<u>**Discovering and ranking validation rules in supervisory data**</u>

Willem Jan Willemse[1] and Annick van Ool[1]

Accepted to "Central bankers go data driven: applications of AI and ML for policy and prudential supervision", organized by De Nederlandsche Bank Amsterdam on 12 - 13 May 2022

Abstract

To meet the increasing data quality challenges posed by a data-driven approach to supervision, new techniques and tools are needed. In this paper, we introduce the Ruleminer software package to discover automatically rules and patterns in regulatory data complemented with a statistical machine learning model for their subsequent classification and ranking. Our approach was developed and tested on Solvency 2 data but is not limited to insurance data. Already available as an open source library and confirmed by a number of use cases described in this paper using granular assets data, insurance claim data and own funds data, our work could serve as a basis for the automatic rule discovery and ranking system with regard to rules optimization and as a systemic approach to the development of new rules.

1. Introduction

In a data-driven approach to supervision, central banks and supervisory authorities increasingly rely on large (granular) datasets from financial entities. Many reporting frameworks contain well-established sets of validation rules defined by statistical experts that are embedded in a data point model (taxonomy) that is regularly updated and communicated to financial entities. However, when it comes to granular data this approach is not always feasible and new techniques and tools are needed to assess the data quality. The fast changing financial services landscape in the recent years introduces new regulatory challenges. It also requires collection of additional datasets from the supervised entities, sometimes on an ad-hoc basis. The most relevant example of such a challenge is the Covid-19 pandemic that led central banks to update existing and to introduce new reporting frameworks to ensure compliance of supervised entities.

In this paper, we present a method and a publicly available Ruleminer software package[2] to generate automatically large sets of validation rules and discover underlying data patterns in (granular) datasets by using so-called association rules mining followed by the rule-ranking algorithm to select the most important rules. With this statistical machine learning (ML) approach, we identify strong relations between features in datasets. We subsequently convert these relations into human readable rules that can be enforced to automate data quality compliance and communicated with relevant central banks or regulatory authority divisions for subsequent analysis. We focus on two aspects

[1] De Nederlandsche Bank
[2] We publish the source code for generating validation rules under MIT license as Python package Ruleminer (https://pypi.org/project/ruleminer/).

of our approach: applicability and novelty. The datasets forming a basis for this work are coming from the Solvency 2 reporting framework.

In the association rules mining approach, data is stored in a so-called a transaction database. In this context, we define a transaction as a single reporting line by a specific entity at a specific reporting or reference date. The database contains all reporting lines by a given number of entities within a reporting period. The approach described in this work is applicable not only to simple reports with fixed rows and columns, but also to more complex reports with granular data and, so called, "open cells" with non-fixed number of entries. We also extend the approach to include results from quantitative values by using rule templates proposed by experts or derived from the structure of existing validation rules. Rule templates can contain Regular Expressions (regex). Regex are a powerful technique developed in computer science and formal language theory where a sequence of characters specifies search patterns. This extension is needed because association rules mining in its original form is based on categorical data only and supervisory reports consist of qualitative data as well as quantitative data. With this approach to association rules mining, we are able to generate validation rules automatically and calculate a wide range of associated metrics for each rule. We prune the set of discovered validations rules to reduce the number of rules that are semantically identical (where for example the ordering of data points does not matter for the results of the rule).

Automatically discovered validation rules are subsequently ranked by their importance with a supervised machine learning algorithm. This algorithm was trained on quantitative statistical rules metrics and expert labeling of the test dataset coming from the supervisory reporting framework with multiple established rules. The features of the model were selected in such a way that the ranking can be applied to various other datasets coming from both new and existing reporting frameworks and other sources of data. After assessment by supervisors, the most important generated validation rules can be submitted for approval. Approved sets of generated validation rules are fully transparent and explainable and can be made public, shared with supervisory authorities and incorporated within both internal and external data quality processes.

The remainder of this paper is structured as follows. In Section 2 we give a brief overview of the literature and discuss academic research that formed a basis for this work. We discuss in detail the development of the rule mining approach to the supervisory data in Section 3. We start this section with a discussion on principles of rules mining – and how it can be seen as a machine learning approach to data pattern discovery including the theory and the metrics developed for the purpose of evaluation of results. The supervisory data has some unique characteristics. In particular, it includes relatively high quality and subject expert availability, which we exploit in the development of our rules ranking approach. Section 4 is dedicated to applications of Ruleminer to various supervisory datasets. The paper is concluded with Section 5 discussing possible extensions and our ongoing work and potential extensions of this work beyond data quality checks, in particular to anomaly and outlier detection.

2. Overview of literature

Rules mining is a subset of the wide data mining field of research dedicated to automatic discovery of patterns and relationships in large datasets. The problem of mining (categorical) association rules was originally introduced by Agrawal et al. (1993) and broadened in Agrawal and Srikant (1994). Quantitative association rules were first considered by Piatetsky-Shapiro (1991) who also introduced some of the first quantitative metrics measuring interestingness of discovered rules. Among the most popular rules mining algorithms are Apriori, Eclat and FP-Growth[3]. See Aumann and Lindell (2003) and Salleb-Aouissi et al. (2013) for detailed reviews. We use a statistical technique derived from association rules mining where we restrict the set of possible association rules. This approach could potentially miss some complex and unusual patterns, but it has very significant advantages compared to other algorithms:

- the computational complexity is much lower;
- the minimum confidence and support thresholds can be set quite low to discover rare rules without significantly slowing down the code.

