An Empirical Study on the (Non-)Importance of Cleaning Categorical Duplicates before ML

ABSTRACT

The tedious grunt work involved in data preparation (prep) before ML reduces ML user productivity and is a major impediment for many ML applications. It is also a roadblock to industrial-scale cloud AutoML workflows that build ML models for millions of datasets. One important data prep step for ML is cleaning duplicates in the *Categorical* columns, e.g., deduplicating *CA* with *California* in a *State* column. However, how deduplication impacts ML is ill-understood as there exist almost no in-depth scientific studies to assess their significance. In this work, we take the first step towards empirically characterizing the impact of *Categorical* duplicates on ML with a three-pronged approach. We first study how duplicates exhibit themselves by creating a labeled dataset of 1248 *Categorical* columns, where true entities in each column are annotated with their duplicates. We then curate a downstream benchmark suite of 14 real-world datasets to make observations on the effect of duplicates on downstream classifiers. We finally use simulation studies to validate our observations and disentangle multiple confounders that impact ML. We find that duplicates can cause Random Forest accuracy to drop by a median of 1.6% and up to 11.5% compared to the deduplicated truth. Moreover, Logistic Regression and *Similarity* encoding for *Categoricals* are more robust to duplicates than Random Forest. We provide intuitive explanations and takeaways that can potentially help both AutoML developers to build better platforms and ML practitioners to reduce grunt work. Our work presents novel data artifacts and benchmarks, as well as novel empirical analyses that can help advance the science of data prep on AutoML platforms.

1 INTRODUCTION

Automated machine learning (AutoML) is beginning to increase access to ML for both small-medium enterprises and non-ML domain experts. This has led to the emergence of several platforms such as Google Cloud AutoML [5], Microsoft's AutomatedML [8], and H2O Driverless AI [6] with the promise to automate the end-to-end ML workflow without any human-in-the-loop. Since ML prediction accuracy is the most critical in AutoML environments, many works have studied the automation and impact of algorithm selection, hyperparameter search, and optimization heuristics on ML [34, 37]. However, little attention has been paid to assessing the significance of data preparation (prep) for ML.

Data prep for ML remains particularly challenging on structured data. It involves manual grunt work that is both tedious and time-consuming. Even AutoML users are often asked to manually perform many data prep steps before using their platforms [3]. Surveys of AutoML users have repeatedly identified such challenges in conducting data prep [28, 62]. One common issue that they often encounter is duplicates in the columns that are *Categorical*, which assumes mutually exclusive values from a known finite set.

Consider a simplified dataset to be used for a common ML classification task, *Customers Churn prediction* in Table 1. Duplicates,

CustId	*Name*	*Age*	*Gender*	*State*	*Country*	*Contract*	*Churn*
101	John	42	Male	California	US	monthly	'Y'
102	Jerry	29	Mail	CA	United states	Month-to -month	'N'

Table 1: Customers data used for ML Churn Prediction.

categories referring to the same real-world object occur in many *Categorical* columns such as *Gender*, *State*, *Country*, and *Contract*. Note that *Name* is not *Categorical* since it offers no discriminative power and cannot be generalized for ML. The presence of duplicates can potentially dilute the signal strength that one can extract from the column for ML. Thus, an ML practitioner would often deduplicate the categories before applying ML. Moreover, AutoML platforms often suggest users manually inspect *Categorical* columns and consolidate duplicates whenever they arise, as part of their guidelines for obtaining an accurate model [2]. Note that deduplication of the *Categorical* column is different from entity deduplication, which involves disambiguating entities given the full tuples from two tables [41, 63]. In contrast, *Categorical* duplicates are resolved at a column-level with no extra metadata.

Considering the laborious nature of the deduplication task, we ask: *Is deduplication effort even worthwhile for ML? Is it always needed regardless of the employed Categorical encoding scheme? Do duplicates impact both extremes of the ML model family equivalently?* We take a step towards answering these questions by developing an in-depth scientific understanding of the importance of deduplication for ML. Our objectives are two-fold. (1) Perform an extensive empirical study to measure the impact of duplicates on ML and distill the findings into actionable insights for handling duplicates. This can help ML practitioners decide when and how to prioritise their effort in cleaning duplicates. Moreover, this can enable AutoML platform builders design better deduplication workflows. (2) Present critical artifacts that can help advance the science of building AutoML platforms by providing researchers an apparatus to tackle open questions in this direction.

Our Approach. We identify that the impact on ML accuracy in presence of duplicates can be characterized with several confounders such as duplication properties (e.g., # duplicates per entity), training data properties (e.g., # available training examples), *Categorical* encoding method (e.g., *One-hot*), and ML model family (e.g., high variance model). Figure 1 details these confounders. Considering this, we make three component contributions to covers our goals. (1) We produce a labeled data to study how real-world duplicates arise. This helps us set up the duplication properties in the simulation study. More importantly, this can serve as a valuable artifact for the community to address many important open questions such as automating deduplication itself. (2) We phenomenalise the impact on ML accuracy by empirically benchmarking with real-world data containing duplicates in presence of multiple

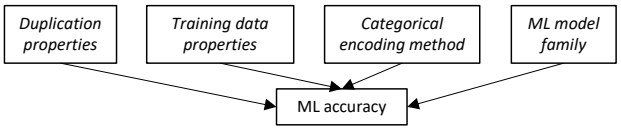

Figure 1: High-level summarization of confounders impacting ML accuracy in presence of duplicates.

confounders. (3) The significance of each confounder is hard to discern when all confounders act together. Thus, we use simulation study to disentangle the impact with all confounders and explain the phenomenon discretely. We explain each component below.

1. Our Labeled Data. To understand the behavior of *Categorical* duplicates, we first ask several important questions: *How do duplicates manifest themselves in the real-world columns? Do they happen often? How much can the domain size of the Categoricals be reduced with deduplication?* We address them by creating the first hand-labeled dataset where true entities within a *Categorical* column are annotated with corresponding duplicates. Our dataset includes 1248 string *Categorical* columns from 217 raw CSV files. The labeling process took about 120 man-hours across 5 months.

The utility of our labeled dataset is two-fold. (1) Get an average-case and worst-case estimate of several properties of duplicates such as their occurrences, cardinalities, and the fraction of entities that are diluted with duplicates. This enables us to control the synthetic duplication process that is real-world representative in the simulation study. (2) Provide a methodological approach to help address several critical open questions such as benchmarking and automating the deduplication task itself. For instance, this can benefit the data cleaning community which is often limited with synthetic datasets to evaluate their methods for a lack of annotated real data such as this, albeit for different cleaning operations [26].

2. Downstream Benchmark Suite. We create a benchmark suite of 14 real-world datasets to empirically benchmark the impact of duplicates on the downstream ML classifiers. We study this to make empirical observations in the two extremes of (1) ML model family in terms of bias-variance tradeoff with Logistic Regression and Random Forest. (2) Feature space representation with *Categorical* encoding schemes, *One-hot* (and raw *String*-based encoding for tree learners), and *Similarity* where the representation of duplicate with respect to the true entity in the feature space are orthogonal and similar respectively. (3) Data regime in terms of the training examples per category value. (4) Amount of duplication in terms of the parameters that characterize duplicates. (5) Relevancy of the *Categorical* column that has duplicates for the underlying ML task.

3. Synthetic Simulation Study. We perform a Monte Carlo-style simulation study to intricately study how six different confounders such as the fraction of entities with duplicates and data regime affect ML accuracy. We embed two different true distributions that present two extremes of how Logistic Regression and Random Forest fit the data. The duplicates are then introduced with a process that is informed by the understanding of how they occur within our labeled data. Here, we have two objectives. (1) Confirm the validity of the observations we make with the downstream benchmark suite. (2) Disentangle and characterize the effect of duplicates with multiple confounders individually to make the impact interpretable. Moreover, we expose a critical shortcoming of the commonly used *Categorical* encodings, *One-hot* and *String* when duplicates arise during the deployment in the test set but not during training.

Empirical Evaluation. An empirical comparison of our downstream benchmark suite reveals that deduplication can often improve the ML accuracy significantly, e.g., the impact with duplicates on Logistic Regression and Random Forest using *One-hot* encoding is significant (more than 1% accuracy) on 6 and 11 datasets (out of 14) respectively. Thus, the linear model gets impacted much less than the high-capacity model. Moreover, we find that *Similarity* encoding is more robust than other encodings to tolerate duplicates. Overall, we make *eight* such observations on the significance of the confounders with the downstream benchmark suite. We confirm all of them with simulation study. Specifically, we find that the presence of duplicates can often cause extra overfitting, which reduces as the number of training examples per category rises. Moreover, we provide explanations and insights into how ML models behave with duplicates. For instance, we find that Logistic Regression is (perhaps counter-intuitively) more robust to duplicates than Random Forest because although duplicates increase feature dimensionality of *Categoricals*, Logistic model often just ignores the extra dimensions by setting their weights close to 0.

Takeaways. Finally, we distill our empirical analysis into a handful of actionable takeaways for ML practitioners and AutoML platform builders. Our work opens up several new research directions at the intersection of ML theory, data management, and ML system design. This includes designing accurate methods for deduplication, quantifying the effect of duplicates with VC dimension-based bounds, and guiding AutoML platforms to effectively clean most impactful columns. Finally, we open source our entire benchmark including our labeled data, downstream suite, and simulation framework.

In summary, our work makes the following key contributions.

1. **A new benchmark dataset.** To the best of our knowledge, this is the first work to curate the first labeled dataset where entities of *Catgeoricals* are annotated with duplicates. We present several insights and takeaways that characterizes how duplicates exhibit themselves.
2. **Empirical benchmarking to understand the significance of deduplication on ML.** Our curated downstream benchmark suite containing "in-the-wild" datasets enables us to point out cases where ML accuracy may or may not benefit with deduplication.
3. **Characterization of confounders with simulation study.** Our comprehensive simulation study can disentangle and explain the impact of various confounders on how duplicates affect ML accuracy.
4. **Utility of our study.** We present several practical insights and takeaways for ML practitioners and AutoML platform designers. We identify several open questions for further research in this direction and how our labeled data can be a key enabler to address them. Also, we open source our benchmark to enable more community-driven contributions.

Symbol	Meaning
|D|	Domain size of the column with duplicates
|E|	Domain size of the deduplicated column
C_i	i^{th} category value in the column; $1 \leq i \leq \|D\|$
O(A)	Sum of occurrences of all categories present in set A; A ⊆ C
E_j	j^{th} real-world entity; $1 \leq j \leq \|E\|$
F_k	k^{th} real-world entity that has at least 1 duplicate; $F \subseteq E, 1 \leq k \leq \|F\|$
G_k	Set of duplicate values for k^{th} entity; $1 \leq k \leq \|F\|$
|T|	% of entities that have at least 1 duplicate: (|F|*100)/|E|
|R|	% reduction in domain size with deduplication: (1-|E|/|D|)*100

Table 2: Notations used in this paper.

