A conceptual vision toward the management of Machine Learning models[⋆]

Daniel N. R. da Silva[1], Adolfo Simões[1], Carlos Cardoso[1], Douglas E. M. de Oliveira[1], João N. Rittmeyer[1], Klaus Wehmuth[1], Hermano Lustosa[1], Rafael S. Pereira[1], Yania Souto[1], Luciana E. G. Vignoli[2], Rebecca Salles[2], Heleno de S. C. Jr[4], Artur Ziviani[1], Eduardo Ogasawara[2], Flavia C. Delicato[3], Paulo de F. Pires[3], Hardy Leonardo da C. P. Pinto[5], Luciano Maia[5], and Fabio Porto[1]

[1] National Laboratory for Scientific Computing (LNCC), Brazil
[2] Federal Center for Technological Education of Rio de Janeiro (CEFET-RJ), Brazil
[3] Federal University of Rio de Janeiro (UFRJ), Brazil
[4] Federal Fluminense University (UFF), Brazil
[5] CENPES/Petrobrás, Brazil
{dramos,ziviani,fporto}@lncc.br

Abstract. To turn big data into actionable knowledge, the adoption of machine learning (ML) methods has proven to be one of the de facto approaches. When elaborating an appropriate ML model for a given task, one typically builds many models and generates several data artifacts. Given the amount of information associated with the developed models performance, their appropriate selection is often difficult. Therefore, appropriately comparing a set of competitive ML models and choosing one according to an arbitrary set of user metrics require systematic solutions. In particular, ML model management is a promising research direction for a more systematic and comprehensive approach for machine learning model selection. Therefore, in this paper, we introduce a conceptual model for ML development. Based on this conceptualization, we introduce our vision toward a knowledge-based model management system oriented to model selection.

Keywords: Machine Learning · Model Management · Knowledge Bases

1 Introduction

The increasing production and availability of large and diverse data, the recent advancements in machine learning research, and the cheapening of digital processing costs are together promoting the emergence of a new paradigm in our society: humankind increasingly ground its decisions on data. This new paradigm impacts many, not to say all, society activities, including scientific development, corporate world decisions, and government policies [7]. Scientists more and more

[⋆] The authors thank the partial support of the Brazilian funding agencies: CAPES, CNPq, FAPERJ, and FAPESP, as well as the project no. 5850.0108913.18.9/4600567373 funded by Petrobras.

elaborate hypothesis and make scientific discoveries using data analysis techniques [2, 21]. Likewise, enterprises are progressively adopting business intelligence strategies and data leveraged technologies in their decision-making processes [5]. To improve their productive capacity, industries monitor and analyze data retrieved by sensors placed all around their manufacturing plants. Finally, foreseeing assets for public policymaking, government agencies have increased funding on data science research and development.

The employment of machine learning (ML) techniques is at the core of several applications, but still lacks systematic solutions. Data analytics has been successfully employed in data classification, data clustering, and rare events discovery [16]. However, the development of ML is still mainly ad-hoc, characterized by a trial-and-error nature. Since there is no unique ML algorithm that works best for every dataset [23], it is necessary to test and manage different ML algorithms configurations and models to meet one's criteria.

The development and maintenance of ML models yields many interrelated artifacts, e.g., models, algorithm configurations, intermediate and transformed datasets, annotations, and scripts. As applications complexity increases, it becomes harder to manage the produced artifacts volume without a systematic approach. For example, to choose a set of appropriate models for their application, users have to understand and trace back the steps taken during model development. They gather data and information on preprocessing, feature engineering, algorithm parametrization as well as compare the performance of developed models according to a set of metrics. The integration of this information to more easily develop better models makes the management of ML models essential.

Machine learning model management encompasses the systematic process of managing relevant artifacts—e.g., models and datasets— produced during the development, deployment, and monitoring of ML models [15]. Due to the relevance of this task, the machine learning and data management communities have recently raised joint efforts to build systems and tackle challenges on model management [11, 19, 20, 24]. Nevertheless, how the management of ML models can leverage their selection according to the application domain features remains an open question.

