FAKULTÄT FÜR !NFORMATIK
Faculty of Informatics

Kombination von Decision Modeling und Machine Learning: Eine Untersuchung im Versicherungssektor

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Data Science

eingereicht von

Georgi Milenov Dinev
Matrikelnummer 01427750

an der Fakultät für Informatik
der Technischen Universität Wien
Betreuung: Ass. Prof. Dominik Bork

Wien, 23. Jänner 2022

Georgi Milenov Dinev	Dominik Bork

FAKULTÄT FÜR !NFORMATIK
Faculty of Informatics

Combining Decision Modeling and Machine Learning: An Investigation in the Insurance Sector

DIPLOMA THESIS

submitted in partial fulfillment of the requirements for the degree of

Diplom-Ingenieur

in

Data Science

by

Georgi Milenov Dinev
Registration Number 01427750

to the Faculty of Informatics

at the TU Wien

Advisor: Ass. Prof. Dominik Bork

Vienna, 23rd January, 2022

Georgi Milenov Dinev	Dominik Bork

Erklärung zur Verfassung der Arbeit

Georgi Milenov Dinev
Adelheid-Popp-Gasse 8/2/7, 1220 Wien

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 23. Jänner 2022

Georgi Milenov Dinev

Acknowledgements

First and foremost I would like to express my deepest gratitude to my Supervisor, Assistant Prof. Dipl.- Wirtsch.Inf.Univ. Dr.rer.pol. Domink Bork for his generous support and guidance. I highly appreciate the ideas, enthusiasm, expertise, and time of Dr.rer.pol. Domink Bork, with which he supported me to complete my Master's Thesis. I gratefully acknowledge the support of the Faculty of Business Informatics, TUWien. Lastly, I would like to express my thanks to my Family for all their love and support.

Abstract

Since the last decade there has been a rapid rise in the use of BPMN (Business Process Model and Notation) standard in modeling of business processes. However, BPMN may be impractical due to its complexity and weak interoperability between business process tools. Recently, the Decision Model and Notation (DMN) standard has been introduced by OMG (Object Management Group), which is able to simplify the latter standard for decision modeling and/or multi-criteria decision-making. The purpose of DMN is to be readable and adjustable for people from business, as well as IT, respectively. The advances of technology and innovation have led to emerging big data analytics and new computational methods. Machine Learning tools are essential for the maximum utilization of the information in decisions makers. Data-driven technologies and BPMN both provide powerful tools, however according to the state-of-the-art there is no solution for coupling them in a synergistic manner. In addition, automation of modeling, using the DMN standard and the application of Machine Learning tools in this domain is still a challenge as modeling in the DMN standard requires manual steps, and ML tools are not natively supported by it. Therefore, in this thesis a Toolchain is proposed for tackling the above mentioned issues. The Thesis presents the design steps of the proposed solution. The input of the Toolchain can be either raw field data or alternatively a generated test case set from a DMN model. The proposed Toolchain implements the following three consecutive automated levels: Statistical Analysis with data preprocessing, a modeling step with three distinguished modeling strategies, and lastly an Evaluation stage. The statistical analysis covers correlation analysis, identification of the distribution of the variables, etc. The modeling stage includes fitting linear, standard Machine Learning CART and ensemble-type XGBoost models. These models are capable to handle the various levels of relationships between variables from linear to highly non-linear, which may compensate for the deficiencies of the original DMN model, since it is rather intuitive and may contain several overlapping or inefficient decision rules due to the manual creation of decision boundaries. The output of the Toolchain is a human readable result package, including the statistical analysis, the model performance evaluation and other partial results. The results obtained from experiments on a big data and a smaller insurance dataset confirms the applicability and validity of the proposed method. The results also indicate that the XGBoost model due to its outstanding performance is a suitable candidate for applying in a DMN standard instead of, e.g., a decision table. Furthermore, ML-based decision models would provide more flexibility and adaptivity

that may result in easier automation of the decision process. Benchmarking in the context of execution and training times are also performed with special regard to the model complexity. The designed Toolchain aims to bridge the gap between ML and the DMN standard. Besides, the Thesis may provide valuable insights to the domain experts' to better understand their models and empower decision makers with a different views on modeling.

Contents

List of Figures

List of Tables

CHAPTER 1

Introduction

Since the last decade there has been a rapid rise in the use of BPMN (Business Process Model and Notation) standard of OMG (Object Management Group) in system analysis and design of models. Although this approach is interesting, it may be impractical due to its complexity of its gateways and compounded symbol system. For instance, a slight modification on decision rules may have a strong impact on the whole business process model. Recently, the Decision Model and Notation (DMN) standard is introduced also by OMG [OMG], which is able to simplify the latter standard for decision modeling and/or multi-criteria decision-making. This approach is designed for describing and modeling repeatable decisions within organizations to ensure that decision models are interchangeable across organizations and to possibly complement the Business Process Model and Notation. The purpose of DMN is to be readable and adjustable for people from business, as well as IT, respectively. The advances of technology and innovation have led to emerging big data analytics and new computational methods and resources. Machine Learning tools are essential for the maximum utilization of the information in decisions making. Since the DMN standard is relatively new, there is only little discussion on the comparison and/or complementary usage with Machine Learning techniques. Rencently, the possible combination of AI and Conceptual Modeling has drawn significant attraction in the scientific community. Requirements of future modeling languages and frameworks are discussed in [Bucchiarone et al.]. In addition, a literature review about the possible applications of AI for Conceptual Modeling with special regard to predictive business process modeling presented in [Fettke], which suggests using Machine Learning and Conceptual Modeling together. Similarly, their mutual benefits are discussed in [Bork]. The same conclusions are supported by the paper [Lukyanenko et al.]. Paper [Mussbacher et al.] highlights the need for Intelligent Modeling Assistance, which is increasingly essential to reduce the cognitive burden on users when faced with complex modeling tasks. The article [Maass and Storey] also examines how Conceptual Modeling can be applied to Machine Learning and presents a framework for incorporating conceptual

modeling into data science projects.

Therefore, this thesis aims to explore the main distinctions between data driven and DMN-based decision modeling and to investigate the applicability of ML-based models to substitute the conventional decision table in a DMN decision process. In addition, the thesis addresses the issue of model generation automation in a DMN environment also.

The investigations are carried out on the available input and output data sets of a certain decision model, applied in a decision process according to the DMN standard. The observed process is from the field of car insurance sector and even though the structure of the applied decision table/model is known, it is only used for comparison with the models created from the ML algorithms. Since only the input-output observations are used, the task is considered as a black-box identification problem. Therefore, my research focuses on automating the generation of a suitable model to be used in a DMN environment either as a replacement or as a complimentary model to the original decision table. Various models are tried out, in order to find the best one with respect to various performance metrics. Since the common ML models are easily constructed/updated online, this corresponds to the pursuit of automation of decision processes. It is theoretically expected, that ML-based models may be able to substitute the conventional decision table in a DMN decision process. Furthermore, the ML-based decision model would provide more flexibility and adaptivity that may result in easier automation and faster prototyping of decision processes.

1.1 Aims of the Thesis

There is an increasing demand and interest in automated decision making and machine learning based decision support systems [Kohavi], however current business modeling tools lack the flexibility and data mining capabilities that data driven technologies can offer, thus this thesis aims to fill some of the gaps that are still present between machine learning and business process management. Based on the above briefly summarized antecedents, this thesis makes an attempt to empower decision makers with modeling and information where a data-driven approach is applicable. In addition, a challenge relies in the automation of modeling using the DMN standard and standard Machine Learning tools. Therefore, in this thesis a Toolchain is proposed, which is validated through a case-study in the car insurance sector. The Thesis presents the design steps of the proposed solution. Since CARTs are proven to be efficient in multi-criteria decision making, special attention is devoted to this technique. Furthermore, preliminary data analysis with standard methods of statistical analysis are being applied on the given dataset in order to better understand the decision rules, identify the most relevant variables that influence the outcome of the model. This Thesis also involves the excessive study of feature extraction. The evaluation takes place on multiple levels, standard metrics (for instance, RMSE, MAE, etc.) are used. Furthermore, benchmarking in the context of execution and training times are also performed with special regard to the model complexity. I also focus on the compatibility of the Toolchain of common business

modeling software, i.e. the Toolchain provides the output models in two distinguished formats. One of the model output formats is .model for applications using a Python environment, while the second one is in PMML extension. This technical solution aims to bridge the gap between ML and DMN standard, as DMN supports the PMML extension natively.

Besides, the Thesis may contribute to the domain experts', to better understand their models and make appropriate improvements if required. On this basis the Thesis covers the following main tasks: exploring the field data; exploring the state–of-the-art and recent applications of DMN; applying statistical analysis (finding the key variables, correlation analysis, data cleaning, etc.); using Python for building a CART model and standard linear/nonlinear models (e.g. Lasso, Ridge, etc.) on the field data; comparison of model performances; comparative analysis regarding the decisions in the DMN approach and in the obtained ML models, i.e. investigation of their substitutability.

1.2 Relevance of the Conducted Case Study

As the technology stack behind machine learning rapidly evolved, researchers have conducted numerous studies among vastly different domains where data mining could be applicable. In order to validate the above mentioned data analysis and modeling Toolchain, the data candidate comes from the car insurance domain. Car insurance and generally the insurance domain are the focus of many publications. [Wang] also identifies the car insurance domain as a suitable candidate for researching the capabilities of different Machine Learning models, such as random forest, GBDT and LightGBM. The paper compared the performance of the models based on different performance indicators, and found that there is no "best" model to be built, but rather, depending on what metric the user actually would like to optimize for, a different model would serve as the optimal selection.

The main problem insurance companies are aiming to solve, is to come up with a fair model that classifies insuree and assigns premiums based on their attributes. The study by [Blows et al.] shows a suitable approach for the necessary data analysis carried out on real world data, while [Hutchinsqn and Rowell] is an early work, describing the challenges, mathematical problems and possible solutions for creating such an insurance model. [Blows et al.] discusses and analyzes the possible relationship between car insurance and injuries caused by car crashes based on the Auckland Car Crash Injury Study. The paper found significant differences in the attributes of people that are uninsured, compared to those, who had insurance, furthermore, correlation between the severity of the injury and the presence of an insurance was also found. Lastly, the study by [Biana et al.] proposes a futuristic model to tackle the problems of car insurance modeling by utilizing real-time usage indicators to calculate a premium. The authors discuss the possibilities for driving risk classification models and usage based insurance pricing models, furthermore according to their research, indicators and behavioural patterns exist, which correlate with the driving risk level, these input can serve as the baseline for

an on- the-fly insurance pricing.

1.3 Formulation of Research Questions and Measurable Goals

According to the aims of the Thesis described above, the following main research questions are formulated:

- Is it possible to generate and automate the generation of a suitable model to be used in a DMN environment?
- Are ML-based models able to substitute the conventional decision table in a DMN decision process?
- To what extent would ML-based decision models provide more flexibility and adaptivity?

As a first step, in this thesis the top-down fashioned Goal Question Metrics (GQM) approach is applied in order to define measurable goals for evaluating the quality of the proposed Toolchain. The GQM approach is chosen to define the specific goals and then to impart a framework for interpreting data according to the predefined goals (see for e.g., [Solingen and Berghout] [Lampasona et al.] [Pilarski et al.]. The original GQM approach was designed for identifying defects of particular projects in the NASA Goddard Space Flight Center environment [Solingen and Berghout]. Later, the approach has been extended to be applicable in broader contexts. GQM provides specifications of measurement systems aiming to interpret measurement data in the context of specific rules and issues. Thus, the model has three levels, namely the conceptual level (Goal), operational level (Question) and the quantitative level (Metric) (see, Table 1.1).

GQM Table for Goal 1.			
-	Goal	Question	Metric
Purpose	Enhance DMN performance by	Are the obtained decision boundaries meaningful?	Subjective assessment by domain expert
Issue	finding key variables	Is There Correlation between variables ? Explaining power and relevance in the models	Functional relationships assessment matrix, Correlation Coefficients P - value, Correlation Coefficients, Convergence properties
Object	of domain data		
Viewpoint	for business decision makers		Readability of results, level of user satisfaction

Table 1.1: Goal Question Metrics 1 for the Toolchain design

Tables 1.1 and 1.2 display the GQM technique defining the problems/questions and describing and quantifying how the Toolchain may solve/answer them. The proposed Toolchain is designed with the purpose to provide a brief overview of the available field data in a statistically aggregated manner. The data should be field data with adequate details depending on the application area. The aggregated approach ensures clear presentation, which helps the decision making process and gives an overview of the data for the stakeholders. The designed Toolchain presents the rank of the importance of the variables that also supports the fast visibility and presentation of the decision problem for decision makers. In addition, the modeling is targeted to answer what decision rules the model can find and how the found decision boundaries may affect the result. Due to its flexibility and low number of dependencies, the proposed tool can be easily integrated in larger decision support systems and is also suitable for standalone usage, both as a CLI and as a Docker container. For evaluation of scalability the running (time between execution and successful finish of the toolchain) and mod-

GQM Table for Goal 2.			
-	Goal	Question	Metric
Purpose	Achieve the best modeling performance		Distributions, functional relationships
Issue	by finding the most suitable model		
Object	for the domain data	What do the statistical analyses tell us about the data to analyze for business purposes?	
Viewpoint	from a mathematical perspective	Which model provides the lowest error on test data?	RMSE, MAE, MAPE

Table 1.2: Goal Question Metrics 2 for the Toolchain design

eling (time for the ML models to finish training) times are analyzed during the design phase. In addition the runtimes on vertically/horizontally changing data is also evaluated.

1.4 Research Methodology

The theoretical considerations and their usability are validated by simulation investigations. Since, in order to build numerical simulations, R and Python programming languages are used, which offer a variety of tools and functions that otherwise are widely used in applied research. The applied scientific methods are ensuring the precision and thoroughness of the simulation results.
Design Science Research (DSR) methodology is a paradigm which offers specific guidelines for evaluation and „iteration" for research projects aiming to produce IT based outcomes, such as algorithms process models, languages, etc. The development and performance of the (designed) artifacts are in the center of this methodology with the intention of enhancing their functional performance. Due to the pragmatic nature of DSR it is widely used in applied researches, e.g. software engineering, by renowned companies also [Peffers et al.]. According to [Peffers et al.] the evaluation method types relevant to this thesis are the following:

- prototype
- case-study
- illustrative scenario

In this thesis the case-study is applied as evaluation method, because this evaluation method is a far more representative way of demonstrating the results that simple argumentation (see, e.g. [Peffers et al.)]. Despite the drawbacks of case-study-based evaluation (e.g., it is difficult to conclude general evidence) this research work prefers the use of a case-study for evaluation, because it could accelerate and ease the understanding of the application of the artifact for the user. It is worth noting, that the issue of model conceptualization and knowledge extraction, especially in case of enterprises aiming to extract internal knowledge to modeling, are also in a focus of many research (see, e.g. [Lamine et al.], [Thabet et al.] [Sandkuhl et al.)]. As there is still a lack of knowledge in conceptualizing and operationalizing conceptual modeling, despite that various modeling tools are viable in various domains [Bork et al.]. For instance, the efficiency of multi-view modeling over diagram-based modeling in Business Process-Risk Management-Integrated Method is proved by [Thabet et al.], which allows reducing the meta-model complexities during model conceptualization processes. In addition, risk consideration is also an important factor in modeling enterprise business processes due to the highly competitive markets [Lamine et al.]. Therefore, [Lamine et al.] introduces the Business Process-Risk Integrated Method framework with the purpose of establishing the integration of risk management and business process management. In this Thesis the knowledge extraction is materialized via using Machine Learning in a data-driven manner, which on the the other hand requires more data than the latter cases, but allows better choice of concepts and/or models with higher approximation power, according to the State-of-the-Art (e.g. Liu et al.).

1.5 Organization of the Thesis

This Thesis is organized as follows. Chapter 2 lays down the foundations of Model Driven Software Engineering (MDSE), Business Process Model Notation (BPMN), Decision Model Notation (DMN) and some of the state-of-the-art Machine Learning methods. Furthermore, the chapter also introduces the mathematical backgrounds of both the ML models and also the metrics, that will be used for the evaluation. In Chapter 3 the main goals of this Thesis are detailed in the context of the state-of-the-art divided into three main directions. The relevance of the proposed Toolchain integration ML in DMN is analyzed in the light of the state-of-the-art scientific results, solutions on the market and also in the light of the state-of-the art results in the insurance domain. Chapter 4 discusses the possibilities of the integration of machine learning and business process modeling and presents the proposed concept. Next, the proposed Toolchain is presented within the frame of the design process. To show the capabilities of the designed Toolchain a case study is conducted using car insurance domain data. The machine learning

models both in terms of raw performance metrics and running times are evaluated and the results are interpreted in Chapter 5. Discussions of the advantages, as well as the disadvantages of the above mentioned domain are detailed. The presented results support, that the designed Toolchain is suitable for bridging data driven techniques with business stakeholders. Chapter 6 summarizes the main findings of this Thesis and draws some apposite conclusions. Finally, Chapter 7 suggests a brief reference to possible directions for further development.

CHAPTER 2

Foundations

2.1 Model Driven Software Engineering

The research domain of Model Driven Software Engineering (MDSE) focuses on enhancing methods of modeling with the purpose of increasing efficiency and effectiveness in software artifacts development, both quantitatively and qualitatively. The importance of this field relies on the growing interest in discussing the complex software artifacts at different levels of abstraction. In addition, software is more and more common in everyday life, which leads to the growing expectations in newer and newer software artifacts. Therefore, MDSE gained much attention. A detailed description of the state-of-the art MDSE technologies and insight into the model driven practices are given in [Brambilla et al.].