2 PRELIMINARIES

2.1 Assumptions and Scope

We focus on structured data, which can be stored with any filesystem and any data format. We focus on the ML classification setting. We call the ML model to be trained over the data as the "downstream model." Note that our goal is not to study any upstream data sourcing or modeling processes that generate *Categorical* duplicates. We focus on understanding how duplicates manifest themselves in the real-world and how they impact the performance of the downstream models. Specifically, we study them in the context of string *nominal Categorical* features, which do not have a notion of ordering among its values. Note that a *Categorical* feature contains mutually exclusive values from a known finite domain set. In contrast, *Text* or *Sentence* type features can take arbitrary string values. Thus, generic open domain addresses or person names are not *Categorical*. We study duplicates arising in *Categorical* column, which is not the actual target for the prediction task.

2.2 Definitions

We define terms and notations to formally study the behavior of *Categorical* duplicates. Table 2 lists our notations, which we use in the rest of this paper. Table 3 explains our terms with an example.

Definition (Category). Given a *Categorical* column with a domain size of $|D|$, a Category set $C = \{C_1, C_2, ..., C_{|D|}\}$ contains all unique domain values occurring in the column. Each distinct value in the column is defined as "category." For Table 3 example, $C = \{$*New York, NY, new york, California, Ca, Wisconsin*$\}$.

Definition (Entity). An Entity set E contains a subset of *Categories* from C that conceptually refers to different real-world objects. Every category from set C can be mapped to a unique entity from set E. In Table 3 example, there are three unique real-world state objects, i.e., $E = \{$*New York, California, Wisconsin*$\}$. Note that entities are defined at a conceptual level; thus, referring to New York as new York or NY is identical. But for ease of exposition, we assume the category that most frequently represents an entity in the column to be the true entity.

Definition (Duplicate). If there exists more than one category for the same entity, then the entity contains a duplicate. Let $F \subseteq E$ such that the entities in F contain at least one duplicate. We define a Duplicate set G for every entity in F such that $G_k (1 \leq k \leq |F|)$ represents a set of duplicate values. Number of duplicates per entity is referred to as "Duplicate Size" $|G_k| (1 \leq k \leq |F|)$, e.g., $F_1 =$ *California*, $G_1 = \{Ca\}$ and $F_2 =$ *New York*, $G_2 = \{$*new york, NY*$\}$.

Category set C_i ($1 \leq i \leq \|D\|$)		Occurrence of Category O({C_i})	Entity set E_j ($1 \leq j \leq \|E\|$)	
New York	C_1	60	New York	E_1
NY	C_2	30		
new york	C_3	10		
California	C_4	70	California	E_2
Ca	C_5	30		
Wisconsin	C_6	100	Wisconsin	E_3

Table 3: A simplified example to illustrate our notions with *State* column categories.

Definition (Category Deduplication). *Category Deduplication* is the task of mapping categories from set C to an entity from set E. More formally, we define the mapping function as $I : C_i \rightarrow E_j$, where $C_i \in C, E_j \in E, 1 \leq i \leq |D|, 1 \leq j \leq |E|$. The new column after the assignment is called the *deduplicated* column. The category set C and entity set E of the *deduplicated* column are identical.

Definition (Occurrence). Given a category C_i mapped to an entity E_j, we define Occurrence (or percentage Occurrence) of category C_i as percentage of times E_j is represented by C_i in the column. For instance, whenever real-world *New York* entity occurs, 30% and 10% of the times it is represented by the duplicates *NY* and *new york* respectively. *New York* is referred to as the entity since it occurs more than *NY* and *new york*.

We define the *Occurrence function* as $O : A \rightarrow B$. Here, the input A is a subset $A \subseteq C$ such that all categories of the subset map to a unique entity from set E, i.e., $I(A_1) = I(A_2) = ... = I(A_{|A|})$. The output B is the sum of occurrence values for all categories present in the input set. $O(A) = O(A_1) + O(A_2) + ... + O(A_{|A|})$. For instance, $O(\{C1\}) = 60, O(\{C2, C3\}) = 40$, and $O(\{C1, C4\}) =$ Undefined.

Definition (Column Relevancy). Let *Acc(X)* be the % classification accuracy obtained by the ML model with table X as input. *Relevancy* of a column Z in the table X is defined as $Acc(W \cup Z) - Acc(W)$, where W is all the columns except Z in the table X. This quantifies the predictive power of Z for the downstream task.

3 OUR LABELED DATASET

We create a labeled dataset of *Categorical* columns where *Entities* in each column is annotated with their duplicates whenever present. This enables us to understand how real-world duplicates manifest themselves and what do the sets E, F, G, and O look like. We now discuss how this dataset is created, the types of real-world duplicates present, and our dataset analysis with stats and important insights into the behavior of duplicates.

3.1 Data Sources

A previous work constructed a large real-world dataset of 9921 columns manually annotated with a standardized 9-class label vocabulary of ML feature types [54]. The classes include feature types such as *Numeric*, *Categorical*, *Datetime*, *Sentence*, and *Not-Generalizable* (e.g., primary keys). We use their dataset to obtain raw CSV files that contain at least one string *Categorical* column in

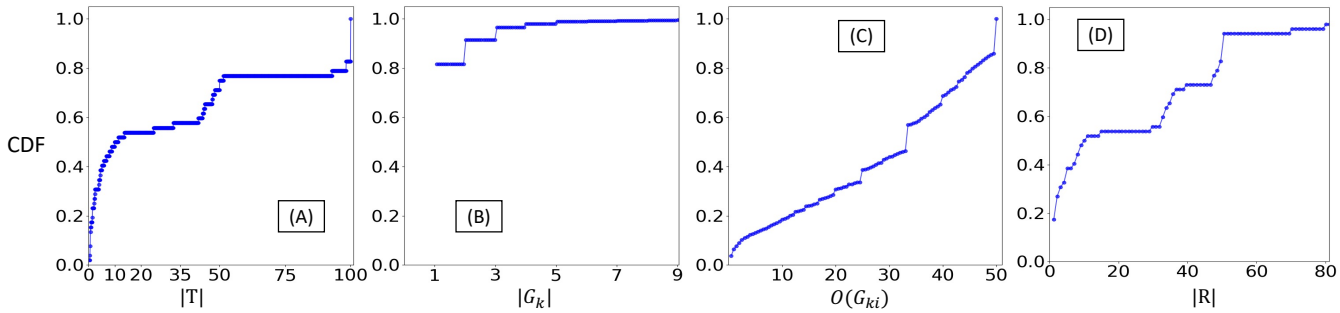

Figure 2: CDF over all *Categorical* columns with at least 1 duplicate on (A) $|T|$. (B) Duplicate sizes ($|G_k|$, $1 \leq k \leq |F|$). The maximum duplicate set size is 148. (C) Duplicate set occurrences ($O(G_{ki}), 1 \leq k \leq |F|, 1 \leq i \leq |G_k|$). (D) $|R|$.

Duplication Types		Column name	*Category* Examples
1	Capitalization	*Country*	*"United States" , "united States"*
2	Misspellings	Gender	"Male" , "Mail" , "Make" , "msle"
3	Abbreviation	State	"California" , "CA"
		topic_key	"asl" , "american sign language"
		preparer_title	"Senior Counsel" , "Sr. Counsel"
4	Difference of Special Characters	City	"New York" , " New York, "
		Colour	"Black/Blue" , "Black-Blue"
5	Different Ordering	Colour	"GoldWhite" , "WhiteGold" , "Gold/White"
6	Synonyms	Gender	"Female" , "Woman"
		Venue	"Festival Theatre" , "Festival Theater"
		Province	"Chalkidiki" , "Halkidiki"
7	Presence of Extra Information	City	"Houston" , "Houston TX" , "Houston TX 77055"
8	Different grammar	Colour	"triColor" , "tricolored"
		Venue	"Auditorium" , "TheAuditorium"

Table 4: Duplication types with examples from our labeled data

order to be inspected manually for duplicates. Overall, we obtain 1248 string *Categorical* columns spanning over 217 raw CSV files.

3.2 Labeling Process

Among the *Categorical* columns we collected, we do not know which columns contain duplicates beforehand. This necessitates us to manually scan through all the 1248 *Categorical* columns and look for duplicates in them. We follow the below process at a column-level to reduce the cognitive load of labeling. For every *Categorical* column, we use an Excel spreadsheet to enumerate its category set along with the count of times each category appears in the column. Before scanning the category set, we sort the categories by their appearance count in descending order and their values in lexicographic order. This helps up catch the true entities early on in the file. Recall that we call the category that most frequently represents a real-world object the true entity. As we scan the category set, we annotate duplicates with their corresponding entities in the column. Thus, we construct sets E, F, G, and O for all the columns. *The entire labeling process took roughly 120 man-hours across 5 months.*

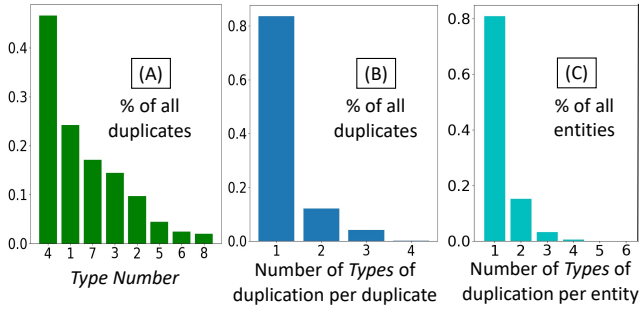

Figure 3: Histogram plots to illustrate how duplication types (from Table 4) arise across all columns in all files. x and y denote x-axis and y-axis respectively. (A) y% duplicates have duplication of at least one x *Type Number* (B) y% duplicates have x different duplication types within (C) y% entities have duplicates with x different duplication types within.

3.3 Types of Duplicates

We find that there exist *eight* types of duplication from our labeled dataset. We present these types with examples in Table 4. The differences shown are relative to the representation of the true entity. We now clarify some of the types. *Type 4* denotes the difference of any non-alphanumeric special characters including comma, period, and white spaces. *Type 5* denotes different ordering within multi-valued categories. *Type 8* categories have either a common stem/lemma, presence of stopwords, or a common singular representation. Note that a duplicate can have duplication of multiple types and an entity can have numerous duplicates, each belonging to multiple types, e.g., given F_1 = New York and $G_1 = \{$*new-york., NY*$\}$, "*new-york.*" has both *Type 1* and *4* duplication, and the entity *New York* has duplicates with duplication of *Type 1, 2,* and *4.*

3.4 Data Statistics and Takeaways

We annotated 56573 entities across all 1248 string *Categorical* columns. We find 4% of those entities have the presence of at least one duplicate with a total of 3475 duplicates. Thus, a large fraction of the entities are already clean without any duplication. Overall, 52 columns from 33 raw CSV files have the presence of at least one duplicate. There are three parameters that quantify the amount of duplication within a column. (1) Fraction of entities that have at least one duplicate ($|T|$). (2) Duplicate set size for all entities present in the column (set $G_k, 1 \leq k \leq |F|$). (3) Duplicate occurrences $O(G_{ki}), 1 \leq k \leq |F|, 1 \leq i \leq |G_k|$. Figure 2 plots the cumulative

distribution function (CDF) of these parameters over all columns in our labeled dataset that has at least 1 duplicate. We also report CDF of the % reduction in domain size of the columns with deduplication. We summarize the presented results with key takeaways below.

