Solution Patterns for Machine Learning

Soroosh Nalchigar[1(✉)], Eric Yu[1], Yazan Obeidi[2], Sebastian Carbajales[2], John Green[2], and Allen Chan[2]

[1] Department of Computer Science, University of Toronto, Toronto, Canada
{soroosh,eric}@cs.toronto.edu
[2] IBM Canada Ltd., Markham, Canada
yazan.obeidi@ibm.com, {sebastia,green,avchan}@ca.ibm.com

Abstract. Despite the hype around machine learning (ML), many organizations are struggling to derive business value from ML capabilities. Design patterns have long been used in software engineering to enhance design effectiveness and to speed up the development process. The contribution of this paper is two-fold. First, it introduces solution patterns as an explicit way of representing generic and well-proven ML designs for commonly-known and recurring business analytics problems. Second, it reports on the feasibility, expressiveness, and usefulness of solution patterns for ML, in collaboration with an industry partner. It provides a prototype architecture for supporting the use of solution patterns in real world scenarios. It presents a proof-of-concept implementation of the architecture and illustrates its feasibility. Findings from the collaboration suggest that solution patterns can have a positive impact on ML design and development efforts.

Keywords: Conceptual modeling · Machine learning · Advanced analytics · Business analytics · Design patterns

1 Introduction

Despite the hype around machine learning (ML), many organizations are struggling to derive business value from ML capabilities [25]. Development of ML solutions have inherent complexities. It requires understanding what ML can and cannot do for organizations [7], specifying a well-defined business case and problem [24], translating and decomposing it into ML problem(s) [28], data preparation and feature selection [14], ML algorithm selection and trade-offs [18], and finding linkages between ML models and business processes [23], among others. Tackling these complexities requires not just a specialized ML skillset and talent (which are hard to obtain and retain) but also executives and stakeholders who know about ML technology and how to use it [12,15].

Design patterns have long been used in software engineering to enhance the design effectiveness and to speed up the development process. By offering collections of well-proven solutions to commonly occurring design problems, design

P. Giorgini and B. Weber (Eds.): CAiSE 2019, LNCS 11483, pp. 627–642, 2019.
https://doi.org/10.1007/978-3-030-21290-2_39

patterns have facilitated software development efforts and streamlined the communication between developers [8].

In earlier work, a conceptual modeling framework for requirement elicitation, design, and development of advanced analytics and ML solutions had been introduced [19–22]. The contribution of this paper is two-fold. *First,* it introduces solution patterns as an explicit way of representing well-proven ML designs for commonly-known and recurring business analytics problems. A solution pattern is comprised of a number of parts, including stakeholders, their decision activities, business questions, ML algorithms, metrics and parameters, contextual information, quality requirements, datasets, and data preparation workflows. It is an artifact, in the form of a conceptual model, that represents generic ML solution designs tailored to particular business contexts or situations. Through several instantiations, the paper illustrates that solution patterns can organize, store, and present knowledge on various aspects of ML solution design, such as:

- What can ML offer, given a business context?
- What type of analytics is applicable to the problem at hand?
- What algorithms belong to that category?
- When to use what algorithm and how to configure different algorithms?
- How to evaluate and compare alternative algorithms?
- What non-functional requirements (NFRs) are critical and relevant?
- How different algorithms are known to influence the NFRs?
- What data is relevant for the problem at hand?
- How to transform and prepare the raw data for different ML algorithms?

By providing reusable answers to these questions, solution patterns tackle a wide range of complexities that one would face in the development of ML solutions.

Second, it reports on the feasibility, expressiveness, and usefulness of solution patterns, in collaboration with an industry partner in the context of business process management. It provides a prototype architecture for using solution patterns in real world scenarios. It presents a proof-of-concept implementation of the architecture and illustrates its feasibility. Moreover, it provides evidence collected during the collaboration, suggesting that solution patterns can have a positive impact on ML design and development efforts.

Organization. Section 2 presents illustrations of solution patterns. Section 3 introduces modeling concepts and their semantic relationships. Section 4 reports on feasibility, expressiveness, and usefulness of solution patterns. Section 5 reviews related work and Sect. 6 concludes the paper.

2 Illustrations

This section illustrates solution patterns for three common business processes, namely: loan approval, fraud detection, and task assignment processes. The content and the knowledge in these models are accumulated from survey papers