In model selection, ML management systems have to take into account diverse ML models, user constraints on data domains, and model performance metrics. In particular, for a given domain $D = (\mathcal{X}, \mathcal{Y})$ and a target unknown function $f : \mathcal{X} \to \mathcal{Y}$, ML management systems should consider (a) a set $H = \{h_1, ..., h_k\}$ of predictive models for f respectively trained on datasets $D_{h_i} \subsetneq D$; (b) a set $C = \{c_1, ..., c_m\}$ of users performance criteria which are functions of ML model performance metrics; and (c) a set of data regions $Q = \{q_1, ..., q_n\}, q_i \subsetneq \mathcal{X}$, where each region possibly contains $x \in \mathcal{X}$ absent of all H training sets. These systems should also deal with situations where no model in H meets the criteria set on a specified region. In this case, the system should automatically employ techniques as model ensembling and automated machine learning to extend H with performative models on the specified region.

In this paper, we present a conceptual framework for the design of Gypscie[6], an ML model management system oriented to model selection. In particular, our contribution is twofold: (a) we describe the main actors and relationships in ML development in a conceptual model that contributes to elicit challenges and research opportunities involved in the management task; and (b) we present our vision toward Gypscie, which includes features for ML data/knowledge integration as well as machine learning lifecycle and model management.

The paper goes as follows. In Section 2, we present a user scenario that motivates the design of Gypscie. In Section 3, we briefly review systems related to the emerging area of management in machine learning. In Section 4, we present our conceptual model for the machine learning lifecycle. In Section 5, we present Gypscie intended features. Finally, in Section 6, we make our final remarks and point out future research directions.

2 Motivating Scenario

In this section, we present a scenario that illustrates the relevance for ML management functionalities supporting users in finding a model that best fits her predictive criteria.

Let us consider the meteorology domain, where a myriad of ML and physics models may be used for temperature prediction, as one can attest by visiting the MASTER website.[7] We also consider a data scientist — Bob — user of the MASTER website, aiming at improving the quality of the predictions. A common approach, given such a variety of available models, would be to adopt an ensemble approach. The latter combines a selection of diverse and potentially simpler models to compute a new one, leveraging the predictive capability of each individual component. In Souto et al. [18], an ensemble approach, using a Deep Neural Network, combines the predictions of independent models registered in MASTER, to produce more accurate temperature predictions in the region of the state of São Paulo, see Figure 1. The ensemble model predictions are compared against registered observations of the same area to produce a color map indicating the error level at each spatial position, where closer to yellow represents a higher predictive error.

One may observe, in Figure 1, that despite combining models with different prediction capabilities, the resulting predictions still highlight a distribution of error level through the covered spatial region.

Now, let us add a second user, Anna, to this scenario, interested in planting coffee during the raining season and looking for days with temperatures between 18°C to 20°C, in the north-west region of the São Paulo state. It is unlucky that the region observes a prediction error level produced by the new model that is beyond acceptable limits.

Anna could envisage a few alternatives. For example, she could ask Bob to search for new training data covering that particular area of interest and use it

[6] The name Gypscie is a combination of the words **Gy**psy and **Scie**nce.
[7] http://www.master.iag.usp.br/lab/

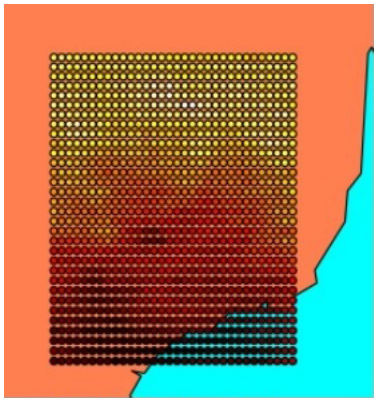

Fig. 1. Distribution of error level of temperature predictions produced by an ensemble ML model in a region of the state of São Paulo. Closer to yellow indicates higher predictive error [17].

to retrain a model component, recomputing the ensemble model. An alternative, in the absence of new training data, could be to look for an existing model, not previously selected to make up the ensemble due to higher global *Root Mean Square Error* (RMSE), but that would offer a smaller local error at the region Anna intends to guide the coffee plantation.

