Bus Inf Syst Eng 65(2):179–199 (2023)
https://doi.org/10.1007/s12599-023-00790-2

RESEARCH PAPER

AI-Enhanced Hybrid Decision Management

Dominik Bork · Syed Juned Ali · Georgi Milenov Dinev

Received: 27 August 2021 / Accepted: 5 December 2022 / Published online: 10 February 2023

Abstract The Decision Model and Notation (DMN) modeling language allows the precise specification of business decisions and business rules. DMN is readily understandable by business users involved in decision management. However, as the models get complex, the cognitive abilities of humans threaten manual maintainability and comprehensibility. Proper design of the decision logic thus requires comprehensive automated analysis of e.g., all possible cases the decision shall cover; correlations between inputs and outputs; and the importance of inputs for deriving the output. In the paper, the authors explore the mutual benefits of combining human-driven DMN decision modeling with the computational power of Artificial Intelligence for DMN model analysis and improved comprehension. The authors propose a model-driven approach that uses DMN models to generate Machine Learning (ML) training data and show, how the trained ML models can inform human decision modelers by means of superimposing the feature importance within the original DMN models. An evaluation with multiple real DMN models from an insurance company evaluates the feasibility and the utility of the approach.

Keywords Enterprise modeling · DMN · Model-driven engineering · Artificial intelligence · Machine learning · Explainable AI

Accepted after 3 revision by Jelena Zdravkovic.

D. Bork (✉) · S. J. Ali · G. M. Dinev
Business Informatics Group, TU Wien, Vienna, Austria
e-mail: dominik.bork@tuwien.ac.at

1 Introduction

The Decision Model and Notation (DMN) is a relatively new standard for modeling business decisions. The first version of DMN was released in 2015, and, since then, DMN has continuously evolved toward a meaningful addition to the Business Process Model and Notation (BPMN). Using DMN in addition to BPMN enables separating the decision logic like deciding on a loan, which often entails nested smaller sub-decisions, from the temporal logic of the process flow. DMN is not only a visual means for defining decision logic and decision requirements, DMN models can be also executed by supporting DMN tools to calculate a decision output based on the modeled decision logic and provided input values.

Using DMN expects the business expert to explicitly define the business rules required to cover all possible cases that shall (and shall not) be covered by the DMN model. Consequently, DMN is very powerful in cases where the complexity of the decision logic is rather low, i.e., comprehensible by human beings (cf. Hasić and Vanthienen 2019). DMN aims to be readable and adjustable for business and IT people alike (OMG 2020). An additional issue of the human-centered decision logic design using DMN concerns the dependency of the quality of the DMN models solely on the expertise of the human. The DMN models thus might have an unconscious or a conscious bias of the modeler.

A contrast to DMN-based decision-making is the use of Artificial Intelligence (AI). AI is now supplementing (or even replacing) human judgment in more and more areas. Machine Learning (ML)-based approaches rely on vast amounts of data that can be used to learn the implicit rules underlying recorded business decisions. Such ML-based approaches are thus favorable in cases, where the rules are either not explicitly known in advance or the rules complexity hampers human comprehension and maintainability. Noteworthy, these approaches also have limitations and introduce challenges for businesses aiming to adopt them (Ransbotham et al. 2016), amongst the most severe are: *collection of data*, *consistency of data*, and *choosing algorithms* (Buxmann 2021).

Using AI for business decision making requires from businesses to be able to *explain* their automated decision-making (cf. the GDPR regulation). Pure AI-driven approaches often don't meet this regulation. For example, imagine if two people with similar financial backgrounds apply for bank loans. Even though they are similar in many ways, one person gets the loan and the other person does not. What factors did the bank's AI use to reject the one customer? How were those factors weighted? The designers of the AI "trained" it with vast amounts of consumer data, but at the end, it may be that nobody knows exactly how the AI made the decision (cf. *Explainable AI* or XAI Mueller et al. 2019). XAI research aims to *"find ways to explain the [AI] system to the decision maker so that they know that their decisions are going to be reasonable"* (Mueller et al. 2019, p. 5). Recently, Lukyanenko et al. further differentiated *Model-agnostic Basic XAI* (Lukyanenko et al. 2021) where *"explanations are being generated post hoc, without altering a typically high-performance AI model"* and feed back to a conceptual model e.g., for purposes of superimposition.

In this paper we explore the mutual benefits of DMN and AI. We show ways to facilitate the reciprocal strengths to mitigate the individual shortcomings stressed at the outset. Concretely, we aim to show, (1) how model-driven ML training data collection from DMN models can address the challenges of data collection, data quality, and XAI (Buxmann 2021; Mueller et al. 2019; Sheng et al. 2008; 2) how AI can improve the understanding of business experts on their DMN decision logic to address comprehensibility issues of DMN (Hasić and Vanthienen 2019; Hasić et al. 2020) by means of superimposition (Lukyanenko et al. 2019, 2021); and (3) how the hybrid use of DMN and AI can be evaluated, enabling business experts to transparently choose the best ML algorithms (Buxmann 2021).

The objective of this research is thus to show possibilities of how domain knowledge specified by conceptual DMN models can be used to generate, in a model-driven, repeatable and transparent manner, consistent and valid ML training data. At the same time, we augment the domain models by superimposing the trained ML model accuracy, thereby providing domain experts (in most cases business people) with insights on the decision logic. Research proposed metrics like RMSE, MAE and R2-score (Makridakis et al. 2008) for pure ML approaches. The final objective of this research is to propose a specific procedure and metric for evaluating the hybrid use of DMN and ML.

This research fits to the vision for the enterprise modeling field to emerge that includes *"Different kinds of model content, formats and purposes can be extracted, integrated and federated on demand, either through human intervention or driven by a symbiosis of humans and intelligent agents* (Sandkuhl et al. 2018, p. 72). This paper also fits within the DMN research agendas proposed recently by Figl et al. (2018) and Kluza et al. (2019) by making contributions to the topics *verification and validation* of DMN rules, *table simplification*, and particularly *code generation/tool support*. Taking a broader scope, this research is well positioned within the current research agenda of combining conceptual modeling with emerging technologies and AI (Bork 2022; Etinger et al. 2019; Bucchiarone et al. 2021; Fettke 2020; Lukyanenko et al. 2020; Mussbacher et al. 2020; Wand and Weber 2017).

The contributions of this research are novel and extend the body of knowledge in many ways: This is, to the best of our knowledge, the first proposal of *AI-enhanced decision management*, an approach that uses DMN models to generate ML training data and, from the opposite perspective, uses ML to improve decision logic comprehensibility. We further use a real case study from the insurance sector to empirically evaluate the feasibility and accuracy of using DMN and ML in a hybrid manner.

In the remainder of this paper, Sect. 2 introduces the necessary background. Related works are then presented in Sect. 3 and contrasted to our own approach, which will be introduced thereafter in Sect. 4. We evaluate our approach in Sect. 5 by using multiple real DMN models of an international insurance company. Lessons learned and threats to validity are discussed in Sect. 6 before we conclude this paper in Sect. 7 with perspectives for future research.

2 Background

2.1 Decision Model and Notation

The Decision Model and Notation (DMN) (OMG 2020; Wiemuth et al. 2017) is a relatively new standard maintained by the Object Management Group (OMG) that complements the Business Process Model and Notation (BPMN) standard with a dedicated modeling language to visually represent business decision logic. DMN models not only enable programmatic evaluation of decisions by computer systems, they also aim for efficient visual comprehension by business users. DMN allows modeling of business decisions and the required decision knowledge (i.e., inputs) in a hierarchical structure. DMN distinguishes two levels: the *Decision Requirement Graph (DRG)* and the *Decision Logic (DL)*. The DRG specifies the requirements for evaluating a decision covering a hierarchical structure of sub-decisions, a number of inputs which can be, e.g. inputs from data sources or users, or results from other decisions, and other sources of business knowledge. The Decision Logic level describes business rules or an algebraic means of deriving an output from a set of inputs. DMN decision tables contain a series of contiguous input and output expressions and indicate which decision rule is evaluated based on the input data.

DMN models are primarily composed of *Decisions* (rectangles), *Business Knowledge Models* (rectangles with cut corners), and *Input Data* (rectangles with rounded corners). Figure 1 shows an illustrative example of the application of DMN for deciding on a credit taken from (OMG 2020). Here, a top-level credit routing decision is hierarchically divided into two sub-decisions whose output forms an input of the top-level decision. Each decision receives the Application as an input data element. Furthermore, each decision individually has a business knowledge model as an input, e.g., for the business knowledge on credit eligibility as shown in the corresponding decision table in Fig. 1b.

2.2 Model-Driven Software Engineering

Conceptual modeling languages are composed of (Brambilla et al. 2017): an *abstract syntax*, commonly referred to and specified by means of a *metamodel* (Bork and Fill 2014; Bork et al. 2020b) (i.e., the concepts that can be used to create valid models), a *concrete syntax*, commonly referred to as a *notation* (i.e., the graphical and/or textual representation of the metamodel concepts), and *semantics* (i.e., the meaning attached to the metamodel concepts). Conceptual modeling has its historic roots as a means of applying abstraction to reduce the complexity of a given domain for a specific purpose. Due to this abstraction, conceptual models highlight relevant aspects for means of understanding and communication by human beings (Mylopoulos 1992).

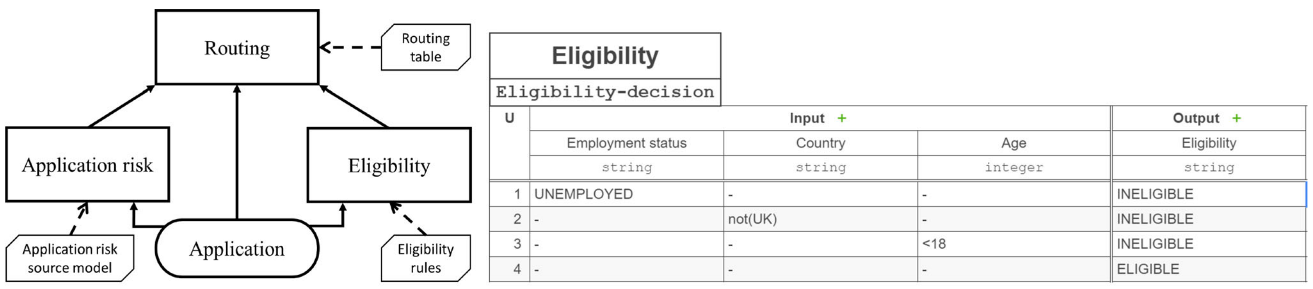
The research domain of Model-Driven Software Engineering (MDSE) focuses on enhancing conceptual modeling to increase efficiency and effectiveness in software artifacts development, both quantitatively and qualitatively. The development process in MDSE, as well as in the more generic field of model-driven development, is driven by conceptual models and model transformations (i.e., manipulation operations on models). The importance of this field relies on the growing complexity of software artifacts at different levels of abstraction (Brambilla et al. 2012; Bucchiarone et al. 2021).

2.3 Supervised Machine Learning

Machine Learning (ML) is a branch of Artificial Intelligence (AI) focused on building applications that learn from data and improve their accuracy over time. In ML, algorithms 'train' ML models to be used for finding patterns and features in vast amounts of data that help to make decisions and predictions when confronted with new data. Quality of the trained ML models is measured by their prediction accuracy (IBM Cloud Education 2021). ML methods can be broadly categorized into *supervised*, *unsupervised*, and *reinforcement* based on the type of data and how the models are trained. As this paper exclusively applies supervised ML algorithms, only these foundations will be introduced.

Supervised machine learning means training an ML model with a labeled data set. The labels provide information that the ML model is being trained to predict for similar data. Supervised ML algorithms can be further classified into *Regression* and *Classification* algorithms. The former include types where the output variables are defined as real numbers. The format for this problem often follows a linear structure. The latter categorize all the variables that form the output into a set of defined classes.

