by processing the complete set of data from the unstructured data set. Similarly, if the unsupervised learning module 208 formed the trial data set using an abbreviated or partial analysis, a complete analysis can be performed. The supervised learning module 206 can then generate one or more learned functions or machine learning ensembles based on the complete data set. These learned functions or machine learning ensembles can be used by the data intelligence module 102 to provide predictive results to the end user(s).

As mentioned, in certain embodiments, the unsupervised learning module 208 is configured to create one or more data sets that can be input into the supervised learning module 206. For example, the organized data set assembled by the unsupervised learning module 208 can be input into the supervised learning module 206. Additionally, in certain embodiments, the unsupervised learning module 208 is configured to create training data from the structured data set. For example, the unsupervised learning module 208 can assemble the data elements in one or more instances that can be used to train the supervised learning module 206. The supervised learning module 206 can be configured to use the training data to generate machine learning ensembles. In one embodiment, the supervised learning module 206 is configured to populate a data visualization tool, a report, or the like based on probabilistic relationships derived from machine learning. These tools may be displayed via a visualization module (not shown), as previously mentioned. The supervised learning module 206, in a further embodiment, may update the original data sources 104, such as one or more databases or the like, with the predicted machine learning results. Alternatively, in some embodiments, the data intelligence module 102 includes an update module (not shown) configured to update the one or more data sources with predicted results generated by the one or more machine learning ensembles.

The supervised learning module 206 may provide machine learning results for strategic decision making and analysis. If the data 110 is material and the unsupervised learning module 208 has made optimal connections, the loaded unstructured data set may not need to be precise. The supervised learning module 206, in certain embodiments, may not provide precise, operational reporting, but accurate analytics reporting. The supervised learning module 206, in one embodiment, may dynamically generate one or more machine learning ensembles 222 *a-c* or other predictive programs, using unstructured or semi-structured data 110 from the load module 204 as training data, test data, and/or workload data, as described below. In this manner, the extract module 202 may first extract data 110 into general buckets (e.g., clusters or focal points), the unsupervised learning module 204 may process and mine the extracted data 110 to form relationships without knowing specific uses for the data 110, just mapping confidence intervals or distances, then feed the data and/or the confidence intervals or distances to the supervised learning module 206 (e.g., the definition of a problem statement, goal, action label, or the like). The supervised learning module 206, in certain embodiments, may provide a report, a visualization, or the like for produced machine learning results or may otherwise catalog the machine learning results for business intelligence or the like.

The load module 204 or the unsupervised learning module 208 may add time variance to the data set, enabling the supervised learning module 206 to refresh or regenerate the machine learning ensembles 222 *a-c* or other predictive programs at various time intervals. The supervised learning module 206 may guide an end-user in terms of governance, prioritization, or the like to find an optimal business value, providing value-based prioritization or the like. As described below with regard to FIGS. 3 and 4, the supervised learning module 206 may be configured to generate machine learning using a compiler/virtual machine paradigm. The supervised learning module 206 may generate a machine learning ensemble with executable program code (e.g., program script instructions, assembly code, byte code, object code, or the like) for multiple learned functions, a metadata rule set, an orchestration module, or the like. The supervised learning module 206 may provide a predictive virtual machine or interpreter configured to execute the program code of a machine learning ensemble with workload data to provide one or more machine learning results.
Reference will now be made to FIG. 2B, which illustrates one embodiment of an unsupervised learning module 208. As shown, embodiments of the unsupervised learning module 208 can include multiple sub-modules, including a clustering module 230, a semantic distance module 232, a metadata mining module 234, a report processing module 236, a data characterization module 238, a search results correlation module 240, a SQL query processing module 242, an access frequency module 244, and an external enrichment module 246. Each of these modules is configured to perform at least one unsupervised learning technique. Unsupervised learning techniques generally seek to summarize and explain key features of a data set. Non-limiting examples of unsupervised techniques include hidden Markov models, blind signal separation using feature extraction techniques for dimensionality reduction, and each of the techniques performed by the modules of the unsupervised learning module 208 (cluster analysis, mining metadata from the data in the unstructured data set, identifying relationships in data of the unstructured data set based on one or more of analyzing process reports and analyzing process SQL queries, identifying relationships in data of the unstructured data set by identifying semantic distances between data in the unstructured data set, using statistical data to determine a relationship between data in the unstructured data set, identifying relationships in data of the unstructured data set based on analyzing the access frequency of data of the unstructured data set, querying external data sources to determine a relationship between data in the unstructured data set, and text search results correlation).

As mentioned, generally the unsupervised learning module 208 can determine relationships between data 110 loaded by the load module 204 into an unstructured data set. For instance, the unsupervised learning module 208 can connect data based on confidence intervals, confidence metrics, distances, or the like indicating the proximity measures and metrics inherent in the unstructured data set, such as schema and Entity Relationship Descriptions (ERD), integrity constraints, foreign key and primary key relationships, parsing SQL queries, reports, spreadsheets, data warehouse information, or the like. For example, the unsupervised learning module 208 may derive one or more relationships across heterogeneous data sets based on probabilistic relationships derived from machine learning such as the unsupervised learning module 208. The unsupervised learning module 208 may determine, at a feature level or the like, the distance between data points based on one or more probabilistic relationships derived from machine learning, such as the unsupervised learning module 208. In addition to identifying simple relationships between data element, the unsupervised learning module 208 may also determine a chain or tree comprising multiple relationships between different data elements. As part of one or more unstructured learning technique the unsupervised learning module 208 may establish a confidence value, a confidence metric, a distance, or the like (collectively "confidence metric") through clustering and/or other machine learning techniques (e.g., the unsupervised learning module 208, the supervised learning module 210) that a certain field belongs to a feature, is associated or related to other data, or the like. For example, if the load module 204 and/or the supervised learning module 206 finds a "ship to zip code" and a "sold to zip code" in two different tables, the load module 204 and/or the supervised learning module 206 may determine certain confidence metrics that they are the same, are related, or the like.

In some unsupervised learning techniques, the unsupervised learning module 208 may determine a confidence that data 110 of an instance belongs together, is related, or the like. The unsupervised learning module 208 may determine that a person and a zip code in one table and a customer number and zip code in another table, belong together and thus join these instances or rows together and provide a confidence metric behind the join. The load module 204 or the unsupervised learning module 208 may store a confidence metric representing a likelihood that a field belongs to an instance and/or a different confidence value that the field belongs in a feature. The load module 204 and/or the supervised learning module 206 may use the confidence values, confidence metrics, or distances to determine an intersection between the row and the column, indicating where to put the field with confidence so that the field may be fed to and processed by the supervised learning module 206. In this manner, the unsupervised learning module 298 and/or the supervised learning module 206 may eliminate a transformation step in data warehousing and replace the precision and deterministic behavior with an imprecise, probabilistic behavior (e.g., store the data in an unstructured or semi-structured manner). Maintaining data in an unstructured or semi-structured format, without transforming the data may allow the load module 204 and/or the supervised learning module 206 to identify signal that would otherwise have been eliminated by a manual transformation, may eliminate the effort of performing the manual transformation, or the like. The unsupervised learning module 208 and/or the supervised learning module 206 may not only automate and make business intelligence more efficient, but may also make business intelligence more effective due to the signal component that may have been erased through a manual transformation.

Referring still to FIG. 2B, in some unsupervised learning techniques, the unsupervised module 206 may make a first pass of the data to identify a first set of relationships, distances, and/or confidences that satisfy a simplicity threshold. For example, unique data, such as customer identifiers, phone numbers, zip codes, or the like may be relatively easy to connect without exhaustive processing. The unsupervised learning module 208, in a further embodiment, may make a second pass of data that is unable to be processed by the unsupervised learning module 208 in the first pass (e.g., data that fails to satisfy the simplicity threshold, is more difficult to connect, or the like). For the remaining data in the second pass, the unsupervised learning module 208 may perform an exhaustive analysis, analyzing each potential connection or relationship between different data elements. For example, the unsupervised learning module 208 may perform additional unsupervised learning techniques (e.g., cross product, a Cartesian joinder, or the like) for the remaining data in the second pass (e.g., analyzing each possible data connection or combination for the remaining data), thereby identifying probabilities or confidences of which connections or combinations are valid, should be maintained, or the like. In this manner, the unsupervised learning module 208 may overcome computational complexity by approaching a logarithmic problem in a linear manner. In some embodiments, the unsupervised learning module 208 and the supervised learning module 206, using the techniques described herein may repeatedly, substantially continuously, and/or indefinitely process data over time, continuously refining accuracy of connections and combinations.

More particular reference will not be made to each of the modules shown in FIG. 2B and each of the unsupervised learning techniques performed by each. As shown, in one embodiment, the unsupervised learning module 208 includes a clustering module 230. The clustering module 230 can be configured to perform one or more clustering analysis on the unstructured data loaded by the load module 204. Clustering involves grouping a set of objects in such a way that objects in the same group (cluster) are more similar, in at least one sense, to each other than to those in other clusters. Non-limiting examples of clustering algorithms include hierarchical clustering, k-means algorithm, kernel-based clustering algorithms, density-based clustering algorithms, spectral clustering algorithms. In one embodiment, the clustering module 230 utilizes decision tree clustering with pseudo labels. In certain embodiments, the clustering module 230 identifies one or more key concepts to cluster around. These key concepts may be based of the key concept or business objective of the data intelligence module 102, as previously mentioned. In some instances, the clustering module 230 may additionally or alternatively cluster around a column, row, or other data feature that have the highest or a high degree of uniqueness. The clustering module 230 may use focal points, clusters, or the like to determine relationships between, distances between, and/or confidences for data. By using focal points, clustering, or the like to break up large amounts of data, the unsupervised learning module 208 may efficiently determine relationships, distances, and/or confidences for the data.
As mentioned, the unsupervised learning module 208 may utilize multiple unsupervised learning techniques to assemble an organized data set. In one embodiment, the unsupervised learning module 208 uses at least one clustering technique to assemble each organized data set. In other embodiments, some organized data sets may be assembled without using a clustering technique. the unsupervised learning module 208 includes a semantic distance module 232. The semantic distance module is configured to identify the meaning in language and words using in the unstructured data of the unstructured data set and use that meaning to identify relationships between data elements. In certain embodiments, the unsupervised learning module 208 includes a metadata mining module 234. The metadata mining 234 modules is configured to data mine declared metadata to identify relationships between metadata and data described by the metadata. For example, the metadata mining module 234 may identify table, row, and column names and draw relationships between them.
In certain embodiments, the unsupervised learning module 208 includes a report processing module 236. The report processing module 236 is configured to analyze and/or read reports and other documents. The report processing module 236 can identify associations and patterns in these documents that indicate how the data in the unstructured data set is organized. These associations and patterns can be used to identify relationships between data elements in the unstructured data set.

