Learning Compact Transparent Models using Neuro-Symbolic Methods

Case Study

Paul Peyssard
Mohamed Hedi Derbel
mohamed-hedi.derbel@etu.univ-cotedazur.fr
paul.peyssard@etu.univ-cotedazur.fr

March 2023

IBM
Côte d'Azur University
MSc Data Science & Artificial Intelligence

Supervised by Greger Ottosson, Yusik Kim[1]

1 Abstract

As businesses increasingly rely on Machine Learning (ML) for automated decision-making, interpretability becomes a crucial requirement in high-stakes industries such as health, finance, or insurance. Models that are transparent, such as decision models based on rules, are highly desirable due to their interpretability. Nevertheless, *current methods* for inducing rules frequently produce *rule models that are lengthy and difficult to interpret.*

To tackle this issue, a solution has been proposed by IBM Research and Lab in the form of the Relational Rule Network (R2N)[2], which is a neural architecture that can **learn literals representing a linear relationship among numerical input features, as well as the rules that use them**. The evaluation conducted on benchmark tasks reveals that R2N-learned literals preserve interpretability and produce more succinct sets of rules when compared to leading rule induction algorithms.

Our case study is centered around the Learning Compact Transparent Models using Neuro-Symbolic Methods project at IBM, which aims to utilize the R2N approach for learning transparent models to enhance interpretability and accountability in a range of business automation products. The purpose of our study is to investigate the context and approach of a project, as well as its potential outcomes, with a focus on extracting new features from R2N. Specifically, we aim to implement these new learned rules on various rule-based decision models, such as Naive Bayes, Decision Tree, and Random Forest. Through this implementation, we intend to examine the impact of the new features on the predictions made by classical ML models.

2 Introduction

Over the last decade, black box decision models (e.g. neural networks) are increasingly used in high-stakes decision-making, yet there has been equally growing concern over their use[1]. Various industries are facing increasing pressure to ensure accountability and transparency in their decision-making models. This demand stems not only from regulatory concerns but also from end-users who are seeking greater interpretability of the final decision. Additionally, recent legislation such as the proposed EU AI Act act has further emphasized the need for transparency and explainability in high-stakes decisions. The R2N model offers a solution to the limitations of rule-based models that rely solely on predefined data. With R2N, a neural network consisting of three layers, literals can be learned directly from the data, improving the expressiveness and interpretability of the resulting rule model. In the first layer, the network learns literals that represent half-spaces, and in the second and third layers, these literals are mapped to binary predictions, creating a logical formula in the Disjunctive Normal Form (DNF). R2N has the advantage of improving the expressivity and accuracy of the model by simplifying it and enabling the representation of ratios as a single literal instead of a conjunction of multiple literals.

3 Goal: Automatic feature engineering

3.1 Goal of the project

One significant benefit of using a neural network-based method is that feature engineering is inherently integrated and transparent within the network. The objective of this project is to explore the potential for automatically extracting features from R2N to enhance other machine learning models.

For instance, if we extract a rule consisting of different conditions based on the existing features of a dataset and apply it to every row, creating a Boolean column (True if the conditions are satisfied), will this increase the model's accuracy compared to using only the original dataset?

To address this question, we aim to develop a code that automates the extraction of features from R2N and evaluates their effectiveness when incorporated into different machine-learning models. By comparing the performance of models with and without the extracted features, we hope to gain insights into the benefits of automatic feature engineering and its potential applications in various domains.

In addition to potentially improving the accuracy of machine learning models, automatic feature engineering from R2N can also contribute to increased explainability. As neural networks are often considered "black boxes" due to their complex internal representations and decision-making processes, extracting conditions and features can provide more interpretability.

By transforming the implicit knowledge within neural networks into explicit rules in a human-readable form, we can make it easier for domain experts and stakeholders to understand and trust the predictions made by the models. This improved explainability can lead to better decision-making, as well as facilitate collaboration between data scientists and subject matter experts.

In summary, the automatic feature engineering from neural networks not only has the potential to enhance the performance of other machine learning models but also plays a crucial role in favoring explainability and trust in AI-driven decision-making processes.