In this paper, supervised ML algorithms of the following categories are relevant:



Eligibility

Eligibility-decision

U	Input			Output
	Employment status	Country	Age	Eligibility
	string	string	integer	string
1	UNEMPLOYED	-	-	INELIGIBLE
2	-	not(UK)	-	INELIGIBLE
3	-	-	<18	INELIGIBLE
4	-	-	-	ELIGIBLE

(a) Decision Requirements Graph (DGR) (b) Decision Logic (DL) of the Eligibility decision

Fig. 1 Example of a DMN DRG and DL (OMG 2020)

- Regression models: Logistic Regression and Linear Regression are examples of regression models to predict the value of a dependent variable based on the value of independent variables. Linear Regression can be used in the case of a continuous dependent variable whereas Logistic Regression can be used in the case of a binary dependent variable. In case of difficult classification of a dependent variable, Support Vector Machines can be used.
- Decision Trees: Decision Trees use classified data to make recommendations based on a set of decision rules. The ensemble versions of Decision Trees like RandomForest, XGBoost, CatBoost "*construct a set of classifiers and then classify new data points by taking a (weighted) vote of their predictions.*" (Dietterich 2000, p. 1).

Supervised machine learning requires less training data than other machine learning methods and makes training easier because the model results can be compared to actual labeled results. However, adequately labeled data is expensive to prepare, and there is the danger of overfitting.

2.4 Conceptual Modeling and AI

The advancements in AI have paved its way toward applications in numerous domains. Recently, an increasing focus on the combination of AI and conceptual modeling can be recognized. This trend is also manifest in the newly established workshops on Conceptual Modeling meets AI (CMAI) (Reimer et al. 2020) and MDE Intelligence (Burgueño et al. 2019) co-located with the flagship conference on conceptual modeling (ER) and model-based software and systems engineering (MoDELS), respectively. Works like (Bork et al. 2020a) apply AI, in particular genetic algorithms and heuristic search, to automatically partition overarching data models into smaller modules. In the opposite direction, first works show possibilities of improving AI-based systems by using conceptual modeling (Lukyanenko et al. 2019).

Recently, Maass et al. (2021) proposed a framework aiming to define the relationships between mental models, conceptual models, and ML models (see Fig. 2). The authors stress the two questions: (1) How can conceptual modeling support the design and development of ML solutions?; and (2) How can ML support the development and evaluation of conceptual models? In our work, we instantiate this framework and thereby respond to these two questions by using conceptual models to *create* ML models by *instantiating* the DMN models to produce training data, and, in the opposite direction by using ML models to *understand* the conceptual models (in our case the decision logic specified in DMN). Thus, the idea is to "*augment a purely data-driven approach to ML with a knowledge-driven one* (Heaven 2019)" and to use the real world domain knowledge represented by conceptual models (Castellanos et al. 2021).

2.4.1 Superimposition

One concrete line of research at the intersection of conceptual modeling and AI is subsumed by the term *Superimposition* that was introduced by Lukyanenko et al. (2019) and further developed in (Lukyanenko et al. 2020; Castellanos et al. 2021; Maass et al. 2022). Superimposition is "*a global, model-agnostic method which enmeshes conceptual modeling diagrams with decision weights available as outputs from common machine learning models.*" (Lukyanenko et al. 2020) (See Fig. 3). Superimposition "*maps the output of machine learning models (i.e., the features, rules and transformation functions) onto a conceptual model of the domain.*" (Maass et al. 2022).

The benefit of Superimposition is that it can facilitate explainability, i.e., it fosters domain understanding of business users by superimposing the learned feature importance by the ML model to the domain model (Lukyanenko et al. 2019). In the thus far proposed Superimposition approaches, the domain models serves only the purpose or representing and communicating the ML feature importance to the business users. In contrast, our approach further uses the domain model (in our case DMN) to generate the ML training data and for representing the results of the training. Consequently, we enable a richer integration of conceptual modeling and AI aiming at "*increasing the interpretability of ML by showing the relative importance of a feature with respect to the target variable.*" (Hall and Gill 2019)

3 Related Works

In the following, existing works and tools that relate to our three objectives, i.e., model-aware training data generation, model-based Explainable AI by means of superimposition, and hybrid use of DMN and AI are presented. Eventually, we close with a summary and a motivation for our approach.

3.1 Model-Aware Training Data Generation

Labelled data availability for ML training is the major bottleneck to apply ML (Zhang et al. 2019; Kababji and Srikantha 2020). Data curation is a time consuming process and can require a lot of effort. One workaround is the use of synthetic data generators to protect the privacy and confidentiality of the actual data (Phua et al. 2010; Yao

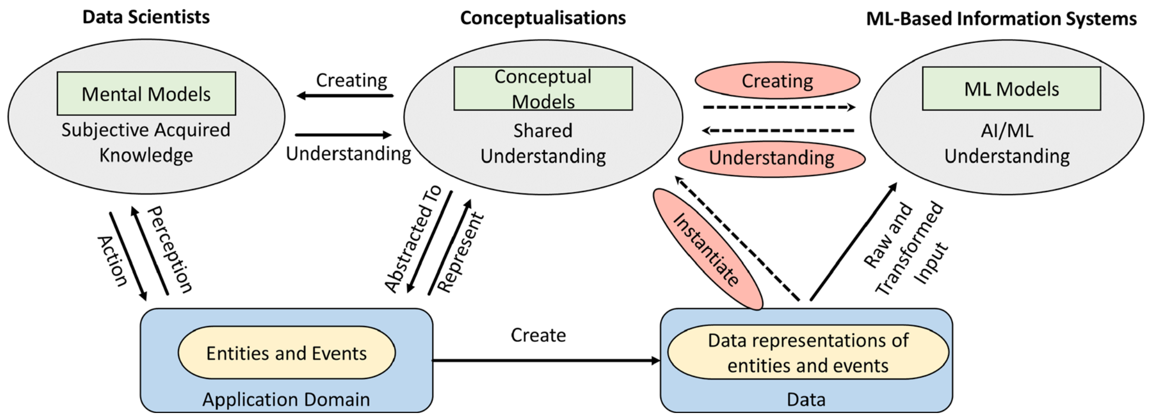
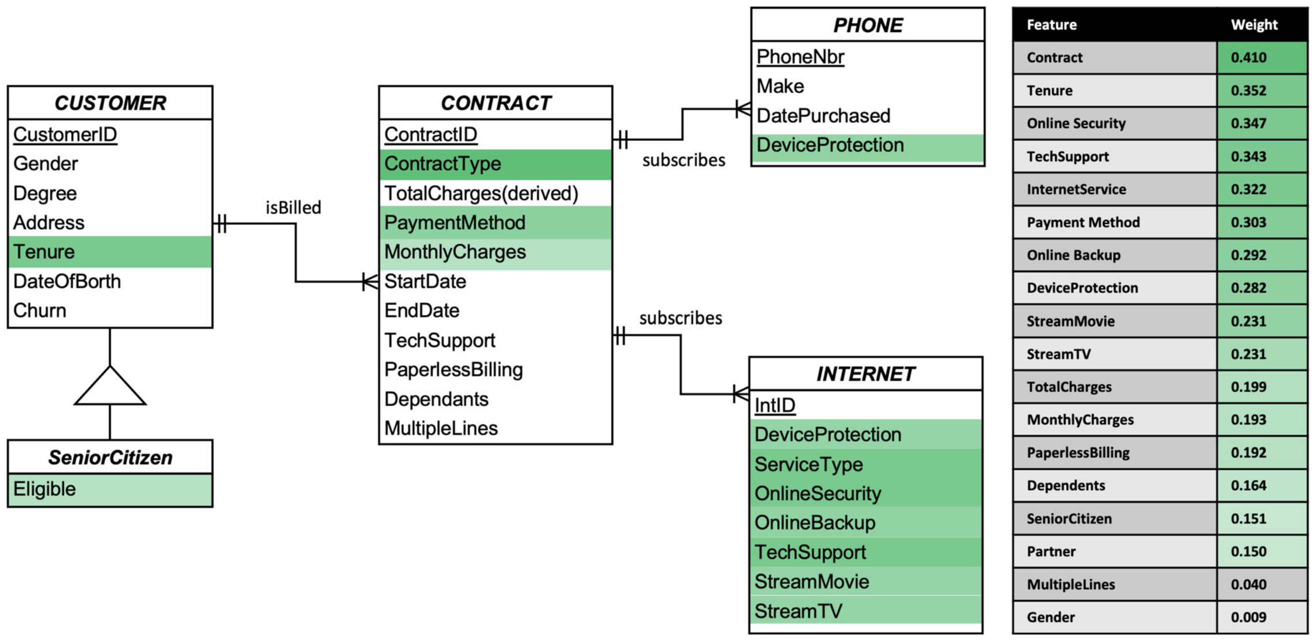


Fig. 2 From mental models to machine learning models – adapted from (Maass et al. 2021)



Feature	Weight
Contract	0.410
Tenure	0.352
Online Security	0.347
TechSupport	0.343
InternetService	0.322
Payment Method	0.303
Online Backup	0.292
DeviceProtection	0.282
StreamMovie	0.231
StreamTV	0.231
TotalCharges	0.199
MonthlyCharges	0.193
PaperlessBilling	0.192
Dependents	0.164
SeniorCitizen	0.151
Partner	0.150
MultipleLines	0.040
Gender	0.009

Fig. 3 Superimpostion example with ML-based ER feature importance (Lukyanenko et al. 2020)

et al. 2015), since it does not hold any personal information and cannot be traced back by any individual. In the following, we report on the few existing works that use conceptual models to inform, support, or even generate ML training data.

Boonmepipit and Suwannasart (2019) propose an approach to generate test cases from traversing BPMN process models linked to DMN models. Existing test cases and models of BPMN and DMN are inputs of their approach which focuses on ensuring that business processes are implemented correctly by generating all valid test cases (i.e., all paths of a process model constrained by DMN decision logic).

Tsymbal et al. (2007) propose an approach that aims to improve the accuracy of ML algorithms by integrating the knowledge provided by ontologies. The authors propose to use a feature ontology as a means to provide relevant domain knowledge that supports the ML algorithms to cope with heterogeneity and independent ML training data sources.

Lukyanenko et al. (2021) propose a method that comprises six guidelines of how domain models represented as Extended Entity Relationship diagrams can be used to improve the accuracy of ML algorithms. The guidelines support the feature generation process and first empirical experiments yielded improved accuracy when following the method instead of conventionally training ML models.

3.2 Conceptual Models for Explainable AI

In the following, we report works that use conceptual models to increase explainability of AI algorithms. More specifically, we report on existing works on *model-agnostic explainable AI* (Lukyanenko et al. 2021) for cases in which "*the explanations are being generated post hoc*".

Etinger et al. (2019) propose an algorithm that takes a decision tree model produced by a decision tree classifier as an input and automatically generates a DMN decision model that represents the decision logic of the decision tree using DMN.

As introduced in Sect. 2.4.1, Superimposition is a recent and compelling research direction for model-agnostic explainable AI. Lukyanenko et al. (2019) introduce an approach that color-codes conceptual models for means of representing the importance of model parts for ML training (e.g., as input data or as an output). In a follow-up work, Lukyanenko et al. (2020) superimpose Extended Entity Relationship (EER) diagrams as a means to visually encode ML aspects to enable explainability (see Fig. 3). Their approach assumes an existing EER domain model and superimpose different aspects within that model. These aspects can be the *feature importance*, *meaningful entities*, *missing values*, and *irrelevant attributes*. Consequently, they link the independent ML model to the conceptual model. From the relevant publications it seems that tool support has not been realized, yet. The authors close with a *"call on research to extend the method in response to the need to improve XAI"* (Lukyanenko et al. 2020, p. 32).