The main concepts within the context of MDSE, are: models and transformations (i.e., manipulation operations on models). MDSE can be considered as a methodology which supports employing the advantages of modeling into software engineering tasks. This methodology involves the following main aspects. The Concepts cover building blocks of the methodology, covering language artifacts, actors etc. The Notations represent languages used in the methodology. The Process and rules include the activities that result in the final product, and also the rules for their coordination, control. This aspect involves also the definition of the properties (e.g., consistency, etc.) of the products or the process. Tools are applications supporting the execution of activities or their coordination and the application of notations. It can be concluded, that models and also transformations have to be represented by notation, which in MDSE is commonly referred to as modeling language. In MDSE the simplest equation is defined as "model + transformation = Software" whose terms are expressed by notations [Brambilla et al.]. However, the model needs to be designed according to the desired software aimed to be produced. At this point, model driven processes can support the selection of model types, orders, etc. and abstraction levels which should be applied depending on the desired software. Besides, an appropriate set of tools are required to allow MDSE to be

feasible in practice. For instance, from the programming point of view IDEs are required for model definitions and transformation, while compilers and interpreters support the execution of the resulted software artifacts.

Model driven approaches cover various paradigms, namely Model Driven Development (MDD), Model Driven Architecture (MDA), Model Driven Engineering (MDE) and Model Based engineering" (MBE). The hierarchy of the governing modeling paradigms are depicted in Fig. 2.1. However, in the literature a greater variety of MD (model driven) acronyms are present. Model Driven Development (MDD) refers to a development paradigm in which the fundamental output of the development process is a model. In MDD the implementation is usually automatically or semi-automatically generated from the models [Brambilla et al.]. Model Driven Architecture (MDA) is a substantive version of model driven Development. The model driven Architecture is introduced by the Object Management Group (OMG) and based on the OMG standards. In light of this, the MDA forms a subset of MDD. In this particular set, the modeling and transformation languages are standardized by OMG [OMG] [Brambilla et al.]. Furthermore, it can be observed in Fig. 2.1 that MDD is a subset of MDE. Model driven engineering covers the full software engineering procedure and various model based tasks beyond development. For instance, model-based evolution of the system or the model driven reverse engineering of a legacy system, etc. belong to the MDE level. The outer layer is called Model-based engineering, which is commonly referred to as "Model-Based Development". Generally speaking, it covers a softer version of MDE [Brambilla et al.]. This actually means, that it is a process, which is based on models, however models are not the key components of the development and may not fully represent the domain knowledge. For example, the models are rather used as blueprints or sketches.

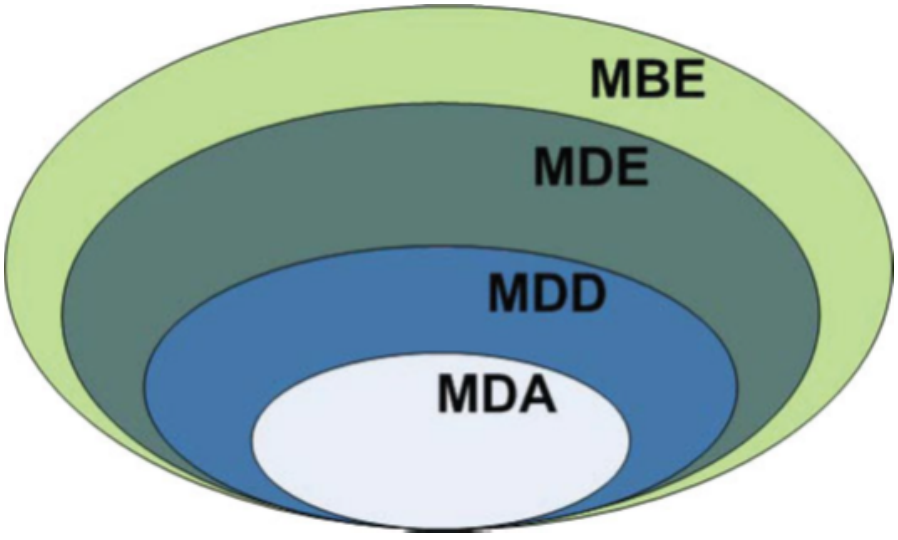

Figure 2.1: MD acronyms and their hierarchy [Brambilla et al.]

Model Driven Software Engineering dispenses a broad picture for system development including numerous aspects and summarizes the treatment of the various issues as it can be seen in Fig. 2.2. In Fig. 2.2 the rows represent the implementation, while the columns stand for conceptualization. The MDSE aims to find the solution along both axes simultaneously.

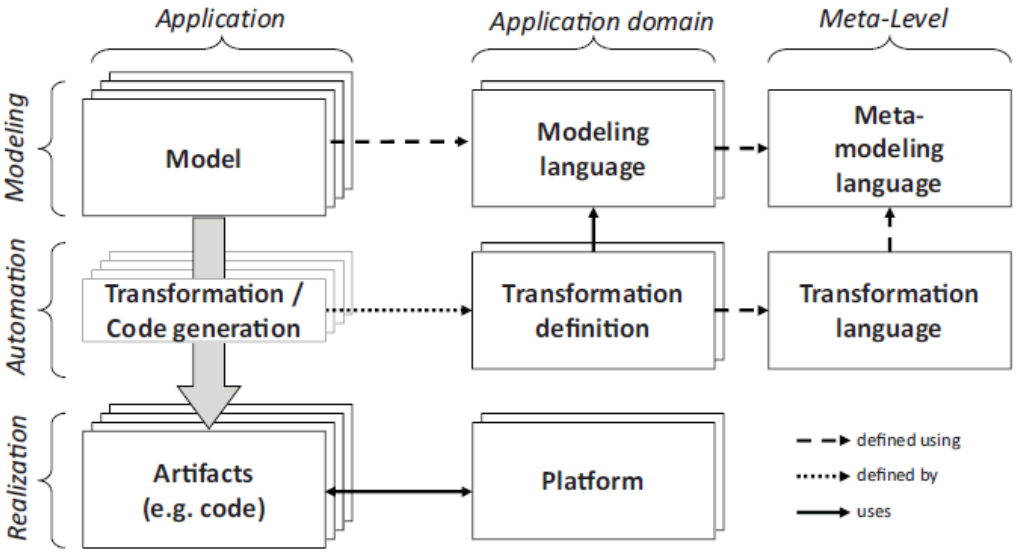

Figure 2.2: MDSE overview [Brambilla et al.]

The implementation task is responsible for mapping of the models to running systems based on the following three levels. The first is the modeling level where the models are defined. Next, on the realization level the solution is implemented, i.e. it covers the coding. The third one is the automation level where the mapping from modeling to realization is carried out. In the frame of the conceptualization task, conceptual models are formulated which describe the real cases/scenarios. The conceptualization can be performed also at three levels. At the application level the models and rules are formulated. Then, at the application domain level the modeling language and transformations are characterized for a certain domain. Finally, at the meta-level the conceptualizations are constructed. It is clear, that the main process of MDSE is a top-down process from application of models to implementation which is composed of subsequent model transformations. The MDSE process concedes the repeated employment of models and execution of systems on various platforms.

2.2 Evolution of decision modeling: from decision tables through BPMN to DMN

In this section the evolution of business process modeling is introduced starting from Decision Tables through more evolved Business Process Model and Notation (BPMN) finally reaching DMN.

2.2.1 Decision Tables

Decision tables are tabular representations aiming to specify actions to be performed in connection to certain conditions. The information collected in decision tables can also be represented by decision trees. Decision tables can be considered supervised classifiers. As it was pointed out by [Kohavi] decision tables are able to provide higher performance, than classical classifiers such as C4.5. A key task in data mining is the development of predictive models. On the basis of large amounts of predictors, a model can be designed which supports the analysts to classify or predict new instances. Many research papers dealing with the improvement of accuracy or precision of these models [Hodge et al.] [Catanzariti et al.], however only few comparative analysis has been conducted on them. For instance, the empirical evaluation of the comprehensibility of decision tables, tree and rule based predictive models is discussed by [Huysmans et al.]. The authors highlight, that the subjective nature of 'comprehensibility', which depends on several external factors (e.g. user's experience or knowledge). The authors present an interesting empirical study on the suitability of a number of alternative representation formats for classification when interpretability is an essential demand. [Huysmans et al.] focuses on decision tables, (binary) decision trees, propositional rules, and oblique rules and concludes that decision tables perform significantly better in terms of accuracy, response time, and answer confidence of users. In addition the dependency structures for decision tables are also important and well-researched areas. The study by [Chiaselotti et al.] gives a general panorama on the consistency and inconsistency of decision tables. Attribute reduction is one of the most important issues in the research of rough data sets. The types of local attribute reduction strategies for decision tables are discussed by [Liu et al.]. The authors present concepts and algorithm for the l^{th} decision class lower approximation reduction, l^th decision class reduction, and l^{th} decision class β-reduction for decision tables using discernibility matrices. [Liu et al.] also introduces the basis of the relationship between positive-region reduction and the l^{th} decision class β-reduction. Establishing the core granules of knowledge from incomplete decision tables is also a fundamental research question. According to the prevailing approaches the basic granules under similarity relation include a large number of objects, which may degrade results in data mining. The granules under limited similarity relation also have limitations, such as high computation need and weaker prediction power [Li et al.]. [Chiaselotti et al.] examines the mathematical background of the notion of similarity between objects in the granulation of decision tables through comparing the endogenous granulation induced by

Pawlak's indiscernibility with the exogenous granulation induced by a similarity measure F.

Three-way decision making also gained much interest in the research community [Yin et al.]. The three way decision making theory is based on the notions of acceptance, rejection and noncommitment, which can be considered as the extension of the binary-decision model. Three way decisions are present in our everyday decision-making and have been widely used in many fields and disciplines [Yao]. With the purpose of addressing the degradation phenomenon of three-way decision making, [Yin et al.] proposes the notion of decision tables. This approach results in a new three way decision model 3WD-D, which adopts the trisection-acting frame, and introduces three strategies namely, accept right decisions, reject decision traps and non-commitment. The authors validate, that 3WD-D resolves the degradation phenomenon and is effective in decision tables [Yin et al.]. [Li et al.] proposes the generalization of the model of three-way decision from 0–1 tables to general information tables. The authors conclude also, that optimal tri-partition can be obtained according to weighted entropy's of the finite tri-partitions.

Decision tables also form the basis of Decision Model and Notation (DMN). The growing interest of DMN decision tables to acquire crucial business knowledge, requires improved support of analysis and refactoring tasks on decision tables. [Calvanese et al.] proposes a general approach to analyze and refactor decision tables based on a geometric interpretation. Their new method is suitable for two analysis tasks (detection of overlapping rules and of missing rules) and one refactoring task (simplification of tables via rule merging).

Incomplete feature selection [Zjao and Oin] is also an important issue of decision table theory. An extended rough set model is proposed by [Zjao and Oin] based on neighborhood-tolerance relation for categorical features. The proposed model is suitable to be used on incomplete data with mixed categorical and numerical features. The authors also present a new entropy measure namely the neighborhood tolerance conditional entropy, which is used to evaluate the feature subsets. The paper reveals that in case of incomplete decision tables the neighborhood-tolerance conditional entropy is a more accurate feature evaluation criterion than the dependency [Zjao and Oin]. The topic of knowledge representation of decision tables using decision trees is discussed by [Azad et al.] based on a dynamic programming method for bi-objective optimization. The authors show that even if the global misclassification rate related to the whole tree is small enough, the local misclassification rate related to the terminal nodes of the tree can be too big. On this basis the authors provide a method which is able to design decision trees with moderate number of nodes as well as moderate global and local misclassification rates. Furthermore, the authors [Hodge et al.] have devised a neural network-based decision table methodology. The above briefly introduced approaches and findings related to decision tables are transferable to DMN based applications and provide important implications towards our understanding of business decision modeling problems.

2.2.2 BPMN

For modeling business processes, the BPMN standard is one of the most well-established and widely used process modeling standard, maintained by the Object Management Group (OMG) [OMG]. Business analytics processes consist of three main steps applied in consecutive order to the source of data. The first step includes a descriptive analysis, which describes the business history. After that, predictive analysis is performed that describes the actual situations. The second step also covers the reasons for analysis and prediction to the future situations. The third step is a prescriptive analysis process, which identifies the way how to gain profit from the predictions. The principal of BPMN is providing a representation of the real business process behaviour for any organization for supporting analyses and documentation and enhancing the processes.

BPMN is a standardized notation developed to enhance the conversation among stakeholders in a business process. The symbols of BPMN form five main sets, i.e. set of symbols of the basic categories of Flow Object, Data Object, Swimlane Method, Artifacts and Connecting Object. First, the so called Flow Object, which is a principle element in determining behaviors of business processes. There are three type of Flow Objects. The first type is called Events, which represents events that raise during business processes. The second Flow Object type represents Activities, i.e. tasks that may raise within business processes. The third type are the so-called Gateways, they are used to regulate decision flows in business processes [OMG]. The Data covers information derived from the activities of the business process. There are four types of Data, the first is Data Object [OMG]. Data Objects are for serving information about activities, which stand in need of an action. The second is Data Input, which is responsible for loading data within business processes. Thirdly, Data Output is employed for delivering data within business processes. Finally, the Data Store stores the data within a business processes. The purpose of the Connecting Object is to connect actual objects to other data. Four types of connecting objects are distinguished. The first type of Connecting Object is the Sequence Flow, which is used to define the order in which the activities are executed in a business process. The second is Message Flow, which represents the messages between participants. The third Connecting Object is called Association Symbol and is used to display relationships between artifacts and objects. The last Connecting Object is called Data Association, which is used to represent the links between data [OMG].

The above described basic BPMN elements can be grouped by using the so called Swimlanes. The Swimlane Method forms two groups. The first is called Pool, which is able to divide a set of activities. The second is called Lane, which stands for setting up and classifying activities. The information within the governing process can be described with Artifacts. Artifacts can be of two types, either Group to group objects, or alternatively in the from of text annotation to be used for additional explanation of the processes [OMG]. The elements used for describing the work in a process are called Tasks, they form the following seven groups. The Service Task carries out automatically tasks without human intervention. The Send Task supports exchanging messages. The Receive Task receives messages and/or information between recipients and senders. The User Task is a task

that can be executed by a user. The Manual Task represents a task, which can be executed without a business process. The Business Rule Task is responsible for executing a decision making task/process. The Script Task denotes an automated activity in the business process .
As it can be seen from this brief description, BPMN is a method in which a detailed component system is designed specifically for business users without in-depth IT knowledge. It is also clear that the method is limited to process modeling, but the decision logic is not reflected. This standard can be considered as the ascending method of DMN, which is a far simpler and versatile notation. A few studies reveal further issues and the combined application of BPMN and DMN, e.g. in [Boonmepipit and Suwannasart]) a test case generation from BPMN with DMN method is demonstrated. The following subsection describes the DMN in detail in order to clarify how much more advanced and advantageous it is than BPMN.

2.2.3 DMN

Decision Model and Notation (DMN) is a standard for providing decision representation as well as implementation that allows easy presentation of decisions in diagrams that are understandable by business users. DMN is suitable for modeling decisions and also their requirements. Two levels are distinguished, namely the Decision Requirement and the Decision Logic Level. The Decision Requirement Level covers an initial value, the decision and is determined from a number of input data. The input data has various sources, e.g. results from other decisions, output of other tasks, or input from devices or users. The Decision Logic Level describes the Decision Requirement Level in more detail. This level contains knowledge models which relate to the business rules and further formalism. These knowledge models are given in the form of a decision table. The decision tables contain the series of contiguous input and output expressions and indicate which decision is made based on the input data. The decisions are represented by value expressions which specify how the output value is determined from the input values [Wiemuth et al.].

Decision making processes are basically built upon two major pillars such as business process models and decision logic. Essentially DMN adds a third pillar, namely the Decision Requirements Diagram. The Decision Requirements Diagram involves all the definitions that are part of the business process models and decision logic.

The aim of the DMN standard is to provide the notation for decision modeling so that the decision could be easily presented in diagrams and understandable by business users [OMG]. The principal goals of the notation is modeling human decision-making, modeling the requirements for automated decision-making, and implementing automated decision-making models. Four types of elements are used in DMN, Decision, Business Knowledge Model, Input Data, and Knowledge Source (see, Fig. 2.3) [OMG]. This can be observed in Figs. 2.2.3-2.6 which is an illustrative example of the application of DMN with BPMN in case of a credit sales business process from [Figl et al.]. Here, the decision making is obtained by using sources e.g., Global Financial Inclusion (Global Findex)

database. It is also visible, that there are two intermediate levels. The output of these intermediate decisions are the inputs of the one final decision on credit sale. The levels of the process in order are depicted in Figs. 2.2.3-2.6 in FEEL code. The acronym FEEL referst to theFriendly Enough Expression Language (FEEL), which is an expression language defined also by the OMG DMN specification.

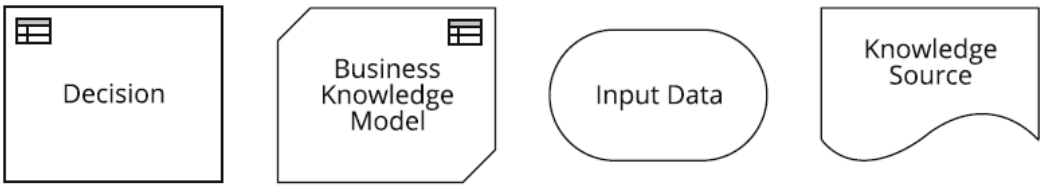

Figure 2.3: Basic types of DMN elements [Kluza et al.]