As one may capture from this scenario, obtaining an optimal prediction is a daunting task. Firstly, it depends on selecting the model that best predicts the target variable for a given query. Secondly, as one single model may not be appointed, the choice may fall in combining the predictions of different models. Finally, in a dynamic environment such as meteorology, changes in the feature space may induce changes to the target variable, a phenomenon known as *Concept drift* [22]. In this context, it may be required to retrain some of the models with the identification of new training data sets and the selection of the models to be retrained. These are very complex tasks that become impossible to be managed in the absence of a systematic approach.

3 Related works

The development of ML models comprises many stages from the problem specification up to the model deployment. Throughout these stages, machine learning users resort to a series of software tools related to the management of machine learning models. We divide them into five non-exclusive categories according to their main functionality: (a) data management systems in machine learning; (b) model development systems; (c) systems for the management of ML model lifecycle (d) systems for the management of ML models; and (e) model serving systems. Hereafter, we briefly review some of these systems and point out how they are relevant and complementary to a management system like Gypscie.

Data management in machine learning Many challenges arise in the management of data produced during ML lifecycle, e.g., data understanding, validation, cleaning, and preparation [14]. A promising research direction to tackle these challenges is the development of solutions for dataset versioning and provenance such as OrpheusDB and Datahub. OrpheusDB [8] is a relational database system with versioning capabilities aiming at two characteristics inherent to ML data, namely, data change and provenance. Datahub [3] is a platform for data analysis built on top of a dataset versioning system. The platform enables its users to load, store, query, collaboratively analyze, interactively visualize, interface with external applications, and share datasets.

Machine learning model development The machine learning community has arguably focused on the development of systems, frameworks, and libraries for model production (e.g., Scikit-Learn, TensorFlow, and MLib). Scikit-Learn [12] is one of the most used open-source ML libraries. It covers many supervised and unsupervised learning methods, including algorithms for classification, regression, clustering, and dimensionality reduction. Tensorflow [1] is a multiplatform framework for deep learning development. Besides the artificial neural networks development API, the framework also provides visualization (TensorBoard) and debugging (TFDBG) tools. Finally, MLib [10] is a machine learning library built on top of Apache Spark. The library encompasses ML methods for classification, regression, clustering, recommendation, and hyperparameter tuning tasks.

Machine learning lifecycle management In model diagnostics tasks, ML users consider a large artifacts set produced during the model lifecycle, e.g., algorithm configurations, input/output data, and experiment files such as data scientists annotations [15]. These artifacts management is a burden to the users. Therefore, ML and data management communities have recently developed systems for the management of models lifecycle, for example, Mlflow, ModelDB, and Mistique.

MLflow [24] is an open-source platform that provides a set of features for instrumentation, reproducibility, and ML model deployment. The platform has three components—Tracking, Project, and Model—which ML users respectively use to track models data, package ML projects, and store ML models. ModelDB [19] is a system to track, store and index modeling artifacts (e.g., hyperparameters associated with trained models, quality metrics, and input data). The system has three components: (a) library wrappers for code instrumentation and logging of models data; (b) an interface to model storage; and (c) an exploratory visual interface for model diagnostics. Mistique [20] is a model diagnostics system aiming at the retrieval, storage, and querying of large model intermediates (e.g., input data and hidden representations yielded by a neural network). The system implements a series of techniques (e.g., discretization, summarization, and compression) to tackle the trade-off between storing and recomputing intermediates.

