Topic 4
An approach to interpretability in regression machine learning algorithm: a closer look at rule-based learning with Linear Programming

Team 7 (Junhee Lee - Vlad Miron - Diep Nhiep - Dawei Zhang)
Seminar in Business Analytics and Quantitative Marketing,
Erasmus School of Economics,
Erasmus University Rotterdam
April 2022

Abstract

This paper extends the classification-focused rule-based interpretable machine learning algorithms based on Linear Programming (LP) to the regression problem. More specifically, we focus on Rule Extraction (RUX) and Rule Generation (RUG) algorithms by adjusting the mathematical formulations of these classification algorithms such that they can process regression data with a continuous target variable. Additionally, using nine datasets, this study assesses the accuracy and interpretability of RUX and RUG. Also, we perform a case study to compare the performance of RUX and RUG when they are applied to the datasets with different sample sizes and number of features. Overall, RUX and RUG algorithms are applicable to regression datasets and have an advantage over conventional tree-ensemble machine learning algorithms in terms of accuracy and interpretability. Our results have confirmed that RUX and RUG output significantly fewer rules compared to conventional tree-ensemble machine learning algorithms without sacrificing in terms of accuracy. Also, we conclude that RUX and RUG predictions are more accurate under datasets with larger sample sizes and datasets with fewer features. However, there is no objectively superior algorithm when comparing RUX with RUG.

1 Introduction

For the past two decades, Machine Learning (ML) draws considerable interest from the academic and scientific community, all the way to the general public. Jordan and Mitchell (2015) state more and more applications and technology surrounding us are the product of ML. Litjens et al. (2017) show these wide applications could be seen in research fields such as Medicine (with medical image analysis) or Policy-making (with Criminology track and trace) in Brennan and Oliver (2013) or business and daily-life. It implies that many important decisions from small scale to large scale nowadays are drawn by computers and ML. With this wide range of applications and the serious consequences of each decision, Akyüz and Birbil (2021) consider that accuracy and interpretability (the ability for humans to understand the results shown by the machines) in ML are of great importance. However, there is a trade-off between the level of accuracy of the output and the ability of users to understand and interpret the output of the ML algorithms. The more advanced and accurate a method is, the less understandable it is for humans. Hence, it remains a goal for researchers around the world to help interpret and explain methods and algorithms for decision-makers and the general public.

Rule-based learning is one of the ML approaches that help the decision-making process using specific conditions (if-then statements). Similar to the rule-based method, tree-based methods perform a similar analysis. A simple example could be: "**if** level of credits > the required credit points), **then** the customer could have a home loan". These two types of ML learning algorithms are specifically useful in classifications. However, to compare, Fürnkranz (1999) draws a list of evidence for the advantages of rule-based learning over tree-based learning. The main reason behind this is that there is independence in the rule-based algorithm resulting in flexibility and interpretability, meanwhile, that is not the case in tree-based algorithms. Akyüz and Birbil (2021) argue that the dependence of leaves in trees (siblings or parents nodes) could lead to complexity and inaccuracy: the case when a left child of a node grows as a rule, and the right child grows in the opposite direction. Fürnkranz (1999) shows in his paper that even though the starting problem appears as a decision tree, we could easily turn that into a ruleset. Additionally, Bottou et al. (2018) demonstrate rule-based algorithms are confirmed to enhance the interpretability of Machine Learning problems.

One of the concerning challenges in ML is that the inability to process a large dataset. Malioutov and Varshney (2013) address this issue by relaxing the complex constraints. The other approach to increase the accuracy and objectivity of large data set from Demiriz et al. (2002) is to use Linear Programming (LP)-based machine learning techniques . Methods that are based on LP are also easier to understand and explain, as LP is one kind of relaxation from other black box paradigms, as shown by Malioutov and Varshney (2013). Mangasarian et al. (1995) state that the improvement in practical cases with the help of a LP-based learning algorithm could be seen in breast cancer diagnosis and prognosis. Demiriz et al. (2002) also demonstrate that the LP approach with boosting leads to more efficient solutions, thanks to the column generation method. Wang and Rudin (2015) propose the idea of an LP-based machine learning algorithm to build the objective function (by minimizing the classification errors in the loss function) and constraints the sum of the length of patterns. In general, with this formulation, Bottou et al. (2018) consider it is possible to apply the method on large scale ML problems with increases in accuracy, efficiency, and interpretability.

As one representative paper for the Rule-based learning Interpretable algorithm with Linear Programming, Akyüz and Birbil (2021) have taken Rule Extraction algorithm (RUX) and Rule Generation algorithm (RUG) with the help of the Random Forest (RF) model and the AdaBoost (ADA) model. There are various reasons why these two algorithms (RUX and RUG) are the most promising methods for the interpretability of ML. In the above-mentioned paper, Akyüz and Birbil (2021) have summarized that RUX and RUG are: scalable for large data sets, able to address multi-class data sets, able to return optimal weights for the rules and offer the possibility to address categorical or continuous features of data.

This paper inherits Akyüz and Birbil (2021) and extends their models such that they are applicable to the regression problems. Furthermore, we examine the performance of these rule-based learning methods on different datasets with different sample sizes and the number of features. Hence, this research aims to answer the following research question:

"How does the LP-based interpretable learning approach work for regression problems?".

To answer the central research question, we address two sub-questions. First, *Sub-Q1: Do*

RUX and RUG exhibit greater mean absolute error reduction for a dataset with a larger sample size than a dataset with a smaller sample size?. This sub-question aims to understand in which cases the LP-based interpretable rule learning approach is most applicable, this paper aims to examine whether RUX and RUG perform better when applied to datasets with larger sample sizes. Cui and Gong (2018) show that, according to the current literature, machine learning algorithms perform better for a dataset with many samples because it prevents bias and increases the generalizability of the prediction.

Our second sub-question, *Sub-Q2: Do RUX and RUG exhibit greater mean absolute error reduction for a dataset with a smaller number of features than a dataset with a larger number of features?*, explores the effect of the number of features on the performance of RUX and RUG. In machine learning, a feature is an equivalent term to an explanatory variable in Econometrics. Reif and Shafait (2014) state that the existence of redundant or irrelevant features during the training process negatively affects the performance of conventional machine learning algorithms such as Support Vector Machine. Additionally, Marsland (2011) shows that the increase in the number of features exponentially increases the required number of samples and complicates the algorithms; known as the curse of dimensionality.

Our results have shown that RUX improve the accuracy and interpretability compared to the tree ensemble methods. Also, RUG performs better than the decision tree (DT) in terms of prediction accuracy while sacrificing interpretability. This finding is consistent with the performance of RUX and RUG when they are applied to the classification problem. Among these proposed methods, we conclude RUX with AdaBoost as a base model performs best with the smallest mean absolute error (MAE) over the nine testing datasets. None of the methods outperforms the others in terms of both accuracy and interpretability, as each of them has a trade-off between accuracy and interpretability.

The roadmap for this paper is as follows. First, in the Literature Review, we discuss the existing papers and their relevance to our research. In the Data section, we introduce datasets and the cleansing process. In the Methodology section, we elaborate on the theoretical framework and mathematical formulations of the ML algorithms that this paper studies. In the Results

section, we display the output in tables while making comparisons and discussions to answer the research question. Finally, the Conclusion section provides a summary of the most important findings, introduces their theoretical and practical implications and discusses the limitations of our research and future research ideas.

2 Literature Review

Existing research investigates and develops rule-based machine learning algorithms for both classification and regression problems. For instance, Li and Liu (2014) apply two rule-based classification approaches to the data in various domains such as text categorization, security, and health care. Namely, they introduce the rule induction approach (CN2 and RIPPER algorithm) and the association rule mining approach. For the regression problem, Weiss and Indurkhya (1995) suggest a rule induction method for predicting the continuous dependent variable value. Furthermore, Weiss and Indurkhya (1995) introduce the pruning approach to the rule set in order to cut off the noisy features with low predictive power. More specifically, Weiss and Indurkhya (1995) explain that the rule pruning can be performed by removing the element in a rule set with the minimum impact on increasing the error. Ideally, discarding the irrelevant rule with mere predictive power would enhance the interpretability of the machine learning method by reducing the number of rules to consider. However, as Weiss and Indurkhya (1995) insist, the rule pruning involves a loss in prediction accuracy. Additionally, they point out that rule pruning has inherent costs for both classification and regression problems. For classification, Weiss and Indurkhya (1995) argue that the pruned set of rules may not cover the classification of the whole sample and the classification gap can occur. For regression, deleting rules changes the median of the outcome variable y in the final region, which adds computational complexity to this method.