An important aspect of association rules mining is the ranking and selection of generated rules for further implementation in data quality checks. Large numbers of rules can be generated but only a small number of these rules are feasible for implementation. The ranking techniques discussed in the literature include an optimization approach coming from the data envelopment analysis (DEA) (Charnes et al. (1978)) and multiple experts ranking (average "collective strength") incorporated into the linear regression model together with quantitative metrics (see for Chen (2007); Bazaldua et al (2014); Toloo et al. (2019)). Our approach to rule ranking is derived from both uniqueness of our dataset of Solvency 2 returns that has a predefined taxonomy and expert availability. It is based on the principles of supervised statistical learning. According to our knowledge, such an approach was never applied to rules ranking before, possibly due to the lack of datasets and expertise available in the central banking community.

Finally, we refer to the paper of Romano et al. (2021) that has a similar background and motivation as our paper. The authors use regression to detect patterns in data and do not rank data rules generated by their method. This is a different approach to rule mining that we propose in our work. We leave it to the reader to decide what technique is more suitable for her or his needs.

3. Applying rule mining to supervisory reports

Rules are discovered automatically using association rules mining algorithms and are evaluated based on the quantitative (statistical) metrics. These rules are afterwards ranked by their interest/importance based on the subject knowledge.

3.1 Supervisory reports

Supervisory reports are based on so-called reporting frameworks or taxonomies, for which often the open XBRL (eXtensible Business Reporting Language) standard is used. A taxonomy consists of many individual supervisory reports for all relevant supervisory topics. The structure of individual reports can be one-dimensional, with columns only, two-dimensional, with, often many, columns and rows, and three-dimensional, with an additional z-axis (for example one report per currency). Furthermore, reports can be closed (with a fixed number of rows and columns) and open (with a fixed number of columns and with a non-fixed number of rows). In this paper we will use the table representations of XBRL reports, with the report name plus the so-called row-column-code to identify a single data point. One report is therefore one line in a dataset. The index of the line identifies the reporting entity, the reporting period and, if applicable, additional fields for open tables. In this way all reports can be analyzed in the same manner. By concatenating reports with the same dimensions, it is possible to analyze mutual relations between datapoints from different reports.

[3] Frequent patterns.

Reporting taxonomies cover a wide range of relevant supervisory topics and a typically relatively comprehensive. The Solvency 2 taxonomy[4] consist of 112 reporting templates for 22 entry points. Different entry points are used depending of the reporting entity type, for example solo or group, and the reporting period, for example annual or quarterly. The specific reporting templates might be different between entry points. Currently, there are approximately 1.300 explicit validation rules in the taxonomy. Every year new validation rules are added to the existing set of rules.

Some limitations that we currently face of in formulating validation rules are the following:

- to embed very large sets of validation rules (> 10.000) within a taxonomy based on XBRL is practically not feasible. Especially for the validation of large granular datasets this is problematic;
- the validation of datapoints between two reporting periods is not possible as validation rules embedded in an XBRL taxonomy are only applicable within one submission, and;
- validation rules in the XBRL standard do not allow the use of external data sources to validate data points, for example data from financial markets and data from other supervisory reports (even by the same entity).

These limitations are understandable given the scope of the current validation rules in XBRL (which is to validate a submission before acceptance). However, it now becomes increasingly clear that on top of the current validation rules new techniques and tooling are needed that meet current data quality demands. This paper presents an approach to overcome these limitations and allows for more complicated validation rules based on complex mathematical functions and enables validation against external data sets such as financial markets data.

3.2 Rule grammar

The academic literature dedicated to data rules mining defines two rule categories:

- conditional or transactional expressions of the form "IF X THEN Y". This rule is true exactly when it is not the case that X is true and Y is false. Condition X is called the antecedent of the rule and Y is called the consequent; and
- quantitative expressions of the form "sum(A, B) = C", "A = B", "A ≥ 1.0". In many cases, conditional rules also include numerical expressions (e.g. "IF A > 0 and B < 0 THEN D = A + B – C").

To be able to formulate a wide range of possible rule expressions and to efficiently use the expressions we have formulated a rule grammar. Every rule expression has to satisfy the rule grammar which is based on basic logical and mathematical functions (see Appendix 1).

[4] The data point model, XBRL taxonomies and all related artifacts of Solvency 2 can be found in https://www.eiopa.europa.eu/tools-and-data/supervisory-reporting-dpm-and-xbrl_en

The following table shows some examples of rule expressions using several functions available in Ruleminer software package:

EXAMPLE	RULE	MEANING
1	{“A”} ≥ 0	The value of column A should be higher than zero
2	{“A”} + {“B”} = {“C”}	The sum of the values of column A and column B should be equal to the value of column C of that transaction
3	IF {“A”} = “reported” THEN {“B”} = 0	If the value of column A is the string “reported” then the value of column B should equals zero
4	ABS({“A”} - {“B”}) ≤ 1.5*10^3	The absolute difference between the value of column A and the value of column B should be smaller than 1.500
5	({“A”} < QUANTILE({"A"}, 0.95))	The value of column A of a transaction should be lower than the 95%-quantile value of that column of all transactions

Example 1 shows a basic rule that says that the reported value of a column named “A” in the report/transaction should be positive. The curly braces around the string “A” are used to indicate a column name in the dataset. Similar rule expressions can be found in existing validation rules of the Solvency 2 reporting framework, for example to make sure that reported values are not negative.

Supervisory reports often contain values that should be the sum of other values, possibly from different reports. Often, but not always, validation rules are included in the reporting taxonomy to check that one value is the sum or some other mathematical function of other values. Example 2 shows a simple form of this rule expression to generate validation rules of this form.

Example 3 combines categorical data with quantitative data. In supervisory reports there are often relations between the type of a reported item and the specific data that are reported for that item. For example, if a reported asset is of type A then columns X and Y should be reported, and if the asset is of type B then columns Y and Z should be reported. Sometimes, these checks are incorporated in the taxonomy, but if the number of possible types is very large or the number of related quantitative columns is large then it is not feasible to explicitly formulate these relations into validation rules of the reporting taxonomy.