The unsupervised learning module 208 includes a data characterization module 238. The data characterization module 238 is configured to use statistical data to ascertain the likelihood of similarities across a column/row family. For example, the data characterization module 238 can calculate the maximum and minimum values in a column/row, the average column length, and the number of distinct values in a column. These statistics can assist the unsupervised learning module to identify the likelihood that two or more columns/row are related. For instance, two data sets that have a maximum value of 10 and 10,000, respectively, may be less likely to be related than two data sets that have identical maximum values. The unsupervised learning module 208 includes a search results correlation module 240. The search results correlation module 240 is configured to correlate data based on common text search results. These search results may include minor text and spelling variations for each word. Accordingly, the search results correlation module 240 may identify words that may be a variant, abbreviation, misspelling, conjugation, or derivation of other words. These identifications may be used by other unsupervised learning techniques.

The unsupervised learning module 208 includes a SQL processing module 242. The search results correlation module 242 is configured to harvest queries in a live database, including SQL queries. These queries and the results of such queries can be utilized to determine or define a distance between relationships within a data set. Similarly, the unsupervised learning module 208 or SQL processing module 242 may harvest SQL statements or other data in real-time from a running database, database manager, or other data source 104. The SQL processing module 242 may parse and/or analyze SQL queries to determine relationships. For example, a WHERE statement, a JOIN statement, or the like may relate certain features of data. The load module 204, in a further embodiment, may use data definition metadata (e.g., primary keys, foreign keys, feature names, or the like) to determine relationships. In certain embodiments, the unsupervised learning module 208 includes an access frequency module 244. The access frequency module 244 is configured to identify correlations between data based on the frequency at which data is accesses, what data is accessed at the same time, access count, time of day data is accessed, and the like. For example, the access frequency module 244 can target highly accessed data first and use access patterns to determine possible relationships. More specifically, the access frequency module 244 can poll a database system's buffer cache metrics for highly accessed database blocks and store that access pattern information in the data set to be used to identify relationships between the highly accessed data.
The unsupervised learning module 208 includes an external enrichment module 246. The external enrichment module 246 is configured to access external sources if the confidence metric between features of a data set is below a threshold. Non-limiting examples of external sources include the Internet, an Internet search engine, an online encyclopedia or reference site, or the like. For example, if a telephone area code column is not related to other columns it may be queried to an external source to establish relationships between telephone area codes and zip codes or mailing addresses. While not an unsupervised learning technique, the unsupervised learning module 208 can be configured to query the user (ask a human) for information that is lacking or for assistance in determining relationships between features of the unstructured data set. In addition to the use of unsupervised learning techniques, the unsupervised learning module 208 can be aided in determining relationships between data elements of the unstructured data set and in assembling organized data sets by the supervised learning module 206. As mentioned, the organized data set(s) assembled by the unsupervised learning module 206 can be evaluated by the supervised learning module 206. Using these evaluations, the unsupervised learning module 208 can identify which relationships are more likely and which are less like. The unsupervised learning module 208 can use that information to improve the accuracy of its processes.The unsupervised learning module 208 may use a machine learning ensemble, such as predictive program code, as an input to unsupervised learning 208 to determine probabilistic relationships between data points. The unsupervised learning module 208 may use relevant influence factors from supervised learning 210 (e.g., a machine learning ensemble or other predictive program code) to enhance unsupervised 208 mining activities in defining the distance between data points in a data set. The unsupervised learning module 208 may define the confidence that a data element is associated with a specific instance, with a specific feature, or the like.

FIG. 3 depicts one embodiment of a supervised learning module 206. As mentioned, the supervised learning module configured to generate one or more machine learning ensembles 222 of learned functions based on the organized data set(s) assembled by the unsupervised learning module 208. In the depicted embodiment, the supervised learning module 206 includes a data receiver module 300, a function generator module 301, a machine learning compiler module 302, a feature selector module 304 a predictive correlation module 318, and a machine learning ensemble 222. The machine learning compiler module 302, in the depicted embodiment, includes a combiner module 306, an extender module 308, a synthesizer module 310, a function evaluator module 312, a metadata library 314, and a function selector module 316. The machine learning ensemble 222, in the depicted embodiment, includes an orchestration module 320, a synthesized metadata rule set 322, and synthesized learned functions 324. The data receiver module 300, in certain embodiments, is configured to receive data from the organized data set, including training data, test data, workload data, or the like, from a client 104, from the load module 204, or the unsupervised learning module 208, either directly or indirectly. The data receiver module 300, in various embodiments, may receive data over a local channel 108 such as an API, a shared library, a hardware command interface, or the like; over a data network 106 such as wired or wireless LAN, WAN, the Internet, a serial connection, a parallel connection, or the like. In certain embodiments, the data receiver module 300 may receive data indirectly from a client 104, from the load module 204, the unsupervised learning module 208 or the like, through an intermediate module that may pre-process, reformat, or otherwise prepare the data for the supervised learning module 206. The data receiver module 300 may support structured data, unstructured data, semi-structured data, or the like.

One type of data that the data receiver module 300 may receive, as part of a new ensemble request or the like, is initialization data. The supervised learning module 206, in certain embodiments, may use initialization data to train and test learned functions from which the supervised learning module 206 may build a machine learning ensemble 222. Initialization data may comprise the trial data set, the organized data set, historical data, statistics, Big Data, customer data, marketing data, computer system logs, computer application logs, data networking logs, or other data that a client 104 provides to the data receiver module 300 with which to build, initialize, train, and/or test a machine learning ensemble 222. Another type of data that the data receiver module 300 may receive, as part of an analysis request or the like, is workload data. The supervised learning module 206, in certain embodiments, may process workload data using a machine learning ensemble 222 to obtain a result, such as a classification, a confidence metric, an inferred function, a regression function, an answer, a prediction, a recognized pattern, a rule, a recommendation, an evaluation, or the like. Workload data for a specific machine learning ensemble 222, in one embodiment, has substantially the same format as the initialization data used to train and/or evaluate the machine learning ensemble 222. For example, initialization data and/or workload data may include one or more features. As used herein, a feature may comprise a column, category, data type, attribute, characteristic, label, or other grouping of data. For example, in embodiments where initialization data and/or workload data that is organized in a table format, a column of data may be a feature. Initialization data and/or workload data may include one or more instances of the associated features. In a table format, where columns of data are associated with features, a row of data is an instance.

As described below with regard to FIG. 4, in one embodiment, the data receiver module 300 may maintain client data (including the organized data set), such as initialization data and/or workload data, in a data repository 406, where the function generator module 301, the machine learning compiler module 302, or the like may access the data. In certain embodiments, as described below, the function generator module 301 and/or the machine learning compiler module 302 may divide initialization data into subsets, using certain subsets of data as training data for generating and training learned functions and using certain subsets of data as test data for evaluating generated learned functions. The function generator module 301, in certain embodiments, is configured to generate a plurality of learned functions based on training data from the data receiver module 300. A learned function, as used herein, comprises a computer readable code that accepts an input and provides a result. A learned function may comprise a compiled code, a script, text, a data structure, a file, a function, or the like. In certain embodiments, a learned function may accept instances of one or more features as input, and provide a result, such as a classification, a confidence metric, an inferred function, a regression function, an answer, a prediction, a recognized pattern, a rule, a recommendation, an evaluation, or the like. In another embodiment, certain learned functions may accept instances of one or more features as input, and provide a subset of the instances, a subset of the one or more features, or the like as an output. In a further embodiment, certain learned functions may receive the output or result of one or more other learned functions as input, such as a Bayes classifier, a Boltzmann machine, or the like.

The function generator module 301 may generate learned functions from multiple different machine learning classes, models, or algorithms. For example, the function generator module 301 may generate decision trees; decision forests; kernel classifiers and regression machines with a plurality of reproducing kernels; non-kernel regression and classification machines such as logistic, CART, multi-layer neural nets with various topologies; Bayesian-type classifiers such as Nave Bayes and Boltzmann machines; logistic regression; multinomial logistic regression; probity regression; AR; MA; ARMA; ARCH; GARCH; VAR; survival or duration analysis; MARS; radial basis functions; support vector machines; k-nearest neighbors; geospatial predictive modeling; and/or other classes of learned functions. In one embodiment, the function generator module 301 generates learned functions pseudo-randomly, without regard to the effectiveness of the generated learned functions, without prior knowledge regarding the suitability of the generated learned functions for the associated training data, or the like. For example, the function generator module 301 may generate a total number of learned functions that is large enough that at least a subset of the generated learned functions are statistically likely to be effective. As used herein, pseudo-randomly indicates that the function generator module 301 is configured to generate learned functions in an automated manner, without input or selection of learned functions, machine learning classes or models for the learned functions, or the like by a Data Scientist, expert, or other user.