A related stream of research is using Semantic Web techniques for explaining ML models. Burkart and Huber (2021) stress that ontologies *"can improve the explainability of any given [ML] model by incorporating knowledge either before the model training or after the explanation generation to further improve them."* (Burkart and Huber 2021, p. 290) Consequently, the authors show, how ontologies can be used to improve the quality of the ML training data by providing ex-ante data consistency checks (Tsymbal et al. 2007), and how ontologies can improve explainability by e.g., summarizing features or establishing semantic links between ML features (Xu et al. 2015).

3.3 Combinations of DMN and AI

Existing works at the intersection of DMN and AI are now categorized and represented along three streams. One stream uses AI for the *generation* of DMN models, another focuses the *analysis* of DMN models, and a final stream, mostly driven from tool vendors, aims at incorporating ML into the DMN DRG to derive the hybrid use of DMN and AI.

Recently, Goossens et al. (2021) and Etikala et al. (2020) proposed approaches that use Deep Learning and Natural Language Processing (NLP) techniques to automatically analyze natural language texts and create DMN DRGs and Decision Logic. Simić et al. (2019) propose a framework for generating DMN models based on complexity-reducing techniques. The authors introduce an ML ensemble method that is essentially compatible with DMN and is thus human-understandable. Their approach uses existing decision logs to train ML models. The Rule-Learner (Open Rules Inc 2021) tool creates decision models with the help of ML. Driven by decision instances, the tool applies two ML algorithms to produce the decision rules. Functionality for testing and analysis, and the deployment of the rules as decision services are provided.

The increasing use of DMN decision tables to capture critical business knowledge raises the need to support analysis tasks on these tables, such as correctness and completeness checking. Calvanese et al. (2016) provide a formal semantics for DMN tables, a formal definition of critical analysis tasks, and scalable algorithms to tackle two such tasks, i.e., detection of overlapping and of missing rules. The authors further propose an approach to refactor decision tables based on geometric interpretations. Hasić and Vanthienen (2019) gather insights from the process modeling and software engineering fields to propose an initial set of complexity metrics for DMN decision models. The set of metrics has been further developed into DMN modeling strategies presented in Hasić and Vanthienen (2020) that aim at decreasing the decision logic complexity by metrics-based analysis.

To the best of our knowledge, no scientific literature yet covers the hybrid use of ML models and DMN decision models. Only initial results by means of case studies on the integration of PMML files in DMN models can be found from DMN tool vendors. PMML is an abbreviation for the Predictive Model Markup Language which is maintained by the Data Mining Group (2020). PMML *"is an XML-based language and has become the de-facto standard to represent [...] predictive and descriptive models."* (Guazzelli et al. 2009, p. 60) The KNIME Analytics Platform (2021) enables visual modeling of ML workflows and executing them on data sources to train and evaluate different ML models. Moreover, KNIME enables the export of ML models in PMML format. Trisotech DMN Modeler (Trisotech 2020) and Redhat Decision Manager (nA 2021) allow the execution of PMML files representing ML models as decision logic at a decision in the DMN model. In all these cases, the traditional approach of training a ML model with data is followed and the resulting PMML specification is then used in the context of DMN.

3.4 Summary

The related works show some scarcity at the intersection of DMN and AI. Most works focus on either DMN modeling and the analysis/refactoring of existing DMN rules or on using data-driven approaches to generate DMN models by transforming ML models into DMN decision tables. Interestingly, tool vendors are the first to report on case studies of how to incorporate ML models in the DMN-based decision management. The following research gaps, linked to our contributions are remaining:

- DMN for Machine Learning: No approach exists that uses declarative knowledge of DMN models to generate the training data for ML models.
  ⇒ We propose the first DMN-driven approach for generating ML training data. Moreover, we provide tool-support on an open source basis.
- Machine Learning for DMN: No approach exists thus far that uses ML to increase comprehensibility of the DMN decision logic.
  ⇒ We propose a concept for DMN Superimposition and a prototypical tool implementation to automatically generate the superimposed DMN model representation.
- Hybrid Use of DMN and Machine Learning: So far, support for business users in setting up and comprehensively evaluating the hybrid use of DMN and ML is lacking.
  ⇒ We propose an evaluation metric and a procedure that aims at helping business users in differentiating and evaluating hybrid cases.

4 Combining Decision Management and Artificial Intelligence

We now introduce DMN&ML, our approach to combine DMN-based decision management with AI. DMN&ML encompasses the three contributions mentioned at the outset and is accompanied by tool support that shifts some control from the developer of ML solutions to the business analyst. Figure 4 shows the workflow with numbered tasks $\mathbf{T}_i$ (i: 1 to 10) for applying DMN&ML using a BPMN model. The swimlanes separate the process activities into areas dedicated to the business analyst, the data generation (DG) module, and the ML module. Tasks that are automated by our tool are modeled as script tasks with a little script icon on the top left corner (e.g., *Train ML models*) whereas manual tasks, performed by the business analyst, show a person on the upper left side (e.g., *Create DMN Model*).

The DMN&ML workflow starts with the $\mathbf{T}_1$ where a business analyst creates a DMN Decision Requirements Graph. In $\mathbf{T}_2$, she creates decision tables to define the decision logic. After defining the DMN model in Camunda, the analyst creates a JSON file to describe the metadata of the DMN model in $\mathbf{T}_3$. The metadata file stores the possible range of values for the DMN input data variables. After that, the business analyst can trigger the data generation module that takes the DMN model and metadata file and in $\mathbf{T}_4$, transforms the model into a graph required for data generation. In $\mathbf{T}_5$, graph traversal simulates the orchestration of the DMN model and generates datasets for each decision output (Sect. 4.1). $\mathbf{T}_6$ sets the hyperparameters for ML model training and the training takes place in $\mathbf{T}_7$. Once the training is concluded, the accuracy of the different ML models is evaluated in $\mathbf{T}_8$ (Sect. 4.2). Then, in $\mathbf{T}_9$, the best ML model is selected to generate the feature importance values for all the dataset inputs. Finally, the feature importance is superimposed to the original DMN model (Sect. 4.3) in the final $\mathbf{T}_{10}$. Then, the business analyst can investigate the superimposition results to better understand the decision logic.

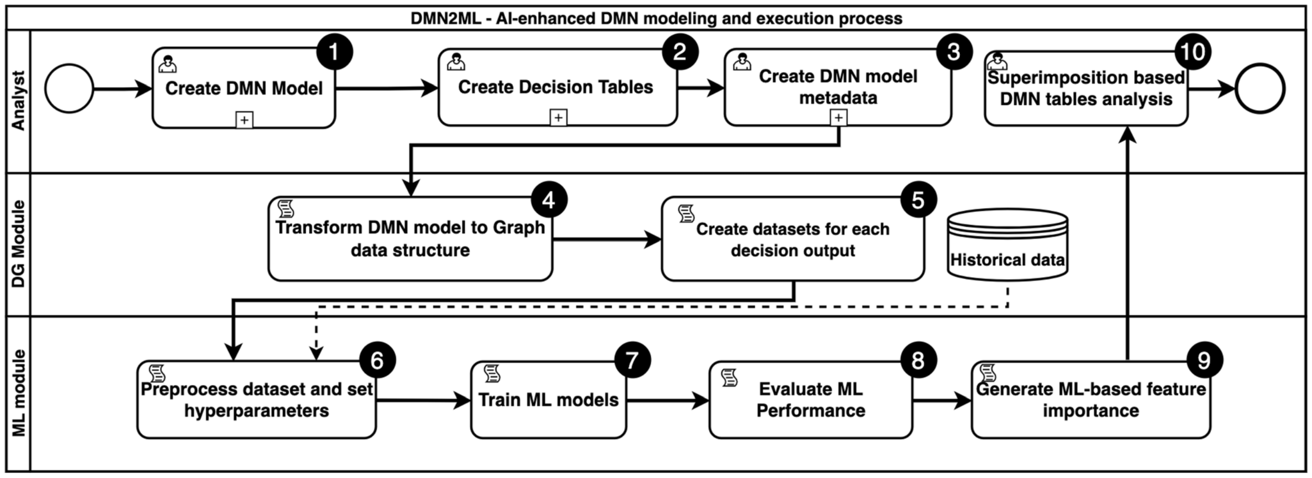

DMN2ML - AI-enhanced DMN modeling and execution process
Analyst
DG Module
ML module
1 Create DMN Model
2 Create Decision Tables
3 Create DMN model metadata
10 Superimposition based DMN tables analysis
4 Transform DMN model to Graph data structure
5 Create datasets for each decision output
Historical data
6 Preprocess dataset and set hyperparameters
7 Train ML models
8 Evaluate ML Performance
9 Generate ML-based feature importance

Fig. 4 Workflow of the AI-enhanced hybrid DMN approach (DMN&ML)

4.1 DMN for ML

Following the DMN for ML direction, our approach uses the DMN models to generate ML training data, thereby facilitating the domain-specificity and user-friendly means of DMN (cf. Lukyanenko et al. 2019). It aims to address some of the data quality and consistency challenges currently limiting ML adoption (Buxmann 2021; Sheng et al. 2008; Lukyanenko et al. 2021). By putting the business analyst at the start of the approach, her business and domain expertise is incorporated *"to increase understanding of available training data for ML tasks and support data preparation activities for ML tasks."* (Lukyanenko et al. 2019, p. 175). Moreover, research shows that incorporating *"domain knowledge [...] can improve the classification performance of the model."* (Burkart and Huber 2021, p. 292)

In the DMN&ML approach, the business analyst uses the openly available Camunda modeling tool (Camunda 2021) to create the Decision Requirements Graph and the decision logic ($\mathbf{T}_1$ and $\mathbf{T}_2$). Once the modeling is finished, the model in XML format, together with a metadata file ($\mathbf{T}_3$) where the domain experts can define the range of input values (e.g., min..max ranges for integer inputs), are then used by the DMN&ML tool to proceed with the script tasks ($\mathbf{T}_4$ to $\mathbf{T}_9$) of the workflow (cf. Fig. 4).

4.1.1 ML Training Data Generation

The ML module uses the exported DMN model and generates datasets for each output variable in each decision of the DMN model. Each dataset comprises the output variable and the inputs required to evaluate the output variable as dataset columns. To generate datasets, the idea is to randomly select an input value from a set of possible values for each DMN input and use these selected values to simulate an orchestration of the DMN model. Then, for each decision, if any of the decision rules are satisfied on the selected input values, i.e., the input values satisfy the rule input condition, then the outputs of the satisfying decision rules are evaluated. Finally, the output values and the input values form a single row in the dataset of that decision. For an example, see the DMN decision rules in Fig. 5. If the selected input values satisfy any of the rules, then the input values together with *House Price* value form a row in the dataset of *House Price*. The approach for dataset generation is presented in Algorithms 1, 2, and 3.

Algorithm 1: Create all datasets for a DMN model

```
Input: dmnModelFile, N
Output: decision outputs datasets
1  function createAllDatasets(dmnModelFile, N)
2      dmnModel ← generateDMNModelObjectFromXML()
3      decisionsEvaluated ← set()
4      for i in seq(1..N) do
5          inputPoints ← generateRandomCombinationFromDMNInputs(dmnModel.inputs)
6          decisionQueue ← dmnModel.leafDecisions
7          while !decisionQueue.empty() do
8              decision ← decisionQueue.pop(0)
9              evalResult ← evaluateDecisionOnInputs(decision, inputPoints)
10             if evalResult then
11                 decisionDatasets[decision] ← addRowToDecisionDataset(decision, inputPoints)
12                 for outgoingDecision in outgoingDecisions do
13                     if all(outgoingDecision.requirements in decisionsEvaluated) then
14                         decisionQueue.insert(outgoingDecision)
15     return decisionDatasets
```

First, to simulate the orchestration, we transform the DMN model that facilitates easier graph traversal and data access in $\mathbf{T}_4$ (see line-2 in Algorithm 1). Therefore, we transform the DMN model to an object (following the OOPS) of a class comprising decisions, inputs, and rules to represent the DMN model's structures and methods required for accessing data in the structures.