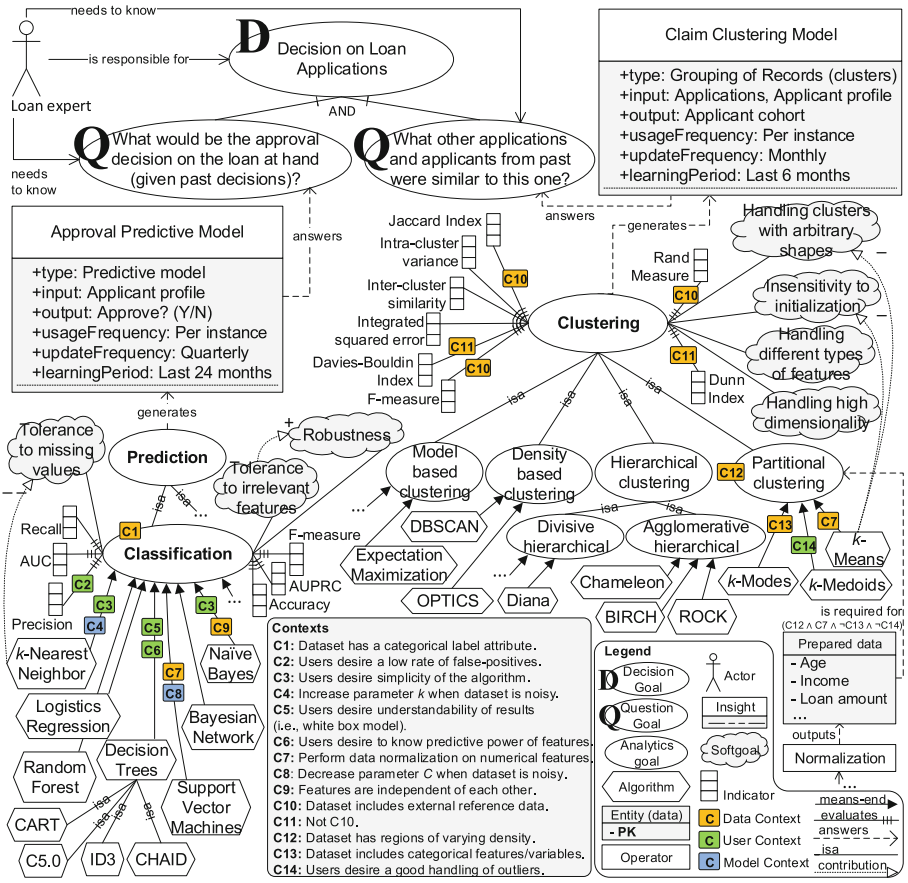


Fig. 1. A portion of loan approval solution pattern. Not all Contexts and Contribution Links are shown due to space limitations.

and textbooks (e.g., [9,13,14,17,29]) supplemented with collective experience of authors from real world advanced analytics projects.

A solution pattern starts with a characterization of the business problem and needs toward which the pattern is targeted. These are represented in terms of *Actors*, *Decision Goals*, *Question Goals*, and *Insights*. Figure 1 shows the pattern for loan approval business process. It shows that a Loan expert, as part of the loan application approval process, is responsible for making the Decision on Loan Applications. Decision Goals are decomposed into one or more Question Goals. What would be the approval decision on the loan at hand (given past decisions)? is an example of a Question Goal. The pattern indicates that in order to make the Decision on Loan Applications, a Loan expert needs to know what will be the (ML-generated) recommendation on a new case, given the past decisions. Question Goals are answered by Insight elements. Figure 1 shows that a Predictive model

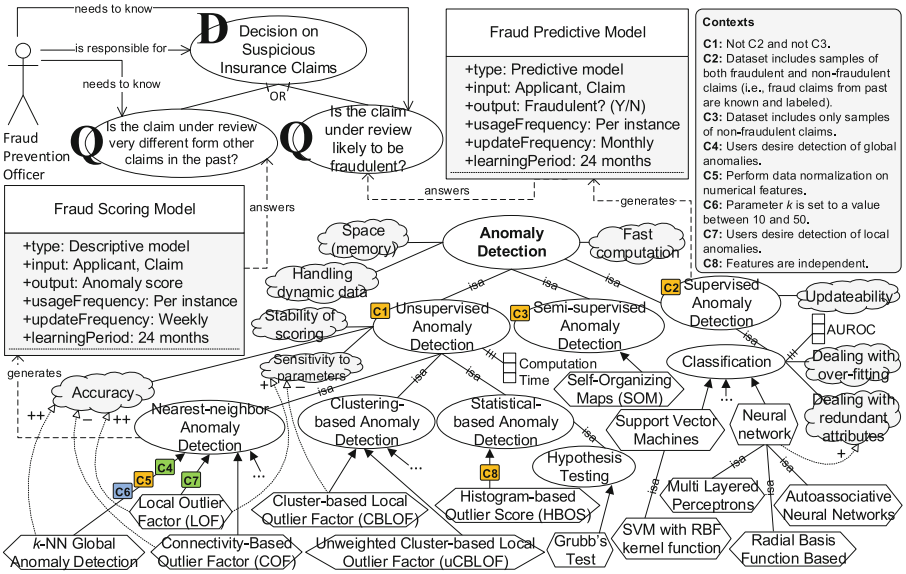


Fig. 2. A portion of fraud detection solution pattern. Refer to Fig. 1 for legend.

that receives Applicant profile as input and generates a binary value of Approve? (Y/N) as output, can answer the Question Goal at hand. By linking actors and decisions to questions and insights, a solution pattern translates a business problem into a (set of) well-defined ML problem(s).

Having defined the ML problem, the pattern then provides solution design(s) for it. It describes what type of analytics is applicable and what algorithms belong to that category. This is represented in terms of *Analytics Goals*, *Algorithms*, and *Means-End* links. Figure 1 shows that in order to generate an Approval Predictive Model, one need to accomplish a Prediction type of Analytics Goal, where Classification goal is a sub-type. The pattern shows that *k*-Nearest Neighbor, Naïve Bayes, and Support Vector Machines are among alternative algorithms for performing Classification.

Every ML algorithm has certain assumptions that limit its applicability to certain contexts. An essential part of a solution pattern provides knowledge on when to use and how to configure different algorithms. These are represented in terms of *User Contexts*, *Data Contexts*, and *Model Contexts*. In Fig. 1, User Context C3 shows that the *k*-Nearest Neighbor algorithm is applicable when Users desire simplicity of the algorithm. On the other hand, Data Context C9 states that the Naïve Bayes algorithm is applicable when Features are independent of each other. In addition, Model Context C8 says that for using the Support vector machines algorithm, one should Decrease parameter *C* when dataset is noisy.