To minimize these issues, the current literature applies LP to optimize the rule pruning problem. For a classification problem, Malioutov and Varshney (2013) propose a rule-based classification method that solves a binary program that minimizes the number of rules for a boolean compressed sensing problem. Wang and Rudin (2015) apply mixed-integer linear programming (MILP) problem to find classification rules for the binary dependent variable. Also, to avoid the drawbacks of rule mining, Dash et al. (2018) suggest an integer programming (IP) formulation for boolean decision rules using column generation (CG), which allows an algorithm to search

the rule clauses efficiently over the exponential number of candidate clauses pool. Dash et al. (2018) argue CG increases interpretability without sacrificing accuracy. Moreover, Akyüz and Birbil (2021) extend the application of LP-based rule learning from binary classification to multiclassification by introducing RUX and RUG algorithms. They use linear programming (LP) which can handle large datasets, unlike MILP. Also, they take advantage of the CG approach in their RUG algorithm. For the regression machine learning problem, Liu et al. (2013) introduce a **Rule Elimination using 1-norm Regularization** method that uses a 1-norm regularized random forest and corresponding LP problem to reduce the number of features/rules and enhance interpretability. Moreover, this method from Liu et al. (2013) assign rule weights that measure the importance of rules and use it as a decision variable for the LP problem.

Similar to this LP-based **Rule Elimination using 1-norm Regularization** method, our research also assigns costs corresponding to each rule and optimizes the balance between the empirical error and the rule costs. However, our research aims to extend the RUX and RUG developed by Akyüz and Birbil (2021) to the regression problem because these methods have advantages over the **Rule Elimination using 1-norm Regularization** method. First, unlike this method that can only be applied to the RF, RUX can be applied to multiple tree ensemble algorithms (RF, ADA) which enhances the robustness of the method. Second, unlike Liu et al. (2013) that do not assign a particular cost for each rule, RUX and RUG allow users to assign various costs corresponding to the rules. This flexibility allows users to extract and/or generate the rules for their preferences or needs. Furthermore, regarding RUX, Barakat and Diederich (2004) have stated four key points: (1) Provision of an explanation component, (2) Overcome the problem of knowledge acquisition for symbolic AI systems, (3) Data exploration and the induction of scientific theories, (4) Improving the generalization of AI solutions. Regarding RUG, the most impressive contribution is the ability to drastically improve interpretability while minimising the accuracy loss.

Akyüz and Birbil (2021) already stated that their RUX and RUG can be applied to the regression problem and advised that extending these algorithms to the regression problem is an interesting future research direction. However, no further research on this topic has been done. Our research inherits this suggestion by introducing new LP formulations for RUX and RUG.

They are adjusted for the regression problem with a new set of constraints, loss function, and rule function for RUX's master problem and RUG's restricted master problem and pricing sub-problem.

3 Data

We will use nine datasets to execute our candidate algorithms. Dua and Graff (2017) put forward eight of them (ABALONE, AIRFOIL, CONCRETE, GARMENTS, POWERPLANT, BIKE, REDWINE, WHITEWINE) via the UCI Machine Learning Repository. James et al. (2013) introduce the HITTERS dataset. The corresponding sample size and the number of features for each dataset are summarized in Table 1. Unlike most research in the Economics discipline, we usually do not need to do a deep dive into the meaning of features in machine learning papers. Since these features in each dataset are not significant for our research algorithm, we do not explain them in detail in this section. Instead, we include a detailed description of the raw datasets and features in the Appendix (see Table 6 to 14).

These collected datasets are frequently used for regression tasks and for most datasets (ABALONE, AIRFOIL, CONCRETE, POWERPLANT, REDWINE and WHITEWINE), we do not need to clean up or transform the data. We pre-process the dataset HITTERS by removing three features (`League`, `Division`, and `NewLeague`). We do this because these features are not easy to quantify numerically. For the dataset GARMENTS, we do not include the features (`date`, `quarter`, `department`, `date` and `team`) as these features are more relevant to time series analysis. In the BIKE dataset, we use the binary variable to denote the values of the `Holiday` and `Functioning Day` features in which we set *'Holiday'* and *'Function Day'* values to 1 and *'No Holiday'* and *'Not function day'* values to 0. We also eliminate the time related features Date and Seasons for reasons similar to the ones mentioned before.

Table 1: Summary of the Datasets

INSTANCE	SAMPLE SIZE	FEATURES
POWERPLANT	9568	4
AIRFOIL	1503	5
ABALONE	4177	7
CONCRETE	1030	8
GARMENTS	1197	9
BIKE	8760	11
REDWINE	1599	11
WHITEWINE	4898	11
HITTERS	322	16

For the datasets presented in this section, we normalize all features, including the dependent variable. We do this because often attributes have different ranges they cover. Usually, this would lead to features with larger ranges having more influence on the final result of the multivariate regression. Normalising our datasets avoids this issue, as all values will then range from 0 to 1. The formula used to implement normalisation is $x' = \frac{x - x_{min}}{x_{max} - x_{min}}$, where x' is the normalised value.

4 Methodology

This paper formulates two rule-based machine learning algorithms that utilize linear programming techniques, namely RUX and RUG. In particular, RUX is an ex-post machine learning algorithm that extracts the important rules among the rule pool generated by tree-ensemble machine learning methods. In this study, we use conventional random forest (RF) and AdaBoost (ADA) as a base model of RUX. Additionally, we use these two tree-ensemble methods as a benchmark to assess the performance of RUX and RUG algorithms in terms of mean absolute error and interpretability, which is measured by the number of rules and the average rule length. On the other hand, RUG is a standalone algorithm that is based on the existing column generation (CG) algorithm. In the methodology section, we first give a general introduction to the RF and ADA algorithms. Next, we present our RUX and RUG algorithms and their mathematical formulations that are tailored to the regression dataset with the continuous target variable.

4.1 Random Forest (RF)

Breiman (2001) states that Random forest (RF) is one of the tree ensemble methods that train multiple regression trees using bootstrap subsamples of the data points and features. This model processes every sample to all of the created trees and forecasts the target variable by averaging the forecast outcomes from multiple trees. Breiman (2001) further shows that RF can solve the overfitting problem, which is a typical issue when a single decision tree or regression tree analyzes a complex dataset and hence produces an accurate forecast. More specifically, the bootstrap sampling process divides a single giant and deep tree into several small trees in which overfitting is less likely to occur. Breiman (2001) proves that within each tree, this algorithm uses the mean squared error (MSE) to calculate the deviation of the prediction from the target value. Also, RF uses a splitting rule to find the best partition that minimizes MSE. However, the advantage of obtaining an accurate forecast comes at a cost of interpretability. Liu et al. (2013) point out that as RF utilizes multiple trees to calculate its forecast, it is less intuitive to understand the decision-making process compared to a single regression tree. Note that in this research, we define the ruleset of RF as the set of leaves of the trees in the forest and use the number of leaves (number of rules) as a proxy of interpretability. Namely, a higher number of leaves implies less interpretability of the model. Also, we set RF as a base model for the RUX method, and use it as a benchmark model to assess the accuracy and interpretability of the RUX and RUG methods. This study uses the standard library in the machine learning package in Python to implement the RF algorithm.

4.2 AdaBoost (ADA)

The way we implemented ADA in this paper is via the use of Decision Trees (DT). In contrast to the above-mentioned method of RF, Freund and Schapire (1997) point out that ADA is a method that works by combining many weak DTs, most of which are usually tree stumps (a decision tree made out of only one node and two leaves). A stump produces relatively inaccurate rules as it can only use one variable to make a decision. Then, using a forest that is made out of (mostly) stumps, ADA goes on to classify the samples. An important distinction that is made between RF and ADA is that ADA attributes different weights to this forest of stumps, whereas RF attributes equal weights to all DTs in the forest. Also, each stump is made by accounting for the previous stump's shortcomings, making ADA impossible to parallelize, as the order of

the generated trees matters (whereas in RF each tree is generated independent of the other trees).

More precisely, ADA first builds stumps for each attribute in the dataset and then selects the stump which has the smallest GINI index, as the DT of said attribute correctly classifies the most observations in the sample. Then, Schapire (2013) shows that ADA has to determine the $OutputWeight$, which is the weight attributed to the stump, via the following formula:

$$\texttt{OutputWeight} = \frac{1}{2} log(\frac{1-\texttt{TotalError}}{\texttt{TotalError}})$$

The `OutputWeight` will be a large negative value if the stump consistently predicts the opposite classification of the correct one, and a large positive value if it consistently correctly classifies. Schapire and Freund (2013) suggest that ADA has to use the above mentioned `OutputWeight` to resample the original dataset. This is done using the following formula for the misclassified samples:

$$\texttt{UpdatedSampleWeight} = \texttt{PreviousSampleWeight} * e^{\texttt{OutputWeight}}$$

When the `OutputWeight` is large (which means the previous DT did a good job in classifying most samples), the `PreviousSampleWeight` is scaled by a large value, since that implies the observation needs to be especially taken into account as the DT used for it was generally an accurate one, yet it misclassified it. The weights of the samples which are correctly classified are scaled via a similar formula to that of the misclassified samples, except that `OutputWeight` is multiplied by -1. The minus sign in front of the `OutputWeight` ensures that correctly classified sample weights are going to be decreased proportionally to how accurate the DT was. This way, samples that were misclassified will gain in weight and samples correctly classified will decrease in weight. This will ensure that the following stump will be additionally incentivized to correctly classify the previously misclassified observations. Schapire and Freund (2013) state that the next step is normalizing the sample weights, such that they sum up to 1. We use the normalized weights to create the second stump and iteratively repeat this process. Finally, we sum up the weights of all stumps in the forest and use the outcome of the group of stumps with the largest weight as the prediction of ADA. This model is implemented using a standard library in the machine learning package in Python.

