Explaining Automated Data Cleaning with CLeanEX

Abstract

In this paper, we study the explainability of automated data cleaning pipelines and propose CLeanEX, a solution that can generate explanations for the pipelines automatically selected by an automated cleaning system, given it can provide its corresponding cleaning pipeline search space. We propose meaningful explanatory features that are used to describe the pipelines and generate predicate-based explanation rules. We compute quality indicators for these explanations and propose a multi-objective optimization algorithm to select the optimal set of explanations for user-defined objectives. Preliminary experiments show the need for multi-objective optimization for the generation of high-quality explanations that can be either intrinsic to the single selected cleaning pipeline or relative to the other pipelines that were not selected by the automated cleaning system. We also show that CLeanEX is a promising step towards generating automatically insightful explanations, while catering to the needs of the user alike.

1 Introduction

When it comes to real-world data, inaccurate, noisy, uncertain, or incomplete data are the norm rather than the exception. In many applications of machine learning (ML), the cost of an error can be high. However, explaining the results of an ML pipeline is as crucial as reducing the impact of dirty data input and estimating the uncertainty in the predictions of a model. Data scientists have to spend considerable time integrating data from multiple sources, manually curating and preparing the data with their expertise of the domain. They use various libraries and tools to correct erroneous values, impute missing ones, eliminate duplicate records or disambiguate conflicting data, to prevent unreliable data being delivered downstream to a machine-learning model. Although data cleaning in ML pipelines is still considered to be "intractable" or "AI-hard" –as it is difficult to fully automate and often requires human expertise– several solutions of automated data cleaning have been proposed recently [Shang *et al.*, 2019; Krishnan *et al.*, 2017; Krishnan *et al.*, 2015]. While important, however, we argue that a critical missing piece of this line of work be the explainability of automated cleaning methods, i.e., why/how a particular data value is detected as an anomaly, how it is fixed, and why it is fixed the way it is fixed.

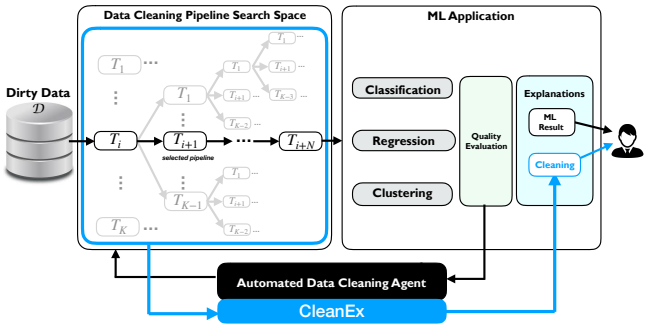


Figure 1: CLeanEx Overview

Current automated cleaning methods hardly ever explain their choice in building automatically data cleaning pipelines. However, the lack of convincing explanations may severely limit the scope of an end-to-end ML pipeline (which includes data cleaning and data preparation) and the broader adoption of ML-based Artificial Intelligence for critical decision-making. In this paper, therefore, we study the explainability of automated data cleaning pipelines and propose CLeanEX, a solution that can generate explanations for a pipeline generated by an automated data cleaning system that can provide its pipeline search space. CLeanEX extends our automated cleaning system XXXX[1] proposed in [Anonymous].

The paper proceeds as follows. In Section 2, we discuss related work. Section 3 presents our definitions and describe how to generate explanations from a sequence of cleaning tasks. Experiments are reported in Section 4. Section 5 concludes the paper.

2 Related Work

As illustrated in Fig. 1, data cleaning and preparation require a sophisticated sequence of tasks T_i for the detection and elimination of a variety of intricate data quality prob-

[1] https://github.com/XXXX

lems (e.g., duplicates, inconsistent, missing, and outlying values). Generally, the users may not know which preprocessing methods can be applied to optimize the final results downstream. The selection of the optimal sequence of tasks would require for him/her executing all possible methods for each potentially relevant preprocessing task, as well as all the possible combinations of the methods with different orderings and configurations (pipelines in grey in the figure). Successfully automating this daunting manual process would definitively improve data scientists' every day life. Related to our work, we review recent advances for scalable and automated data cleaning and explainable systems.

ML-based data and pipeline curation. A recent line of work is to use machine learning to improve the efficiency and reliability of data cleaning and data repairing [Yakout *et al.*, 2013; Rekatsinas *et al.*, 2017; Krishnan *et al.*, 2017; Shang *et al.*, 2019]. [Yakout *et al.*, 2013] train ML models and evaluate the likelihood of recommended replacement values to fix erroneous and missing ones. [Rekatsinas *et al.*, 2017] propose a framework for data repairing based on probabilistic inference to handle integrity constraints or external data sources seamlessly, with quantitative and statistical data repairing methods. [Krishnan *et al.*, 2017] propose a method for data cleaning optimization for user-defined ML-based analytics. Their approach selects an ensemble of methods (statistical and logic rules) and boosting to combine error detection and repair. AutoML approaches can optimize the hyper-parameters of a considered ML model, but they support only a limited number of preprocessing steps with by-default methods. Recently, Alpine Meadow [Shang *et al.*, 2019] combines an AutoML approach and a cost model to select candidate logical ML pipeline plans (as in DB query optimization). Multi-armed bandits are used to select promising logical ML pipeline plans, and Bayesian Optimization is used to fine-tune the hyper-parameters of the selected models in the search space. However, these approaches do not provide explanations justifying how the data preparation pipeline search space is built and pruned. We argue that more efforts should be devoted to proposing a principled and efficient explainable automated data cleaning approach to help the user in understanding the sequence of data preparation tasks selected by the automated cleaning system.

