Towards a Unified Model Representation of Machine Learning Knowledge

Antonio Martínez-Rojas[a], Andrés Jiménez-Ramírez[b] and Jose G. Enríquez[c]
Departamento de Lenguajes y Sistemas Informáticos, Universidad de Sevilla, Seville, Spain

Keywords: Machine Learning, Automated Machine Learning, Knowledge Representation, Model-Driven Engineering.

Abstract: Nowadays, Machine Learning (ML) algorithms are being widely applied in virtually all possible scenarios. However, developing a ML project entails the effort of many ML experts who have to select and configure the appropriate algorithm to process the data to learn from, between other things. Since there exist thousands of algorithms, it becomes a time-consuming and challenging task. To this end, recently, AutoML emerged to provide mechanisms to automate parts of this process. However, most of the efforts focus on applying brute force procedures to try different algorithms or configuration and select the one which gives better results. To make a smarter and more efficient selection, a repository of knowledge is necessary. To this end, this paper proposes (1) an approach towards a common language to consolidate the current distributed knowledge sources related the algorithm selection in ML, and (2) a method to join the knowledge gathered through this language in a unified store that can be exploited later on. The preliminary evaluations of this approach allow to create a unified store collecting the knowledge of 13 different sources and to identify a bunch of research lines to conduct.

1 INTRODUCTION

Machine Learning (ML) entails the study of algorithms that automatically improve through experience (Mitchell, 1997). This kind of algorithms has been successfully and broadly applied in the past (Mitchell, 2006) and nowadays is receiving an increasing attention due to the affordable access to bigger computation power of machines.

A ML project requires selecting an appropriate algorithm to process the data to learn from, what is typically named *creating the data model*. However, there are thousands of algorithms under the paradigm of ML, each of them tailored to some specific tasks or contexts. In addition, many of these algorithms offer a different set of parameters to be configured (e.g., selecting the number of layers in a neural network).

Many existing approaches focus on the latter task, i.e., supporting the user after the algorithm selection is done, and few of them recommend an algorithm always after the user has provided the dataset. As an example, the recent research area of AutoML (Thornton et al., 2012) aims to automate the different steps of ML projects. Nonetheless, such approaches neglect the early stages of the project. Many of them just provide a brute force mechanism that runs several algorithms in later stages of the project, i.e., when the dataset is ready. Thus, little effort has been done to support the user in the algorithm selection in an efficient manner (i.e., without applying brute force) and based on the problem characteristics (i.e., the early information).

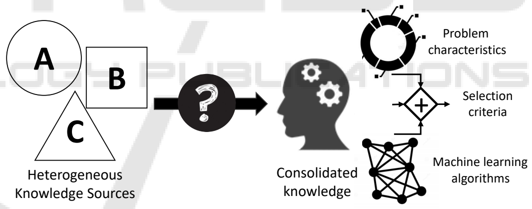

Figure 1: Problem motivation.

The algorithm selection is specifically challenging since the existing knowledge regarding this task is distributed across different sources and each of them is specified in a non-standard manner, thus, making it difficult to consolidate information from different sources, i.e., the name of the algorithms —or family of algorithms—, the selection criteria, and the characteristics of the problem that affect the selection are heterogeneous (cf. Figure 1).

[a] https://orcid.org/0000-0002-2782-9893
[b] https://orcid.org/0000-0001-8657-992X
[c] https://orcid.org/0000-0002-2631-5890

Martínez-Rojas, A., Jiménez-Ramírez, A. and Enríquez, J.
Towards a Unified Model Representation of Machine Learning Knowledge.
DOI: 10.5220/0008559204700476
In *Proceedings of the 15th International Conference on Web Information Systems and Technologies (WEBIST 2019)*, pages 470-476
ISBN: 978-989-758-386-5

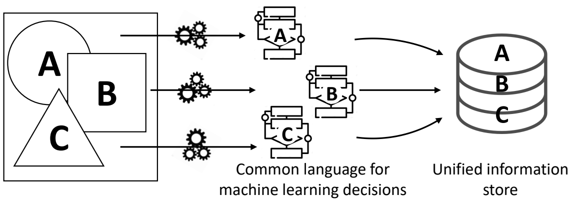

Figure 2: Overall proposal.

