JOURNAL OF
EMERGING INVESTIGATORS

Article

Explainable AI tools provide meaningful insight into rationale for prediction in machine learning models

Soham Lahiri[1], Huaduo Wang[2,3], Parth Padalkar[2,3], Gopal Gupta[2,3]

[1] Lebanon Trail High School, Frisco, Texas

[2] Department of Computer Science, University of Texas at Dallas, Texas

[3] Applied-Logic, Programming Languages, and Systems Lab (ALPS), Department of Computer Science, University of Texas at Dallas, Texas

SUMMARY

The advanced machine learning (ML) algorithms in artificial intelligence (AI) are complex, with limited insight into the interpretability of these models. Explainable AI (XAI) is an emerging research field that seeks to clarify the rationale behind the outcomes predicted by ML and AI models. We hypothesized that contemporary XAI tools, such as the FOLD-R++ (Improved First Order Learner of Default with Recursive Rules) algorithm and the s(CASP) (System for Answer Set Programming with Constraints) utility, could achieve accuracy within 5% of those from well-known ML algorithms. FOLD-R++ offers meaningful reasoning through justification trees for predicted outcomes, similar to LIME (Local Interpretable Model-Agnostic Explanations) used with ML algorithms. We employed FOLD-R++, an automated inductive learning algorithm, and s(CASP), a reasoning interpreter utility, on ten binary classification datasets from various domains. FOLD-R++ consistently produced accuracy within 5% of those from ML algorithms across all datasets and did so in less processing time in most of the tests. The rulesets generated by FOLD-R++ were processed in s(CASP), which uses constraints in Answer Set Programming to create easily understandable reasoning trees. Finally, ML models were run through LIME to generate graphical representations highlighting dominant parameters affecting outcomes. This analysis approach can be a standard pattern for gaining insights into binary classification machine learning problems.

INTRODUCTION

Many advanced machine learning (ML) algorithms use sophisticated, complex mathematical models with several inputs and other hyper-tuning parameters but do not provide enough information about the internal processing logic. So, they are like black boxes, which can provide accurate results within their dataset domain but often offer little transparency for the reasoning and rationale behind the prediction of an outcome. As a result, there is a growing interest in seeking answers that explain the logic and rationale behind the results generated by these algorithms (1). This gave rise to a whole new area of research in Explainable AI (XAI) (2). XAI helps us to understand the reasoning behind the predictions of an ML algorithm, thus making its results, where the processing logic becomes more comprehensible. However, this field of study is new, and there has been continuous advancement. The XAI techniques available at present can provide meaningful insight into the rationale for predictions in supervised learning problems (3).

Binary classification is a popular set of ML problems where the outcome is restricted to two values. We sought to understand how XAI may help provide insight to many common binary classification problems, focusing on FOLD-R++ among the different XAI tools available in the public domain (4). Our primary objective was to test our hypothesis with binary classification problems that FOLD-R++ not only provides accuracy like other ML algorithms but also has the distinct advantage of generating justification with decision trees, and dataset rules that can be presented in a human-readable form. As a secondary objective, we also wanted to validate that our other XAI tool in consideration, LIME, will be able to provide a meaningful graphical interpretation of the reasoning behind a prediction outcome (5).

For our analysis, we picked ten binary classification datasets from different domains available in Kaggle for public research (6). For each dataset, we first looked at the accuracy of predictions from well-known supervised learning algorithms like Logistic regression (LR), Random Forest (RF), X-Gradient Boost (XGB), Gaussian Naïve Bayes (GNB), and K Nearest Neighbor (KNN). We then used the same datasets with Fold-R++ to validate if we can achieve comparable accuracy. In addition, we took test data samples and observed how the FOLD-R++ algorithm generated the human-readable reasoning rules in conjunction with the s(CASP) utility. Finally, we used the fitted model from different ML algorithms with LIME to generate a graphical representation of the reasoning rationale for the predicted result.

Our analysis of all the different datasets confirmed that the FOLD-R++ algorithm achieves comparable accuracy within 5% of the ones obtained from the standard ML algorithms. We also observed that the processing time of FOLD-R++ was, in some cases, less than the processing time for ML algorithms. However, the most significant advantage of FOLD-R++ and s(CASP) was the generation of the human-readable reasoning rules behind each predicted result. In addition, we confirmed that LIME generated effective graphical representations that highlighted dominant parameters contributing to the outcome of any prediction test case. This exercise provided a template process for interpretability of prediction in binary classification problems and can be easily applied to any dataset.

https://doi.org/10.59720/23-341

RESULTS

The overall workflow for analyzing each of the ten binary classification datasets started with running the FOLD-R++ program with the dataset as input. The datasets were chosen from various domains like healthcare, banking, finance, and manufacturing to provide a variety of data. The datasets were also chosen to provide a spread of data volume in terms of number of input parameters and total number of records in the dataset. The rules generated by Fold-R++ were then fed to the s(CASP) utility along with an instance of the test dataset (7). The final output from s(CASP) was a human-readable reasoning tree that explained the logic behind the predicted outcome (**Figure 1**).

For the explainer phase, the rules generated by FOLD-R++ were entered in a data file prefixed by a test data instance record. s(CASP) utility processed the input file to create the reasoning trees for the predicted outcome. LIME was used to generate a graphical representation highlighting prominent parameters responsible for the result of a specific test data instance. The model generated by ML algorithms was used as input for LIME along with a test record to generate the graphical representation of the reasoning insights for the predicted outcome (**Figure 2**).

The explainer is a data object that LIME uses to generate the list of dominant parameters responsible for a specific prediction. So, the explainer was then invoked with candidate test data rows to generate at least a positive and negative outcome. The explaining data was retrieved by calling the function explain_instance(). Finally, the graphical representations were generated using the show_in_notebook() routine in the LIME library.

