Learn2Clean: Optimizing the Sequence of Tasks for Web Data Preparation

Laure Berti-Equille [1,2]
[1]ESPACE-DEV/IRD, UMR 228, IRD/UM/UG/UR, Montpellier, France
[2]Aix Marseille Université, Université de Toulon, CNRS, LIS, DIAMS, Marseille, France
laure.berti@ird.fr

ABSTRACT

In many applications, data mining and machine learning methods are extensively used to analyze Web data and discover actionable knowledge. But, "dirty data" is a chronic plague that causes incorrect results, misleading conclusions, generally followed by inadequate decisions. To ensure the validity of output results, avoid bias or data snooping, it is necessary to control not only the whole Web data analytics pipeline, but most importantly the quality of Web data with appropriate data preparation and curation choices. For a given dataset and a given machine leaning model, a plethora of data preprocessing techniques and alternative data cleaning strategies may lead to dramatically different outputs with unequal quality performance. It is then crucial to rely on a principled method to select the sequence of data preprocessing steps that can lead to the optimal quality performance. In this paper, we propose Learn2Clean, a method based on Q-Learning, a model-free reinforcement learning technique that selects, for a given dataset, a ML model, and a quality performance metric, the optimal sequence of tasks for preprocessing the data such that the quality of the ML model result is maximized. As a preliminary validation of our approach in the context of Web data analytics, we present some promising results on data preparation for clustering, regression, and classification on real-world data.

CCS CONCEPTS

• **Information systems** → **Data extraction and integration**; • **Computing methodologies** → **Reinforcement learning**.

KEYWORDS

Principled data preprocessing; Data cleaning; Q-Learning; Reinforcement learning

ACM Reference Format:
Laure Berti-Equille [1,2]. 2019. Learn2Clean: Optimizing the Sequence of Tasks for Web Data Preparation. In *Proceedings of the 2019 World Wide Web Conference (WWW '19), May 13–17, 2019, San Francisco, CA, USA.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3308558.3313602

WWW '19, May 13–17, 2019, San Francisco, CA, USA

ACM ISBN 978-1-4503-6674-8/19/05.
https://doi.org/10.1145/3308558.3313602

1 INTRODUCTION

Through the use of data mining, statistics, and machine learning, data scientists and researchers can discover relevant patterns and gain crucial and actionable knowledge from data. In the context of Web data analytics, it is necessary to extract, transform, and structure the data and also adapt the data volume, data format, and distributional characteristics in order to better suit the underlying assumptions and constraints of statistical machine learning (ML) models so that they can be effectively applied to the preprocessed input data. It is essential to detect glitches (aka data quality problems) [2] because erroneous data limit the performance of statistical methods and cause misleading results [3, 11].

Web Data can be big, noisy, unreliable, highly imbalanced, heterogeneous, and evolve over time [6]. In practice, Web data analytics can suffer from a wide range of data anomalies such as missing values, inconsistencies, outlying data, duplicates, etc. The results of classification, clustering or predictive models obtained from the data that were preprocessed inappropriately are misleading and can cause inadequate decision-making. To mitigate the impact of data anomalies and thus provide high quality analytical results, a first necessary step is to preprocess data appropriately.

Data preprocessing has been acknowledged as a primary sequence of tasks to correct the negative effects and bias that data anomalies may produce on output results as demonstrated in various contexts [9, 12]. Common data preprocessing tasks mainly include: data extraction and formatting, selection of features, data normalization, treatment of missing values and outliers, deduplication, inconsistency detection (wrt some rules and constraints), data replacement, and pattern enforcement. Although there is a vast number of data preprocessing methods available to achieve these tasks in the context of Web mining [12–16], the selection and orchestration of the sequence of tasks often remains *ad hoc*, difficult, and with no guarantee that it is optimal. An important question is: How do we actually measure the quality of data preprocessing for a given analytics task? This is a challenging problem with combinatorial issues if we consider the infinite "curation space" with hundreds of possible techniques, each with various parameter settings, and multiple possibilities in the combination and the ordering of the cleaning tasks, and even worse when we consider multiple iterations of certain tasks.

