Integrating Fuzzy Logic into Deep Symbolic Regression

Wout Gerdes
University of Amsterdam
Amsterdam, The Netherlands
woutgerdes@gmail.com

Erman Acar
IvI & ILLC, University of Amsterdam
Amsterdam, The Netherlands
erman.acar@uva.nl

Abstract

Credit card fraud detection is a critical concern for financial institutions, intensified by the rise of contactless payment technologies. While deep learning models offer high accuracy, their lack of explainability poses significant challenges in financial settings. This paper explores the integration of fuzzy logic into Deep Symbolic Regression (DSR) to enhance both performance and explainability in fraud detection. We investigate the effectiveness of different fuzzy logic implications, specifically Łukasiewicz, Gödel, and Product, in handling the complexity and uncertainty of fraud detection datasets. Our analysis suggest that the Łukasiewicz implication achieves the highest F1-score and overall accuracy, while the Product implication offers a favorable balance between performance and explainability. Despite having a performance lower than state-of-the-art (SOTA) models due to information loss in data transformation, our approach provides novelty and insights into into integrating fuzzy logic into DSR for fraud detection, providing a comprehensive comparison between different implications and methods.

Keywords

Deep Symbolic Regression, Fuzzy Logic, Fraud Detection, Explainable AI

ACM Reference Format:
Wout Gerdes and Erman Acar. 2024. Integrating Fuzzy Logic into Deep Symbolic Regression. In . ACM, New York, NY, USA, 9 pages.

1 Introduction

Credit card fraud poses a significant and growing challenge for financial institutions, amplified by the advent of innovative technologies such as contactless payment [8]. Global losses due to credit card fraud were estimated at $32.39 billion in 2020 and are projected to exceed $40 billion by 2027 [18]. The Covid-19 pandemic further accelerated the shift from cash to cashless transactions, intensifying the issue of credit card fraud. To protect customers from fraudulent activities, banks deploy Fraud Detection Systems (FDS) to automatically flag and block suspicious transactions in real-time. Significant advancements in these systems have been achieved through improvements in data quality and the enhanced use and performance of Artificial Intelligence (AI) and Deep Learning (DL) techniques [6]. Although DL has demonstrated exceptional performance in classification accuracy [2], a major limitation is its lack of explainability. [16]. This "black box" nature has hindered the adoption of AI in financial settings, where decisions must be transparent and explainable. Explainable Artificial Intelligence (XAI) offers a potential solution to this problem.

Despite the growing interest in XAI, the intersection of fraud detection and XAI remains underexplored. Recent efforts have taken various approaches to bridge this gap. One approach employs XAI methods to interpret Machine Learning (ML) models post-training using techniques such as SHAP (Shapley Additive Explanations) or LIME (Local Interpretable Model-Agnostic Explanations), which have shown only modest improvements in user trust [13].

Another promising approach involves leveraging Symbolic Regression (SR), which seeks to extract closed-form expressions to describe underlying patterns in the data. These expressions are inherently explainable, resolving transparency issues. To advance SR, Petersen et al. [20] combined SR with Reinforcement Learning (RL), resulting in Deep Symbolic Regression (DSR). DSR employs a recurrent neural network (RNN) trained with deep reinforcement learning, where the reward is task-specific, producing expressions tailored to specific problems. These closed-form expressions show high predictive power and transparency, making them a viable solution to many of the previously mentioned issues. DSR utilizes a library of tokens representing features, constants, or mathematical operators to generate expressions. The DSR framework creates a list of tokens subject to constraints, optimizing them using the RNN based on the reward function. An extension to DSR by Visbeek et al. successfully applied DSR to the fraud detection domain, resulting in Deep Symbolic Classification (DSC) [25]. DSC adapts DSR for classification tasks and uses the F1-score as the reward metric, offering competitive predictive performance with improved explainability.

In this paper, we propose extensions to the DSR framework by integrating fuzzy logic. Fuzzy logic, based on the fuzzy set theory by Zadeh [27], categorizes reasoning into multiple levels, similar to human reasoning. Unlike strict classifications, fuzzy logic handles uncertainty and vagueness, making it suitable for real-world complexities. For instance, a person is not simply tall or short but can manifest varying degrees of tallness. Fuzzy logic facilitates smooth transitions between such degrees. In DSC, the output is a closed-form mathematical expression [25].

Logical implications provide intuitive explanations, since they naturally represent *general rules* e.g., *If transaction amount is high and receiver balance is low then fraud is the case.*

These expressions are more intuitive as they mirror human reasoning. Various formulas derive fuzzy implications from fuzzy sets, and fuzzy logic provides a natural medium for expressing the vagueness within this logical structure. For instance, vague expressions such as amount being high might have a varying degree of truth

XAI FIN'24 Workshop, November 2024, New York City, NY, USA
© 2024 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 978-x-xxxx-xxxx-x/YY/MM

from 0 being false, to 1 being absolutely true. This leads us to the main research question:

How can we integrate fuzzy logic into deep symbolic regression for fraud detection?

To address our main research question, we elaborate on the following specific sub-research questions:

(1) What specific fuzzy logic implications are most effective in enhancing the model's ability to handle the inherent complexity and uncertainty in fraud detection datasets?
(2) What specific choice of fuzzy implication is most effective in DSR?
(3) Does the fuzzy logic oriented DSR provide intuitive expressions that are easy to interpret?
(4) What is the trade off between the size of the fuzzy logic formula and performance metrics?

To assess performance, we compare the proposed framework against current state-of-the-art algorithms as detailed in Table 2. Performance evaluation is conducted using accuracy, which is common in real-life fraud detection, and the F1-score, which addresses the inherent imbalance in credit card fraud data. The popular PaySim dataset is utilized [3], offering a controlled environment for comparison. This synthetically generated dataset contains no privacy concerns, as it lacks personally identifiable information (PII). In addition to performance assessment, the explainability of the rule expressions are evaluated. Furthermore, we employ a Pareto front [15] to balance predictive performance and explainability, optimizing the complexity of expressions based on the factors and performance, identifying the most suitable expressions for the task.