A critical aspect of ML solution design is numerical evaluation and comparison of algorithms. Solution patterns describe what metrics are applicable for the problem at hand and when to use which metric. These are represented in

terms of *Indicators* and *Evaluates* links. The pattern in Fig. 1 shows that Accuracy and Precision are among other metrics that can be used for evaluating the Approval Predictive Model. Data Context C2 states that Precision should be used for evaluation when Users desire a low rate of false-positives.

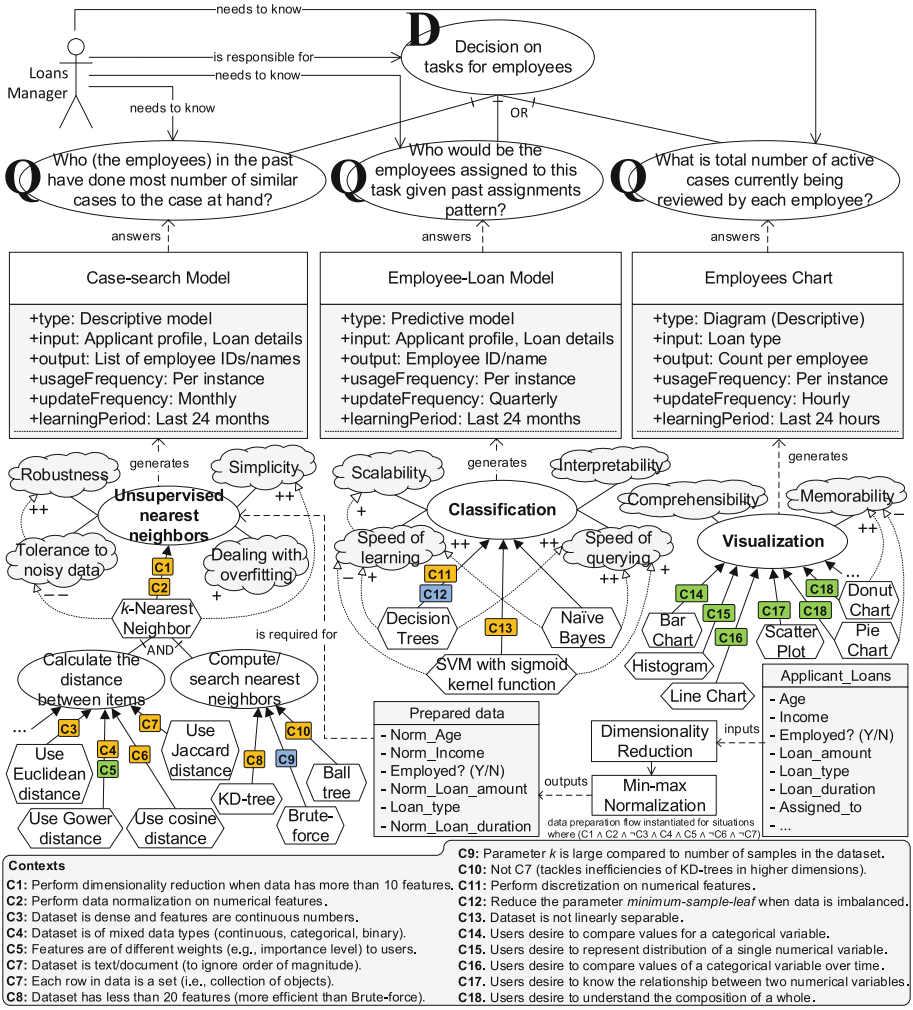


Fig. 3. A portion of task assignment solution pattern. Refer to Fig. 1 for legend.

Designing ML solutions includes taking into account the qualities that are desired by users and trade-offs among them (e.g., interpretability, speed, memory). Towards this, a solution pattern shows what NFRs are relevant to the domain and problem at hand. It also reflects the knowledge on how different algorithms are generally known to perform with respect to those requirements.

These are represented in terms of *Softgoals* and *Contribution* links. Figure 1 shows that Tolerance to missing values is a Softgoal that needs to be considered in performing Classification. It reflects that *k*-Nearest Neighbor would break (−−) this quality.

Data cleansing and preparation have a large impact on ML outcomes and utility. Solution patterns suggest what data is relevant for the problem at hand and how it should be transformed into the right shape for different algorithms. These are represented in terms of *Entities*, *Operators*, and *Data Flows*. Figure 1 shows that a dataset of Age, Income, and Loan amount attributes is required for performing the Partitional clustering. Data Context C7 states that one should Perform data normalization on numerical features when using *k*-Means algorithm.

Figures 2 and 3 show fraud detection and task assignment solution patterns, respectively. The pattern in Fig. 2 shows how different types of Anomaly Detection algorithms can be applied for the problem of fraud detection in insurance claims. This pattern is used later in the paper for implementation and evaluation (See Sect. 4). The pattern in Fig. 3 illustrates a wider range of instances of data preparation elements (e.g., Dimensionality Reduction), decomposition of the *k*-Nearest Neighbor algorithm into finer-grain tasks, along with more instances of contribution links from algorithms to softgoals (e.g., Speed of learning).

3 Metamodel for Solution Patterns

Figure 4 shows the modeling concepts and their semantics relationship in terms of a UML class diagram. The metamodel for solution patterns is comprised of a selected set of elements from our previous works [20,22] complemented with a set of new elements that are necessary for and specific to solution patterns. In this section, first we present a brief summary of selected concepts from our previous work and then define the new elements in more detail.