4.3 Rule Extraction (RUX)

As we explained, the performance of the machine learning algorithm increases at a cost of losing interpretability. Rule extraction is an interpretable ML algorithm that finds the optimal set of rules with less cardinality than the original ruleset generated by conventional tree-ensemble methods while providing reasonable predictive performance. It is done by solving linear programming (LP) problems that minimize the sum of loss function and cost multiplied by weight for every rule, where the weight is a decision variable that represents the relative importance of each rule to calculate prediction. Here, the cost represents the reduction of interpretability by having the rule in the ruleset. This LP returns optimal weights for each rule that balances predictive performance and interpretability. More specifically, if the optimal weight of a certain rule is smaller than a certain threshold value close to zero, then we discard that rule from our optimal RUX ruleset as it does not significantly improve performance, while the cost is substantial. Furthermore, Akyüz and Birbil (2021) suggest that this rule extracting algorithm is suitable for large data sets as it applies LP as a base.

Courtesy of the notation and findings presented by Akyüz and Birbil (2021), suppose the tree-ensemble base model generates a set of $\mathcal{J}$ rules. Given the analysed Random Forest model as an example, the set of rules $\mathcal{J}$ is taken as the resulting leaf nodes. A key metric for the interpretability of the model is reaching a solution with as few rules (leaf nodes) as possible, namely to minimize the $\mathcal{J}$ used. Akyüz and Birbil (2021) attribute the corresponding predicted label for each sample $i \in \mathcal{I}$.

Denote the predicted value, $\hat{y} = \sum_{j \in \mathcal{J}} a_{ij} R_j w_j$, where $R_j = \frac{\sum_{i \in \mathcal{I}} a_{ij} y_i}{\sum_{i \in \mathcal{I}} a_{ij}}$, w_j is the set of non-negative weights and a_{ij} is an indicator variable which equals 1 if sample i is covered by rule j, 0 otherwise. Here, R_j represents the average of y_i values that satisfy rule j. We consider this R_j value as an partial result of target variable corresponding to each rule. Recall that RF takes average of the partial result of each leaves in multiple trees to obtain the final prediction $\hat{y}_i$. For the RUX algorithm, we take weighted average of the R_j using weight w_j, which represent relative importance of the rule j to obtain $\hat{y}_i$.

Unlike a_{ij} and R_j which we can directly obtain from the trained base tree-ensemble algorithm,

w_j is a decision variable that is not yet determined after training the base algorithm. To obtain optimal weights, we formulate the following LP problem (1).

Given a linear programming problem, we use the absolute loss function to minimize the total value of the errors which are calculated as the sum of the absolute values of the differences between the observed values of y_i and the predicted values, $\hat{y}_i$:

$$min \sum_{i \in \mathcal{I}} |y_i - \hat{y}_i|$$

To formulate linear objective function, we introduce auxiliary decision variable v_i. If $y_i \geq \hat{y}_i$, it is defined as $v_i \geq y_i - \hat{y}_i, i \in \mathcal{I}$. On the other hand, if $y_i \leq \hat{y}_i$, it is defined as $v_i \geq \hat{y}_i - y_i, i \in \mathcal{I}$. In words, v_i a value that is greater or equal to the absolute difference between realized value and the prediction. Therefore, v_i is a measure of predictive error (accuracy) for each sample i. With the help of v_i, we formulate our master problem:

$$\begin{aligned}
&\text{minimize } \textstyle\sum_{i \in \mathcal{I}} v_i + \sum_{j \in \mathcal{J}} c_j w_j \\
&\text{subject to } \mathrm{v}_i \geq y_i - \hat{y}_i, i \in \mathcal{I}; \\
&\mathrm{v}_i \geq \hat{y}_i - y_i, i \in \mathcal{I}; \\
&\mathrm{v}_i \geq 0, i \in \mathcal{I}; \\
&\mathrm{w}_j \geq 0, j \in \mathcal{J}
\end{aligned} \tag{1}$$

The cost coefficient c_j is especially important as it ensures the end result does not present many rules (leading to easier to interpret solutions), while at the same time accounting for the costs associated with each rule. The cost coefficient is defined differently depending on the base tree-ensemble model. Akyüz and Birbil (2021) suggest using the length of the rule as a cost for Random Forest and the inverse of the estimator weights to the trees as a cost for AdaBoost.

Hence, this minimization of the objective function in LP problem (1) aims at finding the sweet spot in the error minimization-interpretability trade-off. Accordingly, the first component of the objective function represents the sum of errors and it may decrease by assigning positive non-zero weights to the rule j, but it comes at the expense of an increase in the second component, which represents the cost of having more rules and obtaining an uninterpretable result.

The first and second sets of constraints are mutually exclusive because v_i is defined differently depending on the sign of $y_i - \hat{y}_i$. For example, if sample i is constrained under the first set of constraints, then the corresponding constraint in the second set of constraints is automatically satisfied. Note that we deviate from the paper of Akyüz and Birbil (2021) by relaxing their set covering constraints.

After solving this LP problem, we obtain optimal weights for all the rules, w_j^*. Using this primal LP solution, we calculate RUX prediction: $\hat{y}_i = \sum_{j \in \mathcal{J}} a_{ij} R_j w_j^*$. Akyüz and Birbil (2021) prove that implementing RUX on the trained Random Forest model (used to construct the master problem) leads to more time and computation efficient models without greatly decreasing accuracy. In the Results section, it is shown whether this finding also holds for continuous data sets.

4.4 Rule Generation (RUG)

Rule Generation learning method (RUG) is an interpretable method introduced by Akyüz and Birbil (2021). This method is heavily based on the Column Generation (CG) learning method. The idea behind this is that instead of taking the complete problem set, now, with the subset (rule pools) we could perform a more efficient prediction using the divide and conquer approach. Akyüz and Birbil (2021) point out that this method is optimal when the entire set $\mathcal{J}$ is not available or the size of the set is too large (too costly) to find the optimal solutions. From the given problem and the LP formulation (1) in the RUX method, due to Desaulniers et al. (2006) and the re-examination from the paper of Akyüz and Birbil (2021), the rules are referred to as the columns with an LP-based model. Taking the subset for each iteration with the LP-based formulation, the case is now called Restricted Master Problem (RMP). Solving this RMP could help us to get the dual optimal solution. From these results and the objective function of RMP, a sub-pricing problem could be formed and solved as well. This sub-pricing is aimed to identify the rules (columns) and whether they should be added to the RMP for the next iteration or not. Once it is a negative reduced cost rule, it could be added to the RMP to optimize the final solution.

Inheriting the idea of classification case of Akyüz and Birbil (2021) and LP problem (1), the RUG formulation for regression could be formed as the dual problem of the primal LP formulation in RUX. Akyüz and Birbil (2021) have confirmed that this RUG method could be performed efficiently with continuous data set. Hereby, this section would provide the theoretical base and new formulation for continuous data for the method, we would set the formulation with the methods are being used, and the reason for this set-up.