The bank marketing dataset from the Kaggle repository was selected as a candidate dataset for our analysis (8). The data is captured from a marketing campaign of a Portuguese bank where phone calls were made to clients. The client's different profile characteristics were noted along with the final binary outcome of whether the client ended up opening a term deposit with the bank. The dataset consists of 45,211 rows, 16 input columns and 1 column for binary outcome (**Table 1**).

FOLD-R++ processed this dataset and achieved 90.2% accuracy with the help of ten rules, out of which four were base rules and six were exception rules (**Table 2**), Any of the base rules, if satisfied, generated a positive outcome (yes). If all the base rules were evaluated as false, then the predicted outcome was negative. The exception rules were referenced in the base rules to represent complex logical conditions.

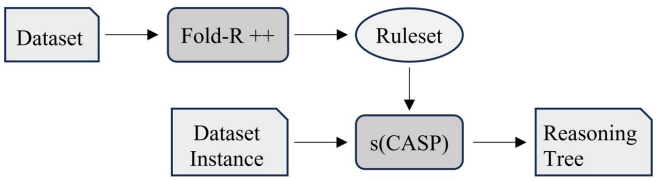

Figure 1: End-to-end workflow for generating explainable models with FOLD-R++ for binary classification problems. To generate explainable models with FOLD-R++, a dataset from the public domain was first processed in the Fold-R++ program, which generated a set of rules that can be applied to any test data instance for the interpretability of predicted results. This ruleset was then entered as an input to the s(CASP) utility along with a test record from the dataset, which generated the reasoning tree showing the logic applied to predict the outcome.

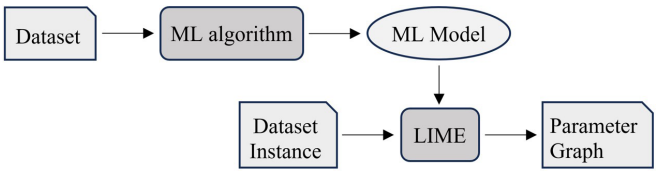

Figure 2: End-to-end workflow for generating graphical interpretation of predictions with LIME for binary classification problems. To generate interpretable predictions with LIME, a dataset from the public domain was first processed in a standard ML algorithm, which generated a model that can be applied to any test data instance for the interpretability of predicted results. This model was then entered as an input to LIME along with a test record from the dataset, which generated a graphical representation of the dominant parameters contributing to the prediction of the outcome.

Rules in both categories had a combination of conditional clauses, and they can be easily transformed into readable text; for example, the first base rule can be stated as the prediction will be "yes" if the contact is cellular, the day is less than or equal to 16.0, and the duration is greater than 827.0 (**Table 2**). The rules were applied to the test dataset, and the accuracy in predicting test dataset instances was calculated as the ratio of correct predictions to total predictions. The time to process each dataset by the FOLD-R++ program was also noted for comparison purposes of computing resources used in the calculation process.

Using the ten rules from FOLD-R++ and a test record instance as input, s(CASP) translated the rules to conditional statements in English for better understanding of the logic and showed how the predicted outcome was arrived at with the help of specific rules (**Figure 3a**). In this case, the following three conditions were all true satisfying the first base rule and hence the predicted outcome was yes. In other words, contact value was cellular (base rule: contact equals 'cellular), day value was 4, which was less than 16 (base rule N8 <= 16.0) and duration value was 1044, which was more than 827 (base rule: N10 > 827.0). Justification tree confirmed this logic by highlighting these conditions that were true for this test record.

s(CASP) program output demonstrates how the rules were applied to an instance of test data with a positive outcome (**Figure 3a**). Part A has the input data elements for the test data instance: a. The next portion, Part B lists the rules generated by the FOLD-R++ program, which has the four base rules on top, followed by six exception rules below them. Finally, Part C shows how a Prolog-style query is added to determine the truth or validity of the claim that the test data instance (a) will generate a positive outcome (yes). The reasoning output for the query is shown in the raw format first, where we can see that the result is yes because one of the base conditions is fully satisfied, namely, value for contact input parameter is cellular, day (4) is less than 16.0, and duration (1044) is greater than 827.0. This logical justification is also printed as a tree in human-readable form by s(CASP) (**Figure 3a**). s(CASP) program takes the test data instance record and the rules from FOLD-R++ as input. The logical conditions satisfied with the input parameters and the justification tree were thus easy to comprehend. Clearly, they showed the rationale behind the positive outcome of the test dataset instance.

Similarly, the s(CASP) program demonstrated how the

https://doi.org/10.59720/23-341

Number (IN/OUT)	Parameter Name - Description	Parameter Type	Sample values
1 (IN)	age - in years	numeric	28, 53
2 (IN)	job - category	categorical	'admin.', 'unemployed', 'unknown'
3 (IN)	marital - status	categorical	'single', 'married', 'unknown'
4 (IN)	education - category	categorical	'primary', 'secondary', 'unknown'
5 (IN)	default – credit in default?	binary	yes, no
6 (IN)	balance – average yearly balance	numeric	121, 593
7 (IN)	housing – has housing loan?	binary	yes, no
8 (IN)	loan – has personal loan?	binary	yes, no
9 (IN)	contact – last contact type	categorical	'telephone', 'cellular', 'unknown'
10 (IN)	day – last contact	numeric	5, 6, 7
11 (IN)	month – last contact	categorical	'jan', 'may', 'dec'
12 (IN)	duration – last contact in seconds	numeric	63, 122
13 (IN)	campaign – number of contacts at present	numeric	2, 6
14 (IN)	pdays – number of days from last contact	numeric	-1 (not previously contacted), 5
15 (IN)	previous – number of earlier contacts	numeric	0, 3
16 (IN)	poutcome – outcome of previous campaign	binary	yes, no
1 (OUT)	y – did the client buy term deposit?	binary	yes, no

Table 1: Input and output parameters for bank marketing dataset. This Kaggle dataset contains the results of a direct phone call marketing campaign of a Portuguese bank to promote term deposits in existing customers between May 2008 and November 2010. The input parameters have a mix of numeric, definite, and binary values with no duplicate rows or null values in the dataset. One binary output parameter (yes/no) indicated whether a client purchased a term deposit at the end.

rules were applied to an instance of test data to generate a negative outcome (**Figure 3b**). Part A has the input data elements for the test data instance, b. The next portion, Part B partially lists the rules generated by the FOLD-R++ program, which has the same four base rules and the six exception rules. The Prolog-style query is then shown in Part C to determine the truth or validity of the claim that the test data instance (b) generates a negative outcome (no). The reasoning output for the query is shown in the raw format first, where we can see that the outcome is yes because none of the base conditions are met. The justification tree showing the rules is truncated here for space limitations, but the first few rules are listed, namely, contact (unknown) is not cellular, outcome (unknown) is not success, and month (may) is not apr. Similar to the process for the positive outcome example shown earlier, s(CASP) program takes the test data instance record b here and the same set of rules from FOLD-R++ as input.