Explainable systems for data quality and data profiling. The explanation of predictive models have been an overgrowing field in the last years [Guidotti *et al.*, 2018; Pedreschi *et al.*, 2019]. Despite the numerous works on black box models explanation, few of them address the problem of explaining the choices performed during data cleaning tasks [Bertossi and Geerts, 2020]. [Rammelaere and Geerts, 2018] proposes to explain repairs performed by the user over small data sets by extracting the most relevant conditional functional dependencies (CFDs) according to the repairs. However, while this approach can be adapted to explain the output of an automated data cleaning task, the CFDs can be hard to understand for the user and they have a limited expressiveness, which might be detrimental to explain the result of non rule-based repairing tasks. The T-REx framework [Deutch *et al.*, 2020] provides explanations taking the form of a ranking of the set of Denial Constraints (DCs) according to their influence during the repairing process. This approach has to provide the same set of DCs that is used as input of the data repairing task. In the context of Entity Resolution (ER) [Ebaid *et al.*, 2019], local explanations have been proposed to explain why a tuple pair was predicted to be a duplicate. In this context, other approaches include identifying a ranked list of features that contribute to the prediction as a duplicate (such as LIME [Ribeiro *et al.*, 2016]) or as a rule that holds in the vicinity of the tuple-pair with high precision (such as Anchor [Ribeiro *et al.*, 2018]). While the current approaches explain the effect of a single repairing task, we choose to address the problem of explaining the full sequence of cleaning tasks (the cleaning pipeline). As this sequence has been selected and generated by an automated data cleaning system, we argue that the explanations should also offer the perspective related to the other candidate pipelines that were not selected by the system using various indicators to characterize the quality of an explanation.

3 Our Definitions

In this work, we represent the pipeline exploration space of the automated cleaning agent A as a tree. Each cleaning task updates the input data set and the updated version is represented as a node in a tree structure: the cleaning tree. As illustrated in Fig. 1, the cleaning tree is composed of branches representing alternative cleaning pipelines among which the cleaning agent A may select the optimal one with respect to an ML goal and a quality performance metric. Each pipeline has features that can be used and exposed to explain the agent decision independently of the ML quality performance metric optimization. Explanation rules can be generated using such features. More formally, we define the cleaning tree as follows.

Definition 1 (Cleaning tree). *Let D be a data set, $\mathcal{T} = \{t_1, \ldots, t_n\}$ be the set of data preprocessing tasks that can be executed by the automated cleaning agent A using an ML model M that maximizes the model quality performance metric q. A* cleaning tree *for A, M, and q over D is an oriented tree $\mathcal{C}_{A,M,q} = (\mathcal{V}, v_r, \mathcal{E})$ such that:*

- *$\mathcal{V}$ is the set nodes formed by triples $v_p = (I_p, q(M, D_p), s_p)$ with:*
  - *D_p, a data set resulting from the application of a cleaning task t to the parent node data set or, for the root node, the input data set D;*
  - *$q(M, D_p)$, the value of the quality metric of the ML model M measured over the pre-processed data set D_p;*
  - *s_p a vector associating, for each explanatory feature $f \in \mathcal{F}$, its value $s_p[f]$ for the current node;*
- *v_r the root node of the cleaning tree $\mathcal{C}_{A,M,q}$ such that $v_r \in \mathcal{V}$ and $v_r = (D, q(M, D), s_r)$;*
- *$\mathcal{E}$ is the set of edges in $\mathcal{C}_{A,M,q}$, with each edge e from a node $v = (D, q(M, D), s)$ to one of its children $v' = (D', q(M, D'), s')$ being labelled with the task t in $\mathcal{T}$ used to clean and update D as D'. We define the bijection $\phi : \mathcal{E} \to \mathcal{V} \times \mathcal{V}$ which maps each edge $e \in \mathcal{E}$ to the pair of nodes (v, v') connected by e.*

Definition 2 (Cleaning pipeline). *Given a cleaning tree $\mathcal{C}_{A,M,q} = (\mathcal{V}, v_0, \mathcal{E})$ and a node $v_n \in \mathcal{V}$. A cleaning pipeline*

p_{v_n} over $\mathcal{C}_{A,M,q}$ is a path $\sigma_{\mathcal{E}} = \langle e_1, \ldots, e_n \rangle$ of edges in $\mathcal{E}$ such that there exists a sequence $\sigma_{\mathcal{V}} = \langle v_0, v_1, \ldots, v_{n-1}, v_n \rangle$ of nodes from $\mathcal{V}$ such that each edges $e_i \in \sigma_{\mathcal{E}}$ connects the node v_{i-1} to the node $v_i \in \mathcal{V}$: $\forall e_i \in \sigma_{\mathcal{E}}$, $\phi(e_i) = (v_{i-1}, v_i)$.

