How do Categorical Duplicates Affect ML? A New Benchmark and Empirical Analyses [Experiment, Analysis & Benchmark]

Vraj Shah†
IBM Research
vraj@ibm.com

Thomas Parashos
California State University,Northridge
thomasjparashos@gmail.com

Arun Kumar
University of California, San Diego
arunkk@eng.ucsd.edu

ABSTRACT

The tedious grunt work involved in data preparation (prep) before ML reduces ML user productivity. 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 such *Categorical* duplicates impact 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 classification with a three-pronged approach. We first study how *Categorical* duplicates exhibit themselves by creating a labeled dataset of 1262 *Categorical* columns. We then curate a downstream benchmark suite of 14 real-world datasets to make observations on the effect of *Categorical* duplicates on three popular classifiers. We finally use simulation studies to validate our observations. We find that Logistic Regression and *Similarity* encoding are more robust to *Categorical* duplicates than two *One-hot* encoded high-capacity classifiers. We provide actionable takeaways that can potentially help AutoML developers to build better platforms and ML practitioners to reduce grunt work. While some of the presented insights have remained folklore for practitioners, our work presents the first systematic scientific study to analyze the impact of *Categorical* duplicates on ML and put this on an empirically rigorous footing. Our work presents novel data artifacts and benchmarks, as well as novel empirical analyses to spur more research on this topic.

PVLDB Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://github.com/anon-categdups/CategDedupRepo.

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 [47, 51]. Also, recently there is a growing interest for studying how data prep specifically affects downstream ML [59, 65, 66].

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 [4]. Surveys of AutoML users have repeatedly identified such challenges in conducting data prep [41, 81]. One issue that they often encounter is duplicates in the columns that are *Categorical*, which assumes mutually exclusive values from a known finite set. This can require significant manual effort to fix duplicates even if a single *Categorical* column contains them in a data file.

Table 1: A simplified dataset for ML Churn Prediction.

Name	*Gender*	*State*	*Title*	*Contract*	*Zipcode*	*Density*	*MostCommon Crime_Zipcode*	*Churn*
John	Male	California	sr. Scientist	monthly	93449	727	BURG	'Y'
Jerry	Mail	CA	snr scientist	Month-to-month	91042	563	burglary	'N'

Consider a dataset to be used for a common ML classification task in Table 1. Duplicates, categories referring to the same real-world object, occur in many *Categorical* columns such as *Gender*, *State*, *Title*, and *Contract*. Note that *Name* is not *Categorical* since it offers no discriminative power and cannot be generalized for ML. The presence of duplicates within a *Categorical* column can potentially dilute signal strength that one can extract for ML. Thus, an ML practitioner would often deduplicate categories before ML. We further discuss the conundrum for an ML practitioner in Section 3. Even, AutoML platforms often suggest users to manually inspect *Categoricals* and consolidate duplicates whenever they arise, as part of their guidelines for obtaining an accurate model [3]. This can involve non-trivial amount of deduplication effort at a *Categorical* column-level as duplicates can arise as misspellings, abbreviations, and synonyms, even within the same column. Note that this problem is related but complementary to entity deduplication issues studied in the data cleaning literature, as we explain in Section 2.1.

In this paper, we ask: *How do Categorical duplicates impact commonly used ML classifiers? Is category deduplication effort even worthwhile for ML? Is it always needed regardless of the employed Categorical encoding scheme?* We take a step towards answering these questions by developing an in-depth scientific understanding of the importance of category deduplication for ML classification (henceforth referred to as "ML"). Our objectives are two-fold. (1) Perform an extensive empirical study to measure the impact of *Categorical* duplicates on ML and distill the findings into actionable insights for handling them. This can help ML practitioners decide when and how to prioritise their cleaning effort. Moreover, this can enable AutoML platform builders design better ML 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.

† Work done at University of California, San Diego and at IBM Research.

Approach Overview. We identify that the impact on ML accuracy in presence of *Categorical* duplicates can be characterized with several confounders such as their duplication properties, training data properties, *Categorical* encoding, and ML model. Considering this, we make three-part contributions to cover our goals. (1) We produce labeled dataset to study how real-world *Categorical* duplicates arise. (2) We create a downstream benchmark suite to phenomenalise the impact on ML on real-world data containing *Categorical* duplicates with multiple confounders. (3) Significance of each confounder is hard to discern when all confounders act together. We use simulation study to disentangle the impact with each confounder and explain the phenomenon discretely.