After the FOLD-R++ results were obtained, the same dataset was processed with a standard ML algorithm. In this case, we tested the dataset with KNN. The KNN algorithm also gave us similar accuracy (90.2%) for the predicted outcome. But the processing time was 32.8 secs as opposed to just 1.3 seconds for FOLD-R++. This demonstrated how FOLD-R++ used much less compute resources than KNN to provide similar accuracy in predictions for this dataset.

The model information generated by KNN was then used as input to LIME along with a test record from the dataset to get an insight into the reasoning behind the predicted outcome with graphical representation. In this case, we saw that LIME was able to identify the key input parameters, previous (0.14), default (0.11), and marital (0.07), that contributed to the positive outcome for the test data instance (**Figure 4a**). The sum of the weights for these parameters on the blue side were greater than that on the orange side to tilt the final prediction to the blue side (positive). The same LIME explainer was again used to generate the graphical representation of a test data instance with a negative result (**Figure 4b**). Here, we saw that the dominant parameters were duration in seconds (0.24), previous (0.14), pdays (0.08), loan (0.06), and contact (0.06).

This template process was repeated for all the binary classification datasets selected for our analysis. A dataset was downloaded and processed in FOLD-R++. Next, the same dataset was processed with well-known ML algorithms in Jupyter Notebook to develop a fitted model. The accuracy and the processing times from FOLD-R++ and ML algorithms were noted for comparison of the performance. Details of observations from each individual dataset were tabulated including parameters, like number of records in the dataset, ML algorithms used, accuracy score, or time taken for processing (**Table 3**).

After processing ten datasets of varying shapes and sizes, the results demonstrated a common trend and validated the initial hypothesis that FOLD-R++ will not only be comparably accurate with predictions of binary classification problems as some of the well-known ML algorithms but also provide meaningful reasoning interpretation behind the predicted outcomes (**Figure 5a**). Also, the results confirmed that FOLD-R++ generated comparable accuracy as standard ML algorithms within a lower processing time in some of the tests (**Figure 5b**). In addition, LIME added its value by delivering graphical illustrations for models generated by well-known

Base Rules – any one of these rules needs to be satisfied for a positive outcome X – a sample record in the dataset that is being evaluated	
y(X,'yes')	contact(X,'cellular'), day(X,N8), N8=<16.0, duration(X,N10), N10>827.0.
	poutcome(X,'success'), duration(X,N10), N10>605.0, not ab1(X), not ab2(X).
	not month(X,'apr'), not month(X,'oct'), not job(X,'admin.'), not campaign(X,'9'), not campaign(X,'3'), not job(X,'housemaid'), not job(X,'unknown'), not marital(X,'married'), day(X,N8), N8>17.0, N8=<30.0, duration(X,N10), N10>937.0, pdays(X,N12), N12=<122.0.
	poutcome(X,'success'), duration(X,N10), N10>225.0, not ab3(X), not ab5(X), not ab6(X).
Exception Rules – these are referred in various rules to represent complex conditions	
ab1(X)	housing(X,'yes'), not job(X,'management'), pdays(X,N12), N12=<119.0.
ab2(X)	job(X,'technician'), contact(X,'cellular'), age(X,N0), N0>51.0.
ab3(X)	duration(X,N10), N10>605.0.
ab4(X)	age(X,N0), N0=<28.0, duration(X,N10), N10=<236.0.
ab5(X)	month(X,'may'), duration(X,N10), N10=<281.0, pdays(X,N12), N12>92.0, not ab4(X).
ab6(X)	housing(X,'yes'), age(X,N0), N0>48.0, pdays(X,N12), N12>371.0.

Table 2: Prediction rules generated by FOLD-R++ for the bank marketing dataset. The rules were generated by running the FOLD-R++ against the different datasets identified for our exercise. The rules in each case were divided into two categories: base and exception rules. Four base rules provided the primary condition. A test data instance, fully satisfying any of the base rules, could generate a positive outcome. The base rules may or may not contain references to the six exception rules generated by the algorithm. As an example, the first base rule above can be stated as the outcome will be yes if the following conditions are true, namely, contact is cellular, day is less than or equal to 16.0, and duration is greater than 827.0.