After the automated cleaning agent A selects the node with the best quality metric value, the path from the root to this selected node corresponds to the best pipeline $p^{opt(M,q)}$ that is the optimal sequence of tasks evaluated by A for the ML model M and quality metric q.

To ground our explanations, we use **explanatory features**: the values of the explanatory features changing from one node to the next in a cleaning pipeline will serve as key elements for generating explanations of an automatically selected cleaning strategy. For each pipeline, we compute the following explanatory features, denoted $s_p[f]$ in Definition 1:

- **`cost`**: The normalized cost of the pipeline;
- **`data_quality_improvement`**: The percentage of data quality problems solved by the pipeline: e.g., imputation will remove 100% of missing values;
- **`distortion`**: The distortion as defined in [Dasu and Loh, 2012] as the Mahalanobis distance between the original and cleaned version of the data set;
- **`satisfaction`**: The satisfaction of ML model requirements by the pipeline defined as a Boolean: e.g., for regression, satisfaction equals 1 if linearity, multivariate normality, no or little multicollinearity, no auto-correlation, and homoscedasticity constraints are satisfied by the cleaned data set;
- **`corr_ratio`**: The fraction of the number of pipelines sharing the same tasks over the sum of their respective ranks and the total number of explored pipelines; and
- **`non_corr_ratio`**: The fraction of the number of pipelines that do not share the same task over the sum of their respective ranks and the total number of explored pipelines.

Other features could be added but the ones we consider define a representative and relevant signature that can be used for generating meaningful explanations of a cleaning pipeline independently from the ML model and the model quality metric used for by the automated cleaning system.

Example 1. *We illustrate the notations with an example using the House Price data set from Kaggle[2] with 81 variables and 1.46k observations to predict the `SalePrice` attribute with regression and MSE as the quality metric. We ran Learn2Clean[3] [?] as the automated cleaning agent and reported the top-5 cleaning strategies presented in Table 1 generated by Learn2Clean for OLS regression. The best strategy has been selected based on minimal MSE. The MSE value, execution time (in milliseconds), and tasks of the explored cleaning pipelines are reported in the table. Data cleaning strategies include various tasks such as K-nearest neighbors (KNN) or MICE imputation, outlier detection/removal using InterQuartile Range (IQR), Z-score-based (ZSB), or LOF methods, approximate duplicate removal (AD), MinMax normalization (MM), and linearly correlated (LC) feature exclusion (see [Berti-Equille, 2019] for more details about the preprocessing methods). These strategies can be represented as the cleaning tree of Fig. 2(a) for which each node v_i has explanatory feature signature s_i presented in the table of Fig. 2(b). In the table, we observe `cost`$= 0$ for the root and `cost`$= 1$ for s_9 as the maximal cost for the leaf node of the pipeline $IQR \rightarrow MICE \rightarrow AD$. The pipeline $KNN \rightarrow ZSB$ selected by Learn2Clean is represented in green. `corr_ratio` of KNN in s_1 is $\frac{2}{5+1+2} = 0.25$ as it appears in 2 pipelines out of 5 with rank# 1 and 2 respectively (similarly for ZSB). `no_corr_ratio` of KNN is $\frac{3}{5+3+4+5} = 0.176$ as it does not appear in 3 pipelines ranked #3, 4, and 5 respectively.*

[2] `https://www.kaggle.com/datasets`

[3] `https://github.com/LaureBerti/Learn2Clean`

Table 1: Learn2Clean results on House Price data set for OLS regression over `SalePrice` attribute.

Rank#	Sequence of Actions for Data Preparation	MSE	Time (ms)
1	$KNN \rightarrow ZSB$	2.20E-25	45.1
2	$MM \rightarrow KNN \rightarrow ZSB$	1.44E-12	64.2
3	$IQR \rightarrow MICE \rightarrow AD$	0.231	154.9
4	$IQR \rightarrow MICE$	0.236	94.1
5	$IQR \rightarrow LC \rightarrow MICE$	0.372	141.5

3.1 Explaining via Feature Changes and Comparisons

Next, we define three types of explanation predicates: (1) *Change predicates* that characterize the evolution of the explanatory features, and (2) *Comparison predicates* that characterize the relative changes by comparing with other pipelines in the cleaning tree.

Change Predicates. Given a cleaning tree $\mathcal{C}_{A,M,q} = (\mathcal{V}, v_r, \mathcal{E})$, a set of features $\mathcal{F}$, two distinct nodes $v, v' \in \mathcal{V}$ and s, s' their respective vectors of features values. We define the following set of change predicates:

- $succ(v, v')$ iff there exists an oriented path from v to v';
- $increase(f, v, v')$ iff: $s[f] < s'[f] \wedge succ(v, v')$
- $decrease(f, v, v')$ iff: $s[f] > s'[f] \wedge succ(v, v')$
- $stable([f_0, \ldots, f_n], v, v')$ iff:

$$\forall f_i \in [f_0, \ldots, f_n], s[f_i] = s'[f_i] \wedge succ(v, v')$$

- $equiv(v, v')$ iff: $\forall f \in \mathcal{F}, s[f] = s'[f]$.