3.2 Understanding of the rules

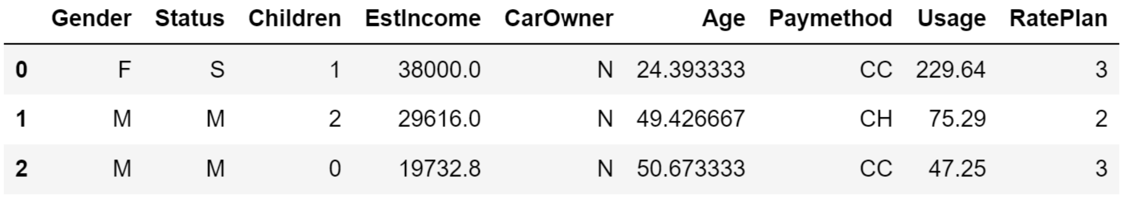
To gain a better understanding of what a rule is, we can examine an example. As mentioned earlier, rules are generated during the execution of R2N, and a single execution can produce thousands of new features consisting of multiple conditions.

The list of generated rules consists of different conditions separated by a logical OR, and within each one, there are multiple conditions separated by a logical AND.

For illustration purposes, let's consider a simple churn dataset. A churn dataset typically includes information about customers or subscribers of a business or service who have either canceled their subscriptions or stopped using the service. This dataset contains a variety of information about each customer, such as their income, age, usage, status, etc.

The target is the Churn feature, a binary column indicating whether the customer churned or not.

We want to use the R2N to predict which customers are at risk of churn, allowing businesses, in general, to take proactive measures to prevent churn and retain valuable customers.



	Gender	Status	Children	EstIncome	CarOwner	Age	Paymethod	Usage	RatePlan
0	F	S	1	38000.0	N	24.393333	CC	229.64	3
1	M	M	2	29616.0	N	49.426667	CH	75.29	2
2	M	M	0	19732.8	N	50.673333	CC	47.25	3

Table 1 : Simple Churn Dataset.

Using these features and executing R2N, the first two features obtained are:

$$\left(\mathbf{-4.28 \times 10^{-6} \cdot EstIncome - 0.0089 \cdot Age + 0.0092 \cdot Usage - 0.02625 \cdot RatePlan \geq 0}\right)$$

$$\vee$$

$$\left(\mathbf{-3.103 \times 10^{-6} \cdot EstIncome - 0.019 \cdot Age + 0.423 \geq 0}\right) \wedge \mathbf{(Gender = F)} \wedge \mathbf{(CarOwner = N)}$$

In this example, we have two possible features. The first consists of a single condition. If this condition is satisfied for a given row, it will be considered as true; otherwise, it will be considered as false. The second one, separated by an OR sign, is composed of three conditions. If all three conditions are satisfied, the rule will be considered as true; if fewer conditions are met, the rule will be considered as false.

4 Team Contributions: Implementation and Discussions

4.1 Step 1: Understanding of the project

The first part of our work involved reading, comprehending, and discussing the R2N paper. We conducted weekly meetings with our supervisor to review progress and ensure a reel understanding of the paper and the project.

In a second time, we began to work with the R2N code and tested it on various datasets. This allowed us to gain insights into the functioning of the method and examine the results, particularly the features generated after execution. These experiences contributed significantly to our team's understanding of the implementation and potential applications of R2N in the context of automatic feature engineering.

The results of the R2N was not really important to us as our goal was to improve other ML methods with the rules, considered as new features for the ML models, extracted from R2N.

4.2 Step 2: Automatically extract the features

All the examples below will be for the simple churn dataset given by IBM, for the sake of simplicity. Moreover, to test the accuracy of the different versions of this dataset, we used simple models such as DecisionTreeClassifier(), RandomForestClassifier(), KNeighborsClassifier(), LogisticRegression(), MultinomialNB(), and GradientBoostingClassifier() from the library Sklearn.

The final part of our contribution involved starting the code. As this is a research project, we were unsure whether the direction we were heading was the correct one. We had many discussions and debates with our supervisors to determine what to extract and how.