To reduce the risk of taking inaccurate decisions due to a lack of information, a central repository of the ML Knowledge which stores the information in a structured way is required. In order to address this problem, this paper proposes (cf. Figure 2), on the one hand, a unified language for representing the knowledge related to the algorithm selection in ML projects. On the other hand, the paper describes a method to transform the knowledge gathered using this language to a unified knowledge store which can be exploited later on.

The unified language is presented as a metamodel that allows a graphical representation of the knowledge related to:

- The characteristics that affect the decisions, e.g., the amount of data which is available or the type of problem.
- The algorithms that can be recommended, e.g., Bayesian network or Support Vector Machine.
- The criteria for recommending an algorithm based on some of the characteristics, e.g., if the project aims to detect anomalies and the number of columns of the dataset is greater than 100, the Support Vector Machine algorithm is an excellent candidate.

Thereafter, the recommendations which are written using this language can be transferred to the proposed unified stored. This store allows to reduce the ambiguity on (1) the name of the characteristics, e.g., number of columns is often used as number of variables or features in some information sources, and (2) the name of the algorithms to recommend, e.g., Bayesian network can be found as Bayes model too.

This approach has been validated in a real industrial project using several publicly-available information sources related to ML. Some of them represent the information as a picture summarizing the knowledge (e.g., (Sckit-learn, 2019; Microsoft, 2019)) while some others are expressed in text form (e.g., (Dataiku, 2019)). After considering an initial set of 13 different information sources, our approach successfully stored 150 recommendations of more than 80 algorithms considering more than 20 characteristics.

Although it remains out of the scope of this paper, this unified knowledge can be used to create a recommendation system that helps the user to decide which is the best algorithm for her new project.

The rest of the paper is organized as follows: section 2 put in context the problem treated in this research, detailing a background related to the Machine Learning and Model-Driven Engineering. Section 3 presents the contributions of the paper. This section is divided into two main elements: (1) a common language for recommendations of ML algorithms and (2) an early approach to reach a unified knowledge store. Finally, section 4 summarizes the conclusions and states a set of future work.

2 BACKGROUND

This paper deals with two main concepts: ML (cf. Section 2.1) and Model-Driven Engineering (cf. Section 2.2).

2.1 Machine Learning

Machine learning emerges as a set of tools under a broader paradigm called artificial intelligence (Mitchell, 1997). A ML project typically follows a life-cycle comprising many activities, e.g., the understanding of the problem, selecting the appropriate algorithm, parametrizing the algorithm, or creating and testing the model. All these steps involve tedious manual work which motivating the arising of AutoML (Thornton et al., 2012; Feurer et al., 2015), a research line that pursues the automation of the ML life-cycle steps.

So far, AutoML has been applied in several domains, like health (Panagopoulou et al., 2018), chemistry (Dixon et al., 2016) or software engineering (Abukwaik et al., 2018). Most of the effort has been applied to automatically generate the ML model, e.g., to look for the algorithm's parameters which allow the most accurate model for a given dataset. However, the majority of these studies applied brute force to look for these parameters and only a few approaches (Panagopoulou et al., 2018; Mohr et al., 2018) include smarter solutions to reduce the search space.

In the industry field, there exist some commercial tools that support AutoML, e.g., BigML (BigML, 2019) or DataRobot (DataRobot, 2019). Similarly to the academic field, these tools mainly apply brute force to find the appropriate algorithm.

At a glance, selecting the appropriate algorithm is a non-deterministic and time-consuming task that depends on many problem and data characteristics. For this, the empirical knowledge is commonly shared in different Internet sources. Beyond the research papers, organizations used to share their experiences with the aim of guiding practitioners to use some software products. For example, scikit-learn (Pedregosa et al., 2011) (cf. Figure 3) and Microsoft Azure (Microsoft, 2019) share a cheat sheet which tries to explain which algorithm better fits according to a set of problem characteristics while Dataiku (Dataiku, 2019) contains technical documents with the same objective. The current work aims to gather all this distributed knowledge to enable a smarter way of AutoML.

2.2 Model-Driven Engineering

The Model-Driven Engineering (MDE) paradigm raises the use of models as a mechanism to reach the concrete from the abstract (Fondement and Silaghi, 2004).

MDE incorporates the elements: concepts, notations, processes, rules and support tools (Brambilla et al., 2012), to provide advantages such as: having a common way of representing processes, facilitating compatibility with other formalisms, enabling the reuse of models or creating specific solutions of domain among others (Mohagheghi et al., 2013).