Comparison Predicates. We define the following set of comparison predicates to express that a value of feature f for node v is respectively: (i) *more, less than*, or *as* a% of the feature f values of the other nodes in the cleaning tree with a, a decimal constant in [0,1]; (ii) the best value, the worst value, or (iii) different from another node v' feature value within a certain distance d in [0,1]:

- $more([f_0, \ldots, f_n], v, a)$ iff:
  $\forall f_i \in [f_0, \ldots, f_n], a = \frac{|\{v' \in \mathcal{V} | s[f_i] > s'[f_i]\}|}{|\mathcal{V}|-1}$
- $less([f_0, \ldots, f_n], v, a)$ iff:
  $\forall f_i \in [f_0, \ldots, f_n], a = \frac{|\{v' \in \mathcal{V} | s[f_i] < s'[f_i]\}|}{|\mathcal{V}|-1}$
- $as([f_0, \ldots, f_n], v, a)$ iff:
  $\forall f_i \in [f_0, \ldots, f_n], a = \frac{|\{v' \in \mathcal{V} | s[f_i] = s'[f_i]\}|}{|\mathcal{V}|-1}$
- $most([f_0, \ldots, f_n], v)$ iff:
  $\forall f_i \in [f_0, \ldots, f_n], \forall v' \in \mathcal{V}$ s.t. $v \neq v', s[f_i] > s'[f_i]$

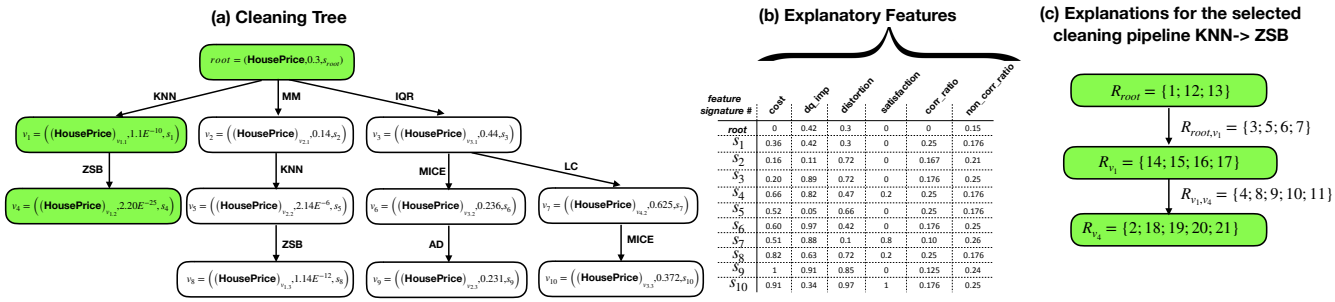


Figure 2: Example of a selected cleaning tree (in green) in the pipeline space of our illustrative example (a), with its corresponding explanatory features (b), and explanation tree (c) composed of the sets of rule IDs described in Table 2.

- $least([f_0, \ldots, f_n], v)$ iff:
  $\forall f_i \in [f_0, \ldots, f_n], \forall v' \in \mathcal{V}$ s.t. $v \neq v', s[f_i] < s'[f_i]$
- $\delta(f, v, v', a)$ iff: $|s'[f] - s[f]| = d$

Additionally to the axioms of the predicate logic, we define the following set of axioms for our explanation language:

- $\forall p_1, p_2 \in \{more, as, less\}$ s.t. $p_1 \neq p_2$:
  $p_1(f, v, 1) \equiv p_1(f, v, 1) \wedge p_2(f, v, 0)$
- $more(f, v, a) \wedge less(f, v, a')$
  $\equiv more(f, v, a) \wedge as(f, v, 1 - (a + a'))$
  $\equiv as(f, v, 1 - (a + a')) \wedge less(f, v, a')$
- $most([f_0, \ldots, f_n], v) \equiv \bigwedge_{i \in [0,n]} more(f_i, v, 1)$
- $least([f_0, \ldots, f_n], v) \equiv \bigwedge_{i \in [0,n]} less(f_i, v, 1)$
- $equiv(v_1, v_2) \Leftrightarrow \bigwedge_{f \in \mathcal{F}} stable(f, v_1, v_2)$

Now, we can define formally the explanation rules that will be automatically generated.

Definition 3 (Explanation Rule). *Given a cleaning tree $\mathcal{C}_{A,M,q} = (\mathcal{V}, v_r, \mathcal{E})$, a feature f in $\mathcal{F}$, v and v' two nodes in $\mathcal{V}$, three decimals constants, $a, a', a'' \in [0, 1]$, and the explanation predicates:*

$p_0 \in \{increase; stable; decrease\}$;
$p_1, p_2 \in \{more; as; less\}$ *such that* $p_1 \neq p_2$;
$p_3 \in \{most; least\}$.

We define the following types of explanation rules:

$$
\begin{aligned}
P_{v,v'} &: \quad \exists v_1, \ldots, v_n \in \mathcal{V}, succ(v, v_1) \wedge \ldots \wedge succ(v_n, v') \\
B_{f,v,v'} &: \quad p_0(f, v, v') \wedge \delta(f, v, v', a) \\
C_{f,v} &: \quad p_1(f, v, a') \wedge p_2(f, v, a'') \\
E_{f,v} &: \quad p_3(f, v) \\
S_{f,v,v'} &: \quad stable(f, v, v') \\
S_{v,v'} &: \quad equiv(v, v')
\end{aligned}
$$

Type P explanation rules are path explanations between two nodes v and v'. Type B explanation rules are behavior change explanations with a constant representing the δ change value of some feature f between two nodes v and v'. Type C explanations are comparative explanations with respect to the change of other nodes. Type E explanation rules are related to extreme value description. Type S explanation rules are stability explanation rules between v and v'.