Example 4 is a variation of a rule for checking that the value of two columns should be the same. It is not uncommon in supervisory reports that there are small differences, for example due to rounding operations of underlying values. In order to ignore these small differences example 4 shows how to formulate a rule with a threshold of an absolute difference of 1500.

The first four example are based on existing validation rules in the taxonomy. Example 5 is a rule to detect outliers in a dataset.

3.3 Regular expressions in rules

For strings and column names within rule expressions we can use regular expressions. Regular expressions (or regexes, or regex patterns) form a small programming language to specify patterns for the set of possible strings that we want to select.

Suppose we have a template with the following column names: "S.01.R010,C010", "S.01.R010,C020", "S.01.R020,C010", "S.01,R020,C020" (identifying four datapoints in template S.01 with row and column code) and "S.02,R010,C010".

Table 1: A typical template structure with rc-codes

Template S.01	S.01.R010 ,C010	S.01.R010 ,C020	S.01.R020 ,C010	S.01.R020 ,C020	S.02.R010 ,C010
Reporting entity 1	.	.	.	.	.
…	.	.	.	.	.

Then, the table below provides some examples of regexes with their result

EXAMPLE	REGEX	RESULT
1	.*	S.01,R010,C010, S.01,R010,C020, S.01,R020,C010, S.01,R020,C020, S.02,R010,C010
2	S.01,R010,C010	S.01,R010,C010
3	S.01,R010,C\d+	S.01,R010,C010, S.01,R010,C020
4	S.\d+,R010,C010	S.01,R010,C010, S.02,R010,C010
5	\d{4}-\d{2}-\d{2}	2022-01-01
6	([A-Z]{2})([A-Z0-9]{9})([0-9]{1})	NL0123456789 (an ISIN)

In example 1 the regex .* expresses zero or more of any character (the dot indicates any character and the * means zero or more instance of the preceding regex token). The dot and the star are metacharacters of regex. Applying the regex to the columns of a dataset results in all columns. Example 2 shows that the regex of a string without regex metacharacters matches the string itself. Example 3 and example 4 use the regex metacharacters \d+ to indicate one or more decimal digits (the \d matches any decimal digits and the plus means one or more times).

Example 5 and 6 show two cases for the string entries in a dataset. Example 5 is a simple date string validation rule and example 6 shows a regex to check whether the ISIN in a dataset is a true ISIN (International Securities Identification Number). ISINs consist of two alphabetic characters (which are the ISO 3166-1 alpha-2 code for the issuing country), nine alpha-numeric characters (the National Securities Identifying Number, or NSIN, which identifies the security, padded as necessary with leading zeros), and one numerical check digit. This structure is expressed in the regex in the example.

In Appendix 2 we listed the most frequently used and basic metacharacters of Regular Expressions.

3.4 Rule discovery

For the rule discovery process, we implemented the following algorithm that generates all possible rules given the rule expression. First, possible substitutions of the columns and strings are determined given the regexes in the template expression, based on the column names and string values in the rows of the relevant columns. The Cartesian product of these possible substitutions provide all possible antecedents of the rule. Then, the possible substitutions of the consequent are determined based on the part of the dataset that satisfies the antecedent (this restricts the number of possible substitutions). The combination of possible antecedents and possible consequents provide the rules for which the metrics are calculated. Metrics are calculated on the basis of the full dataset.

For certain expressions this approach can lead to rules that are syntactically different but semantically identical. A simple example of this is based on the commutative properties of operators used in an expression. The rules "if A>0 and B<0 then C=0" and "if B<0 and A>0 then C=0" are different in form but have the same meaning: changing the order of A>0 and B<0 does not change the outcome because of the commutative property of the logical and-operator. A pruning algorithm has been implemented that prunes out rules that have already been discovered based on the commutative properties of the mathematical operators addition and multiplication and logical operators & (and) and | (or) and the symmetric properties of equality and inequality. The pruning algorithm is applied recursively, so a rule of the form

(({"4"}>{"3"}) & (({"2"}+{"1"})={"0"}))

will be found identical to an already discovered rule

((({"1"}+{"2"})={"0"}) & ({"4"}>{"3"}))

and consequently will be removed from the list. Before the metrics of a rule of possible rules are calculated it is checked whether a semantically identical rule already has been evaluated. If that is the case then the rule will be discarded. If not, then the metrics of the rule will be calculated and added to the list of discovered rules.

3.5 Rule metrics

A wide range of association rules metrics have already been proposed. The main quantitative metrics are support and confidence. Additional and closely related metrics are lift, exceptions and coverage. Each metric can be useful in different situations, depending on the characteristics of the dataset and the goal of the rule discovery.

The table below contains well-known metrics with their definition and interpretation.

METRIC	DEFINITION	INTERPRETATION
absolute support count	n_X	The support count is the number of occurrences of a pattern in a dataset, i.e. the number of transactions in the database that satisfy or confirm the pattern
absolute exception count	$n - n_X$	The exception count is the number of transactions in the database that do not satisfy the pattern
support	$supp(X) = {}^{n_X}/_{n}$	Support is an indication of how frequently the pattern appears in the dataset. It is defined as the proportion of transactions in the dataset which contains the given pattern. Support is often used to represent the significance of a pattern
confidence	$conf(X \rightarrow Y) = {}^{n_{XY}}/_{n_X}$	Confidence is an indication of how often a rule has been found to be true. Thus

