Computer Systems Science and Engineering

Tech Science Press

DOI: 10.32604/csse.2024.056659

ARTICLE

Machine Learning-Driven Classification for Enhanced Rule Proposal Framework

B. Gomathi[1,*], R. Manimegalai[1], Srivatsan Santhanam[2] and Atreya Biswas[3]

[1]Department of Computer Science and Engineering, PSG Institute of Technology and Applied Research, Coimbatore, 641062, India

[2]SAP Labs India Private Limited, Bengaluru, 560066, India

[3]SAP (UK) Ltd., Middlesex, TW148HD, UK

*Corresponding Author: B. Gomathi. Email: gomathi.babu@gmail.com

Received: 27 July 2024 Accepted: 18 October 2024 Published: 22 November 2024

ABSTRACT

In enterprise operations, maintaining manual rules for enterprise processes can be expensive, time-consuming, and dependent on specialized domain knowledge in that enterprise domain. Recently, rule-generation has been automated in enterprises, particularly through Machine Learning, to streamline routine tasks. Typically, these machine models are black boxes where the reasons for the decisions are not always transparent, and the end users need to verify the model proposals as a part of the user acceptance testing to trust it. In such scenarios, rules excel over Machine Learning models as the end-users can verify the rules and have more trust. In many scenarios, the truth label changes frequently thus, it becomes difficult for the Machine Learning model to learn till a considerable amount of data has been accumulated, but with rules, the truth can be adapted. This paper presents a novel framework for generating human-understandable rules using the Classification and Regression Tree (CART) decision tree method, which ensures both optimization and user trust in automated decision-making processes. The framework generates comprehensible rules in the form of *if condition* and then predicts class even in domains where noise is present. The proposed system transforms enterprise operations by automating the production of human-readable rules from structured data, resulting in increased efficiency and transparency. Removing the need for human rule construction saves time and money while guaranteeing that users can readily check and trust the automatic judgments of the system. The remarkable performance metrics of the framework, which achieve 99.85% accuracy and 96.30% precision, further support its efficiency in translating complex data into comprehensible rules, eventually empowering users and enhancing organizational decision-making processes.

KEYWORDS

Classification and regression tree; process automation rules engine; model interpretability; explainability; model trust

1 Introduction

A rule-based system uses predefined rules to organize, classify, and manage enterprise activities. Manual rule generation for enterprise operations involves creating and maintaining rules governing

various aspects of business processes within an organization. These rules are typically developed based on the specialized knowledge and expertise of individuals familiar with the specific domain or industry. This manual process can be resource-intensive, requiring significant time and effort to identify, define, and implement rules that govern workflow management, decision-making, and compliance with regulations or standards. Manual rule generation may also introduce challenges of consistency, scalability, and adaptability as business requirements evolve. In contrast to traditional rule-based systems, which depend on predefined rules, intelligent automation systems can assess data, learn from it, and make decisions independently. Rule-based automation has been around for quite a long time to optimize many enterprise processes. These rules are maintained by business process experts who have in-depth knowledge of that enterprise domain. Rule generation automation for enterprise operations entails leveraging technology, such as machine learning algorithms and data analysis techniques, to automatically create and manage rules governing various business processes within an organization. This automated approach aims to streamline the rule generation process, reduce manual effort, and improve efficiency and accuracy. This allows for more adaptive and flexible automation solutions that handle complex scenarios and dynamic environments. Automated systems can generate rules that govern workflow management, decision-making, and compliance by analyzing large volumes of data and identifying patterns. These rules are derived from data-driven insights rather than relying solely on human intuition or expertise.

This work focuses on using Machine Learning (ML) to automate the production of rules for enterprise operations that include structured tabular data. The proposed strategy improves rule-based automation by incorporating human experience into a computational framework, solving typical shortcomings in existing systems such as manual rule generation, lack of interpretability, and data imbalances. The approach enhances categorization and lowers human error by leveraging the Classification and Regression Trees (CART) algorithm for enhanced interpretability and allowing for dynamic rule modifications. The framework extracts rules from noisy structured data and identifies patterns in massive datasets using data analysis, transforming correlations into actionable rules. This technique is beneficial for managing unstructured or noisy data, as it overcomes the constraints of standard rule-based approaches.

The proposed system uses the CART decision tree method to derive rules from structured tabular input. It builds decision trees using feature vectors generated by binary encoders that interpret historical data and convert them into actionable rule sets. While one-hot encoding is widely employed, it might increase dimensionality and sparsity, potentially leading to overfitting in the CART algorithm. In contrast, label encoding may introduce deceptive ordinal relationships, lowering the quality of decision tree splits. Binary encoding is especially beneficial for high-cardinality variables since it reduces dimensionality while increasing algorithm performance. The system additionally uses a count vectorizer to determine the top k tokens in text variables. The system improves CART performance by combining binary encoding and meticulously analyzing historical data through encoders to create feature vectors, allowing the development of decision trees that specify rules for categorizing electronic texts.

The remainder of the paper is organized as follows. Section 2 discusses related work on automated rule-generation systems using machine learning models. Section 3 explains a general rule-based framework. Section 4 describes an automated rule proposal architecture based on the CART algorithm. Section 5 shows the experimental setup, results, and performance evaluation of the proposed framework. Finally, Section 6 presents conclusions by giving directions for future work.

2 Literature Survey

Manual rule generation for enterprise operations is time-consuming and frequently involves specialized knowledge to ensure rules stay current and compliant. Inconsistent rules can cause confusion and inefficiencies inside businesses. To address these issues, academics are concentrating on automated rule-generation systems based on machine learning. For example, Bharatiya [1] proposed a method based on RULES-6 that uses transfer learning to utilize knowledge from other domains, improving search efficiency and accuracy in rule discovery. This novel approach, known as the Rule Extraction System with Transductive Learning (RULES-TL), enhances specialization and lowers error rates for whole and incomplete datasets. Another work by Adilova et al. [2] described the reasoning behind big interpretable models using a collection of logical rules. Still, it needs to address the development of human-understandable rules from structured tabular data using machine learning.