An *Actor*, defined as an active entity that is capable of independent action [30], *is responsible for* some *Decision Goals*. A Decision Goal describes intention of an Actor towards choosing an option among a set of alternatives. In order to achieve Decision Goals, an Actor *needs to know* the answer to some *Question Goals*. Question Goals symbolize things that an Actor desires to know as part of decision making activity. An *Insight* symbolizes the final outcome of an ML solution (e.g., a *Predictive Model* that is trained and tested) which returns some *output* given some *input* and by doing so, it *answers* a Question Goal. These constructs (from the Business View part of the metamodel [20]) together represent the business problem to which the ML solution is targeted. They represent who (Actor) needs what (Insight), and why (Decision and Question goals).

An *Analytics Goal* symbolizes the high-level intention of extracting Insights from datasets, e.g., *Prediction Goal*. An *Algorithm* is a procedure or task that carries an Analytics Goal. An Algorithm is linked to an Analytics Goal via a *Means-End* link, representing an alternative way of accomplishing that goal. *Indicators* represent ML metrics that evaluate algorithms' performance, while *Softgoals* represent non-functional requirements that are critical in design of

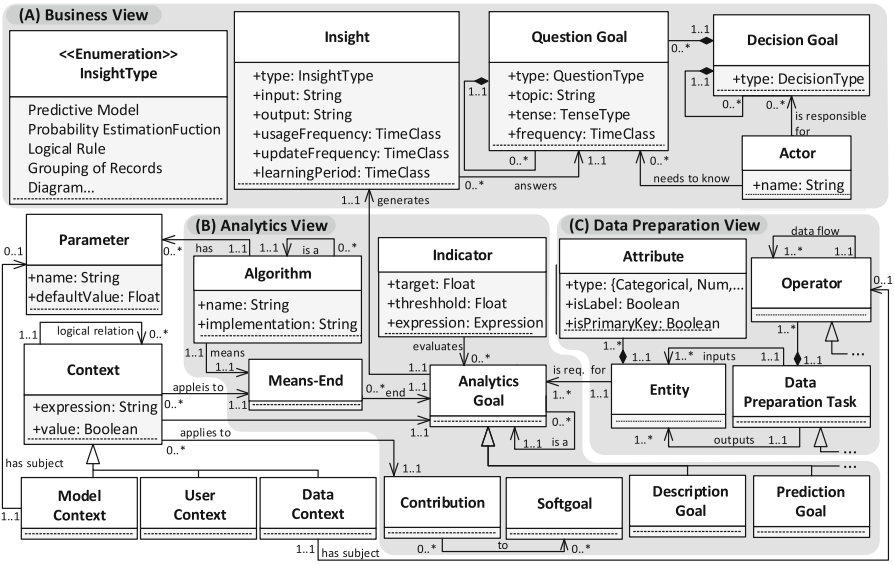


Fig. 4. Metamodel for solution patterns (partial). The gray-shaded areas denote a selected set of elements from our precious works [20,22].

the solution. These constructs (from the Analytics View part of the metamodel [20]) together capture the design of ML solution for the business problem. They represent what ML algorithms are applicable for the problem at hand and how one would compare and select among them.

A *Data Preparation Task* represents some transformation (e.g., *Dimensionality Reduction*) on a dataset. It consists of one or more *Operators* that are connected via *Data Flows*. Formed by a set of *Attributes*, an *Entity*, symbolizes a dataset. These constructs (from the Data Preparation View part of the metamodel [20]) together describe what data preparation workflows are needed to transform the data into the right shape for execution of ML algorithms.

More details and examples of the three views are provided in our previous works [19–22]. The rest of this section focuses on modeling elements that are specific to solution patterns.

Contexts. To define this construct, we use and extend the proposal by Ali et al. [5]. A context is a partial state of the world that is relevant to analytical requirements. It is expressed as a formula of predicates about the domain. Contexts need to be verified. Three kinds of contexts are distinguished here, based on their source of verification. *User Contexts* are contexts that need to be verified based on information from actors (i.e., solution users). *Data Contexts* are contexts that can be verified based on information about the dataset (e.g., size, feature distributions, types). They can also represent certain recommendations on how to transform and prepare dataset for the problem at hand. *Model Contexts*

are contexts that can be verified based on algorithm configurations and parameter values. They can also represent certain recommended configurations to be taken into account in experiments and solution design. This differentiation serves as a way to facilitate representation of knowledge on under what data and user conditions, a solution design (including algorithm choice and data preparation steps) is applicable for the problem at hand and how parameters need to be set. Table 1 shows the structure of context expressions in EBNF formalism.

Table 1. EBNF grammar for context expressions

⟨*user_context*⟩ :- ⟨*User_Predicate*⟩
  | ⟨*user_context*⟩ '`and`' ⟨*user_context*⟩
  | ⟨*user_context*⟩ '`or`' ⟨*user_context*⟩;

⟨*data_context*⟩ :- ⟨*Data_Predicate*⟩
  | '`Perform`' ⟨*operation*⟩ '`when`' ⟨*Data_Predicate*⟩
  | '`Perform`' ⟨*operation*⟩ '`on`' ⟨*data_type*⟩ '`features.`'
  | ⟨*data_context*⟩ '`and`' ⟨*data_context*⟩
  | ⟨*data_context*⟩ '`or`' ⟨*data_context*⟩;

⟨*operation*⟩ :- '`normalization`' | '`dimensionality reduction`' | '`discre...`'