The Decision Requirements Diagram and decision logic form a comprehensive decision model. This model enhances the business process model by providing detailed specification to the decision-making acts of the tasks within the process. It becomes clear that the DMN model can only be created after the Decision Requirement Diagram is built, as they specify the decisions.

The DMN methodology is designed for decisions. DMN can be applied in combination with standards such as BPMN. Furthermore, according to [Wiemuth et al.], DMN can also be applied alongside with Case Management Model and Notation (CMMN) since CMMN, DMN and BPMN are managed by the same group. Case Management Model and Notation (CMMN) is another notation for modeling cases which could be used for different areas like licensing, banking, operations planning, patient treatment, etc., where the case is a high level activity defining actions for a known goal.

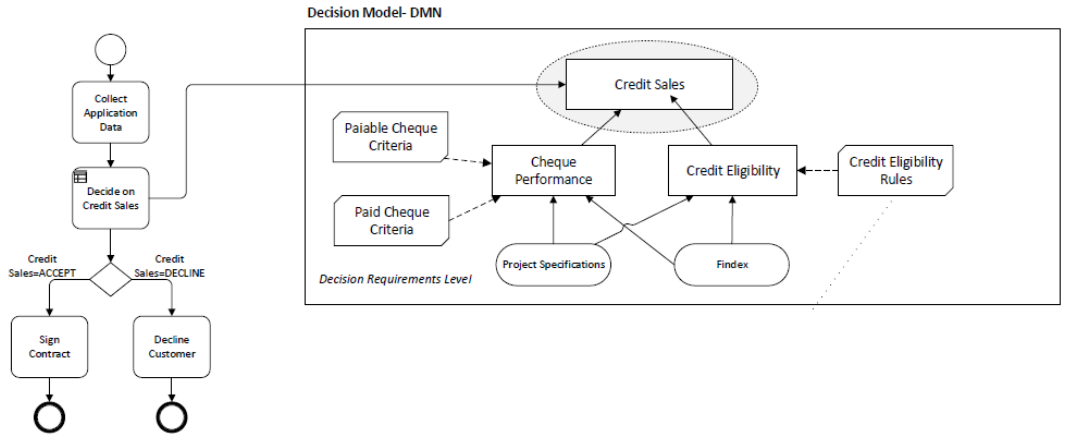

Figure 2.4: DMN operation in BPMN in a Credit Sales example - Decision Requirement Level by [Figl et al.]

Company Credit Eligibility Rules

P	Input +				Output +
	Total Credibility Amount	Project Value(PV)	Available Credit Limit	Bank Credit Warrant Letter	Credit Eligibility
1	-	<5K€	-	-	ELIGIBLE
2	3*PV<=	5K€<= PV<100K€	PV<=	-	ELIGIBLE
3	5*PV<=	100K€<= PV<1M€	1.5*PV<=	-	ELIGIBLE
4	-	>=1M€	-	=PV	ELIGIBLE
5	-	-	-	-	INELIGIBLE

Decision Logic Level

Figure 2.5: DMN operation in BPMN in a Credit Sales example - Decision Logic Logic Level by [Figl et al.]

FEEL- Expression Language

```
FEEL(
decision  table(
    inputs:[Project Value(PV), Total Credibility Amount, Available Credit Limit, Bank,Credit Warrant Letter],
    outputs:[Eligibility],
    rules:[[<5K€,-,-,-,ELIGIBLE],
            [ [5K€..100K€[, >=3*PV, >=PV, -, ELIGIBLE],
            [ [100K€..1M€[, >=5*PV, >=1.5*PV, -, ELIGIBLE],
            [ >=1M€, -,-,Available, ELIGIBLE],
           [-,-,-,-,INELIGIBLE]],
 hit policy: P, completeness: C))
```

Figure 2.6: DMN operation in BPMN in a Credit Sales example - FEEL code [Figl et al.]

In the literature, only few papers have discussed the DMN models from the process control flow's [Janssen et al.] or event logs' point of view [de Leoni et al.]. However, the method of unbundling decisions hidden in BPMN process models is less investigated. For this reason, the paper by [Bazhenova et al.] provides a pattern based approach with

the aim of supporting decision designers and analysts in understanding on how the data explicitly represented in a process model may be modeled in a separate decision model.

2.3 Brief Overview of the Most Relevant ML Algorithms

This section provides the mathematical background of linear regression (lasso, ridge, linear regressions with regularization on parameters with L_1 and L_2 norms respectively), CART and XGBoost [Chen and Guestrin], the methods that are used in this study.

2.3.1 Brief Introduction to Standard Regression Methods

Regression essentially means the measure of the relation between two quantities. If $X = x_1, ..x_n$ and $Y = y_1, \ldots, y_n$ are the measured and predicted examples, then the objective is to find the appropriate f function, which maps the relation between x_i and y_i quantities. With function f now we can determine the value of y^* at a new x^* point. Thus, from the samples we can derive a functional relationship, i.e. this means that regression can be considered as the most elementary learning model. The general form of a linear regression model is given as follows:

$$Y = \beta X + \epsilon \tag{2.1}$$

$$y_i = \beta_0 + \beta x_{i,1} + \ldots + \beta_n x_{i,N} + \epsilon_i, i = 1, \ldots, N, \tag{2.2}$$

in which, Y is the response variable vector and X is the matrix of the explanatory variables, furthermore y_i is the response variable from the i-th observation and $x_{i,p}$ is the p-th input variable of the i-th observation. The symbol β stands for the vector of the regression parameters and ϵ is an error vector.

In order to find the β coefficient vector the most commonly used estimator, the least squares is used, which minimizes the residual sum of squares (RSS), the minimization problem that we need to solve is the following:

$$RSS(\beta) = \sum_{i=1}^{n}(y_i - f(x_i))^2 = \sum_{i=1}^{n}(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j)^2 \tag{2.3}$$

Finding the β coefficient vector that minimizes the RSS can be done by differentiating the above equation and setting it to zero, thus we get the following explicit form for the coefficient vector:

$$\hat{\beta} = (X^T X)^{-1} X^T y \tag{2.4}$$

It is worth noting that $(X^T X)^{-1}$ does not exist if two or more variables perfectly correlate (have a correlation of 1), this would result in X matrix not being of full rank, meaning its inverse will not exist. This also means that there are multiple solutions one could find

that fulfills our optimization problem in such situations [Filzmoser], the mathematical proof of this result can be found by [Filzmoser].

Classical least squares regression suffers from correlated variables in the sense of exploding coefficients, to tackle this problem shrinkage methods can be applied, the most widely used methods are Lasso and Ridge regressions. Both method shrink the coefficients by applying a penalty on the coefficient sizes in the minimization problem.

$$\hat{\beta}_{Ridge} = \underset{\beta}{\operatorname{argmin}} \sum_{i=1}^{n} (y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j)^2 + \lambda \sum_{j=1}^{p} \beta_j^2 \tag{2.5}$$

$$\hat{\beta}_{Lasso} = \underset{\beta}{\operatorname{argmin}} \sum_{i=1}^{n} (y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j)^2 + \lambda \sum_{j=1}^{p} \beta_j \tag{2.6}$$

Above equations show the Ridge and Lasso minimization problems, in both cases $\lambda >= 0$ denotes the complexity parameter, which controls the amount of shrinkage. The Lasso problem is non linear and does not have a solution in a closed analytical form, they require quadratic programming to be solved. The L1 penalty term enables the coefficients to be exactly 0, thus the Lasso regression can also be used for variable selection [Filzmoser].

2.3.2 Mathematical Background of Decision Tree Methods

A Decision Tree is a non-parametric supervised learning method in the machine learning domain that can be used both for classification and regression problems. The main idea behind the algorithm is to build a model that predicts the target value based on learnt decision rules from the available data.

Tree methods are flexible in the fields of applications, but also computationally intensive, while the resulting models are highly dependent on the observed data, even a small change in the observations can induce a drastic change in the shape of the tree structure [Filzmoser].

In order for the Decision Tree Algorithm to work, we must define the following steps:

- Attribute selection rule: Selection of the appropriate feature to split the data.
- Rule of further reduction: With the help of this rule we recursively split the data further in smaller regions, thus creating new nodes in the tree.
- Termination Rule: This rule determines the conditions in which a node will not be further split, thus a leaf (end node) will be created.
- Classification Rule: This rule renders a value for every node.

With the help of the above mentioned 4 rules, a generic tree building algorithm works along the following four steps:

1. Step 1: With the help of the termination rule, we decide whether the current tree needs to be split further or not. If not, then we assign the value to the current node with the classification rule. Otherwise we proceed with Step 2.

2. Step 2: For the given dataset we apply the attribute selection rule to select an attribute and proceed with step 3.

3. Step 3: With the given attribute we split the dataset based on the results of the rule of further reduction, and proceed to step 4.

4. Step 4: We perform the operation steps for all the branches obtained from the subsets.

It is worth noting, that many of the widely-used machine learning Decision Tree techniques basically emerge from the famous ID3 algorithm introduced by [Quinlan]. The goal of the algorithm is to find the simplest tree that still predicts the data well enough. For this reason, variables that are not of high importance, should not be considered in the model building phase.

Building a decision tree is a hierarchical process as we create an ordered set from the feature variables. Building a minimal decision tree is an NP-Hard (Non-deterministic Polynomial time) problem, thus the use of a heuristic is necessary. The key question in the algorithm is the choice of the attribute to split the data with, as we would like to minimize the size of the tree, we want to have the least amount of splitting possible. Entropy is the fundamental idea of the heuristic, namely Shannon's Entropy is used:

$$E_s = -k \sum p_i log_2 p_i \tag{2.7}$$

In the Shanon's Entropy p_i denotes the probability of the information value, $log_2 p_i$ describes the amount of information. The entropy function E_s is the expected value of the information, it has its maximum value, where the probabilities are equal [Dombi].

2.3.3 Ensemble Techniques and XGBoost

Boosting belongs to ensemble learning algorithms [Dietterich], which combine the output of multiple tree models in order to obtain an enhanced solution. The literature presents various results for the prospective application of boosting methods in various fields, including car insurance sector (see for e.g., [Guelman]). XGBoost (*Extreme Gradient Boost* is a recently introduced gradient boosting strategy presented by [Chen and Guestrin], which applies an ensemble of decision tree models as learners. The formal description of the method according to [Chen and Guestrin] is the following:

$$\hat{y} = \mu(x_i) = \sum_{n=1}^{N} f_n(x_i), \tag{2.8}$$

in which $\mu(x_i)$ represents the XGBoost model with $n = 1, .., N$ number of $f_n(\cdot)$ weak learners. The efficiency of this method emerges from the substitution of the classical gradient boosting method with the Newton boosting [Chen and Guestrin]. The optimal parameters are obtained by minimizing the Λ loss function given by Eq.2.9

$$\Lambda(\mu) = \sum_{i=1}^{k} v(\hat{y}_i, y_i) + \sum_{n=1}^{N} R(f_n) \tag{2.9}$$

$$R(f_n) = \gamma T = \frac{1}{2}\lambda \left\| w \right\|^2 . \tag{2.10}$$

In Eqs. 2.9-2.10 the expression $R(f_n)$ represents the regularization term used for penalizing the n^{th} model's complexity. The function v stands for the cost function. The symbol k denotes the sample size and T represents the number of terminal nodes. the symbol γ is a control parameter related to the tree structure and λ is a regularization control parameter. The f_t term is added with the purpose of tackling the difficulties of solving the problem in Euclidean space [Chen and Guestrin]. Then, the function is formulated as follows:

$$\Lambda^t = \sum_{i=1}^{k} v(\hat{y}_i, y_i^{t-1} + f_t(x_i)) + R(f_t) \tag{2.11}$$

In Eq. 2.11 the expression y_i^t represents the prediction of the i^{th} input in the t^{th} iteration of the f_t corresponding model. Next, a second order Taylor series expansion on $v(\hat{y}_i, y_i^{t-1} + f_t(x_i))$ is performed, for providing faster and simplified calculations [Chen and Guestrin]. Now, Eq. 2.12 is obtained [Chen and Guestrin]:

$$g_i f_t(x_i) + \frac{1}{2} h_i {f_t}^2(x_i), \tag{2.12}$$

where

$$g_i = \frac{\partial v((y_i, \hat{y}_i^{t-1})}{\hat{y_i^{t-1}}}, \tag{2.13}$$

$$h_i = \frac{\partial^2 v((y_i, \hat{y}_i^{t-1})}{\hat{y_i^{t-1}}} \tag{2.14}$$

and

$$\Lambda_t = \sum_{i=1}^{k} \left[\lambda \left(y_i, \hat{y}_i^{t-1} \right) + g_i f_t(x_i) + \frac{1}{2} h_i {f_t}^2(x_i) \right] + R(f_t) \tag{2.15}$$

Using Eq. 2.15 the weights of the leaves are calculated as given by Eq. 2.16 [Chen and Guestrin].

$$w_j = -\frac{\sum_{i \in I_j} g_i}{\sum_{i \in I_j} h_i + \lambda} \tag{2.16}$$

In Eq. 2.16 I_j, $j = 1, .., T$ denotes the example set of the j^{th} leaf node. From Eq. 2.16 the following objective function can be deduced for the j^{th} node for q structure, as

$$\widetilde{\Lambda}^{(t)}(q) = -\frac{1}{2} \sum_{j=1}^{T} \frac{\sum_{i \in I_j} {g_i}^2}{\sum_{i \in I_j} h_i + \lambda} + \gamma T. \tag{2.17}$$

Then, the Γ information gain is formulated in Eq. 2.18 as

$$\Gamma = \frac{1}{2} \left[\frac{\left(\sum_{i \in I_{jL}} {g_i}^2\right)^2}{\sum_{i \in I_{jR}} h_i + \lambda} + \frac{\left(\sum_{i \in I_{jR}} g_i\right)^2}{\sum_{i \in I_{jL}} h_i + \lambda} + \frac{\left(\sum_{i \in I_j} g_i\right)^2}{\sum_{i \in I_j} h_i + \lambda} \right] - \gamma, \tag{2.18}$$

in which I_{jL} and I_{jR} denote the example sets of the left and right leaf nodes after splitting [Chen and Guestrin].

2.3.4 Performance Metrics

For performance evaluation the most-commonly used metrics, such as Root Mean Square Error (RMSE), Mean Absolute Error (MAE) and Mean Absolute Percentage Error (MAPE) are applied during my investigations. Percentage errors, such as MAPE (or alternatively, the SMAPE, Symmetric Mean Absolute Error), are scale independent metrics and frequently applied for prediction performance evaluation. RMSE and MAE are scale-dependent measures which are advantageous, also for finding the best performing one of several models. The application of MAE and MAPE is useful for comparing performance of various models. In contrast, from the above three metrics only the RMSE has the same dimension as the input data. Thus the RMSE metric can represent information about the performance on its own by comparing the RMSE value to the range of the variable that needs to be predicted. The formulae of the metrics are given by Eqs. 2.19 - 2.21.

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n} (y - \hat{y})^2}{n}} \tag{2.19}$$

$$MAE = \frac{1}{2} \sum_{i=1}^{n} (y - \hat{y}) \tag{2.20}$$

$$MAPE = \sum_{i=1}^{n} \frac{1}{2} \left| \frac{(y - \hat{y}}{y} \right|, \tag{2.21}$$

where y_i $(i = 1, ..., n)$ are the input data points and $\hat{y}_i$ $(i = 1, .., n)$ are the predicted values at the i^{th} iteration. *RMSE* and *MAE* are scale-dependent measures which are advantageous, also for finding the best one of several models.

CHAPTER 3

State of the Art

3.1 Review of the State-of-the-Art Decision Modeling Approaches

3.1.1 Literature review on BPMN and DMN

A growing number of literature have discussed the various approaches of Decision Support Systems [Sauter, Simon, Caetano et al.] incorporating classical and the most recent approaches. Decision Support Systems, particularly Business Intelligence is one of the well-documented disciplines. However, aligning decision-making within the organization with large data sources is still a challenging task. Latest studies focusing on standards such as Business Process Model and Notation (BPMN), or the recently presented Decision Model and Notation have also gained much attention [OMG]. Detailed description of these two standards, revealing the advantages and disadvantages can be found in the paper by [Biard et al.]. The authors highlight, that DMN offers greater simplicity, but cannot deal with uncertainty and is limited to pre-defined decisions made from known criteria.

[Bazhenova et al.] highlights the challenge of deriving decision models from process models, especially when same data is used in both processes. The Authors identify patterns which represent potential representations of data in BPMN processes and that can be used in building decision models related to existing process models. The paper distinguishes a set of decision patterns that characterize process-related data that is used for making decisions in existing process models via investigating a health-care process. [Bork et al.,Bork et al.] critically analyses the visual modeling languages such as the BPMN and conducts a systematic analysis on currently used standard visual modeling language specifications with a focus on the specification of notational aspects.

3.1.2 Literature review of Decision Modeling in relation to Machine Learning

As machine learning is more widely used, the requirement for understanding how these algorithms make decision became mandatory. The term Explainable AI (XAI) addresses this requirement of both explainability and transparency from the business stakeholder's point of view. According to the results collected by [Mehdiyev et al.] exactly the above mentioned requirements are discussed in the context of creating a machine learning based decision support system in the public administration domain. The publication reveals the challenges (imbalanced data, non-linear relationships etc.) that such a system would face, and proposes the usage of more sophisticated ML methods such as deep learning and ensemble methods. Furthermore recommendations on how to communicate the logic behind these models to the stakeholders are also discussed in the study.