Akyüz and Birbil (2021) show it is possible to formulate the RMP by transforming a primal LP model from the RUX algorithm. Define a subset of rules (column pools) $\mathcal{J}_t \in \mathcal{J}$ with t as the iteration of the algorithm. We denote the dual variables for the formulation as β_i and γ_i as $i \in \mathcal{I}$. In this dual formulation, the vector of dual variables β corresponds to the first set of constraints in the primal problem and the vector of dual variables γ corresponds to the second set of constraints. Similarly, the first set of dual constraints corresponds to the primal decision variables $v_i, \forall i \in \mathcal{I}$ and the second set of dual constraints corresponds to the primal decision variables $c_j, \forall j \in \mathcal{J}_t$.

$$
\begin{aligned}
&\text{maximize } \textstyle\sum_{i=1}^{\mathcal{I}}(y_i\beta_i + (-y_i)\gamma_i) \\
&\text{subject to: } \beta_i + \gamma_i \leq 1, \forall i \in \mathcal{I} \\
&\textstyle\sum_{i=1}^{\mathcal{I}}(a_{ij}R_j\beta_i - a_{ij}R_j\gamma_i) \leq c_j \forall j \in \mathcal{J}_t, and \mathcal{J}_t \subset \mathcal{J} \\
&\beta_i \geq 0, \forall i \in \mathcal{I} \\
&\gamma_i \geq 0, \forall i \in \mathcal{I}
\end{aligned}
\tag{2}
$$

After solving the problem (2), we obtain optimal dual variables. Denote the optimal dual variables calculated at t^{th} iteration as $\beta_i^{(t)}$ and $\gamma_i^{(t)}$. Next, we aim to find and select rules out of a potential rule pool that improves our objective value. We denote a candidate rule as $j' \in \mathcal{J}/\mathcal{J}_t$. Birbil et al. (2020) state that such rules are the ones that satisfy the negative reduced cost condition given by

$$
c_{j'} - \sum_{i=1}^{\mathcal{I}}(a_{ij'}R_{j'}\beta_i - a_{ij'}R_{j'}\gamma_i) < 0 \tag{3}
$$

To find a rule that satisfies this condition (3), we should maximize the second component of the reduced cost formula. This maximization problem is a *pricing subproblem*. Birbil et al. (2020) and Akyüz and Birbil (2021) prove that we need to solve this problem by training a decision tree with a dual optimal solution as a sample weight vector to obtain the best performing rule

as a candidate rule, thus checking whether they satisfy condition (3). Note that this approach is a proxy for the actual pricing subproblem. More specifically, we formulate two different *proxy pricing subproblems* by defining a decision tree and its sample weight differently.

First, as our aim is to maximize $\sum_{i=1}^{\mathcal{I}}(a_{ij'}R_{j'}\beta_i - a_{ij'}R_{j'}\gamma_i)$, we should give higher weight to the sample i with larger β_i and smaller γ_i. Therefore, we may use $(\beta-\gamma)$ as a sample weight vector. However, this definition has a computational limitation. Pedregosa et al. (2011) show that if $\beta_i < \gamma_i$, corresponding elements of the weight vector have negative values and such values are ignored when constructing a decision tree. As a remedy, we normalize the elements in the $(\beta-\gamma)$ such that every element lie between 0 and 1. We refer to the RUG using this approach as RUG1.

Second, note that the optimal dual variables represent the change of the objective value of the primal problem (1) when there is a marginal relaxation or tightness of the corresponding constraints. Birbil et al. (2020) prove that since optimal dual variables (β_i and γ_i) correspond to the sample $i \in \mathcal{I}$, the β_i^* and γ_i^* represent the relative importance of the sample i for the objective value of the problem (1). Therefore, for each iteration, we use $|\beta+\gamma|$ as a sample weight vector and $|y-\hat{y}|$ as a dependent variable, where $\hat{y}$ is RUG prediction with the current rule pool in this iteration. Aware that the values of this sample weight vector and dependent variable change over iterations. We refer to the RUG with this proxy pricing subproblem formulation as RUG2.

After training the decision tree using these two alternative methods, we assess whether the generated rules satisfy the negative reduced cost condition. If one of the rules has a negative cost, then we include such a rule ($\bar{\mathcal{J}}_-$) into our rule pool ($\mathcal{J}_{t-1} \cup \bar{\mathcal{J}}_-$) and iterate by solving RMP with the new rule pool. If not, we terminate the iteration and the current rule pool becomes a final rule pool. Note that as Akyüz and Birbil (2021) suggest, our algorithm train an equally weighted decision tree to obtain the initial set of rules ($\mathcal{J}_0$) which corresponds to the set of leaves.

Also, note that solving the dual problem (2) is equivalent to solving the primal problem (1a). Therefore, we obtain optimal weights for the final rule pool from the RMP at the last iteration. When constructing a set of RUG rules, we only select the rules with optimal weights greater

than the non-negative and non-zero threshold (0.000001). Hence, the RUG ruleset does not necessarily have every rule in the final rule pool. Additionally, optimal weights also affect the RUG prediction: $\hat{y}_i = \sum_{j \in \mathcal{J}_f} a_{ij} R_j w_j^*$, where $\mathcal{J}_f$ denotes the final rule pool.

Figure 1 below is the illustration with a flow chart of the RUG learning method with steps by steps and inherited from the original flow chart of Akyüz and Birbil (2021):

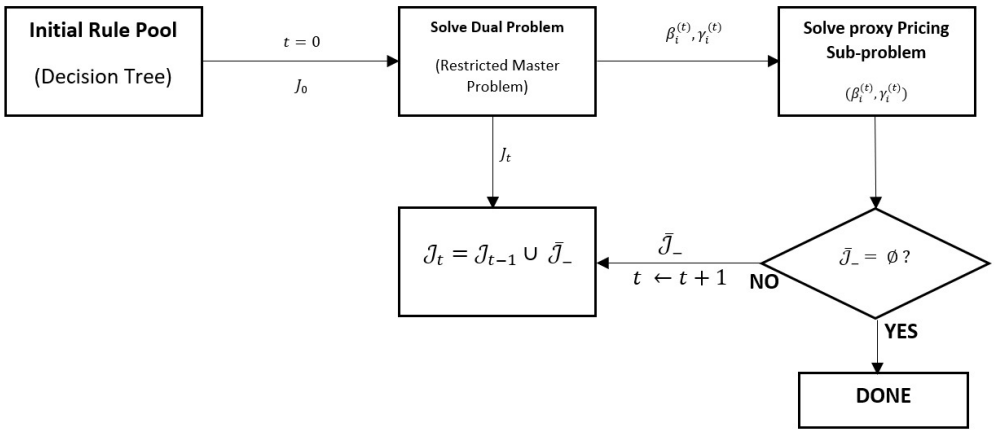

Figure 1: The process of RUG algorithm

5 Results

To obtain these findings below, we use Python and relevant packages from the paper of Akyüz and Birbil (2021) with Python version 3.8, the environment PyCharmCommunity edition and a Windows 10 64-bit PC with Intel(R) Core(TM) i5-7200U CPU with 8.00 GB RAM. Table 2 and Table 3 show the results of all learning methods in terms of mean absolute error (`MAE.`), the number of rules (`RULES`), and the average rule length for each rule (`RULE LEN.`) with different max-depth value. To train the proposed machine learning algorithms, we randomly divide each dataset and use 70% of the total sample as training data and 30% of the total sample as test data. Also, we iterate the training and testing 100 times with different random seeds and report the average results and standard deviations. The value in parenthesis corresponds to the standard deviation.

5.1 Interpretation:

When using Rule Extraction (RUX), Random Forest (RF) and AdaBoost (ADA) are used as trained models, denoted as RUX-RF and RUX-ADA. The max-depth is set to 2 or 3, and the number-of-estimators parameter(the number of trees in tree-ensemble manners) is equal to 100. In order to optimize formulation (1) and formulation (2), we include the cost coefficients corresponding to each rule in the RUX-RF and RUX-ADA models, denoted by c_j, $j \in \mathcal{J}$. We use the length of the rule and the inverse of estimator weights to identify the cost coefficients of RUX-RF and RUX-ADA respectively. In the process of rule generation (RUG), we choose a decision tree (DT) as our initial rule pool.

Tables 2 and Table 3 report the mean absolute error (`MAE.`), number of rules (`#RULES`) and the average length of rules (`RULE LEN.`) when max-depth is 2 or 3 respectively. In both cases, the RUX algorithm did significantly reduce the mean absolute error (`MAE.`) and the number of rules for all datasets. It not only improves the accuracy of predictions but also improves interpretability by reducing the number of rules. Furthermore, when we compare RUX-RF and RUX-ADA, we also find that RUX-RF has a relatively small number of rules, a small rule length (`#RULES, RULELEN.`), and the average mean absolute error (`MAE.`) of RUX-ADA for these nine datasets is relatively small, we can conclude that RUX-RF outperforms RUX-ADA in interpretability and RUX-ADA outperforms RUX-RF in prediction performance.