https://doi.org/10.59720/23-341

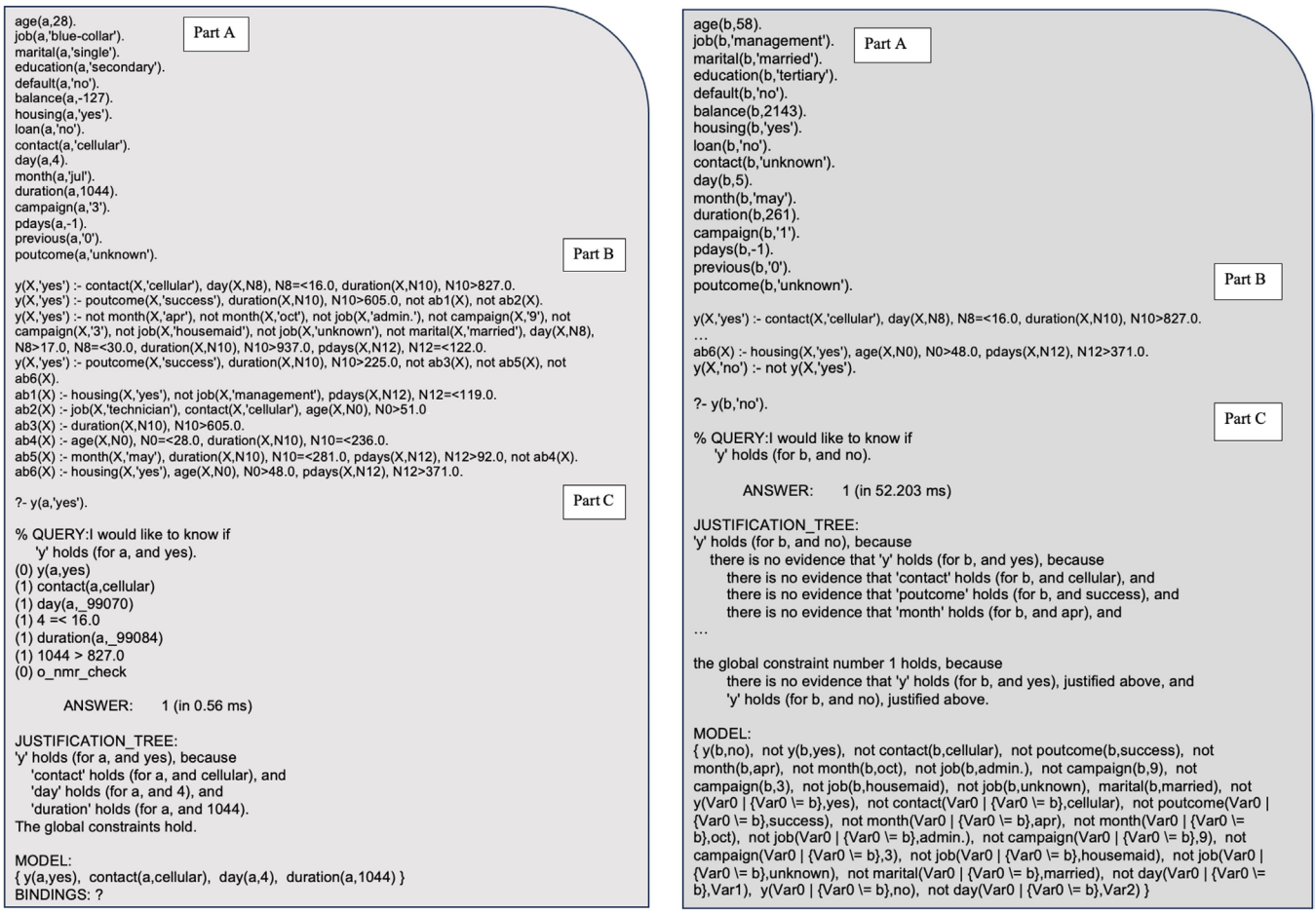

Figure 3: Input data file for s(CASP) and reasoning output with justification tree for a test data instance with a favorable and adverse outcome. A) Part A consists of the values for each parameter in the test data instance. All the base rules and exception rules from FOLD-R++ are shown in **Part B; Part C** shows a Prolog-style query entered as input to validate if the data record instance (a) is positive. We see that all the conditions in the first base rule from FOLD-R++, involving values from the parameters, contact ('cellular'), day (4), and duration (1044), were satisfied. Thus, the reasoning tree generated a positive outcome for the data record. **B)** Results for a negative outcome with similar input set in **Parts A** and **B**. Additionally, we added another rule in **Part B** stating that the negative outcome (no) is unfavorable. **Part C** shows the Prolog-style query entered to validate if the data record instance (b) is negative. The justification tree is long and, therefore, truncated for brevity. It was observed in the s(CASP) output that all four base rules from FOLD-R++ were proved false, thus resulting in a negative outcome.

ML algorithms, highlighting the dominant parameters that contribute to the reasoning behind an outcome prediction.

DISCUSSION

Besides, FOLD-R++ had the critical advantage of providing rules, which could then be used to generate easily understandable reasoning trees to interpret the logic behind the predicted outcome from any test data instance. LIME also provided significant insight with the explainable graphical representation for the expected outcome of a test data instance and highlighted which input parameters were substantial.

When we compared the accuracy of FOLD-R++ and ML algorithms, we only did a little data analysis and data engineering on the datasets for FOLD-R++. For example, we did not carry out any pre-screening like elimination of outliers for FOLD-R++ to focus on a consistent dataset for analysis. However, some of the ML algorithm implementation had the data cleanup steps included in the processing. Therefore, such data cleansing operations could result in different accuracy for the FOLD-R++ algorithm. However, for uniformity on the training dataset volume, we consistently split the training and test datasets in the ratio of 80:20, but there may be a better split across all ten datasets tested. An iterative approach to determine the optimum split ratio for each dataset may have yielded better accuracy from FOLD-R++ and the other ML algorithms. The same methodology could have been followed to find the optimal hyper-tuning parameters for each algorithm to yield better accuracy.

The possibilities explored and demonstrated here on explainable features have many opportunities worth investigating. FOLD-R++ was only used with s(CASP) here but can also be used with LIME. The FOLD-R++ and LIME combination can add another dimension to the explainable information available for any classification problem by providing the logical reasoning tree from FOLD-R++ along with the graphical representation of dominant parameters showing their relative contributions from LIME. Moreover, we only concentrated on binary classification problems, but FOLD-R++ algorithms can also be applied to multi-category