		confidence can be interpreted as the probability of finding the consequent of the rule in transactions under the condition that these transactions also contain the antecedent. Confidence is the proportion of transactions that contain the consequent in the set of transactions that contain antecedent. It is used to measure the accuracy of a given rule
added value	$conf(X \rightarrow Y) - supp(Y)$	The added value quantifies how much the probability of the consequent increases when conditioning on the transactions that contain the antecedent
casual support	$supp(X \cup Y) + supp(\bar{X} \cup \bar{Y})$	The casual support of a rule is the support increased by the support of the negatives (based on the number of occurrences where the antecedent and consequent are not met)
casual confidence	$\frac{1}{2}\left(conf(X \rightarrow Y) + conf(\bar{X} \rightarrow \bar{Y})\right)$	The casual confidence of a rule takes positive evidence as well as negative evidence of a rule into account in equal weight. It is the confidence of the rule increased by negative evidence of the rule
conviction	$\frac{1 - supp(Y)}{1 - conf(X \rightarrow Y)}$	Conviction is an alternative to confidence, which sometimes can produce misleading results (for example when the support of the consequent is higher than the rule confidence)
lift	$\frac{conf(X \rightarrow Y)}{supp(Y)}$	Lift measures how many times more often the antecedent and consequent occur together than expected if they were statistically independent
rule power factor	$supp(X \rightarrow Y)conf(X \rightarrow Y)$	The rule power factor weight the confidence of a rule by its support. If both confidence and support are high then this leads to a high rule power factor

A main problem with the quantitative metrics is their correlation in the datasets with established rules: if the data is present and the data quality is perfect then confidence is equal to relative support. To overcome this problem additional metrics of quality/interestingness have been added to the software package including added value and rule power.

The amount of generated validation rules depends on the constraints imposed on certain metrics. Well-known constraints are minimum thresholds on support and confidence. By varying them we could identify both most frequent rules (high confidence and high absolute support) and rare albeit interesting rules (high confidence, low absolute support).

3.6 Rules ranking

Rules ranking is a critical part of the data patterns discovery and analysis. To the best of our knowledge and the analysis of the rule mining literature, there are two approaches most often used for the data rules ranking: Data Envelopment Analysis (DEA) and expert ranking. Below we discuss both approaches and introduce a new rule-ranking algorithm based on a statistical machine learning approach.

Traditional rule mining algorithms cannot classify the infrequent items to interesting itemsets since the subjective domain knowledge is ignored. A huge amount of subjective domain knowledge may exist, which can be considered as potential subjective constraints and measures for evaluating association rules. Following the discovery and reporting of some rules, a data miner can select the subjective importance (or interestingness) measures. The DEA approach introduces a single score per rule based on the weights that combine quantitative metrics with weights that are given by the experts. Then each candidate score is calculated with their most favorable weights. The resulting score is a preference score for each candidate rule. Constraints are introduced in ensure that the vote of the higher place may have a greater importance than that of the lower place. After the problems are solved for all candidates, several candidates often achieve the maximum attainable score 1. They are called efficient candidates. Then the efficient candidates are discriminated to rank them by importance. This method is in fact an application of the technique called Linear Programming (a deterministic technique of linear optimization aimed at maximizing the objective function under linear constraints).

The expert ranking approach includes multiple experts labelling the same set of rules after which the ranking average ("collective strength") is computed and added as an independent variable to the linear regression model together with the quantitative metrics. This approach is efficient for a small number of rules and a significant number of experts to average their opinions.

Our ML approach is derived from the need to overcome the number of limitations due to the data quality and expert availability, but also from the specific features of our datasets: a very large number of columns, a relatively many data points and the ground truth – multiple existing taxonomy rules. We also approach the problem differently from the previous two techniques by looking at the rules ranking not as a linear optimization or regression problem, but rather as a classification problem. In the DEA, the goal is to find the rules that maximize the linear cost function. In the expert ranking approach, the aim is to predict the strength (importance) of the new rule as a linear combination of factors. The value of this strength could vary from minus to plus infinity and this makes the result very hard to interpret. In our case, we use an expert to label existing rules using binary classification (important or not). Our approach to rule ranking is, in fact, probabilistic. We take into account that the labelling is subjective and could not be 100% accurate.

To rank the rules by their importance a number of steps are taken. First we formed a new dataset including existing rules from multiple templates detected by the Ruleminer and the features coming from their quantitative properties and statistical metrics. In parallel, the rules were labeled as important/not important. In our model, we used the following features:

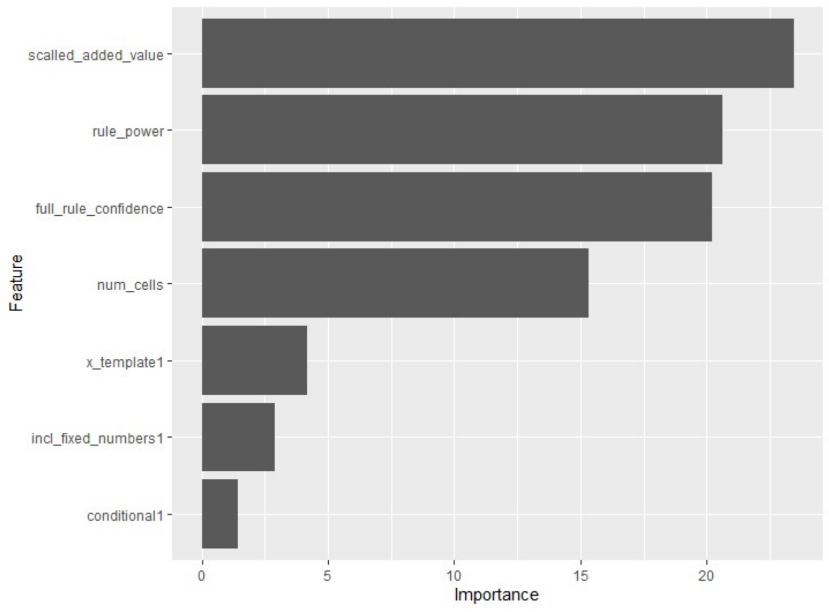
- number of cells;
- if the rule is conditional;
- if it includes fixed numbers;
- if it is cross-template;
- confidence;
- scaled added value (support subtracted from confidence multiplied by the square root of support); and
- rule power (confidence multiplied by support).