After the DMN model to object transformation, we initiate the dataset generation ($\mathbf{T}_5$). The value N in Algorithm 1 denotes the number of times the simulation needs to be performed. Then, *getRandomInputValueFromPossibleValues* in Algorithm 2 is called to generate the initial input values of the DMN model required for the simulation. The input value for each DMN input is picked (pseudo-)randomly using the python module random from a set of all the possible input values of that DMN input. The value is chosen randomly from a set of values for string-based inputs. For integer and floating point-based inputs, the value is chosen from the input range set in the metadata file. Note that higher values of N normalize the number of times each input value is selected. Our approach to randomly select input values for input variables from a set of possible values mitigates any human bias towards specific values during dataset generation. In this way, our approach provides a completely DMN rules and, therefore, model-driven data generation.

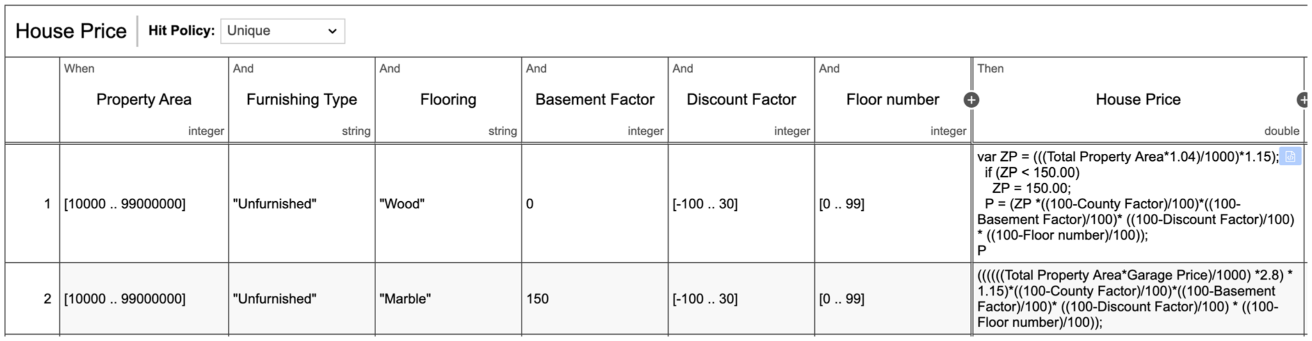
Once the input values are generated, Algorithm 1 initiates the orchestration simulation. The orchestration involves traversing the model as a directed graph and checking each decision for satisfying rules over the generated input values (see Algorithm 3 for rule search). The traversal starts from leaf decisions, i.e., decisions that do not need the output of a sub-decision and only need DMN inputs as the decision inputs. Sub-decisions need to be evaluated before higher-level decisions can be evaluated. Therefore, we use a breadth-first search traversal that evaluates all the decisions at a lower level and then proceeds to higher-level decisions. A decision queue stores the decisions yet to be evaluated on the input values. Initially, all the leaf decisions are added to the decision queue. During the traversal of a particular decision, if the input values satisfy a rule of the decision table, then the decision is considered to be evaluated and added to the list of evaluated decisions. Then, for each outgoing decision connected to the evaluated decision, if all its required sub-decisions are already evaluated then it is added to the queue.

Algorithm 2: Generate a random combination of input data values

```
Input: DMN model input variables with their possible values
Output: Value set of input variables
1 function generateRandomCombinationFromDMNInputs($modelInputs$)
2     $inputValues \leftarrow \{\}$
3     for each modelInput in modelInputs do
4         $inputValue \leftarrow$ getRandomInputValueFromPossibleValues($modelInput$)
5         $inputValues[modelInput] \leftarrow inputValue$
6     return $inputValues$
```

Algorithm 3: Decision evaluation on input values data

```
Input: Decision to evaluate; Values for decision inputs
Output: Decision evaluation result
1  function evaluateDecisionOnInputs($decision, inputValues$)
2      $inputValues \leftarrow \{\}$
3      for each modelInput in modelInputs do
4          $rules \leftarrow decision.rules$
5          for each rule in rules do
6              if all inputValues satisfy their ruleInputExpressions then
7                  for for each output in rule.outputs do
8                      $outputValue \leftarrow$ evaluateOutputExpression($output.Expression, inputValues$)
9                      $inputValues[output] \leftarrow outputValue$
10                 return True
11     return False
```


House Price | Hit Policy: Unique

	When	And	And	And	And	And	Then
	Property Area	Furnishing Type	Flooring	Basement Factor	Discount Factor	Floor number	House Price
	integer	string	string	integer	integer	integer	double
1	[10000 .. 99000000]	"Unfurnished"	"Wood"	0	[-100 .. 30]	[0 .. 99]	var ZP = (((Total Property Area*1.04)/1000)*1.15); if (ZP < 150.00) ZP = 150.00; P = (ZP *((100-County Factor)/100)*((100-Basement Factor)/100)* ((100-Discount Factor)/100) * ((100-Floor number)/100)); P
2	[10000 .. 99000000]	"Unfurnished"	"Marble"	150	[-100 .. 30]	[0 .. 99]	((((((Total Property Area*Garage Price)/1000) *2.8) * 1.15)*((100-County Factor)/100)*((100-Basement Factor)/100)* ((100-Discount Factor)/100) * ((100-Floor number)/100));

Fig. 5 Example DMN decision rules

Once we have a satisfying rule, the method *evaluateOutputExpression* evaluates the outputs of a decision. Each output value of a satisfying rule in a decision is calculated by evaluating the value of the output expression of that decision output using the input values and calculated output values of required sub-decisions.

Figure 5 shows the output expression for the "House Price" output in two different rules. The first rule involves a javascript-like code that is parsed by our tool, and the output expression is evaluated, and the second rule is a straightforward equation that needs to be evaluated using the required input values. Algorithm 3 shows the evaluation of an output by calling a method *evaluateOutputExpression*. The output evaluator parses the code and evaluates the output. Moreover, for a decision ***D***'s output ***OD***, each sub-decision's output ***SDO*** required to evaluate the output expression of ***OD*** is fetched recursively in sub-decisions of ***D***. Once the output is evaluated, the output value is added to the list of input values. These added values can then be used by the output expressions of the higher-level decisions. Once the traversal is completed, a row is added to the dataset of each decision. A dataset is created for each output if the decision has multiple outputs. The columns of each dataset are the decision inputs, the sub-decision outputs that were required in evaluating the decision output's expression and the decision output.

The overall time complexity of the data generation for a DMN model is $O(N * D * R * I)$ where ***D***, ***R*** and ***I*** are the number of decisions, the maximum number of rules out of all the decisions, and the maximum number of inputs out of all decisions, respectively. To evenly distribute the rows in the datasets and for the ML models to learn the rules sufficiently well, ***N*** should be large.

There have been previous works that generate process models (Van der Aalst et al. 2004) and decision models (Bazhenova et al. 2016, 2017) from event data. Our approach goes reverse, by producing data from models. We explore the mutual benefits of DMN and ML. However, due to the lack of historical data or event logs of DMN models, data availability becomes the bottleneck to exploring the benefits of ML on DMN models. Our approach provides an alternative to this bottleneck by artificially generating data such that this data can replace historical data. The data is not as normalized in real scenarios as the artificial data generated through our data generation module. The human bias, e.g., gender or economic status-based bias, is not well incorporated in our data. However, our data generation module is the means to an end of analyzing DMN decisions by explaining the importance of input features on the decision outputs. To this end, the datasets are produced with input values that conform to the valid input sets, and the outputs conform to the outputs expressions per the decision rules. Such datasets provide a quantitative representation of the decision rules. In the next step, we combine ML models on this representation to further provide a quantitative analysis of the DMN decisions.

4.2 ML for DMN

ML methods are used for pattern recognition in event logs and heaps of data. The patterns can provide valuable information that can support decision-making about the systems associated with the data. The ML module retrieves the model-driven test cases to train the ML models. If historical transaction data is available, this data can also be used during training to ensure that the ML models are trained on all valid input data on the one side while simultaneously being optimized for the most recurrent cases. We use the data and apply ML models that aim to support the business analysts by providing insights about the designed DMN model and thereby further improving or validating the DMN models. Concretely, we use ML methods to explain the importance of input features in DMN decisions.

4.2.1 Preprocessing and Pre-Training Analysis

The data needs to be preprocessed before ML training ($\mathbf{T}_6$). The preprocessing task transforms the data such that it becomes more suitable for pattern recognition by ML models, thereby improving learning. For example, a *log X* function transformation is applied to reduce the data variance. In our work, we normalize continuous variables with *log X* and *MinMax* normalizer depending upon the variance and categorical variables with *Label Encoder*. Apart from the preprocessing libraries, different normalizers and encoders from the sklearn library can be used. We observed that applying the *log X* normalizer over continuous variables with huge variance greatly improved the ML models accuracy.

The pre-training analysis module scripts create a brief overview of the data in a statistically aggregated manner. This aggregation helps to retain an overview of the complex decision-making process and associated rules. The analysis module provides, e.g., visualizations regarding the correlations of the input variables with the output variable or the pairwise correlation between various input variables for each decision table in the DMN model. This analysis helps in identifying redundant and unnecessary (because they do not or only marginally influence the output) input parameters which ultimately fosters simplification and refactoring of the decision logic (cf. Calvanese et al. 2016, 2018; Matsubayashi et al. 2012). Data without redundant variables improves the accuracy of the ML models. More concise decision tables are usually more comprehensible for the business analyst. Thus, such an analysis may provide insights into the business analyst's explicit and implicit domain knowledge on which the decision logic is based.

4.2.2 ML Model Training

Once the datasets are preprocessed, the business analyst can trigger the training of different ML models ($\mathbf{T}_7$). Some ML models are suitable for regression purposes (e.g., XGBoost) and some for classification (e.g., Neural Network), but for most models, both regressor and classifier variants are available. Depending on the business decision requirements, the analyst can choose to train the data on the appropriate ML models. Following a pragmatic mode, all implemented ML models can also be trained, then the evaluation step (see next section) will select the best ML model for the business decision at hand.

The hyperparameters of ML models need to be configured before training the models. To mitigate the subjectivity of a specific case in evaluating the ML models accuracy, we fixed the hyperparameters for all our ML models, e.g., the number of epochs for Neural Network, Catboost, XGBoost, the activation function for neural networks, and the loss functions. Although we tuned the hyperparameters for improved accuracy of ML models, we acknowledge that they are still specific to the ten cases available to us (see Sect. 5). Therefore, these parameters can (and should) be adapted to a specific case (i.e. DMN decision logic). After setting hyperparameters, the data is split into 80/20 ratio as training/test data.

We train different types decision tree based ML models: *Decision Tree (DT)*, *Random Forest (RF)*, *XGBoost (XGB)*, *CatBoost (CB)* – but any other simple as well as complex models like *Linear Regression*, *Neural Networks* may be added easily. Decision tree models can visually represent the "decisions" in the form of if-then rules, an analogy to DMN which is why we chose them. Basic decision trees cannot capture all the relationships of inputs and output in the decision rules. More complex models like XGBoost and Catboost can capture the complex relationships between inputs and outputs. Moreover, tree based models learn each feature's importance in classification or prediction task during training. The DMN models of the ten cases of the insurance company involve continuous output variables. Therefore, the models were trained for regression tasks of predicting the output variable's value. After training the ML models on the data, the models are saved for accuracy evaluation. The model with the highest accuracy is used for DMN model feature analysis i.e., superimposition (see Sect. 4.2.4).

4.2.3 Accuracy Evaluation

To analyze the DMN decision models using ML models, we need to enable business analysts to evaluate the overall quality of the ML model ($\mathbf{T}_8$). Conventionally, standard ML accuracy metrics like RMSE, MAE, and R2-score are used to evaluate the accuracy of the regression ML models. However, interpreting these metrics results is cumbersome and prone to error if no benchmark is given. To mitigate this problem, we introduce the *Custom Accuracy Metric (CAM)* (see Eq. 1) for continuous output[1]. CAM is defined as the percentage of test cases correctly predicted given a specific percentage error. The predicted value is correct if it falls within the range spanned by the true output from the ground truth $+/-$ a threshold error. We consider the output calculated when evaluating the DMN decision logic as ground truth.