Most of the DMN and generally business process models are documented in the form of natural text. This approach makes the models easily understandable for humans, but not for computers. To tackle this issue, a recent publication, [Etikala et al.] realized a Natural Language Processing (NLP) pipeline that can build a decision requirement diagram based on natural text. The proposed approach uses state of the art machine learning techniques from the domain of Natural Language Processing (NLTK, SpaCy, CoreNLP, etc.,) to automate the process of building a business process model based on the contents of the given model description. Evaluation of the proposed transformation pipeline shows, that the machine learning based solution is successful in identifying the structure of the Decision Requirement Diagram with the correct number of nodes and items, however the obtained decision labels are not always perfect. Furthermore, the paper discusses also the limitation of the proposed method, namely their pipeline assumes that the textual description of the models are written in grammatically correct English language, contains no irrelevant information, and that the model description is sequential.

A few researchers have addressed the issue of the DMN standard within a big-data context. For instance, [Horita et al.] present a DMN model and [Horita et al.] further improve it by simplifying the notation and use a layer-based structure. The effectiveness of the proposed method is demonstrated via an application example of the Cemaden monitoring system. No one to the best of our knowledge has studied the performance of DMN-based method in combination with ML based techniques in decision modeling, nor in comparison to Machine Learning models. However, Machine Learning methods are found to be efficient in decision making in car insurance sector according to [Wang], in which the most important features of automobile insurance data are identified. This result may serve useful source for my further investigation and basis of the preliminary data engineering.

To sum up, this brief panorama of the literature support, that comparative analyses and investigations of DMN and Machine Learning techniques are still important and challenging issues.

3.1.3 Literature review of Decision Modeling in relation to Nondeterministic Processes

In spite of the success of DMN, it is still not considered as a widely used modeling standard, since it faces some limitations. In the non-deterministic processes of medical treatment, various factors, e.g., combination of different diseases, preexisting illnesses of patients, knowledge, equipment of hospitals, etc. play a role in the decision on a patient's treatment or in a medical diagnostics issue. In other areas, standardization and formalization of processes have led to an increase in productivity, however in medical treatment there is still no general technique for modeling such a complex decision process. Early studies on decision making techniques in medical decision process report the application of the Dempster-Schafer evidential reasoning theory, Bayesian formalisms for managing uncertainty, and techniques from the field of classical statistical analysis. Recently, attempts are made for the introduction of Business Process Model and Notation in clinical practice given that BPMN is a highly standardized and deterministic methodology [Wiemuth et al.].

[Wiemuth et al.] supports that the combination of CMMN and DMN could provide a better overview of the possible tasks by depicting complex decisions in a compact form, i.e. we can showcase flexible and weakly structured processes. Thus, these process models could become easier to read and more understandable.

One of the greatest drawbacks of the DMN methodology is that it lacks the ability dealing with uncertainty present in the data [Abdelsalam et al.].

[Abdelsalam et al.] highlight that DMN is not able to represent and automate tactical and strategic decisions. [Abdelsalam et al.] make an attempt to enhance the decision requirement level and decision logic level by adding new elements (Uncertainty Element and OR Model Base Element). Their solution is however only valid for a limited number of problems. Interchangeability of DMN with other paradigms for knowledge bases (specifically the state-of-the-art IDP Knowledge Base System) is investigated by [Deryck et al.]. The analysis supports that DMN and IDP perform similarly in solving decision-dependent problems. The authors highlight that that DMN and IDP are mostly compatible with each other in problem solving issues of similar nature.

3.1.4 Literature review on the research directions of DMN

According to a literature review based on the exhaustive study of prominent papers published by [Kluza et al.], the research directions related to DMN can be organized into seven main directions (see, Fig. 3.1.4).

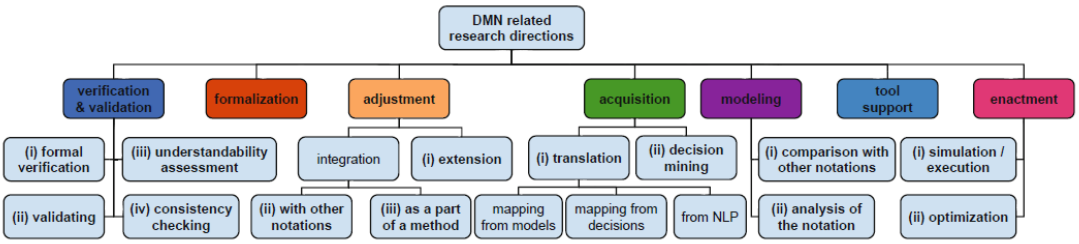

Figure 3.1: Research directions related to DMN [Kluza et al.]

Based on this categorization, **this Theses contributes to the *Validation and Verification* and *Tool support* research directions of DMN**. In addition, the scope of the investigations may **give rise to pursuits towards automation** of decision processes using DMN [Etinger et al.], since the creation of machine learning models can be easily automatized (e.g. at update scenarios).

3.2 Relevance of the Thesis Compared to Available Software Products on the Market

In the past years, the rapid development of data driven technologies, and the increase of highly available large amounts of computation power allowed machine learning and artificial intelligence based solutions to become mainstream. The recent trends indicate that more and more companies are experimenting with their own, or alternatively already existing "open source" and "pay to use" services. This thesis specifically focuses on machine learning assisted decision making, which is an already actively researched and somewhat matured domain, however the available resources (both theoretical and practical) on the combination of business process modeling and machine learning are still scarce. During the development of the practical part of the thesis, I paid attention to develop a tool that is meant to be used by people, who might not have advanced knowledge in the above mentioned fields, thus it does not require a large effort to operate, and its output contains meaningful and understandable content for the user.

In order to position the developed Toolchain and its capabilities, a comparison to an already existing software, RuleLearner [OpenRules] is carried out. This tool aims to address and solve similar problems in the field of business intelligence as this thesis, namely the creation of decision models with the help of machine learning, testing and

analysis of the decision models and lastly the deployment of the models to be used in the real world. The understanding of the available data is crucial for the user to be able to draw meaningful conclusions and make adequate decisions from the results of any machine learning model. RuleLearner however does not offer any data analysis features, this is one key difference that separates RuleLearner from the Toolchain presented in this thesis. Furthermore, RuleLearner offers decision modeling based on two CART algorithms, namely Ripper and C4.5, the presented Toolchain offers a slightly more versatile collection of models, namely on top of CARTs, linear models, and XGBoost. For the validation of the models, 5-fold cross-validation is also implemented.

3.3 Related Works

In this section I conduct a brief review of recent applications/solution which may hold similarities to the proposed Toolchain. I have found two solutions, that are worth discussing. In the study published by [MLF] an approach is outlined, which describes how the DMN methodology can be combined with ML. The designed pipeline is referred to as MLChain by [MLF]. Figure 3.2 shows the MLChain pipeline for a sample Churn (the annual percentage rate at which customers stop subscribing to a service or employees leave a position) use case. The pipeline is designed in Knime, which is an open source tool to create machine learning pipelines through a graphical user interface. The main advantage of Knime is that it allows designing ML pipelines without writing any code. The introduced pipeline integrates both data preparation and the modeling steps. Those outputs then can be used in a DMN environment. Furthermore a job for the evaluation of the fitted models is also present. Similarities to the designed pipeline (introduced in section 4.1.1) can be recognised such as the data partitioning step and also the selected ML method, namely the decision tree regressor are present in both solutions. The depicted pipeline also shows the integration with PMML (Predictive Model Markup Language), which is a syntax that the DMN Standard supports, thus integration and execution of ML models inside a business process could be realized more easily.

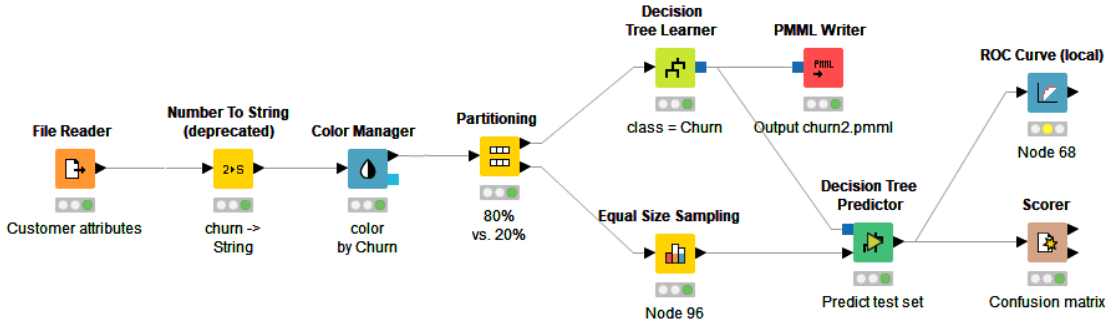

Figure 3.2: The MLChain pipeline.

Figures 3.3-3.4 show the designed pipeline for the *Text2Dec* (Text to Decision) project

[Etikala et al.]. Text2Dec can understand natural language and derive decision requirement diagrams from the corpus. The NLU (Natural Language Understanding) pipeline is depicted in Fig. 3.4, which also shows similarities to the Toolchain design presented in this thesis, namely before the machine learning parts, preprocessing and data analysis is performed.

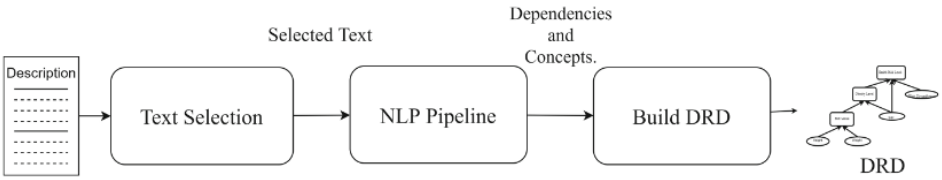

Figure 3.3: Text2Dec pipeline.

The preprocessing step in the Text2Dec pipeline is responsible for cleaning the input text, by removing determinants, such words are e.g. "The", "a", "and" etc. The following step is Coreference Resolution, which identifies and resolves coreferences (synonymous terms and phrases) by replacing them by their first occurrence. Anaphora Resolution is responsible for the clarification of ownership of attributes by the entities, and bringing the text to a consistent syntax to identify ownership relations. The next step is the Concept Recognition, which breaks down sentences into tokens, from which the concepts are derived, which will be the main building blocks of the decisions. The Dependency Parsing step is used to map dependencies between concepts in the form of a dependency tree. Lastly in the Dependency Extraction step, the concepts are linked to the identified dependencies with the help of the dependency tree created by the previous step, the output of the job is a DRG (Decision Requirement Graph), which will serve as the basis for the construction of the DRD (Decision requirement Diagram), which is the final output of Text2Dec.

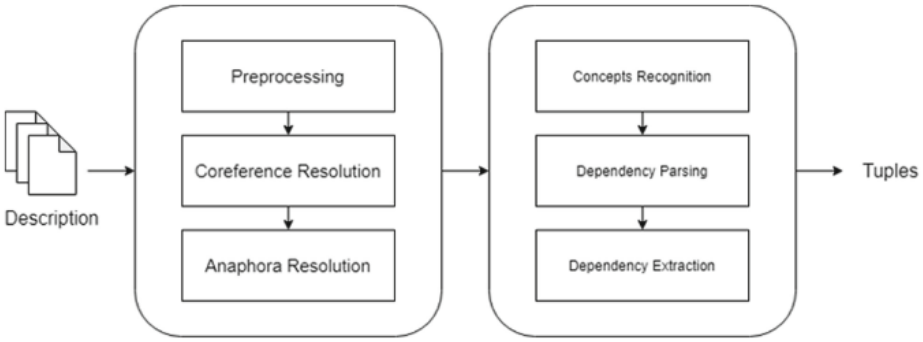

Figure 3.4: Scheme of the NLP for Text2Dec application.

CHAPTER 4

Method

4.1 ML Supported Toolchain for DMN Modeling

4.1.1 Structure of the Proposed Toolchain

In this section I introduce the proposed method through a case study. First, the architecture of the designed Toolchain is introduced as follows. The pipeline of the general method can be seen in Fig.4.1.

The aim of this pipeline is fast model prototyping for decision support in risk management based on DMN model input. The intention behind the proposed structure is to enhance the capabilities of DMN models by using Machine Learning. The Pipeline is in accordance of DSR paradigm principles, since the structure allows multiple modeling approaches, that supports enhancing the resulted artifacts functional performance by finding the best fitted model. In addition the application of Machine Learning also contributes to the improved performance of the final decision, which would not be available with using the DMN model alone. The first stage of the designed pipeline requires an interaction from the domain expert, who is responsible for providing the business process model in XML format used by the DMN notation. Next, the model is imported into an application written in JAVA programming language. Then, by querying the input model from the DMN system, the test cases are generated. The output of the JAVA application serves as an input for the ML modeling pipeline, which is designed in the context of this thesis.

The next parts of the Toolchain are automatic, but first the domain expert / user of the tool needs to configure the program via setting environment variables, such as configurations including e.g., the location/path to the output file of the test case generation from the JAVA application, the name of the target variable that needs to be predicted, etc.

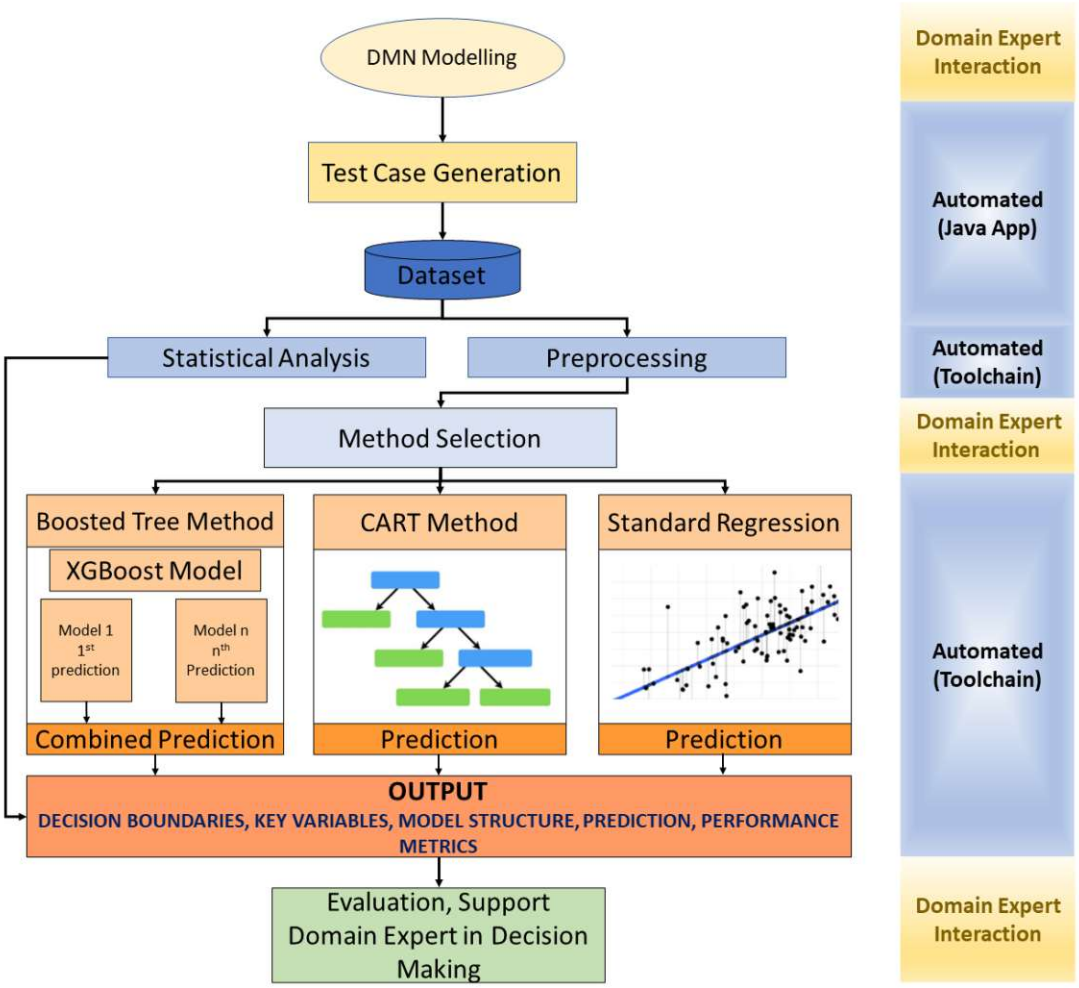

Figure 4.1: The Pipeline of the Proposed Concept.

After the pipeline is triggered, first the statistical analysis is carried out on the raw, unmodified dataset. Parts of this stage include firstly the normality test of the response variable using Kolmogorov-Smirnov (K-S) normality test (alternatively Shapiro-Wilk normality test could also be considered, however Shapiro-Wilk test is less recommended for larger datasets) with the significance level of $\alpha = 0.05$. Second, aggregated statistics (min, max, range, quantiles, standard deviation) of the dataset is calculated and exported in a tabular form for the user. Lastly, various graphical representations (correlation plots, visualizations of pairwise relationships, density graphs and boxplots) of the underlying dataset are also exported and saved for the users. There is a strong emphasis on visual representations of the underlying information in the data. As in the case of many variables and generally complex data, visual representations are more convenient for the users, easier to understand and the information is more effectively communicated. Functional

relationships are hard to find without the knowledge of the type (linear, polynomial, trigonometric etc.) of function to be looking for, however pairwise graphs, pairwaise scatter plots, correlation matrices can give clues for the user for their further investigations, e.g. visually analysing the pairwise plots, one can identify or make assumptions on the presence of more complex functional relationships in the data, which then later can be investigated computationally. This task however would not be easy to automate without the knowledge of the functional form of the relationship. Outputs of the statistical analysis stage are saved in forms of natural text and .png images.