The function generator module 301, in certain embodiments, generates as many learned functions as possible for a requested machine learning ensemble 222, given one or more parameters or limitations. A client 104 may provide a parameter or limitation for learned function generation as part of a new ensemble request or the like to an interface module 402 as described below with regard to FIG. 4, such as an amount of time; an allocation of system resources such as a number of processor nodes or cores, or an amount of volatile memory; a number of learned functions; runtime constraints on the requested ensemble 222 such as an indicator of whether or not the requested ensemble 222 should provide results in real-time; and/or another parameter or limitation from a client 104. The number of learned functions that the function generator module 301 may generate for building a machine learning ensemble 222 may also be limited by capabilities of the system 100, such as a number of available processors or processor cores, a current load on the system 100, a price of remote processing resources over the data network 106; or other hardware capabilities of the system 100 available to the function generator module 301. The function generator module 301 may balance the hardware capabilities of the system 100 with an amount of time available for generating learned functions and building a machine learning ensemble 222 to determine how many learned functions to generate for the machine learning ensemble 222.

The function generator module 301 may generate at least 50 learned functions for a machine learning ensemble 222. In a further embodiment, the function generator module 301 may generate hundreds, thousands, or millions of learned functions, or more, for a machine learning ensemble 222. By generating an unusually large number of learned functions from different classes without regard to the suitability or effectiveness of the generated learned functions for training data, in certain embodiments, the function generator module 301 ensures that at least a subset of the generated learned functions, either individually or in combination, are useful, suitable, and/or effective for the training data without careful curation and fine tuning by a Data Scientist or other expert. Similarly, by generating learned functions from different machine learning classes without regard to the effectiveness or the suitability of the different machine learning classes for training data, the function generator module 301, in certain embodiments, may generate learned functions that are useful, suitable, and/or effective for the training data due to the sheer amount of learned functions generated from the different machine learning classes. This brute force, trial-and-error approach to generating learned functions, in certain embodiments, eliminates or minimizes the role of a Data Scientist or other expert in generation of a machine learning ensemble 222.

The function generator module 301, in certain embodiments, divides initialization data from the data receiver module 300 into various subsets of training data, and may use different training data subsets, different combinations of multiple training data subsets, or the like to generate different learned functions. The function generator module 301 may divide the initialization data into training data subsets by feature, by instance, or both. For example, a training data subset may comprise a subset of features of initialization data, a subset of features of initialization data, a subset of both features and instances of initialization data, or the like. Varying the features and/or instances used to train different learned functions, in certain embodiments, may further increase the likelihood that at least a subset of the generated learned functions are useful, suitable, and/or effective. In a further embodiment, the function generator module 301 ensures that the available initialization data is not used in its entirety as training data for any one learned function, so that at least a portion of the initialization data is available for each learned function as test data, which is described in greater detail below with regard to the function evaluator module 312 of FIG. 3.

The function generator module 301 may also generate additional learned functions in cooperation with the machine learning compiler module 302. The function generator module 301 may provide a learned function request interface, allowing the machine learning compiler module 302 or another module, a client 104, or the like to send a learned function request to the function generator module 301 requesting that the function generator module 301 generate one or more additional learned functions. In one embodiment, a learned function request may include one or more attributes for the requested one or more learned functions. For example, a learned function request, in various embodiments, may include a machine learning class for a requested learned function, one or more features for a requested learned function, instances from initialization data to use as training data for a requested learned function, runtime constraints on a requested learned function, or the like. In another embodiment, a learned function request may identify initialization data, training data, or the like for one or more requested learned functions and the function generator module 301 may generate the one or more learned functions pseudo-randomly, as described above, based on the identified data.

The machine learning compiler module 302, in one embodiment, is configured to form a machine learning ensemble 222 using learned functions from the function generator module 301. As used herein, a machine learning ensemble 222 comprises an organized set of a plurality of learned functions. Providing a classification, a confidence metric, an inferred function, a regression function, an answer, a prediction, a recognized pattern, a rule, a recommendation, or another result using a machine learning ensemble 222, in certain embodiments, may be more accurate than using a single learned function. The machine learning compiler module 302 is described in greater detail below with regard to FIG. 3. The machine learning compiler module 302, in certain embodiments, may combine and/or extend learned functions to form new learned functions, may request additional learned functions from the function generator module 301, or the like for inclusion in a machine learning ensemble 222. In one embodiment, the machine learning compiler module 302 evaluates learned functions from the function generator module 301 using test data to generate evaluation metadata. The machine learning compiler module 302, in a further embodiment, may evaluate combined learned functions, extended learned functions, combined-extended learned functions, additional learned functions, or the like using test data to generate evaluation metadata.

The machine learning compiler module 302, in certain embodiments, maintains evaluation metadata in a metadata library 314, as described below with regard to FIGS. 3 and 4. The machine learning compiler module 302 may select learned functions (e.g. learned functions from the function generator module 301, combined learned functions, extended learned functions, learned functions from different machine learning classes, and/or combined-extended learned functions) for inclusion in a machine learning ensemble 222 based on the evaluation metadata. In a further embodiment, the machine learning compiler module 302 may synthesize the selected learned functions into a final, synthesized function or function set for a machine learning ensemble 222 based on evaluation metadata. The machine learning compiler module 302, in another embodiment, may include synthesized evaluation metadata in a machine learning ensemble 222 for directing data through the machine learning ensemble 222 or the like.The feature selector module 304 determines which features of initialization data to use in the machine learning ensemble 222, and in the associated learned functions, and/or which features of the initialization data to exclude from the machine learning ensemble 222, and from the associated learned functions. As described above, initialization data, and the training data and test data derived from the initialization data, may include one or more features. Learned functions and the machine learning ensembles 222 that they form are configured to receive and process instances of one or more features. Certain features may be more predictive than others, and the more features that the machine learning compiler module 302 processes and includes in the generated machine learning ensemble 222, the more processing overhead used by the machine learning compiler module 302, and the more complex the generated machine learning ensemble 222 becomes. Additionally, certain features may not contribute to the effectiveness or accuracy of the results from a machine learning ensemble 222, but may simply add noise to the results.

The feature selector module 304, in one embodiment, cooperates with the function generator module 301 and the machine learning compiler module 302 to evaluate the effectiveness of various features, based on evaluation metadata from the metadata library 314 described below. For example, the function generator module 301 may generate a plurality of learned functions for various combinations of features, and the machine learning compiler module 302 may evaluate the learned functions and generate evaluation metadata. Based on the evaluation metadata, the feature selector module 304 may select a subset of features that are most accurate or effective, and the machine learning compiler module 302 may use learned functions that utilize the selected features to build the machine learning ensemble 222. The feature selector module 304 may select features for use in the machine learning ensemble 222 based on evaluation metadata for learned functions from the function generator module 301, combined learned functions from the combiner module 306, extended learned functions from the extender module 308, combined extended functions, synthesized learned functions from the synthesizer module 310, or the like. The feature selector module 304 may cooperate with the machine learning compiler module 302 to build a plurality of different machine learning ensembles 222 for the same initialization data or training data, each different machine learning ensemble 222 utilizing different features of the initialization data or training data. The machine learning compiler module 302 may evaluate each different machine learning ensemble 222, using the function evaluator module 312 described below, and the feature selector module 304 may select the machine learning ensemble 222 and the associated features which are most accurate or effective based on the evaluation metadata for the different machine learning ensembles 222. In certain embodiments, the machine learning compiler module 302 may generate tens, hundreds, thousands, millions, or more different machine learning ensembles 222 so that the feature selector module 304 may select an optimal set of features (e.g. the most accurate, most effective, or the like) with little or no input from a Data Scientist, expert, or other user in the selection process.

The machine learning compiler module 302 may generate a machine learning ensemble 222 for each possible combination of features from which the feature selector module 304 may select. In a further embodiment, the machine learning compiler module 302 may begin generating machine learning ensembles 222 with a minimal number of features, and may iteratively increase the number of features used to generate machine learning ensembles 222 until an increase in effectiveness or usefulness of the results of the generated machine learning ensembles 222 fails to satisfy a feature effectiveness threshold. By increasing the number of features until the increases stop being effective, in certain embodiments, the machine learning compiler module 302 may determine a minimum effective set of features for use in a machine learning ensemble 222, so that generation and use of the machine learning ensemble 222 is both effective and efficient. The feature effectiveness threshold may be predetermined or hard coded, may be selected by a client 104 as part of a new ensemble request or the like, may be based on one or more parameters or limitations, or the like. During the iterative process, in certain embodiments, once the feature selector module 304 determines that a feature is merely introducing noise, the machine learning compiler module 302 excludes the feature from future iterations, and from the machine learning ensemble 222. In one embodiment, a client 104 may identify one or more features as required for the machine learning ensemble 222, in a new ensemble request or the like. The feature selector module 304 may include the required features in the machine learning ensemble 222, and select one or more of the remaining optional features for inclusion in the machine learning ensemble 222 with the required features.

The based on evaluation metadata from the metadata library 314, the feature selector module 304 determines which features from initialization data and/or training data are adding noise, are not predictive, are the least effective, or the like, and excludes the features from the machine learning ensemble 222. In other embodiments, the feature selector module 304 may determine which features enhance the quality of results, increase effectiveness, or the like, and selects the features for the machine learning ensemble 222. In one embodiment, the feature selector module 304 causes the machine learning compiler module 302 to repeat generating, combining, extending, and/or evaluating learned functions while iterating through permutations of feature sets. At each iteration, the function evaluator module 312 may determine an overall effectiveness of the learned functions in aggregate for the current iteration's selected combination of features. Once the feature selector module 304 identifies a feature as noise introducing, the feature selector module may exclude the noisy feature and the machine learning compiler module 302 may generate a machine learning ensemble 222 without the excluded feature. In one embodiment, the predictive correlation module 318 determines one or more features, instances of features, or the like that correlate with higher confidence metrics (e.g. that are most effective in predicting results with high confidence). The predictive correlation module 318 may cooperate with, be integrated with, or otherwise work in concert with the feature selector module 304 to determine one or more features, instances of features, or the like that correlate with higher confidence metrics. For example, as the feature selector module 304 causes the machine learning compiler module 302 to generate and evaluate learned functions with different sets of features, the predictive correlation module 318 may determine which features and/or instances of features correlate with higher confidence metrics, are most effective, or the like based on metadata from the metadata library 314.