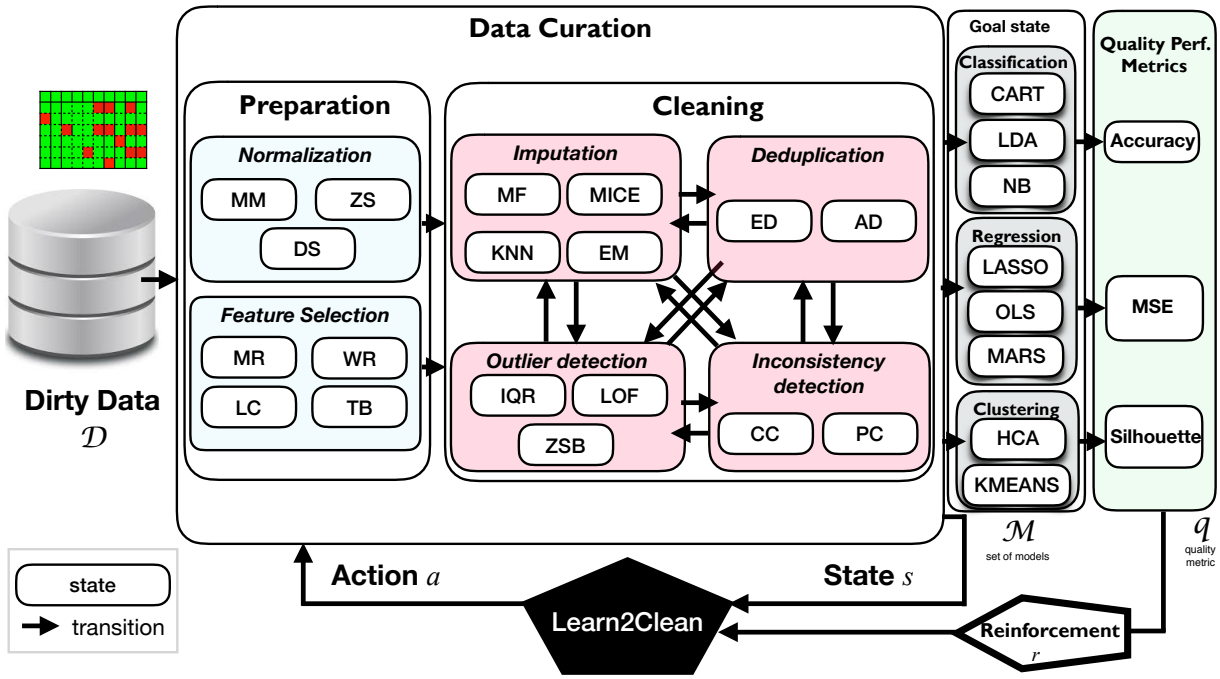

Figure 1: Learn2Clean Architecture

For example, in the context of Web usage data mining, data characterizing the interaction of Internet-users with Web sites are collected and analyzed in order to provide personalized Web-pages or enhance search engines. To prepare such data before analysis, one may define a preprocessing strategy that starts with a distance-based deduplication for Web user and session identification, then the removal of outliers, and ends with a regression-based imputation of missing values; another expert may use a totally different sequence of methods with another ordering and add the re-iteration of outlier detection and removal. Ultimately, the two preprocessed datasets may be so different that the conclusions from classification or clustering results may also differ radically. Similarly in the context of Web content mining, the information presented on Web pages combining text and multimedia content is used for page classification, page summarization, entity extraction, and semantic processing. The data need to be filtered because many objects present on Web pages are not relevant with the stated objectives (e.g., digital marketing or advertising). Text content by itself is noisy: for example, sentences contain many words that have a poor influence on the semantic content and various strategies for text preprocessing, Web content cleaning and transformation into feature vectors may lead to very different mining results.

There are several methods that optimize single data cleaning operations based on downstream accuracy metrics (see [18] for a survey) and also recent approaches such as [17], data transformation learning [22], and AutoML line of work [19, 20] in the area of automated machine learning. However, the problem of selecting the optimal data preprocessing pipeline has been understudied.

The focus of our work is first to study the impact of various sequences of data preprocessing methods on the output results of some representative ML models; second, we propose a novel method that learns by reinforcement the sequence of tasks that can optimize a preselected quality performance metric for the considered ML model. In our study, we consider the following tasks of (1) selecting features, (2) normalizing data, dealing with (3) missing values and (4) outliers, (5) deduplicating, and (6) consistency checking, since these methods are the most frequently used for preparing data.