Machine learning model management In a narrower definition, the management of ML models aims at providing more specific features, e.g., the efficient storage and retrieval of ML models, and model operations such as selection, composition, and comparison. ModelHub [11] is a system that partially implements these features for neural networks management. The system comprises three components: (a) a neural network version control system; (b) a domain-specific language to the adjustment of network parameters and hyperparameters; and (c) a system for deep learning models hosting and sharing.

Machine learning model serving ML projects increasingly require real-time, accurate, and robust predictions under heterogeneous and massive production workloads. However, few systems aim at model deployment and monitoring tasks which are essential for serving an ML model to production. In this context, Clipper [6] is a low latency prediction serving system. The system encapsulates frameworks and models heterogeneity using containerization technology. It also implements a set of techniques to improve prediction accuracy and latency by model composition and selection.

To comprehensively perform model selection, Gypscie ought to retrieve model artifacts scattered throughout the whole ML lifecycle, from input data preprocessing to model serving. Firstly, the input data transformation process significantly determines the performance of ML models [14]. Therefore, Gypscie is going to employ techniques and systems for data management, versioning, and provenance. Also, Gypscie has to take into account the model development process. The nature of a model (i.e., its properties and relationships) and its generative process underlie the model selection task. The management of ML lifecycle is also fundamental to Gypscie goals. In model selection, Gypscie ought to consider data and metadata associated with the lifecycle of relevant models. Finally, model serving is complementary to Gypscie. Gypscie is going to use model serving systems to abstract away insignificant production details and consequently more easily select and compare deployed models.

Gypscie aims at a particular research problem: *model selection, for complex data domains, based on the comprehensive comparison of ML models.* This research problem fundamentally differs from systems on categories (a), (b), (c), and (e). Also, in model management (e), to the best of our knowledge, no system has the same goal.

4 Conceptualization

Gypscie should cover the set of entities and relationships usual to ML literature and practice. Thus, we have organized this set into a conceptual model for ML model lifecycle, shown in Figure 2.

[9] We drew this diagram in Dia Diagram Editor (http://dia-installer.de/) using Martin Style Cardinality. We have also omitted entities attributes to make the diagram clearer and more general.

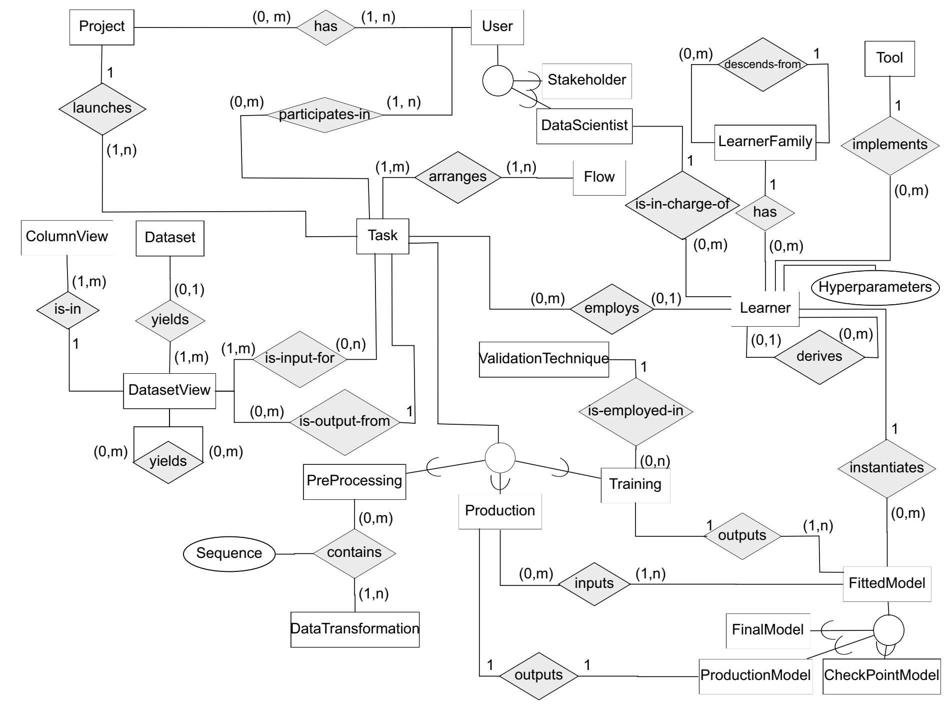

Fig. 2. Extended Entitity-Relationship diagram depicting ML lifecycle. Relationships should be read from up to bottom, from left to right.[9]