Our number of quantitative metrics (model features) is limited due to the fact that almost all existing taxonomy rules are so called "blocking rules", which means that if both antecedent (left-hand side) and the consequent (right-hand-side) are present that the rule is definitely satisfied, so many metrics suggested in literature show a very strong correlation with each other.

The total number of rules used for analysis (individual template and cross template rules) was 259. These were all rules discovered by the Ruleminer algorithm. 11 was considered as a maximum length of the rules and it was a natural threshold as the next size of the rules was 14. Very long rules had a very small support as many of the cells involved in them had missing values in most of the returns. The expert labelled dataset was well balanced: the number of important rules was 154 and the number of rules considered as non-important was 105.

The dataset was randomly divided into two parts: training/testing (75% -- 195 rules) and validation (25% -- 64 rules). To build and test the model we used the 10-fold cross-validation approach that includes multiple random sampling from the training/testing dataset and is considered the most efficient approach to develop the ML models for medium size datasets.

The variables ranked by their importance are below:



To build and test the model we considered a number of ML algorithms including logistic regression, Support Vector Machine (SVM) and Naïve Bayes. After comparing the various methods, the Random Forest was selected as the best performer for both training and validation. Our result is a rating (probability) of the rule importance between 0 and 1 that we can use that to divide rules into groups and classify as critical, most important, useful, least important or using any other qualitative scale.

Next, the model was applied to both the validation dataset to compute the probabilities and test its accuracy (and ensure that there is no over-fitting) and to the testing/training dataset to rank the rules. Based on the 50% important/non-important threshold, the algorithm had 85% accuracy for testing/training and 91% accuracy for validation dataset.

Based on our analysis the importance score is driven by the combination of three metrics: scaled added value, rule power and confidence. In taxonomy rules, confidence is technically equal to support: if the data is in then the rule is true. So the rule power is a square of confidence. In general, this measure favors rules with high confidence and high support at the same time. The most important metric in the model is a scaled added value. This metric is equal to support subtracted from confidence multiplied by the square root of support. This metric (if confidence is equal to support) has a critical point, a minimum when confidence=4/9. This metric is highly correlated to the interest factor metric (see Tan et al. (2004)). Technically the confidence is the main parameter in the model and this corresponds to the interestingness of the discovered rule.

4. Applications

The rule mining approach can be applied to various datasets to discover underlying data patterns. We will discuss three different use cases:

1. rule discovery with granular asset data;
2. rule discovery with actuarial run-off triangles between two periods; and
3. rule discovery and ranking with Solvency 2 data

4.1 Rule discovery in granular asset data

The rule mining approach was applied to the granular Solvency 2 assets and derivatives data from Dutch insurance undertakings. This data contains information of all individual assets and derivatives included in the balance sheet of insurance undertakings, for example the asset identification code, Solvency 2 value and credit rating. We generated several thousands of validation rules for each individual asset that was reported by one or more insurance undertakings in the last four quarterly reporting periods. Clearly, the number of generated rules went beyond what is feasible to incorporate within the current Solvency 2 reporting framework and would be very time consuming to formulate individually by statistical experts.

An example is a rule template for the valuation method that was used to measure the Solvency 2 value of an asset. When valuing assets in Solvency 2 insurance undertakings have to apply a valuation hierarchy: as a default method market prices in active markets for the same assets have to be used. If for a specific asset the use of market prices is not possible (because market prices are not reliable or not available) then market prices of similar assets are to be used. And if this is also not possible, then an alternative valuation method has to be used. The Solvency 2 legislation contains criteria for the application of the valuation method that has to be satisfied. Of the total assets of Dutch insurance undertakings, on average during 2021, 49% of the total asset value was measured with the default valuation method, 10% with market prices of similar assets and 36% with an alternative valuation method (for rest some less often used methods were used).

Due to market circumstances, the valuation method of certain assets could change over time, it is however expected that at a certain reporting date an asset is valued in a consistent way by all reporting insurance undertakings. To check this, we generate validation rules for the valuation method with the following rule template:

```
if ({"S.06.02,c0040"} = ".*") then ({"S.06.02,c0150"} = ".*")
```

The column S.06.02,c0040 contains the ISIN code of the asset and column S.06.02,c0150 contains the valuation method that was used to value the assets.

An example of a generated rule is:

if ({"S.06.02,c0040"}="ISIN/NL0123456789") then ({"S.06.02,c0150"}="Quoted market price in active markets for the same assets")

This validation rule states that if the reported asset has identification code *ISIN/NL0123456789* (an anonymized number) then the valuation method is *Quoted market price in active markets for the same assets*. We calculated metrics of this rule, such as confidence and rule power, to determine how strong this relation is given a certain reporting period. The Ruleminer software package generated the following output including several metrics:

Rule_definition	Rule support count	Rule exception count	Confidence	Casual confidence
if ({"S.06.02,c0040"}="ISIN/ NL0123456789") then ({"S.06.02,c0150"}="Quote d market price in active markets for the same assets")	93	4	0.958763	0.604052
Support rule	Added value	Conviction	Lift	Rule power factor
0.000421	0.208013	6.044308	1.277073	0.000404