[1] Note that for categorical output only binary accuracy metrics are meaningful where the prediction is either correct or not.

$$CAM(p) = \frac{\text{Number of predictions within } p\% \text{ error from truth value}}{\text{Total predictions in the dataset}} * 100 \qquad (1)$$

In our approach, the business analyst is also involved in the ML lifecycle. She chooses the decisions to be trained with ML models and whether to involve available historical data in training. The business analyst can choose the type of ML model to train on the dataset. This feature allows business analysts to learn the impact of different ML models on decisions and datasets. Using different DMN decisions trained on different ML models allows the use of any combination of ML models for different decisions in a potentially large DRG. For example, one decision has a categorical output variable that requires an ML model that is good at classifying, and another has a continuous output variable for which a regression model will be more suitable. DMN&ML enables ML training for an individual DMN decision, a combination of DMN decisions, or even the entire DRG. Once the training is concluded, DMN&ML executes analysis and evaluation experiments to provide the business analyst with many visualizations of the decision logic and information about the accuracy of the individual ML models. The business analyst can then make a conscious decision on whether to remain with the DMN decision table, integrate an ML model, or even substitute (parts of) the DMN with one of the trained ML models (i.e., the hybrid use).

4.2.4 *Superimposition*

DMN decisions with a huge number of inputs and decision rules can challenge the comprehension by human business analysts. A business analyst cannot easily infer the importance of the variables to evaluate a decision output only by looking at the DMN rules. Moreover, DMN rules only show the relationship of rule inputs with the rule output. However, the output of a rule also depends on the outputs from the sub-decisions. It is difficult to quantitatively evaluate the importance of the output from the sub-decisions. The CSV file generated from our data generation module incorporates all the decision outputs from the sub-decisions as columns and trains the ML model to learn the decision rules over this data. We can then use the trained ML model to provide us with the importance of all the inputs (rule inputs and sub-decision outputs) involved in evaluating a decision output. Therefore, with DMN&ML, we can not only use the computational power of ML to analyze and simplify DMN decisions, we also use the ML evaluation results to superimpose the original DMN model with ML results. By this, we provide business analysts with a visual representation of the feature importance, which gives a quantitative statement about the influence of a specific DMN input variable on the output variable. The relationship between feature and output variable can be non-linear. Therefore, we use feature importance which captures non-linear relationships as well. The ML model with the highest accuracy provides the feature importance values.

After we get the learnt feature importance values from the trained models, a script dynamically generates a color-coded representation of the feature importance of the DMN variables on the output (**T**$_9$). The color coding produces various shades of the base color used, depending on the importance value of the variable (i.e., the darker the color, the more important the feature). The base color corresponds to a 50% importance and is used for calculating the shading. For importance less than 50%, we use lighter shades; for importance more than 50% we use darker shades. A superimposed DMN of a real case is reported in Sect. 5.

4.3 Hybrid: DMN and ML

The possibility of following a pure DMN approach heavily depends on the complexity of the decision logic and the capabilities of humans in coping with this complexity. As the cognitive abilities of humans are limited, maintaining large DMN Decision Requirements Graphs and complex decision tables might threaten scalability. Moreover, in DMN, only rules explicitly incorporated in the model can be executed, leaving no flexibility, e.g., for producing results for small variants in input data. Eventually, the business analysis might want to consider incorporating unsupervised trained ML models, e.g., crime data of a specific region to be incorporated in calculating a live insurance policy.

The possibility of following a purely data-driven approach and the accuracy of the derived decision heavily depends on the quantity and quality of available ML training data. Often, problems of over-fitting or under-fitting are faced, which cause the quality of ML-based predictions heavily differ when confronted with real cases whose data is not well reflected in the training data. Moreover, approaches in this category face a lack of transparency and comprehensibility. Data-driven approaches often lack the context to interpret and comprehend the decision process.

DMN&ML enables the business analyst to use ML models on decisions for analysis according to the requirements at hand. This approach allows to control the AI and keeps a healthy balance between the expertise of the analyst and AI in the decision-making process. To the best of our knowledge, there is no work yet that discusses a *hybrid approach* that intertwines decision management and machine learning, thereby amplifying the mutual benefits of both approaches.

The output of a rule in a decision table depends on the rule inputs and the sub decision's outputs. If the decision is a leaf decision, the required inputs only originate from the initial input data of the DMN model. Thereby, the feature importance of each of the rule inputs can provide sufficient information about the relationship of rule inputs and outputs to the business analyst. However, if the rule input is derived, i.e., an output of a sub-decision, then the feature importance of such an input may not be sufficient because the sub-decision output further depends on the inputs of its decision table. Therefore, the business analyst might want to evaluate the feature importance of the sub-decision output as well. Our DMN&ML approach supports the selective application of ML models on DMN decision data, thereby supporting a deeper (from a higher level decision to lower) analysis of any decision and its sub-decisions. This selective application of ML models provides a hybrid use of DMN and ML.

The evaluation of the hybrid use of DMN and ML is lacking entirely in research. In the following, we introduce four separate hybrid cases that we identified in our cooperation with the insurance company (see Fig. 6). The four cases exemplify the ML-based DMN decision analysis based on the depth of the decision in the DRG. When considering the DRG as a tree, the first case is characterized by using an ML model on a leaf node. The second case is characterized by using an ML model in an intermediate layer of the tree, i.e., a tree node with a sub-decision (child node) and a parent node, both regular DMN decisions. Case three is characterized by using an ML model as the DRG's root node (i.e., the top-level decision). Eventually, the fourth case represents the use of multiple ML models in one DRG. The DMN&ML tool is capable of separating and evaluating all these cases.

To deploy the hybrid use of DMN and ML, the DMN&ML tool generates a PMML file for each trained ML model and each decision. The generated PMML files can be directly used in DMN&ML and some industrial DMN tools (see Sect. 3).

5 Evaluation

In the following, a comprehensive case-based evaluation of DMN&ML using realistic DMN models from an insurance company will be presented. We aim to show, whether the DMN&ML approach is feasible to (1) generate ML training data; (2) train ML models; and (3) superimpose DMN models.

5.1 Study Description

We apply DMN&ML in ten real cases from an international insurance company. We anonymized the data to respect the confidentiality by changing the domain toward deciding on the price for insuring a property. We used non-perturbative masking (Xu et al. 2014) to ensure that the statistical properties of the changed dataset and the original dataset are identical. In the project with the insurer, we collaboratively created 15 DMN models incorporating 101 decision tables. While we report our results in this paper on ten cases with 57 decision tables and show the steps of our approach concretely on one DMN model (see Fig. 7), we did successfully apply our approach to all DMN models.

Figure 7 shows the DRG with the top-level decision *House Price* using the inputs *Discount Factor*, *Floor number*, and the output of the sub-decisions *Garage Price*, *Land Price*, and *County Factor*. Figure 7 shows which inputs are required by decisions. The decision tables are defined for each of the four decisions present in the DRG. The top-level decision *House Price* determines the price of the property based upon several input factors like the discount factor applied to the house, the floor level, the outcome of the *Garage Price* decision, and the outcome of the *Land Price* decision. The *Discount Factor* and *Floor number* are defined by an integer value in a fixed interval. *Garage Price* and *Land Price* in turn, depend upon several factors. The *County Factor* depends upon the Federal Information Processing Standard Publication (FIPS) county code[2]. These four decisions are associated with their corresponding decision tables. For the *County Factor*, a one-to-one mapping of the FIPS code to an output variable exists. We, therefore, do not consider this decision when training an ML model.

5.2 Model-Driven ML Training Data Generation

The DMN&ML tool uses the DMN models serialized in XML format. The XML file captures the DRG, and the decision tables, including the decision logic and the hit policy. Several single-hit policies exist (cf. OMG 2020). We focus on single hit policies, which have a unique output for every input, as during the modeling project with the insurance company, we exclusively needed this hit policy for all decisions.

We defined the metadata files for all ten cases involving 57 decisions tables for our case study (see Table 1). We set the value of *N* to 100,000 as the number of data points to be generated through the simulation of the orchestration of the DMN models. Table 1 shows the time taken to generate

[2] FIPS is a standardized five-digit code that uniquely identifies counties and county-equivalents in the United States.

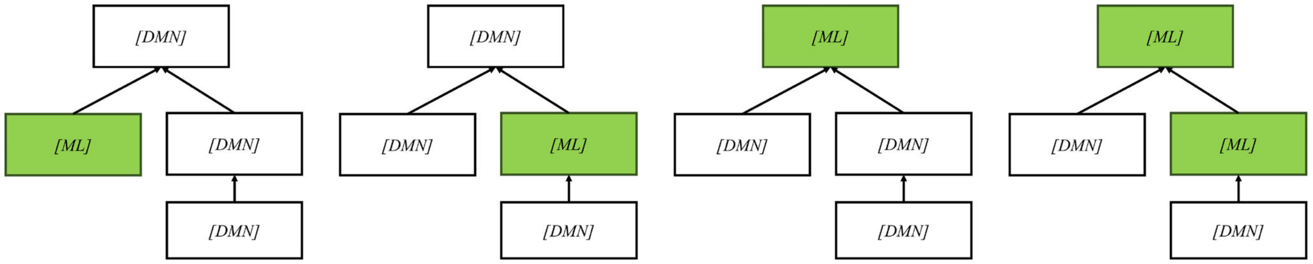


Fig. 6 Generic experimentation cases for hybrid DMN and ML use

datasets with N points. Each DMN model produces the number of datasets equal to the number of output variables combined in all the decisions because each decision can have multiple output variables. The top-level decision only had a single output variable in our ten cases. Therefore the output decision produces a single CSV file. However, the sub-decisions can have multiple outputs (e.g., Case #9 and Case #10). Therefore, our 57 decisions in all the 10 cases produced 61 CSVs. In our report, we picked 15 CSVs from all the 61 CSVs, which include the CSV corresponding to the top-level decision output in the DRG from all the ten cases as well as output CSVs from subdecisions for three cases (Case #4, #9, #10) where the DRG is sufficiently large. The 15 CSVs are chosen to cover all the different hybrid configurations as introduced in Sect. 4.3. Table 1 shows the number of rules in each of the 15 CSVs chosen from the DMN model. The required times are consistent with the algorithm's complexity, i.e., $O(N * D * R * I)$.

The objective of the data generation step is to make data available for ML models to sufficiently learn the DMN rules and thereby support the analysis of the DMN model. Therefore, the data produced should cover all decision rules such that the proportion of the data rows in the CSV corresponding to a particular rule is consistent with the possible decision paths covered by that rule. A decision rule covers more decision paths if the rule conditions are less restrictive over the input variable. E.g., in Fig. 9, an input variable "Garage Type" has three different unique values while each rule is more restrictive by allowing only one of those values. However, if the rule allows two of the three values, then the rule becomes less restrictive and covers more decision paths. Therefore, the data produced should have a higher proportion of relaxed rules and a lesser proportion of more restrictive rules.

We found that the data generation results were consistent with the rule proportion idea. E.g., the decision table of the top-level decision for Case #7 had four rules. The table has a decision input "Garage Type" such that two (say R_1 and R_2) out of four rules allow "Semi-tools Equipped" and "Fully tools equipped" whereas two rules (say R_3 and R_4) allow only "Tools Unequipped". The results show that the number of rows for R_1 were almost double to that of R_3 with more than 32,000 rows for R_1 and around 16,000 rows for R_3 out of the total of 100,000 rows. Our data generation approach thereby provides a data availability-independent and flexible way to be used in combination with many ML models. We thus proved the feasibility of the model-driven ML training data generation using real DMN models.