Litao et al. [3] have proposed a novel method for learning interpretable decision rule sets by training a neural network in a two-layer architecture in which each neuron in the first layer maps directly to an interpretable if-then statement. The output neuron represents a disjunction of the first-layer rules. It also provides a sparsity-based regularization strategy to balance classification accuracy with the simplicity of the resulting regulations while using state-of-the-art neural net training approaches for accurate classification models. It tests the proposed method using publicly accessible binary classification datasets, comparing it to other rule-learning algorithms and uninterruptable black-box machine-learning systems with comparable prediction accuracy. The Stable and Interpretable Rule Set (SIRUS) [4] addresses the essential need for interpretable machine learning models in high-stakes decision-making applications by ensuring transparency and accessibility. Based on random forests, SIRUS provides a symbolic representation of tree pathways for rule extraction. SIRUS provides a simple and robust rule-learning technique, making it a realistic option for producing concise and interpretable rule sets from random forests, improving the reliability and usefulness of machine-learning models in numerous real-world applications. The stability metric is empirically derived based on the average fraction of standard rules between models of different folds in cross-validation, and it may not fully describe the stability of the algorithm for all cases.

While Decision Trees (DTs) are widely regarded as interpretable models, new research has challenged this assumption by demonstrating that DT paths can contain insignificant literals, which can obscure their clarity. A linear time test [5] developed in this paper identifies unnecessary literals, allowing polynomial-time computation of Probabilistic Independence (PI) explanations to improve DT interpretability. This study reveals that conventional DT algorithms can generate explanations that lack coherence due to path redundancy, calling into doubt the intrinsic interpretability of DT classifiers. Further investigation in [6] compares standard hierarchical agglomerative clustering to semi-supervised algorithms, indicating that the latter can create higher-quality clusters, especially when prior information is used. Bogdanova et al. [7] presented an Explainable Data Collaboration Framework for remote machine learning that uses KernelSHAP to reduce user feature attribution discrepancies. Despite developments, the framework needs to prioritize creating human-readable rules for structured tabular data. As businesses move toward intelligent organizations, the change to automation through machine learning and rule-based systems stresses the importance of clarity and verifiability in decision-making processes, with rule-based systems frequently perceived as more trustworthy than black-box ML models. Table 1 presents a comparative analysis of related works in the Rule Generation Framework.

Table 1: Comparative analysis of related works in rule generation framework

Aspect	Traditional manual rule generation	Automated rule generation	Transfer learning approaches	Interpretable models	Neural network innovations	Explainable data collaboration
Efficiency	Labor-intensive and time-consuming	Streamlined process	Reduces search time significantly	Enhances transparency	Balances accuracy with simplicity	Addresses feature attribution discrepancies
Knowledge Dependency	Requires specialized domain knowledge	Less dependency on human input	Leverages knowledge from other domains	Focuses on user trust	Maps neurons to interpretable rules	Does not generate rules, focuses on explanations
Interpretability	Often lacks clarity	Varies by the model used	Improves rule discovery	High interpretability	Provides clear explanations	Model-agnostic, but not rule-based
Application Context	Limited to specific business processes	Broad applicability	Enhances accuracy in diverse domains	High-stakes applications	Directly applicable to rule generation	Focuses on collaborative data use
Advancements	Inefficiencies and inconsistencies	Reduces manual errors	Significant advancements in rule accuracy	Emphasizes transparency	Innovative architecture for rules	Addresses gaps in feature attribution

The existing literature highlights the need for enhanced interpretability and efficiency in automated systems, which the proposed framework aims to address.

Hence, this research considerably improves automated rule generation by automating the process, decreasing the need for manual intervention and specialized knowledge, as demonstrated by developing systems such as RULES-6 and RULES-TL. It highlights interpretability using frameworks such as SIRUS, which converts neural networks into understandable if-then rules, ensuring transparency in high-stakes applications. Furthermore, the proposed architecture effectively solves the issues of noisy data while remaining understandable and adaptable to regulations. The uniqueness resides in implementing the CART algorithm for rule creation, which optimizes the process while increasing user trust by giving explicit, verifiable rules. This is a significant improvement in making machine learning more accessible and dependable for enterprise operations.

3 Rule-Based System

Automated rule generation operates on rules with trigger points and corresponding actions, following a basic *IF...THEN...ELSEIF* format reminiscent of programming logic. If there are numerous triggers, it results in a corresponding number of conditional statements. Managing changes or adaptations to these rules can be challenging but feasible. However, rule-based systems have inherent limitations. They are static and lack intelligence. They operate strictly according to their predefined instructions and design. Sequential Covering methods [8] that stand out among the original algorithms include RIPPER (Repeated Incremental Pruning to Produce Error Reduction), CN2 (Clark and Niblett's 2 Algorithm), and Algorithm quasi-optimal (AQ) Learning [9]. Decision Trees are well-known in Machine Learning Algorithms for their high accuracy and interpretability. A decision tree model evaluates the significance of each variable by measuring how well it divides the dataset into various classes. In the Play Tennis dataset [10], features such as "Outlook" or "Humidity" may be more relevant depending on how well they predict "Play Tennis," as shown in Fig. 1. Using the categories presented in Table 2, a decision tree would automatically determine which variables are most successful in classifying whether tennis may be played.

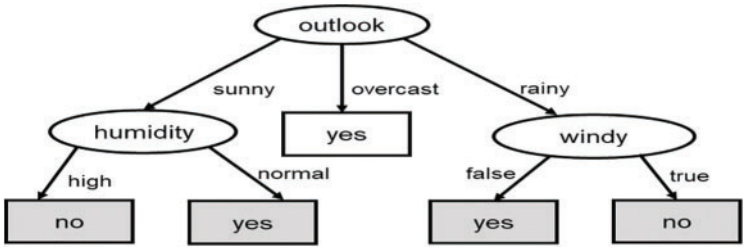

Figure 1: Decision tree learnt on the tennis dataset

Table 2: Play tennis dataset

Outlook	Temperature	Humidity	Wind	Play tennis
Sunny	Hot	High	Weak	No
Sunny	Hot	High	Strong	No
Overcast	Hot	High	Weak	Yes
Rain	Mild	Normal	Weak	Yes
Rain	Cool	Normal	Weak	Yes
Rain	Cool	Normal	Strong	No
Overcast	Cool	High	Strong	Yes
Sunny	Mild	Normal	Weak	No
Sunny	Hot	Normal	Weak	Yes
Rain	Mild	Normal	Strong	Yes
Sunny	Cool	Normal	Strong	Yes
Overcast	Mild	High	Strong	Yes
Overcast	Hot	Normal	Weak	Yes
Rain	Mild	High	Strong	No

The hierarchical structure of the decision trees can be easily converted into rules, as shown in Fig. 2.