4.2.1 All the rules

Initially, we decided to extract all the rules as features. For example,

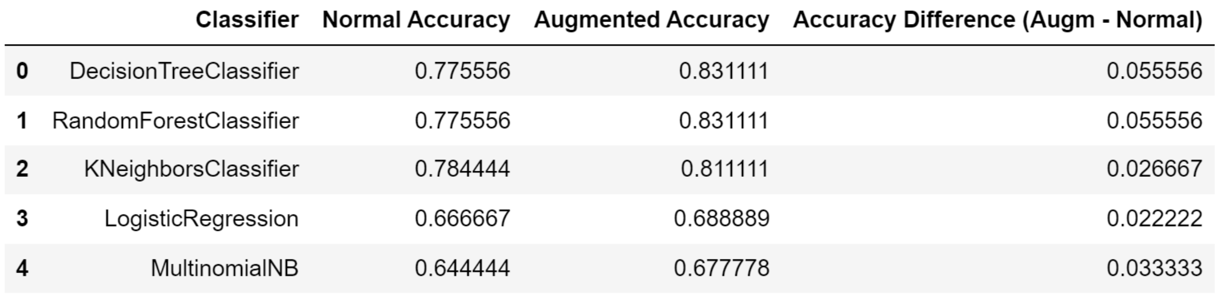
$$\left(\mathbf{-4.28 \times 10^{-6} \cdot EstIncome - 0.0089 \cdot Age + 0.0092 \cdot Usage - 0.02625 \cdot RatePlan \geq 0}\right)$$

This feature would become a new boolean feature called Rule0, and

$$\left(\mathbf{-3.103 \times 10^{-6} \cdot EstIncome - 0.019 \cdot Age + 0.423 \geq 0}\right) \wedge \mathbf{(Gender = F)} \wedge \mathbf{(CarOwner = N)}$$

would become another boolean feature called Rule1. After executing the R2N on this dataset, we obtained 12 features more than the original features.

This approach provided a promising start, as we already observed an increase in accuracy between 2 and 6 % depending on the dataset :



	Classifier	Normal Accuracy	Augmented Accuracy	Accuracy Difference (Augm - Normal)
0	DecisionTreeClassifier	0.775556	0.831111	0.055556
1	RandomForestClassifier	0.775556	0.831111	0.055556
2	KNeighborsClassifier	0.784444	0.811111	0.026667
3	LogisticRegression	0.666667	0.688889	0.022222
4	MultinomialNB	0.644444	0.677778	0.033333

Table 2 : Normal and Augmented Accuracy for Method 1.

We sensed that something was missing and, therefore, decided to experiment with alternative approaches.

4.2.2 True Rule Counter

We explored two distinct concepts for experimenting with new approaches. The initial idea involved introducing a True Rule Counter. This counter would tally the number of True rules in a dataset row consisting of 12 new features, resulting in a value of n. Once this counter was computed, we would eliminate all individual features and retain only the counter value.

Utilizing this method, we observed a modest improvement in accuracy, ranging between 1% and 2%. While this constitutes progress, it is not as significant as the gains achieved in the previous implementation.

4.2.3 Only Rule Dataset

An alternative strategy we considered involved enhancing accuracy by selecting only the new RuleX features and discarding all original ones. While we initially viewed this technique as more of an exploratory idea without high expectations, we were surprised to find that it resulted in a slight improvement in accuracy for some models, but not for others. Given that the changes in accuracy were not substantial, we decided to not continue with this approach.

4.3 Step 3: Final Technique

Upon examining the code, reviewing the various rules and conditions, and consulting with our supervisors, we identified a minor issue. We had been including complete rules, representing all conditions in a rule antecedent, even those with extremely simple conditions, such as **Gender = F**. This approach could result in an excess of unnecessary features, especially since models like decision trees already handle such rule-based decisions effectively. Consequently, we opted to separate all conditions and treat each of them as either a newly learned feature or an individual condition. A crucial nuance in this process was to eliminate the simplest conditions; we removed any categorical conditions that were basically just verifying the value of one feature.

With this approach, we generated a larger number of RuleX features, since rules originally consisted of multiple conditions. Adding numerous features might be concerning due to the potential increase in computations, but this method was proven successful in our case, yielding higher accuracy compared to previous techniques.

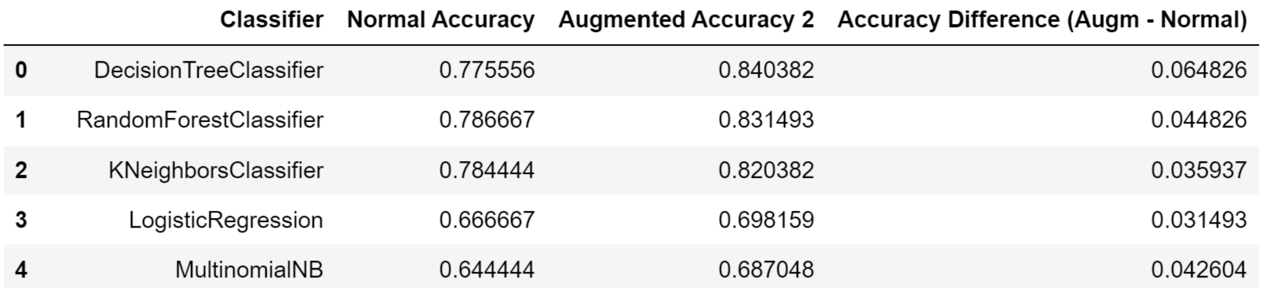
5 Results

In the context of this research project, we initially had no certainty about whether extracting rules as features would enhance the performance of other machine learning methods. Consequently, we were unsure if the accuracy would increase, decrease, or remain constant. Nevertheless, we were pleasantly surprised to discover that our efforts were fruitful, and we observed improvements in accuracy depending on the technique employed.