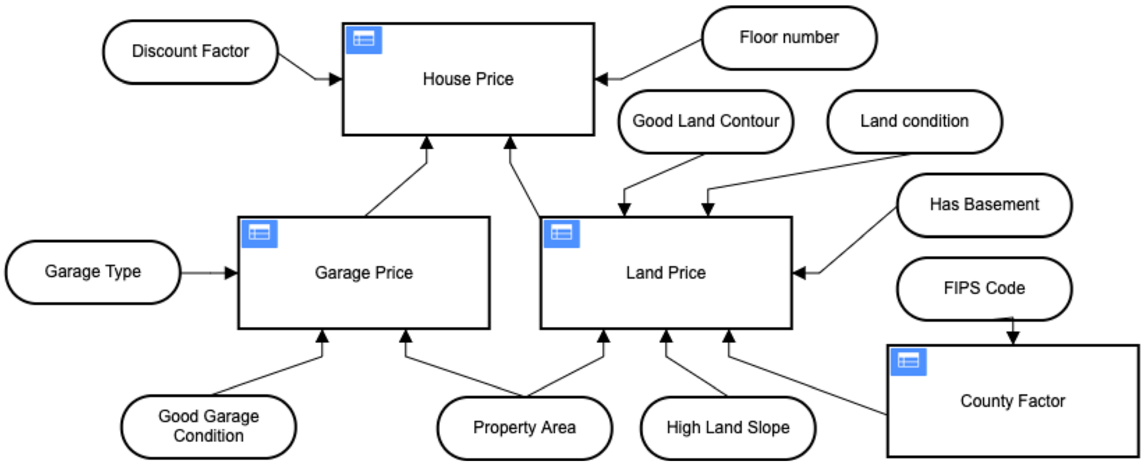
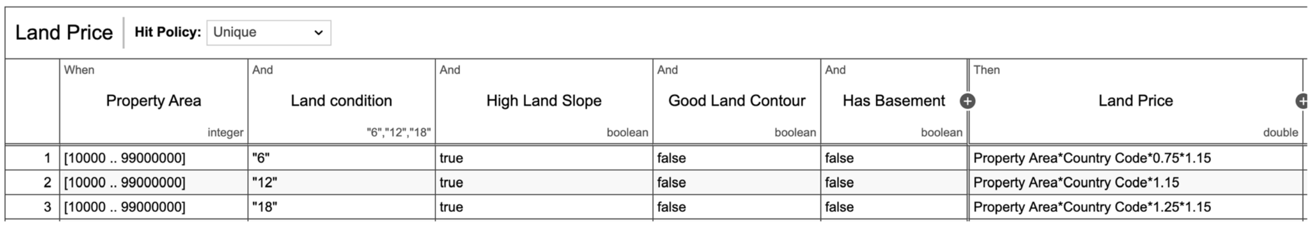


Fig. 7 DMN model of the House Price prediction case study


Land Price Hit Policy: Unique

	When	And	And	And	And	Then
	Property Area	Land condition	High Land Slope	Good Land Contour	Has Basement	Land Price
	integer	"6","12","18"	boolean	boolean	boolean	double
1	[10000 .. 99000000]	"6"	true	false	false	Property Area*Country Code*0.75*1.15
2	[10000 .. 99000000]	"12"	true	false	false	Property Area*Country Code*1.15
3	[10000 .. 99000000]	"18"	true	false	false	Property Area*Country Code*1.25*1.15

Fig. 8 Land price decision DMN table

Table 1 Results for data generation for ten different DMN models

Case #	Total inputs	Total decisions	No. of rules	Generation time (s)
1	5	1	6	477.31
2	9	4	36	1271.12
3	4	1	5	427.76
4	10	4	25	1251.23
5	11	5	101	1562.16
6	8	4	130	1654.33
7	9	4	7	1794.98
8	5	5	21	1813.62
9	15	15	130	6377.9
10	22	14	51	6052.26

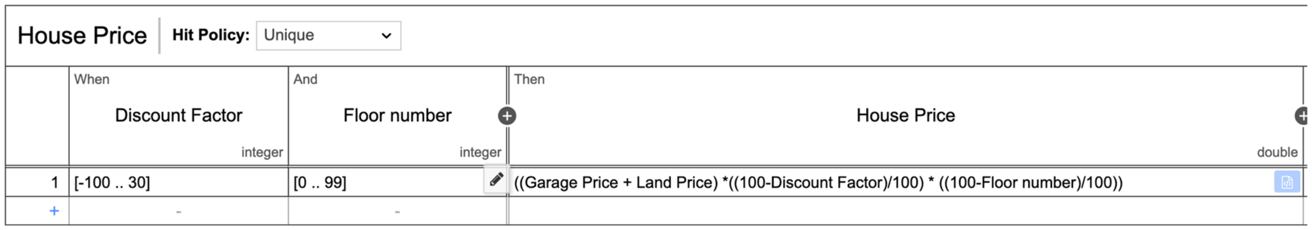
5.3 ML Models Accuracy Evaluation

Once the data generation is concluded, a preprocessing step with the *Log normaliser* to the continuous variables and the *Label Encoder* to the categorical variables was applied. Before training the ML models on the data, the data can be further analyzed to reduce features and improve training. Figure 10 shows the correlation of different extracted features with each other. The closer the value is to ± 1, the higher the two features correlate. If there is a low correlation, the value is close to 0. The analysis module focuses on data reduction and data discretization. DMN&ML analyses the importance of key variables using the correlation analysis. Moreover, the tool aims to identify redundant variables which can be removed to reduce the dimensionality of the data – and thereby increase human comprehensibility (Simić et al. 2019; Hasić and Vanthienen 2020). Very high correlations between input variables or low correlations between input and output indicate candidates for removal.

In all experiments, we were interested in two aspects: (1) the accuracy of substituting an individual DMN decision by an ML model, which is measured by the standardized Root Mean Square Error (RMSE), and (2) the accuracy of the prediction as this was the business-relevant decision of the insurance company which is measured by our introduced CAM metric. RMSE calculates the average error that an ML model has in predicting an output, but the interpretability of the average error is only possible against a benchmark. There are no benchmark models against which we can evaluate these metrics. As this work is the first of its kind, our work establishes a benchmark against which future research can compare to. On the other hand, our CAM metric (Eq. 1) does not depend upon other ML models. Hence, we used CAM to evaluate the accuracy of ML models from the business analyst's perspective.

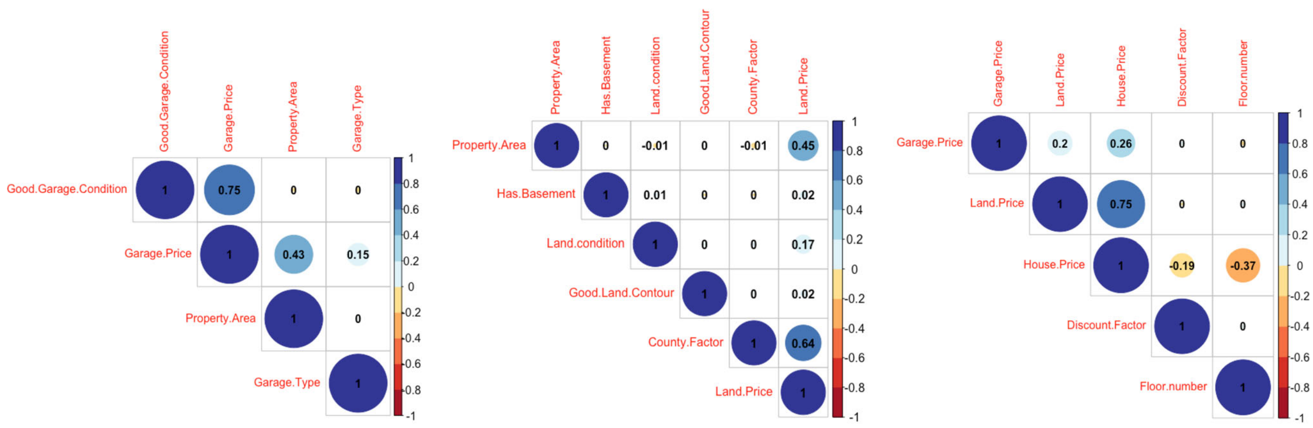
After data preprocessing, the datasets were fitted into four different ML models to predict the output variable. Table 2 summarizes the accuracy with a percentage error of $p = 1\%$. While we note that the accuracy of different ML models differs to great extents, delving into the reasons is not in the scope of this first investigation in the field and would also far exceed the available space. Our results still imply that even without much hyperparameter tuning (we also leave this for future work), ML models can learn declarative rules from DMN models. Our developed automated evaluation approach seems feasible in providing the business with relevant insights into the hybrid use of DMN and AI.

During training, the ML models learn the decision rules using the dataset (historical or produced from the data generation module) of the decision table. Each decision rule can have an expression or a more complex conditional formula to calculate the output. Moreover, each decision rule has its output expression (see examples in Fig. 5).


House Price Hit Policy: Unique

	When	And	Then
	Discount Factor (integer)	Floor number (integer)	House Price (double)
1	[-100 .. 30]	[0 .. 99]	((Garage Price + Land Price) *((100-Discount Factor)/100) * ((100-Floor number)/100))
+	-	-	

Fig. 9 House price decision DMN table



(a) Garage Price Decision (b) Land Price Decision (c) House Price decision

Fig. 10 Correlation between DMN variables (input and output) of the house pricing case

Such complexities introduce non-linearity in the generated data, and thus basic regression models perform poorly on such a dataset. This requires advanced models that handle non-linearity in the data like XGBoost or Catboost. We see that ML models learn the DMN rules to significantly different extents (cf. Table 2). We also trained the data on non-tree based models however such models performed poorly in learning DMN rules. Relative to decision tree and random forest, CatBoost and XGBoost performed the best. Hence, we conclude that with the advanced ML models, it is feasible to train the ML models with the model-driven data.

We further see that the tree-based ML models (DT, RF, XGB, CB) outperformed LR and NN, therefore, we focused only on tree based models for feature importance. This can be explained by the fact that decision tables, in a literal sense, include decisions and rules and decision trees follow a similar intuition of dividing the prediction into different rules based on the input feature values. However, in most cases, the number and complexity of rules are higher, and their decision trees and random forest are unable to perform well in such cases, and XGB and CB are resilient towards the added complexities and therefore learn the rules fairly well. Table 2 shows that XGB and CB are the top two performing ML models in almost all the cases. In some cases involving datasets of the output variable of sub-decisions (Case #9.1, #9.2, #10.1, #10.2) which have rules with much lower complexity, it is easier to learn the rules and the added complexity of advanced models become an overkill. In two cases (Case #4.1 and #9.1), linear regression also performs very well. However, this is because these cases are part of the sub-decisions and involve rules with lower complexity. Overall, the ML models learn the decision rules very well, with an average accuracy greater than 95% at an error of 1% from the truth value. Such an accuracy serves as a high confidence score for using the models to evaluate the importance of features learned during training, thereby supporting the business analyst.