Table 2: Explanation rules generated for pipeline#1 $KNN \rightarrow ZSB$

Rule#	Type	Rule
0	P	$succ(root, n1) \wedge succ(n1, n4)$
1	E	$least([\texttt{cost}, \texttt{corr_ratio}, \texttt{non_corr_ratio}], root)$
2	E	$most([\texttt{cost}, \texttt{dq_imp}, \texttt{distortion}, \texttt{satisfaction}], n4)$
3	S	$stable([\texttt{dq_imp}, \texttt{distortion}, \texttt{satisfaction}], root, n1)$
4	S	$stable([\texttt{corr_ratio}, \texttt{non_corr_ratio}], n1, n4)$
5	B	$increase(\texttt{cost}, root, v1) \wedge \delta(\texttt{cost}, root, v1, 0.36)$
6	B	$increase(\texttt{corr_ratio}, root, v1) \wedge \delta(\texttt{corr_ratio}, root, v1, 0.25)$
7	B	$increase(\texttt{non_corr_ratio}, root, v1)$ $\wedge\delta(\texttt{non_corr_ratio}, root, v1, 0.026)$
8	B	$increase(\texttt{cost}, v1, v4) \wedge \delta(\texttt{cost}, v1, v4, 0.3)$
9	B	$increase(\texttt{dq_imp}, v1, v4) \wedge \delta(\texttt{dq_imp}, v1, v4, 0.4)$
10	B	$increase(\texttt{distortion}, v1, v4) \wedge \delta(\texttt{distortion}, v1, v4, 0.17)$
11	B	$increase(\texttt{satisfaction}, v1, v4) \wedge \delta(\texttt{satisfaction}, v1, v4, 0.2)$
12	C	$as([\texttt{dq_imp}, \texttt{distortion}, \texttt{satisfaction}], root, 0.5)$
13		$less([\texttt{dq_imp}, \texttt{distortion}, \texttt{satisfaction}], root, 0.5)$
14	C	$less(\texttt{cost}, v1, 0.5)$
15	C	$more(\texttt{cost}, v1, 0.5)$
16	C	$as([\texttt{dq_imp}, \texttt{distortion}, \texttt{satisfaction}, \texttt{corr_ratio}, \texttt{non_corr_ratio}], v1, 0.5)$
17	C	$less([\texttt{dq_imp}, \texttt{distortion}, \texttt{satisfaction}, \texttt{corr_ratio}, \texttt{non_corr_ratio}], v1, 0.5)$
18	C	$as(\texttt{corr_ratio}, v4, 0.5)$
19	C	$more(\texttt{corr_ratio}, v4, 0.5)$
20	C	$as(\texttt{non_corr_ratio}, v4, 0.5)$
21	C	$more(\texttt{non_corr_ratio}, v4, 0.5)$

These explanation rules can be either focused on one feature f for the $S_{f,v,v'}$ explanation rules, or all the features covered by $\mathcal{C}_{A,M,q}$ for the $S_{v,v'}$ explanation rules. Table 2 gives the explanation rules automatically generated for the cleaning pipeline $KNN \rightarrow ZSB$. Finally, all the generated rules can be organized into an explanation tree mirroring the cleaning tree defined earlier, as follows.

Definition 4 (Explanation Tree). *Let $\mathcal{C}_{A,M,q} = (\mathcal{V}, v_r, \mathcal{E})$ be a cleaning tree and R be a set of explanation rules such that $\mathcal{C}_{A,M,q} \models R$. An* explanation tree *for R in $\mathcal{C}_{A,M,q}$ is an oriented tree $\mathcal{X}_{\mathcal{C}} = (\mathcal{V}, \mathcal{E})$ such that:*

- *$\mathcal{V}$ is the set nodes such that each node v_{R_i} is formed by the subset of rules $R_i \subseteq R$ such that each $r \in R_i$ is a rule of type C_{f,v_i} or E_{f,v_i};*
- *$\mathcal{E}$ is the set of edges in $\mathcal{X}_{\mathcal{C}}$, with each edge e from a node v_{R_i} to one of its children v_{R_j} being labelled with the set of rules $R_{i,j}$ such that each $r \in R_{i,j}$ is a rule of type B_{f,v_i,v_j}, S_{f,v_i,v_j} or S_{v_i,v_j}.*

The branch of the explanation tree corresponding to the selected pipeline of the example is provided in Fig. 2(c).

3.2 Quality Indicators of Explanations

Given an explanation tree $\mathcal{X}_{\mathcal{C}}$, its set of predicates, denoted $\mathcal{P}_{\mathcal{C}}$ including the set of predicates $\mathcal{P}_{ml}$ of

the form $most([f_0, \ldots, f_n], v)$ or $least([f_0, \ldots, f_n], v)$, $features(p)$, the function returning the set of features covered by a predicate p, and $delta(r)$ the function returning the value a of the delta predicate $\delta(f, v, v', a)$ in a rule r. We select the best explanations to provide to the user using the following relevant quality indicators defined as follows.