2 Related Work and Background Knowledge

2.1 Fraud Detection

Prior research in fraud detection has explored various ML and DL algorithms to address the challenges associated with different types of fraud. Traditional ML approaches, such as Random Forests, k-Nearest Neighbour, Naive Bayes, Logistic Regression, and XGBoost have been extensively studied due to their efficiency in handling structured data [2, 5, 10]. However, these methods often struggle with class imbalance and lack explainability, leading researchers to investigate novel techniques. Recent advancements in DL, particularly convolutional neural networks (CNNs), have shown promise in improving fraud detection by leveraging intricate patterns present in transaction data [2]. Furthermore, studies have emphasized the importance of considering factors like data imbalance, false positives, and the financial implications of fraud detection systems when designing effective frameworks. Recently, the black box nature of these ML and DL algorithms has been critiqued. The introduction of the Artificial Intelligence Act [19] has formalized the importance of governance in AI applications, highlighting the need for XAI both in general and in fraud detection in particular.

2.2 Explainable AI

While recent developments in AI promises stellar predictive performance, it often lacks explainability compared to simpler models. XAI aims to increase transparency and explainability of AI systems by providing insights into the decision-making process, model behavior, and factors influencing predictions. This is particularly crucial in sensitive applications such as healthcare, finance, and security, where stakeholders require clear explanations for AI-driven decisions to ensure trust and accountability [16]. These sensitive applications often face regulatory constraints, such as the EU's General Data Protection Regulation (GDPR), requiring enhanced explainability before adopting advanced AI applications [9]. XAI can generally be classified into two categories: *potentially interpretable models*, which have some level of inherent transparency (e.g., linear regression, decision trees, k-Nearest Neighbors), and *post-hoc techniques*, which enhance explainability after model training (e.g., Local Interpretable Model-agnostic Explanations (LIME) or SHapley Additive Explanations (SHAP)) [26].

2.3 Symbolic Regression

Symbolic regression (SR) is a method to obtain mathematical expressions that accurately capture information from data. Unlike neural networks, SR aims to provide transparent closed-form expressions, modeling information from the underlying dataset in a form that is more easily interpretable by humans. Given a dataset $(\mathbf{X}, \mathbf{y})$, where each $\mathbf{X}_i$ has inputs $\mathbf{X}_i \in \mathbb{R}^n$ and response $y_i \in \mathbb{R}$, SR attempts to generate a function $f : \mathbb{R}^n \rightarrow \mathbb{R}$ that best fits the dataset, where f is a closed-form mathematical expression [17]. The formula generated by symbolic classification is much more compact, and tells about the relation between features in contrast to rules generated by decision trees which are often verbose and flat in relation to other features.

Deep symbolic regression (DSR) combines symbolic regression with deep learning techniques, such as RNN's to extract underlying patterns and relationships from data. Unlike traditional symbolic regression, which relies only on predefined mathematical forms, DSR can learn from data and adapt its expressions to capture intricate dependencies, resulting in more accurate and flexible models. The use of an RNN allows for a gradient-based approach to optimize these closed-form expressions, leveraging the performance of certain expressions as the reward for the RNN. With training, the RNN can optimize the reward function per specific mathematical operator, re-evaluating the underlying distribution of each iteration [20]. It utilizes mathematical operators from a predefined library λ, which includes basic operators (e.g.,$\div$, $\cdot$), constants and features from the dataset. This library enables the generation of closed-form expressions using all available features and performing mathematical operations [20, 25]. Constraints are applied to prevent certain mathematical expressions from being generated, and these constraints can be tailored to improve the DSR framework's performance.

2.4 Fuzzy Logic

Fuzzy logic allow for expressing vagueness and uncertainty which can then be incorporated into machine learning [24]. By using fuzzy sets, variables are no longer strictly categorized but assigned to multiple categories. This flexibility allows fuzzy logic to generate fuzzy implications in various forms, which are particularly useful in fields like symbolic regression. Different formulas can deduce fuzzy implications from fuzzy sets. In this paper, we focus on Gödel, Product, and Łukasiewicz implications.

Another approach to enable fuzzy logic applications is fuzzy clustering, relying on unsupervised learning instead of domain experts. Fuzzy clustering is the process of acquiring k amount of categories, or clusters, per feature of the dataset. Different from human reasoning, fuzzy clustering attempts to gather the categories from the underlying data. Suppose we have a set of categories, $\{A_1, A_2, \ldots, A_n\}$, and a set of objects $\{x_1, x_2, \ldots, x_n\}$, and a degree of x_i belonging to A_j called $\mu_{A_j}(x_i)$, where $0 < \mu_{A_j}(x_i) < 1$. If $\mu_{A_j}(x_i) = 0$, then x_i is not a member of A_j, and similarly if $\mu_{A_j}(x_i) = 1$, then x_i is a member of A_j [4]. Fuzzy clustering is the process of acquiring categories or clusters, $\{A_1, A_2, \ldots, A_n\}$, per feature of the dataset using clustering algorithms such as k-means [1]. Each object $\{x_1, x_2, \ldots, x_n\}$ is assigned to a category, based on its highest membership value, in turn enabling the use of fuzzy logic.

In fuzzy logic, we use strong negation to complement a fuzzy set. If a is the membership value of an element in a fuzzy set, the strong negation is defined as $N(a) = 1 - a$

T-norms (triangular norms) model the conjunction (AND) operation in fuzzy logic and generalize the intersection operation in classical set theory.

T-conorms (triangular conorms) model the disjunction (OR) operation in fuzzy logic and generalize the union operation in classical set theory [11].

In fuzzy logic, implications play a crucial role in reasoning. There are two primary types of implications [11]:

- **S-implications** are derived from the residuation principle and are defined using T-norms.
- **R-implications** are defined using a residual function and T-conorms.

For the Gödel, Product, and Łukasiewicz implications, the specific formulas can be found in Table 1.