Empirical Evaluation. An empirical comparison of our downstream benchmark reveals that category deduplication can often improve the ML accuracy significantly, e.g., the median lifts in % accuracies due to category deduplication on *One-hot* encoded Logistic Regression (LR), Random Forest (RF), and artificial neural network (ANN) are 0.6, 1.6, and 2 (over 14 datasets) resp. Thus, LR gets impacted much less with *Categorical* duplicates than RF and ANN. Overall, we make *eight* such observations on the significance of confounders with downstream benchmark. We validate them with simulation study and provide explanations into how ML models with different biases behave with *Categorical* duplicates.

Takeaways for Practitioners. We distill our empirical analysis into a handful of actionable takeaways for ML practitioners and AutoML developers. For instance, LR is more robust to the adverse impact of *Categorical* duplicates than high-capacity RF and ANN as it overfits less. Also, *Similarity* encoding [38] and Transformer-based embedding [61] are more robust than other encodings to tolerate *Categorical* duplicates, thereby diminishing the utility of category deduplication task. We also expose a critical shortcoming of *One-hot* and *String* encoding [70], when *Categorical* duplicates arising in the deployment (or inference) but not during training can affect ML performance significantly.

Some of these insights may be considered folklore by practitioners, but this work is the first in-depth systematic scientific study to assess the impact of *Categorical* duplicates on ML. We explain the impact from the bias-variance tradeoff perspective to put empirical results on a rigorous footing. Our analyses can benefit practitioners to systematically understand the various confounders that matter for accuracy. Also, this can be useful to develop better practices and design ML workflows that are robust to *Categorical* duplicates. Moreover, our work opens up new research directions at the intersection of ML theory, data management, and ML system design. Overall, our work is novel in terms of new labeled dataset, benchmark, and novel empirical analyses. We make four contributions:

1. **A new benchmark dataset.** To the best of our knowledge, this is the first work to curate a large labeled dataset specifically for *Categorical* duplicates where the entities are annotated. We present several insights that characterizes how *Categorical* duplicates exhibit themselves.
2. **Empirical benchmarking to understand the significance of category deduplication on ML.** Our curated downstream benchmark containing "in-the-wild" datasets enables us to point out cases where the task may or may not benefit ML.
3. **Characterization of confounders with simulation study.** Our study can disentangle and explain the impact of confounders on how *Categorical* duplicates affect ML.
4. **Utility of our work.** We present the first in-depth scientific empirical study to systematically charactertize when and why category deduplication can help/not help ML. We present several practical insights for practitioners. We identify open questions for further research where our labeled data can be a key enabler to address them. Also, we open source our benchmark to enable more community-driven contributions [2].

2 RELATED WORK

2.1 Entity Matching (EM) and String Matching (SM) Approaches

EM, the task of identifying whether records from two tables refer to the same real-world entity has received much attention with rule-based [67, 75, 76], learning-based [55, 60, 64, 82], semi-supervised [54], unsupervised [44, 80], and active-learning [63] approaches. They operate at a tuple-level since they have access to the entire feature vectors of the two tables. Note that tuple-level duplicates do not necessarily imply duplication in *Categorical* strings, and also vice versa. Thus, the problem of EM is orthogonal to category deduplication. Admittedly, it is possible to view category deduplication as an extension of row-level deduplication but doing so is non-trivial. *Regardless, our focus is to study only the impact of category deduplication on ML and not how to perform deduplication or compare deduplication methods.* Thus, prior work on EM is complementary to ours in terms of utility for AutoML platforms.

SM, finding strings from two sets that refer to the same real-world entity has been explored with an active learning solution Smurf [37] and an unsupervised learning approach [58]. However, such SM approaches are orthogonal to our focus on studying how *Categorical* duplicates affect ML. We leave automating category deduplication to future work, including potentially extending existing row-level deduplication and SM works.