The most comprehensive entity in our conceptual model is the *Project*. Projects encompass the intent to solve problems using ML techniques. For example, in one project, data scientists can develop ML models for predicting temperature levels in the Northwest region of São Paulo during the first week of May. To capture Data Science project modularity, we divide projects into a collection of *Tasks* corresponding to minimally independent sub-problems.

In ML practice, project roles (e.g., data analyst, scientist, engineer, and so on) do not observe consistent definitions. Therefore, we assume two sets of people involved in an ML project: (i) *Stakeholders*, individuals not involved in technical project tasks (e.g., team managers and domain experts); and (ii) *DataScientists*, individuals who technically carry out the project tasks.

Tasks establish the methodology to (partially) solve (sub)problems. According to the nature of the target problem, we classify tasks as *PreProcessing*, *Training*, or *Production*. The conjunction of these task types captures machine learning lifecycle. In particular, in a pre-processing task, data scientists carry out a sequence of *DataTransformations*—e.g., data validation, cleaning, and preparation—to emphasize important input data features. On the other hand, in a training task, data scientists build, evaluate, and validate machine learning models. In a production task, data scientists turn models into pieces of deployable and maintainable software. Finally, to explicitly capture the model and data dependency between Tasks, users restort to *Flow*s, e.g., the input dataset of a training task b is the output of a preprocessing task a.

The model concept is often used ambiguously in machine learning practice. Thus, we divide this concept into three main entities, *LearnerFamily*, *Learner*, and *FittedModel*. Specifically, *LearnerFamilies* stand for ML algorithms in a broad sense, e.g., neural networks and decision trees, including their inductive biases. On the other hand, by instantiating a *LearnerFamily*, *Learner*s are ML methods capable of fitting data. A learner includes the functional form, hyper-parameters, and parametric space definition (for parametric models) necessary for the elaboration of an ML model. *FittedModels* are functions produced by fitting a Learner to some dataset. The Fitted Model specializations include *FinalModel*, *ProductionModel*, and *CheckPointModel*. While a *FinalModel* is the best model according to some validation strategy (*ValidationTechnique*), a *CheckPointModel* is a model saved during training based on some secondary user criteria. *ProductionModel*s are deployable and maintainable ML models. Finally, every learner and model is developed in an ML *Tool*, e.g., scikit-learn [12].

In machine learning, a *Dataset* usually goes through a series of *DataTransformation*s, e.g., data imputation and feature extraction. Moreover, not all dataset information may be relevant to one's task; e.g., she may be interested only in particular columns and rows of a tabular dataset. In this regard, *DatasetView*s are tabular partitions of data derived from datasets or other dataset views. We associate descriptive statistics with each column of a dataset view.

Dataset views are input and output to preprocessing, training, and production tasks. In particular, we establish the following constraints on *is-input-for* and *is-output-for* relationships: (i) training tasks input only one dataset view and output zero or more dataset views; (ii) production tasks input and output one or more dataset views; and (iii) preprocessing tasks input and output one more dataset views.

5 Gypscie Features

Gypscie aims at the constrained selection and comparison of competitive models, i.e., selecting and comparing models for the same phenomenon depending on their performance at specific data domain regions. This application requires a comprehensive understanding of the available models, which in turn depends on artifacts scattered throughout their lifecycle. To consider the whole machine learning lifecycle, Gypscie is intended to have four components: knowledge base, model lifecycle management, model management, and interfaces.

5.1 Knowledge Base

The knowledge base component comprises two subcomponents: knowledge base management (KBM) and data management (DM), described below.