2 OVERVIEW OF LEARN2CLEAN

Our overall approach can be summarized in Figure 1. We consider a ML pipeline with a particular goal such as classification, regression or clustering. The ML pipeline includes: (i) the data $\mathcal{D}$; (ii) the feature set $\mathcal{X}$; (iii) the learning algorithm $\mathcal{M}$, along with the objective function $\mathcal{L}$ that maximize a quality performance metric q; and, possibly, (iv) its (trained) parameters $\mathbf{w}$. The ML pipeline can thus be formalized in terms of a space $\Theta = (\mathcal{D}, \mathcal{X}, \mathcal{M}, \mathbf{w})$. Input data preprocessing is strongly dependent on the given practical scenario. For example, additional constraints while preprocessing the data have to be considered to conform the data with the ML model requirements (e.g., normalization, discretization, deduplication, missing value imputation, extreme values replacement, etc.). Typically, these constraints can be nevertheless accounted for in the definition of the optimal curation strategy.

In particular, we characterize them by assuming that $\mathcal{D}'$ is the dataset resulting from a sequence of data modifications in the space of all possible data cleaning transformations of $\mathcal{D}$ such as $D' \in \Phi(\mathcal{D})$ with $\Phi(\mathcal{D}) = \{\phi(\mathcal{D})|\forall\phi\}$. The problem can be formalized as follows:

Given a ML pipeline $\theta \in \Theta$ and the data sets resulting from all preprocessing alternatives $\Phi(\mathcal{D})$, the goal of preprocessing can be characterized in terms of an objective function $\mathscr{P}_q \in \mathbb{R}$ which evaluates how effective the preprocessing alternative D' is with respect to a given quality metric q. The result of the optimal preprocessing strategy can be thus given as:

$$D^\star = \underset{D' \in \Phi(\mathcal{D})}{\operatorname{argmax}} \mathscr{P}_q(D', \mathcal{X}, \mathcal{M}, \mathbf{w}). \quad (1)$$

More intuitively, the problem we address is the following: *Given a dataset as input $\mathcal{D}$, a ML pipeline θ to apply to the input dataset, a quality performance metric q, and the space of all possible data preparation and cleaning strategies $\Phi(\mathcal{D})$: Find the dataset D' in $\Phi(\mathcal{D})$ that maximizes the quality metric q.*

To solve this problem, we propose Learn2Clean a system based on Q-learning that learns in an on-line fashion with interleaving learning and execution. The intuition is the following. In the early stages of execution, when little is known, it is important to explore and try some data curation actions when their consequences on the output quality performance of the given ML model are still unknown. Rewards are given when the quality performance improves. Later on, the system may want to almost always choose the action that gets highest rewards; when there is little information to be gained, the value of learning can be negligible. This is modeled with a Q-learning function that assigns a probability of being chosen for each possible action in a given state such that it should tend to choose actions with higher Q-values, but should sometimes select lower Q-value actions.

3 Q-LEARNING

Q-learning [10] is a form of model-free reinforcement learning, where the system only needs to know what state exists next and what actions are possible in each state. An estimated value Q-value is assigned to each state. When the system visits a state and receives a reward, the Q-value of that state is updated accordingly such as:

$$Q(s, a) = R(s, a) + \gamma max_{a'} Q(s', a') \quad (2)$$

where the value of taking action a in state s is the immediate reward denoted $R(s, a)$ for the state-action pair (s, a), plus the value of the best possible state-action pair for the successor state. The γ parameter has a range of 0 to 1 ($0 \leq \gamma < 1$). If γ is closer to zero, the system will tend to consider only immediate rewards. If it is closer to 1, the system will consider future rewards with greater weight, willing to delay the reward. We can use Eq. (2) to update the Q-value of a state-action pair as a reward r' is observed:

$$Q_{t+1}(s, a) = r' + \gamma max_{a'} Q(s', a'). \quad (3)$$

r' is the reinforcement received when the system selected the a action in the state s to move to the state s'. In our context, we

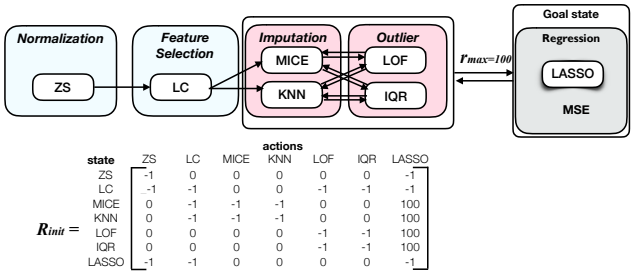

Figure 2: Simplified example of state-action graph with initial reward matrix.