https://doi.org/10.59720/23-341

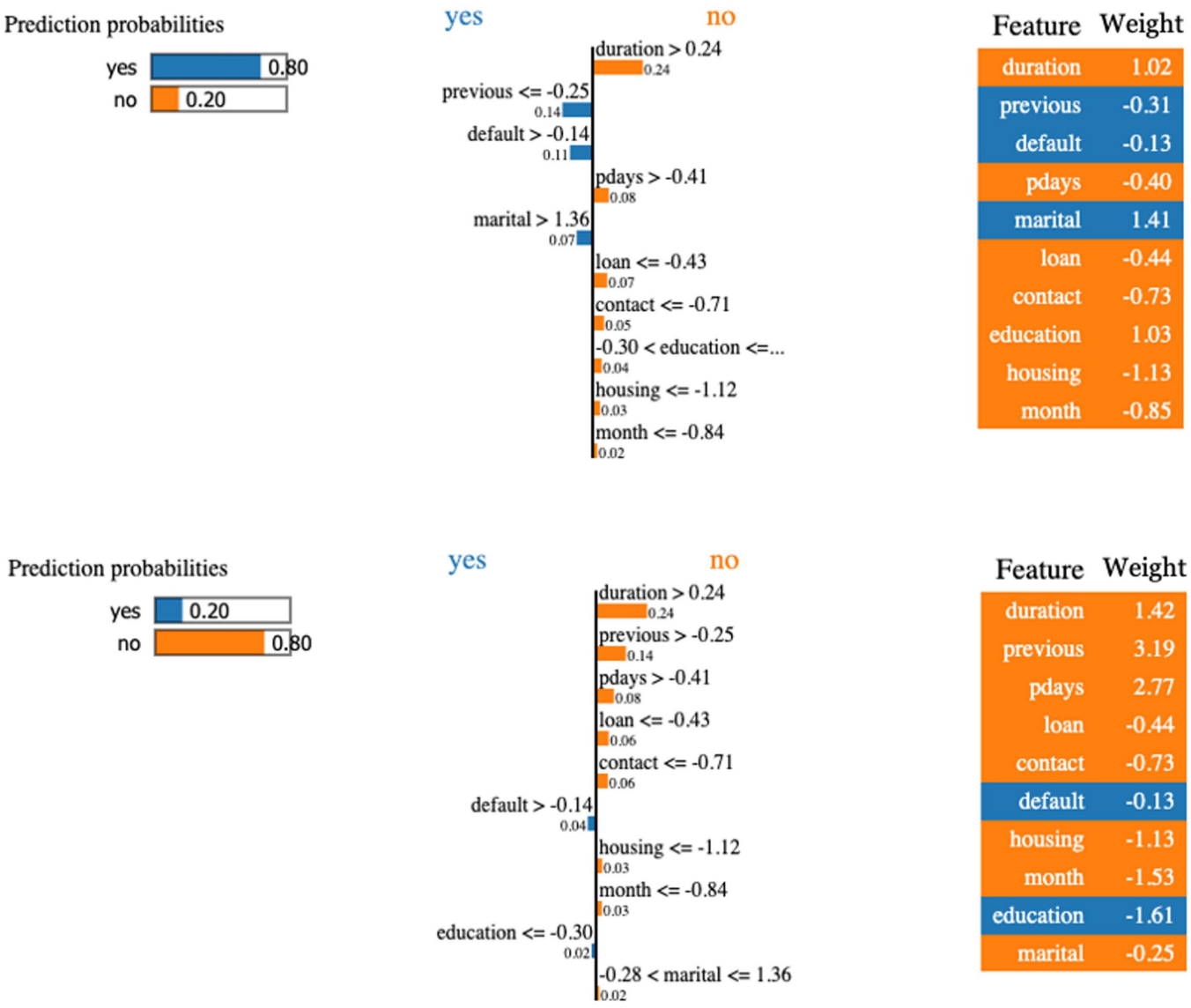

Figure 4: Positive and negative outcome predictions highlighting dominant parameters. A) In this test record, the positive outcome (yes) is predicted because the sum of the weights for all the dominant parameters satisfying the rules for yes (blue) were more than the corresponding sum of weights (orange) for the negative outcome (no). The weights for the dominant parameters were previous (0.14), default (0.11), and marital (0.07), all shown on the blue side of the chart. **B)** The negative outcome, for this instance, from the test dataset was attributed to the sum of the weights for dominant parameters, duration (0.24), previous (0.14), days (0.08), loan (0.06), and contact (0.06), all shown on the orange side of the chart.

classification problems. We could then have explainable capability extended to a broader range of classification problems.

A clear understanding of the justification generated by a prediction model allows us to investigate the guidance rules that come into play behind any prediction outcome. Thus, detecting anomalies or biases in the reasoning rule set becomes more accessible and helps us tweak the models to make them more uniform and non-biased. For example, if FOLD-R++ generates a base rule with gender parameter in a condition, then it is quite likely that the reasoning logic will have a gender bias. So, we can fine tune the model by removing the gender condition. Therefore, the rules from FOLD-R++ can reveal a lot of information related to the actual input parameter and their range of values used in a dataset. This possibility itself could lead to various exciting studies to examine the justification rules, thereby providing the option to detect and control any bias in the datasets.

We found out in the bank marketing dataset that an outcome in FOLD-R++ was positive if the type, day, and duration of last contact satisfied few rules. Based on this valuable interpretation of predictions from FOLD-R++, the business can focus on these parameters to ensure the rules are satisfied to maximize the positive responses. On the other hand, the parameters which were not used in the rulesets from FOLD-R++ could be ignored and instead attention could be given to the parameters that matter most, thus optimizing the company resources to efficiently increase the percentage of positive outcomes.