5.4 AI-Enhanced Decision Analysis

The correlation analysis provides a quantitative measure of the linear relationship but does not provide a good estimate if the relationship between two variables is non-linear. ML models trained with the generated data learn the non-linear relationship of the decision input variables with the decision outputs and thus can be used to support DMN decision analysis. DMN&ML comes with the *Superimposition* functionality that feeds back the analysis results of training the ML models to the business analyst. The tool reads the *feature importance* value provided by the ML models and

Table 2 Accuracy metrics of the evaluated ML models

Case #	Machine learning model					
	RMSE, CAM					
	Non tree-based models		Tree-based models			
	LR	NN	DT	RF	XGB	CB
1	0.288, 16.04%	0.039, 90.64%	**0.016, 100.0%**	0.035, 96.49%	**0.002, 100.0%**	0.001, 99.99%
2	0.439, 20.22%	0.282, 48.10%	**0.273, 99.99%**	0.257, 32.55%	0.055, 92.64%	**0.113, 95.74%**
3	0.420, 15.1%	0.265, 29.83%	0.100, 65.75%	0.110, 56.07%	**0.020, 98.07%**	0.063, 97.58%
4	0.470, 19.62%	0.386, 31.78%	0.140, 62.29%	0.125, 63.37%	**0.035, 97.58%**	**0.065, 98.29%**
4.1	0.405, 12.6%	0.376, 36.81%	0.034, 98.87%	0.042, 98.06%	**0.012, 99.69%**	0.062, 98.65%
5	0.435, 13.66%	0.340, 29.77%	0.153, 41.43%	0.147, 40.27%	**0.026, 96.00%**	**0.027, 98.16%**
6	0.428, 17.36%	0.161, 69.37%	0.183, 42.42%	0.171, 41.86%	**0.028, 97.99%**	**0.040, 98.6%**
7	0.419, 13.41%	0.122, 50.31%	0.103, 53.32%	0.109, 48.50%	**0.014, 99.22%**	**0.016, 99.07%**
9	0.147, 55.60%	0.140, 57.40%	0.122, 64.39%	0.111, 67.65%	**0.014, 99.85%**	**0.017, 99.44%**
9.1	**0.002, 100.0%**	0.002, 99.91%	**0.001, 99.98%**	0.003, 99.95%	**0.002, 99.99%**	0.026, 99.07%
9.2	0.176, 9.17%	0.029, 99.69%	**0.002, 99.97%**	0.013, 99.87%	**0.003, 100.0%**	0.012, 99.32%
10	0.460, 18.23%	0.456, 26.56%	0.237, 34.81%	0.167, 43.82%	**0.049, 91.75%**	**0.094, 93.64%**
10.1	0.096, 40.00%	0.013, 99.74%	**0.001, 99.96%**	0.002, 99.81%	**0.001, 99.92%**	0.037, 98.50%
10.1	0.183, 14.16%	0.032, 93.89%	**0.0009, 99.96%**	0.004, 99.72%	**0.001, 99.94%**	0.024, 97.90%

automatically generates a graphical DMN-inspired representation of feature importance values over the decision inputs for each decision as shown in Fig. 11. Such a representation transfers the reported positive value provided by Superimposition (Lukyanenko et al. 2019; Maass et al. 2022) to the domain of decision management and DMN models. Thus, this kind of automation should foster comprehension of ML applications for business analysts.

Figure 11 shows the importance of each feature involved in the prediction of *House Price*, *Land Price*, and *Garage Price* decisions as well as the best performing ML model in Fig. 7. The superimposed DMN shows the business analysts the relative importance of the individual decision inputs for deriving the output for each decision. For example, the most important feature for *Land Price* is predicted as property area with almost 49%, and the other two important features include the *county factor* as well as *land condition*. It is interesting to see that the *high land slope* feature has zero importance implying that the decision input does not contribute anything to the output value calculated and thereby can be a good candidate to be removed. The other alternative could be that the decision rules need to be altered such that the feature has some importance in predicting the output. Therefore, the feature importance provides valuable insights supporting business analysts in validating and revising the DMN models.

5.5 Hybrid Use of DMN and ML

In the above sub-sections, we introduced the individual modules for generating data from the DMN model, then used the generated data to train ML models, and finally used the trained ML model in DMN decision table analysis. In this subsection, we focus on applying the output of individual modules to support the business analyst in ML-based analysis of DMN decision rules and tables. Figure 11 shows the four decisions of our house pricing case (cf. the original DMN model in Fig. 7). Figure 11 covers all the four hybrid evaluation cases. The first case of applying ML on the leaf decision shows the feature importance values of decision D_3. The second case is supported by applying ML on D_2. The third case is covered by the feature importance list of the top level decision D_4, and finally, cases 1, 2, and 3 can be applied in conjunction to cover case 4. Furthermore, D_1 is not trained on ML models because it involves a direct one-to-one mapping and, therefore, is unsuitable for applying ML. Note that the feature importance values of each decision can be evaluated independently of all the other decisions in the DMN. This allows us to analyze a decision individually; if further analysis is required, the sub-decision can also be trained and analyzed using the feature importance output. These cases provide an efficient way to understand the importance of individual decision inputs for deriving the decision output. Figure 11 shows D_4 i.e., *House Price* is highly dependent on the D_2 i.e., *Land*

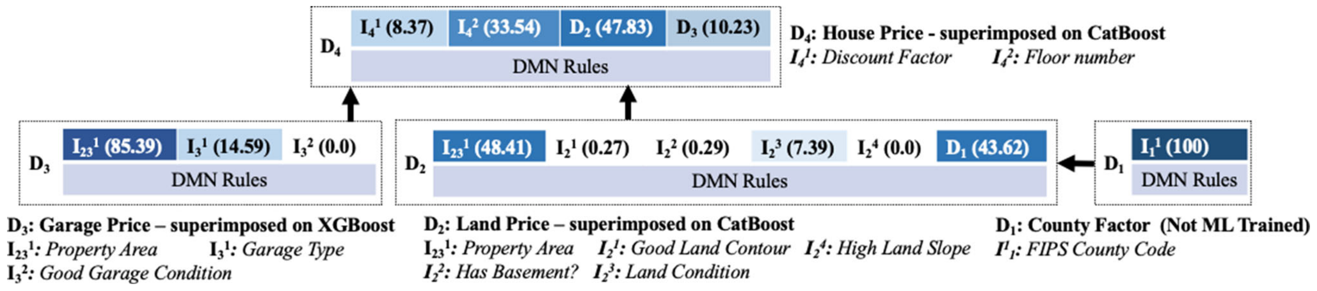


Fig. 11 ML-based feature importance superimposition on DMN decisions

Price. In order to better understand the relationship of D_2 with DMN input data, ML models are trained on the data from D_2 as well. We see that D_2 highly depends on D_1 as well as I_{23}^1. In this way, the business analyst can gain quantitative knowledge about the relationship of *House Price* with the other decisions like *Land Price* and *Garage Price* and further analyze the individual decisions as well, which will not be easily possible only by looking at the output expression of rule output, e.g., *House Price* in Fig. 9.

6 Discussion

Considering the evaluation results, we believe that our model-driven approach eases the adoption of AI for business users for DMN decision modeling. We further believe that our DMN&ML approach provides insights into the otherwise often black-box appearing AI and ML applications (Storey et al. 2022). We can thus state that we – with empirical evidence gained from the experiments – made first contributions toward achieving AI-enhanced decision management. Based on a report of ten different cases, originating from a collaboration with an insurance company, we proved the feasibility of DMN&ML with a set of real DMN models in representative complexity and size. In order to validate our findings with real users i.e., DMN modelers, we conducted a survey with two professional DMN modelers. The objective of the survey was to find out the added value of the superimposition results produced by our approach. We presented each modeler with six different DMN models with varying complexities and asked the modelers to provide an estimate of the importance ranging from 1 to 100 for each input related to a decision.

Based on the importance estimates from the modelers, we found out that the the human assessment of the relative importance of inputs was well correlated with the ML-based one for the most relevant inputs. At the same time, we learnt, that the two domain experts were very surprised and interested in the inputs to which our approach assigned a very low importance. Immediately after comparing their assessment with the ML feature importance, the domain experts engaged in discussions to elaborate on the cause and the implications of the findings. We furthermore asked the domain experts for their feedback regarding strengths, weaknesses, and usefulness of our approach. Their feedback is provided in the following:

Strengths – (1) Provides additional knowledge about the importance of the inputs which allows for a reduction in complexity and reveals hidden connections, thereby increasing and strengthening the understanding of already generated DMN models; (2) Optimizes DMN models by incorporating the weights of all inputs; (3) Acts as an additional evaluation step to find bugs for e.g., unexpected importance value of an input triggers checking of rules definition; (4) Allows already generated DMN models to be updated based on the weighting of the inputs. For example, inputs with a weight below 1% could be questioned and removed; and (5) Allows a versioning comparison of DMN models, in which the weights of the inputs from the new version are compared to the old one to detect and avoid potential errors.

Weaknesses – (1) The results involve a high complexity to be able to give the result directly to the end customer. It can be seen, that the positive feedback clearly outnumbers and outweighs the negative one. In the latter case, we assert that our approach supports the DMN modelers to assist in DMN modeling and currently does not focus on the end customer perspective. Our approach provides the DMN modeler a signal, and with her expertise the modeler can interpret the signal towards DMN model improvement. Our approach accounts for humans' limited cognitive abilities to support DMN modeling. From a business perspective, the approach fosters comprehension by visually superimposing the original DMN model with the feature importance derived from ML training.

Of course, this research also comes with limitations, some of which were already addressed above, others will be discussed in the following. Our work uses cases from a specific domain i.e., an insurance domain. More domains are required to foster the generalizability of the findings. Currently, all cases of the insurance company deal with continuous outcomes (i.e., the insurance rate customers shall pay). Future research will investigate the feasibility of applying DMN&ML for categorical outputs (e.g., credit eligibility in a bank). In our work, we did not have access to a real case producing multiple outputs (something that is theoretically possible in DMN). Our solution can, however, also generate datasets for such cases. Eventually, we want to stress that the superimposition is only possible when the original DMN model is given; without that, the feature importance can be computed, but not visualized in a DMN model. With respect to the empirical evaluation of the usefulness, we are aware of the limiting factor that we only had two domain experts participating. Still, we believe this initial empirical evaluation further adds credibility to this research and motivates us to extend the empirical investigations in the future.

7 Conclusion and Future Work Perspectives

In this paper, we proposed the first contributions toward combining humans' mutual benefits of explicit and declarative decision modeling with the computation power of data-driven machine learning (ML) approaches. We presented a framework positioned between the domains of ML and Decision Management. In this work, we explored the feasibility of using DMN models to automatically generate valid training data for ML models. We further proposed a metric and methodical support to evaluate the hybrid use of DMN and ML. Eventually, we also introduced a concept for DMN Superimposition. An approach that uses the feature importance derived from ML models and visually superimposes the corresponding DMN model with such information to support comprehension by business analysts.

We evaluated our approach with a real case of an international insurance company. The good accuracy of the ML models in our experiments shows the potential in the hybrid use of ML and DMN. The model-driven nature of our approach makes it very easy to test and analyze possibilities before deciding whether and where to apply a trained ML model. This not only complements the expertise of the stakeholder, but recent research also found that this leads to better satisfaction and protection of the role identity of the employee (Strich et al. 2021).

In our future work, we plan to further extend and improve our approach in all modules, especially concerning training data generation, analysis of DMN rules, and DMN Superimposition. Our goal is to develop a Camunda extension that integrates DMN&ML. With our work, we showcased the potential of ML in decision management. We will continue by tailoring ML to suit specific decision modeling techniques. Furthermore, we are discussing with the insurance company to partly replace their policy calculation system with our hybrid DMN and ML-based one. Eventually, we aim to extend the empirical experiments to research the perceived usefulness of our approach and the extent to which it supports business analysts in understanding the decision logic.

Acknowledgements This work has been partially funded through the Austrian Research Promotion Agency via the Austrian Competence Center for Digital Production (contract no. 854187).

Funding Open access funding provided by TU Wien (TUW).