Takeaways. (1) We first explain the worst-case scenario that can arise due to duplicates. This is critical to understand because in a real-world deployment setting, say with AutoML platforms, the developers have to factor in the tail percentiles when building end-to-end automation pipelines. We find that almost 17% of the columns that have duplicates have them in all of their entities! Furthermore, 7% of duplicates across all columns occur at 50%, i.e., the representation of the duplicate and the true entity are identical. Additionally, 1% of all entities (with a non-empty G set) have more than five duplicates. However, the presence of more than 10 duplicates per entity is quite unlikely (less than 0.5%). Finally, deduplication can reduce the domain size of the *Categorical* column substantially by up to 99%. Overall, we find that whenever duplicates arise in the *Categorical* columns, they can occur quite often. As a result, they can potentially dilute the strength of the signal that one can extract from the column for ML.

(2) We now discuss the presence of duplicates with the average case scenario. We find that whenever an entity is diluted with duplicates, almost 90% of the time they have one or two duplicates! Duplicate set sizes follow a long-tail distribution, most entities have small duplicate set sizes and very few entities have a lot of duplicates. This can make catching duplicates and deduplicating them particularly challenging, as they can go unnoticed by ML practitioners. Moreover, the occurrence of duplicates ($O(G_{ki})$, $1 \leq k \leq |F|$, $1 \leq i \leq |G_k|$) approximately follows a uniform distribution, i.e., all occurrence values up to 50% are roughly equally likely. Finally, $|T|$ values of 10-35% fall close to the median.

(3) We present how different duplication types (from Table 4) are represented in our labeled data in Figure 3. We find that the *Difference of Special Characters* and *Capitalization* issues represents the two most common types of duplicates. On the other hand, *Synonyms* and *Grammar* issues are relatively less common. Moreover, whenever duplicates exist, 17% of the time they belong to more than one duplication type (maximum observed is 4 duplication types). Also, 19% of entities have duplicates that can be mapped to multiple types (maximum observed is 6 duplication types).

4 DOWNSTREAM BENCHMARK

We now empirically study the impact of category duplicates on the downstream ML tasks. We curate a downstream benchmark suite of 14 real-world datasets, each containing a column with duplicates. We use this to empirically evaluate and compare three commonly used *Categorical* encoding schemes both with and without the presence of duplicates. Finally, we make several important observations on the different confounders that impact the relationship of *Categorical* duplicates with downstream ML classifiers.

4.1 Models and Encodings

We showcase two extremes of the bias-variance tradeoff in terms of downstream ML model family [35]: (1) High bias and low variance approach with L2-regularized Logistic regression. (2) Low bias and high variance approach with Random Forest.

We encode the *Categorical* columns with three commonly used encoding schemes: (1) *One-hot* encoding (*OHE*). (2) *String* encoding (*StrE*). (3) *Similarity* encoding (*SimE*) [9]. *OHE* is the standard approach to encode nominal *Categoricals* as it follows their two properties. (1) Each category is orthogonal to one another. (2) Pairwise distance between any two categories is identical. Random Forest with *OHE* performs binary splits on the data. Random Forest can also handle raw "stringified" *Categorical* values by performing set-based splits on the data. We refer to this as *StrE*. Note that *StrE* is not applicable for the Logistic model, since it cannot handle raw string values. Both *OHE* and *StrE* assume that the *Categorical* domain is closed with ML inference, i.e., new categories in the test not seen during training are handled by mapping them to a special category, "*Others*." *SimE* takes into account the morphological variations between the categories. Given a closed *Categorical* domain set $\{x_1, x_2, ..., x_k\}$, it encodes x_i with a k sized feature vector $V = [Sim(x_i, x_1), Sim(x_i, x_2), ..., Sim(x_i, x_k)]$, where $Sim(x, y)$ is a similarity metric defined as the dice-coefficient over n-gram (n ranges from 2 to 4) strings [27]. The feature vector V can be computed even for any new categories in test unseen during training.

4.2 Real Datasets and Labeling

We collect 14 datasets from real open-source data portals such as Chicago city, New York state, California state, Pittsburgh health, Open source mental illness project, and real data surveys from FiveThirtyEight, EveryDayData, and Kaggle. Specifically, we obtain the following datasets: Midwest Survey [18], Mental Health [20], Relocated Vehicles [15], Health Sciences [13], Salaries [17], TSM Habitat [12], EU IT [22], Halloween [19], Utility [16], Mid or Feed [23], Wifi [24], Etailing [25], San Francisco [21], and Building Violations [14]. Each dataset has a column with duplicates. We manually deduplicate them by following the labeling process described in Section 3.2. Table 5 presents the downstream dataset statistics with different confounders that can impact the downstream performance. Note that the notion of data regime is relative to the complexity of the prediction task. We simplify the data regime as the number of training examples available per category value. We binned data regime and amount of duplication into low, medium, and high with hand-picked thresholds to better visualize and interpret the results.

4.3 Methodology and Setup

We partition each dataset into an 80:20 split of train and held-out test set. We perform 5-fold cross-validation of the train set and use a fourth of the examples in the train set for hyper-parameter search. Due to space constraints and CFP restrictions of not citing a technical report, we will describe the grids for hyper-parameter tuning in a technical report after paper acceptance.

Experimental Setup. We use CloudLab [31] with custom OpenStack profile running Ubuntu 18.04, 10 Intel Xeon cores, and 160GB of RAM. We use python scikit-learn [51], H2O [7], and SimilarityEncoder [9] packages to implement *OHE*, *StrE*, and *SimE* respectively.

4.4 Results

Table 6 shows the end-to-end comparison of the ML downstream models built with different encoding schemes. As an example, on

Datasets	|X|	|A|	|Y|	Amount of Duplication							Data Regime		% P
				|T|	$\|G_k\|$		$O(G_{ki})$		|D|	|R|	|XC|	|XC| after dedup (Increase w.r.t Raw)	
					median	max	median	max					
Midwest Survey	2778	29	9	33.1	2	95	4	50	1008	64	2.5	6.5 (2.6x)	23.6
Mental Health	1260	27	5	40	3.5	15	2.3	33.3	49	69.4	23.2	81.2 (3.5x)	25.3
Relocated Vehicles	3263	20	4	33.2	1	8	20	50	1097	35.8	2.5	3.8 (1.5x)	14.9
Health Sciences	238	101	4	36.4	2	14	6	50	56	60.7	3.6	8.3 (2.3x)	26.4
Salaries	1655	18	8	24	1	10	25	50	647	29.2	1.8	2.2 (1.2x)	10.9
TSM Habitat	2823	48	19	11	1	3	25	50	912	11.4	2.6	2.9 (1.1x)	14.6
EU IT	1253	23	5	24	1.5	8	12.5	50	256	34.8	3.9	5.9 (1.5x)	19.5
Halloween	292	55	6	31.3	2	12	11.1	50	163	50.9	1.5	3 (2x)	22.8
Utility	4574	13	95	38.4	1	2	20	44.6	199	30.7	16.2	24.3 (1.5x)	6.2
Mid or Feed	1006	78	5	21.4	6	15	0.8	33.3	37	62.2	20.6	59.7 (2.9x)	24.3
Wifi	98	9	2	30.3	2.5	12	12.5	50	69	52.2	1.3	2.5 (1.9x)	26.1
Etailing	439	44	5	47.8	4	10	5.9	50	71	67.6	5.3	14.3 (2.7x)	28.7
San Francisco	148654	13	2	10.7	1	2	25	50	2159	9.8	46.3	50.9 (1.1x)	3.2
Building Violations	22012	17	6	51	2	23	4.8	50	270	63	53.7	145 (2.7x)	4.4

Table 5: Statistics of the column containing duplicates in our downstream benchmark datasets. $|X|$, $|A|$, and $|Y|$ are the total number of examples, number of columns, and number of target classes in the given dataset respectively. $|XC|$ denotes the number of training examples (averaged over 5 folds) per category of the set C. P is the fraction of $|E|$ (averaged across 5-folds) that has at least 1 duplicate being mapped to "*Others*" category in the validation set with *OHE* and *StrE*. We use colors green, blue, red with hand-picked thresholds to visually present and better interpret the cases where the amount of duplication is low ($|R|$<25), moderate ($|R|$ >25 & <50), and high ($|R|$ >50) respectively. We use the following thresholds with the same colors to better interpret the data regime: low ($|XC|$<5), moderate ($|XC|$ >5 & <25), and high ($|XC|$ >25). Note that the data regime moves up with deduplication as category set size has shrunk.

Midwest Survey, Random Forest with *OHE* of *Categoricals* delivers a 9-class classification accuracy of 49.1% on the *Raw* dataset. Cleaning its duplicates (*Deduped*) lead to an 11.5% lift in accuracy relative to the *Raw*. Table 7 shows summary statistics of how the different encoding schemes perform with the two ML models and also relative to one another on the 14 datasets. Finally, we present the generalization performance of the ML classifiers with the overfitting gap (difference between train and validation set accuracies) on both *Raw* and *Deduped* in Table 8. We summarize our results with *eight* important observations below.

O1. We find that there exist several downstream cases where *Deduped* improves the accuracy of ML over *Raw* for any encoding scheme. For instance, the delta increase in accuracy with *Deduped* on *OHE* Random Forest is of median 1.6% and up to 11.5% compared to *Raw* (across 14 datasets). Moreover, the delta increase in accuracy is of median 0.6% and a maximum of 9.4% for Logistic Regression.

O2. Delta increases in accuracies with *Deduped* are typically higher with Random Forest than Logistic. The number of datasets where we see >1% delta increases in % accuracy with Random Forest *OHE*, *StrE*, and *SimE* are 11, 8, and 6 respectively. In contrast, deduplication helps significantly on Logistic with *OHE* and *SimE* on 6 and 5 datasets respectively. Thus, the linear model Logistic Regression is more robust to duplicates than the high-capacity Random Forest.

O3. Deduped helps Random Forest using *OHE* the most, *StrE* the second most, and *SimE* the least. We can observe this trend from Table 7. Interestingly, *SimE* improves the ML performance in just ~40% downstream cases. This is because the similarity representation already benefits from the presence of the duplicates. Thus, further lift in accuracy with *Deduped* is low.