Definition 5 (Polarity). *The polarity of $\mathcal{X}_\mathcal{C}$ is defined as:*

$$polarity(\mathcal{X}_\mathcal{C}) = \frac{\sum_{\forall p \in \mathcal{P}_{ml}} |features(p)|}{\sum_{\forall p' \in \mathcal{P}_\mathcal{C}} |features(p')|}.$$

Definition 6 (Distancing). *Given the set $\mathbf{X}$ of explanation trees over $\mathcal{C}_{A,M,q}$ such that $\mathcal{X}_\mathcal{C} \notin \mathbf{X}$. The distancing of $\mathcal{X}_\mathcal{C}$ is defined as:*

$$distancing(\mathcal{X}_\mathcal{C}) = \frac{\sum_{\forall \text{ delta predicate } r \text{ in } \mathcal{X}_\mathcal{C}} delta(r)}{\sum_{\forall \text{ delta predicate } r \text{ in } \mathbf{X}} delta(r)}.$$

Definition 7 (Surprise). *Given a set $\mathbf{X}$ of explanation trees over $\mathcal{C}_{A,M,q}$ such that $\mathcal{X}_\mathcal{C} \notin \mathbf{X}$. The surprise of $\mathcal{X}_\mathcal{C}$ is defined as:*

$$surprise(\mathcal{X}_\mathcal{C}) = \frac{\sum_{\forall \mathcal{X} \in \mathbf{X}} \sum_{\forall \text{ rule } r \text{ of type } B \text{ in } \mathcal{X}} delta(r)}{\sum_{\forall \text{predicate } p \text{ in } \mathbf{X}} features(p)}.$$

Definition 8 (Diversity). *Given $\mathcal{S}$ the set of all predicates symbols excepting δ in the rules of $\mathcal{X}_\mathcal{C}$ (defined in Def. 3 for P,B,C,E and S types of explanations), and $\mathcal{P}_s$ the set of all predicates with predicate symbol s in the rules of $\mathcal{X}_\mathcal{C}$.The diversity of $\mathcal{X}_\mathcal{C}$ is defined as:*

$$diversity(\mathcal{X}_\mathcal{C}) = -\frac{\sum_{\forall s \in \mathcal{S}} \frac{n_s}{N} \log_2 \frac{n_s}{N}}{|\mathcal{S}|}$$

with n_s, the number of nodes features over predicates with symbol s: $n_s = \sum_{\forall p \in \mathcal{P}_s} |features(p)|$, and N, the total number of nodes features: $N = \sum_{\forall s \in \mathcal{S}} \sum_{\forall p \in \mathcal{P}_s} |features(p)|$.

3.3 Multi-Objective Optimization

Next, we use these quality indicators as optimization objectives to select the **optimal explanations** of the automated choice of a cleaning pipeline. When dealing with more than one dimension to optimize, there may be many incomparable sets of explanation. Therefore, we adapt the notion of explanation plan and Pareto plan as follows.

Definition 9 (Explanation Plan). *Explanation plan π_i, associated to x_i, a branch of an explanation tree $\mathcal{X}_\mathcal{C}$, is a tuple $\pi_i = \langle polarity(x_i), distancing(x_i), surprise(x_i), diversity(x_i) \rangle$.*

Definition 10 (Explanation Sub-Plan). *Explanation plan π_i is the sub-plan of another plan π_j if their associated explanation sub-trees satisfy $x_i \subseteq x_j$.*

Definition 11 (Dominance). *Plan π_1 dominates π_2 if π_1 has better or equivalent values than π_2 in every quality indicator. The term better is equivalent to greater for maximization objectives (e.g., diversity or polarity), and lower form in minimization ones (e.g., distancing or surprise) depending on the user's preferences. Furthermore, plan π_1 strictly dominates π_2 if π_1 dominates π_2 and the values of indicators for π_1 and π_2 are not equal.*

Algorithm 1 Approximated Pareto-optimal Explanations

Input: The user-defined optimization objective O, the precision value α, the maximum number of rules of an explanation k, the explanation tree $\mathcal{X}$, and the pipeline to explain x

Output: The set of explanations Π_α for x

```
Π_α ⟵ π_x
for all explanation branches B of the tree X \ x do
    π_B ⟵ construct_plan(B)
    if π_B is not α-dominated wrt O by any other plan in Π_α
        then Π_α.add(π_B)
for d ∈ [1, k] do
    for all explanation branches b of d rules in X \ x do
        π_b ⟵ construct_plan(b)
        if π_b is not α-dominated wrt O by any other plan in Π_α
            then Π_α.add(π_b)
return Π_α
```

Definition 12 (Pareto Plan). *Plan π_i is Pareto if no other plan strictly dominates π_i. The set of all Pareto plans is denoted as Π.*

Now, we define our problem as a multi-objective optimization (MOO) problem as follows: for an explanation tree $\mathcal{X}_\mathcal{C}$, the problem is to find all the explanation branches x, such that each branch satisfies:

- $diversity(x)$ is maximized;
- $surprise(x)$ is optimized;
- $polarity(x)$ is optimized;
- $distancing(x)$ is optimized;