These fuzzy implications are required when working with fuzzy sets to capture underlying patterns from the data.

2.5 Integrating Fuzzy Logic in Symbolic Regression

Symbolic regression traditionally focuses on generating mathematical expressions that best fit the data. However, using logical operators instead of purely mathematical ones can improve explainability and enhance alignment with human reasoning. The language of logic is more natural for explaining phenomena, as logical operators like "and", "or", and "not" are closer to daily language. For instance, rules in the form of implications can be easily understood as "if-then" statements, which are common in human reasoning. Historically, logic stems from natural language, originating with syllogisms by Aristotle, and abstracting important aspects of human reasoning [7]. The Łukasiewicz implications and logic are essentially a modern reconstruction of these syllogisms [14]. This historical context indicates the similarities of logic with how humans understand and describe the world [22]. Therefore, fuzzy logic, which incorporates similar logical operators, offers a more intuitive and explainable framework compared to purely mathematical expressions. Fuzzy logic is particularly suitable for application in symbolic regression because it retains the numerical foundation of symbolic regression while leveraging the explainability of logical expressions. By incorporating fuzzy sets and implications, we can handle vagueness and uncertainty more effectively while aligning the explanation with human reasoning.

3 Methodology

3.1 Model

We adapt DSR to integrate fuzzy logic, extending the existing DSR approach by integrating fuzzy implications into the expression generation and evaluation stages. Traditionally, the DSR framework utilizes a generated traversal of tokens to create a binary syntax tree, which is then used to generate expressions of these tokens and test their performance on the dataset. Our approach augments this framework to integrate fuzzy logic implications [20].

To facilitate the use of fuzzy logic in our model, we transform relevant features into fuzzy sets, capturing degrees of membership such as "very large" or "somewhat small." This transformation allows the model to better manage the uncertainty and variability present in financial transactions. Fuzzy logic formulas, as shown in Table 1, are employed in generating the expressions.

We test and compare three different fuzzy logic implications: Gödel, Product, and Łukasiewicz. Fuzzy membership functions determine the degree to which a transaction belongs to each class, providing a spectrum of likelihood rather than a binary decision. We implement and evaluate the S-implication, together with the T-norm, T-conorm, and strong negator for these three fuzzy implications. Note that the S-implications, in contrast to R-implications, are the ones that generalize material implications, since they use the T-conorm in their construction [20]. Therefore, we utilize the S-implication and disregard the R-implication. Additionally, we compare a combination of all three fuzzy logic implications, resulting in a comparison between four function sets.

Our implementation is flexible, allowing the use of any S-implications, T-norms, T-conorms, and strong negators. To research the impact of implementing the S-implication as the root of the expression, we modify the DSR package. We add a constraint ensuring the S-implication is always the parent of a T-norm, T-conorm, or strong negator. This ensures the S-implication is never a child of any other function. However, due to the nature of syntax trees generated from functions and features, not every traversal includes the S-implication. To address this, the token for the S-implication was added to the traversal at index 0 if it was not yet included. If the traversal already contains the S-implication token, we swap its position with the token at index 0.

Through reinforcement learning techniques, we train the model to optimize for F1-score, following the research of [25]. During training, a RNN produces mathematical expressions and evaluates them based on their ability to describe the dataset, a measure known as "fitness". This fitness score links to a reward, which trains the RNN through a risk-seeking policy gradient algorithm. The risk-seeking policy gradient algorithm is defined in Equation 1. This risk-seeking policy is utilized to improve the model's functioning. The risk-seeking policy focuses on improving the reward of the top ϵ fraction of samples, disregarding samples below the threshold. This method aims to enhance the highest performing expressions, even if it results in a lower average performance [20]. We use the F1-score as the main reward function, although we also test the F2-score as the reward function. The policy gradient algorithm

Operation Type	Gödel	Product	Łukasiewicz
T-norm	$T_{\text{gd}}(a,b) = \min(a,b)$	$T_{\text{pr}}(a,b) = a \cdot b$	$T_{\text{lk}}(a,b) = \max(a+b-1,0)$
T-conorm	$S_{\text{gd}}(a,b) = \max(a,b)$	$S_{\text{pr}}(a,b) = a+b-a\cdot b$	$S_{\text{lk}}(a,b) = \min(a+b,1)$
S-implication	$I_{\text{gd}}(a,c) = \max(1-a,c)$	$I_{\text{pr}}(a,c) = 1-a+a\cdot c$	$I_{\text{lk}}(a,c) = \min(1-a+c,1)$
R-implication	$I_{\text{gd}}(a,c) = \begin{cases}1 & \text{if } a \le c\\ c & \text{otherwise}\end{cases}$	$I_{\text{pr}}(a,c) = \begin{cases}1 & \text{if } a \le c\\ \frac{c}{a} & \text{otherwise}\end{cases}$	$I_{\text{lk}}(a,c) = \min(1-a+c,1)$

Table 1: T-norms, T-conorms, S-implications, and R-implications for Gödel, Product, and Łukasiewicz [24].

adjusts the probabilities of sampling an expression according to its corresponding reward. Specifically, expressions with higher fitness scores receive higher rewards, which in turn increase their likelihood of being sampled in subsequent iterations. This method ensures that the RNN progressively favors expressions that better fit the data.

$$J_{risk}(\theta;\epsilon) \equiv \mathbb{E}_{\tau\sim p(\tau|\theta)}[R(\tau) \mid R(\tau) \ge R_\epsilon(\theta)] \tag{1}$$

By integrating fuzzy logic into DSR, our approach aims to improve the robustness and explainability of fraud detection systems.

3.2 Data

In this research, we utilize the popular and well-researched synthetically generated dataset, PaySim [3]. This dataset consists of simulated transactions based on a distribution of proprietary real transactions. Due to inherent privacy concerns with real-life datasets, this is the most suitable alternative available. PaySim contains over 6 million financial transactions over a period of one month, with a fraud rate of 0.13%. Given the true/false rate of 0.13/99.87, this is a highly imbalanced dataset and should be handled accordingly.