The fundamental elements of MDE are models and transformations between models (Cetinkaya and Verbraeck, 2011), which must be expressed through some notation (called a modeling language), and defines the syntax or notation of the model, as well as its semantics or meaning.

Everything in MDE can be expressed as a model (Bézivin, 2005). The term "metamodeling" is known as the action of modeling a model or modeling a modeling language. A metamodel is an abstraction of a model itself, which defines the properties of that model, the structure and restrictions for a family of models (Mellor et al., 2004).

MDE is probably one of the best-known modeling techniques in software engineering (Kent, 2002). Modeling languages are the mechanisms that allow designers to specify the models of their processes or systems. They establish the way in which the concrete representation of a conceptual model is defined and can be composed of graphical representations, textual representations, or even both. In any case, modeling languages are formally defined and oblige designers to use their syntax when creating models (Brambilla et al., 2012). There are two major groups of modeling languages.

- Domain-Specific Languages (DSLs), which are designed specifically for a certain domain.
- General-Purpose Modeling Languages (GPMLs), which can be used for any application domain.

The Meta Object Facility (MOF) language (OMG, 2016), proposed by the reference body in this field, the Object Management Group (OMG), is one of the best-known languages for the definition of metamodels. In this language, metainformation is specified that makes data understandable by a computer (Schmidt, 2006).

Considering the background presented, it is possible to assume that MDE can be used to standardize the way in which the ML Knowledge is created.

3 CONTRIBUTION

This section describes the two main contributions of this paper: (1) a common language is proposed to enable a consolidated way of representing the ML Knowledge (cf. Section 3.1), and (2) a process to incorporate heterogeneous information sources into a unified knowledge store (cf. Definition 1) using the previous language (cf. Section 3.2).

Definition 1. *A* ***Unified knowledge store*** $UKS = (CharTerms, AlgTerms, KnowlSources, Rules)$ *consists of*

- *A set of pairs* $\langle char_{id}, char_{name}\rangle$ *which contains an id and a name associated to problem characteristics, e.g., the amount of data (i.e., CharTerms),*
- *A set of pairs* $\langle alg_{id}, alg_{name}\rangle$ *which contains an id and a name associated to the ML algorithms, e.g., Naives Bayes (i.e., AlgTerms),*
- *A set of tuples* $\langle source_{id}, source_{name}\rangle$ *which contains an id and a name associated to the knowledge sources that have been considered in the store, e.g., Sckit-Learn Algorithm Cheat Sheet (i.e.,* $AlgTerms$*),*
- *And a set of tuples* $\langle rule_{id}, source_{id}, antecedents, consequences\rangle$ *which contains an id, the reference of the knowledge source which motivates this rule, and the rule itself (i.e., Rules). On the one hand,* $antecedents$ *is a set of pairs* $\langle char_{id}, value\rangle$ *stating that this rule is fired if the problem characteristics have the given value. On the other hand, consequences is a set* alg_{id} *which indicates that these algorithms are recommended if the rule is fired.*