Table 2: The results of all learning methods when the Max-depth is set to 2

DATASET	RF		ADA		RUX-RF			RUX-ADA			RUG1			RUG2			DT	
	MAE.	#RULES	MAE.	#RULES	MAE.	#RULES	RULE LEN.	MAE.	#RULES	RULE LEN.	MAE.	#RULES	RULE LEN.	MAE.	#RULES	RULE LEN.	MAE.	#RULES
ABALONE	0.066	400	0.097	128.2	0.061	26.9	1.048	**0.058**	31.3	1.666	0.064	4.4	1.020	0.061	10.8	1.488	0.068	4
	(0.001)	(0.000)	(0.008)	(44.239)	(0.002)	(2.891)	(0.026)	(0.002)	(7.161)	(0.117)	(0.002)	(0.566)	(0.098)	(0.002)	(2.918)	(0.168)	(0.001)	(0.000)
AIRFOIL	0.110	400	0.100	148.7	0.095	36.1	1.995	**0.074**	43.5	1.893	0.116	4.2	2.000	0.101	13.6	1.905	0.116	4
	(0.004)	(0.000)	(0.005)	(57.034)	(0.005)	(4.087)	(0.014)	(0.006)	(8.372)	(0.043)	(0.004)	(0.409)	(0.000)	(0.011)	(4.056)	(0.105)	(0.004)	(0.000)
CONCRETE	0.111	400	0.092	200.0	0.088	35.0	1.994	**0.060**	59.2	1.977	0.122	4.2	2.000	0.107	9.5	1.974	0.122	4
	(0.005)	(0.000)	(0.004)	(0.000)	(0.005)	(4.167)	(0.015)	(0.003)	(3.609)	(0.018)	(0.005)	(0.735)	(0.000)	(0.008)	(2.865)	(0.051)	(0.005)	(0.000)
GARMENTS	0.121	400	0.126	43.0	0.102	24.5	1.839	**0.100**	20.2	1.784	0.114	7.4	1.858	0.112	10.2	1.815	0.126	4
	(0.005)	(0.000)	(0.006)	(11.718)	(0.007)	(3.562)	(0.059)	(0.006)	(3.934)	(0.106)	(0.007)	(1.275)	(0.155)	(0.007)	(2.360)	(0.135)	(0.005)	(0.000)
HITTERS	0.097	398.9	0.127	176.7	**0.094**	36.3	1.903	0.102	30.4	1.959	0.106	4.7	1.936	0.102	7.0	1.934	0.110	4
	(0.009)	(1.489)	(0.010)	(40.384)	(0.010)	(3.756)	(0.043)	(0.010)	(5.537)	(0.037)	(0.011)	(1.379)	(0.149)	(0.013)	(1.679)	(0.098)	(0.010)	(0.000)
POWERPLANT	0.062	400	0.061	83.0	0.055	65.1	1.053	**0.040**	42.2	1.624	0.067	7.3	1.073	0.055	17.1	1.605	0.067	4
	(0.002)	(0.000)	(0.002)	(26.645)	(0.001)	(4.990)	(0.022)	(0.001)	(6.563)	(0.056)	(0.001)	(1.328)	(0.152)	(0.004)	(4.457)	(0.108)	(0.001)	(0.000)
BIKE	0.096	400	0.126	186.0	0.089	33.9	1.620	**0.072**	28.2	1.865	0.097	4.9	1.542	0.089	8.2	1.618	0.099	4
	(0.002)	(0.000)	(0.006)	(7.385)	(0.003)	(4.100)	(0.090)	(0.002)	(2.665)	(0.038)	(0.002)	(1.051)	(0.081)	(0.002)	(1.119)	(0.068)	(0.002)	(0.000)
REDWINE	0.108	400	0.107	106.6	**0.093**	15.0	1.900	0.096	26.3	1.983	0.102	4.1	1.950	0.101	7.4	1.966	0.115	4
	(0.003)	(0.000)	(0.004)	(47.894)	(0.005)	(7.357)	(0.111)	(0.004)	(17.658)	(0.037)	(0.005)	(0.384)	(0.180)	(0.006)	(2.251)	(0.084)	(0.003)	(0.000)
WHITEWINE	0.102	400	0.100	89.8	**0.090**	10.4	1.739	0.091	11.3	1.890	0.095	4	1.955	0.091	7.8	1.928	0.104	4
	(0.002)	(0.000)	(0.002)	(32.769)	(0.002)	(1.994)	(0.131)	(0.003)	(7.995)	(0.123)	(0.002)	(0.000)	(0.143)	(0.003)	(2.368)	(0.113)	(0.002)	(0.000)
Average	0.097	399.9	0.104	129.1	0.085	31.5	1.677	**0.078**	32.5	1.849	0.098	5.0	1.704	0.091	10.2	1.804	0.103	4
	(0.003)	(0.165)	(0.005)	(29.785)	(0.004)	(4.100)	(0.057)	(0.004)	(7.055)	(0.064)	(0.004)	(0.792)	(0.106)	(0.006)	(2.675)	(0.103)	(0.004)	(0.000)

Table 3: The results of all learning methods when the Max-depth is set to 3

DATASET	RF		ADA		RUX-RF			RUX-ADA			RUG1			RUG2			DT	
	MAE.	#RULES	MAE.	#RULES	MAE.	#RULES	RULE LEN.	MAE.	#RULES	RULE LEN.	MAE.	#RULES	RULE LEN.	MAE.	#RULES	RULE LEN.	MAE.	#RULES
ABALONE	0.062	800	0.085	336.9	**0.056**	91.2	1.607	0.057	126.9	2.497	0.061	8.3	1.418	0.057	23.4	2.025	0.064	8
	(0.001)	(0.000)	(0.005)	(83.528)	(0.002)	(18.479)	(0.063)	(0.002)	(26.079)	(0.083)	(0.002)	(0.646)	(0.227)	(0.002)	(4.910)	(0.167)	(0.001)	(0.000)
AIRFOIL	0.099	800.0	0.084	395.3	0.075	94.4	2.323	**0.056**	114.4	2.440	0.107	10.4	2.256	0.082	33.4	2.505	0.108	8
	(0.004)	(0.099)	(0.004)	(21.446)	(0.004)	(6.366)	(0.045)	(0.003)	(7.570)	(0.046)	(0.004)	(1.336)	(0.130)	(0.009)	(7.459)	(0.134)	(0.004)	(0.000)
CONCRETE	0.092	800	0.079	399.4	0.069	94.7	2.691	**0.053**	133.9	2.817	0.104	9.1	2.514	0.092	21.8	2.639	0.105	8
	(0.005)	(0.000)	(0.003)	(0.864)	(0.004)	(8.807)	(0.046)	(0.003)	(6.993)	(0.033)	(0.005)	(1.636)	(0.148)	(0.006)	(6.241)	(0.109)	(0.004)	(0.000)
GARMENTS	0.108	799.8	0.113	96	**0.092**	64.5	2.461	0.093	45.9	2.441	0.106	12.3	2.562	0.099	18.5	2.620	0.113	8
	(0.005)	(0.517)	(0.005)	(24.298)	(0.006)	(6.583)	(0.077)	(0.006)	(8.504)	(0.106)	(0.006)	(2.069)	(0.192)	(0.006)	(3.384)	(0.138)	(0.005)	(0.000)
HITTERS	0.091	756.4	0.114	368.1	**0.100**	60.6	2.678	0.105	61.6	2.811	0.105	7.2	2.782	0.103	9.3	2.751	0.103	7.3
	(0.009)	(24.891)	(0.008)	(17.845)	(0.013)	(5.458)	(0.071)	(0.011)	(6.306)	(0.059)	(0.013)	(1.756)	(0.181)	(0.013)	(2.126)	(0.168)	(0.012)	(0.458)
POWERPLANT	0.050	800	0.058	323.5	0.048	157.3	1.381	**0.043**	146.2	2.159	0.054	9.1	1.457	0.050	26.2	1.944	0.054	8
	(0.001)	(0.000)	(0.003)	(89.440)	(0.001)	(9.641)	(0.039)	(0.001)	(25.397)	(0.056)	(0.001)	(0.978)	(0.059)	(0.002)	(6.475)	(0.152)	(0.001)	(0.000)
BIKE	0.080	800	0.105	365.4	0.074	59.5	2.553	**0.059**	74.3	2.611	0.081	9.7	2.543	0.078	13.7	2.460	0.083	8
	(0.001)	(0.000)	(0.004)	(8.129)	(0.002)	(6.151)	(0.045)	(0.001)	(4.854)	(0.053)	(0.002)	(1.552)	(0.078)	(0.002)	(1.815)	(0.059)	(0.001)	(0.000)
REDWINE	0.104	798.7	0.103	390.9	**0.096**	134.3	2.686	0.099	149.6	2.861	0.096	8.5	2.678	0.098	13.5	2.692	0.109	8
	(0.003)	(1.399)	(0.004)	(39.815)	(0.004)	(15.751)	(0.070)	(0.003)	(17.771)	(0.039)	(0.006)	(1.646)	(0.180)	(0.006)	(5.851)	(0.164)	(0.004)	(0.000)
WHITEWINE	0.098	799.7	0.098	308.2	**0.089**	47.8	2.436	0.092	124.8	2.763	0.092	8.3	2.373	0.091	15.8	2.559	0.100	8
	(0.002)	(0.617)	(0.002)	(98.657)	(0.002)	(19.636)	(0.096)	(0.002)	(55.083)	(0.083)	(0.002)	(0.624)	(0.176)	(0.002)	(3.770)	(0.139)	(0.002)	(0.000)
Average	0.087	795.0	0.093	331.5	0.078	89.4	2.313	**0.073**	108.6	2.600	0.090	9.2	2.287	0.083	19.5	2.466	0.093	7.9
	(0.003)	(3.048)	(0.017)	(42.669)	(0.004)	(10.764)	(0.061)	(0.004)	(17.617)	(0.032)	(0.005)	(1.360)	(0.152)	(0.005)	(4.670)	(0.137)	(0.004)	(0.051)