represent data preprocessing and curation strategy as a graph: each state is a node representing a preparation or cleaning task that generates a "cleaner" version of the original dirty dataset. A sequence of tasks is represented by a link between two nodes (i.e., between a cleaning task and the next cleaning task). In the architecture presented in Figure 1, we can start with any normalization method as a task (e.g., ZS, MM, SM or DS) or feature selection method (e.g., CFS, WR, LC)[1], and move to the next task choosing one method amongst the ones proposed either for deduplication, inconsistency checking, outlier detection or imputation, until the system reaches the goal state. We define the goal state and a particular quality metric we want to maximize denoted $G(q)$. For example, as shown in Figure 2, the goal state for a LASSO regression applied to the input dataset is the state of preprocessing that minimizes the mean squared error, MSE. To reach this node as a goal, we associate a reward value to each link between nodes. The nodes that lead immediately to the goal node have an instant reward of r_{max}. There may be no link between nodes (encoded as -1 in the Reward R-matrix represented in Figure 2). In this example, particular cleaning paths can be avoided or task precedence can be enforced: e.g., after normalization, there is feature selection, then imputation (either using MICE or KNN), and then outlier detection (either with LOF or IQR), and again imputation. This figure represents a particular graph instantiation of Learn2Clean architecture presented in Figure 1. The final goal state is ML model-dependent since the dataset should be prepared and curated with respect to the specific requirements of the ML model considered for the analysis. The goal state in the example of Figure 2 is reached after a maximum number of 100 iterations when LASSO regression is executed after either imputation or outlier detection such that MSE (the considered quality performance metric) is minimal.

Reinforcement via reward. The reinforcement function provides, for each new state s' reached from state s, a signal r' that can be a reward or punishment, this signal takes the value $\{-1\}$ when the state s is not linked to the new state s' or a value in $[0, 1]$, which is used by the update function in Eq. (3) to adjust the Q-value function associated with the

[1]Acronyms of the methods are described in Section 3.1.

state-action pair. The reward r' is computed as follows:

$$r' = \beta(Norm(s, q_m) - Norm(s', q'_m)) \quad (4)$$

where $Norm(s, q_m)$, $Norm(s', q'_m)$ are the normalized quality metric values q and q' in [0,1] for the considered ML model m and

$$\beta = \begin{cases} -r_{max} & \text{if } Norm(s, q_m) - Norm(s', q'_m) < 0 \\ r_{max} & \text{if } Norm(s, q_m) - Norm(s', q'_m) > 0 \end{cases}$$

with r_{max}, the user-defined maximum reward value. This penalizes the next data curation task when it provides a lower quality performance metric compared to the ones obtained after the current cleaning task.

Selection of the next curation tasks. The probability of selecting the highest Q-value action should increase over time and this can be defined by the Boltzmann distribution as follows:

$$P(a|s) = \frac{e^{Q(s,a)/k}}{\sum_j e^{Q(s,a_j)/k}}. \quad (5)$$

The k parameter (often referred to as temperature) controls the probability of selecting non-optimal actions. If k is large, all actions will be selected fairly uniformly. If k is close to zero, the best action will always be chosen. We begin with k large and gradually decrease it over time. Each edge contains an instant reward value, as shown in the initial reward matrix R_{init} given as example in Figure 2.

Learn2Clean will learn through experience in an unsupervised way. It will explore from state to state until it reaches the goal. Each exploration consists of the system moving from the initial state to the goal state. Each time, it arrives at the goal state, the program goes to the next exploration. Learn2CLean algorithm is given in Algorithm 1 where the system learns from experience. Each exploration is equivalent to one training session. In each training session, the system explores the curation graph (represented by matrix R), receives the reward (if any) until it reaches the goal state (lines 6–8). The purpose of the training is to enhance the decision of the system, represented by matrix Q to get more training results in a more optimized matrix Q. In this case, if the matrix Q has been enhanced, instead of exploring around, and going back and forth to the same tasks, the system will find the fastest route to the goal state.