R1: IF "outlook" = "overcast" THEN "play tennis" = "yes"
R2: IF "outlook" = "sunny" AND "humidity" = "normal" THEN "playtennis" = "yes"
R3: IF "outlook" = "rainy" AND "windy" = "false" THEN "playtennis" = "yes"

Figure 2: Rules derived from the tennis decision tree

Decision trees are used because of their interoperable natures in the framework to develop rules. Trees are used because of their interoperable natures in the framework to create rules. Automated rule generation reduces complex data into understandable rules, providing benefits similar to human decision-making. When data quantities increase, it can quickly scale to meet demand and perform well in settings without pre-processing. It also performs exceptionally well in complicated situations, using sophisticated algorithms to traverse complicated decision logic and a variety of datasets to traverse complicated decision logic and a variety of datasets efficiently. In dynamic business situations, these attributes facilitate decision-making, build confidence, and streamline operations. Additionally, the proposed system manages redundant rules by recognizing and integrating similar patterns that may appear across multiple subsets of data throughout the rule-creation process. This entails examining the created rules to identify overlaps or duplications and then combining these rules to minimize redundancies. This approach assures that the final rule set is compact and efficient, preserving simplicity and efficiency.

4 Automated Rule Proposal Framework

This work proposes a framework for automated rule generation using machine learning for classification problems [11,12], as shown in Fig. 3. Decision trees are created using feature vectors produced by processing historical data using binary encoders. The decision trees are then converted into sets of rules. These rules help classify electronic documents in an enterprise system by defining the criteria for assigning them to different classes [13].

4.1 Binary Encoder

The historical data [14,15] includes a set of variable vectors V_i such as $V_1,...,V_n$, and each vector V_i includes variables $v_1,...,v_m$. Further, each variable vector V_i is associated with a target variable y that indicates a respective class C in a set of classes $C_1,...,C_n$ that is assigned to the variable vector V_i. Here, the historical data is processed by the categorical binary encoders to provide feature data, as shown in Fig. 4, which is also provided as tabular data.

The feature data includes a set of feature vectors F, such as $F_1, ...,F_n$. Each feature vector F_i consists of values $f_1, ...,f_k$. Further, each feature vector F is associated with a target variable y that indicates a class C in classes $C_1, ...,C_k$ assigned to the value vector V_i corresponding to the respective feature vector F_i. Binary encoding for categorical variables is similar to one-hot encoding. This encoder encodes the data in fewer dimensions, which reduces data sparsity. Additionally, text variables are converted into variables of top k tokens, where k is used as a hyper-parameter by the framework to improve accuracy on the hold-out dataset [16]. More particularly, the encoders include a count vectorizer, which is used to determine the top k tokens.