O4. Deduplication reduces the overfitting gap for both ML models (From Table 8). Thus, deduplication can reduce the variance component of the test error and improve the generalization ability of the classifiers. Since Random Forest is more prone to overfitting than Logistic, the lifts in accuracies with *Deduped* are more significant for Random Forest. This also explains the observation *O2.*

O5. If the magnitude of overfitting gap on *Raw* is insignificant (less than 1%), then the amount of possible reduction in overfitting with *Deduped* is also small. Thus, it's not worthwhile to deduplicate if the overfitting gap on *Raw* is already low, to begin with. We observe this will all the datasets where the overfitting gap is close to 1%, e.g., *San Francisco* and *Building Violations.*

O6. Deduplication increases the column *Relevancy* for both ML models. In other words, the column becomes more predictive for the downstream tasks after deduplication. This is because the amount of increase in *Relevancy* of the column with *Deduped* also quantifies the accuracy lift with *Deduped.*

O7. The accuracy lifts with *Deduped* on the two models are more significant when the column has high *Relevancy* unless the high-data regime (a large number of training examples per category). Thus, if a column has already high *Relevancy* on *Raw*, to begin with,

Dataset	Logistic Regression						Random Forest							
	Relevancy OHE		*OHE*		*SimE*		*Relevancy OHE*		*OHE*		*StrE*		*SimE*	
	Raw	*Deduped*	*Raw*	*Deduped*	*Raw*	*Deduped*	*Raw*	*Deduped*	*Raw*	*Deduped*	*Raw*	*Deduped*	*Raw*	*Deduped*
Midwest Survey	10.6	+9.4	57.2	+9.4	66.7	+2.1	4.6	+11.5	49.1	+11.5	59.2	+10	64.9	+4.4
Mental Health	-1.3	+1.3	46.9	+1.3	46.3	+0.6	0.2	+1.1	47.9	+1.1	47.8	-0.1	47.4	-1.7
Relocated Vehicles	18.1	+4	82.9	+4	88.4	+0.4	6.1	+3	72.5	+3	81.3	+4.1	88.3	-0.1
Health Sciences	-1.3	+0.9	58.7	+0.9	60	+1.8	-1.8	+2.2	53.3	+2.2	61.8	+0	60	-2.7
Salaries	-1.1	+0.1	30.4	+0.2	32.4	-1.3	-1	+1.7	64.7	+1.7	69.6	+1.3	94.6	+0.4
TSM Habitat	0	+0	50.7	+0	50.7	+0	4.8	+0.4	71.2	+0.4	84.1	+1.4	71.2	+0.4
EU IT	0	+0	29.1	+0	29.1	+0	2.1	+1.2	41.2	+1.2	43.6	-0.6	47.8	+4
Halloween	0.4	+3.4	42.6	+3.4	49.8	+1.1	-1.9	+1.5	40	+1.5	36.2	+1.5	34.7	-4.9
Utility	1.4	-0.2	42.4	-0.2	43	+0.3	-6.7	+1.4	58.8	+1.4	46.3	+1.2	43.2	+1.4
Mid or Feed	0	+1.7	40.5	+1.7	41.5	-1.2	-1	+2.5	40.2	+2.5	35.7	-0.2	36.2	+1.8
Wifi	-2.1	+1.1	64.2	+1.1	58.9	+8.4	-1.1	+5.3	60	+5.3	57.9	+4.2	50.5	+3.2
Etailing	0.7	-0.5	41.1	-0.5	38.9	+1.8	-2.5	+2	40	+2	44.5	+1.1	38.2	+3
San Francisco	26.9	-0.1	86	-0.1	85.5	+0	24.3	+0.1	83.4	+0.1	83.9	-0.3	86	+0
Building Violations	0.1	+0	91.6	+0	91.9	+0	0	-0.1	97.5	-0.1	97.3	+0.1	97.6	+0

Table 6: Classification accuracy comparison of downstream models with different *Categorical* encoding schemes on *Raw* (column with duplicates) vs. *Deduped* (deduplicated column) data. Accuracy results for *Deduped* are shown relative to the *Raw* as delta drop in % accuracy. Green, blue, and red colors denote cases where the *Deduped* accuracy relative to *Raw* is significantly higher, comparable, and significantly lower (error tolerance of 1%) respectively.

	Logistic Regression		Random Forest		
	OHE	*SimE*	*OHE*	*StrE*	*SimE*
% lift in accuracy with *Deduped*					
Mean	1.5	1	2.4	1.7	0.7
Median	0.6	0.4	1.6	1.2	0.4
75th percentile	1.6	1.6	2.4	1.5	2.7
Max	9.4	8.4	11.5	10	4.4
# downstream datasets where					
>1% lift in accuracy on *Deduped*	6	5	11	8	6
Best performing encoding on *Raw*	6*	10*	5	3	6
Best performing encoding on *Deduped*	5*	11*	5	3	6

Table 7: Summary statistics to illustrate the impact of deduplication with the two ML models using different encoding schemes on 14 downstream datasets. $*$ denotes two cases where both encoding schemes perform the best.

Dataset	Logistic Regression (*OHE*)		Random Forest (*OHE*)	
	Raw	*Deduped*	*Raw*	*Deduped*
Midwest Survey	24.4	-9.4	50.7	-14.2
Mental Health	11.7	-3.5	42.3	-7.2
Relocated Vehicles	17	-4.1	27.3	-3.1
Health Sciences	9.3	-5.9	35	-8.1
Salaries	1.9	+0.2	34.6	-1
TSM Habitat	1.9	-0	28	-0
EU IT	1.2	-0	53.1	-6.6
Halloween	38.3	-3.5	50.9	-5.8
Utility	0.7	-0.3	41.2	-1.4
Mid or Feed	34.2	-12.8	58.4	-1.1
Wifi	11.1	-2.1	26.2	+1.3
Etailing	41.2	-7.7	54.4	-1.6
San Francisco	0.5	-0	-0.2	-0
Building Violations	0.2	+0.1	1.8	-0.1

Table 8: Delta drop in % overfitting gap in accuracy (difference between training and validation accuracy) of the ML models with *OHE*. The overfitting gap for *Deduped* is shown relative to the *Raw*.

it may be worthwhile conservatively to deduplicate the column, e.g., *Relocated Vehicles* and *Midwest Survey*.

O8. High-data regime is robust to the impact of duplicates than the low-/moderate-data regime, regardless of the amount of duplication. Thus, even a high amount of duplication (i.e., large $|R|$ values) has a negligible impact in the high-data regime, e.g., *Building Violations* has a massive 63% reduction in domain size due to deduplication, but there exist a large number of training examples per category value. We do not see any lift in accuracy with deduplication.

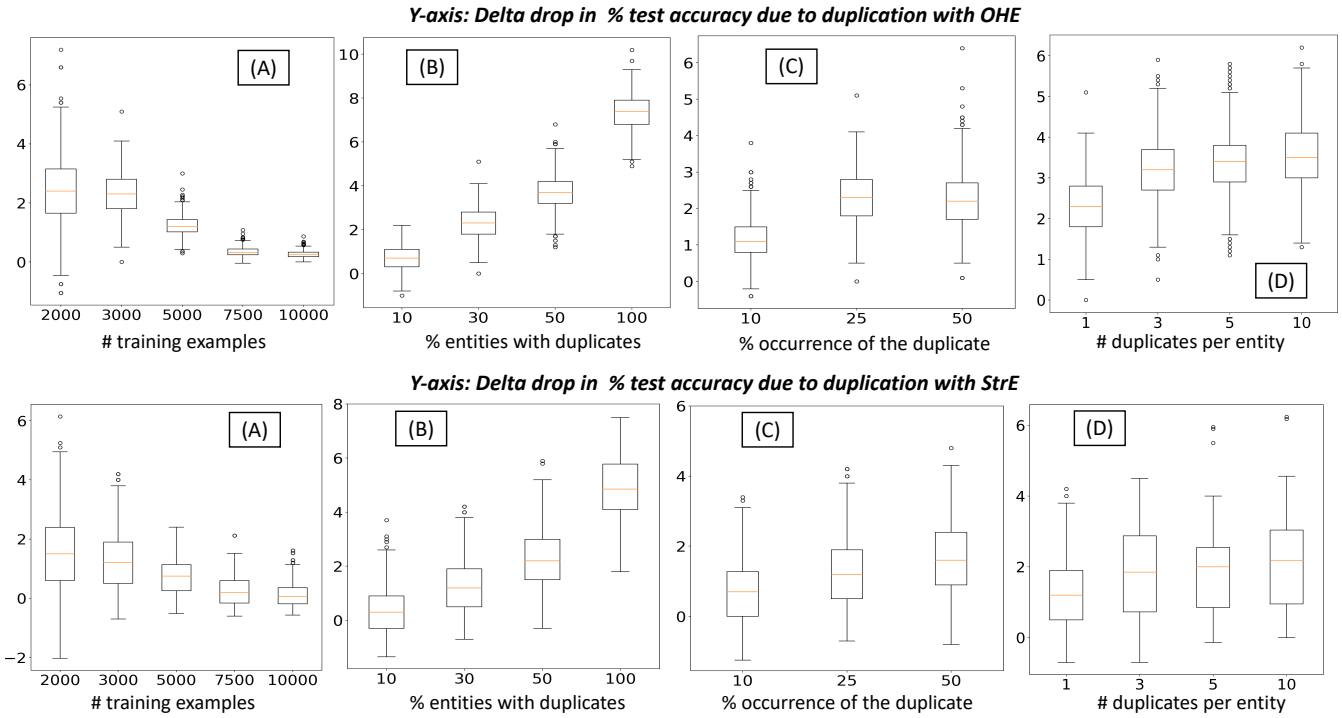

Figure 4: `AllX` simulation scenario results for Random Forest with *OHE* and *StrE*. $|A|$ is preset to 3. (A) Vary $|X|_t$, while fixing $(|T|, O(G_k), |G_k|) = (30, 25, 1)$, for all $k, 1 \leq k \leq |F|$. (B) Vary $|T|$, while fixing $(|X|_t, O(G_k), |G_k|) = (3000, 25, 1)$. (C) Vary $O(G_k)$, while fixing $(|X|_t, |T|, |G_k|) = (3000, 30, 1)$. (D) Vary $|G_k|$, while fixing $(|T|, |X|_t, O(G_k)) = (30, 3000, 25)$.

There exist *six* data-dependent confounders that can potentially impact the downstream performance. (1) Three parameters that control the amount of duplication: $|T|, |G_k|, O(G_{ki}), 1 \leq k \leq |F|, 1 \leq i \leq |G_k|$. (2) column *Relevancy*. (3) Amount of training data available per category value: $|XC|$. (4) Fraction of entities that have duplicate(s) being mapped to "Others" along with their occurrences in the unseen test set. Beyond our observations, there exists a non-trivial interaction of all *six* parameters that impact downstream ML. Thus, we use the simulation study in the next Section to disentangle them and study them separately.

5 IN-DEPTH SIMULATION STUDY

We now use simulation study to dive deeper into the impact of each confounder on the downstream ML models. We focus this study in the context of *OHE* and *StrE* and leave *SimE* to future work. This study helps us not only validate our empirical observations but also makes the significance of each confounder impacting ML more interpretable. Moreover, it reveals the limitations of the commonly used encoding schemes such as *OHE* and *StrE* when unseen duplicates arise in the test set.