We compare RUG1/RUG2 with DT and the result shows that the `MAE` of RUG1/RUG2 is relatively small, but at the cost of generating more rules for each dataset. The one exception is when the maximum depth is set to 3, then the HITTERS dataset has a larger `MAE` with a smaller number of rules compared to the DT. If we compare the RUX and RUG1/RUG2 algorithms, it shows that the `MAE` of RUX-RF and RUX-ADA is significantly smaller than that of RUG1/RUG2 since RUG is trading off accuracy for a smaller number of rules (interpretability). This is suggesting that rule generation improves interpretability at the expense of prediction performance. We also find that between RUG1 and RUG2, the mean absolute error (`MAE.`) of RUG2 is relatively small in both cases (Table 2/3), except for instance REDWINE with a maximum depth of 3, the `MAE` of RUG1 and RUG2 are 0.096 and 0.098, respectively, with RUG1 having a lower prediction error. Compared to RUG1, the number of rules is significantly increased in RUG2, so we can conclude that RUG2 and RUG1 have a trade-off between prediction error and interpretability.

When we increase the max-depth from 2 to 3, we observe that the number of rules and the rule length also increase corresponding to the same method for each dataset. Meanwhile, the mean absolute error decreases. Therefore, we can improve the prediction performance by increasing the maximum depth, while sacrificing interpretability.

5.2 Case Study: A closer look at different dataset and translation of results

In this section, we perform an interpretation of both RUX and RUG results with two different cases, both using "Combined Cycle Power Plant dataset(original)" (POWERPLANT) versus two other datasets: (1) "Seoul Bike Sharing dataset (original)" (BIKE) and (2) "Airfoil self-

noise dataset (original)" (AIRFOIL) from Dua and Graff (2017). For this case study, we only focus on the results with max-depth equal to 2 and `RUG2`.

5.2.1 Case 1: RUX and RUG performance on: POWERPLANT and BIKE

The overall information about the two datasets can be seen in Table 1 above. The POWERPLANT dataset has 9568 observations with 4 features, meanwhile, the sample size of BIKE is 8760 but with 11 features. The reason that we chose these for comparison is: Sample sizes of these two datasets are similar while the BIKE has the highest number of features (11) and POWERPLANT has the minimum number of features (4). This could help to examine the partial effect of having a larger number of features on the performance of RUX and RUG by fixing the other factors constant.

In terms of RUX, the Mean Absolute Errors (`MAE`) for POWERPLANT using RUX-RF is 0.055, and 0.040 when using RUX-ADA. Meanwhile, for BIKE, the `MAE` of RUX-RF is 0.089 and `MAE` of RUX-ADA is 0.072. Overall, RUX performs better with POWERPLANT that has fewer features. For these two datasets only, the RUX-ADA is performing better in terms of `MAE`. RUX algorithm is based on the idea to minimize the errors and the costs of choosing high-weighted rules from the rules generated by a tree-ensemble method. Hence, having fewer features would simplifies the process of spotting the high-weighted rules.

In terms of RUG (`RUG2`), the `MAE` for POWERPLANT is 0.067 by generating 7 (7.3) rules, meanwhile, it is 0.097 for BIKE with only 5 (4.9) rules. In this case, the `RUG2` is performing better with POWERPLANT which has fewer features. This result is in concordance with the result of the RUX approach: the dataset with fewer features is better in terms of performance.

5.2.2 Case 2: RUX and RUG performance on: POWERPLANT and AIRFOIL

In this case study, we compare the performance of RUX to RUG via the AIRFOIL dataset and the POWERPLANT dataset. The reason for choosing this pair is to investigate the consequence of having different sample sizes. Both datasets have the same number of features (5) but there is a large gap in the sample size (1503 for AIRFOIL versus 9568 for POWERPLANT).

With regards to the POWERPLANT dataset, the `MAE` has been reported as 0.055 for RUX-RF and 0.040 for RUX-ADA. Meanwhile, for AIRFOIL, the `MAE` is 0.095 for RUX-RF and 0.074 for RUX-ADA. Similar to the first case study, the RUX-ADA performs better in both datasets. The standard deviation is low for both cases, which shows little fluctuation in the range of the values among the iterations. This comparison shows that the RUG-RF and RUX-ADA algorithms perform better for POWERPLANT in general. Hence, in this case study, RUX is performing better with the dataset that has the larger sample size (POWERPLANT). Accordingly, the more observations, the better the algorithms are able to generate rules to minimise the cost. In contrast, with few observations, the algorithms may be affect by the biased data and outliers.

For RUG, `MAE` under POWERPLANT is 0.067 with 7 (7.3) rules. Meanwhile, `MAE` under AIRFOIL is 0.116 with 4 (4.2) rules. Similarly, RUX is performing better under POWERPLANT, which means it is performing better with dataset having more observations.

These two case studies help us to answer the two sub-questions on the performance of the rule-based learning method on datasets with different sizes and features. If the two datasets have the same large size of sample but a different number of features, the one with fewer features performs better. Secondly, with the two datasets having the same small number of features, the sample with a larger number of observations gives better performance than the other. Even though these conclusions shares similarity across the two methods, it is biased to take only three datasets with two different cases. To give a clearer explanation, one more dataset with a small size but large features would be added to make a full comparison.

5.2.3 RUX improvements

In this case study, we take a closer look at the improvements RUX brings in terms of accuracy via the Mean Absolute Error (MAE). Thus we compare the performance of RF versus RUX-RF and ADA versus RUX-ADA on datasets which are similar in all regards other than (1) the number of features and then (2) the sample size.

First assessing (1), the impact RUX has on datasets with varying number of features, we com-

pare POWERPLANT to BIKE. Regarding POWERPLANT (the dataset with a smaller number of features, otherwise comparable to BIKE), there is an improvement from RF to RUX-RF of 0.062 to 0.055. In the case from ADA to RUX-ADA the improvement is from 0.061 to 0.040. The gap of improvement for RUX-RF in terms of `MAE` is 0.007 (for example: $= 0.062 - 0.055$) and similarly we calculate an improvement of 0.021 for RUX-ADA. Considering the BIKE dataset (the dataset with a larger number of features), there is also an improvement from RF to RUX-RF with a gap of 0.007 and from ADA to RUX-ADA with a change of 0.054. From these outcomes, we observe that RUX-RF achieves a comparable improvements and RUX-ADA achieves a larger improvements on the datasets with increasing number of features (BIKE). Although the accuracy is better for the datasets with fewer features, their ability to improve the result is less than the one having more features. All these results and comparisons are in Table 4.

Then assessing (2), the impact RUX has on datasets with varying sample sizes, we compare AIRFOIL to POWERPLANT. In terms of AIRFOIL (the dataset with a smaller sample size, otherwise comparable to POWERPLANT), there is an improvement from RF to RUX-RF of 0.110 to 0.095 with a gap of 0.015. Similarly, the performance of ADA is improved by RUX-ADA from 0.100 to 0.074, leading to a gap of 0.26. From the comparison, the improvement in AIRFOIL is greater than the one in POWERPLANT for both RUX-RF and RUX-ADA. Therefore, with smaller sample size, in this case, the increase in accuracy is higher.

To complete and generalize our summary table, we would take into consideration the HITTERS dataset because it has the smallest sample size (322) but the largest number of features (16). This sums up the four cases in improvement via RUX. This is an interesting case since it shows different outcomes than most datasets. With 9 datasets, there are only 3 datasets (HITTERS, REDWINE, and WHITEWINE) that show the RUX-RF outperform the RUX-ADA. All three datasets shared one similarity: they have a large number of features. The RUX-RF improvement is the highest for a dataset with small sample size and fewer features but RUX-ADA works best at enhancement gap with large sample size and more features. The comparison between four separate datasets shows the interactive effect of the number of features and sample size on the improvements in `MAE`.

Table 4: Summary of the Datasets with RUX performance case study

dataset	Sample Size	#Features	RUX-RF improvement gap	RUX-ADA improvement gap	Characteristics
AIRFOIL	1503	5	0.015	0.026	small size, small feature
POWERPLANT	9568	4	0.007	0.021	large size, small features
HITTERS	322	16	0.003	0.025	small size, large features
BIKE	8760	11	0.007	0.054	large size, large features

5.2.4 Interpretation of RUG rules

In this case, we overview the results with max-depth equal to 2 obtained by `RUG2`. This dataset is "Airfoil Self-Noise" which gave information about the comprise of airfoils from NASA at various wind tunnel speeds and attacked angles thanks to Dua and Graff (2017). This has 5 features (frequency, angle of attack, chord length, free-stream velocity, suction side displacement thickness; all in different units of measurement) which are all continuous and the output is scaled sound pressure level, measured in decibels. By using RUG2 (updated RUG), the predicted error is reported as 0.101 by generating 13 rules. In this section, we provide an insight into how rule weights are attributed in the process of prediction.