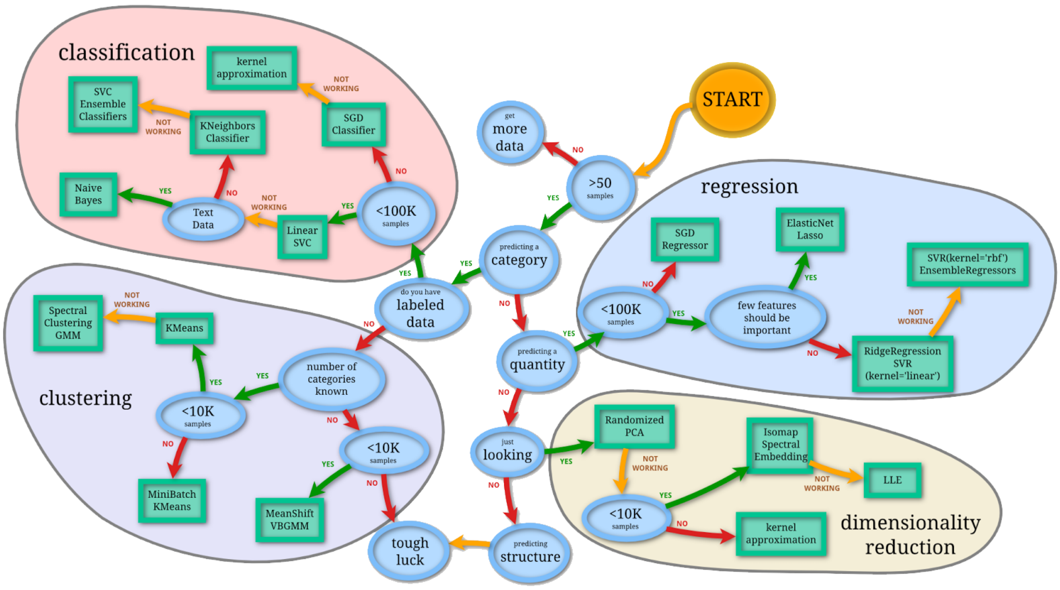

Figure 3: Example knowledge source. *source:scikit-learn.com*.

3.1 A Common Language for Recommendations of ML Algorithms

This paper proposes a formal language to abstract from the different languages which are used to represent the knowledge. More precisely, in the context of recommendations for the usage of a ML algorithm, sources of knowledge can be found in research papers, Web forums, cheat sheets of organizations, etc. Nonetheless, if these sources are analyzed and the non-relevant information is wiped out, the knowledge that they contain shares a similar and simple format: some algorithms are recommended if a set of problem characteristics have some specific values (e.g., the problem is to predict a discrete value and the amount of data is over 10K).

Herein, we propose an abstract syntax or metamodel (cf. Figure 4) that allows: (1) representing this knowledge in a graphical way and (2) being interpreted by a computer program.

The proposed metamodel is composed of six metaclasses. The "*MLKnowledge*" metaclass allows the content of the knowledge source to be represented in the format of the target knowledge source. This representation format is composed of Decisions and Nodes.

The "*Node*" metaclass, defined by the attribute "*name*", allows to represent each origin points of the different branches of the knowledge source. This metaclass can be represented as three different ways:

- **Start:** it represents the initial node.
- **Characteristic:** it represents the antecedents that are considered for making a decision and generate a consequence.
- **Algorithm:** it represents a consequence, that is, an algorithm resulting from the recommendation based on some antecedents).

The "*Decision*" metaclass allows to represent the antecedents of the knowledge source, that is, a set of characteristics that affects the decisions and the criteria for recommending an algorithm based on them. These criteria are represented through the "expression" attribute of "*Decision*" metaclass. Moreover, this metaclass connects instances of the metaclass "*Node*" through the "*source*" and "*target*" attributes.

In addition to the abstract syntax, a concrete syntax that allows to create models based on the ML Knowledge Language was defined. This concrete syntax is a DSL composed of a set of specific symbols (cf. Figure 5) that let the software engineer instantiate each of the metaclasses of the metamodel.

A small example of the use of the DSL is illustrated in Figure 6. This figure represents a piece of the knowledge source of scikit-learn (cf. Figure 3) modelled with the DSL described above.

As evidenced in Figure 6, the model begins with the *Start Node*. Next, it is presented a *Characteristic Node* called "*data*", which represents the amount of data that the user has. This *Characteristic* is connected to a pair of *Nodes* called "*Linear SVC*" and "*SGD Classifier*" by two different *Decisions*, called "*>100K*" and "*<100K*" respectively. It means, if the antecedent "*data*" takes value "*<100K*" the conse-

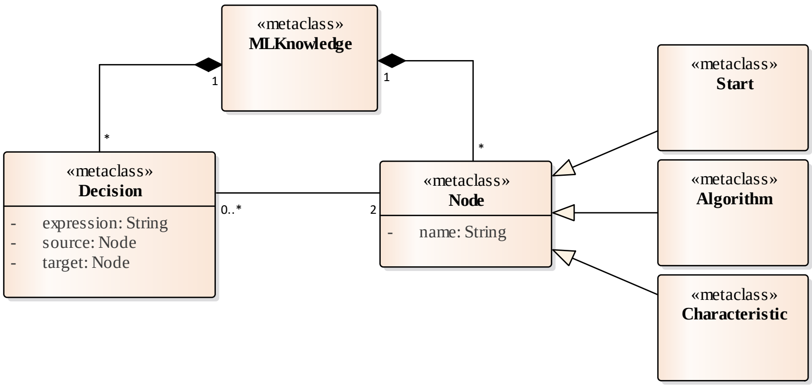

Figure 4: MLKnowledge Metamodel.