Setup. There is one table X and target Y is boolean (domain size is 2). We include only the *Categorical* features in the table. We fix $|A|$ (number of *Categorical* columns) to 3. We set the entity set size of every columns to $|E| = 10$, i.e., all columns have a domain size of 10.

Data Synthesis. We set up a "true" distribution $P(X, Y)$ and sample examples in an independently and identically distributed manner. We study two different simulation scenarios: `AllX` and `Hyperplane`. These scenarios represent two opposite extremes of how Logistic and Random Forest fits the data. `AllX` represents a complex joint distribution where all features in X determine Y. Although high-capacity model like Random Forest can recover this with rule-based splits, Logistic can extremely underfit. `Hyperplane` represents a distribution where the data is simply separable with a hyperplane. This distribution is well-suited for Logistic but represents a bad-case scenario for Random Forest that requires many numbers of splits to recover the true concept.

We sample $|X|$ number of total examples, where the examples for training, validation, and test are split in 60:20:20 ratio. We then introduce synthetic duplicates in one of the columns of the table in different ways. We vary the *six* confounders one at a time and study their impact on ML accuracy along with how they trend as the parameter is varied. We generate 100 different (clean) training datasets and 10 different dirty datasets for every clean one. We measure the average test accuracy and the average overfitting gap of Logistic Regression and Random Forest obtained from these 1000 runs. We use a standard grid search for hyper-parameter tuning for them. We will describe the grids in the technical report.

5.1 Scenario AllX

Data generating process. We create a true distribution that maps all features in X to Y. The exact sampling process is as follows. (1) Construct a conditional probability table (CPT) with entries for all possible values of $\mathbf{X}$ from 1 to $|E|$. We then assign $P(Y = 0|X)$ to either 0 or 1 with a random coin toss. (2) Construct $|X|$ tuples of $\mathbf{X}$ by sampling values uniform randomly from $|E|$. (3) We assign

Y values to tuples of **X** by looking up into their respective CPT entry. (4) We perform the train, validation, and test split of this clean dataset and obtain the binary-class classification accuracy of the ML models on the test split, $Acc(W \bigcup Z')$. Here, Z' is the clean categorical column where duplicates are to be introduced and W are all the other columns without any duplicates.

Duplication process. We introduce duplicates in column Z' of the clean dataset in the following fashion. (1) Fix fraction of entities to be diluted with duplicates, e.g., $|T| = 30$ (2) Form set F (set of entities that are to be diluted with duplicates) by sampling uniformly randomly $|F|$ categories from $|E|$ (E_1 to $E_{|E|}$), e.g., $F = \{E_3, E_5, E_8\}$. (3) For every entity in set F, fix the duplicate set size $|G_k|, \forall 1 \leq k \leq |F|$, e.g., $|G_k| = 1, \forall 1 \leq k \leq 3$. We assume that all entities have identical duplicate set sizes. We relax this assumption in Section 5.1.3. (4) Given $|G_k|$, we form the set G_k by introducing duplicates, e.g., $G_1 = \{E_3 - duplicate_1\}, G_2 = \{E_5 - duplicate_1\}, G_3 = \{E_8 - duplicate_1\}$. (5) Fix $(O(G_k), 1 \leq k \leq |F|, 1 \leq i \leq |G_k|)$. For every duplicate value d in G_k, set occurrence $O(d) = O(G_k)/|G_k|$, i.e., we assume that all the duplicates representing an entity are equally likely to occur. We relax this assumption in Section 5.1.3. (6) We perform the same train, validation, and test split of the resulting dataset as obtained in step 4 of the data generating process. We obtain the test accuracy of the ML models on the dirty dataset, $Acc(W \bigcup Z)$. Here, Z is the *Categorical* column with duplicates.

Results. We use our labeled dataset to pick appropriate values for the confounders such that we can showcase an average and worst-case scenario. We vary all confounder parameters one at a time while fixing the rest. We find that Logistic Regression's accuracy on the clean dataset is close to random. Since this joint distribution is too complex to classify with a linear hyperplane, we cannot draw any reliable conclusions about the effect of duplication on Logistic. Instead, we present the results on Random Forest in-depth below. As each parameter is varied, we confirm the trends and observations made with them in *italics.*

5.1.1. Varying the data regime. Figure 4 (A) presents the delta drop in % test accuracy with duplication relative to the ground truth clean dataset on Random Forest as the number of training examples are varied. We find that with the rise in the number of training examples, the delta drop in accuracy due to duplication decreases. With just 3 training examples per CPT entry ($|X|_t = 3k$ and total entries in CPT are 1000), the presence of duplicates can cause a drop of median 2% and up to 5.1% accuracy with *OHE*. With 10 training examples, the median and max drops in accuracy due to duplicates with *OHE* are 0.4% and 2.1% respectively. This confirms our observation on the downstream benchmark suite: *A higher data regime is more robust to duplication than a lower data regime. The same trend holds with StrE as well.* Moreover, increasing the amount of duplication for a high data regime ($|X|_t = 10k$) has a marginal impact on the ML accuracy. *Thus, even high duplication has a marginal impact in the high-data regime.* Due to space constraints, we will present the corresponding accuracy plots of the effects of duplicates in a high-data regime in the technical report.

5.1.2. Varying the parameters that control the amount of duplication. Figure 4 (B-D) shows how different duplication parameters influence downstream Random Forest. We notice a clear trend:

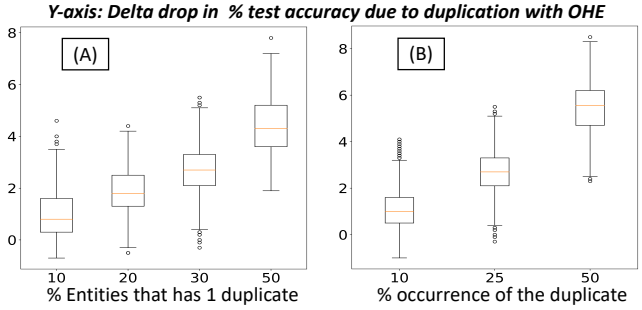

Figure 5: `A11X` simulation scenario results on Random Forest with *OHE*. Only test set is diluted with duplicates. $|A| = 3$**, (A) Vary** $|T|$**, while fixing** $(|X|_t, O(G_k), |G_k|) = (3000, 25, 1)$**. (B) Vary** $O(G_k)$**, while fixing (**$|X|_t, |T|, |G_k|) = (3000, 30, 1)$**.**

the drop in accuracy with Random Forest rises with the increase in any of the three duplication controlling parameters, % entities with duplicates ($|T|$), duplicate occurrences ($O(G_k)$), and the number of duplicates per entity ($|G_k|$). As $|T|$, $O(G_k)$, and $|G_k|$ are each increased 5 folds, the magnitude increase in delta drop accuracy with duplicates using *OHE* are $3.9x$, $2.1x$, and $1.5x$ respectively. In contrast, the same magnitude increases using *StrE* are $5.5x$, $2.3x$, and $1.8x$ respectively. Thus, among the three duplication parameters, $|T|$ has the most drastic effect on Random Forest. The effects of the increase in $|G_k|$ are less pronounced because all other parameters including $O(G_k)$ are kept fixed. Thus, there exist more duplicates for the same occurrence. *Interestingly, we find from Figure 4 that StrE is more robust to duplicates than OHE regardless of the parameter being varied, as the delta drop in accuracy with StrE is comparatively lower, although significant in high duplication cases.*

5.1.3. Introducing skewness in the duplication parameters. Until now, we assumed that all entities in F have identical duplicate set sizes $|G_k|$ and all duplicates in G_k are equally likely to occur. From our labeled dataset, we find that most entities have small duplicate set sizes and only a few entities have many duplicates. Also, some duplicates in the same set G_k are more likely to occur than others. Thus, we relax these two assumptions and include distributions in $|G_k|$ and $O(G_k)$ that can better represent the duplication process. We approximate $|G_k|(1 \leq k \leq |F|)$ with a long-tail Zipfian distribution and $O(G_k)$ with a needle-and-thread distribution. We alter our duplication process to capture skewness in $O(G_k)$ and $O(G_k)$, varying one at a time. *The overarching conclusion from this analysis is that none of our results or takeaways change or get invalidated with this new setup.* Due to space constraints, we defer further discussions to the technical report.

5.1.4. Varying properties of duplicates being mapped to "*Others.*" We now study how duplicates that do not arise in the train set but are present in the test set (say during deployment) can impact the downstream ML. We modify and repeat our duplication process on just the test set but we keep the train set intact. Thus, we do not encounter any duplicates while training, but during the test we introduce just one duplicate that gets mapped to "*Others.*" Figure 5 presents the results on Random Forest with *OHE* where $|T|$ and $O(G_k)$ are varied. *We find that the delta drop in accuracies with all parameters are even more higher than the corresponding delta*

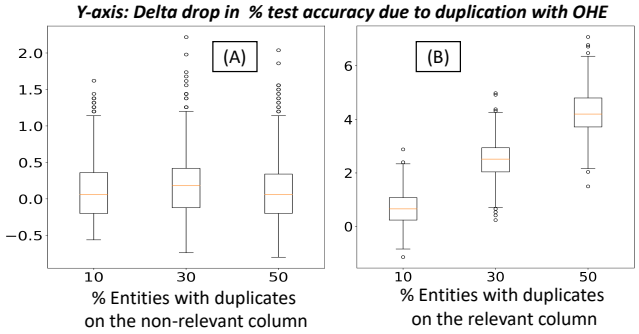

Figure 6: `AllX` simulation scenario results on Random Forest with *OHE*. We set $|A| = 4$ and vary $|T|$, while fixing $(|X|_t, zp) = (5000, 2)$. Duplicates are introduced on the column with (A) non-positive *Relevancy* (noisy column) and (B) high *Relevancy* (predictive column).

drops when both train and test set were duplicated (Figure 4 (B)-(C)). We notice the trends with *SimE* being similar. This study suggests that the presence of unwarranted duplicates during the test can cause downstream ML to suffer significantly.

5.1.5. Varying column *Relevancy*. So far, we have used all the columns in the dataset as part of the CPT. Thus, all columns have high *Relevancy*. We now study low vs. high *Relevancy* setting with a slight twist in our simulation. We introduce an additional noisy column in the clean dataset: All except one column participates in CPT. That is, we use the same data generating process as discussed earlier, but we use all columns except one to identify Y. Thus, we now have the presence of both high and low *Relevancy* columns in the dataset. We now introduce duplicates in both types of columns one at a time. Figure 6 presents the results. *We find that duplication on a highly relevant column has a significant adverse impact on Random Forest performance. On the other hand, the impact is negligible when duplicates are introduced over the noisy column.* Even increasing the amount of duplication creates no impact on accuracy with the low relevancy column.