⟨*data_type*⟩ :- '`numerical`' | '`categorical`' | '`datetime`' | '`binary`' | ...;

⟨*model_context*⟩ :- ⟨*Model_Predicate*⟩
  | ⟨*Model_Predicate*⟩ '`when`' ⟨*data_context*⟩
  | ⟨*model_context*⟩ '`and`' ⟨*model_context*⟩
  | ⟨*model_context*⟩ '`or`' ⟨*model_context*⟩;

Contexts may apply to the Means-End, Contribution, and Evaluation links, as well as to the Analytics Goals. A context applied to a Means-End link shows that the corresponding Analytics Goal (i.e., the end) can be achieved by the Algorithm (i.e., the means) only if the context holds. When a context is applied to a Contribution link, it represents situations under which the Algorithm contributes (positively or negatively) to the Softgoal. On the other hand, contexts applied to Evaluation links represent knowledge on when the Indicator is applicable for evaluating the associated Analytics Goal. Lastly, a context that is applied to an Analytics Goal represents the activation rule of the goal towards generating Insights for the business questions at hand. A context can be defined in terms of other contexts via some logical relations. For example, in Fig. 2, $C1$ is defined as $\neg(C2 \wedge C3)$. Multiple contexts applied to a single element is equivalent to a conjunction of those contexts.

Parameters. This element (not shown in the graphical notation) represents configuration values that are required as input by an Algorithm. An Algorithm

can have zero to many Parameters. The meta-attribute *defaultValue* captures information on default or recommended values for parameters. A Parameter can be the subject of some Model Contexts.

4 Implementation and Evaluation

In this section we report on the feasibility, expressiveness, and usefulness of the proposed approach in this paper. In particular, we have attempted to answer the following questions: (Q.i) Can ML solution patterns be implemented and used in real world scenarios?, (Q.ii) Are the modeling concepts adequate for expressing the solution patterns?, and (Q.iii) What are the benefits of using solution patterns when applying machine learning to business problems?

We first devised a prototype architecture for using the patterns in real world scenarios. We implemented the architecture for the fraud detection solution pattern to test its applicability over sample datasets. In the findings section, we report our observations and lessons learned through the course of these steps.

This study was conducted in collaboration with an industry team within a large information technology company. The high level objective of this collaboration was to embed a ML component within a business process management platform. The metamodel, pattern instances, and a prototype architecture were developed by the academic team (first two authors) and then implemented by developers from the industry partner. The industry team had deep expertise in developing and supporting workflow solutions, but was incorporating a ML component for the first time. This allowed us to obtain feedback and collect observations about ML solution patterns in real world scenarios. The feedback was obtained through a questionnaire and follow-up discussions.

4.1 Prototype Architecture

The prototype architecture is composed of nine logical components that together provide semi-automated support for using solution patterns (Fig. 5).

Pattern Repository. This component stores a collection of solution patterns (such as those in Figs. 1, 2 and 3). Following the semantics defined in the metamodel (Fig. 4), patterns can be expressed in standard, machine understandable formats (e.g., XML) so that the content can be queried and retrieved at run-time.

Data Extractor. It collects the raw data file(s) along with metadata information (e.g., variable types, label attribute flags). The component serves the Data Preparator and Data Context Monitor components.

Context Analyzer. This component is responsible for providing the status of context elements. It depends on the Pattern Repository to provide relevant context elements to be investigated. Given a pattern, this component parses the structure and generates a context analysis workflow (depending on structure of the graph). This allows for systematic discovery of alternatives for applying ML

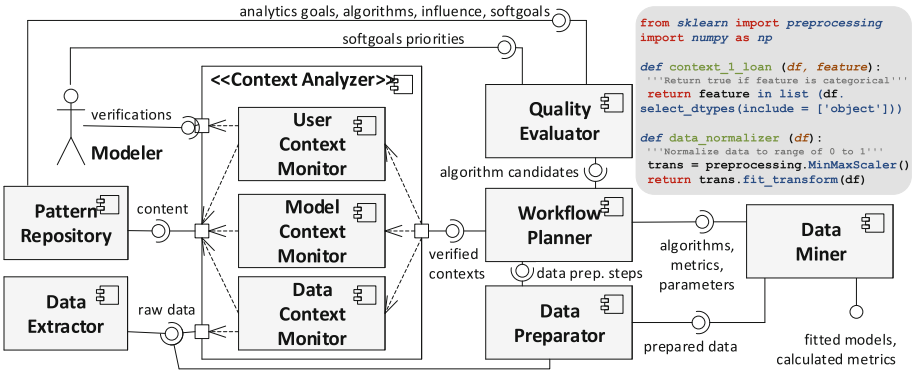


Fig. 5. UML component diagram of prototype architecture (partial). The gray shaded area shows sample codes, in Python, for the Data context monitor and Data preparator components.

algorithms to the problem at hand, depending on which contexts hold. Figure 6 presents two samples of workflows generated from the loan approval and fraud detection patterns. For each context, a binary status value is generated, representing if the contexts holds true or not. The *User Context Monitor* verifies User Contexts, which needs input from the Modeler. The *Data Context Monitor* parses the raw dataset, which comes from the Data Extractor, and analyzes the Data Context elements against data types, feature distributions, and values. The *Model Context Monitor* verifies model configurations and parameter values. Results of these components are provided to the Workflow Planner component.