Metaclass	Description	Symbol
Start Node	It represents the initial node	Start
Characteristic Node	It represents a feature of the knowledge source	
Algorithm Node	It represents an algorithm	*name*
Decision	It represents the decision based on certain criteria and characteristics	*expression*

Figure 5: Concrete syntax definition.

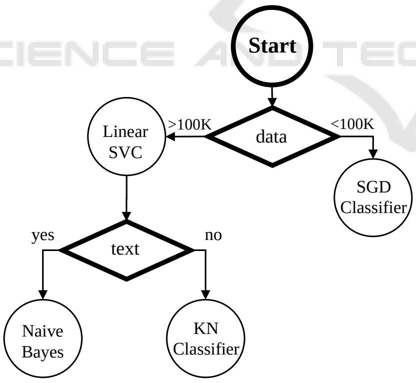

Figure 6: Example of DSL use.

quence value will be "*SGD Classifier*", else, the consequence value will be "*Linear SVC*".

There is another *Characteristic Node* called "*text*" linked to the "*Linear SVC*" *Node*. This *Characteristic Node* represents the type of data that the user has. It means, if the antecedent "*Linear SVC*" takes value "*yes*" the consequence value will be "*Naive Bayes*", else, the consequence value will be "*KNeighbors Classifier*".

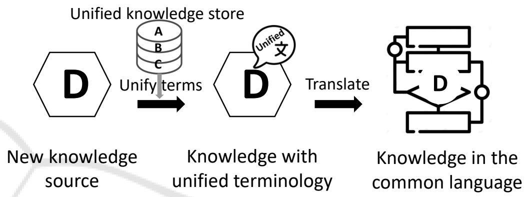

Figure 7: Consolidate terms of knowledge source.

3.2 Towards a Unified Knowledge Store

As seen in Figure 2, this approach consists of two phases. In a first phase, each knowledge source is expressed using the suggested language (cf. Figure 7). For this, the terminology of the considered knowledge source is unified with the terminology that is already used in the existing unified knowledge store. That is, each problem characteristic or algorithm that appears in the knowledge source is mapped to a term in *CharTerms* and *AlgTerms* respectively. In case that some new term has not a mapping to any exiting term, it remains with the original one since it will be included later in the store.

Thereafter, the knowledge source with unified terminology is manually modeled using the previous language. That is, the different relations between the problem characteristics and the algorithms is written in a formal way.

In a second phase, the knowledge source is processed to extract the individual recommendation rules (cf. Figure 8) and store them in the unified knowledge store. For this, since the model present a tree-like structure, it is divided into the different paths that exist from the root (i.e., the start node) to any algorithm node.

Each path is composed of (1) a "*Start Node*, (2) a set of "*Characteristic Nodes* together with a labeled

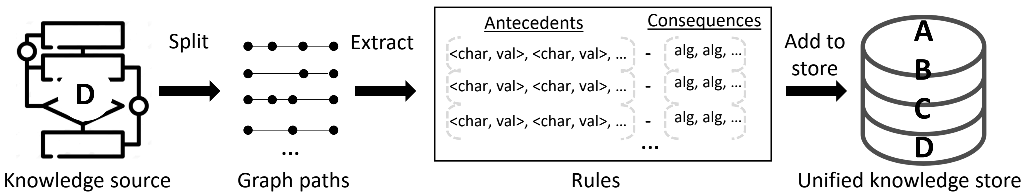

Figure 8: Add knowledge to the unified knowledge store.