The algorithm finds the actions with the highest reward values recorded in matrix Q (lines 9–10) and returns the sequence of states from the initial state to the goal state where the quality metric is optimal (either minimized for regression MSE or maximized for classification accuracy or clustering silhouette) and the ultimately cleaned dataset to be used for the considered ML model (line 13).

Why Q-learning? In model-based reinforcement methods, the model learns the transition probability $T(s_1|(s_0, a))$ from the pair of current state s_0 and action a to the next state s_1. However, model-based algorithms become impractical as the state space and action space grows, which is typically the case with the space of data cleaning strategies. Moreover, the transition probability and the dynamics of the system is not given a priori. These are the reasons why model-free

Algorithm 1 Learn2CLean Algorithm

```
1. INPUT γ and reward matrix R
2.        ML model m, quality metric q
3. Initialize Q matrix
4. FOR each exploration
5.     Select the initial state
..     Apply the corresponding curation method to generate
..     a "cleaner" dataset, and apply m to it
6.     WHILE {goal G(m,q) has not been reached}
7.         Select one among all possible actions with prob.
..             in Eq.(5) for the current state
8.         Using this possible action, move to the next state
9.         Get max Q-value for this next state based on all
..             possible actions
10.        Compute Eq.(2)
11.        Set the next state as the current state.
12.    END_WHILE
13. RETURN the list of actions with highest Q-values and
..         optimal quality metric and the final curated dataset
```

algorithms are more adequate for our problem. We choose Q-Learning instead of other model-free reinforcement learning methods (like Monte-Carlo or SARSA methods) because Q-learning will not wait until the end of the episode (i.e., cleaning task) to update the expected future reward estimation, it will only wait until the next time step to update the value estimates and may not use the same policy to choose the next action. This makes it agile and performant compared to other model-free reinforcement learning methods.

3.1 Data preprocessing methods

As shown in Figure 1, various types of representative preprocessing techniques were used in our study with the long-term goal to provide an extensible library for data cleaning and preparation.

- **Normalization**. Three methods were applied to numerical data: min-max (MM), Z-score (ZS), and decimal scale normalization (DS);
- **Feature selection**. Four methods were tested to select an optimal set of features, namely: based on a user-defined acceptable ratio of missing values (MR), removing collinear features (LC), using a wrapper subset evaluator (WR) [8], and a tree-based classifier for feature selection (TB);
- **Imputation**. Four imputation methods were used to impute missing values: two distance-based imputation methods: *Expectation-Maximization* (EM), *K-NN* [1], Multiple Imputation by Chained Equations (*MICE*) [5], and replacement by the most frequent value (MF);
- **Outlier detection**. Outliers were detected using three methods including: a statistic-based approach, Inter Quartile Range (IQR), the Z-score-based method (ZSB) [7], and a density-based approach (Local Outlier Factor, LOF) [4]. Outliers were replaced using one of the previous imputation methods;

- **Deduplication**. Exact duplicate (ED) and approximate duplicate (AD) detection methods were used. In the later case, various comparison functions were used to measure the similarity distance between pairs of records to detect duplicates based on the user-defined threshold (e.g., Jaro-Winkler or Levenshtein distance);
- **Consistency checking**. Two methods based on constraint discovery and checking (CC) and pattern checking (PC) were used to identify inconsistencies.

3.2 ML Models and Quality Performance

Our goal is to study the impact of data preprocessing pipeline selection and how it can affect the output results of statistical ML methods. In particular for Web usage data analysis, we considered the following ML models and corresponding quality performance metrics.

- **Clustering**. We used two clustering methods: HCA (Hierarchical Clustering) and K-means with silhouette metric to evaluate the quality of the clustering as $silhouette = (b(i) - a(i))/max\{a(i), b(i)\}$ with $a(i)$ the average distance between i and all other data within the same cluster and $b(i)$ the smallest average distance of i to all points in any other cluster, of which i is not a member;
- **Regression**. We used three regression methods: LASSO (Least Absolute Shrinkage and Selection Operator), OLS (Ordinary Least Squares Regression), and MARS (Multivariate Adaptive Regression Splines) and as quality metrics, we used MSE (Mean Squared Error) to measure the robustness of regression as $MSE = \sum_{i=1}^{N}(Y_i - \hat{Y}_i)^2/N$; and
- **Classification**. We used three classification methods: LDA (Linear Discriminant Analysis), NB (Naive Bayes), and CART (Classification and Regression Trees) with accuracy metric as $accuracy = (TP + TN)/(TP + TN + FP + FN)$ with TN and FN true and false negatives and TP and FP true and false positives respectively in the classification result.