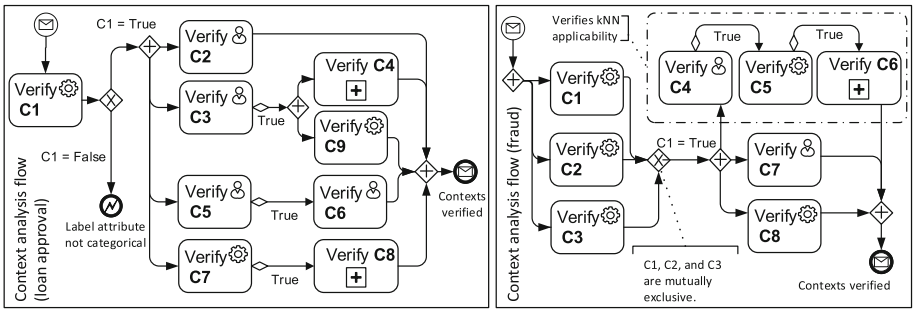


Fig. 6. Examples of context analysis workflows generated by the Context Analyzer from loan approval and fraud detection patterns (partial).

Quality Evaluator. This component is responsible for analyzing the influences of algorithms on (hierarchies of) softgoals. It recommends a list of algorithms to be included in or excluded from the workflow based on their influences on

non-functional requirements. It depends on the Pattern Repository component to retrieve the list of relevant softgoals and also to retrieve the knowledge on how each algorithm is known to perform with respect to those softgoals. It also receives the importance and priority of softgoals from the Modeler.

Workflow Planner. This component is responsible for suggesting analysis scenarios to be performed on the dataset. It interprets the verified contexts along with candidate algorithms and specifies the order of actions to be executed by the Data Preparator and Data Miner components. It also ensures that the necessary data preparation and model configurations (e.g., parameter values) are collected and transmitted for the execution.

Data Preparator. This includes an implementation of a wide range of data preparation tasks and techniques such as data cleansing, noise removal, missing values treatment, data normalization, and data integration. It performs common data preparation and transformation tasks and generates the prepared data table that is ready to be consumed by the Data Miner component. This component depends on the Workflow Planner component to provide the list and order of the necessary preparation steps to be performed on the data. It also received the raw data from the Data Extractor component.

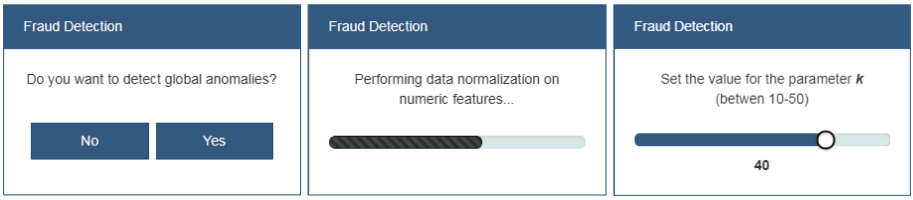
Data Miner. It includes an implementation of a wide range of ML algorithms and is responsible for executing algorithms on an input dataset, storing the fitted models, and reporting on evaluation metrics. This component depends on the Workflow Planner to provide a list of algorithms to be executed and metrics to be calculated. It receives the prepared dataset(s) from the Data Preparator.

4.2 Implementation

We have developed a proof-of-concept implementation of the architecture to test its feasibility and to identify potential logical shortcomings (Q.i). The implementation is written in Python programming language which offers a wide range of libraries for data manipulation (e.g., Pandas[1]), scientific computing (e.g., NumPy[2] and SciPy[3]), and machine learning functions (e.g., Scikit-learn[4]). The graphical user interface was developed in IBM Business Automation Workflow. The focus of implementation was the fraud detection solution pattern (Fig. 2).

A randomly generated dataset of 1,000,000 insurance claims was prepared in two steps. First, a set of claims following a typical distribution was created to represent the non-fraudulent samples. Then, a small number of anomalies were manually inserted into the dataset. This included applicants with unusual claim amounts for certain policies, compared to the rest of population. The dataset included attributes such as applicant's age, claim amount, employment status, policy cost, claim type, among others.

[1] https://pandas.pydata.org/.
[2] https://www.numpy.org/.
[3] https://www.scipy.org/.
[4] https://scikit-learn.org/.



	ProcessName	age	claim_amount	claim_id	customer_id	date	employed	policy_Cost	policy_number	qualified	type	anomaly_score
0	fraud_insurance	45	216	200	4644	2014-01-16	yes	131	99583	196	dental	0.039249
1	fraud_insurance	58	222	201	5482	2014-10-23	yes	224	99504	182	dental	0.026026
2	fraud_insurance	61	229	202	3414	2013-04-26	no	145	99377	205	house	0.023168
3	fraud_insurance	65	444	203	6935	2016-10-20	no	172	99248	219	health	0.810520
4	fraud_insurance	76	275	204	3266	2010-05-14	no	163	99756	182	house	0.043809

Fig. 7. User views from prototype implementation of fraud detection solution pattern.

Figure 7 shows screen shots from the implementation of the above data scenario. In the top-part, the dialog boxes show that design choices were led by the status of various contexts. It shows that the system verifies contexts $C4, C5$, and $C6$ from Fig. 2, following the process model in Fig. 6. At the bottom, it shows a sample of anomaly scores generated for a sample of claims.

4.3 Findings

Through the course of the collaboration with the industry partner, the academic team has collected some feedback and observations about expressiveness and usefulness of solution patterns. The findings suggest that solution patterns can have a positive impact on design and development of ML solutions.