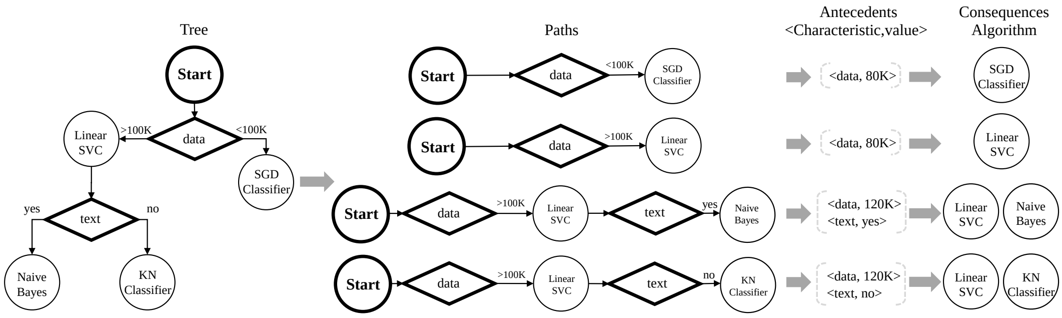

Figure 9: Example for extracting rules from a sample knowledge source.

outgoing edge where the label indicates the value that takes the characteristic, and (3) one or various "*Algorithm Nodes*. Therefore, the paths are processed to extract the rules of the knowledge source. These rules have a similar shape to the unified knowledge store. Each one keeps a set of antecedents (i.e., the names and the values of the characteristics that appears in the path) and a set of consequences (i.e., the names of the algorithms that appears in the path).

As a final step, these intermediate rules are incorporated in the unified knowledge store. First, if the knowledge source does not exist in the *KnowlSources* of the store, a new entry is included with a new $source_{id}$. Second, each characteristic name and algorithm name that exists in the *CharTerms* and the *AlgTerms* are substituted by the $char_{id}$ and the alg_{id} respectively. If some characteristics or algorithms were not previously included in the store, new entries are created in the *CharTerms* or *AlgTerms* and the new $char_{id}$ or alg_{id} are used to substitute the names in the antecedents and consequences. And third, a new $rule_{id}$ is obtained and the tuple $\langle rule_{id}, source_{id}, antecedents, consequences\rangle$ is the *Rules* of the unified knowledge store.

For example, Figure 9 depicts how the different rules can be extracted from the model of Figure 6.

4 CONCLUSIONS AND FUTURE WORK

This paper presents an approach to deal with the distributed knowledge of ML. Specifically, it aims to create a repository with rules that help to decide which ML algorithms are suitable to solve a given problem. For this, a common language for modeling this knowledge is proposed. Such a language is stated in form of a metamodel that a computer program can process. In addition, a procedure to transfer these models to a unified knowledge store is described. This store will enable exploiting the knowledge of the distributed sources to make decisions with less risk.

However, this work considers some assumptions that limit its application. First, it considers that all the knowledge sources are equally relevant and not biased, e.g., an organization may not recommend certain algorithms just because they do not provide a component for it. Second, the suggested unified knowledge store keeps simple rules lacking more complex syntax like *OR* or *NOT* expressions. And finally, using this proposal requires a manual work to model the knowledge source through the provided language which may entail a considerable effort depending on the number of sources. Nonetheless, this effort will be leveraged not only by a single ML project but by the future ones too.

As further future work, we plan (1) to exploit the unified knowledge store in order to generate a decision support tool, (2) to improve traceability of term mappings between knowledge sources and the unified store, so that we can enable a revision of the translations that have been done related to a term in the unified knowledge store. (3) to weight knowledge sources according to their relevance, soundness or reliability, in order to optimize the search for the right algorithm, (4) to extend the proposed

MLKnowledge language capabilities since complex expressions, such as disjunctions and negations, occasionally appear within the antecedents of knowledge sources, (5) to introduce the concept of intensity of recommendation with the aim of expressing the degree of acceptance of the recommendations, since some knowledge sources express a distinction between the value of different recommendations (e.g., excellent vs acceptable recommendations), (6) to analyze the impact of fuzzy terms within knowledge sources since some of them specify fuzzy values for the characteristics (e.g., few data instead of a concrete number), (7) to use ML to automatically translate knowledge sources, so that it takes as input the source as its (e.g., either in graphic or text mode) and generates as output the associated models in the ML Knowledge Language.

ACKNOWLEDGEMENTS

This research has been supported by the Pololas project (TIN2016-76956-C3-2-R) of the Spanish Ministerio de Economía y Competitividad and the PROMETEO project (P009-18/E09) of the Centre for Industrial Technological Development (Centro para el Desarrollo Tecnológico Industrial, CDTI) of Spain.

REFERENCES

Abukwaik, H., Burger, A., Andam, B. K., and Berger, T. (2018). Semi-automated feature traceability with embedded annotations. *2018 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, pages 529–533.