Data integration and knowledge reasoning features inherent to knowledge bases are relevant assets for model management. First, in model management, one can use a knowledge base (graph) to integrate the myriad of heterogeneous ML artifacts; from models encoded in different tools to datasets from diversified domains. Precisely, the KBM component is in charge of integrating the information spread across miscellaneous artifacts into a semantic representation that encodes, through their relationships and properties, essential knowledge on models generative process and nature. Secondly, reasoning and inference on knowledge bases can promote model selection tasks; for example, the KBM reasoner is intended to suggest the inclusion of relevant models in selection.

The DM component covers the data lifecycle in machine learning applications. This component includes (a) data storage and retrieval; (b) data provenance and versioning; and (c) data preprocessing. Machine learning data may have a prohibitively large volume. Also, ML applications may require very low latency in evaluating prediction workloads. Thus, efficient data storage and retrieval are features Gypscie ought to provide. Furthermore, change underlies ML data in many aspects. For example, the data distribution may change along time, affecting production models performance. Consequently, Gypscie features is going to include dataset provenance and versioning. Finally, Gypscie is also going to take into consideration data preprocessing, a fundamental task for the appropriate performance of ML models [14, 15].

5.2 Model lifecycle management

This component comprises the whole lifecycle of ML models. Even though Gypscie is not a platform for model development, it ought to be aware of model production aspects, including model interoperability as well as data, metadata, and parametrization employed and produced during model training/production.

Due to the lack of interoperability between ML platforms, even comparing models from the same class, but produced in different systems, can be a burdensome task. For example, the very same neural network architecture, if not cautiously implemented, may yield inconsistent prediction results throughout different deep learning frameworks. To establish a ground for model comparison, Gypscie should resort to some promising model exchange formats as ONNX and PFA. The Open Neural Network Exchange Format (ONNX)[10] is an initiative that allows users to easily share models across different deep learning frameworks. In ONNX, models are described by computational dataflow graphs and tensor operators (e.g., matrix multiplication and convolution). The Portable Format for Analytics (PFA) [13] is also an ML model interchange format. Using PFA, users describe learners and models in JSON documents employing a set of control structures, provided library functions, and user-defined functions.

Additionally, this component includes the processes of instrumentation and monitoring of ML models lifecycle. We intend this component to build up on lifecycle management systems as MLflow [24], ModelDB [19], and Clipper [6].

[10] https://onnx.ai/

During model training, users are going to log into Gypscie's knowledge base many artifacts including a collection of metrics on model performance (e.g., accuracy, precision, recall) as well as the hyperparameter configurations and datasets used to train and produce models. In production, users are going to log data and metadata associated with prediction workloads into Gypscie.

5.3 Model management

In this component, we include Gypscie's target functions. Consider that a given collection of competitive models (e.g., for weather forecasting in the same country region) with different performance characteristics are available on the system. From this collection, a given user wants to compare and select the best model based on some criteria. In the above motivation scenario, we envision Gypscie target functions: model quantification and qualification, selection, and composition.

Model quantification and qualification In Gypscie, users are going to employ quantification and qualification functions to retrieve model information for selection tasks. In particular, quantification functions retrieve extrinsic model properties, i.e., measuring information such as training and production predictive performances. On the other hand, qualification functions retrieve intrinsic model properties, i.e., information that defines model nature and structure such as its target (e.g., classification or regression) and its family (e.g., neural networks or boosted trees).

The retrieved information is combined to express arbitrary selection metrics. These metrics may assess models predictive performance (e.g., classification accuracy, mean absolute error), computational performance (e.g., prediction time, memory consumption), soft attributes (e.g., explainability power), comparability properties (e.g., scale, spatiotemporality), and so on.

Model Selection Gypscie is intended to select models according to users arbitrary criteria. These criteria establish thresholds on model assessment and restrict the model selection according to data regions of users interest. For example, the selection criteria may impose to the system (a) to only select models whose classification precision and interpretability degree are greater than the user established thresholds, and (b) to perform the computation of precision and interpretability on a specific data region.