Note that while we always maximize diversity, we may either minimize or maximize the distancing, surprise, and polarity based on the user's needs. The main challenge in designing an algorithm for finding optimal explanations of automated data cleaning is the multi-objective nature of the problem. A multi-objective problem can be easily solved if it is possible to combine all quality indicators into one or if the optimization of one indicator leads an optimized value of other indicators. However, in our problem, the indicators may be conflicting, i.e., optimizing one does not necessarily lead to an optimized value for others and they cannot be combined into one single indicator. Therefore, we propose Algorithm 1 based on approximation to select the Pareto-optimal explanation rules. The algorithm makes less enumerations with a theoretical guarantee on the quality of results. Its pruning mechanism uses the precision value α. In the special case of $\alpha = 1$, the algorithm operates exhaustively. If $\alpha > 1$, the algorithm prunes more and hence is faster. In the latter case, a new plan is only compared with all plans that generate the same result. But a new plan are only inserted into the buffer if no other plan approximately dominates it. This means that the algorithm ends to insert fewer plans than the exhaustive variant. Given a maximization objective O (e.g., diversity, surprise, distancing, polarity), plan π_1 with sub-plans π_{11} and π_{12}, and α. Derive π_2 from π_1 by replacing π_{11} by π_{21} and π_{12} by π_{22}. Then $O(x_{21}) \geq O(x_{11}) \times \alpha$ and $O(x_{22}) \geq O(x_{12}) \times \alpha$ together imply $O(x_2) \geq O(x_1) \times \alpha$. Similarly, the extension for a minimization objective is straightforward.

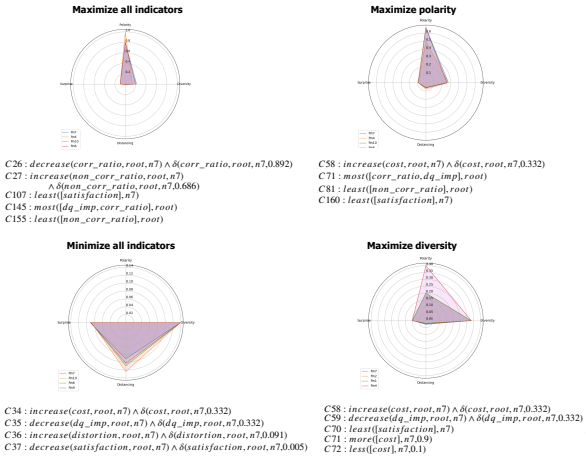


Figure 3. Optimal sets of explanations with their quality indicators depending on the multiple (left) single (right) objectives

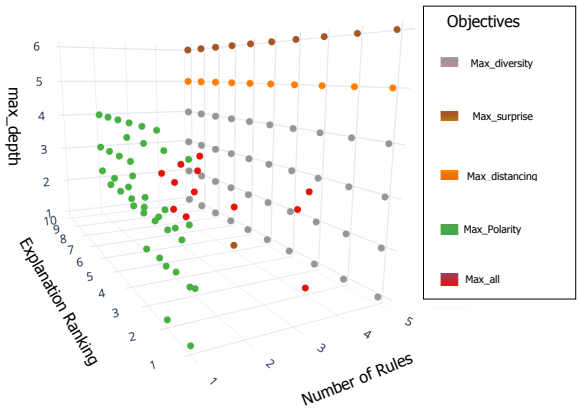


Figure 4. Explanation ranking with varying explanation tree depth, objectives, and number of rules

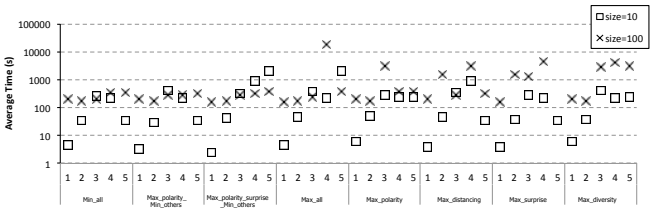


Figure 5. Time with varying explanation tree size (10, 100), max_depth (1 to 5), and objectives

The algorithm 1 exploits a dynamic programming approach. The algorithm begins by constructing a plan for each single branch of an explanation tree (lines 2 to 3). It keeps all non α-dominated plans in a buffer. Then, it examines all the branches of the explanation tree with depth 1 up to k (lines 6 to 10). After each iteration, it removes α-dominated plans from the buffer. Finally, it returns the buffer content as result.

4 Preliminary Experiments

Based on our observations from the automated cleaning pipeline search spaces, we generated two types of cleaning tree structures with 10 and 100 nodes respectively and a maximal depth varying from 1 to 5 levels from root to leaves. The number of nodes per level follows a Beta distribution (alpha=2, beta=2.35). The values of the six explanatory features were generated randomly five times. We ran five trials of picking one cleaning pipeline to explain among all the candidate ones for each of the 250 configurations ($2 \times 5 \times 5 \times 5$) and we averaged the results. We perform all experiments on a laptop Dell XPS machine with an Intel Core i7-7500U quad-core, 2.8 GHz, 16 GB RAM, powered by Windows 10 64-bit with g++ (GNU). CLeanEX, data sets, and more detailed experiments are available at `https://github.com/XXXX/cleanex`.