The dataset includes the following features [25]:

- **step** - unit of time; one step corresponds to one hour of time,
- **type** - a categorical feature with values: cash-in, cash-out, debit, payment, or transfer,
- **amount** - amount of the transaction,
- **nameOrig** - name of the customer,
- **oldbalanceOrg** - customer's balance before the transaction,
- **newbalanceOrig** - customer's balance after the transaction,
- **nameDest** - name of the recipient,
- **oldbalanceDest** - recipient's balance before the transaction,
- **newbalanceDest** - recipient's balance after the transaction,
- **isFlaggedFraud** - an indicator of whether the transaction has been flagged as fraudulent in the simulation,
- **isFraud** - an indicator of the transaction being legitimate or fraudulent.

The isFraud feature represents the target variable, which serves as the ground truth for training our model.

Per the creators' request [3], the features oldbalanceOrig, newbalanceOrig, oldbalanceDest, newbalanceDest should not be used directly. This is because fraudulent transactions are cancelled, meaning the balances of the originator and recipient remain unchanged. Using these features directly would result in artificially high accuracy, failing to accurately classify real fraud detection cases.

Instead, these features provide insights into typical transaction behavior. To avoid direct usage, we use imputation techniques to generate new features that capture balance changes without relying on the original balance columns. Specifically, we engineer features such as newbalanceDest, oldbalanceOrig based on the transaction amount plus oldbalanceDest, newbalanceOrig. Hourly, daily, and monthly features are also derived from the step feature. The feature is_workday is created based on the transaction volume distribution over a week, assuming the five highest volume days correspond to weekdays and the lowest two to weekends.

Categorical features are one-hot encoded, converting textual categorical data into binary columns, making them suitable for DSR. Although these features are not fuzzy sets, they are still useable for this research.

To improve model robustness and generalizability, we introduce random noise into the dataset. For each feature column, Gaussian noise is added using a method where the standard deviation of the target column is multiplied by a noise level parameter of 0,05. This noise simulates real-world data imperfections, preventing the model from overfitting to clean training data. This approach enhances the model's resilience and its ability to generalize to new, unseen data.

We also incorporate transaction history by creating rolling average and rolling max features for the amount column. Using 3 and 7 transaction windows, we calculate the mean and max transaction amounts for each recipient (nameDest), helping to identify discrepancies based on transaction history.

To transform the dataset into fuzzy sets, we divide the dataset into percentiles. Each feature is represented as a fuzzy set according to these percentiles. We calculate the 20th, 40th, 60th, and 80th percentiles for each feature, segmenting the data into five ranges. For each feature value, we assign a fuzzy membership value based on its percentile. For example, values less than or equal to the 20th percentile receive a membership value of 0.2, the 20th to 40th percentile values receive 0.4, and so on. This transformation is applied to all relevant features, ensuring they are represented as fuzzy sets, facilitating the integration of fuzzy logic into our model [21]. The features used are all either transformed into fuzzy sets, or binary values.

3.3 Hyperparameter Tuning

To prepare the dataset for machine learning tasks, it is necessary to partition it into distinct sets for training, validation, and final testing. This process ensures that the model's performance can be accurately assessed and optimized. Due to computational constraints and efficiency considerations, hyperparameter tuning is performed on a subset of the data, while the final testing is conducted on another separate subset.

Initially, a sample of one million transactions is extracted from the dataset. It is essential to ensure that the balance of fraudulent and non-fraudulent transactions in this sample accurately reflects that of the full dataset. This maintains the representativeness of the sample and ensures that any insights or models derived from it remain applicable to the entire dataset. For hyperparameter tuning, we partition the sample into training and validation sets using a 70/30 split ratio, respectively. We perform this split in a stratified manner, preserving the balance of fraudulent and non-fraudulent transactions across the sets. Stratification helps mitigate the risk of introducing biases during hyperparameter tuning, ensuring a comprehensive evaluation of the performance across different classes.

A common concern when utilizing DSR is its sensitivity to hyperparameters [20]. Although computational constraints limited extensive testing and optimization of hyperparameters, key parameters were tuned. Important hyperparameters tested are:

- **Sigmoid threshold** - defines the cutoff value for the sigmoid activation function, squashing the output into a range between 0 and 1.
- **Learning rate** - controls the step size at each iteration while moving towards a minimum of the loss function.
- **Entropy weight** - scales the entropy term in the loss function, promoting exploration by encouraging a more uniform probability distribution.
- **Risk factor** (ϵ) - used by the risk-seeking policy gradient.
- **Batch size** - number of training samples used in one iteration of model training.
- ***N* samples** - the total number of samples to generate.

The higher the value for N samples, the more batches DSR uses to improve the found expressions. For each run, a sample size of 200,000 was used. This ensures minimal progress in the last quarter of iterations, balancing computational feasibility and optimal performance.

We perform hyperparameter tuning on the one million transaction subset. Each hyperparameter setting was tested on 16 runs, seeded from seed 0 to seed 15 to ensure comparability. The average F1 score and highest F1 score were compared across all hyperparameter settings. The best performing hyperparameter configuration was chosen for the model.

For the final evaluation of the model's performance, a separate sample comprising two million transactions was extracted from the dataset. Similar to the initial sample, the balance of fraudulent and non-fraudulent transactions was maintained. The two-million transaction subset is split into training and test sets using a ratio of 70/30. This split was also stratified based on the fraud column, ensuring proportional representation of fraudulent and non-fraudulent transactions in both sets.

To eliminate potential biases introduced by the ordering of the dataset, both the subsampling and splitting processes were accompanied by shuffling. Shuffling ensures that each subset is a random representation of the original dataset, preventing any unintentional patterns or correlations from influencing the model's performance. Additionally, to ensure reproducibility, both the shuffling and splitting procedures were performed using a specific seed value, allowing for consistent results across multiple runs and ensuring the reproducibility of the experimental setup.