The predictive correlation module 318, in certain embodiments, is configured to harvest metadata regarding which features correlate to higher confidence metrics, to determine which feature was predictive of which outcome or result, or the like. In one embodiment, the predictive correlation module 318 determines the relationship of a feature's predictive qualities for a specific outcome or result based on each instance of a particular feature. In other embodiments, the predictive correlation module 318 may determine the relationship of a feature's predictive qualities based on a subset of instances of a particular feature. For example, the predictive correlation module 318 may discover a correlation between one or more features and the confidence metric of a predicted result by attempting different combinations of features and subsets of instances within an individual feature's dataset, and measuring an overall impact on predictive quality, accuracy, confidence, or the like. The predictive correlation module 318 may determine predictive features at various granularities, such as per feature, per subset of features, per instance, or the like. In one embodiment, the predictive correlation module 318 determines one or more features with a greatest contribution to a predicted result or confidence metric as the machine learning compiler module 302 forms the machine learning ensemble 222, based on evaluation metadata from the metadata library 314, or the like. For example, the machine learning compiler module 302 may build one or more synthesized learned functions 324 that are configured to provide one or more features with a greatest contribution as part of a result. In another embodiment, the predictive correlation module 318 may determine one or more features with a greatest contribution to a predicted result or confidence metric dynamically at runtime as the machine learning ensemble 222 determines the predicted result or confidence metric. In such embodiments, the predictive correlation module 318 may be part of, integrated with, or in communication with the machine learning ensemble 222. The predictive correlation module 318 may cooperate with the machine learning ensemble 222, such that the machine learning ensemble 222 provides a listing of one or more features that provided a greatest contribution to a predicted result or confidence metric as part of a response to an analysis request.

In determining features that are predictive, or that have a greatest contribution to a predicted result or confidence metric, the predictive correlation module 318 may balance a frequency of the contribution of a feature and/or an impact of the contribution of the feature. For example, a certain feature or set of features may contribute to the predicted result or confidence metric frequently, for each instance or the like, but have a low impact. Another feature or set of features may contribute relatively infrequently, but has a very high impact on the predicted result or confidence metric (e.g. provides at or near 100% confidence or the like). While the predictive correlation module 318 is described herein as determining features that are predictive or that have a greatest contribution, in other embodiments, the predictive correlation module 318 may determine one or more specific instances of a feature that are predictive, have a greatest contribution to a predicted result or confidence metric, or the like. In the depicted embodiment, the machine learning compiler module 302 includes a combiner module 306. The combiner module 306 combines learned functions, forming sets, strings, groups, trees, or clusters of combined learned functions. In certain embodiments, the combiner module 306 combines learned functions into a prescribed order, and different orders of learned functions may have different inputs, produce different results, or the like. The combiner module 306 may combine learned functions in different combinations. For example, the combiner module 306 may combine certain learned functions horizontally or in parallel, joined at the inputs and at the outputs or the like, and may combine certain learned functions vertically or in series, feeding the output of one learned function into the input of another learned function.

The combiner module 306 may determine which learned functions to combine, how to combine learned functions, or the like based on evaluation metadata for the learned functions from the metadata library 314, generated based on an evaluation of the learned functions using test data, as described below with regard to the function evaluator module 312. The combiner module 306 may request additional learned functions from the function generator module 301, for combining with other learned functions. For example, the combiner module 306 may request a new learned function with a particular input and/or output to combine with an existing learned function, or the like. While the combining of learned functions may be informed by evaluation metadata for the learned functions, in certain embodiments, the combiner module 306 combines a large number of learned functions pseudo-randomly, forming a large number of combined functions. For example, the combiner module 306, in one embodiment, may determine each possible combination of generated learned functions, as many combinations of generated learned functions as possible given one or more limitations or constraints, a selected subset of combinations of generated learned functions, or the like, for evaluation by the function evaluator module 312. In certain embodiments, by generating a large number of combined learned functions, the combiner module 306 is statistically likely to form one or more combined learned functions that are useful and/or effective for the training data.
The machine learning compiler module 302 includes an extender module 308. The extender module 308, in certain embodiments, is configured to add one or more layers to a learned function. For example, the extender module 308 may extend a learned function or combined learned function by adding a probabilistic model layer, such as a Bayesian belief network layer, a Bayes classifier layer, a Boltzmann layer, or the like.Certain classes of learned functions, such as probabilistic models, may be configured to receive either instances of one or more features as input, or the output results of other learned functions, such as a classification and a confidence metric, an inferred function, a regression function, an answer, a prediction, a recognized pattern, a rule, a recommendation, an evaluation, or the like. The extender module 308 may use these types of learned functions to extend other learned functions. The extender module 308 may extend learned functions generated by the function generator module 301 directly, may extend combined learned functions from the combiner module 306, may extend other extended learned functions, may extend synthesized learned functions from the synthesizer module 310, or the like.

In one embodiment, the extender module 308 determines which learned functions to extend, how to extend learned functions, or the like based on evaluation metadata from the metadata library 314. The extender module 308, in certain embodiments, may request one or more additional learned functions from the function generator module 301 and/or one or more additional combined learned functions from the combiner module 306, for the extender module 308 to extend.While the extending of learned functions may be informed by evaluation metadata for the learned functions, in certain embodiments, the extender module 308 generates a large number of extended learned functions pseudo-randomly. For example, the extender module 308, in one embodiment, may extend each possible learned function and/or combination of learned functions, may extend a selected subset of learned functions, may extend as many learned functions as possible given one or more limitations or constraints, or the like, for evaluation by the function evaluator module 312. In certain embodiments, by generating a large number of extended learned functions, the extender module 308 is statistically likely to form one or more extended learned functions and/or combined extended learned functions that are useful and/or effective for the training data.

The machine learning compiler module 302 includes a synthesizer module 310. The synthesizer module 310, in certain embodiments, is configured to organize a subset of learned functions into the machine learning ensemble 222, as synthesized learned functions 324. In a further embodiment, the synthesizer module 310 includes evaluation metadata from the metadata library 314 of the function evaluator module 312 in the machine learning

ensemble 222 as a synthesized metadata rule set 322, so that the machine learning ensemble 222 includes synthesized learned functions 324 and evaluation metadata, the synthesized metadata rule set 322, for the synthesized learned functions 324.The learned functions that the synthesizer module 310 synthesizes or organizes into the synthesized learned functions 324 of the machine learning ensemble 222, may include learned functions directly from the function generator module 301, combined learned functions from the combiner module 306, extended learned functions from the extender module 308, combined extended learned functions, or the like. As described below, in one embodiment, the function selector module 316 selects the learned functions for the synthesizer module 310 to include in the machine learning ensemble 222. In certain embodiments, the synthesizer module 310 organizes learned functions by preparing the learned functions and the associated evaluation metadata for processing workload data to reach a result. For example, as described below, the synthesizer module 310 may organize and/or synthesize the synthesized learned functions 324 and the synthesized metadata rule set 322 for the orchestration module 320 to use to direct workload data through the synthesized learned functions 324 to produce a result.
In one embodiment, the function evaluator module 312 evaluates the synthesized learned functions 324 that the synthesizer module 310 organizes, and the synthesizer module 310 synthesizes and/or organizes the synthesized metadata rule set 322 based on evaluation metadata that the function evaluation module 312 generates during the evaluation of the synthesized learned functions 324, from the metadata library 314 or the like.

The machine learning compiler module 302 includes a function evaluator module 312. The function evaluator module 312 is configured to evaluate learned functions using test data, or the like. The function evaluator module 312 may evaluate learned functions generated by the function generator module 301, learned functions combined by the combiner module 306 described above, learned functions extended by the extender module 308 described above, combined extended learned functions, synthesized learned functions 324 organized into the machine learning ensemble 222 by the synthesizer module 310 described above, or the like. Test data for a learned function, in certain embodiments, comprises a different subset of the initialization data for the learned function than the function generator module 301 used as training data. The function evaluator module 312, in one embodiment, evaluates a learned function by inputting the test data into the learned function to produce a result, such as a classification, a confidence metric, an inferred function, a regression function, an answer, a prediction, a recognized pattern, a rule, a recommendation, an evaluation, or another result.

Test data, in certain embodiments, comprises a subset of initialization data, with a feature associated with the requested result removed, so that the function evaluator module 312 may compare the result from the learned function to the instances of the removed feature to determine the accuracy and/or effectiveness of the learned function for each test instance. For example, if a client 104 has requested a machine learning ensemble 222 to predict whether a customer will be a repeat customer, and provided historical customer information as initialization data, the function evaluator module 312 may input a test data set comprising one or more features of the initialization data other than whether the customer was a repeat customer into the learned function, and compare the resulting predictions to the initialization data to determine the accuracy and/or effectiveness of the learned function. The function evaluator module 312, in one embodiment, is configured to maintain evaluation metadata for an evaluated learned function in the metadata library 314. The evaluation metadata, in certain embodiments, comprises log data generated by the function generator module 301 while generating learned functions, the function evaluator module 312 while evaluating learned functions, or the like.