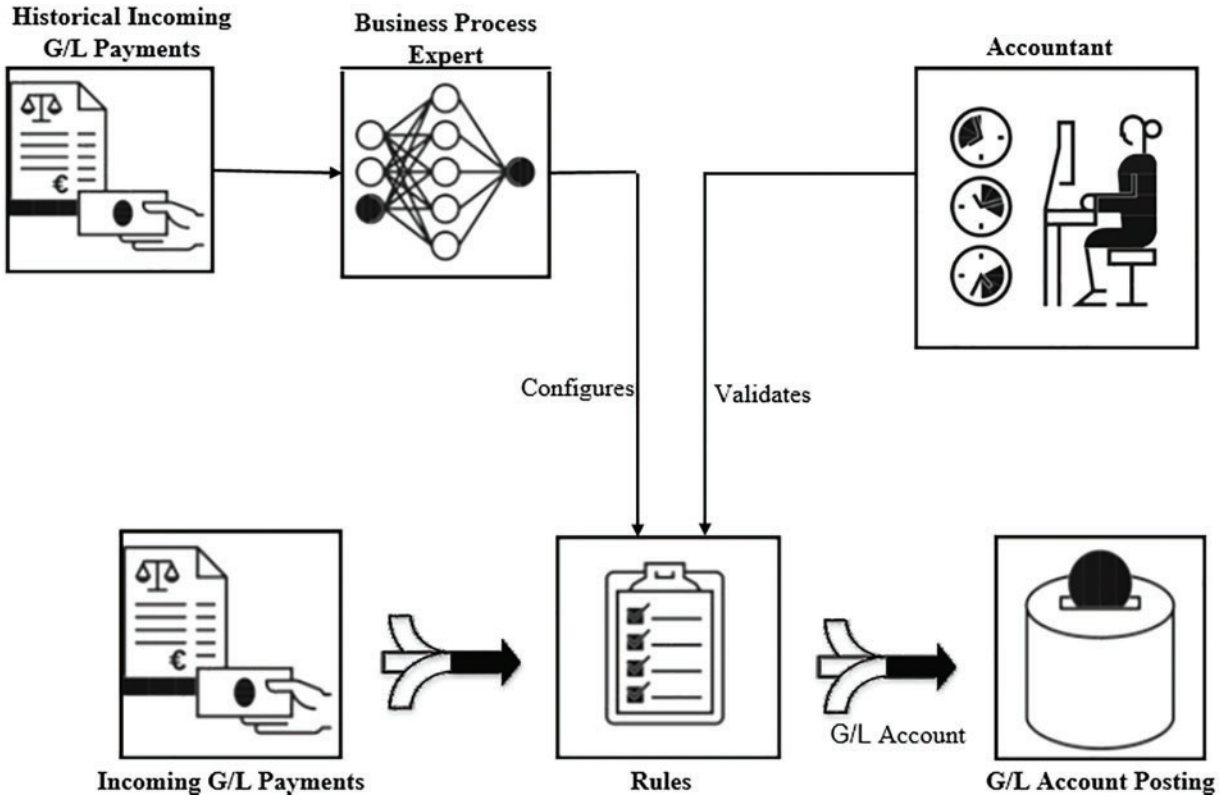

Figure 3: The proposed framework for automated rule generation framework

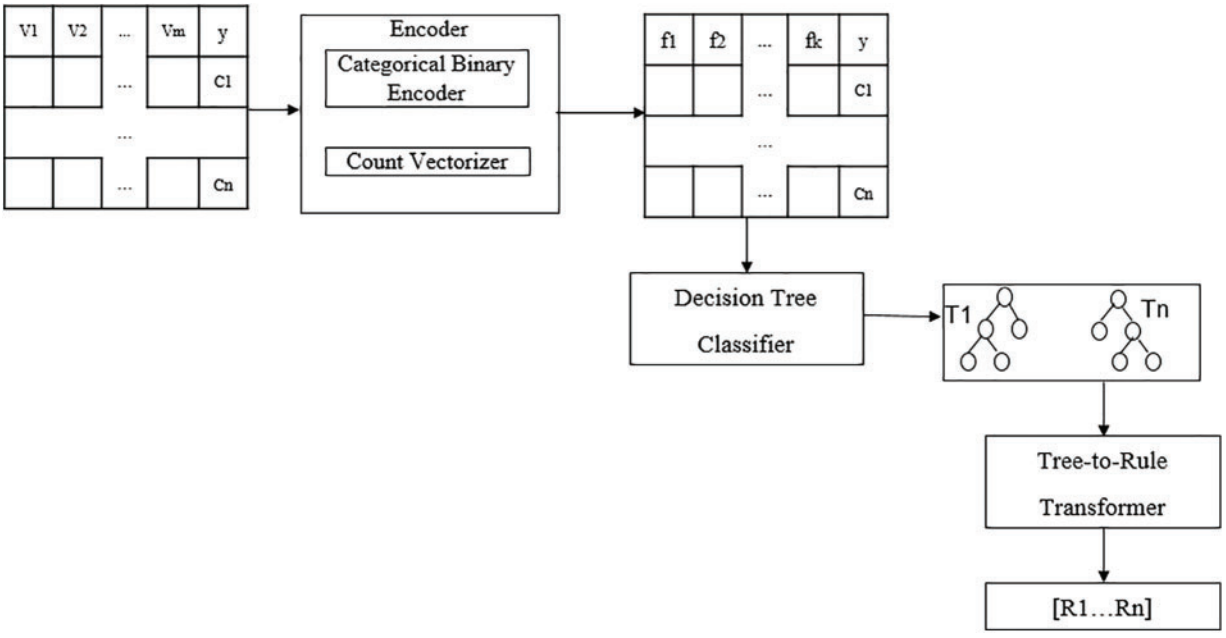

Figure 4: Tabular data to rule generation flow

4.2 CART Algorithm for Rule Proposal Framework

Regression and Classification Decision Trees can be constructed with CART [17]. The model is a classification tree when a decision tree divides a dataset into multiple classes. The goal of utilizing a classification tree is to divide the dataset into two halves based on the homogeneity of the data. CART is based on impurity measures such as the Gini index to choose the attributes for splitting. Every feature in the dataset is divided into two halves; the difference between the actual and predicted values is computed, and the split point with the lowest sum of squared errors is selected as the root node. From there, the process repeats itself, splitting each feature in half. The rules determining when tree growth ends are known as stopping criteria. To prevent a node from being split, it must consist of records with the same value for the target variable, i.e., a pure node's size must not surpass a user-defined threshold; the tree's depth must not exceed a predefined maximum value; the node must contain fewer cases than a predefined minimum number; and any split must not increase purity beyond a certain threshold. With out-of-depth analysis and comprehension of the data, it is hard to forecast the appropriate threshold values. Hence, CART employs a pruning method to get to the ideal tree. The pruning algorithm initially lets the tree grow to its full size before trimming it back. Cutting down a tree too soon may result in losing significant, highly informative sub-trees. When a tree grows unchecked and unpruned, the predictive model that overfits the data supplied cannot be replicated as a good fit when applied to further samples [18].

In the proposed framework, variable types of structured tabular data can be summarized into the following types: Numerical, Categorical, and Text. The Rules Proposal framework uses the CART algorithm, which cannot handle categorical variables. Hence, Binary Encoding is used to hold the categorical variables. The decision tree receives the numerical values from the framework in their original form without any extra handling. Since decision trees are invariant to monotonic transformations of the variables, there is no need to normalize the numeric input data. The Rules Proposal Framework uses the CART algorithm to generate decision trees from feature vectors. Binary encoding for categorical variables is similar to one-hot encoding but stored as binary bit strings. This encodes the data in fewer dimensions than one-hot encoding and thereby, helps to reduce the sparsity. Text variables are converted into variables of top k tokens where, k is used as a hyperparameter by the framework to improve accuracy on the hold-out dataset. Here, the Count vectorizer is used to determine the top k tokens. After the transformations, the variables are fed into the Decision Tree Classifier in a One *vs*. Rest Fashion per class. Hence, we create a decision tree for every class and convert it into a rule set.

The decision tree branches with numeric variables are transformed into the rule condition with the comparison operators such as–Less than($<$), Greater than($>$), Less than Equal to($<=$), Greater than Equal to ($>=$), Equal to($==$) and Not Equal to ($!=$). Similarly, the decision tree branches with the categorical variables are transformed into the rule condition with the Equality Operator ($==$). In contrast, it is converted into a rule condition for the decision tree branches containing the Text Variable using the Contains Operator ($\subset$). For text processing, the proposed automation-based framework can process tokens faster in parallel, with an extensible framework where additional skills can be added to usage, a low-power processing model. Existing NLP frameworks work to decipher embodied learning, synthesize statements, decipher innate biases, and work on natural language understanding. In contrast, the proposed framework emphasizes key tokens, their occurrences normalized over a data set, and metric implication with other feature dimensions, making it more relevant to the table structure being processed.