3.4 Evaluation

Evaluation of the proposed framework can be approached from two perspectives: classification performance and explainability performance. Classification performance represents how accurately the framework detects fraudulent transactions. To correctly assess this performance, we use two different measures: accuracy and F1 score. Accuracy is a standard gauge for comparing the proposed framework to current industry standards. However, due to the highly imbalanced nature of our dataset, accuracy can be misleading. A high accuracy is easily achieved by marking all transactions as non-fraudulent, making it less meaningful in our context. Therefore, the F1 score is also used to address this imbalance problem [12].

$$\text{F1-score} = \frac{2 \cdot P \cdot R}{P + R} \quad (2) \qquad \text{F2-score} = \frac{5 \cdot P \cdot R}{4 \cdot P + R} \quad (3)$$

The F1 score is the harmonic mean of precision and recall. It balances the importance of both metrics, which is crucial for our imbalanced dataset. The RNN will also be optimized for the F1 score. In highly imbalanced datasets, recall is often more important than precision. Hence, the F2 score [12] is also investigated, where recall is twice as important as precision.

Explainability of the framework is assessed using a Pareto front. A Pareto front evaluates the trade-off between expression complexity and predictive performance. Operators in the DSR framework are weighted based on their pre-determined complexity. For instance, the complexity of T-norm, T-conorm, and strong negator are set to one, while S-implication is set to two. The complexity of a generated expression is the sum of its components' complexities. The Pareto front represents the set of solutions where no other expression achieves superior performance for the corresponding complexity [23]. This helps determine a balance between performance and explainability, resulting in a final expression that is both effective and explainable.

Comparing performance is challenging due to the novelty of the proposed method, and the reduction in information in the features by transforming them into fuzzy sets. Previous research on this dataset uses k-NN, SVM, RF, and XGBoost [10], while [25] used DSC. These methods are used for rough comparison to evaluate if integrating fuzzy logic into the DSR framework might be useful.

Table 2: SOTA performance metrics

Method	F1	Accuracy	Precision	Recall
k-NN	0.16	0.99	0.087	0.87
SVM	0.47	0.99	0.95	0.31
RF	0.84	0.99	0.92	0.78
XGBoost	0.84	0.99	0.88	0.81
DSC	0.78	0.99	0.95	0.67

Sources: k-NN, SVM, RF, XGBoost [10], DSC [25]

4 Results

4.1 Performance of Different Implications

The results compare the performance of the Gödel, Product, and Łukasiewicz implications in the DSR framework. Table 3 shows the highest score achieved by any expression found across 16 seeds, and Table 4 presents the average performance of the best-performing expressions per seed. Each seed utilizes a different slice of the sample, resulting in different formulas and expressions. All expressions are evaluated on the test set.

Table 3: Performance metrics of the best single expression per method. Bold indicates the best score in each column.

Method	F1-score	Accuracy	Precision	Recall
Product	0.163	**0.998**	**0.161**	0.165
Łukasiewicz	**0.193**	0.997	0.154	**0.256**
Gödel	0.073	0.996	0.050	0.131
Combined	0.177	0.996	0.126	0.295

Table 3 highlights the best single expression's performance per method. The Łukasiewicz implication achieves the highest F1-score and recall, while the Product implication has the highest precision and accuracy. Table 4 presents the average performance of the best expressions per seed. The Łukasiewicz implication leads with the highest average F1-score and accuracy, indicating robust performance across both the best expression and average expression.

Table 4: Average performance of the best expression per seed. Bold indicates the best score in each column.

Method	F1-score	Accuracy	Precision	Recall
Product	0.157	0.997	0.183	0.141
Łukasiewicz	**0.169**	**0.998**	**0.194**	0.164
Gödel	0.072	0.996	0.131	0.050
Combined	0.158	0.998	0.152	**0.180**

Currently, the best performing expression is Equation 4, with performance results in Table 3. The best-performing fuzzy logic expression uses the Łukasiewicz implication. It involves nested implications and norms, leveraging features indicating balance of the receiving address both before and after the transaction, the type of transaction, and the maximum transaction amount of the past 7 transactions to predict fraud. Abbreviations are used to improve readability of the formula. *NBD* corresponds to newbalanceDest, *OBD* is oldbalanceDest, *MD7* is maxDest7 and *C_O* is type_CASH_OUT. For formal details of fuzzy logic, we refer to [11].

$$\begin{aligned} \forall x, y \quad & \neg\Bigg(\bigg(\Big(\Big(\neg\Big(\neg(\text{NBD})\Big) \otimes (\text{NBD})\Big) \otimes \Big(\Big(\text{OBD}\Big) \\ & \to \Big(\neg\Big(\Big(\text{OBD}\Big) \to (\text{MD7})\Big)\Big)\Big) \otimes (\text{C\_O})\bigg) \to (\text{OBD})\Bigg) \end{aligned} \tag{4}$$

The new balance of the destination account (NBD) compared with its old balance (OBD) can indicate anomalies. For example, if the new balance significantly exceeds typical values following a series of large transactions, it could be suspicious.

The nested implications involving the old balance of an account (OBD) and the maximum of the past 7 transactions (MD7) evaluate how recent large transactions relate to the current balance. If the OBD does not logically align with past transaction patterns, it might signal fraud.

4.2 F2 Score

We utilize the Łukasiewicz implication for further research into the usage of F1 versus F2 scores as the reward function, as shown in Table 5. We train the model using F2 as a reward score and compare it against the model trained using the F1 as the reward score, testing whether prioritizing recall over precision improves performance. In Table 5, we compare models trained with F1 and F2 scores as the reward function. Prioritizing recall (F2 score) slightly improves accuracy but decreases the overall F1-score, indicating a trade-off between precision and recall. Utilising F2 score results in a high average precision, which is contrary to expectations and will be discussed further in Section 5.

Table 5: Comparison of F1 versus F2 score.