Exploring the explainable features currently available in

https://doi.org/10.59720/23-341

No.	Dataset	No. of samples	No. of input vars	Machine Learning			FOLD-R++			
				Algorithm	Accuracy	Proc Time	Accuracy (% improvement)	Proc Time	Base Rules	Exception Rules
1	Gender Classification	5001	7	Logistic Regression	0.967	1.2	0.982 (1.6%)	0.1	5	10
2	Stroke	5110	10	KNN	0.967	0.1	0.972 (0.5%)	0.2	3	1
3	Telecom Churn	6589	41	Random Forest	0.863	0.7	0.870 (0.8%)	1.3	5	9
4	Mushrooms	8124	23	Random Forest	1.0	3.2	1.0 (no change)	0.2	4	3
5	Machine Maintenance	10000	6	Random Forest	0.96	28.5	0.979 (1.9%)	0.4	10	15
6	Phishing	11054	30	KNN	0.956	1.7	0.956 (no change)	0.5	6	21
7	Income > 50K	43957	9	Decision Tree	0.890	7.16	0.849 (-6%)	1.6	2	11
8	Bank Marketing	45211	16	KNN	0.902	32.8	0.902 (no change)	1.3	4	6
9	Smoking	55692	26	Logistic Regression	0.751	0.3	0.724 (-3.6%)	6.2	3	3
10	Heart Disease	319795	15	XGB Classifier	0.913	14.4	0.916 (0.3%)	16.1	5	8

Table 3: Summary of comparison results between FOLD-R++ and other ML algorithms. Ten datasets from different domains were analyzed. At first, the datasets were processed for prediction of binary outcome via well-known ML algorithms, where the accuracy and processing times were noted. Next, FOLD-R++ algorithm was applied to the same dataset and prediction accuracy along with processing times were noted down for comparison. The Accuracy column with percentage improvement numbers shows that FOLD-R++ consistently generated accuracy comparable to those produced by other ML algorithms. However, the added advantage for FOLD-R++ was that the processing times in some cases were much lower, and most importantly, it provided explainable reasoning for predictions with s(CASP) utility. KNN = K-nearest neighbor, XGB = Extreme Gradient Boosting. A brief description of all the datasets used in this table is found in Appendix.

XAI tools has demonstrated the convenience of investigating various machine learning problems and gaining meaningful insight into their reasoning and rationale. We have observed that open-source XAI tools like FOLD-R++, s(CASP), and LIME can achieve comparable accuracy as well-known ML algorithms by using fewer computing resources. Moreover, they can also provide valuable insight by providing interpretability of the predictions. We can analyze the human-readable justification logic applied behind each prediction.

Some ML algorithms like Linear Regression models provide some interpretable information in terms of the gradient and intercept of the best fit line. However, if we consider the set of ML algorithms that are used, the predictions are not interpretable to the real input parameters. For instance, we saw in the KNN example with LIME that some parameters were dominant behind a predicted outcome but the values that were used as weights in evaluating the corresponding rules were all abstract and had no relevance to their actual values in the dataset. However, this was not the case with FOLD-R++, where the justification tree used actual values of the input parameters to deduce the logical outcome.

As research in academia and industry progresses, the field of XAI has potential to gain significant momentum. It is emerging as a popular area in the field of machine learning. Other XAI tools are available today, like SHAP (SHapley Additive exPlanation) and CAM (Class Activation Maps), which can be compared with FOLD-R++ (9-11). This would give us a chance to get a better understanding of how the different XAI tools perform with different types of datasets and perhaps provide insights into situations when a certain XAI tool may be a preferable choice. A comparative study between FOLD-R++ and other Deep Learning (DL) models like CNN (Convolutional Neural Network) and MLP (Multi-Layer Perceptrons) can also prove meaningful because these ML algorithms have not been compared with FOLD-R++ in this study.

The observed results were encouraging and validated our hypothesis. The primary XAI tools explored in this exercise, namely, FOLD-R++ and s(CASP), were very effective in providing meaningful insight into the reasoning behind the prediction of binary classification problems using significantly fewer computing resources. Along with this, LIME was also found to be effective in generating graphical representations of ML models that provided meaningful insights into the dominant parameters contributing to a prediction outcome of a test record in the dataset.

MATERIALS AND METHODS

The FOLD-R++ algorithm was developed at the University of Texas at Dallas in the Computer Science Department and an open-source Python implementation of the algorithm is available on GitHub (12). The FOLD-R++ algorithm is based on an automated inductive learning algorithm. It can process datasets with numerical, binary, and categorical data elements. It generates a set of base and exception rules as its explainable model from a dataset. Base rules are a set of rules where each rule consists of a combination of conditions using input parameters. A positive outcome is generated if any one of the base rules is evaluated to be true and the outcome is negative if all the base rules are evaluated to be false. The exception rules are also combinations of conditions that are referenced in other base or exceptions rules and used to capture complex logic.

The set of generated rules from FOLD-R++ was then fed to another XAI tool called s(CASP) (13). s(CASP) is an interpreter utility program based on Prolog (Logic Programming Language) and Ciao (multiparadigm logic programming system based in Prolog) developed at Madrid Institute for Advanced Studies (IMDEA) Software Institute, Madrid Polytechnic University, and the University of Texas at Dallas. The utility is a goal-directed answer set programming execution engine that can read the set of rules generated

https://doi.org/10.59720/23-341

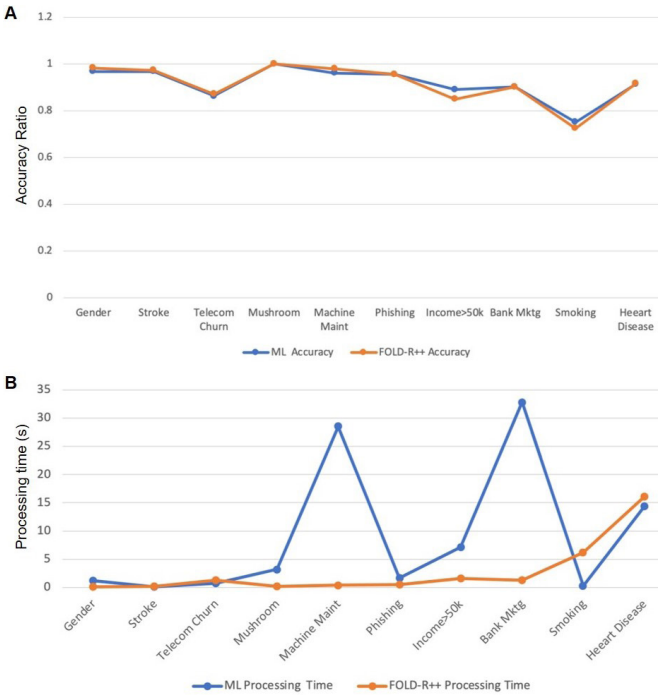

Figure 5: Accuracy Ratios and Processing Times in seconds for ML algorithms and FOLD-R++. A) The accuracy obtained by FOLD-R++ is close to that obtained by well-known ML algorithms for all the datasets. **B)** The processing times for FOLD-R++ compare with the ones obtained by well-known ML algorithms for all the datasets. For some datasets like Machine Maintenance and Bank Marketing, we see FOLD-R++ is much faster than the other ML algorithms.

by the FOLD-R++ program and produce human-readable reasoning trees for the predicted outcome of any binary classification dataset.