The evaluation metadata includes indicators of one or more training data sets that the function generator module 301 used to generate a learned function. The evaluation metadata, in another embodiment, includes indicators of one or more test data sets that the function evaluator module 312 used to evaluate a learned function. In a further embodiment, the evaluation metadata includes indicators of one or more decisions made by and/or branches taken by a learned function during an evaluation by the function evaluator module 312. The evaluation metadata, in another embodiment, includes the results determined by a learned function during an evaluation by the function evaluator module 312. In one embodiment, the evaluation metadata may include evaluation metrics, learning metrics, effectiveness metrics, convergence metrics, or the like for a learned function based on an evaluation of the learned function. An evaluation metric, learning metrics, effectiveness metric, convergence metric, or the like may be based on a comparison of the results from a learned function to actual values from initialization data, and may be represented by a correctness indicator for each evaluated instance, a percentage, a ratio, or the like. Different classes of learned functions, in certain embodiments, may have different types of evaluation metadata.

The evaluation metadata for learned functions to the feature selector module 304, the predictive correlation module 318, the combiner module 306, the extender module 308, and/or the synthesizer module 310. The metadata library 314 may provide an API, a shared library, one or more function calls, or the like providing access to evaluation metadata. The metadata library 314, in various embodiments, may store or maintain evaluation metadata in a database format, as one or more flat files, as one or more lookup tables, as a sequential log or log file, or as one or more other data structures. In one embodiment, the metadata library 314 may index evaluation metadata by learned function, by feature, by instance, by training data, by test data, by effectiveness, and/or by another category or attribute and may provide query access to the indexed evaluation metadata. The function evaluator module 312 may update the metadata library 314 in response to each evaluation of a learned function, adding evaluation metadata to the metadata library 314 or the like.The function selector module 316, in certain embodiments, may use evaluation metadata from the metadata library 314 to select learned functions for the combiner module 306 to combine, for the extender module 308 to extend, for the synthesizer module 310 to include in the machine learning ensemble 222, or the like. For example, in one embodiment, the function selector module 316 may select learned functions based on evaluation metrics, learning metrics, effectiveness metrics, convergence metrics, or the like. In another embodiment, the function selector module 316 may select learned functions for the combiner module 306 to combine and/or for the extender module 308 to extend based on features of training data used to generate the learned functions, or the like.

The machine learning ensemble 222, in certain embodiments, provides machine learning results for an analysis request by processing workload data of the analysis request using a plurality of learned functions (e.g., the synthesized learned functions 324). As described above, results from the machine learning ensemble 222, in various embodiments, may include a classification, a confidence metric, an inferred function, a regression function, an answer, a prediction, a recognized pattern, a rule, a recommendation, an evaluation, and/or another result. For example, in one embodiment, the machine learning ensemble 222 provides a classification and a confidence metric for each instance of workload data input into the machine learning ensemble 222, or the like. Workload data, in certain embodiments, may be substantially similar to test data, but the missing feature from the initialization data is not known, and is to be solved for by the machine learning ensemble 222. A classification, in certain embodiments, comprises a value for a missing feature in an instance of workload data, such as a prediction, an answer, or the like. For example, if the missing feature represents a question, the classification may represent a predicted answer, and the associated confidence metric may be an estimated strength or accuracy of the predicted answer. A classification, in certain embodiments, may comprise a binary value (e.g., yes or no), a rating on a scale (e.g., 4 on a scale of 1 to 5), or another data type for a feature. A confidence metric, in certain embodiments, may comprise a percentage, a ratio, a rating on a scale, or another indicator of accuracy, effectiveness, and/or confidence.

The machine learning ensemble 222 includes an orchestration module 320. The orchestration module 320, in certain embodiments, is configured to direct workload data through the machine learning ensemble 222 to produce a result, such as a classification, a confidence metric, an inferred function, a regression function, an answer, a prediction, a recognized pattern, a rule, a recommendation, an evaluation, and/or another result. In one embodiment, the orchestration module 320 uses evaluation metadata from the function evaluator module 312 and/or the metadata library 314, such as the synthesized metadata rule set 322, to determine how to direct workload data through the synthesized learned functions 324 of the machine learning ensemble 222. As described below with regard to FIG. 8, in certain embodiments, the synthesized metadata rule set 322 comprises a set of rules or conditions from the evaluation metadata of the metadata library 314 that indicate to the orchestration module 320 which features, instances, or the like should be directed to which synthesized learned function 324.

For example, the evaluation metadata from the metadata library 314 may indicate which learned functions were trained using which features and/or instances, how effective different learned functions were at making predictions based on different features and/or instances, or the like. The synthesizer module 310 may use that evaluation metadata to determine rules for the synthesized metadata rule set 322, indicating which features, which instances, or the like the orchestration module 320 the orchestration module 320 should direct through which learned functions, in which order, or the like. The synthesized metadata rule set 322, in one embodiment, may comprise a decision tree or other data structure comprising rules which the orchestration module 320 may follow to direct workload data through the synthesized learned functions 324 of the machine learning ensemble 222.

FIG. 4 depicts one embodiment of a system 400 for a machine learning factory. The system 400, in the depicted embodiment, includes several clients 404 in communication with an interface module 402 either locally or over a data network 106. The supervised learning module 206 of FIG. 4 is substantially similar to the supervised learning module 206 of FIG. 3, but further includes an interface module 402 and a data repository 406. The interface module 402, in certain embodiments, is configured to receive requests from clients 404, to provide results to a client 404, or the like. The supervised learning module 206, for example, may act as a client 404, requesting a machine learning ensemble 222 from the interface module 402 or the like. The interface module 402 may provide a machine learning interface to clients 404, such as an API, a shared library, a hardware command interface, or the like, over which clients 404 may make requests and receive results. The interface module 402 may support new ensemble requests from clients 404, allowing clients 404 to request generation of a new machine learning ensemble 222 from the supervised learning module 206 or the like. As described above, a new ensemble request may include initialization data; one or more ensemble parameters; a feature, query, question or the like for which a client 404 would like a machine learning ensemble 222 to predict a result; or the like. The interface module 402 may support analysis requests for a result from a machine learning ensemble 222. As described above, an analysis request may include workload data; a feature, query, question or the like; a machine learning ensemble 222; or may include other analysis parameters.

The supervised learning module 206 may maintain a library of generated machine learning ensembles 222, from which clients 404 may request results. In such embodiments, the interface module 402 may return a reference, pointer, or other identifier of the requested machine learning ensemble 222 to the requesting client 404, which the client 404 may use in analysis requests. In another embodiment, in response to the supervised learning module 206 generating a machine learning ensemble 222 to satisfy a new ensemble request, the interface module 402 may return the actual machine learning ensemble 222 to the client 404, for the client 404 to manage, and the client 404 may include the machine learning ensemble 222 in each analysis request. The interface module 402 may cooperate with the supervised learning module 206 to service new ensemble requests, may cooperate with the machine learning ensemble 222 to provide a result to an analysis request, or the like. The supervised learning module 206, in the depicted embodiment, includes the function generator module 301, the feature selector module 304, the predictive correlation module 318, and the machine learning compiler module 302, as described above. The supervised learning module 206, in the depicted embodiment, also includes a data repository 406.

The data repository 406, in one embodiment, stores initialization data, so that the function generator module 301, the feature selector module 304, the predictive correlation module 318, and/or the machine learning compiler module 302 may access the initialization data to generate, combine, extend, evaluate, and/or synthesize learned functions and machine learning ensembles 222. The data repository 406 may provide initialization data indexed by feature, by instance, by training data subset, by test data subset, by new ensemble request, or the like. By maintaining initialization data in a data repository 406, in certain embodiments, the supervised learning module 206 ensures that the initialization data is accessible throughout the machine learning ensemble 222 building process, for the function generator module 301 to generate learned functions, for the feature selector module 304 to determine which features should be used in the machine learning ensemble 222, for the predictive correlation module 318 to determine which features correlate with the highest confidence metrics, for the combiner module 306 to combine learned functions, for the extender module 308 to extend learned functions, for the function evaluator module 312 to evaluate learned functions, for the synthesizer module 310 to synthesize learned functions 324 and/or metadata rule sets 322, or the like.

The data receiver module 300 is integrated with the interface module 402, to receive initialization data, including training data and test data, from new ensemble requests. The data receiver module 300 stores initialization data in the data repository 406. The function generator module 301 is in communication with the data repository 406, in one embodiment, so that the function generator module 301 may generate learned functions based on training data sets from the data repository 406. The feature selector module 300 and/or the predictive correlation module 318, in certain embodiments, may cooperate with the function generator module 301 and/or the machine learning compiler module 302 to determine which features to use in the machine learning ensemble 222, which features are most predictive or correlate with the highest confidence metrics, or the like. Within the machine learning compiler module 302, the combiner module 306, the extender module 308, and the synthesizer module 310 are each in communication with both the function generator module 301 and the function evaluator module 312. The function generator module 301, as described above, may generate an initial large amount of learned functions, from different classes or the like, which the function evaluator module 312 evaluates using test data sets from the data repository 406. The combiner module 306 may combine different learned functions from the function generator module 301 to form combined learned functions, which the function evaluator module 312 evaluates using test data from the data repository 406. The combiner module 306 may also request additional learned functions from the function generator module 301.

The extender module 308, in one embodiment, extends learned functions from the function generator module 301 and/or the combiner module 306. The extender module 308 may also request additional learned functions from the function generator module 301. The function evaluator module 312 evaluates the extended learned functions using test data sets from the data repository 406. The synthesizer module 310 organizes, combines, or otherwise synthesizes learned functions from the function generator module 301, the combiner module 306, and/or the extender module 308 into synthesized learned functions 324 for the machine learning ensemble 222. The function evaluator module 312 evaluates the synthesized learned functions 324, and the synthesizer module 310 organizes or synthesizes the evaluation metadata from the metadata library 314 into a synthesized metadata rule set 322 for the synthesized learned functions 324. As described above, as the function evaluator module 312 evaluates learned functions from the function generator module 301, the combiner module 306, the extender module 308, and/or the synthesizer module 310, the function evaluator module 312 generates evaluation metadata for the learned functions and stores the evaluation metadata in the metadata library 314. In the depicted embodiment, in response to an evaluation by the function evaluator module 312, the function selector module 316 selects one or more learned functions based on evaluation metadata from the metadata library 314. For example, the function selector module 316 may select learned functions for the combiner module 306 to combine, for the extender module 308 to extend, for the synthesizer module 310 to synthesize, or the like.