In Table 3, the tokens "Keev", "outstanding", and "payment" have a high implication score when compared to "done" in the free text Comments. Among the three tokens, "Keev" would have a higher

score, followed by "outstanding" and "payment" at the same level. This implication metric is calculated after identifying top K tokens and normalizing to ensure speed. A skill that can be introduced to meet user requirements is enhancing content with synonyms. Assume that for the same table, there is a need to check for synonyms for "payment". Then near related keywords would be "cash", "premium", "fee", "amount,", etc. This would ensure that the metric "implication" gives the same score if an alternate word is used. Another example of a skill is checking for a business partner's name with standard data services such as Dun & Bradstreet D-U-N-S Number, which can help improve the *implication* score. Tabular data can be represented as $T_{i(j-1),}$ where i is the number of rows, $(j-1)$ is the number of input variables, and y is the target variable.

Table 3: Metric implication

Vendor ID	Vendor name	Address	Fee posted	Payment time	Outstanding	Comments
10000098765	Keev enterprises	1,Woodway, Austin	$95,000	90	$200,765	Keev outstanding payment done

Let v represent the input variables of size $(j-1)$, where type $V_j \in$ {categorical, numeric, text, target} and the target variable be represented by y where n is the number of classes and described as C.

The steps involved in the proposed framework to generate a Rule Set are shown in Fig. 5. As described above, the tabular data is transformed into the feature vector Fik by the encoder's component. The feature vector is then fed into the One *vs*. The rest of the decision tree classifier results in n trees (the same as the number of classes). Each decision tree is associated with classification metrics like accuracy, RoC, Area under the curve, etc. In the last step, the decision tree is transformed into a Rule Set, and the Rule Set is also associated with the classification metrics as well as derived from the Decision Tree.

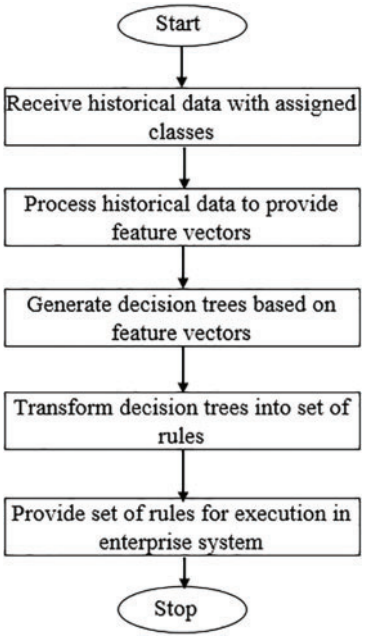

Figure 5: Flow diagram for rule generation framework

Furthermore, the proposed automated rule generation framework uses the CART algorithm to adapt to new categories efficiently. The CART algorithm adjusts to previously unknown categories by updating feature vectors and improving adaptability. Furthermore, the flexibility of the CART algorithm allows decision trees to evolve with new data without requiring full retraining, and the framework's robustness allows it to manage noisy data, enabling reliable classification in quickly changing environments. As a result, the suggested methodology promotes continuous learning, allowing it to incorporate new categories from incoming data while successfully evolving situations such as cyber security or market trends [19,20].

4.3 ERP Case Study for Account Determination

Account Determination in Enterprise Resource Planning (ERP) solutions is one of the easiest and essential applicability of this rule framework, which has already been discussed. Account Determination is the key to linking Asset classes to G/L (General Ledger) accounts based on transaction type. The MM-FI (Materials Management and Financial Accounting) link is crucial in ensuring that FI postings are done to the correct G/L account based on movement types. These transaction keys are used to determine general ledger accounts used by the system.

This is done when a new ERP system is set up as fresh or configured for use and later adjusted based on usage. This process is tedious and requires functional expertise to maintain the configurations correctly. It would lead to incorrect financial postings, lengthy reconciliations, and an impact on an enterprise's financial books of accounts. However, most of this setup stays the same from one enterprise to another. An intelligent rule-based framework proposed in Fig. 6 can help to evolve various possible rules and propose the correct G/L account assignment in the Account Determination config phase. A similar approach is also helpful for determining the G/L account when an incoming payment comes into the system. This eliminates the requirement for a Business Process Expert or Functional Expert to spend time generating and configuring rules for an ERP system.

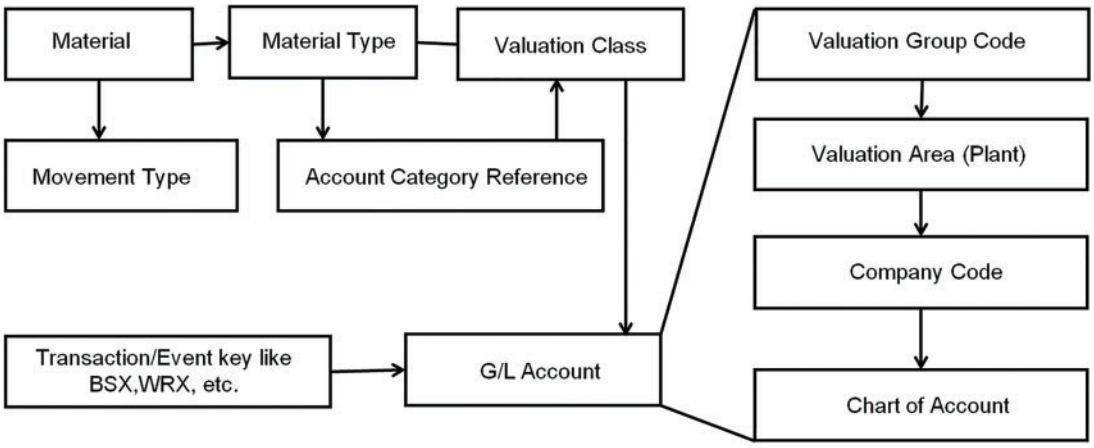

Figure 6: Steps involved in sample account determination

5 Experimental Results

Before evaluating a machine learning model, examining its performance and confirming that its predictions are consistent with data patterns is critical. This analysis focuses on two primary areas: performance metrics for model evaluation and statistical validation. The model's performance is measured using conventional metrics such as accuracy and precision, which provide information about how well the model predicts. In addition to these measures, Statistical Validation checks that the model's conclusions are statistically correct. The Chi-Square Test and ANOVA are used to examine whether there are significant relationships or differences in the data, which provide a better understanding of the model's validity. Together, these methods offer a thorough assessment of the model's performance.