Our approach is not limitative and other ML models, quality metrics or preprocessing methods could be added and used by Learn2CLean.

4 EXPERIMENTS

We have performed preliminary experiments to compare the preprocessing alternatives (described in Section 3.1) for the classification, regression, and clustering methods (described in Section 3.2) using real-world datasets from Kaggle[2] repository as shown in Table 1.

All methods and Learn2CLean are implemented in Python (Jupyter Notebook 5.5, Python 3.6.5). Source code and datasets used in our study are available at: https://github.com/LaureBerti/Learn2Clean.

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

Table 1: Real-world datasets

Datasets	[#]Att.	[#]Rows	Clustering	Regression	Classification
Google Play Store Apps	13	10.8k	X		X
Google Play Store Users	5	64.3k	X		
House Prices	81	1.46k	X	X	X

4.1 Compared Methods

We have compared the results of data preprocessing strategy returned by Learn2CLean to four types of data cleaning: (1) random selection ($RAND$) of a sequence of one (or none) curation task from each block of possible actions for normalization, feature selection, imputation, outlier, duplicate and inconsistency detection and removal (as depicted in Figure 1) with no particular order; (2) manual cleaning by a human expert in Data Science (DS_EXP); (3) one automated ML approach ($AUTO$) obtained by MLBox[3] including cleaning and preprocessing; and (4) *NO-PRE* when no data preparation/curation is achieved. In particular, we have compared the quality metrics of the ML models applied to the preprocessed datasets obtained from these various preparation strategies, respectively MSE for regression, accuracy for classification, and silhouette for clustering[4]. In the case of random preprocessing, we averaged the results in terms of quality and time over the total number of execution runs.

We used default parameter settings for all the ML models with no particular hyper-parameter optimization and future work will be to integrate this important aspect and provide an in-depth experimental analysis and a fair comparison with other AutoML approaches [20] (such as AutoSklearn[5]). In these approaches, the user has to predefine the pipeline of default preprocessing methods to be used for feature selection (e.g., drift thresholding for MLBox or variance thresholding for AutoSklearn) and imputation techniques (e.g., One-Hot encoding for AutoSklearn). They do not include approximate duplicate or inconsistency detection/correction. Feature preprocessing is a single transformer which implements, for example, feature selection or transformation of features into a different space (i.e., PCA). The AutoML preprocessors do not test and select the optimal preprocessing method for a given dataset and their core added-value is on hyper-parameter optimization of the ML models.

In average, it takes 1.55 seconds for Learn2CLean to enumerate all possible cleaning strategies given the 19x19 Q-matrix for the 18 possible preprocessing methods. Then, each strategy as a sequence of curation tasks is executed with a duration depending on the dataset size, the number of glitches to detect and correct, and the time complexity of each task.

[3]https://mlbox.readthedocs.io/

[4]For technical details on the quality metrics: https://scikit-learn.org/stable/modules/classes.html#module-sklearn.metrics

[5]https://github.com/automl/auto-sklearn/tree/master/autosklearn/pipeline/components/data_preprocessing

Table 2: Experimental results for clustering, classification, and regression applied to three real-world datasets

Dataset	Model (metric)	*NO_PRE*	*RAND*	*DS_EXP*	*AUTO*	Learn2CLean	Learn2CLean Best strategy	Time (seconds)
Google Play Store Apps	HCA (Silhouette)	**0.965**	0.775	**0.963**	**0.965**	**0.968**	LC→MF→HCA	55.6
	KMEANS (Silhouette)	0.668	0.798	0.652	**0.808**	**0. 801**	LOF→AD→KMEANS	46.1
	LDA (Accuracy)	0.071	0.085	0.734	0.766	**0.796**	MICE→CART	15.5
	NB (Accuracy)	0.175	0.158	0.679	**0.786**	0.722	WR→IQR→MICE→NB	13.5
Google Play Store Users	HCA (Silhouette)	**0.965**	0.775	0.878	0.804	**0.965**	HCA	14.8
	KMEANS (Silhouette)	0.417	0.429	**0.676**	0.587	**0.655**	MR→AD→KMEANS	39.5
House Prices	HCA (Silhouette)	0.793	0.714	**0.846**	**0.844**	**0.844**	MM→ED→HCA	32.2
	KMEANS (Silhouette)	0.644	0.609	0.632	**0.674**	**0.678**	KMEANS	28.3
	CART (Accuracy)	NA	NA	**0.611**	0.345	0.449	MM→AD→CART	58.7
	NB (Accuracy)	NA	NA	**0.663**	0.457	0.518	LC→MF→NB	20.1
	LASSO (MSE)	> 1	> 1	0.165	0.138	**1.44E-12**	KNN→ZSB→LASSO	24.2
	MARS (MSE)	0.175	0.192	**0.163**	NA	0.231	IQR→MICE→MARS	134.5
	OLS (MSE)	> 1	> 1	0.176	0.137	**2.20E-25**	KNN→ZSB→OLS	45.1