To make it more concrete, suppose a user is interested in ranking and selecting the most accurate models for rain forecasting in Nice, a Southern France town. In this case, Gypscie should only consider models which are predictors for France weather. Those predictors are evaluated by Gypscie according to the user performance criteria expressed in a combination of quantification and qualification functions. Now, suppose the same criteria establish a constraint (e.g., the predictive performance above some threshold) which the available models in Gypscie do not meet. In this case, to better capture Nice's weather, Gypscie would have to produce a new model.

Automated model production during selection may include data augmentation, data distribution estimation, model ensembling, and automated machine learning techniques. For instance, to create ensembles for domain regions where the available models' predictive performance is insufficient, the system may learn new and more specialized models.

5.4 Interfaces

Gypscie's external interfaces are intended to (a) meet different user interests and expertise levels, and (b) provide the appropriate isolation between the expression and accomplishment of a technical task. Therefore, Gypscie is going to provide two interface types, a high-level one based on declarative machine learning (DML) [4] and a more coupled-to-programming one based on APIs usage. Users interested in high-level specifications of ML algorithms and tasks details would use the DML interface. On the other hand, to instrument and log models into the system, users would interact with APIs which are more coupled to model lifecycle routines such as logging training metadata into Gypscie's KB.

Finally, in the lack of ML expertise, users may resort to AutoML. AutoML aims at reducing human interference in ML production. For example, one can employ AutoML techniques in hyperparameter tuning and production of more accurate models [9]. In Gypscie, the employment of AutoML is going to help users more easily producing models for complex domains where the available models have low performance.

6 Conclusions and Future Work

In this paper, we described our vision toward the management of machine learning models. We discussed the challenges in managing this class of models, in particular, their dependence on a set of heterogeneous data artifacts produced throughout their lifecycle. We also introduced a conceptualization for ML model lifecycle which is paramount to Gypscie, our vision toward ML model management. We outlined Gypscie envisioned features and their relevance to model management, especially, to promote model selection tasks.

In future work, we will refine Gypscie architecture and more deeply investigate machine learning model selection. First, extending our conceptual model, we will concisely define Gypscie entities attributes, metadata, relationships, and behaviors. Secondly, we will investigate the definition of an abstract framework for model comparison, selection, and composition applications. Also, we will investigate how the usage of AutoML, ensemble learning, and knowledge reasoning (based on machine learning) can leverage the performance of those applications.

References

1. Abadi, M., et al.: TensorFlow: Large-scale machine learning on heterogeneous systems (2015), software available from tensorflow.org