Score Type	F1-score	Accuracy	Precision	Recall
F1 Score (best)	0.193	0.997	0.154	0.256
F2 Score (best)	0.135	0.998	0.150	0.123
F1 Score (avg)	0.169	0.998	0.194	0.164
F2 Score (avg)	0.126	0.991	0.493	0.082

4.3 Constrained Implementation

We also test the Łukasiewicz implication when it is constrained as the root node and compare this against the unconstrained implementation, as shown in Table 6. The model is trained while being constrained such that the S-implication occurs in the root node of the expression, and compare this against the unconstrained implementation. Abbreviations are used to improve readability, where Unc corresponds to the unconstrained implementation, and Con corresponds to the constrained implementation.

Table 6: Performance comparison of unconstrained versus constrained implementation.

Constraint	F1-score	Accuracy	Precision	Recall
Unc (best)	0.193	0.997	0.154	0.256
Con (best)	0.163	0.998	0.161	0.165
Unc (avg)	0.169	0.998	0.194	0.164
Con (avg)	0.133	0.996	0.195	0.138

Table 6 highlights that the unconstrained implementation generally outperforms the constrained ones, suggesting that allowing more flexibility in the structure can lead to better performance.

5 Discussion

Performance Comparison. The performance analysis indicates that the Łukasiewicz and Product fuzzy logic implications perform comparatively well, with the Łukasiewicz implication achieving the highest F1-score and overall accuracy. In contrast, the Gödel implication significantly lags behind, highlighting the importance of selecting the appropriate fuzzy logic implication for fraud detection tasks. The combination of different fuzzy logic implications does not yield performance improvements, suggesting that the unique characteristics of each fuzzy logic implication are best utilized independently. This finding underscores further exploration of individual fuzzy logic implications to enhance their performance.

Performance against State-Of-The-Art. The performance of all fuzzy logic implications within the DSR framework is lower than SOTA methods such as k-NN, SVM, RF, and XGBoost, as detailed in Table 2. It is important to note that although these methods are trained on the same dataset, different transformations are used. In this research, all data is compressed into values between [0, 1]. In the research regarding other methods, the data is kept in its original format, ensuring more information is retained in the dataset. Therefore, a comparison between these performance metrics should be considered more of a proof of concept for integrating fuzzy logic into the DSR framework then an accurate performance comparison. Furthermore, this outcome was anticipated due to the transformation of transaction data into fuzzy sets, which results in substantial information loss. While the current approach compresses data into five categories based on percentiles, this simplification inherently limits the information density of the original data. Enhancements in the way fuzzy sets are defined, such as through unsupervised fuzzy clustering or using transformations like logarithmic scaling, could help retain more information and improve performance.

Analyzing the best-performing expression from DSC [25], we see that features like *maxDest7* and *amount* capture information about transaction values, which fuzzy sets currently fail to represent adequately. For instance, the *maxDest7* feature is subtracted from *amount* to highlight anomalies in the transaction history. This nuanced relationship is lost when data is compressed into fuzzy sets. By utilizing fuzzy clustering, we might better capture the intricate balance between current and previous transactions, potentially enhancing the model's performance.

F1 versus F2 Score. Training the DSR framework with the F2 score as the reward function demonstrates the impact of emphasizing recall over precision. In fraud detection, it is crucial to identify as many fraudulent transactions as possible, even if it means a higher false-positive rate. The F2 score, which weighs recall twice as much as precision, aligns with this priority. As shown in Table 5, prioritizing recall (F2 score) slightly improves accuracy for the best performing expression, but decreases the overall F1-score. On average, utilising the F2 score as the reward function yields worse performance compared to the F1 score.

Interestingly, utilizing the F2 score results in a very high average precision, contrary to expectations. The expectation when utilizing the F2 score was to achieve higher recall at the expense of precision. This is because the F2 score places more emphasis on recall, encouraging the model to identify as many fraudulent transactions as possible. Typically, this approach would result in a larger number of false positives, thereby reducing precision. Due to the nature of DSR, expressions can be found that achieve a high F1 score, but are not in line with the expectations.

The best performing expression extracted using the F2 score as a reward function can be seen in Expression 5. OBD stands for oldBalanceDest, TF stands for type_TRANSFER and MD7 represents maxDest7. The T-norm, T-conorm and S-implication all correspond to the Łukasiewicz implications.

$$\neg\Big(\oplus\Big(\text{OBD}, \rightarrow\Big(\text{TF}, \text{MD7}\Big)\Big)\Big) \tag{5}$$

Constrained Implementation. Constraining the implementation of fuzzy implications results in worse performance compared to the unconstrained version. This finding suggests that the flexibility to explore a wide range of expression structures is more suitable for optimal performance. By constraining the model to use specific logical rules, such as placing the S-implication at the root node, the model's ability to discover effective patterns is limited. This insight challenges the hypothesis that strict logical constraints always lead to better performance, highlighting the importance of allowing the model to explore diverse expression configurations. However, the expression still generates interesting insights into the implementation of fuzzy logic into the DSR package. Expression 6 shows the best performing expression generated with the given constraints. OBD stands for oldBalanceDest, NBD stands for newBalanceDest and TF stands for type_TRANSFER. The T-norm, T-conorm and S-implication all correspond to the Łukasiewicz implications.

$$\begin{aligned}\rightarrow\Big(\oplus\Big(\text{TF}, \neg\Big(\text{TF}\Big)\Big), \otimes\Big(\otimes\Big(\neg\Big(\neg\Big(\text{NBD}\Big)\Big),\\ \otimes\Big(\neg\Big(\text{OBD}, \neg\Big(\text{OBD}\Big)\Big)\Big), \text{NBD}\Big)\Big)\end{aligned} \tag{6}$$

Explainability and Trade Off. Fuzzy logic offers intuitive rule-based expressions that are easier to understand compared to traditional black-box models. However, there is a trade-off between the complexity of these expressions and their predictive performance. More complex expressions can capture nuanced patterns in the data but may become less explainable. The Pareto front, shown in Figure 1, helps visualize this trade-off, allowing for the selection of expressions that balance performance and complexity. This balance is crucial for ensuring that the fraud detection system is both effective and explainable.