After the analysis stage, the input data is prepared for the machine learning algorithms with the help of the preprocessing job. This stage handles missing data by simply removing them. Alternatively different data imputation strategies could also be implemented, but these strategies are mostly useful in the case of low number of observations. Furthermore, as the data comes from querying an already existing DMN model, missing data is not expected to be present. Moreover, most machine learning algorithms are relying on the fact that the various columns of the input data are of the same datatype. The conversion of the dataset to obtain consistent datatypes among the columns is also achieved in the preprocessing stage.

Finally, various machine learning models (linear models such as Lasso and Ridge Regression, CARTs, XGBoost) are trained in series and are also evaluated on multiple metrics (RMSE, MAE, MAPE). As a default setting, 5-fold crossvalidation is also performed.

From Fig. 4.1 the output of the toolchain is designed to provide insightful information for the user about the statistics of the used data in the form of both graphical and tabular representation. Furthermore the toolchain provides the results from the above mentioned test phase, which can serve as a meaningful baseline for the user to compare the different ML methods, furthermore conclusion about their applicability can also be drawn.

4.1.2 Data Definition

In order to train the ML models offered by the toolchain and make use of all its capabilities, it is necessary to collect sufficient data in the defined data structure. Tabular data with .csv format or plain text is required to use the toolchain. The default setting of the separator is the comma. The change of the separator needs modification in the code. The data types of the columns can be negligible, because the built in on-hot-label encoder fits the data in the required data type for the ML algorithms. The data from the perspective of the source can be a field data or synthetic data, respectively. Generation of synthetic data is a cheap way to increase the size of an already existing smaller dataset (which in itself would not yield sufficient results), while ensuring that the statistical properties of the original dataset remains. Generation of synthetic data, can either happen outside of the toolchain, or alternatively be implemented in the toolchain's preprocessing stage. If a certain task requires, further data processing methods can be included in the toolchain, such as the SMOTE (Synthetic Minority Oversampling Technique) in the case of an

imbalanced dataset. SMOTE is an easy to implement technique to balance out biased datasets.

In some cases raw field data might not be available, in such cases one can generate the required dataset by querying an existing DMN model with various combinations of inputs and save both the input and query results in a .csv file, which later can be used as the input data for the toolchain. Depending on the purpose and requirements from the toolchain, one should pay attention on the completeness of the query parameters, meaning is it required to use all input variables of the original DMN model or not. It is usually not necessary to have a data which is the dot product (all possible input parameter combinations present) for an ML algorithm to give usable results, however too little variety in the data can lead to biased and under-performing results. One must not forget that an ML algorithm can only provide as much information as the input data contains.

4.1.3 Specification of the ML Algorithms

In this section the two distinguished ML models offered by the toolchain are reviewed, and a brief insight is given into their specifications and considerations for model selection. One particular implementation is provided in the toolchain for the below mentioned models. A critical aspect may be the parameter tuning of the methods. The toolchain can be extended with grid search and Genetic Algorithm-based methods if necessary.

Decision Trees. Decision Trees hold many advantages such as easy understanding of the obtained model and analysis results. Beyond their great capability of precise model fitting Decision Trees provide easily accessibility and readability for the users to the mapped relations. Furthermore, this technique adds robustness with respect to the outliers and missing values, which is a great advantage for data mining tasks. However, there are a few disadvantages of the application of the Decision Trees. It is known, that a high-level node can be modified by including new input attributes. Overall change of the tree structure can be challenging and may have significant effect on the prediction. In addition, the parameter setting may also become complex, requiring further parameter tuning and / or optimization methods. The selection of this model is proposed when the attributes are fixed and does not change along the data. It is important to note that as it is possible to derive decision tables from decision trees, the above mentioned drawbacks also impact decision tables, which are the main building blocks of DMN models. Furthermore it is also important to note, that decision tables are usually used when modeling simple logic, with a low number of variables. In case of modeling a complex problem with a variety of input parameters, decision tables would come short in both performance and explainability.

Boosted Tree Method. In the ML model the term Boosting refers to the method of combining multiple so called weak learner algorithms (in most cases tree based methods) to produce a single strong learner. Boosting iteratively adjusts the weights of the different weak learners in the model to better fit the provided training data. Gradient Boosting is

a more advanced method, which instead of modifying the predictor weights, it tries to generate a new predictor based on the errors the previous predictor made. XGBoost is based on the idea of gradient boosting with enhanced performance, speed and scalability. Generally Boosting methods provide an easy to read model thus tracking back the predictions is easy. It is an efficient and resilient method, which handles overfitting well. However model convergence becomes harder if outliers are present in the training data. Also the algorithm is difficult to scale.

4.1.4 Interpretation

The Toolchain saves all trained models in a dedicated folder to avoid unnecessary re-training, furthermore it also writes the results from the model evaluation part into a text file in a tabular format. An overview of the output files of the program can be seen in Fig. 4.2. The operation and output of the Toolchain is detailed in the following chapter through a case study.

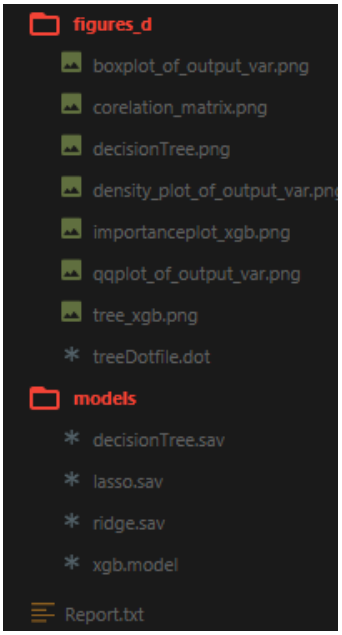

Figure 4.2: Screenshot of outputs of the Toolchain in the docker container.

CHAPTER 5

Evaluation

A JAVA application is used to generate the dataset using the imported DMN model by querying it with unique input variable configurations. The target is to predict the response variable named Prämie by using the proposed Toolchain with the help of the explanatory variables included in the car insurance dataset. The dataset consists of six explanatory variables (three logical and three numerical), and one response variable with a total of 12 000 observations. Later a larger dataset consisting of 12 000 000 observations was also generated and the results from that dataset will also be presented. As the DMN model, which was used to generate the dataset is currently deployed and used in production, due to confidentiality reasons, the applied variable names are anonymized. The investigation covers the comparative analysis of common regression models and the latest ensemble methods, such as XGBoost [Chen and Guestrin]. For fitting the ML models, the standard 80/20 train-test split is used with a fixed seed to ensure reproducibility, and the same split is used to train all the models.

In order to fit linear regression models, the data must show the following characteristics:

- The residuals follow a normal distribution with expected value 0.
- The independent variables are not strongly collinear (not correlated with each other).
- The model should have linear parameters. (Fulfilled)
- The data should be a random sample from the population. (Fulfilled)

In the next subsection I will analyze the data and see whether the first two points of the above list holds.

5.1 Case Study Setup Description

Fist of all, let's take a look at the structure of the original DMN model, which is the starting point of this thesis' work. The DMN model that serves as the Ground Truth for this case study is a model from a car insurance company. This model is used for calculating the premium one must pay for car insurance. Due to privacy restrictions the model and the data itself needed to be anonymized.

The original model is depicted in **Fig. 5.1**, as it is visible from the diagram, the model itself is rather simple, **Decision1_1 - Decision1_3** hold simple logic behind calculating factors, that are then used in **Decision1** in the context of a simple linear model. The original DMN model produces the final decision using three sub-decisions (1_1, 1_2, 1_3). All the sub-decisions have the same two inputs. The sub-decisions determine the weights an/or scores that play a key role in the linear model used for the final decision **Decision1**.

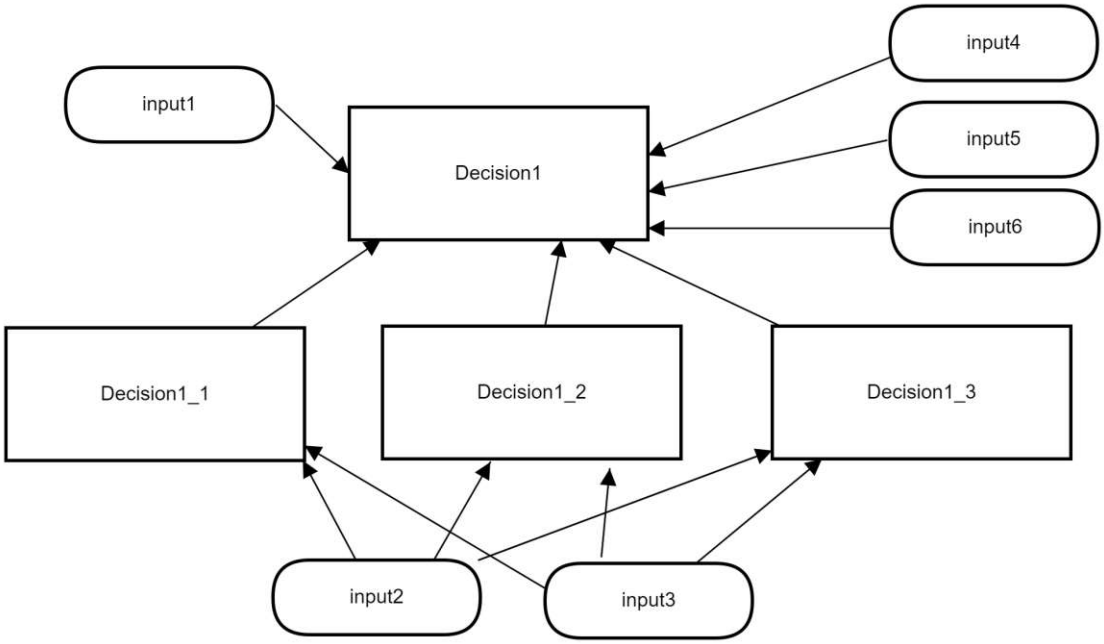

Figure 5.1: Decision Diagram of the Ground Truth model. Input variables and decision logic are anonymized.

5.2 Pre-analysis

In this section I investigate the results of the Statistical Analysis stage from the Toolchain. With the help of the Toolchain some characteristics of the data is uncovered in this subsection. First, I examine the distribution of the variables. The distribution of the response variable is depicted in Fig. 5.2.

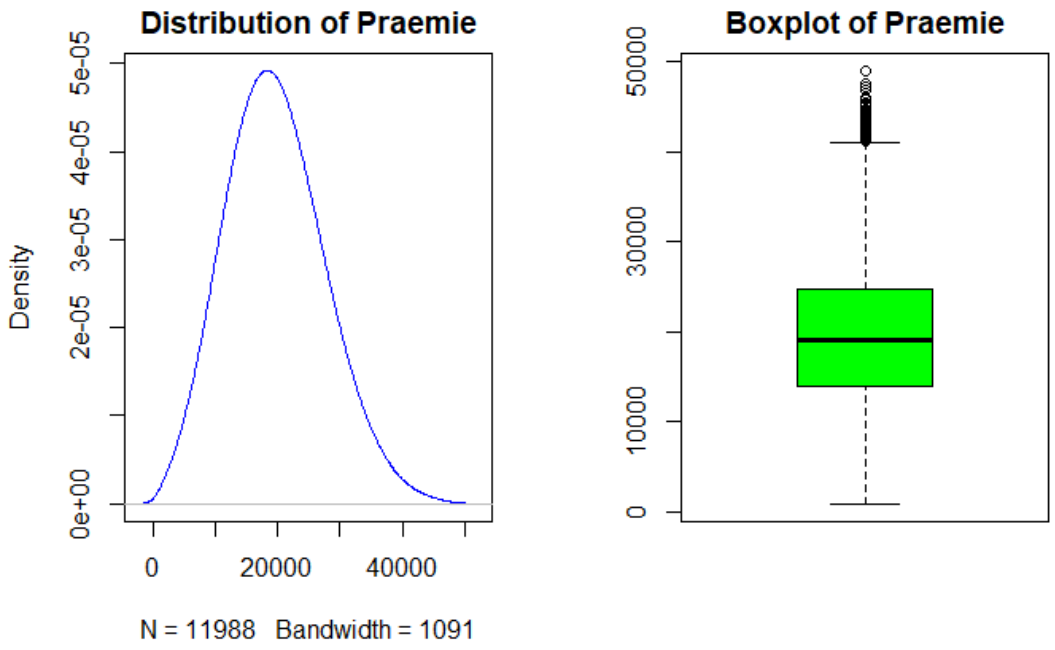

Figure 5.2: The distribution and the boxplot of the response variable Prämie. The range of the variable is 900 - 49000

As the variable is uncovered with the help of Shapiro-Wilk test, i.e. its distribution follows the Gaussian distribution with a P-value of 0.21. Thus, we cannot reject the hypothesis H_0 which states that the variable follows a Gaussian distribution given $\alpha = 0.05$. Figure 5.2 shows the density range and distribution of the response variable Prämie. Thus, the first requirement of fitting linear model is fulfilled.

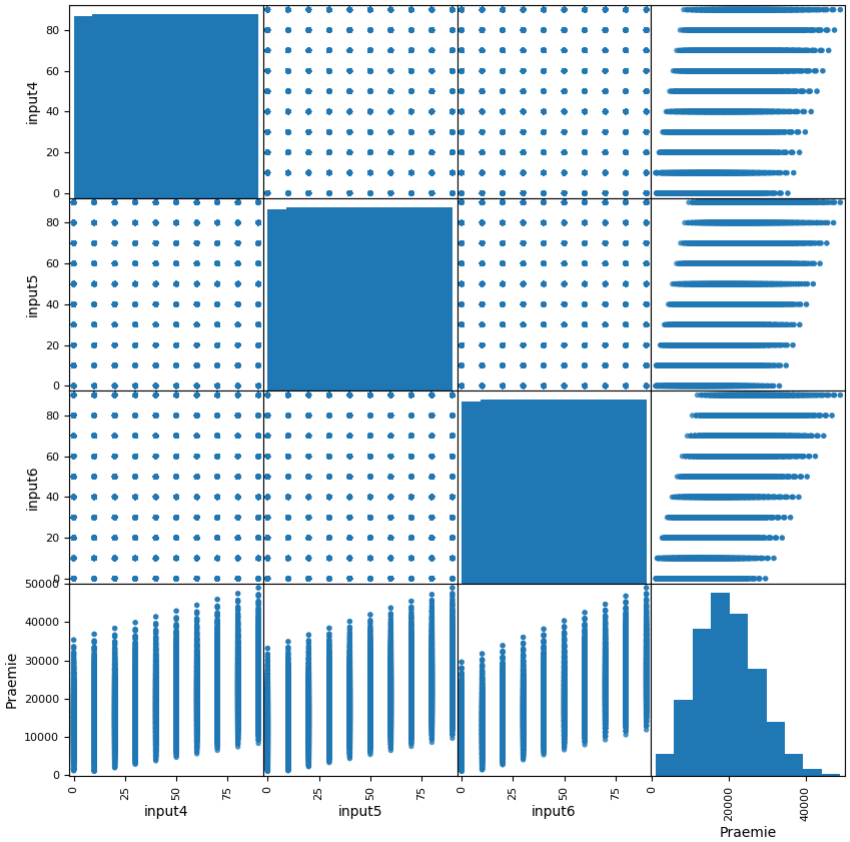

Figure 5.3: Relationships between all pairs of variables

It is also important to check whether functional relationships between the variables can be found or not. Automation of the identification of such information is very hard without the knowledge of the function that one would want to look for, thus based on manual assessment of the pairplots in Fig. 5.3 we can conclude, that there is a linear relationship between all input variables (Input 4-6) and the response variable pairwise, and there is no visible functional relationship between any two explanatory variables.

Next, the investigation of the correlation between the explanatory variables are carried out. Here, the Pearson correlation coefficient is used, which measures the linear association between two quantities. It is a common misconception that zero correlation does not mean association. We clarify that correlation strictly measures the linear relationship between two variables. The correlation coefficient of nonlinear associations can be zero

or close to zero, which means that the value of the Pearson correlation coefficient of variables with high associations is not necessarily high. Let be two probability variables X and Y. Mathematically, if cov(XY) is the covariance between X and Y and σX is the standard deviation of X and σY is the standard deviation of Y, then the Pearson correlation coefficient p can be written as follows:

$$p(X,Y) = \frac{cov(X,Y)}{\sigma X \sigma Y} \tag{5.1}$$

For providing better understanding of the results, it is visualized in a correlation plot, where the colour gradient changes based on the value of the correlation in the correlation matrix.

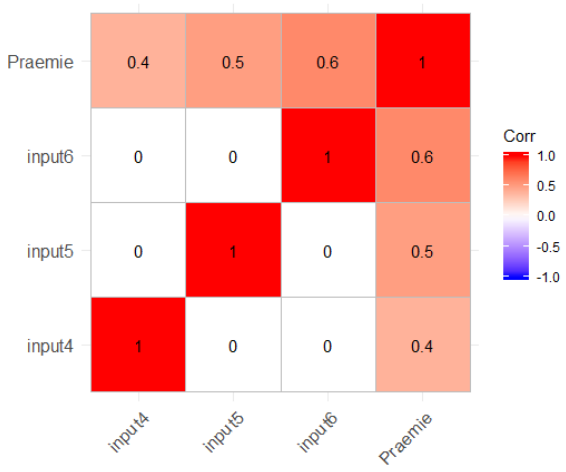

Figure 5.4: Visualization of the correlation matrix

In the correlation analysis (see, Fig. 5.4) only the numeric variables are present. It is clearly visible from the correlation matrix, that there is no linear correlation between the explanatory variables, only between the response and each numeric explanatory feature. Thus, the second requirement of fitting linear regression models is also fulfilled. The third requirement is fulfilled, since linear models are fitted. The fourth requirement can be considered fulfilled, since it is assumed that the original field data is randomly sampled.