5.2 Scenario Hyperplane

Data generating process. We set up distribution with a true hyperplane to separates the classes. (1) We define and fix the normal vector of the hyperplane with weights, $W_i, 1 <= i <= |A|$. Each weight W_i with cardinality $|E|$ is randomly sampled from a list of non-zero integers $i \in= \{-5, ... -1, 1, ..., 5\}$ without replacement. Note that the integer weights are chosen to make the distance calculation simpler in step (3). The trends of our results do not change even if the weights are chosen from real number uniform distribution. (2) Construct $|X|$ tuples of $\mathbf{X}$ by sampling values uniformly randomly from $|E|$. Thus, with fixed weights, the hyperplane over *One-hot* encoded example feature vectors X_i $(1 <= i <= |A|)$ is given by $\sum_{i=1}^{i=|A|} W_i \cdot X_i = 0$. (3) Examples for which $\sum_{i=1}^{i=|A|} W_i \cdot X_i >= 0$ are labeled positive $(Y = 0)$ and the remaining examples are labeled negative $(Y = 1)$. This generates the true dataset where all columns have high *Relevancy*. We introduce duplicates in them by following the same duplication process as Section 5.1.

Results. Figure 7 shows the delta drop in test accuracy due to duplicates with both models using *OHE*. We find that Logistic Regression is robust to the presence of duplicates as it shows no drop in accuracy even if all entities are diluted with duplicates. In contrast, duplicates affect Random Forest significantly in a high-duplication regime, when more than 50% entities are diluted. We explain this interesting behavior in Section 5.3. We vary other confounders such as skew in the duplication parameters, the fraction of entities being mapped to "*Others,*" and column *Relevancy*. We also introduce and vary noise in the data generating process by randomly flipping the label for a fraction of examples within a distance threshold. Due to space constraints, we will present the plots in the technical report. We briefly summarize the key takeaways below. We confirm the same trends that we saw with `AllX` scenario on Random Forest with the presence of duplicates. (1) Marginal impact in a higher data regime setting. (2) Larger impact with a high *Relevancy* column than a column that provides no signal for the prediction. (3) Significant impact with unseen duplicates, although they remain applicable a higher-duplication setting. (4) *StrE* is more robust than *OHE*. However, we see no impact of duplicates on Logistic regression whatsoever, regardless of the parameter being varied.

5.3 Explanations

We now intuitively explain the general behavior of the ML classifiers in presence of duplicates on the two simulation settings with *OHE*. We check the generalization ability of the ML classifiers with the overfitting gap. Figure 8 presents the overfitting gap results of Random Forest with *OHE* on the `AllX` scenario. We find that the delta drop in accuracy (Figure 4) closely follows the increase in the overfitting gap due to duplicates. That is, the increase in overfitting or variance with duplicates explains the accuracy drop we see. *Thus, duplicates can negatively impact the generalization capability of Random Forest.* However, as the number of available training examples rises, the amount of overfitting subsides. This explains our trends in the high-data regime.

Now with the `Hyperplane` setting, Logistic exhibits no amount of extra overfitting with duplicates. This is because the VC dimension of the logistic model is linear in the number of features. As the dimensionality of the feature space expands with duplicates, the VC dimension of Logistic expands. We get an expanded logistic hypothesis space with duplication that is a superset of the true logistic hypothesis space. Thus, a larger hypothesis space can potentially lead to more variance unless the true concept is simple enough to recover in an expanded feature space. We check the weights of the hyperplane learned with Logistic in presence of duplicates where a higher weight indicates higher importance. We find that the absolute weights of duplicate features are often close to zero. *This suggests that the hyperplane can learn the true concept by completely ignoring the extra dimensions. Thus, the variance does not rise.*

In contrast, Random Forest with *OHE* makes many binary splits on the clean data to recover the hyperplane, causing the tree to fully grow to the restricted height. Chances of further overfitting with duplicates are reduced with a limited height. Thus, we less significant drop in accuracy with duplicates on Random Forest in `Hyperplane` setting compared to `AllX`. *This also explains why a set-based split with StrE is more robust than binary splits with OHE as it allows to pack more category splits within the same tree height.*

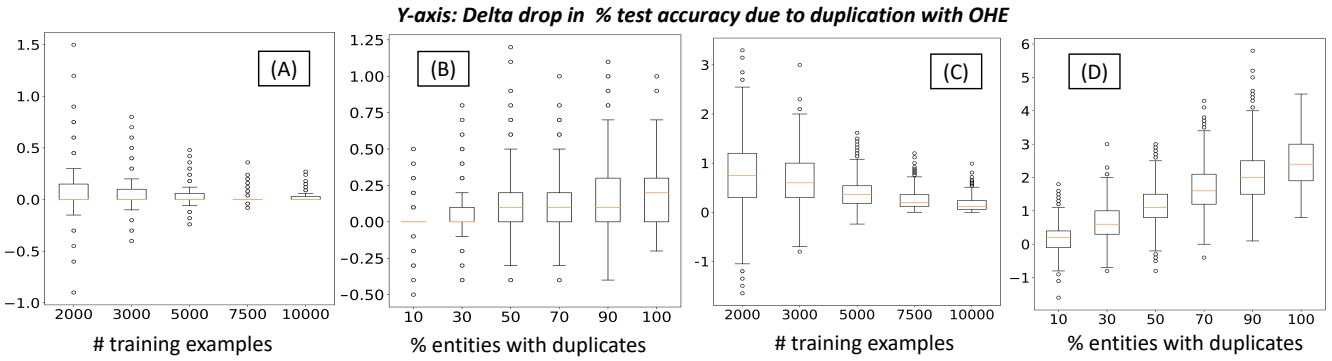

Figure 7: `Hyperplane` simulation scenario results with *OHE* on (A-B) Logistic Regression (C-D) Random Forest. (A,C) Vary $|X|_t$, while fixing $(|T|, O(G_k), |G_k|) = (30, 25, 1)$, for all $k, 1 \leq k \leq |F|$. (B,D) Vary $|T|$, while fixing $(|X|_t, O(G_k), |G_k|) = (3000, 25, 1)$.

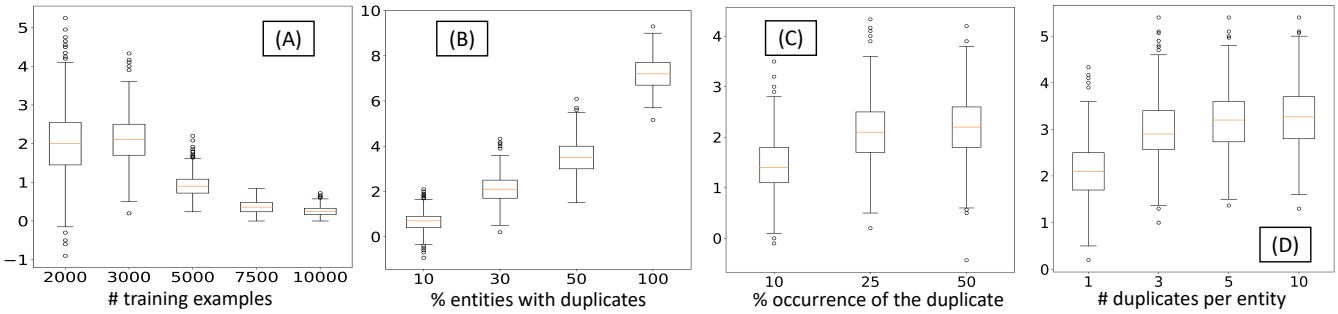

Figure 8: `AllX` simulation setting results on Random Forest using *OHE* (same setup as Figure 4).

6 DISCUSSION

6.1 Public Release

We release a public repository on GitHub with our entire benchmark suite [1]. This includes our labeled dataset of entities in the string *Categorical* columns annotated with their category duplicates, along with their raw CSV files. We also release the downstream benchmark suite with raw and deduped versions of all datasets, synthetic benchmarks, and the code to run them.

6.2 Takeaways

We find that the presence of duplicates can potentially impact downstream ML accuracy significantly. The amount of impact can be characterized by multiple confounders that interact in non-trivial ways. It is not always possible to disentangle the impact on downstream ML with each confounder individually. However, our analysis with downstream benchmark suite and simulation studies can provide insights into when cleaning effort would be more or less beneficial. The current practice among ML practitioners and AutoML platform developers to handle *Categorical* duplicates is largely ad hoc rule-based and completely oblivious to many confounders. We first give general guidelines and actionable insights to help them prioritise their deduplication effort and also potentially design better end-to-end automation pipelines respectively. We then lay out the critical open questions in this direction for researchers.

6.2.1 For ML practitioners and AutoML platform developers.

1. Make ML workflows less susceptible to the adverse performance impact of category duplicates. Logistic Regression is less prone to overfitting than Random Forest when duplicates arise. Moreover, *Similarity* encoding inherently exploits the presence of similarly valued duplicates in the *Categoricals*. This makes it significantly more robust from duplicates compared to *One-hot* and *String* encoding. Moreover, unseen duplicates that arise during the deployment phase can degrade ML performance with *OHE* or *StrE*. Overall, *Similarity* encoding and/or a linear model can be utilized by ML practitioners and AutoML developers if they desire to guard their pipelines against any adverse drop in ML performance from likely duplicates. Moreover, the impact of duplicates get mitigated in a higher-data regime compared to a low-data regime. Thus, whenever possible, one can consider getting more train data to offset their impact by trading off the runtime.

2. Track the overfitting gap of ML models. Deduplication can reduce the overfitting gap caused by duplicates on downstream ML. Thus, cleaning duplicates may not be worthwhile if the overfitting gap is already low on the raw dataset. Monitoring and presenting it as an auxiliary metric to the AutoML user can provide them with more confidence about the downstream performance.

3. Roughly estimate the impact of duplicates with synthetic simulation suite. Our synthetic simulation suite provides an empirical methodological infrastructure for understanding the category deduplication effect. Given an arbitrary dataset, it can create semi-synthetic variants of category duplicates by modeling them with various observed properties from our labeled data. Also, the

impact becomes more interpretable when confounders are disentangled, e.g., quantifying the impact of "*Others*" in a deployment setting with *OHE* (Section 5.1.4).

6.2.2 For Researchers. We discuss three major open questions for research that require contributions from the community.

1. Design accurate methods for deduplication. Although category duplicates can often impact ML accuracy substantially, none of the existing open source AutoML tools such as AutoGluon [32], AutoML Tables [4], and TransmogrifAI [10] support an automated deduplication workflow. Cleaning duplicates manually can be slow and frustrating for many users, especially non-technical lay users who were promised end-to-end automation of the entire ML workflow. Moreover, the present art of deduplicating categories is still largely manual and relies on ad hoc rule-based approaches. This may not suffice in an AutoML environment, where given arbitrary data, it is difficult to decide the number of rules to use, which rules to apply, and with what threshold to apply.

In contrast to rule-based approaches, our labeled dataset presents an alternative learning-based approach to automate deduplication. Moreover, this will lead to an objective assessment of the accuracy of automation. This opens up two new avenues for further research. (1) Capturing semantic-level characteristics of the categories with either designing features or with deep learning models like Siamese neural network [48]. (2) Data amplification through augmentation in slices where examples are under-represented. Table 4 can provide guidelines on how to synthesize potential duplicates. Existing weak supervision environments can augment them further [53, 59].