Ultimately, the strategy with best quality metric is selected by Learn2CLean. In the last columns, Table 2 presents the pipeline of curation selected by Learn2CLean and the overall execution time.

4.2 Clustering

Clustering analysis was performed using K-Means (KMEANS) and Hierarchical clustering (HCA) methods. The number of clusters that maximizes the silhouette for each clustering method was automatically selected to perform the clustering. Table 2 presents the silhouette measures obtained by KMEANS and HCA applied to the Learn2CLean-preprocessed vs no-preprocessing, random, manually, and AutoML curated datasets. We can observe that Learn2CLean and *AUTO* show the best silhouette values for the three datasets, close or better to what an expert would obtain in reasonable time: less than one minute for Google Play Store Apps dataset and from 15 to 40 seconds for the other datasets.

4.3 Classification

For classification, the observations were split into a training and test set where 70% of data was used for training and 30% for testing. The model was fit to the training set, and the fitted model was used to predict the responses for the observations in the testing set. Classification methods cannot be executed and accuracy cannot be computed in presence of missing values, which is the case for House Prices dataset where 3 variables have from 80% to 99% missing values (MiscFeature, Fence, PoolQC). The classification methods require first feature selection based on the ratio of missing values for example, then imputation for other missing values, and another feature selection to avoid collinearity. Consequently, *RAND* and *NO-PRE* are not applicable (NA) and *DS_EXP* curation is the best strategy for this dataset because the preprocessing pipeline is quite complex with reiteration of feature selection methods before and after handling missing values. For this set of experiments, we initialized the Q-matrix of Learn2CLean to avoid re-iteration of methods from the same block. In other words, feature selection methods are mutually exclusive in the same pipeline; similarly only one imputation method can be applied. This explains the relatively low accuracy of Learn2CLean in this case and the need of expertise. However, for Google Play Store Apps classification, *AUTO* and Learn2CLean the best accuracy values for NB and and LDA respectively; not surprisingly, *NO-PRE* and *RAND* give the worst accuracy values.

4.4 Regression

For LASSO and OLS regression, we can observe that Learn2CLean has the lowest MSE for House Prices dataset and it outperfoms the other curation approaches. For *RAND* and *NO-PRE*, MSE values are not reliable (>1) because of high collinearity of the dataset variables.

In conclusion, the results of our limited set of experiments is promising and has opened several directions for research and improvements of Learn2CLean algorithm and the experimental set-up. In particular, we plan further experiments with semi-synthetic data where noise injection is carefully controlled in order to study more complex and intricate distributions of various glitch types and patterns [3] (combining duplicates, inconsistencies, missing and outlying values) and their impact on the robustness of our approach. We also plan to combine AutoML hyper-optimization with Learn2CLean extended library of preprocessing methods and Q-learning based selection of the best preparation strategy.

5 CONCLUSIONS

We proposed an approach based on Q-learning that identifies the optimal data preprocessing strategies for representative ML models applied to various real-world datasets. Although there is no universally adequate data preprocessing procedure (as it depends on many characteristics of the dataset in terms of size, data and glitches distributions), we were able to learn hidden features for selecting the optimal data preprocessing for a given dataset and a given ML task. Our preliminary results show that an instantiation of the proposed framework based on Q-Learning is efficient and a promising direction for

automating data curation. As future work, we plan to explore a wider spectrum of datasets from diverse applications of Web content extraction and mining, and extend the range of data preprocessing methods and ML tasks in order to test Learn2CLean in a larger range of settings.