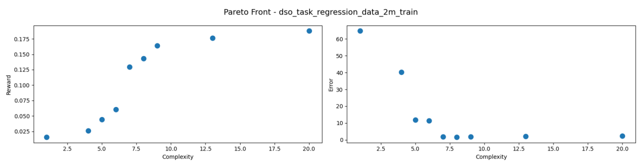

Figure 1: Pareto front showcasing the complexity and performance curve for the Łukasiewicz implication

The best-performing formula using the Łukasiewicz implication is shown in Expression 4. This formula leverages several fuzzy logic operations to evaluate the likelihood of a transaction being fraudulent. The operations involve taking the minimum and maximum of various terms, which correspond to different transaction features

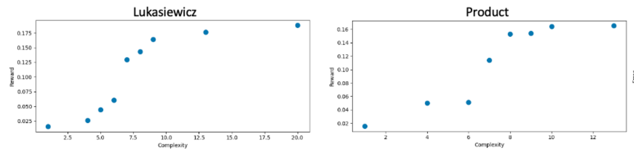

Figure 2: Pareto front showcasing the complexity versus reward for the Łukasiewicz implication versus the Product implication.

such as *newBalanceDest*, *oldbalanceDest*, and *maxDest7*, combined with transaction type *type_CASH_OUT*. The formula's structure has a series of logical conditions and implications that aim to capture the complexities of fraud detection. The complexity of this formula makes it difficult to interpret. The nested operations and multiple levels of logical implications obscure the direct relationships between input features and the final output. Understanding how each part contributes to the overall decision requires analysing each nested operation, which is not improving explainability.

In contrast, the best-performing expression using the Product implication is simpler and more explainable. We use abbreviations for convenience; NBD is newBalanceDest, OBD is oldBalanceDest.

$$\otimes\Big(\text{NBD}, \otimes\Big(\neg\Big(\oplus\Big(\text{OBD}, \text{OBD}\Big)\Big), \oplus\Big(\text{OBD}, \text{OBD}\Big)\Big)\Big) \tag{7}$$

The inclusion of the new balance of the destination (NBD) directly in the conjunction signifies its importance in determining the likelihood of fraud. A significant change in the new balance of the destination account could indicate unusual activity, especially if it does not align with typical transaction patterns.

The old balance (OBD) appears in both the negation and the disjunction operations, suggesting that the old balance of the account is crucial for understanding its normal behavior.

The simpler structure of the Product formula means fewer nested operations, making it more transparent how each feature affects the outcome. The use of basic fuzzy operations without excessive nesting helps in understanding the logical flow from input features to the fraud likelihood score. This can also be see in Figure 2, where the complexity of formulas is generally lower for the Product implication compared to the Łukasiewicz implication.

Balancing the complexity and explainability of fuzzy logic expressions is crucial. While more complex formulas like the one using the Łukasiewicz implication can capture detailed patterns in data, they can become challenging to interpret. Simpler formulas, such as those using the Product implication, offer better transparency but might miss some intricate relationships. The Pareto front approach assists in selecting the optimal balance between these trade-offs, ensuring that the model remains both effective and explainable.

6 Conclusion

This paper explored the integration of fuzzy logic into DSR to enhance the explainability and performance of fraud detection models. In particular, using the obtained framework, we compared different fuzzy logic implications. Furthermore, we investigated the alternatives of training DSR on different reward functions to tailor to specific situations, particularly regarding highly imbalanced datasets. Moreover, we analysed the performance of unconstrained and constrained implementations of fuzzy logic to gather insights into manipulating fuzzy implications to alter performance and explainability.

Comparative Analysis of Fuzzy Logic Implications. It is noteworthy to mention that the Łukasiewicz implication consistently outperforms the Product, Gödel and combined implications in terms of raw performance, answering Question 1. This can be concluded from both the single best expression results in Table 3 and average expression results in Table 4. However, when evaluating the implications and their complexity, we draw a different conclusion. The Łukasiewicz implications are generally harder to interpret compared to other implications. Although the performance of the Product implication is lower, both in the best expression and the average expression, the complexity associated with this implication is much lower. The average complexity for Łukasiewicz is 15.9, whereas the Product implication averages 13.9. Furthermore, the best performing expression of the Product implication has a complexity of 10, compared to a complexity of 20 for the Łukasiewicz implication. Therefore, we deem the Product implication the most effective in enhancing the model's ability to handle and explain the inherent complexity and uncertainty in fraud detection datasets while also improving explainability, answering Question 2.

Enhancing Model Explainability through Fuzzy Logic Integrating fuzzy logic into DSR provides compact formulas, however the obtained formulas truly require deeper look; and they are in general not very easy to interpret themselves. However, the potential seems to exist for getting clearer insights into the model's decision-making process, thereby improving the explainability of the fraud detection model. While the expressions generated by this research are not inherently easy to interpret, they could offer possibly valuable insights when analyzed by experts.

Trade-off between Complexity and Performance Our findings reveal a trade-off between the complexity of fuzzy logic formulas and their performance. Unconstrained implementations tend to produce more complex formulas with better performance, while constrained implementations simplify the formulas at the cost of reduced performance. Utilizing Pareto front analysis and expert interpretation helps in balancing predictive accuracy with formula simplicity, ensuring that the models remain both effective and explainable, answering Question 4. Experts should evaluate the expressions and choose a fuzzy logic formula according to the performance, explainability and suitability to different scenarios.

Limitations and their Impact on Conclusions Despite the promising results, our study has limitations that must be acknowledged. The performance of fuzzy logic implementations was lower than SOTA methods, primarily due to the information loss when transforming continuous data into fuzzy sets based on percentiles. Future work should explore advanced methods like fuzzy clustering and alternative data transformations to retain more information.

Additionally, computational constraints limited the thoroughness of hyperparameter tuning and the complexity of the generated expressions. Addressing these constraints will be essential for enhancing the efficiency and scalability of the DSR framework.