2. Define new benchmark tasks. In an AutoML production setting, there can be thousands of datasets with millions of features. One cannot possibly perform deduplication over all the columns to fix duplicates. Given a cleaning budget both in terms of accuracy and runtime, how to guide an AutoML platform to prioritize which column to clean? One can consider designing a coarse-grained binary classifier to help identify them, but how does a column-level featurization look like? Our labeled data can be leveraged and amplified at a column-level to assess the accuracy of any approaches.

3. Theoretical quantification. Our empirical study suggests that duplicates can increase variance since the hypothesis space of the model can grow. This opens up several research questions at the intersection of ML theory and data management: Is it possible to establish bounds on the increase in variance using VC-dimension theory [58]? Can we set up a decision rule to formally characterize when deduplication would be needed?

7 RELATED WORK

Empirically Studying the Impact on ML. CleanML [44] analyses the impact of many data cleaning steps on downstream ML classification tasks. However, they do not cover *Categorical* deduplication step. Open source AutoML benchmark [34] performs a comparative study of many AutoML tools in terms of the overall classification accuracy with their model selection routines. However, understanding any data prep step from downstream ML accuracy standpoint is not the focus of their work. A previous work [54] performed an objective benchmarking of a specific data prep step for ML, namely the feature type inference task. We build upon their open-sourced datasets but we study a completely different problem.

AutoML Platforms. Several AutoML libraries allow users to perform automated ML algorithm and hyper-parameter search without covering any data prep tasks [33, 49, 57]. Other AutoML platforms that offer (or claim to offer) end-to-end ML support such as Salesforce Einstein TransmogrifAI [10], Google AutoML Tables [5], and Amazon AutoGluon [32] do automate many data prep tasks. However, none of them handles *Categorical* duplicates. Instead, the users are asked to explicitly clean and remove inconsistencies in *Categorical* columns before using their platforms [3]. Our labeled data can lead to more contributions from the community to automate the deduplication task. Moreover, we believe that our downstream benchmark, our analysis with simulation studies, and takeaways are all valuable to improve their AutoML platforms.

Data Prep and Cleaning for ML. There exists numerous data prep tools such as rule-based tools [38], exploratory data analysis-based libraries [52], visual interfaces [11], and program synthesis-based tools [36, 39] to reduce user's manual grunt work effort and allow them to productively prepare their data for ML. In addition, Auto-insight generator tools [29, 30, 60] allow users to discover interesting statistical properties of a given dataset. Our work's insights can complement all these tools to reduce human time and effort and make their analysis more interpretable. Some works have studied human-in-the-loop data cleaning to improve ML accuracy and reduce user effort [42, 43]. However, they do not support a cleaning operation when *Categorical* duplicates arise. Our labeled data can spur more follow-up works in this general direction of automating and improving data prep for ML.

Entity Matching (EM). EM, the task of identifying whether records from two tables refer to the same real-world entity has received much attention with rule-based [50, 55, 56], learning-based [41, 45, 47, 63], semi-supervised [40], unsupervised [61], and active-learning [46] approaches. Our focus is on analyzing the impact of category duplicates and category deduplication on downstream ML. In the service of this goal, we offer new hand-labeled data, benchmarks, and simulations. We do not propose new techniques for EM or even category deduplication. Thus, prior work on EM is complementary to ours in terms of utility for AutoML platforms.

Note that all EM solutions have access to the entire feature vectors of the two tables. Thus, they operate at a tuple-level. On the other hand, category deduplication operates at a column-level of one table with no additional metadata, except the category strings. Admittedly, it is possible to view category deduplication as an extension of EM but doing so is non-trivial. This is because converting raw category strings to full entity feature vectors is unclear and ill-specified. We leave it to future work to automate category deduplication, including potentially extending past EM work.

REFERENCES

[1] Accessed September 5, 2021. Github Repository for studying the impact of Cleaning Category Duplicates on ML, https://github.com/anoncategdedup/CategDedupRepo.

[2] Accessed September 5, 2021. Google AutoML Tables Cleaning Duplicates User Guidelines, https://cloud.google.com/automl-tables/docs/data-best-practices#make_sure_your_categorical_features_are_accurate_and_clean.

[3] Accessed September 5, 2021. Google AutoML Tables Data Prep User Guidelines, https://cloud.google.com/automl-tables/docs/data-best-practices.

[4] Accessed September 5, 2021. Google AutoML Tables, https://cloud.google.com/automl-tables.

[5] Accessed September 5, 2021. Google Cloud AutoML, https://cloud.google.com/automl/.

[6] Accessed September 5, 2021. H2O Driverless AI, https://www.h2o.ai/products/h2o-driverless-ai/.

[7] Accessed September 5, 2021. H2o.AI, https://www.h2o.ai/.

[8] Accessed September 5, 2021. Microsoft AutoML, https://azure.microsoft.com/en-us/services/machine-learning/automatedml/.

[9] Accessed September 5, 2021. Similarity Encoder Library, https://github.com/dirty-cat/dirty_cat.

[10] Accessed September 5, 2021. TransmogrifAI: Automated Machine Learning for Structured Data, https://transmogrif.ai/.

[11] Accessed September 5, 2021. Trifacta: Data Wrangling Tools & Software, https://www.trifacta.com/.

[12] Accessed September 5, 2021. *https://data.ca.gov/dataset/tsm-habitat-rapid-assessment-survey-2016-ds28271*.

[13] Accessed September 5, 2021. *https://datacatalog.hsls.pitt.edu/dataset/77*.

[14] Accessed September 5, 2021. *https://data.cityofchicago.org/Buildings/Vacant-and-Abandoned-Buildings-Violations/kc9i-wq85*.

[15] Accessed September 5, 2021. *https://data.cityofchicago.org/Transportation/Relocated-Vehicles/5k2z-suxx*.

[16] Accessed September 5, 2021. *https://data.ny.gov/Energy-Environment/Utility-Company-Customer-Service-Response-Index-CS/w3b5-8aqf*.

[17] Accessed September 5, 2021. *https://everydaydata.co/2017/02/07/hacker-news-part-one.html*.

[18] Accessed September 5, 2021. *https://github.com/fivethirtyeight/data/tree/master/region-survey*.

[19] Accessed September 5, 2021. *https://maxcandocia.com/article/2018/Oct/22/trick-or-treating-ages/*.

[20] Accessed September 5, 2021. *https://osmihelp.org/research*.

[21] Accessed September 5, 2021. *https://transparentcalifornia.com/salaries/san-francisco/*.

[22] Accessed September 5, 2021. *https://www.asdcode.de/2021/01/it-salary-survey-december-2020.html*.

[23] Accessed September 5, 2021. *https://www.kaggle.com/definitelyliliput/rawscores*.

[24] Accessed September 5, 2021. *https://www.kaggle.com/mlomuscio/wifi-study*.

[25] Accessed September 5, 2021. *https://www.kaggle.com/pushpaltayal/etailing-customer-survey-in-india*.

[26] Ziawasch Abedjan, Xu Chu, Dong Deng, Raul Castro Fernandez, Ihab F. Ilyas, Mourad Ouzzani, Paolo Papotti, Michael Stonebraker, and Nan Tang. 2016. Detecting Data Errors: Where are we and what needs to be done? *Proc. VLDB Endow.* 9, 12 (2016), 993–1004. https://doi.org/10.14778/2994509.2994518

[27] Richard C. Angell, George E. Freund, and Peter Willett. 1983. Automatic Spelling Correction Using a Trigram Similarity Measure. *Inf. Process. Manag.* 19, 4 (1983), 255–261. https://doi.org/10.1016/0306-4573(83)90022-5

[28] Anamaria Crisan and Brittany Fiore-Gartland. 2021. Fits and Starts: Enterprise Use of AutoML and the Role of Humans in the Loop. In *CHI '21: CHI Conference on Human Factors in Computing Systems, Virtual Event / Yokohama, Japan, May 8-13, 2021*, Yoshifumi Kitamura, Aaron Quigley, Katherine Isbister, Takeo Igarashi, Pernille Bjørn, and Steven Mark Drucker (Eds.). ACM, 601:1–601:15. https://doi.org/10.1145/3411764.3445775

[29] Çagatay Demiralp, Peter J. Haas, Srinivasan Parthasarathy, and Tejaswini Pedapati. 2017. Foresight: Recommending Visual Insights. *Proc. VLDB Endow.* 10, 12 (2017), 1937–1940. https://doi.org/10.14778/3137765.3137813

[30] Rui Ding, Shi Han, Yong Xu, Haidong Zhang, and Dongmei Zhang. 2019. QuickInsights: Quick and Automatic Discovery of Insights from Multi-Dimensional Data. In *Proceedings of the 2019 International Conference on Management of Data, SIGMOD Conference 2019, Amsterdam, The Netherlands, June 30 - July 5, 2019*, Peter A. Boncz, Stefan Manegold, Anastasia Ailamaki, Amol Deshpande, and Tim Kraska (Eds.). ACM, 317–332. https://doi.org/10.1145/3299869.3314037

[31] Dmitry Duplyakin, Robert Ricci, Aleksander Maricq, Gary Wong, Jonathon Duerig, Eric Eide, Leigh Stoller, Mike Hibler, David Johnson, Kirk Webb, Aditya Akella, Kuangching Wang, Glenn Ricart, Larry Landweber, Chip Elliott, Michael Zink, Emmanuel Cecchet, Snigdhaswin Kar, and Prabodh Mishra. 2019. The Design and Operation of CloudLab. In *Proceedings of the USENIX Annual Technical Conference (ATC)*. 1–14. https://www.flux.utah.edu/paper/duplyakin-atc19

[32] Nick Erickson, Jonas Mueller, Alexander Shirkov, Hang Zhang, Pedro Larroy, Mu Li, and Alexander J. Smola. 2020. AutoGluon-Tabular: Robust and Accurate AutoML for Structured Data. *CoRR* abs/2003.06505 (2020). arXiv:2003.06505 https://arxiv.org/abs/2003.06505

[33] Matthias Feurer, Aaron Klein, Katharina Eggensperger, Jost Tobias Springenberg, Manuel Blum, and Frank Hutter. 2015. Efficient and Robust Automated Machine Learning. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, Corinna Cortes, Neil D. Lawrence, Daniel D. Lee, Masashi Sugiyama, and Roman Garnett (Eds.). 2962–2970. http://papers.nips.cc/paper/5872-efficient-and-robust-automated-machine-learning

[34] Pieter Gijsbers, Erin LeDell, Janek Thomas, Sébastien Poirier, Bernd Bischl, and Joaquin Vanschoren. 2019. An Open Source AutoML Benchmark. *arXiv preprint arXiv:1907.00909* (2019).

[35] Trevor Hastie, Jerome H. Friedman, and Robert Tibshirani. 2001. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer. https://doi.org/10.1007/978-0-387-21606-5

[36] Yeye He, Xu Chu, Kris Ganjam, Yudian Zheng, Vivek Narasayya, and Surajit Chaudhuri. 2018. Transform-Data-by-Example (TDE): An Extensible Search Engine for Data Transformations. *Proceedings of the VLDB Endowment* 11, 10 (2018), 1165–1177.