5.3 Interpretation of results in the frame of the Research Questions

This section presents the evaluation of results for the research questions of this thesis, as follows. First, the Thesis seeks whether *the Toolchain is able to provide meaningful decision boundaries, i.e. automating the generation of a model, which is suitable to be used*

in a DMN environment (**RQ1**). Second, the statistical validity is also crucial, therefore the second research questions aims to *reveal the key variables*(**RQ2**). This research may be further decomposed into sub-questions, such as is; Is there correlation between variables during the pre-analysis of the field data (**RQ2.1**); What is the explaining power and relevance in the models (**RQ2.2**)? The third research question seeks *the best fitting model* and also raises the issue of substitutability of ML-based models for the traditional decision table in a DMN decision process (**RQ3**). Finally, the last research question (**RQ4**) is in connection with the performance analysis, i.e. *how does the running time of the Toolchain scale with the data*?

Based on the above research questions, I make an attempt to justify the use of the proposed Toolchain (Fig. 4.1), compare the performance of various ML algorithms and also validate the correctness of the method and its usability for business stakeholders in a real-world setting. In order to answer these research questions the results of the case study are discussed here.

Answers to RQ1
Figures 5.5-5.6 display the obtained tree structures, i.e. the decision boundaries for both the CART tree and one of the XGBoost trees. The XGBoost tree shown in the Fig. 5.5 is the tree, which holds the highest explaining power in the XGBoost model. It can be concluded, that the standard Decision Tree is far simpler, which explains the significant difference between their performance. All the preseneted results are generated automated by the Toolchain. It can be concluded the the automated model generation approach is able to find the appropriate and suitable model in a DMN environment.
It is also worth noting that the depth of the decision tree is comparable to the original DMN model shown in Fig. 5.1. It can be argued that standard regression models performed better than decision and regression trees thanks to the linear nature of the original model.

Answers to RQ2.1-2.2.
As it can be seen from the correlation matrix Fig.5.4, there is no correlation between explanatory variables, however strong positive correlation can be found between the response and each explanatory variables. This means that there seems to be no relationship between the explanatory variables, and all explanatory variables influence the response variable. The convergence properties should be analyzed, which is dependent on the data. In the presented case study, the explaining power and relevance is confirmed according to the following results. Figure 5.7 shows the error with respect to the tree size. The exponential decrease indicates that the growing tree results in performance improvement until a certain limit, at which we can stop the growing process. The process does not show any fluctuations, thus it confirms that the algorithm converges well.

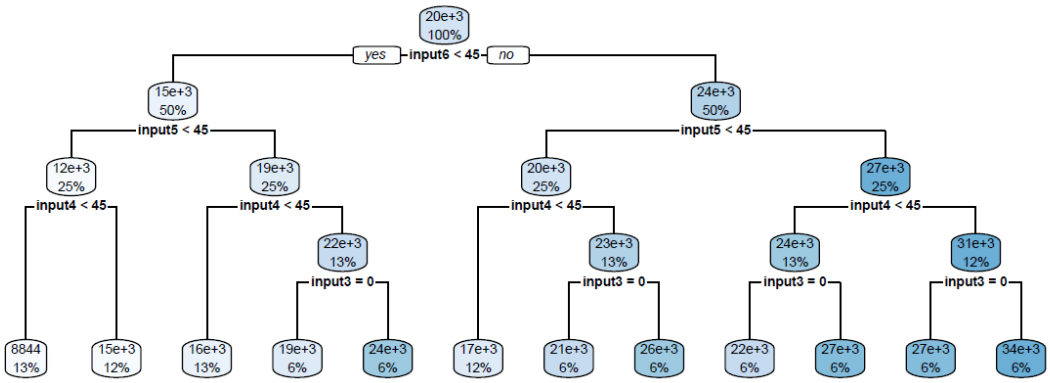

Figure 5.5: Obtained Decision Tree structure.

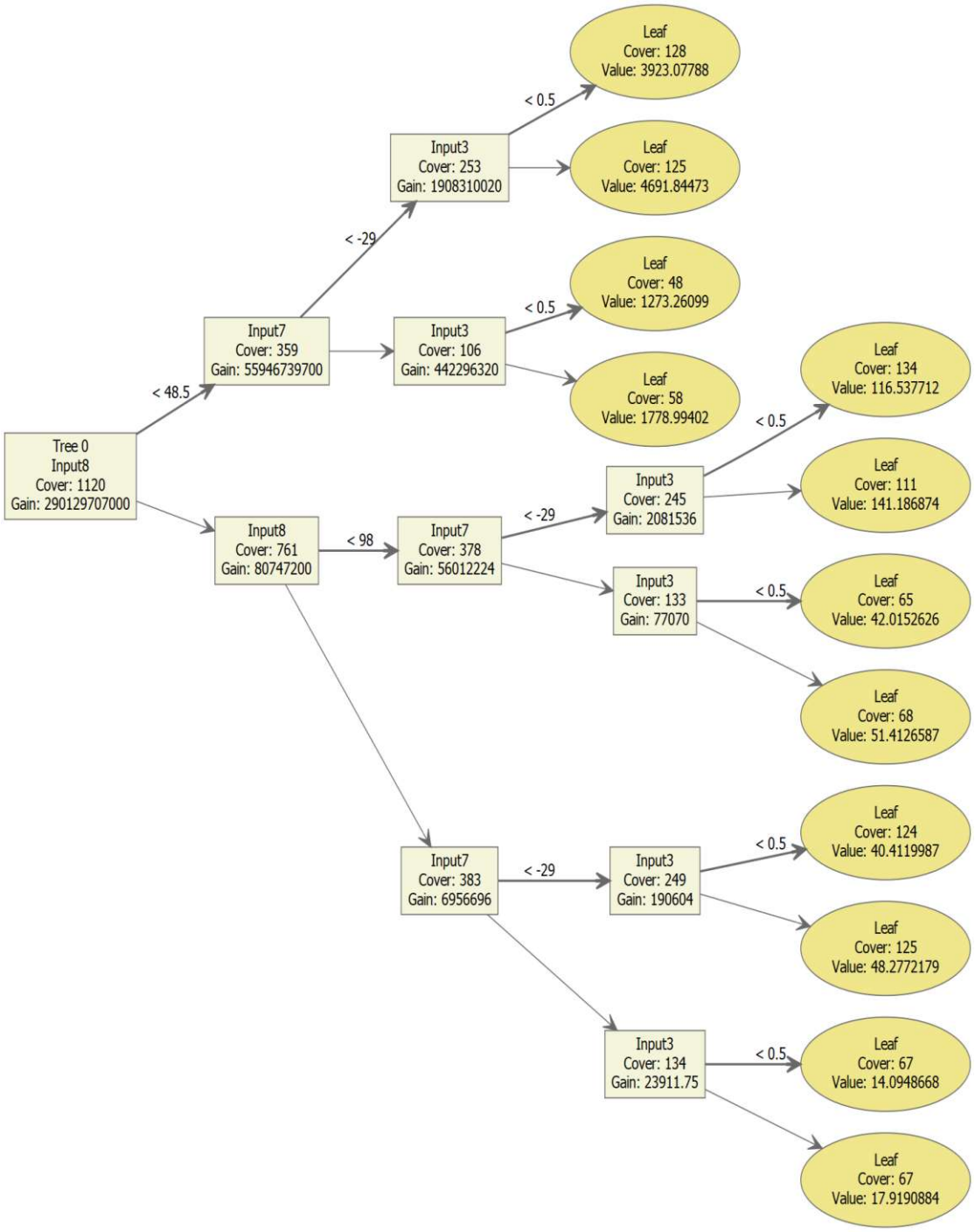

Figure 5.6: Obtained XGBoost Tree structure.

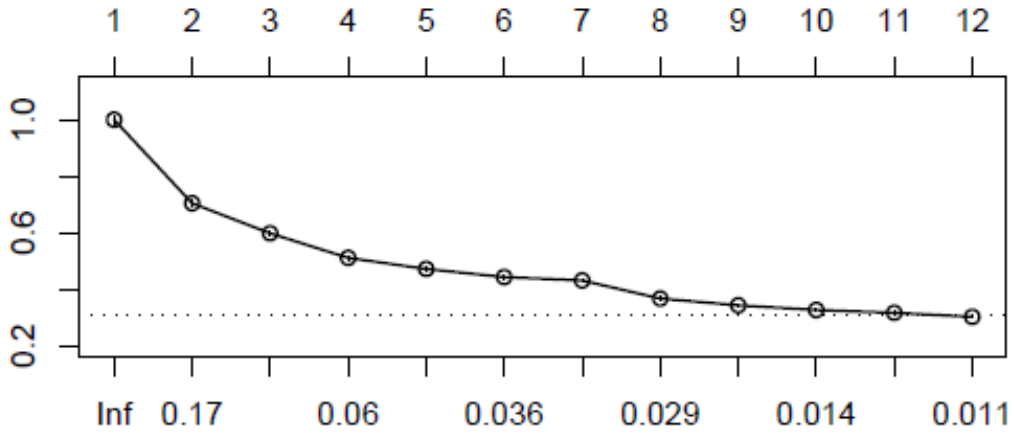

Figure 5.7: Error with respect to the size of trees. [Kluza et al.]

Answers to RQ3
In the above presented case study, without domain knowledge, the appropriate model is selected by comparing all the performances from Table 5.1. According to the study, the lowest RMSE, MAE and MAPE metrics are achieved by using an **XGBoost** model on all six input variables (see, Table 5.1). The functional relationship is found to be linear between the explanatory and response variables in our test data, which coincides with the structure of the original DMN model, and the distribution of the response variable is Gaussian. Between the explanatory variables no functional relationship is found in Fig. 5.2.
The results indicate the possible substitutability of decision tables in the DMN process with ML models, since the appropriate model is found. However, at the current readiness level of the proposed solution Toolchain, the users' itervention in model selection based on the statistical results is advised.

The output file of the Toolchain contains the necessary metrics. In the presented case study the best model is resulted in RMSE=4.3 MAE 3.3 MAPE=0.0003. The builtin visualizations and the clearly defined metrics may support most of the domain experts. In case of special needs, new metrics and visualizations could be easily integrated into the Toolchain, for the convenience of the user.

5.4 Experimental results

In this Section I present the experimental performance results of the Machine Learning models employed in the Toolchain. Furthermore, the runtime analysis on the Toolchain is performed. This section details also the answer to **RQ4**, that is experimentally validated, that the runtime of the Toolchain scales linearly with the length of the data, as follows.

Table 5.1 collects the performance results. It is observable, that RMSE is highest in case of Regression tree method, and also the other two metrics are one order of magnitude larger than the others. Regarding the results of the regression models, we can see that the regularized regression methods performed similarly and consistently worse, than classical linear regression among all metrics. It can be concluded, that all present variables are important to explain the response variable. Furthermore, the possible overfit of the classical regression model is negligible. In addition, it is clear that the **XGBoost** outperforms all the other models by multiple orders of magnitude across all analyzed metrics.

Performance Results			
ML Algorithm	RMSE	MAE	MAPE
Linear Regression	1129.016	780.5236	0.067
Ridge Regression	1471.782	1122.426	0.078
Lasso Regression	1413.217	1083.574	0.077
Regression Tree	4400.957	3567.689	0.25
XGBoost	**4.299**	**3.261**	**0.0003**

Table 5.1: Table of performance metrics of the models evaluated on the 20% unseen data

Furthermore the same modeling experiment is ran on the Min-Max scaled version of the dataset, the results are visible in Table 5.2. Scaling the data into the range between 0 and 1 yields the scale variant performance metrics to be also in the same range, this helps the intuitive understanding of the performance results. Moreover it is beneficial especially in the case of regression models to bring the input variables in the same range, as otherwise columns with larger values will influence the prediction results more than inputs with lower values.

Performance Results			
ML Algorithm	RMSE	MAE	MAPE
Linear Regression	0.029294	0.022389	7.997728
Ridge Regression	0.163039	0.132072	56.998929
Lasso Regression	0.029306	0.022395	7.998303
Regression Tree	0.074205	0.059407	19.468455
XGBoost	**0.001060**	**0.000834**	**0.274443**

Table 5.2: Table of performance metrics of the models evaluated on the 20% unseen normalized data

From table 5.2 we can clearly see that Regression Models benefited from normalization and with the exception of Ridge Regression, also performed better than the Regression Tree method. XGB continues to significantly outperform all other models across all performance metrics.

Answers to RQ4

As the proposed Toolchain is a data driven solution, it is always best practice to identify how the software scales with the data both horizontally and vertically. The selected data for the running time analysis is a publicly available dataset from Kaggle, with 22 columns and 44000 rows [Rajarshi].

This data is split into smaller sets with varying number of columns and rows, and the Toolchain is used on all the produced subsets of the original data. It is important to note, that in this analysis I am only interested in the execution and modeling times of the designed software, and not the actual model performance metrics. The running times that are of interest are the time it takes to build the models and the full running time of the program. The graphs displayed in Figs. 5.8-5.9 visually represent the identified processing times with respect to the horizontal and vertical sizes of the data. Figure 5.8 shows the model building time (upper chart) and Toolchain running times with respect to the number of columns by executing on dataset of 2980 rows (blue line) and 44000 rows (red line). While, Fig. 5.9 displays the time for model building (upper chart) and the Toolchain running time (lower chart) with respect to the number of rows with fixed number of columns of 22.

Even though only a few number of tests were conducted a general conclusion of the scalability can be drawn, namely the modeling part of the Toolchain seems to scale logarithmically with the number of rows of the dataset, while the Toolchain itself is linearly scaling. When it comes to identifying a relationship between the running times and the number of columns present in the dataset, the Toolchain seems to have a linear relationship, while the modeling part did not give a reasonable response. Interestingly the running time of the model building part of the program seemed to decrease with the number of columns present in the data, which may allow to suppose that richer data may speed up the convergence. However, this point may need further investigations.

For the sake of comparison, I have tested the performance on a big data set of 7 Columns and 12 000 000 rows. The big data set is the original full data set of the case study with the same variables as described in the case study. The results can bee seen in Table 5.3.

Performance Results on Big Data			
ML Algorithm	RMSE	MAE	MAPE
Lasso Regression	1484.712	1148.35	6.682
Ridge Regression	1484.712	1148.54	6.687
Regression Tree	3647.138	2918.33	15.99
XGBoost	**110.842**	**88.113**	**0.496**

Table 5.3: Table of performance metrics of the models evaluated on the big data

The total execution time resulted in 15032.0s ~ 4.17h for the set. The performance results indicate that the Regression and CART models perform similarly compared to the previous results obtained on the smaller subset. Comparing the RMSE obtained of the **XGBoost** model of 4.3 in case of the small set and the RMSE value of 110.0 obtained at

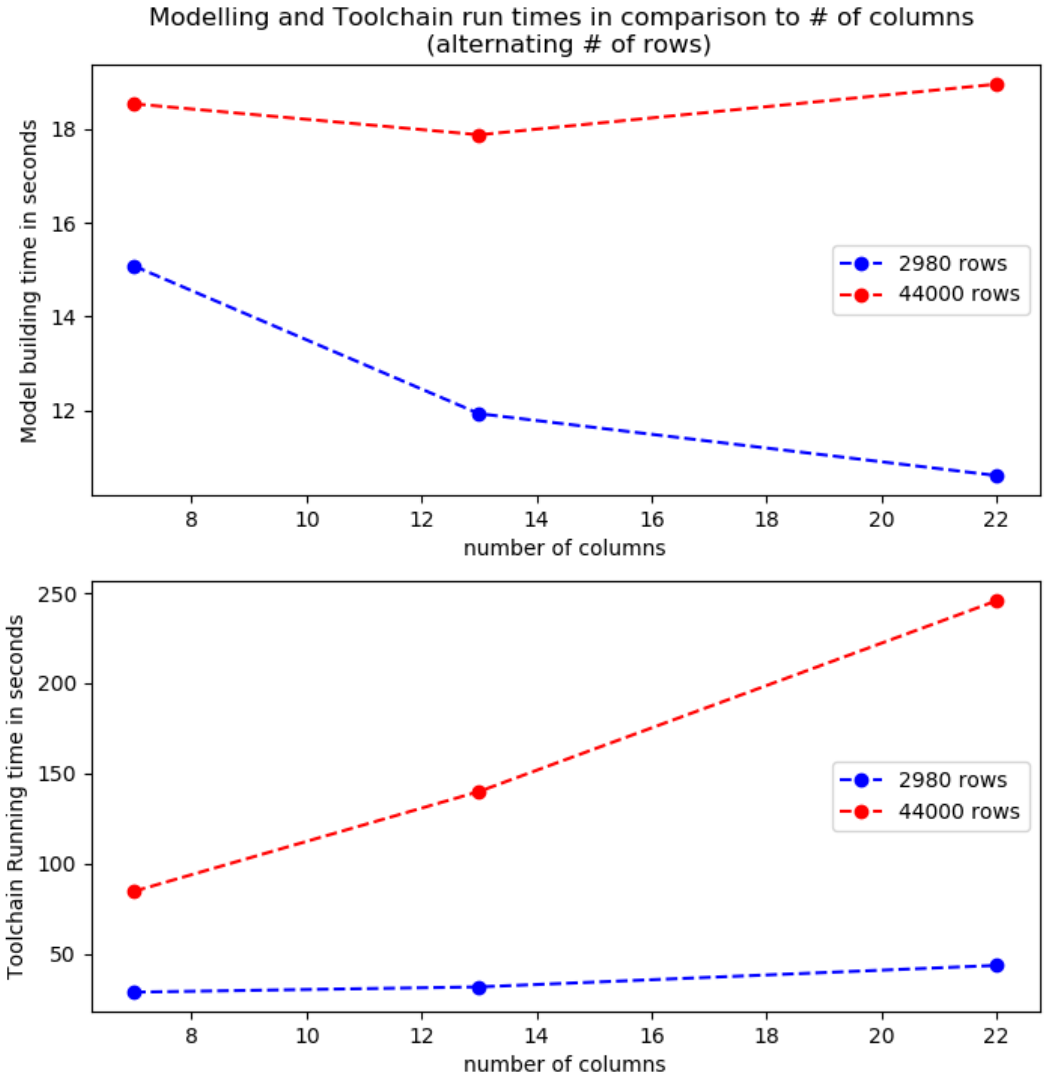

Figure 5.8: Running times of both model building and Toolchain with respect to different number of columns in the dataset

the big data suggests, that the **XGBoost** model clearly suffers from overfitting, however it still outperforms all other models across all metrics.