2. Angermueller, C., Pärnamaa, T., Parts, L., Stegle, O.: Deep learning for computational biology. Molecular Systems Biology **12**(7) (2016)
3. Bhardwaj, A., Deshpande, A., Elmore, A.J., Karger, D., Madden, S., Parameswaran, A., Subramanyam, H., Wu, E., Zhang, R.: Collaborative data analytics with datahub. Proc. VLDB Endow. **8**(12), 1916–1919 (Aug 2015)
4. Boehm, M., Evfimievski, A.V., Pansare, N., Reinwald, B.: Declarative machine learning-a classification of basic properties and types. arXiv preprint arXiv:1605.05826 (2016)
5. Chen, H., Chiang, R.H., Storey, V.C.: Business intelligence and analytics: From big data to big impact. MIS quarterly **36**(4) (2012)
6. Crankshaw, D., Wang, X., Zhou, G., Franklin, M.J., Gonzalez, J.E., Stoica, I.: Clipper: A low-latency online prediction serving system. In: 14th USENIX Symposium on Networked Systems Design and Implementation (NSDI 17). pp. 613–627. USENIX Association, Boston, MA (2017)
7. Hey, T., Tansley, S., Tole, C. (eds.): The Forth Paradigm: Data-Intensive Scientific Discovery. Microsoft Research (2009)
8. Huang, S., Xu, L., Liu, J., Elmore, A.J., Parameswaran, A.: Orpheusdb: bolt-on versioning for relational databases. Proceedings of the VLDB Endowment **10**(10), 1130–1141 (2017)
9. Hutter, F., Kotthoff, L., Vanschoren, J. (eds.): Automatic Machine Learning: Methods, Systems, Challenges. Springer (2018), in press, available at http://automl.org/book.
10. Meng, X., Bradley, J., Yavuz, B., Sparks, E., Venkataraman, S., Liu, D., Freeman, J., Tsai, D., Amde, M., Owen, S., et al.: Mllib: Machine learning in apache spark. The Journal of Machine Learning Research **17**(1), 1235–1241 (2016)
11. Miao, H., Li, A., Davis, L.S., Deshpande, A.: Towards unified data and lifecycle management for deep learning. In: 33rd IEEE International Conference on Data Engineering, ICDE 2017, San Diego, CA, USA, April 19-22, 2017. pp. 571–582 (2017)
12. Pedregosa, F., et al.: Scikit-learn: Machine learning in Python. Journal of Machine Learning Research **12**, 2825–2830 (2011)
13. Pivarski, J., Bennett, C., Grossman, R.L.: Deploying analytics with the portable format for analytics (pfa). In: Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 579–588. KDD '16, ACM, New York, NY, USA (2016)
14. Polyzotis, N., Roy, S., Whang, S.E., Zinkevich, M.: Data lifecycle challenges in production machine learning: A survey. SIGMOD Rec. **47**(2), 17–28 (Dec 2018)
15. Schelter, S., Biessmann, F., Januschowski, T., Salinas, D., Seufert, S., Szarvas, G.: On challenges in machine learning model management. IEEE Data Engineering Bulletin, Special Issue on Machine Learning Life-cycle Management **41**(4) (December 2018)
16. Shalev-Shwatrz, S., Ben-David, S.: Understanding Machine Learning From Theory to Algorithms. Cambridge University Press (2017)
17. Souto, Y.M.: A Spatio-Temporal Approach for Ensemble Learning using CONVLSTM Networks. Ph.D. thesis, National Laboratory of Scientific Computing, Petrópolis, Rio de Janeiro, Brazil (6 2018), in Portuguese
18. Souto, Y.M., Porto, F., de Carvalho Moura, A.M., Bezerra, E.: A spatiotemporal ensemble approach to rainfall forecasting. In: 2018 International Joint Conference on Neural Networks, IJCNN 2018, Rio de Janeiro, Brazil, July 8-13, 2018. pp. 1–8 (2018)

19. Vartak, M., Subramanyam, H., Lee, W.E., Viswanathan, S., Husnoo, S., Madden, S., Zaharia, M.: Modeldb: a system for machine learning model management. In: Proceedings of the Workshop on Human-In-the-Loop Data Analytics. p. 14. ACM (2016)
20. Vartak, M., da Trindade, J.M.F., Madden, S., Zaharia, M.: Mistique: A system to store and query model intermediates for model diagnosis. In: SIGMOD Conference (2018)
21. Venderley, J., Khemani, V., Kim, E.A.: Machine learning out-of-equilibrium phases of matter. Phys. Rev. Lett. **120**, 257204 (Jun 2018)
22. Widmer, G., Kubat, M.: Effective learning in dynamic environments by explicit context tracking. In: Proceedings of the European Conference on Machine Learning. pp. 227–243. ECML '93, Springer-Verlag, London, UK, UK (1993)
23. Wolpert, D.H.: The lack of a priori distinctions between learning algorithms. Neural Comput. **8**(7), 1341–1390 (Oct 1996)
24. Zaharia, M., Chen, A., Davidson, A., Ghodsi, A., Hong, S.A., Konwinski, A., Murching, S., Nykodym, T., Ogilvie, P., Parkhe, M., et al.: Accelerating the machine learning lifecycle with mlflow. Data Engineering p. 39 (2018)