[37] Frank Hutter, Lars Kotthoff, and Joaquin Vanschoren (Eds.). 2019. *Automated Machine Learning - Methods, Systems, Challenges*. Springer. https://doi.org/10.1007/978-3-030-05318-5

[38] Nick Hynes, D Sculley, and Michael Terry. 2017. The Data Linter: Lightweight, Automated Sanity Checking for ML Data Sets. In *NIPS MLSys Workshop*.

[39] Zhongjun Jin, Michael R Anderson, Michael Cafarella, and Hosagrahar V Jagadish. 2017. Foofah: A Programming-By-Example System for Synthesizing Data Transformation Programs. In *Proceedings of the 2017 ACM International Conference on Management of Data*. 1607–1610.

[40] Mayank Kejriwal and Daniel P. Miranker. 2015. Semi-supervised Instance Matching Using Boosted Classifiers. In *The Semantic Web. Latest Advances and New Domains - 12th European Semantic Web Conference, ESWC 2015, Portoroz, Slovenia, May 31 - June 4, 2015. Proceedings (Lecture Notes in Computer Science, Vol. 9088)*, Fabien Gandon, Marta Sabou, Harald Sack, Claudia d'Amato, Philippe Cudré-Mauroux, and Antoine Zimmermann (Eds.). Springer, 388–402. https://doi.org/10.1007/978-3-319-18818-8_24

[41] Pradap Venkatramanan Konda. 2018. *Magellan: Toward building entity matching management systems*. The University of Wisconsin-Madison.

[42] Sanjay Krishnan, Michael J. Franklin, Ken Goldberg, Jiannan Wang, and Eugene Wu. 2016. ActiveClean: An Interactive Data Cleaning Framework For Modern Machine Learning. In *Proceedings of the 2016 International Conference on Management of Data, SIGMOD Conference 2016, San Francisco, CA, USA, June 26 - July 01, 2016*, Fatma Özcan, Georgia Koutrika, and Sam Madden (Eds.). ACM, 2117–2120. https://doi.org/10.1145/2882903.2899409

[43] Sanjay Krishnan, Michael J. Franklin, Ken Goldberg, and Eugene Wu. 2017. BoostClean: Automated Error Detection and Repair for Machine Learning. *CoRR* abs/1711.01299 (2017). arXiv:1711.01299 http://arxiv.org/abs/1711.01299

[44] Peng Li, Xi Rao, Jennifer Blase, Yue Zhang, Xu Chu, and Ce Zhang. 2019. CleanML: A Benchmark for Joint Data Cleaning and Machine Learning [Experiments and Analysis]. *arXiv preprint arXiv:1904.09483* (2019).

[45] Yuliang Li, Jinfeng Li, Yoshihiko Suhara, AnHai Doan, and Wang-Chiew Tan. 2020. Deep Entity Matching with Pre-Trained Language Models. *Proc. VLDB Endow.* 14, 1 (2020), 50–60. https://doi.org/10.14778/3421424.3421431

[46] Venkata Vamsikrishna Meduri, Lucian Popa, Prithviraj Sen, and Mohamed Sarwat. 2020. A Comprehensive Benchmark Framework for Active Learning Methods in Entity Matching. In *Proceedings of the 2020 International Conference on Management of Data, SIGMOD Conference 2020, online conference [Portland, OR, USA], June 14-19, 2020*, David Maier, Rachel Pottinger, AnHai Doan, Wang-Chiew Tan, Abdussalam Alawini, and Hung Q. Ngo (Eds.). ACM, 1133–1147. https://doi.org/10.1145/3318464.3380597

[47] Sidharth Mudgal, Han Li, Theodoros Rekatsinas, AnHai Doan, Youngchoon Park, Ganesh Krishnan, Rohit Deep, Esteban Arcaute, and Vijay Raghavendra. 2018. Deep Learning for Entity Matching: A Design Space Exploration. In *Proceedings of the 2018 International Conference on Management of Data, SIGMOD Conference 2018, Houston, TX, USA, June 10-15, 2018*, Gautam Das, Christopher M. Jermaine, and Philip A. Bernstein (Eds.). ACM, 19–34. https://doi.org/10.1145/3183713.3196926

[48] Paul Neculoiu, Maarten Versteegh, and Mihai Rotaru. 2016. Learning Text Similarity with Siamese Recurrent Networks. In *Proceedings of the 1st Workshop on Representation Learning for NLP, Rep4NLP@ACL 2016, Berlin, Germany, August 11, 2016*, Phil Blunsom, Kyunghyun Cho, Shay B. Cohen, Edward Grefenstette, Karl Moritz Hermann, Laura Rimell, Jason Weston, and Scott Wen-tau Yih (Eds.). Association for Computational Linguistics, 148–157. https://doi.org/10.18653/v1/W16-1617

[49] Randal S. Olson, Nathan Bartley, Ryan J. Urbanowicz, and Jason H. Moore. 2016. Evaluation of a Tree-based Pipeline Optimization Tool for Automating Data Science. In *Proceedings of the 2016 on Genetic and Evolutionary Computation Conference, Denver, CO, USA, July 20 - 24, 2016*, Tobias Friedrich, Frank Neumann, and Andrew M. Sutton (Eds.). ACM, 485–492. https://doi.org/10.1145/2908812.2908918

[50] Fatemah Panahi, Wentao Wu, AnHai Doan, and Jeffrey F. Naughton. 2017. Towards Interactive Debugging of Rule-based Entity Matching. In *Proceedings of the 20th International Conference on Extending Database Technology, EDBT 2017, Venice, Italy, March 21-24, 2017*, Volker Markl, Salvatore Orlando, Bernhard Mitschang, Periklis Andritsos, Kai-Uwe Sattler, and Sebastian Breß (Eds.). OpenProceedings.org, 354–365. https://doi.org/10.5441/002/edbt.2017.32

[51] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. 2011. Scikit-learn: Machine learning in Python. *the*

Journal of machine Learning research 12 (2011), 2825–2830.

[52] Jinglin Peng, Weiyuan Wu, Brandon Lockhart, Song Bian, Jing Nathan Yan, Linghao Xu, Zhixuan Chi, Jeffrey M. Rzeszotarski, and Jiannan Wang. 2021. DataPrep.EDA: Task-Centric Exploratory Data Analysis for Statistical Modeling in Python. In *Proceedings of the 2021 International Conference on Management of Data (SIGMOD '21), June 20–25, 2021, Virtual Event, China.*

[53] Alexander Ratner, Stephen H. Bach, Henry R. Ehrenberg, Jason Alan Fries, Sen Wu, and Christopher Ré. 2017. Snorkel: Rapid Training Data Creation with Weak Supervision. *Proc. VLDB Endow.* 11, 3 (2017), 269–282. https://doi.org/10.14778/3157794.3157797

[54] Vraj Shah, Jonathan Lacanlale, Premanand Kumar, Kevin Yang, and Arun Kumar. 2021. Towards Benchmarking Feature Type Inference for AutoML Platforms. In *Proceedings of the 2021 International Conference on Management of Data*. 1584–1596.

[55] Rohit Singh, Venkata Vamsikrishna Meduri, Ahmed K. Elmagarmid, Samuel Madden, Paolo Papotti, Jorge-Arnulfo Quiané-Ruiz, Armando Solar-Lezama, and Nan Tang. 2017. Generating Concise Entity Matching Rules. In *Proceedings of the 2017 ACM International Conference on Management of Data, SIGMOD Conference 2017, Chicago, IL, USA, May 14-19, 2017*, Semih Salihoglu, Wenchao Zhou, Rada Chirkova, Jun Yang, and Dan Suciu (Eds.). ACM, 1635–1638. https://doi.org/10.1145/3035918.3058739

[56] Rohit Singh, Venkata Vamsikrishna Meduri, Ahmed K. Elmagarmid, Samuel Madden, Paolo Papotti, Jorge-Arnulfo Quiané-Ruiz, Armando Solar-Lezama, and Nan Tang. 2017. Synthesizing Entity Matching Rules by Examples. *Proc. VLDB Endow.* 11, 2 (2017), 189–202. https://doi.org/10.14778/3149193.3149199

[57] Chris Thornton, Frank Hutter, Holger H Hoos, and Kevin Leyton-Brown. 2013. Auto-WEKA: Combined Selection and Hyperparameter Optimization of Classification Algorithms. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 847–855.

[58] Vladimir Naumovich Vapnik. 2000. *The Nature of Statistical Learning Theory, Second Edition.* Springer.

[59] Paroma Varma and Christopher Ré. 2018. Snuba: Automating Weak Supervision to Label Training Data. *Proc. VLDB Endow.* 12, 3 (2018), 223–236. https://doi.org/10.14778/3291264.3291268

[60] Manasi Vartak, Sajjadur Rahman, Samuel Madden, Aditya G. Parameswaran, and Neoklis Polyzotis. 2015. SEEDB: Efficient Data-Driven Visualization Recommendations to Support Visual Analytics. *Proc. VLDB Endow.* 8, 13 (2015), 2182–2193. https://doi.org/10.14778/2831360.2831371

[61] Renzhi Wu, Sanya Chaba, Saurabh Sawlani, Xu Chu, and Saravanan Thirumuruganathan. 2020. ZeroER: Entity Resolution using Zero Labeled Examples. In *Proceedings of the 2020 International Conference on Management of Data, SIGMOD Conference 2020, online conference [Portland, OR, USA], June 14-19, 2020*, David Maier, Rachel Pottinger, AnHai Doan, Wang-Chiew Tan, Abdussalam Alawini, and Hung Q. Ngo (Eds.). ACM, 1149–1164. https://doi.org/10.1145/3318464.3389743

[62] Doris Xin, Eva Yiwei Wu, Doris Jung Lin Lee, Niloufar Salehi, and Aditya G. Parameswaran. 2021. Whither AutoML? Understanding the Role of Automation in Machine Learning Workflows. In *CHI '21: CHI Conference on Human Factors in Computing Systems, Virtual Event / Yokohama, Japan, May 8-13, 2021*, Yoshifumi Kitamura, Aaron Quigley, Katherine Isbister, Takeo Igarashi, Pernille Bjørn, and Steven Mark Drucker (Eds.). ACM, 83:1–83:16. https://doi.org/10.1145/3411764.3445306

[63] Chen Zhao and Yeye He. 2019. Auto-EM: End-to-end Fuzzy Entity-Matching using Pre-trained Deep Models and Transfer Learning. In *The World Wide Web Conference, WWW 2019, San Francisco, CA, USA, May 13-17, 2019*, Ling Liu, Ryen W. White, Amin Mantrach, Fabrizio Silvestri, Julian J. McAuley, Ricardo Baeza-Yates, and Leila Zia (Eds.). ACM, 2413–2424. https://doi.org/10.1145/3308558.3313578