Figure 2 and Table 5 give details of the weight and 13 rules being generated at the same time. Our RUG algorithm assigns optimal weight for each rule. In this case study, we would like to examine the percentage of a rule's weight over the total weight. According to Table 5, we observe that all 13 rules contributes differently from each other in descending order. Rule 1 shows the largest contribution with 18.596% overall, meanwhile Rule 13 is the least to contribute with only 0.661%.

Table 5: The rules and weights for AIRFOIL RUG2

	Rule 1	Rule 2	Rule 3	Rule 4	Rule 5	Rule 6	Rule 7	Rule 8	Rule 9	Rule 10	Rule 11	Rule 12	Rule 13	Total weight
Weight	0.5397	0.4671	0.3959	0.3525	0.3019	0.2380	0.1890	0.1640	0.0906	0.0623	0.0602	0.0218	0.0192	2.9022
Percentage	18.596	16.094	13.641	12.145	10.402	8.2	6.512	5.650	3.121	2.146	2.074	0.751	0.661	100

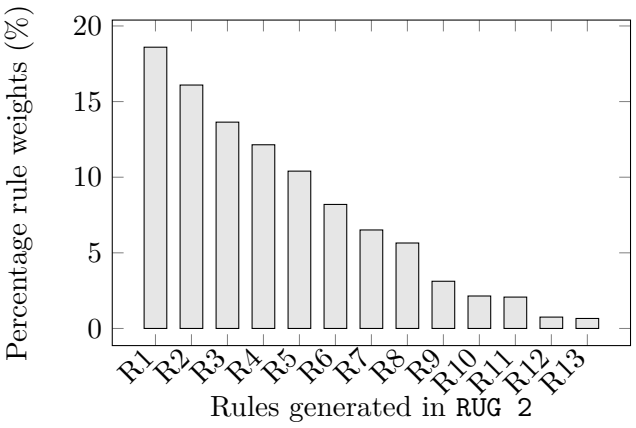

Figure 2: Interpretation plot of the RUG2 on AIRFOIL dataset

6 Conclusion

The central question of this research paper was *"How does the LP-based interpretable learning approach work for regression problems?"*. We found that it is possible to apply RUX and RUG algorithms to the regression case, on a continuous dataset, while solving for the actual pricing sub-problem instead of a proxy for it.

RUX-RF and RUX-ADA offer accuracy improvements over their RF and ADA alternatives across all of the datasets we tested, while at the same time reducing the number of rules used. Regarding RUG 1 and RUG 2, we notice similar or marginally worse accuracy compared to RUX-RF and RUX-ADA, while further improving upon the interpretability of the results.

If we are interested strictly in which method is the most accurate, RUX-ADA outperforms over the 9 datasets we tested, having the lowest average `MAE` out of all methods. On the flip side, out of RF, ADA, RUX-RF, RUX-ADA, RUG 1 and RUG 2, the most interpretable results (low average number of rules, low average rule length) are given by the RUG 1 method.

However, there is no method that is objectively superior to the others. Each method illustrates the well-known accuracy-interpretability trade-off of ML. Furthermore, the extent to which the result of each method differs from the others depends on the dataset it is applied to.

These findings show it is possible to extend the research of Akyüz and Birbil (2021) to con-

tinuous datasets, using RUX and RUG to majorly improve interpretability with little to no compromise in terms of accuracy. This finding has a positive societal impact since ML models are more accurate than many other types of models, leading to increased welfare in various sectors. Such fields are medicine, law and banking, where ML adoption can greatly ramp up in the following years as the crucial topic of interpretability is being improved upon using methods such as the ones presented in this paper. Furthermore, these findings have a positive impact on the scientific community, as LP-based methods may be used to make other conventional ML models more interpretable, not only tree-ensemble models.

A few limitations of our research include the fact that for RUX, even though the total number of rules is significantly reduced compared to RF and ADA, it still ends up being larger than we would like, particularly as we increase the maximum allowed tree depth. Another issue regards the solution to the pricing subproblem of the RUG algorithm. Currently, we normalize the difference between two dual variables, which ends up turning large negative results into small positive values. This is not desirable as we would like to assign high importance to high values that are both negative or positive. Another limitation encountered was that we needed to delete features that were not quantifiable with numerical values, which may pose a problem if those deleted features were relevant to the dependent variable. A final issue would be that there is no clear ranking between the effectiveness of RUX-RF, RUX-ADA and RUG. Their performance is dependent on the dataset they are being used on, which indicates a possible lack of robustness.

Further research on this topic could address the previously-mentioned limitations. One possibility would be to change the cost structure of the LP problem, which would especially benefit the interpretability of the RUX-RF and the RUX-ADA algorithms. This change could ensure that a higher number of rules get increasingly penalized with regard to cost. Furthermore, a cap on the total number of rules could be set to ensure interpretability. Another possible extension could be designing the RUG algorithm such that the sample weights are calculated in an alternative, unbiased way, or such that the pricing subproblem is formulated more fairly. Also, training RUX-RF, RUX-ADA and RUG on a larger pool of datasets may improve the current robustness issues and offer consensus regarding the ranking of the algorithms with regard to the MAE, the total number of rules and the average rule length. An additional extension could fol-

low the track of RUG interpretation from Akyüz and Birbil (2021) to see how specific each rule weighted. For example, how is the case with only Rule 1 involved, the case only 2 rules (Rule 1 and 2), etc. This might take time to modify the code to give the most accurate results. With that target in the near future, we aim at giving a detailed plot also on the predicted errors at each iteration. Finally, designing the RUX algorithm in a way that optimizes not only the total number of rules but also the depth of the individual trees generated and/or the total number of estimators used (the number of DTs in the forest) could be a useful extension.

Appendix

RUG Algorithm

Here by is the pseudocode for the regression RUG learning algorithm, extended and implemented from the version of Akyüz and Birbil (2021) and RUX formulation of this paper in Methodology part 1:

Algorithm: Exact Rule Generation (Akyüz and Birbil, 2021)

Input: `training data`, $(\mathbf{x}_i, y_i \in \mathcal{I})$
`t = 0;`
`Construct initial rule pool,` $\mathcal{J}$,
While True do
`Return optimal` β^t `and` γ^t (solve the restricted master problem)
$\mathcal{J}$: solve pricing subproblem
`if` $\mathcal{J} = \emptyset$ `then` return $\mathcal{J}_t$
end if $t \leftarrow t + 1$
$\mathcal{J}_t = \mathcal{J}_{t-1} \cup \mathcal{J}$ -
end while

Dataset explanation

Dataset POWERPLANT

This POWERPLANT dataset is collected from a combined cycle power plant from 2006 to 2011, it contains 9568 samples and 4 explanatory attributes (T, AP, RH, V) and we going to use these features to predict the electrical energy output (EP) of plant per hour.

Table 6: Meaning of data input variables - POWERPLANT data

Input variable	Meaning
T	The Temperature
AP	Ambient Pressure
RH	Relative Humidity
V	Exhaust Vacuum
EP	The electrical energy output net hourly

Dataset AIRFOIL

It is a NASA dataset with different size NACA 0012 airfoils at different wind tunnel speeds an d angles of attack. The airfoil dataset includes 1503 instances and six input variables.

Table 7: Meaning of data input variables - AIRFOIL data

Input variable	Meaning
Frequency	Frequency of airfoils
Angle of attack	Angle of attack
Chord length	The chord length of airfoils
Free-stream velocity	The free-stream velocity of airfoils
Suction side displacement thickness	The suction side displacement thickness of airfoils
Scaled sound pressure level	The scaled sound pressure level of airfoils

Dataset ABALONE

We plan to use physical measurements of the abalone to predict its age instead of cutting the shell by cones, staining and then counting the rings by microscope. This dataset includes 4177 instances and 9 attributes (Sex, Length, Diameter, Height, Whole weight, Shucked weight, Viscera weight, Shell weight, Rings).

Table 8: Meaning of data input variables - Abalone data

Input variable	Meaning
Length	Length of the abalone
Diameter	Diameter of the abalone
Height	Height of the abalone
Whole weight	Whole weight of the abalone
Shucked weight	Shucked weight of the abalone
Viscera weight	Viscera weight of the abalone
Shell weight	Shell weight of the abalone
Rings	Number of rings present of the abalone

Dataset CONCRETE

This dataset wants to have a research on the relation between concrete compressive strength and concrete ingredients. There are totally 1030 instances and 9 input variables.