Future research agenda is to focus on several key areas to enhance the current framework: including fuzzy clustering and data transformations, and investigating integrating fuzzy logic and related formalisms into newer symbolic architectures.

References

[1] P. Alam. 2009. Application of Fuzzy Logic Fraud Detection. (2009). https://www.researchgate.net/publication/243444002

[2] F. Alarfaj, I. Malik, H. Khan, N. Almusallam, M. Ramzan, and M. Ahmed. 2022. Credit Card Fraud Detection Using State-of-the-Art Machine Learning and Deep Learning Algorithms. *IEEE Access* 10 (2022), 39700–39715. https://doi.org/10.1109/ACCESS.2022.3166891

[3] E. Alonso Lopez-Rojas, A. Elmir, and S. Axelsson. 2016. Paysim: A financial mobile money simulator for fraud detection. (2016).

[4] S. Askari and A. Hussain. 2020. IFDTC4.5: Intuitionistic fuzzy logic based decision tree for E-transactional fraud detection. *Journal of Information Security and Applications* 52 (2020). https://doi.org/10.1016/j.jisa.2020.102469

[5] J. Awoyemi, A. Adetunmbi, and S. Oluwadare. 2017. Credit card fraud detection using machine learning techniques: A comparative analysis. In *2017 International Conference on Computing Networking and Informatics (ICCNI)*. 1–9. https://doi.org/10.1109/ICCNI.2017.8123782

[6] A. Cherif, A. Badhib, H. Ammar, S. Alshehri, M. Kalkatawi, and A. Imine. 2023. Credit card fraud detection in the era of disruptive technologies: A systematic review. *Journal of King Saud University - Computer and Information Sciences* 35 (2023), 145–174. https://doi.org/10.1016/J.JKSUCI.2022.11.008

[7] C. Dutilh Novaes. 2017. *The syllogism as defined by Aristotle, Ockham, and Buridan*. Springer. 217–231 pages. https://doi.org/10.1007/978-3-319-66634-1_14

[8] Europol. 2021. Serious and Organised Crime Threat Assessment. (2021).

[9] B. Goodman and S. Flaxman. 2016. European Union regulations on algorithmic decision-making and a "right to explanation". (2016). https://doi.org/10.1609/aimag.v38i3.2741

[10] P. Hajek, M. Z. Abedin, and U. Sivarajah. 2023. Fraud Detection in Mobile Payment Systems using an XGBoost-based Framework. *Information Systems Frontiers* 25, 5 (2023), 1985–2003. https://doi.org/10.1007/s10796-022-10346-6

[11] P. Hájek. 2013. *Metamathematics of fuzzy logic*. Vol. 4. Springer Science & Business Media.

[12] L. A. Jeni, J. F. Cohn, and F. De La Torre. 2013. Facing imbalanced data - Recommendations for the use of performance metrics. In *Proceedings - 2013 Humaine Association Conference on Affective Computing and Intelligent Interaction, ACII 2013*. 245–251. https://doi.org/10.1109/ACII.2013.47

[13] Y. Ji, T. Helldin, and H. J. Steinhauer. 2021. Explainable AI methods for credit card fraud detection : Evaluation of LIME and SHAP through a User Study. (2021), 49.

[14] P. Kulicki. 2020. Aristotle's Syllogistic as a Deductive System. 9 (05 2020). https://doi.org/10.3390/axioms9020056

[15] W. B. Langdon. 1998. *Genetic programming and data structures : genetic programming + data structures = automatic programming!* Kluwer Academic Publishers.

[16] E. Mill, W. Garn, N. Ryman-Tubb, and C. Turner. 2023. Opportunities in Real Time Fraud Detection: An Explainable Artificial Intelligence (XAI) Research Agenda. 14 (2023). www.ijacsa.thesai.org

[17] T. N. Mundhenk, M. Landajuela, R. Glatt, C. P. Santiago, D. M. Faissol, and B. K. Petersen. 2021. Symbolic Regression via Neural-Guided Genetic Programming Population Seeding. (2021). http://arxiv.org/abs/2111.00053

[18] Nilson, S. 2019. The Nilson Report: Card Fraud Losses Reach $27.85 Billion. (2019).

[19] European Parliament. 2021. Artificial intelligence act. (2021).

[20] B. K. Petersen, M. Landajuele Larma, T. N. Mundhenk, C. Prata Santiago, S. Kyung Kim, and J. Taery Kim. 2021. Deep symbolic regression: Recovering mathematical expressions from data via risk-seeking policy gradients. In *International Conference on Learning Representations*. https://openreview.net/forum?id=m5Qsh0kBQG

[21] T. Razooqi, K. Raahemifar, P. Khurana, and A. Abhari. 2016. Credit Card Fraud Detection Using Fuzzy Logic and Neural Network. (2016).

[22] B. Russell. 1919. *Introduction to Mathematical Philosophy*. Dover Publications.

[23] G. F. Smits and M. Kotanchek. 2005. *Pareto-Front Exploitation in Symbolic Regression*. Springer US, Boston, MA. 283–299 pages. https://doi.org/10.1007/0-387-23254-0_17

[24] E. van Krieken, E. Acar, and F. van Harmelen. 2020. Analyzing Differentiable Fuzzy Logic Operators. (2020). https://doi.org/10.1016/j.artint.2021.103602

[25] S. Visbeek, E. Acar, and F. d. Hengst. 2023. Explainable Fraud Detection with Deep Symbolic Classification. (2023). http://arxiv.org/abs/2312.00586

[26] D. Vishnu Kute, B. Pradhan, N. Shukla, and A. Alamri. 2021. Deep Learning and Explainable Artificial Intelligence Techniques Applied for Detecting Money Laundering-A Critical Review. (2021). https://doi.org/10.1109/ACCESS.2021.3086230

[27] L. A. Zadeh. 1965. Fuzzy sets. *Information and Control* 8, 3 (1965), 338–353. https://doi.org/10.1016/S0019-9958(65)90241-X

Received 29 June 2024; revised 02 October 2024