The absolute support count of this rule equals 93. This means that ISIN code *ISIN/NL0123456789* with valuation method *Quoted market price in active markets for the same assets* occurred 93 times in the dataset. Moreover, ISIN code *ISIN/NL0123456789* occurred 4 times in the dataset with another valuation method since the absolute number of exceptions equals 4. As a result, the confidence equals 93 / 97 = 0.958763. This might indicate an error in the data, but could also indicate a discrepancy in the interpretation of valuation hierarchy between insurance undertakings of particular asset (and by aggregating these results we can look for insurance undertakings that might have a prevalence towards alternative valuation methods). The casual confidence is the weighted average of the confidence of the rule and the confidence of the negative rule, and is for this rule of limited value. The support of the rule, indicating how frequently the pattern appears in the dataset, is very low. This makes sense for a rule on granular asset data since a single asset only occurs a limited number of times in such a large dataset (220.951 assets in total). The added value, equal to 0.208, quantifies how much the probability of an asset containing valuation method *Quoted market price in active markets for the same assets* (the consequent) increases when conditioning on assets with ISIN code *ISIN/NL0123456789* (the antecedent). Conviction (6.044) is an alternative to confidence which does not capture the direction of the association adequately and can also take values above 1. The lift (1.277) measures how many times more often the antecedent and the consequent occur together than expected if they were statistically independent where are value of 1 indicates independence. Finally, the rule power factor is equal to the support of the rule times the confidence.

Besides the calculation of different metrics one can also impose constraints on metrics to restrict the amount of generated validation rules. The best-known constraints are minimum thresholds on support and confidence. Of course, the number of generated validation rules decreases when the minimum thresholds on support or confidence is higher. Figure 1 shows the relation between the number of generated rules and the minimum thresholds on confidence (minimum threshold on support is fixed) when generating validation rules for the valuation method in a Solvency 2 assets dataset.

In this application we discussed a rule template for the valuation method that was used to measure the Solvency 2 value of an asset. Of course, various other rule templates can be set up, e.g. rule templates for credit rating, country of issue or issuer sector.

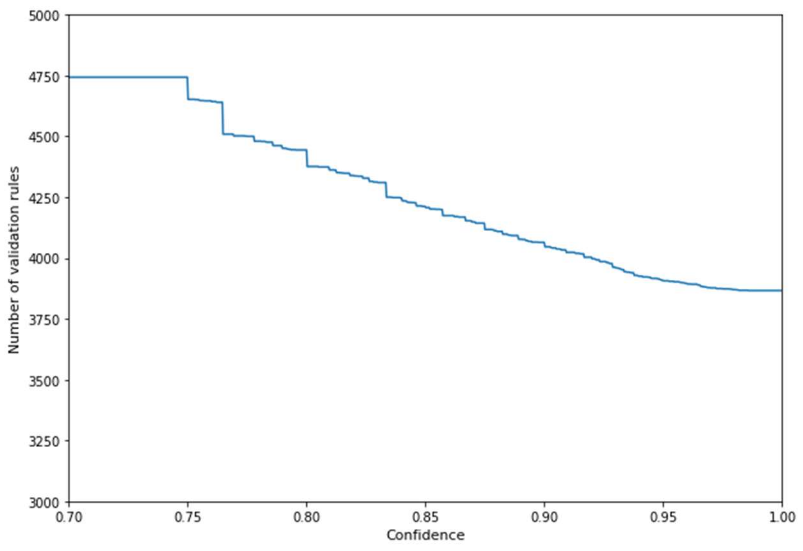


Figure 1: Relation between number of generated validation rules and minimum threshold on confidence

4.2 Rule discovery in actuarial run-off triangles

Another interesting application of rule mining is the discovery of validation rules between two periods. This is not possible in most reporting frameworks since they contain validation rules applicable to one period only.

Run-off triangles are used to estimate how much or how many insurance claims have been incurred in a financial year but are not yet reported and settled; for this a technical provision has to be held. This is called an IBNR - incurred but not reported. When a claim event occurs it takes some time before it is settled by the insurance undertaking. This is known as a claim delay. Table 1 below shows an example of a run-off triangle in which the vertical axis represents the accident year and the horizontal axis represents the development year. The accident year specifies in which year the claim is reported. The development year specifies after how many years the reported claim is getting settled. For example, in year *n* - 3 at time *t* a claim of 34 has been settled by the insurance undertaking belonging to a claim event that also took place in year *n* - 3. A year after the claim event took place (*n* - 2) another claim of 21 has been settled belonging to accident year *n* - 3. It can take several years before all claims belonging to a certain accident year have been settled.

There are clear relations between datapoints in a run-off triangle over time. A claim of accident year *i* that is settled in development year *j* at time *t* is also present in the run-off triangle at time *t* + 1: at accident year *i* - 1 in development year *j*. So claims are present in the run-off triangle for a number of years.

Besides the upper-left-side triangle formed with settled claim amounts – shown in table 1 – one can also estimate a lower-right-side triangle that contains a projection of future claim payments. There are several methods available to make such projections, e.g. using Mack's chain-ladder method or the Bornhuetter-Ferguson method.

Table 2: Example of two consecutive run-off triangles

Reported run-off triangle t

	Development year				
Accident year	n-4 (c010)	n-3 (c020)	n-2 (c030)	n-1 (c040)	n (c050)
n-4 (r010)	60	29	15	6	3
n-3 (r020)	24	34	21	8	
n-2 (r030)	32	16	11		
n-1 (r040)	19	27			
n (r050)	40				

Reported run-off triangle t+1

	Development year				
Accident year	n-4 (c010)	n-3 (c020)	n-2 (c030)	n-1 (c040)	n (c050)
n-4 (r010)	24	34	21	8	4
n-3 (r020)	32	16	11	9	
n-2 (r030)	19	27	16		
n-1 (r040)	40	19			
n (r050)	56				

We can use rules mining to generate validation rules that capture the relations between datapoints in a run-off triangle over time. To generate those rules we first have to change the set-up of the dataset. Each datapoint is represented by two columns: value of the datapoint in the first period (t) and value of the datapoint in the second period (t+1). Subsequently, we can set up a set of rule templates:

if ({"column1 (t)"} != 0)& ({"column2 (t+1)"} != 0) then ({"column1 (t)"} = {" column2 (t+1)"})

where *column1* and *column2* are iterated over all columns. An example of a generated rule is:

if ({"r050c010 (t)"} != 0)& ({"r040c010 (t+1)"} != 0) then ({"r050c010 (t)"} = {" r040c010 (t+1)"})

This is an example of a relation between datapoints in a reporting template over time that should in principle be satisfied. With rules mining validation rules for these clear relations can be generated automatically that typically need to be satisfied. However, in practice there can be cases in which these validation rules are not satisfied (due to for example claims corrections, mergers, etc.). Rule mining makes it possible to highlight these situations.