FIG. 5 depicts one embodiment 500 of learned functions 502, 504, 506 for a machine learning ensemble 222. The learned functions 502, 504, 506 are presented by way of example, and in other embodiments, other types and combinations of learned functions may be used, as described above. Further, in other embodiments, the machine learning ensemble 222 may include an orchestration module 320, a synthesized metadata rule set 322, or the like. In one embodiment, the function generator module 301 generates the learned functions 502. The learned functions 502, in the depicted embodiment, include various collections of selected learned functions 502 from different classes including a collection of decision trees 502 *a*, configured to receive or process a subset A-F of the feature set of the machine learning ensemble 222, a collection of support vector machines (“SVMs”) 502 *b* with certain kernels and with an input space configured with particular subsets of the feature set G-L, and a selected group of regression models 502 *c*, here depicted as a suite of single layer (“SL”) neural nets trained on certain feature sets K-N. The example combined learned functions 504, combined by the combiner module 306 or the like, include various instances of forests of decision trees 504 *a* configured to receive or process features N-S, a collection of combined trees with support vector machine decision nodes 504 *b* with specific kernels, their parameters and the features used to define the input space of features T-U, as well as combined functions 504 *c* in the form of trees with a regression decision at the root and linear, tree node decisions at the leaves, configured to receive or process features L-R.

Component class extended learned functions 506, extended by the extender module 308 or the like, include a set of extended functions such as a forest of trees 506 *a* with tree decisions at the roots and various margin classifiers along the branches, which have been extended with a layer of Boltzmann type Bayesian probabilistic classifiers. Extended learned function 506 *b* includes a tree with various regression decisions at the roots, a combination of standard tree 504 *b* and regression decision tree 504 *c* and the branches are extended by a Bayes classifier layer trained with a particular training set exclusive of those used to train the nodes. FIG. 6 depicts one embodiment of a method 600 for a machine learning factory. The method 600 begins, and the data receiver module 300 receives 602 training data. The function generator module 301 generates 604 a plurality of learned functions from multiple classes based on the received 602 training data. The machine learning compiler module 302 forms 606 a machine learning ensemble comprising a subset of learned functions from at least two classes, and the method 600 ends.

FIG. 7 depicts another embodiment of a method 700 for a machine learning factory. The method 700 begins, and the interface module 402 monitors 702 requests until the interface module 402 receives 702 an analytics request from a client 404 or the like. If the interface module 402 receives 702 a new ensemble request, the data receiver module 300 receives 704 training data for the new ensemble, as initialization data or the like.

The function generator module 301 generates 706 a plurality of learned functions based on the received 704 training data, from different machine learning classes. The function evaluator module 312 evaluates 708 the plurality of generated 706 learned functions to generate evaluation metadata. The combiner module 306 combines 710 learned functions based on the metadata from the evaluation 708. The combiner module 306 may request that the function generator module 301 generate 712 additional learned functions for the combiner module 306 to combine. The function evaluator module 312 evaluates 714 the combined 710 learned functions and generates additional evaluation metadata. The extender module 308 extends 716 one or more learned functions by adding one or more layers to the one or more learned functions, such as a probabilistic model layer or the like. In certain embodiments, the extender module 308 extends 716 combined 710 learned functions based on the evaluation 712 of the combined learned functions. The extender module 308 may request that the function generator module 301 generate 718 additional learned functions for the extender module 308 to extend. The function evaluator module 312 evaluates 720 the extended 716 learned functions. The function selector module 316 selects 722 at least two learned functions, such as the generated 706 learned functions, the combined 710 learned functions, the extended 716 learned functions, or the like, based on evaluation metadata from one or more of the evaluations 708, 714, 720.

The synthesizer module 310 synthesizes 724 the selected 722 learned functions into synthesized learned functions 324. The function evaluator module 312 evaluates 726 the synthesized learned functions 324 to generate a synthesized metadata rule set 322. The synthesizer module 310 organizes 728 the synthesized 724 learned functions 324 and the synthesized metadata rule set 322 into a machine learning ensemble 222. The interface module 402 provides 730 a result to the requesting client 404, such as the machine learning ensemble 222, a reference to the machine learning ensemble 222, an acknowledgment, or the like, and the interface module 402 continues to monitor 702 requests. If the interface module 402 receives 702 an analysis request, the data receiver module 300 receives 732 workload data associated with the analysis request. The orchestration module 320 directs 734 the workload data through a machine learning ensemble 222 associated with the received 702 analysis request to produce a result, such as a classification, a confidence metric, an inferred function, a regression function, an answer, a recognized pattern, a recommendation, an evaluation, and/or another result. The interface module 402 provides 730 the produced result to the requesting client 404, and the interface module 402 continues to monitor 702 requests.

FIG. 8 depicts one embodiment of a method 800 for directing data through a machine learning ensemble. The specific synthesized metadata rule set 322 of the depicted method 800 is presented by way of example only, and many other rules and rule sets may be used.
A new instance of workload data is presented 802 to the machine learning ensemble 222 through the interface module 402. The data is processed through the data receiver module 300 and configured for the particular analysis request as initiated by a client 404. In this embodiment the orchestration module 320 evaluates a certain set of features associates with the data instance against a set of thresholds contained within the synthesized metadata rule set 322. A binary decision 804 passes the instance to, in one case, a certain combined and extended function 806 configured for features A-F or in the other case a different, parallel combined function 808 configured to predict against a feature set G-M. In the first case 806, if the output confidence passes 810 a certain threshold as given by the meta-data rule set the instance is passed to a synthesized, extended regression function 814 for final evaluation, else the instance is passed to a combined collection 816 whose output is a weighted voted based processing a certain set of features. In the second case 808 a different combined function 812 with a simple vote output results in the instance being evaluated by a set of base learned functions extended by a Boltzmann type extension 818 or, if a prescribed threshold is meet the output of the synthesized function is the simple vote. The interface

module 402 provides 820 the result of the orchestration module directing workload data through the machine learning ensemble 222 to a requesting client 404 and the method 800 continues.

FIG. 9: depicts one embodiment of a method 900 for performing data analytics using machine learning. The method 900 begins, and the extract module 202 extracts data from one or more data sources (e.g., structured data sources). The load module 204 loads the extracted 902 data into an unstructured data set. The unsupervised learning module 906 assembles 906 the unstructured data set into one or more organized data sets using a plurality of unsupervised learning techniques and the method 900 ends.

FIG. 10A shows a system 100 configured in accordance with an illustrative embodiment of the present invention. The system 100 comprises cloud infrastructure 110 and a data analytics lifecycle automation and provisioning system 120. As will be explained in detail below, the data analytics lifecycle automation and provisioning system 120 enables a data scientist to automatically, yet still interactively, create a work package 122 that can be executed to solve one or more complex data problems. By "work package" it is meant a specific set of instructions that are used for analysis, preparation, and/or support of steps within a data analytic lifecycle (e.g., a data analytic plan) for solving the one or more complex data problems. System 120 accomplishes this, as will be explained in detail below, by providing processing elements that embody phases of a data analytics lifecycle (DAL) including, but not limited to, discovery, data preparation, model planning, model building, and operationalization of results. Cloud infrastructure 110 is illustratively depicted in the figure as comprising an execution environment with execution components comprising one or more central processing units (CPUs) 112, one or more VMs 114, and storage devices 116 (upon which logical units (LUNs) are implemented) that execute one or more processes 118 that operate on one or more process input data sets that generate one or more process output data sets. Thus, the work package generated by system 120 can be operationalized using execution components (both physical and virtual computing resources) in the cloud infrastructure 110. A dataset discovery system and methodologies used to discover datasets usable by the data analytics lifecycle automation and provisioning system 120 will be described below in the context of FIG. 14.
Although system elements 110 and 120 are shown as separate elements in FIG. 10A, these elements or portions thereof may be implemented at least in part on a common processing platform. In other embodiments, one or more of the system elements 110 and 120 may each be implemented on a separate processing platform, such as the processing platform to be described below in conjunction with FIG. 11. For example, the cloud infrastructure 110 may be implemented on a first processing device of a first processing platform and the data analytics lifecycle automation and provisioning system 120 may be implemented on a second processing device of a second processing platform. It is also to be understood that a given embodiment of the system 100 may include multiple instances of the system elements 110 and 120, although only single instances of such elements are shown in the system diagram for clarity and simplicity of illustration. As shown in FIG. 10B, the cloud infrastructure 130 (corresponding to 110 in FIG. 10A) comprises virtual machines (VMs) 132-1, 132-2, . . . 132-N implemented using a hypervisor 134. The hypervisor 134 is an example of what is more generally referred to herein as "virtualization infrastructure." The hypervisor 134 runs on physical infrastructure 136 (e.g., such as may include CPUs 112 and/or storage devices 116 in FIG. 10A). The cloud infrastructure 130 further comprises sets of applications 138-1, 138-2, . . . 138-N running on respective ones of the virtual machines 132-1, 132-2, . . . 132-N (utilizing associated LUNs or virtual disks) under the control of the hypervisor 134.