Regarding adequacy of concepts (Q.ii), the academic team found that *the primitive concepts of the metamodel can capture various aspects of ML design knowledge.* The semantics and expressiveness of the three solution patterns were discussed with and reviewed by the industry team during regular meetings. The feedback received from the industry partner was generally positive, reflecting that solution patterns encapsulate a large and critical body of ML design knowledge. An important outcome of the collaboration with industry partner was some enhancements to the metamodel, such as classification of the Context element into the Data, User, and Model Context sub-types. Before this collaboration, there was only a general context element, with no grammar for expressions.

Regarding usefulness of the approach (Q.iii), some feedback was reported by the industry team as follows: (a) Solution patterns *offer an efficient way of transferring ML design knowledge among developers.* In particular, the industry team was able to gain a useful amount of ML design knowledge by reviewing the content of patterns. This included the content of data contexts (e.g., how to prepare the dataset?), user contexts (e.g., what do I need to discuss with end-users?), model contexts (e.g., what should be the range of parameters?), algorithms (e.g., what algorithms are relevant for my problem?) and their influences on softgoals (e.g., how would each algorithm influence quality requirements?). By using the

fraud detection pattern, the project team quickly realized that depending on the data availability, some anomaly detection approaches might be more suitable than others. In particular, if both fraud and non-fraud samples are available, a supervised anomaly detection approach should be used; but if only examples of non-fraud claims were present, a semi-supervised approach is appropriate.

(b) Solution patterns can *potentially reduce time and cost of ML development efforts.* When tasked with the problem of fraud detection for insurance claims, rather than starting out from scratch, solution patterns enabled the project team to start with algorithms and techniques that have been shown to work for the problem at hand. This reduced the exploration and experimentation efforts at the early phases of the project. Also, the patterns simplified the coding phase by pointing developers to existing libraries and implementations of ML algorithms.

(c) The patterns offer *a way to constrain the solution space into a narrower scope based on qualities that are deemed critical by the modeler and end-users.* For example, in the fraud detection pattern, if dealing with redundant attributes is critical, the pattern recommends neural network techniques as a means for satisfying that requirement (See Fig. 2).

(d) Visual representations of patterns was *a convenient and easy to understand encapsulation of ML design.* The development team, initially not familiar with goal-oriented modeling languages, was able to comprehend and consume the produced patterns after a quick tutorial on the notation.

As a result of prototype implementation, certain improvements and extensions to the proposal were deemed necessary. During implementation, the team encountered situations where there is a conflict between softgoals and the contexts that system has verified. Certain mechanisms need to be developed for handling such situations. From an extensibility perspective, there needs to be specific guidelines and procedures on how one can create new and extend existing patterns. The current graphical format of presenting a pattern can be extended with a structured template or domain-specific language that allows for effective retrieval, connection to existing ML libraries, and linking patterns to each other.

There are several threats to validity of the findings and results in this work. The content of patterns presented in this work are mainly drawn from survey papers and textbooks, supplemented with authors' practical experience. Such content would require further validations and extensions by ML experts with practical experiences. Moreover, a formal and structured approach could be developed and used for developing the patterns. The current study was conducted in collaboration with a single industry team. Stronger evidences towards adequacy of concepts (i.e., Q.ii) could be obtained by involving a group of ML experts for evaluating the approach and sufficiency of modeling elements. In addition, stronger evidences about benefits of solution patterns (i.e., Q.iii) could be collected by comparing the approach against ML development without solution patterns (e.g., relying on current professional practice or literature searches). Quantitative research methods could be leveraged to measure the time and costs saved as a result of using solution patterns.

5 Related Work

Related to our work are those that provide best practices, catalogues and patterns for analytics and ML solution development. Chen et al. [10,11] provide a reference architecture and technology catalogue for big data system development. Sculley et al. [26] provide a set of anti-patterns to be avoided while designing ML systems. Zinkevich [31] provides a set of best practices for engineering ML solutions. Breck et al. [6] offers a range of test cases for ensuring reliability of ML systems. The approach in this paper is different in that it offers an explicit and systematic way of representing business requirements and linking them to relevant ML algorithms, while capturing user-, data-, and model-contexts.

A range of machine learning services on cloud platforms are offered by various providers such Microsoft (Azure ML Studio [3]), Google (Cloud AI [4]), and Amazon (SageMaker [1]). They offer (semi) automated tool support for data preparation, model training, testing, and deployment tasks. These platforms come with documentation, guidelines, and community support (e.g., [2]). The work in this paper is different in that business questions and decisions play a critical role in deriving solution design. Furthermore, quality requirements (Softgoals), user preferences (User Contexts), data characteristics (Data Contexts), and parameter configurations (Model Contexts) influence the solution design.

A number of data mining formal ontologies have been developed (e.g., [27]), some with the goal of offering intelligent assistance to domain users during the analytics process (e.g., [16]). Differently, the approach in this paper starts from business decisions and questions and link them to alternative ML algorithms and data preparation techniques, while considering quality requirements.

The choice of the term solution patterns is intended to convey the differences between the approach in this paper and existing software design patterns (e.g., [8]). Patterns in this paper are more problem-domain specific and are less-generic compared to design patterns. They are (conceptually) closer to a working solution and hence for adopters to use them as starting point of implementation.