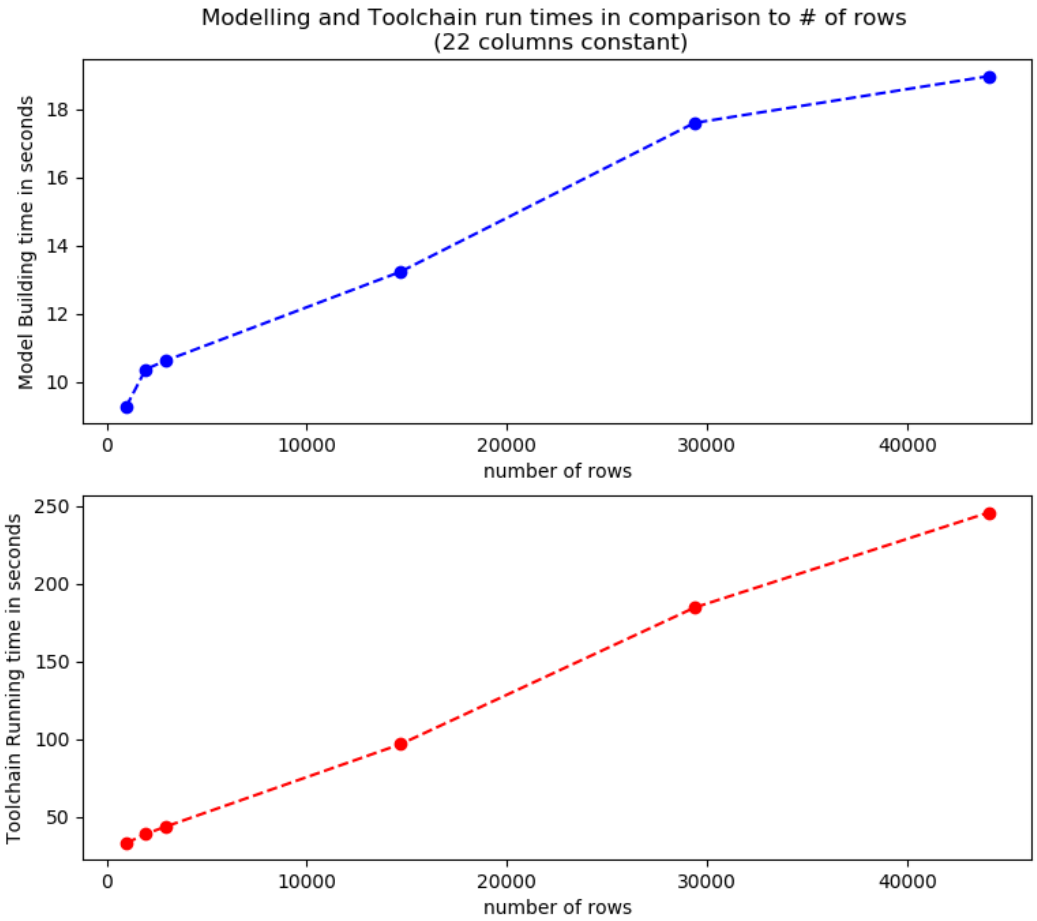

Figure 5.9: Running times of both model building and Toolchain with respect to different number of rows in the dataset

CHAPTER 6

Discussion

Practical solutions of decision modeling issues may involve model-based techniques, which offer a very challenging way to integrate the a priori knowledge into the procedure. According to the state-of-the art, the Decision Model Notation approach, inspite of its popularity and advantages, there is still room for improvement. For instance, it has been proven in the literature that DMN is not efficient in case of non-deterministic processes (e.g., in case of medical treatment decision processes) and provides low performance in case of unseen or unfamiliar data. The state-of-the art clearly shows, that the method needs improvement in order to enhance its robustness and flexibility, which inspired my investigations. Due to the rapid development of Machine Learning and Big Data technologies, it has become obvious to choose methods from these fields to improve the DMN method. On the other hand, based on some research findings, the question also arises whether the DMN methodology could be replaceable with ensemble ML techniques, which could provide the same high performance.

DMN's key advantage is that it can be easily integrated in most of the business processes. Despite, the application of ML tools in business process is not straightforward, since there is no framework for this purpose. In addition, building such a model requires experienced users and needs to be fitted to each specific application example. Therefore, there is a real need to design software tools that compensate for these shortcomings, such as the proposed Toolchain, which may help users to integrate ML in a DMN context, complement the abilities and also compensate the deficiencies of DMN. The proposed application areas for DMN combined with ML tools could be especially beneficial for areas with high uncertainty, such as medical decision processes - in harmony with literature. However, in case of car insurance sector the categorization based on probability (user group similarity) may include uncertainty where the inclusion of ML techniques would also be beneficial.

Results of the investigations support, that the proposed method has fulfilled the set goal of this thesis and provides sufficient performance. As it was mentioned above the result

is highly dependent of the available data. The investigations underline that linear models can be valuable baselines. In addition, **XGBoost** outperformed the expectations and it is clear that **XGBoost** is a candidate for further investigations.

Overfitting is usually undesirable in most ML applications and various methods are proposed for avoiding it. In this study the range of the variables are given. This indicates that overfitting may improve the performance. Therefore, in this investigation I did not make significant efforts to overcome overfitting by exploring the learning evolution, the only regularization for model parameters came from the implementation of the algorithms. It is plausible that some limitations could have influenced the results obtained. First, the appropriate hardware is required to obtain the desired speed and performance. Another source of error could be the bad quality raw input data. A major source of unreliability may emerge from the different and incompatible Python package versions that are used by the Toolchain. The *requirements.txt* file attached to the Thesis, it lists the used packages and their versions to overcome the previously mentioned issue. In addition, since Python is an interpreted language, the execution time becomes slower than in the case of compiled languages. It is worth to consider translating the program to, for e.g. C++, or alternatively to use Cython.

Possible threats to validity need to be mentioned and discussed. According to [Wohlin et al., Fleck et al.] four essential types of threats may affect the validity of the conducted study. Conclusion validity covers the relationship between the treatment and the outcome. The results of the modeling are valid, due to their reproducibility. In case of a linear model the same input results in the same outcome due to a simple matrix multiplication operation, which is the source of the model. However, some parts of the models are deterministic in nature, but some sampling or splitting operations are stochastic (for e.g., train and test set splitting operations). Therefore the reproducibility can be ensured by fixing the seed state for initializing the random generator object, see Appendix A. Since the problem formulation presented in this thesis does not include any higher-order theoretical concepts, the threats to Construct validity is not applicable here. Threats of External Validity may affect the presented results. As it can be seen from results of investigation on big data 5.3 and subset of our data collected in Table 5.1 the performance of XGBoost degraded on big data. The causal relationship observed and proved in this thesis needs further investigation in order to be transferable to a larger sample or population. Internal validity is concerned with the causal effects between the independent and dependent variables. The proposed Toolchain includes exhausting statistical analysis, e.g. the correlation matrix can reveal the necessary information to provide the internal validity.

CHAPTER 7

Conclusions and Future Work

In this thesis efforts were made to design a Toolchain for enhancing the performance of DMN-based solutions. Furthermore, the aim is to bridge the gap between business stakeholders and data scientists.

Based on an exhaustive literature review, the method proposed in the Thesis has been placed among the development directions of the field. According to the results of critical analysis of DMN methodology, the main design aspects of the Toolchain are defined. Built upon a briefly summarized foundations and recent advances of Machine Learning techniques, three modeling approaches are built in the Toolchain, covering linear models, CART models and the ensemble-type **XGBoost** technique. The performance of the Toolchain is investigated through a case study based on insurance data set. The investigations cover performance analysis on a smaller set and on big data. The results confirm, that in both cases the proposed method ensures high performance. The proposed Toolchain application supports also the comparative analyses of the performance of the ML models integrated in a business process. This may compensate for the deficiencies of the DMN standard, since it is rather intuitive and may contain several overlapping or inefficient decision rules due to the manual creation of decision boundaries. In contrast to that, the CART model is able to find the optimal structure, thus may also help further enhance a DMN methodology. So the user can get a relatively fast feedback on the data, i.e. the key-variables, response variables, etc. In addition the user can quickly produce ML models and gain information about their performances with the ability to import the created models into a DMN context with the help of **sklearn2pmml** Python package. Furthermore, the application enables the user to manually compare the decision rules obtained by the ML model and the DMN model's decision rules with the purpose of further improvement or possible extensions of the DMN model.

The obtained **XGBoost** model due to its outstanding performance may be applied in a DMN standard instead of, e.g. a decision table. The automated generation of such

methods (e.g., when data source is updated) may contribute the automation of the whole decision-making with DMN.

This thesis is the first step towards enhancing our understanding of the possible combination of DMN standard and Machine Learning. This work has revealed, that ML models are suitable for integrating in business processes and these early results may contribute to tackle some deficiencies of the DMN methodology. However, the results and investigations revealed some issues in need for further research. It is recommended to undertake further examinations in the following areas.

The empirical/experiment-based models in DMN-based decision process do not include any optimizations or tuning, but instead might contain great redundancies. It can be suggested to carry out exhaustive investigation on the applicable optimization methods and parameter tuning of the models. Furthermore, future work should concentrate on the performance evaluation in case of datasets with various size or characteristics.

Human reasoning or decisions rules may include fuzzy components and results are not obtained on a wide operation range (test set). Future work may include also the analysis of further applicable models, for e.g., models from the field of computational intelligence, fuzzy approaches.

The results presented in this study indicate, that the **XGBoost** ensemble technique is highly suitable for approximating the human-constructed decisions. However, the human decision structure is far simpler than the **XGBoost**'s set of multiple trees. The exploration of the effects of the added complexity coming from the models built by ML on the performance could also be the scope of the future research.

Bibliography

Dmn, meet machine learning. https://www.trisotech.com/dmn-meet-machine-learning/. Accessed: 2021-02-01.

H. M. Abdelsalam, A. R. Shoaeb, and M. M. Elassal. Enhancing decision model notation (dmn) for better use in business analytics (ba). In *Proceedings of the 10th International Conference on Informatics and Systems*, page 321–322, 2016.

M. Azad, I. Chikalov, and M. Moshkov. Representation of knowledge by decision trees for decision tables with multiple decisions. *Procedia Computer Science*, 176:653 – 659, 2020. ISSN 1877-0509. doi: https://doi.org/10.1016/j.procs.2020.09.037. URL http://www.sciencedirect.com/science/article/pii/S1877050920319323. Knowledge-Based and Intelligent Information Engineering Systems: Proceedings of the 24th International Conference KES2020.

E. Bazhenova, F. Zerbato, B. Oliboni, and M. Weske. From bpmn process models to dmn decision models. *Information Systems*, 83:69–88, 2019.

Y. Biana, C. Yang, L. Zhaoc, and L. Lianga. Good drivers pay less: A study of usage-based vehicle insurance models. *Transportation Research Part A*, 107:20–34, 2018.

T. Biard, A. L. Mauff, M. Bigand, and J.-P. Bourey. *Separation of Decision Modeling from Business Process Modeling Using New "Decision Model and Notation" (DMN) for Automating Operational Decision-Making*, volume 463 of *IFIP Advances in Information and Communication Technology (IFIPAICT)*, chapter Risks and Resilience of Collaborative Networks. PRO-VE 2015., pages 489–496. Springer, 2015.

S. Blows, R. Q. Ivers, J. Connor, S. Ameratunga, and R. Norton. Car insurance and the risk of car crash injury. *Accident Analysis and Prevention*, 35(6):987–990, 2003.

B. Boonmepipit and T. Suwannasart. Test case generation from bpmn with dmn. ICSEB 2019, page 92–96, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450376495. URL https://doi.org/10.1145/3374549.3374582.

D. Bork. Conceptual modeling and artificial intelligence: Mutual benefits from complementary worlds. *arXiv preprint arXiv:2110.08637*, 2021.

D. Bork, D. Karagiannis, and B. Pittl. Systematic analysis and evaluation of visual conceptual modeling language notations. In *12th International Conference on Research Challenges in Information Science (RCIS), Nantes*, pages 1–11, 2018.

D. Bork, R. Buchmann, D. Karagiannis, M. Lee, and E.-T. Miron. An open platform for modeling method conceptualization: The omilab digital ecosystem. *Communications of the Association for Information Systems*, 44:673–697, May 2019. ISSN 1529-3181. doi: 10.17705/1CAIS.04432. URL `http://eprints.cs.univie.ac.at/5462/`.

D. Bork, D. Karagiannis, and B. Pittl. A survey of modeling language specification techniques. *Information Systems*, 87:101425, 2020. URL `https://model-engineering.info/publications/papers/BorkEtal20-InfSys-SurveyModelingLanguageSpecificationTechniques.pdf`.

M. Brambilla, J. Cabot, and M. Wimmer. *Model-Driven Software Engineering in Practice*. Morgan Claypool, 2012.

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

A. Caetano, C. Pereire, and P. Sousa. Generation of business process model views. In *CENTERIS 2012 - Conference on ENTERprise Information Systems / HCIST 2012 – International Conference on Health and Social Care Information Systems and Technologies, Procedia Technology, vol. 5*, pages 378–387, 2012.

D. Calvanese, M. Dumas, Ülari Laurson, F. M. Maggi, M. Montali, and I. Teinemaa. Semantics, analysis and simplification of dmn decision tables. *Information Systems*, 78:112 – 125, 2018. ISSN 0306-4379.

F. Catanzariti, G. Chiaselotti, F. G. Infusino, and G. Marino. Object similarity measures and pawlak's indiscernibility on decision tables. *Information Sciences*, 539:104–135, 2020.

T. Chen and C. Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '16), Association for Computing Machinery, New York, NY, USA*, page 785–794, 2016.

G. Chiaselotti, T. Gentile, and F. G. Infusino. Dependency structures for decision tables. *International Journal of Approximate Reasoning*, 88:333–370, 2017. doi: 10.1016/j.ijar.2017.06.007.

G. Chiaselotti, F. Catanzariti, F. G. Infusino, and G. Marino. Object similarity measures and pawlak's indiscernibility on decision tables. *Information Sciences*, 539:104–135, 2020. doi: 10.1016/j.ins.2020.05.030.

M. de Leoni, M. Dumas, and L. García-Bañuelos. Discovering branching conditions from business process execution logs. In V. Cortellessa and D. Varró, editors, *Fundamental Approaches to Software Engineering*, pages 114–129, 2013.

M. Deryck, F. Hasic, J. Vanthienen, and J. Vennekens. A case-based inquiry into the decision model and notation (dmn) and the knowledge base (kb) paradigm. In *Rules and Reasoning, Second International Joint Conference, RuleML+RR 2018 Luxembourg, Luxembourg, September 18–21 Proceedings*, pages 239–263, 2018.

T. G. Dietterich. Ensemble methods in machine learning. In *Proceedings of the First International Workshop on Multiple Classifier Systems*, MCS '00, page 1–15, Berlin, Heidelberg, 2000. Springer-Verlag. ISBN 3540677046.

J. Dombi. Intelligent systems, lecture notes (in hungarian), Nov. 2012.

V. Etikala, Z. VanVeldhoven, and J. Vanthienen. Text2dec: Extracting decision dependencies from natural language text for automated dmn decision modelling. In *Business Process Management Workshops*, 2020.

D. Etinger, S. D. Simić, and L. Buljubašić. 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), Opatija, Croatia*, pages 1309–1313, 2019.

P. Fettke. 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 & Demo Papers*, pages 157–164, 2020.

K. Figl, J. Mendling, G. Tokdemir, and J. Vanthienen. What we know and what we do not know about dmn. *Enterprise Modelling and Information Systems Architectures*, 13 (2):1–16, 2018.

P. Filzmoser. Advanced methods for regression and classification, lecture notes, October 2018.

M. Fleck, J. Troya, M. Kessentini, M. Wimmer, and B. Alkhazi. Model transformation modularization as a many-objective optimization problem. *IEEE Transactions on Software Engineering*, 43(11):1009–1032, 2017. doi: 10.1109/TSE.2017.2654255.

L. Guelman. Gradient boosting trees for auto insurance loss cost modeling and prediction. *Expert Syst. Appl.*, 39:3659–3667, 02 2012. doi: https://doi.org/10.1016/j.eswa.2011.09.058.

V. Hodge, S. O'Keefe, and J. Austin. A binary neural decision table classifier. *Neurocomputing*, 69:1850–1859, 10 2006a. doi: 10.1016/j.neucom.2005.11.012.

V. J. Hodge, S. O'Keefe, and J. Austin. A binary neural decision table classifier. *Neurocomputing*, 69:1850–1859, 2006b.

F. Horita, D. Link, J. Albuquerque, and B. Hellingrath. odmn: An integrated model to connect decision-making needs to emerging data sources in disaster management. In *Proceedings of the 49th Hawaii International Conference on System Sciences (HICSS)*, page 2882–2891, 2016.

F. E. A. Horita, J. P. de Albuquerque, V. Marchezini, and E. M. Mendiondo. Bridging the gap between decision-making and emerging big data sources: An application of a model-based framework to disaster management in brazil. *Decision Support Systems*, 97:12–22, 2017.