5.1 Performance Metrics for Model Evaluation

The proposed rule generation framework is tested and validated with the Car Evaluation Dataset [21,22]. This dataset comprises categorical variables, making it ideal for classification tasks. Table 4 shows that each variable represents a separate part of car appraisal. These variables provide a comprehensive collection of properties necessary for determining the acceptability and desirability of various cars from a consumer standpoint [23].

Table 4: Car evaluation dataset variable

Variables	Type	Description
buying	Categorical	Buying price—vhigh, high, med, low
maint	Categorical	Maintenance price—vhigh, high, med, low
doors	Categorical	Number of doors—2, 3, 4, 5, more
persons	Categorical	Persons—2, 4, more
lug_boot	Categorical	Size of luggage boot—small, med, big
safety	Categorical	Estimated safety—low, med, high
class	Target	Car acceptability—unacc, acc, good, vgood

The categorical character of the dataset is a common issue for machine learning models, especially those developed for classification tasks. The dataset's structured style and clearly defined categories allow for a thorough examination of the proposed framework's performance in handling categorical data and conducting accurate classifications. Table 5 summarizes the distribution of the goal variable, Car Acceptability, across the Car Evaluation Dataset. This dataset consists of 1728 items divided into four degrees of car acceptability. The *unacceptable*, i.e., *unacc* class, contains the majority of the samples, with 1210 autos classified as unsatisfactory. This accounts for 70.02% of the whole dataset, showing that a substantial proportion of the vehicles tested are unsuitable. The *acceptable*, i.e., *acc* class contains 384 samples, which accounts for 22.22% of the dataset. Cars in this category fulfill the minimum standards of acceptability. The *good*, i.e., *good* class has 69 samples, or 3.99% of the total. These cars are considered to be of good quality, but not the best. The smallest class, *very good*, i.e., *vgood*, with 65 samples, accounts for 3.76% of the dataset. Cars in this category are thought to be in excellent condition and represent the highest quality in this collection.

Table 5: Class distribution

Class	Number of samples (N)	Percentage samples (N%)
unacc	1210	70.02%
acc	384	22.22%
good	69	3.99%
vgood	65	3.76%

The major goal of this experiment is to find a rule set that accurately identifies cars in the *very good*, i.e., *vgood* state. The imbalanced nature of the dataset, with the *vgood* class being the least represented, presents a problem for the classification model. The proposed framework's efficacy will be assessed based on its ability to identify and effectively categorize these high-quality cars despite their low prevalence in the dataset. The identification of clear and specific rules for this class will illustrate the framework's robustness and practical application when dealing with skewed categorical data. Table 6 summarizes the categorization metrics [24] for the Car Evaluation Dataset. The dataset was divided into two sets: 1036 samples, i.e., 60% for training and 692 samples, i.e., 40% for testing.

Table 6: Classification metrics

Training dataset size	1036
Test dataset size	692
Accuracy on test dataset	99.85%
Precision on test dataset	96.30%

The model's performance on the test set is evaluated using two major metrics: accuracy and precision. The model's accuracy on the test dataset is 99.85%. This high accuracy shows that the model generated using the proposed framework successfully classified 99.85% of the test samples, proving its ability to forecast car acceptability categories based on the provided parameters. Precision, defined as the fraction of accurate positive forecasts among all positive predictions, is stated to be 96.3%. This metric emphasizes the model's ability to correctly recognize *very good*, i.e., *vgood* condition cars, demonstrating its consistency in producing positive classifications with few false positives. These metrics demonstrate the model's outstanding performance, especially in the setting of an imbalanced dataset. The framework's high accuracy and precision indicate that it is well-suited to accurately classifying cars into their respective acceptability categories, particularly in identifying those in *very good* condition, which was the major goal of this experiment. Table 7 shows the confusion matrix for classifying the *very good*, i.e., *vgood* class in the Car Evaluation Dataset using the model's predictions. The model properly categorized 665 of the 692 samples as *not vgood* when they were *not very good*. This reflects a high true negative rate. There was only one case where the model predicted *vgood* for a sample that was *not vgood*, resulting in a false positive. The model accurately detected all 26 *vgood* samples, with no false negatives.

This confusion matrix demonstrates the model's accuracy in detecting the *vgood* class, which has a perfect true positive rate and an almost perfect true negative rate. This confusion matrix further supports the 96.3% precision and 99.85% accuracy on the test dataset, proving the stability and robustness of the rule set created for classifying autos as *very good*, i.e., *vgood*. The model's ability

to correctly distinguish *vgood* cars from the other categories confirms its usefulness in real-world circumstances where identifying high-quality vehicles is critical.

Table 7: Rule Set generated for class = "vgood"

N = 692	Predicted: *not vgood*	Predicted: *vgood*
Actual: *not vgood*	665	1
Actual: *vgood*	0	26

5.2 Statistical Validation

As shown in Table 8, General Ledger (G/L) account data collection is crucial for assessing payment transactions in an enterprise. This data set is statistically analyzed using the Chi-Square Test and ANOVA to investigate correlations and differences among distinct G/L account types. The Chi-Square Test, in particular, demonstrates a substantial link between payment amounts and G/L account types, with a p-value of less than 0.05, indicating a noteworthy relationship.