Although only a single hypervisor 134 is shown in the example of FIG. 10B, a given embodiment of cloud infrastructure configured in accordance with an embodiment of the invention may include multiple hypervisors, each running on its own physical infrastructure. Portions of that physical infrastructure might be virtualized. An example of a commercially available hypervisor platform that may be used to implement portions of the cloud infrastructure 130 (110) in one or more embodiments of the invention is the VMware vSphere® which may have an associated virtual infrastructure management system such as the VMware vCenter®. The underlying physical infrastructure 136 may comprise one or more distributed processing platforms that include storage products such as VNX® and Symmetrix VMAX®, both commercially available from EMC Corporation of Hopkinton, Mass. A variety of other storage products may be utilized to implement at least a portion of the cloud infrastructure 130 (110).

An example of a processing platform on which the cloud infrastructure 110 and/or the data analytics lifecycle automation and provisioning system 120 of FIG. 10A may be implemented is processing platform 200 shown in FIG. 11. The processing platform 200 in this embodiment comprises at least a portion of the system 100 and includes a plurality of servers, denoted 202-1, 202-2, 202-3, . . . 202-P, which communicate with one another over a network 204. One or more of the elements of system 100 may therefore each run on a server, computer or other processing platform element, which may be viewed as an example of what is more generally referred to herein as a “processing device.” As illustrated in FIG. 11, such a device generally comprises at least one processor and an associated memory, and implements one or more functional modules for controlling certain features of system 100. Again, multiple elements or modules may be implemented by a single processing device in a given embodiment. The server 202-1 in the processing platform 200 comprises a processor 210 coupled to a memory 212. The processor 210 may comprise a microprocessor, a microcontroller, an application-specific integrated circuit (ASIC), a field programmable gate array (FPGA) or other type of processing circuitry, as well as portions or combinations of such circuitry elements.

Memory 212 (or other storage device) having program code embodied therein is an example of what is more generally referred to herein as a processor-readable storage medium. Articles of manufacture comprising such processor-readable storage media are considered embodiments of the invention. A given such article of manufacture may comprise, for example, a storage device such as a storage disk, a storage array or an integrated circuit containing memory. The term “article of manufacture” as used herein should be understood to exclude transitory, propagating signals. Furthermore, memory 212 may comprise electronic memory such as random access memory (RAM), read-only memory (ROM) or other types of memory, in any combination. One or more software programs when executed by a processing device such as the processing device 202-1 causes the device to perform functions associated with one or more of the elements/components of system environment 100. One skilled in the art would be readily able to implement such software given the teachings provided herein. Other examples of processor-readable storage media embodying embodiments of the invention may include, for example, optical or magnetic disks.

Also included in the server 202-1 is network interface circuitry 214, which is used to interface the server with the network 204 and other system components. Such circuitry may comprise conventional transceivers of a type well known in the art. The other servers 202 of the processing platform 200 are assumed to be configured in a manner similar to that shown for server 202-1 in the figure. The processing platform 200 shown in FIG. 11 may comprise additional known components such as batch processing systems, parallel processing systems, physical machines, virtual machines, virtual switches, storage volumes, logical units, etc. Again, the particular processing platform shown in the figure is presented by way of example only, and system 200 may include additional or alternative processing platforms, as well as numerous distinct processing platforms in any combination. Also, numerous other arrangements of servers, computers, storage devices or other components are possible in system 200. Such components can communicate with other elements of the system 200 over any type of network, such as a wide area network (WAN), a local area network (LAN), a satellite network, a telephone or cable network, or various portions or combinations of these and other types of networks. Illustrative details of the data analytics lifecycle automation and provisioning system 120 will now be described with reference to FIGS. 3 and 4.

FIG. 12: illustrates a system for assisting the data scientist, inter alia, to overcome the problems mentioned above. More particularly, FIG.12 depicts a data analytics lifecycle automation and provisioning system 300 (e.g., corresponding to system 120 of FIG. 1) that allows a data scientist 301 (or some other user or users, e.g., business user) to design and generate a provisioned system 320 that can be used to analyze and otherwise process data associated with a given complex data problem. As shown, system 300 includes a graphical user interface 302, a discovery module 304, a data preparation module 306, a model planning module 308, a model building module 310, a results communication module 312, an operationalizing module 314, and one or more work packages 316. Note that the components of system 300 in FIG. 12 may be implemented on a single computing system, or one or more components of system 300 may be implemented in a distributed computing system, e.g., across multiple servers 202 in FIG. 11.

The graphical user interface (GUI) 302 is the interface(s) through which the data scientist 301 interacts (e.g., enters data, responses, queries to one or more modules, and receives data, results, and other output generated by one or more modules) with system 300. It is to be understood that the interface used to interact with system 300 does not necessarily have to be a graphical user interface, but rather could be through command lines or some other form of input/output. As such, embodiments of the invention are not limited to any particular form of user interface. Note that the six modules of the system 300 respectively correspond to the phases of a data analytics lifecycle (DAL). FIG. 13 depicts the six phases of a DAL 402, according to one embodiment of the invention, including: a discovery phase 404, a data preparation phase 406, a model planning phase 408, a model building phase 410, a results communication phase 412, and an operationalizing phase 414. Each component of the system 300 assists the data scientist 301 in generating work package 316 that is used to provision the actual analytics system (provisioned system 320) that addresses the given complex data problem.

A description of each DAL phase will now be given with an exemplary problem for which the system 320 is being designed and provisioned. In this example, the problem is a business problem. More specifically, and by way of example only, the business problem is assumed to be the task of accelerating innovation in a global technology corporation. Three aspects of this problem may be: (a) the tracking of knowledge growth throughout the global employee base of the corporation; (b) ensuring that this knowledge is effectively transferred within the corporation; and (c) effectively converting this knowledge into corporate assets. Developing an analytics system (320 in FIG. 12) that executes on these three aspects more effectively should accelerate innovation, which will thus improve the viability of the corporation. Thus, the task of system 300 is to develop such an analytics system. Of course, it is to be understood that this corporate innovation acceleration problem is just one of a myriad of examples of complex data problems that system 300 using DAL 402 can be used to address.

Discovery Phase 404 (Performed by Module 304 in System 300).

The data scientist develops an initial analytic plan. The analytic plan lays the foundation for all of the work in the analytic project being developed to address the business problem. That is, the analytic plan assists the data scientist 301 in identifying the business problem, a set of hypotheses, the data set, and a preliminary plan for the creation of algorithms that can prove or disprove the hypotheses. By way of example only, in the corporate innovation acceleration problem mentioned above, one hypothesis identified by the user as part of the analytic plan may be that an increase in geographic knowledge transfer in a global corporation improves the speed of idea delivery. This hypothesis paves the way for what data will be needed and what type of analytic methods will likely need to be used.

Data Preparation Phase 406 (Performed by Module 306 in System 300).

As the arrows in DAL 402 indicate, the six phases are iterative and interrelated/interconnected, and as such, one phase can be returned to/from one of the other phases in the process. Also, proceeding to the second phase (406) is often a matter of whether or not the data scientist is ready and comfortable sharing the analytic plan developed in the first phase (404) with his/her peers (this comfort level is reflective of the maturity of the analytic plan—if it is too rough and unformed, it will not be ready to be shared for peer review). If so, then the data preparation phase 406 can begin. That is, once the analytic plan has been delivered and socialized, the next step focuses on the data. In particular, the next step is about conditioning the data. The data must be in an acceptable shape, structure, and quality to enable the subsequent analysis. As will be described below in Section II, the discovery module 304 and/or the data preparation module 306 can operate in conjunction with a dataset discovery system to be described below in the context of FIG. 14. In an alternative embodiment, such a dataset discovery system may be implemented as part of the discovery module 304 and/or the data preparation module 306. However, as will be further explained below in the context of FIG. 14, such a dataset discovery system is configured to operate in conjunction with one or more of the other modules (302 through 312) of the system 300.

Model Planning Phase 408 (Performed by Module 308 in System 300).
Model planning represents the conversion of the business problem into a data definition and a potential analytic approach. A model contains the initial ideas on how to frame the business problem as an analytic challenge that can be solved quantitatively. There is a strong link between the hypotheses made in phase 404 (discovery phase) and the analytic techniques that will eventually be chosen. Model selection (part of the planning phase) can require iteration and overlap with phase 406 (data preparation). Multiple types of models are applicable to the same business problem. Selection of methods can also vary depending on the experience of the data scientist. In other cases, model selection is more strongly dictated by the problem set. Described below are a few exemplary algorithms and approaches (but not an exhaustive list) that may be considered by the data scientist 301 in the exemplary accelerated corporate innovation hypothesis given above:
(i) Use Map/Reduce for extracting knowledge from unstructured documents. At the highest level, Map/Reduce imposes a structure on unstructured information by transforming the content into a series of key/value pairs. Map/Reduce can also be used to establish relationships between innovators/researchers discussing the knowledge.
(ii) Natural language processing (NLP) can extract “features” from documents, such as strategic research themes, and can store them into vectors.

At this point in the DAL 402, the data scientist 301 has generated some hypotheses, described potential data sets, and chosen some potential models for proving or disproving the hypotheses.

Model Building Phase 410 (Performed by Module 310 in System 300).

In the model building phase, the system experimentally runs the one or more models that the data scientist 301 selected in phase 408. The model(s) may be executed on a portion of the original (raw) data, a portion of the conditioned data (transformed in phase 406), or some combination thereof. In this phase, the initial data analytic plan is updated to form a refined data analytic plan.For example, Map/Reduce algorithm, NLP, clustering, classification, regression analysis and/or graph theory models are executed by module 310 on a test sample of the data identified and conditioned by module 306 in phase 406 (data preparation). Here the data scientist 301 is able to determine whether the models he/she selected are robust enough (which depends on the specific domain of the data problem being addressed) and whether he/she should return to the model planning phase 408. For example, in the corporate innovation acceleration example, some portion of the data sets identified in the earlier phases (e.g., structured idea submissions and unstructured support documents) is processed with the selected models.