6 Conclusions

This paper introduced solution patterns as an explicit and systematic way of representing well-proven ML designs for business problems. It illustrated that, by representing ML design knowledge in a reusable form, patterns can address a wide range of complexities that one would face in designing ML solutions. Based on an industry collaboration, the paper reported on the feasibility, expressiveness, and usefulness of the patterns. Further empirical studies are under way to evaluate the benefits and limitations of the overall framework as well as the solution patterns work presented in this paper. This includes involving a group of ML experts for evaluating adequacy of concepts, validating and expanding the content of the patterns, assessing benefits of the approach against a baseline, and experiments and case studies with real-world datasets.

References

1. Amazon SageMaker. http://aws.amazon.com/sagemaker/. Accessed 11 Mar 2018
2. Azure AI Gallery. http://gallery.azure.ai/. Accessed 11 Oct 2018
3. Azure Machine Learning Studio. http://azure.microsoft.com/en-us/services/machine-learning-studio/. Accessed 11 Mar 2018
4. Google Cloud AI products. http://cloud.google.com/products/ai/. Accessed 11 Mar 2018
5. Ali, R., Dalpiaz, F., Giorgini, P.: A goal-based framework for contextual requirements modeling and analysis. Requirements Eng. **15**(4), 439–458 (2010)
6. Breck, E., Cai, S., Nielsen, E., Salib, M., Sculley, D.: The ML test score: a rubric for ML production readiness and technical debt reduction. In: 2017 IEEE International Conference on Big Data, pp. 1123–1132. IEEE (2017)
7. Brynjolfsson, E., McAfee, A.: The business of artificial intelligence: what it can –and cannot– do for your organization. Harv. Bus. Rev. **7**, 3–11 (2017)
8. Buschmann, F., Henney, K., Schimdt, D.: Pattern-Oriented Software Architecture, vol. 5. Wiley, Hoboken (2007)
9. Chandola, V., Banerjee, A., Kumar, V.: Anomaly detection: a survey. ACM Comput. Surv. **41**(3), 15 (2009)
10. Chen, H.-M., Kazman, R., Haziyev, S.: Agile big data analytics for web-based systems: an architecture-centric approach. IEEE Trans. Big Data **2**, 234–248 (2016)
11. Chen, H.-M., Kazman, R., Haziyev, S., Hrytsay, O.: Big data system development: an embedded case study with a global outsourcing firm. In: Proceedings of the First International Workshop on BIG Data Software Engineering, pp. 44–50. IEEE Press (2015)
12. Davenport, T.H., Ronanki, R.: Artificial intelligence for the real world. Harv. Bus. Rev. **96**(1), 108–116 (2018)
13. Goldstein, M., Uchida, S.: A comparative evaluation of unsupervised anomaly detection algorithms for multivariate data. PloS ONE **11**(4), e0152173 (2016)
14. Han, J., Pei, J., Kamber, M.: Data Mining: Concepts and Techniques. Elsevier, Amsterdam (2011)
15. Henke, N., et al.: The Age of Analytics: Competing in a Data-Driven World, vol. 4. McKinsey Global Institute, New York (2016)
16. Keet, C.M., et al.: The data mining optimization ontology. Web Seman. Sci. Serv. Agents World Wide Web **32**, 43–53 (2015)
17. Kotsiantis, S.B.: Supervised machine learning: a review of classification techniques. Informatica **31**, 249–268 (2007)
18. Luca, M., Kleinberg, J., Mullainathan, S.: Algorithms need managers, too. Harv. Bus. Rev. **94**(1), 20 (2016)
19. Nalchigar, S., Yu, E.: Conceptual modeling for business analytics: a framework and potential benefits. In: 19th IEEE Conference on Business Informatics, pp. 369–378 (2017)
20. Nalchigar, S., Yu, E.: Business-driven data analytics: a conceptual modeling framework. Data Knowl. Eng. **117**, 359–372 (2018)
21. Nalchigar, S., Yu, E.: Designing business analytics solutions: a model-driven approach. Bus. Inf. Syst. Eng. (2018)
22. Nalchigar, S., Yu, E., Ramani, R.: A conceptual modeling framework for business analytics. In: Comyn-Wattiau, I., Tanaka, K., Song, I.-Y., Yamamoto, S., Saeki, M. (eds.) ER 2016. LNCS, vol. 9974, pp. 35–49. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46397-1_3

23. Ng, A.: What artificial intelligence can and can't do right now. Harv. Bus. Rev. **9** (2016)
24. Ransbotham, S., Gerbert, P., Reeves, M., Kiron, D., Spira, M.: Artificial intelligence in business gets real. MIT Sloan Manag. Rev. (2018)
25. Schreck, B., Kanter, M., Veeramachaneni, K., Vohra, S., Prasad, R.: Getting value from machine learning isn't about fancier algorithms – it's about making it easier to use. Harv. Bus. Rev. (2018)
26. Sculley, D., et al.: Machine learning: the high interest credit card of technical debt. In: SE4ML: Software Engineering for Machine Learning (2014)
27. Vanschoren, J., Soldatova, L.: Exposé: an ontology for data mining experiments. In: International Workshop on Third Generation Data Mining: Towards Service-Oriented Knowledge Discovery, pp. 31–46 (2010)
28. Veeramachaneni, K.: Why you're not getting value from your data science. Harv. Bus. Rev. **12**, 1–4 (2016)
29. Xu, R., Wunsch, D.: Survey of clustering algorithms. IEEE Trans. Neural Netw. **16**(3), 645–678 (2005)
30. Yu, E.: Modelling strategic relationships for process reengineering. Soc. Model. Requirements Eng. **11**, 2011 (2011)
31. Zinkevich, M.: Rules of machine learning: best practices for ML engineering (2017)