T. Hutchinsqn and S. Rowell. Points systems for car insurance. *Insurance: Mathematics and Economics*, 5:255–259, 1986.

J. Huysmans, K. Dejaeger, C. Mues, J. Vanthienen, and B. Baesens. An empirical evaluation of the comprehensibility of decision table, tree and rule based predictive models. *Decision Support Systems*, 51(1):141 – 154, 2011. ISSN 0167-9236. doi: https://doi.org/10.1016/j.dss.2010.12.003. URL `http://www.sciencedirect.com/science/article/pii/S0167923610002368`.

L. Janssen, J. D. Smedt, and J. Vanthienen. Modeling and enacting enterprise decisions. In *in: Springer-Verlag; CAISE '16 International Conference on Advanced Information Systems Engineering, Date: 2016/06/13 - 2016/06/17, Location: Ljubljana (Slovenia) Lecture Notes in Business Information Processing (LNBIP), Vol. 249., pp. 169-180*, 2016.

K. Kluza, W. Adrian, P. Wiśniewski, and A. Ligęza. *Understanding Decision Model and Notation: DMN Research Directions and Trends*, pages 787–795. 08 2019. ISBN 978-3-030-29550-9. doi: 10.1007/978-3-030-29551-6_69.

R. Kohavi. The power of decision tables. In *Proceedings of the 8th European Conference on Machine Learning*, ECML'95, page 174–189, Berlin, Heidelberg, 1995. Springer-Verlag. ISBN 3540592865.

E. Lamine, R. Thabet, A. Sienou, D. Bork, F. Fontanili, and H. Pingaud. Bprim: An integrated framework for business process management and risk management. volume 117, page 103199, 2020. doi: 10.1016/j.compind.2020.103199. URL `https://model-engineering.info/publications/papers/COMIND2020-BPRIM-FinalAcceptedWithHeader.pdf`.

C. Lampasona, A. Trendowicz, M. Kläs, and J. Heidrich. Measurement-based software quality evaluation. In *Tagungsband des DASMA Software Metrik Kongresses (MetriKon 2009), Shaker*, pages 3–16, 2009. ISBN 3832286497.

R. Li, T. Yu, C. Zhou, and H. Li. A multi-threshold granulation model for incomplete decision tables. *Journal of Software*, 9, 07 2014. doi: 10.4304/jsw.9.7.1922-1929.

X. Li, X. Wang, B. Sun, Y. She, and L. Zhao. Three-way decision on information tables. *Information Sciences*, 545:25 – 43, 2021. ISSN 0020-0255. doi: https://doi.org/10.1016/j.ins.2020.07.064. URL http://www.sciencedirect.com/science/article/pii/S0020025520307490.

G. Liu, Z. Hua, and J. Zou. Local attribute reductions for decision tables. *Information Sciences*, 422, 09 2017a. doi: 10.1016/j.ins.2017.09.007.

S. Liu, X. Wang, M. Liu, and J. Zhu. Towards better analysis of machine learning models: A visual analytics perspective. *Visual Informatics*, 1(1):48–56, 2017b. ISSN 2468-502X. doi: https://doi.org/10.1016/j.visinf.2017.01.006. URL https://www.sciencedirect.com/science/article/pii/S2468502X17300086.

R. Lukyanenko, A. Castellanos, J. Parsons, M. C. Tremblay, and V. C. Storey. Using conceptual modeling to support machine learning. In C. Cappiello and M. Ruiz, editors, *Information Systems Engineering in Responsible Information Systems - CAiSE Forum 2019*, volume 350, pages 170–181. Springer, 2019.

W. Maass and V. C. Storey. Pairing conceptual modeling with machine learning. *Data & Knowledge Engineering*, 134:101909, 2021.

N. Mehdiyev, C. Houy, O. Gutermuth, L. Mayer, and P. Fettke. Explainable artificial intelligence (xai) supporting public administration processes – on the potential of xai in tax audit. In *Innovation Through Information Systems*, pages 413–428, 2021. doi: 10.1007/978-3-030-86790-4.

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

OMG. Business process model and notation v.2.0.2. Technical report, Object Manager Group, Milford, MA, USA, Jan. 2014.

OMG. Decision model and notation v.1.3. Technical report, Object Manager Group, Milford, MA, USA, Sept. 2020.

I. OpenRules. Rulelearner, 2020. URL https://rulelearner.wordpress.com/.

K. Peffers, M. Rothenberger, T. Tuunanen, and R. Vaezi. Design science research evaluation. In *Design Science Research in Information Systems. Advances in Theory and Practice*, pages 398–410. Springer Berlin Heidelberg, 2012. ISBN 978-3-642-29863-9.

S. Pilarski, M. Staniszewski, M. Bryan, F. Villeneuve, and D. Varró. *Software and Systems Modeling 20(5).*

J. R. Quinlan. Induction of decision trees. *Machine Learning*, 1:81–106, March 1986. doi: 10.1007/s12599-017-0516-y.

K. Rajarshi. Data on life expectancy, 2018. URL `https://www.kaggle.com/kumarajarshi/life-expectancy-who`.

K. Sandkuhl, H.-G. Fill, S. Hoppenbrouwers, J. Krogstie, F. Matthes, A. Opdahl, G. Schwabe, Uludag, and R. Winter. From expert discipline to common practice: A vision and research agenda for extending the reach of enterprise modeling. *Business Information Systems Engineering*, 60:69–80, 02 2018. doi: 10.1007/s12599-017-0516-y.

V. L. Sauter. *Decision Support Systems for Business Intelligence*. John Wiley & Sons, 2011.

H. A. Simon. *The New Science of Management Decision*. New York: Harper and Row, 1960.

R. Solingen and E. Berghout. *The Goal/Question/Metric Method: A Practical Guide for Quality Improvement of Software Development*. McGraw-Hill Publishing Company, 01 1999.

R. Thabet, D. Bork, A. Boufaied, E. Lamine, O. Korbaa, and H. Pingaud. Risk-aware business process management using multi-view modeling: Method and tool. *Requirements Engineering*, 26(3):371–397, 2021. doi: 10.1007/s00766-021-00348-2. URL `https://doi.org/10.1007/s00766-021-00348-2`.

H. D. Wang. Research on the feature of car insurance data based on machine learning. In *3rd Intl. Conf. on Mechatronics and Intelligent Robotics (ICMIR-2019), Procedia Comp. Sci. vol. 166*, pages 582–587, 2020.

M. Wiemuth, D. Junger, M. A. Leitritz, J. Neumann, T. Neumuth, and O. Burgert. 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 CARS*, 12:1439–1449, 2017.

C. Wohlin, P. Runeson, M. Höst, M. Ohlsson, B. Regnell, and Wesslén. *Experimentation in Software Engineering*. Springer, 2012.

Y. Yao. An outline of a theory of three-way decisions. In J. Yao, Y. Yang, R. Słowiński, S. Greco, H. Li, S. Mitra, and L. Polkowski, editors, *Rough Sets and Current Trends in Computing*, pages 1–17, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

L. Yin, X. Xu, J. Ding, Z. Jiang, and K. Sun. A three-way decisions model for decision tables. In *2017 29th Chinese Control And Decision Conference (CCDC)*, pages 258–263, 2017. doi: 10.1109/CCDC.2017.7978102.

H. Zjao and K. Oin. Mixed feature selection in incomplete decision table. *Knowledge-Based Systems*, 57:181–190, 2014.

APPENDIX A

Source Code of the Proposed Toolchain

In [ ]:

```
import pandas as pd
import numpy as np
import math
import seaborn as sn
import xgboost as xgb
import graphviz
import os
import pickle

from scipy.stats import shapiro
from sklearn import preprocessing
from matplotlib import pyplot
from statsmodels.graphics.gofplots import qqplot
from statistics import mean

from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression, Lasso, Ridge
from sklearn.metrics import mean_squared_error, mean_absolute_error
from sklearn.tree import DecisionTreeRegressor
from sklearn import tree
from sklearn.model_selection import cross_val_score

# helpder functions:
SEED = 123
XGB_EPOCH_NR = 2500
np.random.seed(SEED)


def doNormalityTest(data):
    # lets use shapiro test wehther the output var is normally distributed

    stat, p = shapiro(data[OUTPUT_VAR_NAME])
    print('Statistics=%.3f, p=%.3f' % (stat, p))

    alpha = 0.05
    if p > alpha:
        nomarlity_test_result = "Sample looks Gaussian (fail to reject H0)"
        print('Sample looks Gaussian (fail to reject H0)')
    else:
        nomarlity_test_result = "Sample does not look Gaussian (reject H0)"
        print('Sample does not look Gaussian (reject H0)')

    return nomarlity_test_result


def createPlotsOfResponsevar(data):
    # create Q Q plot for vusal normaility check
    qqplot(data[OUTPUT_VAR_NAME], line='s')
    pyplot.title(
        "Q-Q Plot of the response Variable for Visual  Normality check")
    pyplot.savefig("figures/qqplot_of_output_var.png", bbox_inches='tight')
    # pyplot.show()
    pyplot.clf()

    # Create density and Box plots:
    data[OUTPUT_VAR_NAME].plot.density()
    pyplot.title("Density Plot of Resposne Variable")
    pyplot.savefig("figures/density_plot_of_output_var.png",
                   bbox_inches='tight')
    # pyplot.show()
    pyplot.clf()

    data.boxplot(column=OUTPUT_VAR_NAME)
    pyplot.title("BoxPlot of Resposne Variable")
    pyplot.savefig("figures/boxplot_of_output_var.png", bbox_inches='tight')
    # pyplot.show()
    pyplot.clf()

    numerics = ['int16', 'int32', 'int64', 'float16', 'float32', 'float64']
    pd.plotting.scatter_matrix(data.select_dtypes(include=numerics), figsize=(10, 10), marker='.', hist_kwds={
                               'bins': 10}, s=60, alpha=0.8)
    pyplot.savefig("figures/pairplot.png")
```

In [ ]:

```
def createCorrMatrix(data):
    corrMatrix = data.corr()
    sn.heatmap(corrMatrix, annot=True)
    pyplot.savefig("figures/corelation_matrix.png", bbox_inches=None)
    # pyplot.show()
    pyplot.clf()


def prePorcessData(data):
    le = preprocessing.LabelEncoder()
    columns_to_replace = data.select_dtypes(
        include=["bool", "object"]).columns.tolist()
    for column in columns_to_replace:
        data[column] = le.fit_transform(data[column].values)

    return data.dropna()

def fitModels(X_train, y_train):

    modelList = []
    modelnameList = []

    model = Lasso()
    model.fit(X_train, y_train)
    modelList.append(model)
    modelnameList.append("Lasso")
    pickle.dump(model, open("models/lasso.sav", 'wb'))

    model = Ridge()
    model.fit(X_train, y_train)
    modelList.append(model)
    modelnameList.append("Ridge")
    pickle.dump(model, open("models/ridge.sav", 'wb'))

    model = DecisionTreeRegressor(max_depth=5)
    model.fit(X_train, y_train)
    modelList.append(model)
    modelnameList.append("Regression tree")
    pickle.dump(model, open("models/decisionTree.sav", 'wb'))

    tree.export_graphviz(model, out_file="figures/treeDotfile.dot",
                         feature_names=X_train.columns,
                         filled=True)

    fig = pyplot.figure(figsize=(30, 30))
    _ = tree.plot_tree(model, feature_names=X_train.columns, filled=True)
    pyplot.savefig("figures/decisionTree.png")
    pyplot.clf()
    # fitting the cgboost model requires a slightly different dataformat:
    dtrain = xgb.DMatrix(X_train, label=y_train)
    dtest = xgb.DMatrix(X_test, label=y_test)
    # define model hyperparamters
    param = {'max_depth': 15, 'eta': 0.1, 'objective': 'reg:squarederror'}
    param['nthread'] = 16
    param["subsample"] = 0.5
    param["colsample_bytree"] = 0.5
    param['eval_metric'] = 'rmse'
    num_round = XGB_EPOCH_NR
    evallist = [(dtest, 'eval'), (dtrain, 'train')]
    # fitting the model

    bst = xgb.train(param, dtrain, num_round, evallist)

    bst.save_model('models/xgb.model')

    modelList.append(bst)
    modelnameList.append("xgb")
    xgb.plot_importance(bst)
    pyplot.title("Importance from XGB model")
    pyplot.savefig("figures/importanceplot_xgb.png")
    pyplot.clf()

    xgb.plot_tree(bst, num_trees=0)
    fig = pyplot.gcf()
    fig.set_size_inches(150, 100)
    fig.savefig('figures/tree_xgb.png')
    pyplot.clf()
    return modelList, modelnameList


def MAPE(y_test, y_pred):
    mape = np.mean(np.abs((y_test - y_pred) / y_test)) * 100
    return mape

def evalModels(X_test, y_test, modelList, modelnameList):

    results_dict = {}

    for model, modelName in zip(modelList, modelnameList):
        if modelName == "xgb":
            print(model)
            dtest = xgb.DMatrix(X_test)
            y_pred = model.predict(dtest)
            rmse = math.sqrt(mean_squared_error(y_test, y_pred))
            mae = mean_absolute_error(y_test, y_pred)
            mape = MAPE(y_test, y_pred)
            results_dict[modelName] = {"rmse": rmse, "mae": mae, "mape": mape}
        else:
            y_pred = model.predict(X_test)
            rmse = math.sqrt(mean_squared_error(y_test, y_pred))
            mae = mean_absolute_error(y_test, y_pred)
            mape = MAPE(y_test, y_pred)

            results_dict[modelName] = {"rmse": rmse, "mae": mae, "mape": mape}
```

In [ ]:

```
def crossValidation(data):
    """
    This function get the dataframe as input and performs 5 fold crossvalidation on the fitted models
    for the metric RMSE is used
    """
    X, xt, y, yt = train_test_split(data.drop(
        OUTPUT_VAR_NAME, axis=1), data[OUTPUT_VAR_NAME], test_size=0.01, random_state=SEED)

    model = pickle.load(open("models/lasso.sav", 'rb'))
    lassoCV = -mean(cross_val_score(
        model, X, y, cv=5, scoring='neg_root_mean_squared_error'))

    model = pickle.load(open("models/ridge.sav", 'rb'))
    ridgeCV = -mean(cross_val_score(
        model, X, y, cv=5, scoring='neg_root_mean_squared_error'))

    model = pickle.load(open("models/decisionTree.sav", 'rb'))
    decTreeCV = -mean(cross_val_score(
        model, X, y, cv=5, scoring='neg_root_mean_squared_error'))

    param = {'max_depth': 15, 'eta': 0.1, 'objective': 'reg:squarederror'}
    param['nthread'] = 16
    param["subsample"] = 0.5
    param["colsample_bytree"] = 0.5
    param['eval_metric'] = 'rmse'
    num_round = XGB_EPOCH_NR

    dtrain = xgb.DMatrix(X, label=y)
    cv_results = xgb.cv(
        param,
        dtrain,
        num_boost_round=num_round,
        seed=SEED,
        nfold=5,
        metrics={'rmse'}
    )["test-rmse-mean"][-1:]

    return lassoCV, ridgeCV, decTreeCV, cv_results


FILE_NAME = os.getenv("FILE_NAME", "DecisionAnalysisResults-12000-Dinev.csv")
OUTPUT_VAR_NAME = os.getenv("OUTPUT_VAR_NAME", "Praemie")

data = pd.read_csv(FILE_NAME, sep=",")


# here we generate the results of the analysis, modelling and evaluation:

summaryStatistics = data.describe()
print(data.dtypes)
createPlotsOfResponsevar(data)
nomarlity_test_result = doNormalityTest(data)
createCorrMatrix(data)


# data pre processing step, where we clean the data and perform an 80/20 train test split for future use
data = prePorcessData(data)

X_train, X_test, y_train, y_test = train_test_split(
    data.drop(OUTPUT_VAR_NAME, axis=1), data[OUTPUT_VAR_NAME], test_size=0.20, random_state=SEED)

# fitting the models
modList, modNameList = fitModels(X_train, y_train)

# evaluating the models
resultDicts = evalModels(X_test, y_test, modList, modNameList)
print(resultDicts)


# Perfroming cv:
a, b, c, d = crossValidation(data)

print("5 fold mean rmse cv results of lasso ridge dectree xgb:")
print(a)
print(b)
print(c)
print(d)
# stroing the CV results in a dictionary
cvResultsDict = {"Lasso": {"rmse": a}, "Ridge": {"rmse": b},
                 "Regression Tree": {"rmse": c}, "XGBoost": {"rmse": d}}


# Last part of the script
# Writing a brief report on what has been found out about the data itself
# and the performance results of the different models
try:
    os.remove("Report.txt")
except:
    pass

f = open("Report.txt", "a")
f.write("This file hold some aggreagted information about the data and also the results of the different models. \n also take a
look at the figures folder for visualizations. \n \n \n")
f.write("1. Summary statistics of the data: \n \n")
f.write(summaryStatistics.to_string() + "\n")
f.write("with the Shapiro test it was decided that the " +
        nomarlity_test_result + "\n \n \n")
f.write("2. Model performance results \n \n")
f.write("WE fitted 4 models, namely Lasso and Ridge regression models, Regression tree model and finally XGBoost. \n For the eva
luation of the performances we used RMSE MAE and MAPE scores on 80/20 train test splits, \n and we also performed 5-fold Cross V
alidation for the RMSE metric, the results are below: \n \n")
f.write(pd.DataFrame.from_dict(resultDicts, orient='index').to_string())
f.write("\n The 5-fold cross validation results on RMSE are: \n")
f.write(pd.DataFrame.from_dict(cvResultsDict, orient='index').to_string())
```