Results Communication Phase 412 (Performed by Module 312 in System 300).
In the results communication phase, the results of the model execution of phase 410 are reported to the data scientist 301 (via GUI 302). This phase is also where the analytic plan that was initially developed in phase 404 and fine-tuned through phases 406, 408 and 410 can be output by the system 300 (i.e., as a refined or final analytic plan). The final analytic plan at this point in the DAL 402 may be referred to as a work package (316 in FIG. 3).

Operationalizing Phase 414 (Performed by Module 314 in System 300).
Operationalizing refers to the process of actually provisioning computing resources (physical and/or virtualized) to generate the system that will be deployed to handle the analytics project in accordance with the final analytic plan, e.g., system 320 in FIG. 12. This may involve provisioning VMs and LUNs as well as other virtual and physical assets that are part of cloud infrastructure 110 in FIG. 1. The provisioned system will then analyze subsequent data that is obtained for the given complex data problem. Given the detailed description of the data analytics lifecycle phases above, we now make some observations and introduce some other features and advantages of the system. Assume that the data scientist 301 is at a later phase in the process but then realizes that he/she forgot to include some data in the discovery phase 404 that is needed to complete the analysis. Advantageously, the interrelated and iterative nature of DAL 402 and the flexibility of the system used to automate the DAL (system 300) provide the data scientist with the ability to return to the discovery phase, correct the error, and return to a subsequent stage with the results for each stage affected by the change being automatically updated.

During the model building phase 410, it is not known what resources are going to be needed, which have a specific cost, and definition of what would be included (amount of storage, number of VMs, the analytics tools needed, etc.). Being able to know the approximate cost and configuration needed would be very useful for the process of tuning the model based on cost or configuration constraints. Thus, during each phase of the DAL 402, the data scientist 301 is presented (at GUI 301) with an inventory of the current infrastructure, services, and tools needed and their approximate cost as changes are made to the parameters associated with the analysis. This will be further described below in the context of FIG. 14. This allows the data scientist to remove or change the model dynamically based on resource constraints (e.g., cost or VM limits). Once the analytics work package 316 is defined, provisioning the resources needed to most efficiently support the analysis is important. As such, embodiments of the invention automate and execute work packages for the data scientist by constructing the work package and providing resource and cost estimates throughout the DAL.
Many times, introducing new raw, source data sets into a project can have cascading effects on the size of the analytic sandbox (see data preparation phase 406 above) needed to support the analysis. Embodiments of the invention provide selectable sizing multiples to dynamically provision the system parameters, such as a storage capacity, bandwidth required, and compute power depending on the type of new data involved and its size. For example, these sizing multiples could be used between phases 404 and 406, between 406 and 408, and even between phase 408 and 410. The sizing multiples serve as a mechanism for dynamically provisioning and adjusting the size, capacity, and constraints needed for the analytic sandbox.

By way of example only, assume there is 100 GB worth of innovation data that is to be analyzed. The data preparation module 306 multiplies this value by some constant (e.g., 10 or 20 times) in order to estimate the capacity of the analytic sandbox. That is, the data scientist will take the 100 GB of data and run transformations and other experiments that will require additional amounts of capacity. Therefore, the data preparation module 306 creates a work package specification that states: “allocate 1 TB of sandbox data which has the following features . . . . ” This aspect of the work package instructs cloud provisioning software to allocate appropriately. It is also realized that privacy of data is a major concern when mining large amounts or correlating various types of data. Privacy of the individuals needs to be protected while still allowing useful analysis and presentation of the data. Embodiments of the invention provide for masking capabilities in the work package 316, as well as any data presented by the system, for the data scientist, as well as creating contextual views based on the identity of the consumer of the output. This feature is very useful, particularly in a highly regulated data environment.

FIG. 14: is a dataset discovery engine and methodology, in accordance with one embodiment of the invention. As shown, dataset discovery engine 500 is operatively coupled to a data analytics lifecycle 510 and a work package 512. It is to be appreciated that the data analytics lifecycle 510 and the work package 512 represent, in one embodiment, data analytics lifecycle automation and provisioning system 300 whereby data analytics lifecycle 510 is automated to generate work package 512. Dataset discovery engine 500 includes a data crawling agent 502, a feature extractor 504, and a data store 506. Data crawling agent 502, within the context of the work package 512, crawls local and wide area public and private information networks to identify, collect and catalog datasets. These datasets can be comprised of, by way of example only, text, email, images, video, voice mail, and more traditional RDBMS (relational database management system) structures. The data crawling agent 502 write log files cataloguing the data it finds, and also describe these data sources using metadata, such as type of data, size, and topic areas. Such data catalog files (data catalogs) are stored in data store 506.
Thus, for example, the data crawling agent 502 locates, in an enterprise (or outside the enterprise), relevant datasets that satisfy the hypothesis from the work package 512. These datasets can be dynamically located or located from pre-stored datasets. The data crawling agent 502 also filters the located datasets based on what the agent learns from examining and/or analyzing the work package 512. In one example, the agent filters hypothesis-relevant data from non-hypothesis-relevant data. This can be done by identifying associations between datasets in the data catalogs. Also, provenance data can be generated and used for datasets in the data catalogs. Feature extractor 504 extracts features from the datasets identified by the data crawling agent 502. By interpreting data attributes written to the log files that describe the enterprise's data elements, it is possible to infer features of these datasets. These features could relate to the presence of conditions or properties related to the data, such as, but not limited to, its rate of change, the presence of information that contains personally identifiable information (P1), and the type of data, both in terms of the structure (such as, for example, unstructured, quasi-structured, semi-structured, and structured) and its potential use cases (for example, revenue information that can be used for a financial analysis byproduct). When the feature extractor 504 determines the presence of private data (PII) in the identified datasets, the engine 500 can prevent use of the private data from the identified datasets. One prevention example is by masking the private data in the identified datasets.

WE CLAIMS

1. My Invention “**DAI- Dataset Discovery”** is a to Apparatuses, systems, methods, technology and AI- Based computer program products are presented for performing data analytics using machine learning. The invented technology an unsupervised learning module is configured to assemble an unstructured data set into multiple versions of an organized data set. A supervised learning module is configured to generate one or more machine learning ensembles based on each version of multiple versions of an organized data set and to determine which machine learning ensemble exhibits a highest predictive performance. The invention also initial work package defines at least one hypothesis associated with a given data problem, and is generated with one or more phases of an automated data analytics lifecycle. A plurality of datasets is identified. One or more datasets in the plurality of datasets that are relevant to the at least one hypothesis are discovered. The at least one hypothesis is tested using at least a portion of the one or more discovered datasets.
2. According to claim1# The invention is to a to Apparatuses, systems, methods, technology and AI- Based computer program products are presented for performing data analytics using machine learning.
3. According to claim1,2# The invention is to the invented technology an unsupervised learning module is configured to assemble an unstructured data set into multiple versions of an organized data set.
4. According to claim1,2,3# The invention is to a supervised learning module is configured to generate one or more machine learning ensembles based on each version of multiple versions of an organized data set and to determine which machine learning ensemble exhibits a highest predictive performance.
5. According to claim1,2,4# The invention is to the invention also initial work package defines at least one hypothesis associated with a given data problem, and is generated with one or more phases of an automated data analytics lifecycle.
6. According to claim1,2,4# The invention is to a plurality of datasets is identified. One or more datasets in the plurality of datasets that are relevant to the at least one hypothesis are discovered. The at least one hypothesis is tested using at least a portion of the one or more discovered datasets.
7. According to claim1,2,6# The invention is to wherein the plurality of unsupervised learning techniques includes identifying relationships in data of the unstructured data set based on one or more queries submitted for the one or more structured data sources.
8. According to claim1,2,6# The invention is to wherein the plurality of unsupervised learning techniques includes identifying relationships in data of the unstructured data set by identifying semantic distances between data in the unstructured data set.

9. According to claim1,2,5# The invention is to wherein the plurality of unsupervised learning techniques includes using statistical data to determine a relationship between data in the unstructured data set.
10. According to claim1,2,5# The invention is to wherein the plurality of unsupervised learning techniques includes identifying relationships in data of the unstructured data set based on an access frequency of data of the unstructured data set.

Date: 15/8/20

Dr. Surbhi Bhatia (Assistant Professor)
Dr. Mohammad Tabrez Quasim
Ms. Rashi Kohli (Senior Member IEEE)
Dr. Kapal Dev, Affiliation: Trinity College Dublin, Ireland
Dr. Anchal Garg (Associate Professor, IEEE Senior Member)
Prof.(Dr.) Rakesh Kumar. E R (Manager Examination)
Dr. Meghna Chhabra (Associate Professor)
Dr. Ankur Singh Bist (Associate Professor)
Mr. Revanth Madamala
Prof.(Dr.) Biplab Kumar Sarkar (Founder GEH Research LLP & Project Director TCS-G)

Patent Title: **DAI- Dataset Discovery**: DATASET DISCOVERY IN **D**ATA **A**NALYTICS USING A**I**- BASED PROGRAMMING.

ABSTRACT

My Invention "**DAI- Dataset Discovery"** is a to Apparatuses, systems, methods, technology and AI- Based computer program products are presented for performing data analytics using machine learning. The invented technology an unsupervised learning module is configured to assemble an unstructured data set into multiple versions of an organized data set. A supervised learning module is configured to generate one or more machine learning ensembles based on each version of multiple versions of an organized data set and to determine which machine learning ensemble exhibits a highest predictive performance. The invention also initial work package defines at least one hypothesis associated with a given data problem, and is generated with one or more phases of an automated data analytics lifecycle. A plurality of datasets is identified. One or more datasets in the plurality of datasets that are relevant to the at least one hypothesis are discovered. The at least one hypothesis is tested using at least a portion of the one or more discovered datasets.