Run-off triangles are often part of reporting requirements by insurance undertakings to supervisory entities. Under Solvency 2 insurance undertakings have to report multiple run-off triangles containing non-life insurance claims for 9 different metrics (e.g. gross claims paid, reinsurance recoveries received, Gross Reported but not Settled (RBNS)[5]. For each line of business these run-off triangles should be reported separately. The run-off triangles that are reported contain 15 periods which implies, given the number of cells in a triangle, that 0.5 * 14 * 15 = 105 validation rules can be generated for each run-off triangle. This implies that the number of validation rules that is applicable to an insurance undertaking with several lines of business is big: for example, for an insurer with 12 lines of business 12 * 9 * 105 = 11.450 validation rules are applicable. It would be very time consuming to set up validation rules for all these triangles individually. Rule mining makes it possible to generate sets of validation rules for multiple templates in an efficient way.

4.3 Rule discovery, analysis and ranking with Solvency 2 data

In another project we discovered a number of new patterns that could potentially become rules.

Examples:

In the template S.25.02, we discovered a pattern that makes a perfect logical sense, but it has not been included in the taxonomy rules: ({"S.25.02.01.02,r0110c0100"} + {"S.25.02.01.02,r0060c0100"} = {"S.25.02.01.02,r0220c0100"})
That can be interpreted as "Total undiversified components + Diversification = Solvency capital requirement".

All taxonomy rules in the template S.23.04 correspond to formatting of the data (alphabetic or date). So the pattern: {"S.23.04.01.04,c0445,c0480"} = {"S.23.04.01.04,c0445,c0450"} could be used to check the quality of the data even if all the formatting is correct.

After generating new validation rules for a set of Solvency 2 templates and computing their features and metrics we sorted the rules based on their importance. For example, the rule coming from the first example had an importance score of 0.744 and was ranked 11th among all the rules discovered in this template.

5. Conclusions and extensions

In this paper we have introduced the Ruleminer software package that enables the discovery of a wide range of rules with complex mathematical formulations and by using data sources from previous submissions and other external datasets. The approach presented here allows for the formulation of new validation rules that are not possible or feasible within current reporting frameworks. The approach fits within the already mature field of association rules mining which allows for the use of a wide range of rule metrics as objective measures to rank and select the most relevant rules. Furthermore, machine learning models can be trained given subject experts labelling to rank discovered rules in an efficient manner.

[5] More details about this reporting template (S.19.01) can be found in the annotated templates for Solvency 2: https://www.eiopa.europa.eu/tools-and-data/supervisory-reporting-dpm-and-xbrl_en

Rule discovery is a well-established process. However, to the best of our knowledge, this work is the first time when it was applied to the supervisory data. The automatic rule detection and ranking methodology is a feasible and useful tool for data quality checks. The ML rule ranking approach introduced in this work is both a novel and efficient way to determine the rule importance that was derived from the uniqueness of the data at our disposal. The statistical nature of the approach requires significant number of returns to detect and analyze rules in new templates, however, it could be tens rather than hundreds or thousands of records.

The Ruleminer software package aims at a wide applicability. It is a pure-Python software package and can be applied by data analysts within data science environments. The package uses the well-known pandas DataFrame format (similar to the table representation of supervisory reports). A range of parameters can be set to steer the rule discovery process, for example selection of rule metrics that are to be calculated, filters (minimum thresholds discovered rules should satisfy) and precision variables. A reasonably fast algorithm has been implemented which makes it possible to apply the rule discovery algorithm to very large datasets.

We have discussed a number of relatively straightforward applications of the rule mining approach. However, other interesting extensions are possible as well.

Despite being originally developed as a part of the insurance framework, Ruleminer is currently also applied to the data coming from the banking sector, for instance. Loan Level Data (LLD) is one of the largest sets of the granular data submitted to central banks. This data does not have well established taxonomies similar to Solvency 2. So the Ruleminer can play an important role in the discovery of new rules and in ongoing data quality checks.

Furthermore, with the help of the Ruleminer we could identify the rules that can be imposed on the data providers (banks) to ensure the quality of submissions (pre- or post-processing) and the rules' exceptions can be used to identify anomalies due to the data quality problems. The approach used to identify the rules follows the direction of the data rules project implemented for Solvency 2 data, but it has three distinct features:

- it comprises of a very large dataset (x15 more records compared to the largest Solvency 2 dataset);
- a significant part of the fields is formed by encoded categorical variables;
- there are no (or very few) taxonomy requirements, meaning that the quality of data is potentially lower and detected rules might have multiple exceptions due to the data quality problems.

In addition to applying Ruleminer to other datasets there are also extensions possible with respect to the type of patterns that are found. Some patterns in supervisory data are specific for certain (types of) financial entities, or are only relevant at a certain point during the year or within a certain time period. For example, specific rules can be discovered that are only applicable to a small subset of reporting financial entities or even a single entity. Time-dependent rules could capture certain developments over time and variations during the year, for example if some insurance contracts are yearly payable then insurance premiums are reported in one quarterly report and not in the other quarterly reports. By adding the type of entity and the time or point during the year as features in the dataset, rules can be discovered based on these features.