For this study, we downloaded the FOLD-R++ program (release 05242022) from the open-source software repository on GitHub. We downloaded binary classification datasets from Kaggle that offered a mix of different domains and dataset sizes. The FOLD-R++ program had to be set up with metadata for each dataset. No data cleanup operations were carried out to remove null values, eliminate outliers or any other operations to obtain a cohesive, consistent dataset. Instead, the raw datasets were simply processed with declaration of categorical and numeric data elements as metadata. Each dataset was separated into training and test datasets using an 80-20 split for uniformity. The FOLD-R++ program was run against the training dataset to generate a set of rules. The free IDE tool, PyCharm CE Community Edition 2022.3.2, was used to run the FOLD-R++ program. It is a Python program that uses a few basic libraries imported for datetime and dataset management but does not need any extra libraries for its core functionality.

s(CASP) interpreter utility (version 0.23.05.11) was then set up locally on the MacBook by following installation instructions from GitHub (14). As a prerequisite, the ciao engine was installed before the s(CASP) installation (15). After the FOLD-R++ program generated the explainable rule model, an input file was created to feed the s(CASP) utility, including the entire ruleset consisting of default and exception rules. In addition, a candidate test instance record was identified from within the dataset that would be tested for the predicted outcome and was also included in the input file, s(CASP) was invoked from the command-line prompt in a terminal window, and it took the input file with rules and instance data as the input argument. The output, by default, is produced in the terminal window, but the output can also be redirected to an HTML file. This feature is handy for viewing large reasoning trees in a browser with collapsible and expandable sections. After execution, the s(CASP) utility generated the human-readable reasoning tree that resulted in the predicted outcome for the test instance record.

Visual Studio Code (VS-Code version V1.81.1) was used to set up the Jupyter Notebook, where the CSV files for the same datasets were read to compare the results with known ML algorithms. Standard libraries like numpy, pandas, seaborn, matplotlib, scipy, and learn were imported and used for data engineering and analysis. The ipynb notebook files were set up in VS-Code to read and process the datasets from CSV files. Some standard data analysis and cleanup were done before applying the ML algorithms, like removal of records with null values, outliers for certain parameters to get a consistent dataset for model generation. On the other hand, the raw dataset in its entirety was used with FOLD-R++. After the ML algorithms were tested to be functional, the LIME plugin-specific portions were added to the notebook to build the explainer.

The exact split of 80-20 between training and test data was configured for uniformity with the FOLD-R++ processing algorithm. A variety of popular algorithms, including Logistic Regression (LR), K-Nearest Neighbor (KNN), Random Forest (RF), Decision Tree (DT), and X-Gradient Boost (XGB), were used for prediction with different datasets. All these algorithms were implemented in Jupyter notebooks and run in Visual Studio Code. The accuracy and processing times were noted for each dataset. The LIME plugin library was installed to read the ML model and generate an explainable graphical representation (16).

For processing the well-known ML algorithms, the Jupyter Notebook was set up to read the bank marketing dataset. Standard data analysis and data engineering were applied to prepare the dataset for feeding to the ML model. These included but were not limited to checking for null values, removal of duplicate rows, and outliers. Details of these tasks can be found in the implementation details provided with the corresponding datasets in Kaggle (4). For the bank marketing dataset, we chose the KNN model and generated a model fitted for the bank marketing dataset in the Jupyter Notebook. The model was then used as input for the LIME utility (version 0.2.0.1) to build an explainer framework in LIME in the same notebook. Finally, we used this explainer to generate a graphical representation of dominant parameters contributing to the positive and negative outcome for a test data instance (**Figure 5a, b**).

ACKNOWLEDGMENTS

The first author would like to acknowledge the summer internship opportunity provided by the HS Outreach Program and the Explainable AI Lab of the Computer Science Department at the University of Texas at Dallas.

Received: December 13, 2023
Accepted: April 30, 2024
Published: September 9, 2025