References

Bazhenova E, Buelow S, Weske M (2016) Discovering decision models from event logs. In: International conference on business information systems. Springer, Heidelberg, pp 237–251

Bazhenova E, Haarmann S, Ihde S, Solti A, Weske M (2017) Discovery of fuzzy dmn decision models from event logs. In: International conference on advanced information systems engineering. Springer, Heidelberg, pp 629–647

Boonmepipit B, Suwannasart T (2019) Test case generation from bpmn with dmn. In: Proceedings of the 2019 3rd international conference on software and e-business, pp 92–96

Bork D (2022) Conceptual modeling and artificial intelligence: challenges and opportunities for enterprise engineering. In: Aveiro D, Proper HA, Guerreiro S, de Vries M (eds) Advances in enterprise engineering xv. Springer, Cham, pp 3–9

Bork D, Fill HG (2014) Formal aspects of enterprise modeling methods: a comparison framework. In: 47th Hawaii international conference on system sciences. IEEE, pp 3400–3409

Bork D, Garmendia A, Wimmer M (2020a) Towards a multi-objective modularization approach for entity-relationship models. In: Michael J, Torres V (eds) ER forum, demo and posters 2020, CEUR-WS.org, CEUR workshop proceedings, vol 2716, pp 45–58

Bork D, Karagiannis D, Pittl B (2020b) A survey of modeling language specification techniques. Inf Syst. https://doi.org/10.1016/j.is.2019.101425

Brambilla M, Cabot J, Wimmer M (2012) Model-driven software engineering in practice. Morgan & Claypool, San Rafael

Brambilla M, Cabot J, Cánovas Izquierdo JL, Mauri A (2017) Better call the crowd: using crowdsourcing to shape the notation of domain-specific languages. In: 10th ACM SIGPLAN international conference on software language engineering. ACM, pp 129–138

Bucchiarone A, Ciccozzi F, Lambers L, Pierantonio A, Tichy M, Tisi M, Wortmann A, Zaytsev V (2021) What is the future of modeling? IEEE Softw 38(2):119–127

Burgueño L, Burdusel A, Gérard S, Wimmer M (2019) Preface to MDE intelligence: 1st workshop on artificial intelligence and model-driven engineering. In: 22nd ACM/IEEE international conference on model driven engineering languages and systems companion. IEEE, pp 168–169

Burkart N, Huber MF (2021) A survey on the explainability of supervised machine learning. J Artif Intell Res 70:245–317. https://doi.org/10.1613/jair.1.12228

Buxmann P (2021) Interview with Karl–Heinz streibich on "artificial intelligence''. Bus Inf Syst Eng 63(1):69–70

Calvanese D, Dumas M, Laurson Ü, Maggi FM, Montali M, Teinemaa I (2016) Semantics and analysis of dmn decision tables. In: International conference on business process management. Springer, Heidelberg, pp 217–233

Calvanese D, Dumas M, Laurson Ü, Maggi FM, Montali M, Teinemaa I (2018) Semantics, analysis and simplification of dmn decision tables. Inf Syst 78:112–125

Camunda (2021) Camunda modeler. Technical report. https://camunda.com/de/products/camunda-platform/modeler/

Castellanos A, Castillo A, Tremblay MC, Lukyanenko R, Parsons J, Storey VC (2021) Improving machine learning performance using conceptual modeling. In: Proceedings of the AAAI 2021 spring symposium on combining machine learning and knowledge engineering

Data Mining Group (2020) Predictive model markup language (pmml) v.4.4.1. Technical report. http://dmg.org/pmml/v4-4-1/GeneralStructure.html

Dietterich TG (2000) Ensemble methods in machine learning. In: International workshop on multiple classifier systems. Springer, Heidelberg, pp 1–15

Etinger D, Simić SD, Buljubašić L (2019) Automated decision-making with dmn: from decision trees to decision tables. In: 2019 42nd international convention on information and communication technology, electronics and microelectronics (MIPRO). IEEE, pp 1309–1313

Etikala V, Van Veldhoven Z, Vanthienen J (2020) Text2dec: extracting decision dependencies from natural language text for automated dmn decision modelling. In: International conference on business process management. Springer, Heidelberg, pp 367–379

Fettke P (2020) Conceptual modelling and artificial intelligence: overview and research challenges from the perspective of predictive business process management. In: Companion proceedings of Modellierung 2020 short, workshop and tools and demo papers, pp 157–164

Figl K, Mendling J, Tokdemir G, Vanthienen J (2018) What we know and what we do not know about dmn. Enter Model Inf Syst Arch 13(2):1–16

Goossens A, Claessens M, Parthoens C, Vanthienen J (2021) Extracting decision dependencies and decision logic from text using deep learning techniques. In: 2021 business process management workshops, revised selected papers. Springer, Heidelberg, Lecture Notes in Business Information Processing, vol 436, pp 349–361

Guazzelli A, Zeller M, Lin W, Williams G (2009) PMML: an open standard for sharing models. R J 1(1):60. https://doi.org/10.32614/rj-2009-010

Hall P, Gill N (2019) An introduction to machine learning interpretability. O'Reilly Media, Sebastopol

Hasić F, Vanthienen J (2019) Complexity metrics for dmn decision models. Comput Stand Interfaces 65:15–37

Hasić F, Vanthienen J (2020) From decision knowledge to e-government expert systems: the case of income taxation for foreign artists in Belgium. Knowl Inf Syst 62(5):2011–2028

Hasić F, Corea C, Blatt J, Delfmann P, Serral E (2020) A tool for the verification of decision model and notation (dmn) models. In: International conference on research challenges in information science. Springer, Heidelberg, pp 536–542

Heaven D et al (2019) Why deep-learning ais are so easy to fool. Nature 574(7777):163–166

IBM Cloud Education (2021) What is machine learning? Technical report, IBM. https://www.ibm.com/cloud/learn/machine-learning. Accessed 26 Apr, 2021

Kababji SE, Srikantha P (2020) A data-driven approach for generating synthetic load patterns and usage habits. IEEE Trans Smart Grid 11(6):4984–4995

Kluza K, Adrian WT, Wiśniewski P, Ligęza A (2019) Understanding decision model and notation: DMN research directions and trends. Heidelberg, pp 787–795

KNIME Analytics Platform (2021) Knime analytics platform. Technical report. https://www.knime.com/knime-analytics-platform. Accessed 21 May 2021

Lukyanenko R, Castellanos A, Parsons J, Tremblay MC, Storey VC (2019) Using conceptual modeling to support machine learning. In: CAiSE forum 2019. Springer, Heidelberg, pp 170–181

Lukyanenko R, Castellanos A, Storey VC, Castillo A, Tremblay MC, Parsons J (2020) Superimposition: augmenting machine learning outputs with conceptual models for explainable AI. In: Advances in conceptual modeling-ER 2020 workshops. Springer, Heidelberg, pp 26–34

Lukyanenko R, Castellanos A, Samuel BM, Tremblay MC, Maass W (2021) Research agenda for basic explainable AI. In: Chan YE, Boudreau M, Aubert B, Paré G, Chin W (eds) 27th Americas conference on information systems, AMCIS 2021. Association for Information Systems

Maass W, Storey VC, Lukyanenko R (2021) From mental models to machine learning models via conceptual models. In: Enterprise, business-process and information systems modeling—22nd international conference, and 26th international conference, vol 421. Springer, Heidelberg, pp 293–300

Maass W, Castellanos A, Tremblay MC, Lukyanenko R, Storey VC (2022) ConceptSuperimposition: using conceptual modeling method for explainable AI. In: AAAI spring symposium on machine learning and knowledge engineering for hybrid intelligence (in press)

Makridakis S, Wheelwright SC, Hyndman RJ (2008) Forecasting methods and applications. Wiley, Hoboken

Matsubayashi T, Kato Y, Saeki T (2012) A new rule induction method from a decision table using a statistical test. In: International conference on rough sets and knowledge technology. Springer, Heidelberg, pp 81–90

Mueller ST, Hoffman RR, Clancey W, Emrey A, Klein G (2019) Explanation in human-ai systems: a literature meta-review, synopsis of key ideas and publications, and bibliography for explainable AI. arXiv:1902.01876

Mussbacher G, Combemale B, Kienzle J, Abrahão S, Ali H, Bencomo N, Búr M, Burgueño L, Engels G, Jeanjean P et al (2020) Opportunities in intelligent modeling assistance. Softw Syst Model 19(5):1045–1053

Mylopoulos J (1992) Conceptual modelling and Telos. Conceptual modelling, databases, and CASE: an integrated view of information system development, pp 49–68

nA (2021) Redhat machine learning. Technical report. https://developers.redhat.com/blog/2021/01/22/knowledge-meets-machine-learning-for-smarter-decisions-part-2/. Accessed 09 Apr, 2021

OMG (2020) Decision model and notation v.1.3. Technical report. Object Manager Group, Milford. https://www.omg.org/spec/DMN

Open Rules Inc (2021) Rule learner. Technical report. https://rulelearner.wordpress.com/. Accessed 21 May, 2021

Phua C, Lee V, Smith K, Gayler R (2010) A comprehensive survey of data mining-based fraud detection research. arXiv:1009.6119

Ransbotham S, Kiron D, Prentice PK (2016) Beyond the hype: the hard work behind analytics success. MIT Sloan Manag Rev 57:3

Reimer U, Bork D, Fettke P, Tropmann-Frick M (2020) Preface of the first workshop models in AI. In: Michael J, Bork D, Fill H, Fettke P, Karagiannis D, Köpke J, Koschmider A, Mayr HC, Rehse J, Reimer U, Striewe M, Tropmann-Frick M, Ullrich M (eds) Companion proceedings of Modellierung 2020 short, workshop and tools and demo papers, 2020, CEUR-WS.org, CEUR workshop proceedings, vol 2542, pp 128–129

Sandkuhl K, Fill HG, Hoppenbrouwers S, Krogstie J, Matthes F, Opdahl A, Schwabe G, Uludag Ö, Winter R (2018) From expert discipline to common practice: a vision and research agenda for extending the reach of enterprise modeling. Bus Inf Syst Eng 60(1):69–80

Sheng VS, Provost FJ, Ipeirotis PG (2008) Get another label? improving data quality and data mining using multiple, noisy labelers. In: Proceedings of the 14th ACM SIGKDD international conference on knowledge discovery and data mining. ACM, pp 614–622

Simić SD, Tanković N, Etinger D (2019) Automated decision modeling with DMN and BPMN: a model ensemble approach. In: International conference on human systems engineering and design: future trends and applications. Springer, Heidelberg, pp 789–794

Storey VC, Lukyanenko R, Maass W, Parsons J (2022) Explainable AI: opening the black box or Pandora's box? Commun ACM 65(4):27–29

Strich F, Mayer AS, Fiedler M (2021) What do I do in a world of artificial intelligence? investigating the impact of substitutive decision-making ai systems on employees' professional role identity. J Assoc Inf Syst 22(2):9

Trisotech (2020) Trisotech—business modeling and automation tool. Technical report. https://www.trisotech.com/. Accessed 27 Apr, 2021

Tsymbal A, Zillner S, Huber M (2007) Ontology-supported machine learning and decision support in biomedicine. In: Data integration in the life sciences, 4th international workshop, DILS 2007. Springer, Heidelberg, pp 156–171

Van der Aalst W, Weijters T, Maruster L (2004) Workflow mining: discovering process models from event logs. IEEE Trans Knowl Data Eng 16(9):1128–1142

Wand Y, Weber R (2017) Thirty years later: some reflections on ontological analysis in conceptual modeling. J Database Manag (JDM) 28(1):1–17

Wiemuth M, Junger D, Leitritz MA, Neumann J, Neumuth T, Burgert O (2017) Application fields for the new object management group (OMG) standards case management model and notation (CMMN) and decision management notation (DMN) in the perioperative field. Int J Comput Assist Radiol Surg 12(8):1439–1449

Xu Y, Ma T, Tang M, Tian W (2014) A survey of privacy preserving data publishing using generalization and suppression. Appl Math Inf Sci 8(3):1103

Xu N, Wang J, Qi G, Huang TS, Lin W (2015) Ontological random forests for image classification. Int J Inf Retr Res 5(3):61–74. https://doi.org/10.4018/IJIRR.2015070104

Yao W, Basu S, Wei-Nchih L, Singhal S (2015) Synthetic healthcare data generation. US Patent App. 14/762,590

Zhang L, Gonzalez-Garcia A, van de Weijer J, Danelljan M, Khan FS (2019) Synthetic data generation for end-to-end thermal infrared tracking. IEEE Transact Image Process 28(4):1837–1850. https://doi.org/10.1109/TIP.2018.2879249