Table 9: Meaning of data input variables - CONCRETE data

Input variable	Meaning
Cement	The Cement
Blast Furnace Slag	The Blast Furnace Slag
Fly Ash	The Fly Ash
Water	The Water
Superplasticizer	The Superplasticizer
Coarse Aggregate	The Coarse Aggregate
Fine Aggregate	The Fine Aggregate
Age	The Age
Concrete compressive strength	The Concrete compressive strength

Dataset GARMENT

This garment dataset contains important attributes of the garment manufacturing process and predicts employee productivity, there are totally 1197 instances and 15 input variables are included.

Table 10: Meaning of data input variables - GARMENT data

Input variable	Meaning
Date	The Date
Day	Day of the week
Quarter	The quarter in a year
Department	The associated department corresponding to the instance
Team_No	Associated team number with the instance
No_Of_Workers	Number of workers in each team
No_Of_Style_Change	The number of changes in a particular product style
Targeted_Productivity	The daily target productivity for each team
Smv	The allocated time for a task in standard minute value
Wip	Work in progress also included the number of unfinished items
Over_Time	Denote the amount of over time in minutes for each team
Incentive	Denote the amount of financial incentive that enables a particular action
Idle_time	Denote the amount of time when the production stops
Idle_men	Denote the number of idle workers
Actual_Productivity	The percentage of real productivity provided by workers

Dataset BIKE

This bike dataset includes the number of public bikes rented per hour in Seoul's bike-sharing system along with the corresponding weather data and holiday information in which there has 8760 instances and 14 input variables inside.

Table 11: Meaning of data input variables - BIKE data

Input variable	Meaning
Date	The Date year-month-day
Hour	Hour of the day
Temperature	The Temperature
Humidity	The Humidity
Windspeed	The Windspeed
Visibility	The Visibility
Dew point temperature	The Dew point temperature
Solar radiation	The Solar radiation
Rainfall	The Rainfall
Snowfall	The Snowfall
Seasons	The seasons in a year
Holiday	Holiday/No holiday
Functional Day	NoFunction(Non Functional Hours)/Function(Functional hours)
Rented Bike Count	Denote the number of bikes rented per hour

Dataset REDWINE

This redwine dataset use the Portuguese "Vinho Verde" red wine to conduct a chemical analysis on the wine quality, this dataset has 1599 instances and 12 input varibales.

Table 12: Meaning of data input variables - REDWINE data

Input variable	Meaning
Fixed Acidity	The Fixed Acidity
Volatile Acidity	The Volatile Acidity
Citric Acid	The Citric Acid
Residual Sugar	The Residual Sugar
Chlorides	The Chlorides
Free Sulfur Dioxide	The Free Sulfur Dioxide
Total Sulfur Dioxide	The Total Sulfur Dioxide
Density	The Density
pH	The pH
Sulphates	The Sulphates
Alcohol	The Alcohol
Quality	The Quality

Dataset WHITEWINE

This whitewine dataset use the Portuguese "Vinho Verde" white wine to conduct a chemical analysis on the wine quality, this dataset has 4898 instances and 12 input varibales.

Table 13: Meaning of data input variables - WHITEWINE data

Input variable	Meaning
Fixed Acidity	The Fixed Acidity
Volatile Acidity	The Volatile Acidity
Citric Acid	The Citric Acid
Residual Sugar	The Residual Sugar
Chlorides	The Chlorides
Free Sulfur Dioxide	The Free Sulfur Dioxide
Total Sulfur Dioxide	The Total Sulfur Dioxide
Density	The Density
pH	The pH
Sulphates	The Sulphates
Alcohol	The Alcohol
Quality	The Quality

Dataset HITTERS

We use the 'Hitters' dataset which has the yearly salary (SALARY) of 323 Major League Baseball (MLB) hitters in the 1987 season as a target variable. To explain and predict the players' salary, 20 input variables describing their performance are available. Most of these variables except for the league and division that the players played in are continuous numerical values. Also, the target variable is a monetary value (in thousands of dollars) and hence it is a continuous variable. This dataset is relevant to our research about interpretability, as players may willing to know about the reason for their wage cut if their salary is calculated based on machine learning.

Table 14: Meaning of data input variables - MLB data

Input variable	Meaning
AtBat	Total number of batter's turn batting against a pitcher
Hits	Total number of hits were made by each player
HmRuns	Total number of HomeRuns were made by the player
Runs	Total number of runs were made by the player
RBI	Runs Batted In value for each player
Walks	Total of walks by the player
CAtBat	Number of times at bat during his career
CHits	Number of hits during his career
CHmRuns	Number of home runs during his career
CRuns	Number of runs during his career
CRBI	Number of runs batted in during his career
CWalks	Number of walks during his career
Years	Total years that player were in MLB
League	Which league: N - National, A - America
Division	Which division E - East, W - West
PutOuts	Total value of Put-outs
Assists	Total value of Assists
Errors	Total errors were made
Salary	The salary of the players
NewLeague	New League

References

Akyüz, M. H. and Birbil, Ş. İ. (2021). Discovering classification rules for interpretable learning with linear programming. *arXiv preprint arXiv:2104.10751*.

Barakat, N. and Diederich, J. (2004). Learning-based rule-extraction from support vector machines.

Birbil, S. I., Edali, M., and Yuceoglu, B. (2020). Rule covering for interpretation and boosting. *arXiv preprint arXiv:2007.06379*.

Bottou, L., Curtis, F. E., and Nocedal, J. (2018). Optimization methods for large-scale machine learning. *Siam Review*, 60(2):223–311.

Breiman, L. (2001). Random forests. *Machine learning*, 45(1):5–32.

Brennan, T. and Oliver, W. L. (2013). Emergence of machine learning techniques in criminology: implications of complexity in our data and in research questions. *Criminology & Pub. Pol'y*, 12:551.

Cui, Z. and Gong, G. (2018). The effect of machine learning regression algorithms and sample size on individualized behavioral prediction with functional connectivity features. *Neuroimage*, 178:622–637.

Dash, S., Gunluk, O., and Wei, D. (2018). Boolean decision rules via column generation. *Advances in neural information processing systems*, 31.

Demiriz, A., Bennett, K. P., and Shawe-Taylor, J. (2002). Linear programming boosting via column generation. *Machine Learning*, 46(1):225–254.

Desaulniers, G., Desrosiers, J., and Solomon, M. M. (2006). *Column generation*, volume 5. Springer Science & Business Media.

Dua, D. and Graff, C. (2017). UCI machine learning repository.

Freund, Y. and Schapire, R. E. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, 55(1):119–139.

Fürnkranz, J. (1999). Separate-and-conquer rule learning. *Artificial Intelligence Review*, 13(1):3–54.

James, G., Witten, D., Hastie, T., and Tibshirani, R. (2013). *An introduction to statistical learning*, volume 112. Springer.

Jordan, M. I. and Mitchell, T. M. (2015). Machine learning: Trends, perspectives, and prospects. *Science*, 349(6245):255–260.

Li, X. and Liu, B.-R. (2014). Rule-based classification. In *Data Classification: Algorithms and Applications*.

Litjens, G., Kooi, T., Bejnordi, B. E., Setio, A. A. A., Ciompi, F., Ghafoorian, M., Van Der Laak, J. A., Van Ginneken, B., and Sánchez, C. I. (2017). A survey on deep learning in medical image analysis. *Medical image analysis*, 42:60–88.

Liu, S., Dissanayake, S., Patel, S., Dang, X., Mlsna, T., Chen, Y., and Wilkins, D. (2013). Rule based regression and feature selection for biological data. In *2013 IEEE International Conference on Bioinformatics and Biomedicine*, pages 446–451. IEEE.

Malioutov, D. and Varshney, K. (2013). Exact rule learning via boolean compressed sensing. 28(3):765–773.

Mangasarian, O. L., Street, W. N., and Wolberg, W. H. (1995). Breast cancer diagnosis and prognosis via linear programming. *Operations Research*, 43(4):570–577.

Marsland, S. (2011). *Machine learning: an algorithmic perspective*. Chapman and Hall/CRC.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Reif, M. and Shafait, F. (2014). Efficient feature size reduction via predictive forward selection. *Pattern Recognition*, 47(4):1664–1673.

Schapire, R. E. (2013). Explaining adaboost. In *Empirical inference*, pages 37–52. Springer.

Schapire, R. E. and Freund, Y. (2013). Boosting: Foundations and algorithms. *Kybernetes*.

Wang, T. and Rudin, C. (2015). Learning optimized or's of and's. *CoRR*, abs/1511.02210.

Weiss, S. M. and Indurkhya, N. (1995). Rule-based machine learning methods for functional prediction. *CoRR*, abs/cs/9512107.