https://doi.org/10.59720/23-341

REFERENCES

1. "Explainable Artificial Intelligence (XAI)" DARPA, www.darpa.mil/research/programs/explainable-artificial-intelligence. Accessed 6 Dec. 2023.
2. "What is explainable AI?" IBM, www.ibm.com/think/topics/explainable-ai. Accessed 6 Dec. 2023.
3. "Explainable AI" Google Cloud, https://cloud.google.com/vertex-ai/docs/explainable-ai/overview. Accessed 6 Dec. 2023.
4. Wang, Huadoa and Gupta, Gopal. "FOLD-R++: A Scalable Toolset for Automated Inductive Learning of Default Theories from Mixed Data." arXiv, 2021, https://doi.org/10.48550/arXiv.2110.07843.
5. Rothman, Denis. "Hands-On Explainable AI (XAI) with Python" July 2020, Chapter "Getting started with LIME." Packt Subscription. Accessed 6 Dec. 2023.
6. "Binary Classification Datasets" Kaggle, www.kaggle.com/datasets?tags=14201-Binary+Classification. Accessed 6 Dec. 2023.
7. Aries, Joaquin, et al. "Constraint Answer Set Programming without Grounding." arXiv, 2018, https://doi.org10.48550/arXiv.1804.11162.
8. "Bank Marketing Data Set" UC Irvine Machine Learning Repository. Accessed 6 Dec. 2023.
9. Angelov, P. P., et al. (2021). "Explainable artificial intelligence: an analytical review." *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, (2021) 11(5), e1424. https://doi.org/10.1002/widm.1424.
10. Lundberg, S. M., & Lee, S.-I. "A unified approach to interpreting model predictions. Advances in Neural Information Processing Systems" (2017), 30, 4765–4774, https://proceedings.neurips.cc/paper/2017/hash/8a20a8621978632d76c43dfd28b67767-Abstract.html.
11. Zhou, B., Khosla, A., et al. (2016). "Learning deep features for discriminative localization." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.
12. Wang, Huadoa. "FOLD-R-PP," GitHub Repository hwd404/FOLD-R-PP. Accessed 6 Dec. 2023.
13. s(CASP): Goal directed Constraint Answer Set Programming, https://swish.swi-prolog.org/example/scasp.swinb.
14. Arias, Joaquin. "sCASP" GitLab Repository, Project ID: 61. Accessed 6 Dec. 2023.
15. Ciao Project. "Ciao Install" GitHub Public Repository ciao-lang/ciao. Accessed 6 Dec. 2023.
16. Ribeiro, Marco Tulio Correira. "lime" GitHub Repository marcotcr/lime. Accessed 6 Dec. 2023.

APPENDIX

Fold-R++ code snippet

Bank marketing dataset setup in Python for Fold-R++

```
def bank():
    attrs = ['age','job','marital','education','default','housing',
             'loan','contact','day','month','duration','campaign',
             'pdays','previous','poutcome']
    nums = ['age','day','duration','pdays']
    model = Classifier(attrs=attrs, numeric=nums, label='y', pos='yes')
    data = model.load_data('data/bank/bank-full.csv')
    print('\n% bank dataset', len(data), len(data[0]))
    return model, data
```

Fold-R++ main routine in Python

```
from foldrpp import *
from datasets import *
from timeit import default_timer as timer
from datetime import timedelta

def main():
    model, data = bank()

    data_train, data_test = split_data(data, ratio=0.8, rand=True)

    X_train, Y_train = split_xy(data_train)  # split data into features and
label
    X_test,  Y_test = split_xy(data_test)

    start = timer()
    model.fit(X_train, Y_train, ratio=0.5)
    end = timer()

    model.print_asp(simple=True)
    Y_test_hat = model.predict(X_test)
    acc, p, r, f1 = get_scores(Y_test_hat, Y_test)
    print('% acc', round(acc, 4), 'p', round(p, 4), 'r', round(r, 4), 'f1',
round(f1, 4))
    print('% foldr++ costs: ', timedelta(seconds=end - start), '\n')

if __name__ == '__main__':
    main()
```

LIME setup in Jupyter notebook

Lime explainer initialization

```
# Import the LimeTabularExplainer module
from lime.lime_tabular import LimeTabularExplainer

# Define outcome class names
class_names = ['yes', 'no']

# Define the feature names
feature_names = list(x_train.columns)
```

Lime explainer used to generate graphs for test dataset instances

```
# Explainer setup for training data set using explainer
explainer = LimeTabularExplainer(x_train.values, feature_names = feature_names,
                    class_names = class_names, mode = 'classification')

#Perform the explanation on the 30th instance in the test data – positive outcome
explanation = explainer.explain_instance(x_test.iloc[30], knn_model.predict_proba)

# show the result of the model's explanation
explanation.show_in_notebook(show_table = True, show_all = False)

#Perform the explanation on the 18th instance in the test data – negative outcome
explanation = explainer.explain_instance(x_test.iloc[18], knn_model.predict_proba)

# graphical representation for the result of model's explanation
explanation.show_in_notebook(show_table = True, show_all = False)
```

https://doi.org/10.59720/23-341

Summary of Kaggle datasets used in this experiment

Brief description of all the Kaggle datasets referenced in the analysis of observed results

1. Gender Classification – This is a dataset that predicts the gender of a person based on seven input parameters involving measurements of face aspects along with a Boolean value for long hair. The dataset has 8 input parameters and 5001 records.
2. Stroke - This dataset is used to predict whether a patient is likely to get a stroke. The dataset consists of 10 input parameters like gender, age, and various diseases and smoking status and has 5110 records. Value of 0 in stroke field means no risk and 1 means person is at risk.
3. Telecom Churn - This dataset contains captures information from customers of a telecom company. Parameters include their demographic profile, subscriptions for various services and information about the billing. As outcome, it records binary data for the churn status providing valuable information on probable causes for customer departure. The dataset has 41 input parameters and 6589 records.
4. Mushrooms – This dataset has prediction of whether a mushroom is poisonous, or edible based on certain physical characteristics, like shape, color, odor etc. The dataset has 23 input parameters and 8124 records.
5. Machine Maintenance - This dataset is used for prediction of machine failure risk in industrial manufacturing environments, using real-time sensor data from various types of machinery. The dataset has 6 input parameters, like temperature, vibration, humidity and has 10000 records.
6. Phishing – This dataset has information for over 11000 websites with their URLs. Each record in the dataset contains 30 website parameters and has a binary label to indicate if it is a phishing website or not (1, -1).
7. Income > 50k – This dataset predicts if a person's income is greater than 50k based on input parameters like occupation, education, marital status and others. It has 9 input parameters and over 43,000 records.
8. Bank Marketing - This dataset is obtained from a marketing campaign of a Portuguese bank tracking information from phone calls made to clients. There are 16 input parameters like job, marital status, education level etc. and the prediction of whether the client will open a term deposit in the bank. The dataset has over 45,000 records.
9. Smoking – This dataset predicts if a person is a smoker or not based on certain input parameters like age, height, tartar formation, cholesterol, blood sugar etc. It has 26 input parameters and over 55,000 records.
10. Heart Disease – This dataset predicts the risk of heart disease in a person based on different input parameters like age category, bmi, physical activity, diabetic etc. It has 15 input parameters and over 319,000 records.