Bézivin, J. (2005). On the unification power of models. *Software & Systems Modeling*, 4(2):171–188.

BigML (2019). Bigml. Available at: https://bigml.com/. Last accessed: July 2019.

Brambilla, M., Cabot, J., and Wimmer, M. (2012). Model-driven software engineering in practice. *Synthesis Lectures on Software Engineering*, 1(1):1–182.

Cetinkaya, D. and Verbraeck, A. (2011). Metamodeling and model transformations in modeling and simulation. In *Proceedings of the Winter Simulation Conference*, pages 3048–3058. Winter Simulation Conference.

Dataiku (2019). Dataiku blog. Available at: https://blog.dataiku.com/. Last accessed: July 2019.

DataRobot (2019). Datarobot. Available at: https://www.datarobot.com/. Last accessed: July 2019.

Dixon, S. L., Duan, J., Smith, E. D., Bargen, C. D. V., Sherman, W., and Repasky, M. P. (2016). Autoqsar: an automated machine learning tool for best-practice quantitative structure-activity relationship modeling. *Future medicinal chemistry*, 8 15:1825–1839.

Feurer, M., Klein, A., Eggensperger, K., Springenberg, J., Blum, M., and Hutter, F. (2015). Efficient and robust automated machine learning. In Cortes, C., Lawrence, N. D., Lee, D. D., Sugiyama, M., and Garnett, R., editors, *Advances in Neural Information Processing Systems 28*, pages 2962–2970. Curran Associates, Inc.

Fondement, F. and Silaghi, R. (2004). Defining model driven engineering processes. In *Third International Workshop in Software Model Engineering (WiSME), held at the 7th International Conference on the Unified Modeling Language (UML)*.

Kent, S. (2002). Model driven engineering. In *International Conference on Integrated Formal Methods*, pages 286–298. Springer.

Mellor, S. J., Scott, K., Uhl, A., and Weise, D. (2004). *MDA distilled: principles of model-driven architecture*. Addison-Wesley Professional.

Microsoft (2019). Machine learning algorithm cheat sheet for azure machine learning studio. Available at: https://docs.microsoft.com/en-us/azure/machine-learning/studio/algorithm-cheat-sheet. Last accessed: July 2019.

Mitchell, T. M. (1997). *Machine Learning*. McGraw-Hill, Inc., New York, NY, USA, 1 edition.

Mitchell, T. M. (2006). *The discipline of machine learning*, volume 9. Carnegie Mellon University, School of Computer Science, Machine Learning ....

Mohagheghi, P., Gilani, W., Stefanescu, A., and Fernandez, M. A. (2013). An empirical study of the state of the practice and acceptance of model-driven engineering in four industrial cases. *Empirical Software Engineering*, 18(1):89–116.

Mohr, F., Wever, M., and Hüllermeier, E. (2018). Ml-plan: Automated machine learning via hierarchical planning. *Machine Learning*, 107(8):1495–1515.

OMG (2016). Meta object facility (MOF) 2.5 core specification. Version 2.5.1. Available at: https://www.omg.org/spec/MOF/2.5.1/PDF. Last accessed: July 2019.

Panagopoulou, M. S., Karaglani, M., Balgkouranidou, I., Biziota, E., Koukaki, T., Karamitrousis, E., Nena, E., Tsamardinos, I., Kolios, G., Lianidou, E. S., Kakolyris, S. S. J., and Chatzaki, E. (2018). Circulating cell-free dna in breast cancer: size profiling, levels, and methylation patterns lead to prognostic and predictive classifiers. *Oncogene*, 38:3387–3401.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., et al. (2011). Scikit-learn: Machine learning in python. *Journal of machine learning research*, 12(Oct):2825–2830.

Schmidt, D. C. (2006). Model-driven engineering. *COMPUTER-IEEE COMPUTER SOCIETY-*, 39(2):25.

Sckit-learn (2019). scikit-learn algorithm cheat-sheet. Available at: https://scikit-learn.org/stable/tutorial/machine_learning_map/index.html. Last accessed: July 2019.

Thornton, C., Hutter, F., Hoos, H. H., and Leyton-Brown, K. (2012). Auto-weka: Automated selection and hyper-parameter optimization of classification algorithms. *CoRR*, abs/1208.3719.