In number of experiments we observed that exceptions to discovered rules with high confidences could be confirmed with the use of outlier detection models. A combined approach with association rules mining and quantitative models might be fruitful. For example discovered rules could be strengthened by results from unsupervised quantitative models for outlier detection, and, vice versa, outliers are explained by more transparent rules.

Based on a number of heuristics and rules of thumb, a standard set of default rule expressions can be formulated to discovery rules from and assess the data quality of new and ad-hoc surveys and reports. For example, rule expressions to discover rules that check whether values were reported with inconsistent data types or unit of measures and to validate that submitted reports were complete and that no datapoints were missing.

Acknowledgments

The authors acknowledge the help of Dr Eugene Kashdan (Central Bank of Ireland) with testing the Ruleminer software package and introducing the authors to additional rule interestingness metrics and rule ranking algorithms.

References

Agrawal, R., Imieliński, T., & Swami, A. (1993). Mining association rules between sets of items in large databases. In *Proceedings of the 1993 ACM SIGMOD international conference on Management of data* (pp. 207-216).

Agrawal, R., & Srikant, R. (1994). Fast algorithms for mining association rules. In *Proc. 20th int. conf. very large data bases, VLDB* (Vol. 1215, pp. 487-499).

Aumann, Y., & Lindell, Y. (2003). A statistical theory for quantitative association rules. *Journal of Intelligent Information Systems*, *20*(3), 255-283.

Bazaldua, D. L., Baker, R., & Pedro, M. O. (2014). Comparing expert and metric-based assessments of association rule interestingness. In Proceedings of the *Educational Data Mining conference, London, UK*

Bornhuetter, R. L., & Ferguson, R. E. (1972, November). The actuary and IBNR. In *Proceedings of the casualty actuarial society* (Vol. 59, No. 112, pp. 181-195). Arlington: Casualty Actuarial Society.

Charnes, A., Cooper, W.W., & Rhodes, E. (1978). "Measuring the Efficiency of Decision Making Units". In *European Journal of Operational Research*, Vol. 2, No. 6, pp. 429–444.

Chen, M. C. (2007). Ranking discovered rules from data mining with multiple criteria by data envelopment analysis. *Expert Systems with Applications*, *33*(4), 1110-1116.

Mack, T. (1993). Distribution-free calculation of the standard error of chain ladder reserve estimates. *ASTIN Bulletin: The Journal of the IAA*, *23*(2), 213-225.

Piatetsky-Shapiro, G. (1991). Discovery, analysis, and presentation of strong rules. *Knowledge discovery in databases*, 229-238.

Romano, S., Martinez-Heras, J., Raponi, F. N., Guidi, G., & Gottron, T. (2021). *Discovering new plausibility checks for supervisory data* (No. 41). European Central

Bank.

Salleb-Aouissi, A., Vrain, C., Nortet, C., Kong, X., Rathod, V., & Cassard, D. (2013). QuantMiner for mining quantitative association rules. *The Journal of Machine Learning Research*, *14*(1), 3153-3157.

Tan, P-N., Kumar V., Srivastava J., (2004) *Selecting the right objective measure for association analysis*, Information Systems 29, 293–313

Toloo, M., Sohrabi, B., & Nalchigar, S. (2009). A new method for ranking discovered rules from data mining by DEA. *Expert Systems with Applications*, *36*(4), 8503-8508.

Appendix 1: rule grammar of the Ruleminer package

The rule template describes the structure of the rule. Columns and quoted strings in the rule template can contain simple regular expressions. The syntax of the template follows a grammar defined as follows:

A *template* is of the form:

if cond_1 then cond_2

or simply a single:

cond_1

A *condition* is either a combination of comparisons with logical operators ('&' and '|') and parenthesis:

(comp_1 & comp_2 | comp_3)

or simply a single comparison:

comp_1

A *comparison* consists of a term, a comparison operator (>=, >, <=, <, != or ==) and a term, so:

term_1 > term_2

A *term* can be a number (e.g. 3.1415 or 9), quoted string (a string with single or double quotes), or a function of columns.

A *function of columns* is either a prefix operator (min, max, quantile, or abs, in lower or uppercase) on one or more columns, and of the form, for example:

min(col_1, col_2, col_3)

or infix operators with one or more columns:

(col_1 + col_2 * col_3)

A *column* is a string with braces, so:

{"Type"}

where "Type" is the name of the column in the DataFrame with the data

A *string* consists of a-z A-Z 0-9 _ . , ; ; < > * = + - / ? | @ # $ % ^ & ( )

Appendix 2: overview of Regular Expressions

We list here the most frequently used metacharacters of Regular Expressions. A full description of regular expression operators can be found here https://docs.python.org/3/library/re.html.

Character classes

\s	matches any white space
\S	matches not a white space
\d	matches any digit
\D	matches any non-digit
\w	matches any alphanumeric character
\W	matches any non-alphanumeric character
\c	matches a control character

Quantifiers

*	zero or more times
+	one or more times
?	zero or one
{3}	exactly 3
{3,}	3 or more
{3, 5}	3,4 or 5

Anchors

^	start of string, or start of line in multiline pattern
\A	start of string
$	end of string, or end of line in multiline pattern
\Z	end of string
\b	word boundary
\B	not a word boundary
\<	start of word
\>	end of word

POSIX

[:upper:]	uppercase letters, similar to [A-Z]
[:lower:]	lowercase letters, similar to [a-z]
[:alpha:]	upper- and lowercase letters, similar to [A-Za-z]
[:digit:]	digits, similar to [0-9]
[:alnum:]	digits, upper- and lowercase letters, similar to [A-Za-z0-9]
[:punct:]	punctuation (all graphic characters except letters and digits)
[:blank:]	space and TAB characters
[:space:]	blank (whitespace) characters, similar to [\t\n\r\f\v]
[:word:]	alphanumeric characters with underscore character _ (alnum + _)