REFERENCES

[1] Batista G., Monard M.C., et al. (2002). A Study of K-Nearest Neighbour as an Imputation Method, HIS, vol. 87, pp.48.

[2] Berti-Equille L., Loh J.M., Dasu T. (2015). A masking index for quantifying hidden glitches. Knowledge and Information Systems, 44, pp.253–277.

[3] Berti-Equille L., Dasu T., Srivastava D. (2011). Discovery of complex glitch patterns : A novel approach to Quantitative Data Cleaning. Proc. ICDE, pp.733–744.

[4] Breunig M.M., Kriegel H.-P., Ng R.T., Sander J. (2000). LOF: identifying density-based local outliers, ACM SIGMOD Record, vol. 29(2), pp.93–104.

[5] Buuren S., Groothuis-Oudshoorn K. (2011). MICE: Multivariate imputation by chained equations in R, J. of Statistical Software, vol. 45, num. 3.

[6] Berti-Equille L., Scannapieco M. (2016). Quality of Web Data (Chapter). In the 2nd Edition of the book Data Quality: Concepts, Methodologies and Techniques, Springer, 2016

[7] Filzmoser P., Garrett R.G., Reimann D. (2005). Multivariate outlier detection in exploration geochemistry, Computers & Geosciences, vol. 31, pp.579–587.

[8] Kohavi R., John G. H. (1997). Wrappers for feature subset selection, Artificial intelligence, vol. 97, pp.273–324.

[9] Serrano Balderas E.C., Berti-Equille L., Armienta Hernandez M.A., Grac C., (2017). Principled Data Preprocessing: Application to Biological Aquatic Indicators of Water Pollution. Proc. of the DEXA-BIOKDD'17 workshop.

[10] Watkins J.C.H. (1989). Learning from Delayed Rewards, PhD Thesis, University of Cambridge, England.

[11] Zaveri A., Maurino A., Berti-Equille L. (2014). Web Data Quality: Current State and New Challenges, Int. J. Semant. Web Inf. Syst.,10(2):1552-6283, IGI Global.

[12] Suresh R.M., Padmajavalli R. (2006). An Overview Of Data Preprocessing In Data and Web Usage Mining, Proc. of the 1st International Conference on Digital Information Management, pp.193–198.

[13] Vellingiri J., Chenthur Pandian S. (2011). A Novel Technique for Web Log Mining with Better Data Cleaning and Transaction Identification, Journal of Computer Science, pp.683–689.

[14] Chang-bin J., Li C. (2010). Web Log Data Preprocessing Based On Collaborative Filtering, Proc. of IEEE 2nd International Workshop On Education Technology and Computer Science, pp.118–121.

[15] Zheng L., Hui Gui H., Li F. (2010). Optimized Data Preprocessing Technology For Web Log Mining, Proc. of IEEE International Conference On Computer Design and Applications (ICCDA), pp. VI-19-VI-21.

[16] Sudheer Reddy K., Kantha Reddy M., Sitaramulu V. (2013). An effective data preprocessing method for Web Usage Mining, Proc. of the 2013 International Conference on Information Communication and Embedded Systems (ICICES).

[17] Bilalli B., Abelló A. (2018). PRESISTANT: Learning based assistant for data preprocessing, Arxiv CoRR abs/1803.01024.

[18] Chu X., Ilyas I. F., Krishnan S., Wang J. (2016). Data Cleaning: Overview and Emerging Challenges. Proc. of the 2016 International Conference on Management of Data (SIGMOD'16), pp.2201–2206.

[19] Feurer M., Hutter F. (2018). Towards Further Automation in AutoML, Proc. of ICML 2018 AutoML Workshop.

[20] Feurer M., Klein A., Eggensperger K., Springenberg J., Blum M., Hutter F. (2015). Efficient and Robust Automated Machine Learning, Advances in Neural Information Processing Systems 28, December, pp.2962–2970.

[21] Vanschoren J., van Rijn J. N., Bischl B., Torgo L. (2014). OpenML: Networked science in machine learning, ACM SIGKDD Explorations Newsletter, 15(2):49-60.

[22] Wu B., Knoblock C. A. (2016). Maximizing Correctness with Minimal User Effort to Learn Data Transformations. IUI 2016, pp.375-384.