The first approach, extracting rules as single features, was already successful, yielding an accuracy increase between 2% and 5%.

The techniques involving rule counters and extracting only the conditions were not particularly effective, as some models showed improvements while others did not. Furthermore, any increase in accuracy, when present, was not particularly notable.

However, the final approach, which involved discarding simple conditions and retaining only the complex ones, proved to be highly successful. We observed an accuracy improvement ranging from 3.1% to 6.4%, as illustrated in the table below:



	Classifier	Normal Accuracy	Augmented Accuracy 2	Accuracy Difference (Augm - Normal)
0	DecisionTreeClassifier	0.775556	0.840382	0.064826
1	RandomForestClassifier	0.786667	0.831493	0.044826
2	KNeighborsClassifier	0.784444	0.820382	0.035937
3	LogisticRegression	0.666667	0.698159	0.031493
4	MultinomialNB	0.644444	0.687048	0.042604

Table 3 : Normal and Augmented Accuracy for Method 2.

6 Conclusion

In this case study, we investigated the potential of utilizing the R2N approach for learning compact transparent models using neuro-symbolic methods. Our primary objective was to examine whether extracting features from R2N could improve the performance of other machine-learning models while enhancing their interpretability and explainability.

To achieve this, we extracted the learned features of R2N and implemented them to assess any changes in the performance of classical machine-learning models, with a particular focus on examining changes in accuracy.

We presented various approaches to extract features from the R2N-generated rules and assessed their impact on the accuracy of different machine learning models. Our initial attempts showed promising results, with some improvements in accuracy. However, it was the final approach of discarding simple conditions and retaining only complex ones that demonstrated the most significant improvements.

Our findings suggest that automatic feature engineering using the R2N approach has the potential to enhance the performance of various machine learning models while maintaining transparency and interpretability. This improvement in both accuracy and explainability can lead to better decision-making and foster greater trust in AI-driven processes, particularly in high-stakes industries such as health, finance, and insurance.

Future work in this area could explore different techniques for extracting rules and features from R2N, as well as investigate the scalability of the approach to larger and more complex datasets. Additionally, it would be worthwhile to examine the impact of the R2N approach on other aspects of machine learning models, such as training time, computational efficiency, and robustness to adversarial examples. One other experiment could be to check the efficiency of the R2N on larger datasets with more complex features.

In conclusion, this case study demonstrates the potential of the R2N approach for learning compact transparent models using neuro-symbolic methods, paving the way for more interpretable, accurate, and trustworthy AI systems in various domains.

7 Future Work

At the end of the project, we thought of a final idea to further enhance feature extraction. This idea involves selecting only the features that would optimize the machine learning model we aim to improve. For example, if we want to enhance a decision tree, we would only select rules that could positively impact the accuracy of this one. This approach could be promising, but it would require a deeper understanding of the inner workings of each algorithm to effectively extract the most efficient rules.

One potential improvement for R2N is to learn decision trees instead of rule sets. However, our current work on new feature implementation is optimized for rule sets and is not currently suitable for other machine learning models.

To address this limitation, one possible enhancement for R2N would be to encode decision trees, which could potentially allow for better integration with other machine-learning models beyond rule sets.

8 Acknowledgments

This work has been done in collaboration with IBM, supervised by Greger Ottosson and Yusik Kim for the final project of the 1st year of MSc Data Science and Artificial Intelligence for the Class Case Studies, under the guidance of Charles Bouveyron.

References

1. Kusters, R., Kim, Y., Collery, M., de Sainte Marie, C., Gupta, S.: Differentiable rule induction with learned relational features. arXiv preprint arXiv:2201.06515 (2022), https://arxiv.org/abs/2201.06515

2. IBM. "Symlearn: r2n." GitHub, IBM, 2021, https://github.com/IBM/symlearn/tree/main/symlearn/algorithms/r2n.