Table 8: Sample General Ledger (G/L) payment data

Transaction ID	Date	G/L account no.	G/L account category	Account description	Debit amount
T1001	2024-09-01	5001	Liabilities	Accounts payable	0.00
T1002	2024-09-01	4005	Income	Sales revenue	2500.00
T1003	2024-09-02	6002	Expenses	Office supplies	300.00
T1004	2024-09-03	7004	Expenses	Utilities expense	0.00
T1005	2024-09-03	5003	Liabilities	Payroll expense	1200.00
T1006	2024-09-04	4002	Income	Consulting income	0.00
T1007	2024-09-04	6005	Expenses	Advertising expense	500.00
T1008	2024-09-05	5002	Liabilities	Rent expense	0.00
T1009	2024-09-06	4006	Income	Interest income	100.00
T1010	2024-09-06	7001	Expenses	Travel expense	800.00
T1011	2024-09-07	5004	Liabilities	Insurance expense	0.00
T1012	2024-09-08	4003	Income	Service fees income	0.00
T1013	2024-09-08	6001	Expenses	Legal fees	350.00
T1014	2024-09-09	7003	Expenses	Equipment expense	0.00
T1015	2024-09-09	4004	Income	Product sales	2700.00

A Chi-Square Test of Independence is performed to cross-validate the model results by comparing incoming payment kinds to G/L account categories from the dataset. The framework developed by the model had an accuracy of 96.2%. This test is used to determine whether there is a significant relationship between two categorical variables, in this case, "Payment Status" (containing categories such as "Paid", "Received", and "Due") and "G/L Account Category" (such as "Expenses", "Income", and "Liabilities"). The steps for conducting a Chi-Square Test are outlined below:

1. The following Contingency Table 9 is constructed based on the provided data set.
2. The expected frequency in Table 10 is calculated for each G/L Account Category.
3. Total Chi-Square Value (χ^2) = 2.0 + 6.65 + 1.33 + 1.76 + 2.33 + 0.01 + 0.03 + 1.0 + 1.8 ≈ 16.91.
4. Degrees of Freedom (df) = (Rows–1) ∗ (Columns–1) = (3–1) ∗ (3–1) = 4.
5. Using a Chi-Square distribution Table, for df = 4, a χ^2 value of 16.91 gives a *p*-value well below 0.05.
6. Since the *p*-value is less than 0.05, the null hypothesis is rejected and concludes that there is a significant association between "Payment Status" and "G/L Account Category". This suggests that certain types of payments are more likely to be associated with specific G/L account categories, validating the classification of payments to G/L accounts.

Table 9: Contingency table for the given data set

G/L account category	Paid	Received	Due
Income	0	5	0
Expenses	5	0	2
Liabilities	1	0	2
Total	6	5	4

Table 10: Expected frequency for each G/L account category

G/L account category	Paid	Received	Due
Income	0	5	0
Expenses	5	0	2
Liabilities	1	0	2

Another validation method used was ANOVA (Analysis of Variance). ANOVA compares the variation within groups to that between groupings. If the variance between groups is much more significant than the variance between groups, it means that at least one group's mean differs from the others. One use of ANOVA was to determine whether there is a statistically significant difference in mean payment amounts (debit and credit amounts) across different G/L account types.

1. Define the Hypotheses
   Null Hypothesis (H_0):
   There is no significant difference in the mean payment amounts among the different G/L account categories.
   Alternative Hypothesis (H_1):
   There is a significant difference in the mean payment amounts among the different G/L account categories.
2. Calculate the Group Means and Overall Mean:
   Mean for Income = (2500 + 3200 + 1500 + 100 + 2700)/5 = 2000.00
   Mean for Expenses = (300 + 500 + 800 + 350 + 1200 + 200 + 2500)/7 ≈ 835.71
   Mean for Liabilities = (0 + 400 + 1800)/3 ≈ 733.33
   Overall Mean = 1206.67

3. Calculate the Sum of Squares Between Groups (SSB)
   For Income = 3,146,835.56
   For Expense = 963,290.02
   For Liabilities = 669,159.94
   Total Sum of Squares (SST): 16,852,107.65
4. Calculate the Mean Squares
   MSW = 1,006,068.51
5. Calculate the F-Statistic
   F = 2.38
6. See an F-distribution table or a calculator to determine the critical value for (2, 12) degrees of freedom. At a significance level of 0.05, the critical value is around 3.89.

The computed F-value (2.38) is smaller than the critical value (3.89); hence, we cannot reject the null hypothesis.

6 Conclusions

To solve classification problems, this work suggests an automated rule generation framework that uses machine learning and involves processing historical data through encoders to produce feature vectors. These feature vectors are used to create decision trees, converted into rules specifying requirements for classifying electronic documents into particular groups. With the help of the proposed framework, electronic records may be categorized inside an enterprise system, simplifying processes and boosting productivity. The proposed framework achieves an accuracy of 99.85% and a precision of 96.30% on the test dataset, producing rules that are understandable to humans by utilizing the Classification and Regression Tree (CART) decision tree algorithm for structured tabular data. Compared to previous methods, the proposed model simplifies document categorization in corporate systems, increasing productivity while improving accuracy, transparency, and adaptability. The proposed framework efficiently categorizes electronic documents in business systems, streamlining workflow management and compliance. This improves overall organizational productivity. However, the framework's emphasis on uniformly categorized qualities may limit its applicability to a wide range of datasets and failto address the issues of integrating conflicting information from diverse sources. Future research may investigate the framework with various datasets, address issues related to data conflicts, and improve the interpretability of decision trees and rules to enhance their applicability in critical decision-making contexts.

Acknowledgement: We would like to extend our special thanks to SAP Labs, Bengaluru, India.

Funding Statement: The authors received no specific funding for this study.

Author Contributions: B. Gomathi conceived and designed the experiments and authored or reviewed drafts of the article. R. Manimegalai performed the experiments, prepared figures and tables, and approved the final draft. Srivatsan Santhanam conceived and designed the experiments performed them, analyzed the data, and approved the final draft. Atreya Biswas conceived and designed the experiments, performed the experiments, analyzed the data, prepared figures and/or tables, and approved the final draft. All authors reviewed the results and approved the final version of the manuscript.