Fig. 3 illustrates different Type C explanations for explaining the same cleaning pipeline. Each set of explanations has been selected as it optimizes specific objectives in terms of quality indicators. We can observe that in general, no correlation exists between the optimized value of the objectives and the explanation set can be very different. Thus, each objective should be optimized independently and according to the user's preferences. Fig. 4 shows the result of our algorithm (with $\alpha = 1$ exhaustive search) for the top-10 ranked explanations depending on the optimization objectives. The ranking depends on the explanation size and the maximal depth of the tree: 2-3 rules for the explanations maximizing all objectives for depth from 2-3 (Max_all). In Fig. 5, we can also examine the effect of the tree structure (in terms of the number of nodes and maximal depth) on execution time of CLeanEX to find the optimal explanations. We observe that the execution times for trees containing 10 and 100 nodes have similar values as their respective depth increases. The reason is that the number of subsets of candidate rules for a given branch increases quickly when the depth of the tree also increases. Inversely, reducing the depth of the branches increases the number of branches to explore, for which the depth generally decreases, reducing the time complexity for the cases with few nodes. The execution time of our algorithm is spent on the computation of the quality indicators, the generation of the rules, and the exploration of a large number of candidates rules sets. However, changing α will reduced this last dimension while providing an approximation guarantee.

5 Conclusions

In this paper, we propose CleanEX, a framework to generate explanations for automated data cleaning. The key advantages of our approach is that our explanation system is: (1) model-agnostic: it can be applied to any ML model, any set of data cleaning tasks, and any automated cleaning agent that can provide its cleaning pipeline search space; (2) logic-based: explanations are comprehensible to humans with diverse expertise and can be extensible to handle causal reasoning; (3) both local and global: it can explain both some part or the whole cleaning pipeline; (4) model quality-independent: it provides a reliable set of explanations independently from the ML model quality performance metric that is used by the automated cleaning agent for the selection of the optimal cleaning pipeline; and (5) user-defined: optimization is based on the user's objectives.

References

[Berti-Equille, 2019] L. Berti-Equille. Learn2Clean: Optimizing the Sequence of Tasks for Web Data Preparation. In *The Web Conf.*, 2019.

[Bertossi and Geerts, 2020] L. Bertossi and F. Geerts. Data quality and explainable ai. *ACM Data and Information Quality J.*, 2020.

[Dasu and Loh, 2012] T. Dasu and J. M. Loh. Statistical distortion: Consequences of data cleaning. *PVLDB*, 5(11):1674–1683, 2012.

[Deutch *et al.*, 2020] D. Deutch, N. Frost, A. Gilad, and O. Scheffer. T-rex: Table repair explanations. In *ACM SIGMOD*, 2020.

[Ebaid *et al.*, 2019] A. Ebaid, S. Thirumuruganathan, W. G. Aref, A. K. Elmagarmid, and M. Ouzzani. EXPLAINER: entity resolution explanations. In *IEEE ICDE*, pages 2000–2003, 2019.

[Guidotti *et al.*, 2018] R. Guidotti, A. Monreale, S. Ruggieri, F. Turini, F. Giannotti, and D. Pedreschi. A survey of methods for explaining black box models. *ACM Comput. Surv.*, 51(5), August 2018.

[Krishnan *et al.*, 2015] S. Krishnan, J. Wang, M. J. Franklin, K. Goldberg, T. Kraska, T. Milo, and E. Wu. Sampleclean: Fast and reliable analytics on dirty data. *IEEE Data Eng. Bull.*, 38(3):59–75, 2015.

[Krishnan *et al.*, 2017] S. Krishnan, M. J. Franklin, K. Goldberg, and E. Wu. Boostclean: Automated error detection and repair for machine learning. *CoRR*, abs/1711.01299, 2017.

[Pedreschi *et al.*, 2019] D. Pedreschi, F. Giannotti, R. Guidotti, A. Monreale, S. Ruggieri, and F. Turini. Meaningful explanations of black box ai decision systems. In *AAAI*, volume 33, pages 9780–9784, 2019.

[Rammelaere and Geerts, 2018] J. Rammelaere and F. Geerts. Explaining repaired data with cfds. 11(11):1387–1399, 2018.

[Rekatsinas *et al.*, 2017] T. Rekatsinas, X. Chu, I. F. Ilyas, and C. Ré. Holoclean: Holistic data repairs with probabilistic inference. *PVLDB*, 10(11):1190–1201, 2017.

[Ribeiro *et al.*, 2016] M. T. Ribeiro, S. Singh, and C. Guestrin. "why should I trust you?": Explaining the predictions of any classifier. In *ACM SIGKDD 2016*, pages 1135–1144, 2016.

[Ribeiro *et al.*, 2018] M. T. Ribeiro, S. Singh, and C. Guestrin. Anchors: High-precision model-agnostic explanations. In *AAAI*, 2018.

[Shang *et al.*, 2019] Z. Shang, E. Zgraggen, B. Buratti, F. Kossmann, P. Eichmann, Y. Chung, C. Binnig, E. Upfal, and T. Kraska. Democratizing data science through interactive curation of ML pipelines. In *ACM SIGMOD*, pages 1171–1188, 2019.

[Yakout *et al.*, 2013] M. Yakout, L. Berti-Équille, and A. K. Elmagarmid. Don't be scared: use scalable automatic repairing with maximal likelihood and bounded changes. In *ACM SIGMOD*, pages 553–564, 2013.