Availability of Data and Materials: The Car Evaluation dataset is available at https://archive.ics.uci.edu/dataset/19/car, accessed on 14 August 2023.

Ethics Approval: Not applicable.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] J. P. Bharadiya, "The role of machine learning in transforming business intelligence," *Int. J. Comput. Artif. Intell.*, vol. 4, no. 1, pp. 16–24, 2023. doi: 10.33545/27076571.2023.v4.i1a.60.

[2] L. Adilova, M. Kamp, G. Andrienko, and N. Andrienko, "Re-interpreting rules interpretability," *Int. J. Data Sci. Anal.*, vol. 51, no. 5, pp. 1–21, Jul. 2023. doi: 10.1007/s41060-023-00398-5.

[3] Q. Litao, W. Wang, and B. Lin, "Learning accurate and interpretable decision rule sets from neural networks," in *Proc. AAAI Conf. Artif. Intell.*, May 2021, pp. 4303–4311.

[4] B. Clément, G. Biau, S. Da Veiga, and E. Scornet, "Interpretable random forests via rule extraction," in *Int. Conf. Artif. Intell. Stat.*, Mar. 2021, pp. 937–945.

[5] I. Yacine, A. Ignatiev, and J. Marques-Silva, "On explaining decision trees," 2021, *arXiv:2010.11034*.

[6] P. Drouin *et al.*, "Semi-supervised clustering of quaternion time series: Application to gait analysis in multiple sclerosis using motion sensor data," *Stat. Med.*, vol. 42, no. 4, pp. 433–456, 2023. doi: 10.1002/sim.9625.

[7] A. Bogdanova, A. Imakura, and T. Sakurai, "DC-SHAP method for consistent explainability in privacy-preserving distributed machine learning," *Hum.-Cent. Intell. Syst.*, vol. 3, no. 3, pp. 197–210, 2023. doi: 10.1007/s44230-023-00032-4.

[8] J. Błaszczyński, R. Słowiński, and M. Szeląg, "Sequential covering rule induction algorithm for variable consistency rough set approaches," *Inform. Sci.*, vol. 181, no. 5, pp. 987–1002, 2011. doi: 10.1016/j.ins.2010.10.030.

[9] R. S. Michalski, "On the quasi-minimal solution of the general covering problme," in *Proc. V Int. Symp. Inf. Process.*, 1969, pp. 125–128.

[10] J. G. Carbonell, R. S. Michalski, and T. M. Mitchell, "An overview of machine learning," *Mach. Learn.*, vol. 1, no. 1, pp. 3–23, 1983. doi: 10.1016/B978-0-08-051054-5.50005-4.

[11] J. R. Quinlan, "Induction of decision trees," *Mach. Learn.*, vol. 1, no. 1, pp. 81–106, 1986. doi: 10.1007/BF00116251.

[12] M. M. Ali, M. S. Qaseem, L. Rajamani, and A. Govardhan, "Extracting useful rules through improved decision tree induction using information entropy," *Int. J. Inf. Sci. Tech.*, vol. 3, no. 1, pp. 27–41, Jan. 2013.

[13] F. Stahl and M. Bramer, "P-Prism: A computationally efficient approach to scaling up classification rule induction," in *Artif. Intell. Theory Pract. II*, Boston, MA: Springer, 2008, pp. 77–86.

[14] J. Cendrowska, "PRISM: An algorithm for inducing modular rules," *Int. J. Man Mach. Stud.*, vol. 27, no. 4, pp. 349–370, 1987. doi: 10.1016/S0020-7373(87)80003-2.

[15] V. Margot, J. -P. Baudry, F. Guilloux, and O. Wintenberger, "Rule induction partitioning estimator: A new deterministic algorithm for data dependent partitioning estimate," in *Int. Conf. Mach. Learn. Data Mining Pattern Recognit.*, 2018, pp. 288–301.

[16] J. R. Quinlan, "Improved use of continuous attributes in C4.5," *J. Artif. Intell. Res.*, vol. 4, pp. 77–90, 1996. doi: 10.1613/jair.279.

[17] N. Z. Zacharias, "Classification and Regression Trees (CART) for predictive modeling in blended learning," *Int. J. Intell. Syst. Appl.*, vol. 3, no. 1, pp. 1–9, 2018. doi: 10.5815/ijisa.2018.03.01.

[18] S. Marek, Ł. Wróbel, and A. Gudyś, "GuideR: A guided separate-and-conquer rule learning in classification, regression, and survival settings," *Knowl.-Based Syst.*, vol. 173, no. 5, pp. 1–14, 2019. doi: 10.1016/j.knosys.2019.02.019.

[19] Y. Wang, Y. Li, Y. Sun, and Y. Jiang, "Identifying industrial control equipment based on rule matching and machine learning," *Comput. Model. Eng. Sci.*, vol. 137, no. 1, pp. 577–605, 2023. doi: 10.32604/cmes.2023.026791.

[20] H. Lin, J. Lin, and F. Wang, "An innovative machine learning model for supply chain management," *J. Innov. Knowl.*, vol. 7, no. 4, pp. 1–15, 2022. doi: 10.1016/j.jik.2022.100276.

[21] M. Bohanec, "Car evaluation-UCI machine learning repository," 1997. Accessed: Oct. 17, 2024. [Online]. Available: https://archive.ics.uci.edu/dataset/19/car+evaluation

[22] D. Dua and C. Graff, *UCI Machine Learning Repository*. Irvine, CA: School of Information and Computer Science, University of California, 2019.

[23] E. V. Kotelnikov, and V. R. Milov, "Comparison of rule induction, decision trees, and formal concept analysis approaches for classification," *J. Phys.: Conf. Ser.*, vol. 1015, no. 3, pp. 1–5, 2018. doi: 10.1088/1742-6596/1015/3/032068.

[24] B. Gomathi, R. Manimegalai, S. Srivatsan, and K. Ravikiran, "Streamlining software development in enterprises: The power of metrics-driven development," *Int. J. Recent Innov. Trends Comput. Commun.*, vol. 11, no. 9, pp. 1610–1617, 2023. doi: 10.17762/ijritcc.v11i9.9146.