2.2 Existing Labeled Datasets

Many works have introduced labeled data for related tasks such as EM [42, 43, 55], SM [37, 42, 58], and resolving column-level inconsistencies [59]. Table 2 gives examples of pairs of duplicates from three different datasets in Magellan Data Repository [42], which is used in several prior works [37, 43, 48, 55]. Open domain attributes such as person names and addresses are not *Categorical* features for ML. Instead, such features are context-specific where either custom features are extracted or are completely dropped as they may not generalize for ML. Note that *Categorical* feature assumes mutually exclusive values from a known finite domain. Moreover, *Title* in *Citations* has rich semantic information and is typically used as a *Text* type feature. We find that a large fraction of the datasets used in prior works involve duplication in non-*Categorical* features. Thus, they are not relevant for us to study category deduplication. Although incidental *Categorical* duplicates do arise in a few datasets

Table 2: Examples of labeled duplicates from Magellan Data Reposity [42] with dataset and column names.

A. Restaurants	**Address**	**Phone number**	**Name**
	1929 Hillhurst Ave, Los Angeles, CA	(323) 644-0100	Alcove Cafe & Bakery
	1929 Hillhurst Ave, Los Angeles, CA 90027	(323) 644-0100	Alcove Cafe & Bakery
B. Citations	**Author**	**Entry Type**	**Title**
	David A. Cohn and Michael I. Jordan	article	Active Learning with Statistical Models
	Cohn, David A and Jordan, Michael I	article	Active learning with statistical models
C. Researchers	**Name**	X	
	alicia n aarnio		
	alicia nicole aarnio		

in [37, 59], we posit that we need a systematic benchmark to characterize and understand their impact on ML accuracy that prior works do not focus on. We focus exclusively on the *Categorical* features and curate the first large labeled dataset of entities annotated with duplicates within a *Categorical* column.

2.3 Data Cleaning and Data Prep for ML

CleanML [59] analyses the impact of many data cleaning steps on ML. Our work is along the same direction, but they do not specifically explore *Categorical* deduplication and its causal confounders that matter for accuracy. Although they do study string-level inconsistencies within a column with four real datasets, they are not all *Categorical*. Also, they focus on deriving a broad perspective and a coarse-grained study of many cleaning steps such as this. In contrast, we dive deep into *Categorical* deduplication. We study its causal confounders that matter for accuracy to offer empirical rigor and understand the importance of task scientifically.

We performed an objective benchmarking of a specific ML data prep step, namely the feature type inference task [72]. We build upon our open-sourced datasets but we study a completely different problem. There exist numerous data prep tools such as rule-based [52], exploratory data analysis-based [69], program synthesis [50, 53], and visual interfaces [12] to reduce manual grunt work effort and allow users to productively prepare their data for ML. 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 cleaning to improve ML accuracy and reduce user effort [56, 57]. However, they do not support a cleaning operation with *Categorical* duplicates. Our labeled data can spur more follow-up works in this general direction of automating and improving data prep for ML. Error detection [40], ML for data cleaning methods [71, 78], and even techniques that perform value standardization [35] are orthogonal to our focus since we do not propose new techniques for *Categorical* deduplication.

2.4 AutoML Platforms

Several AutoML tools such as AutoML Tables [5], TransmogrifAI [11], and AutoGluon [45] performing automated model selection do automate many data prep tasks. However, they do not explicitly handle *Categorical* duplicates. Instead, the users are asked to explicitly clean and remove inconsistencies in *Categorical* columns before using their platform [4]. Our labeled data can lead to contributions from community to automate deduplication with a supervised learning-based approach, including potentially extending AutoML with deduplication processor in the optimization process [46, 65, 66]. Moreover, we believe that our empirical analyses and takeaways provide valuable insights to improve AutoML platforms.

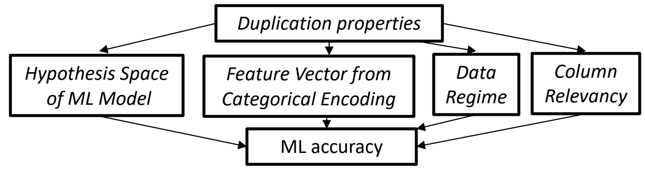

Figure 1: Summarization of confounders impacting ML in the context of *Categorical* column that has duplicates.

3 OUR APPROACH

Consider again an ML practitioner predicting Customer Churn with Table 1 data. She sources the data from multiple tables such as *Customers* and *Zipcode*, which contains details about the area where the customers live. She expects likely duplicates in many *Categorical* columns such as *Gender*, *State*, *Title*, and *Contract* since they are collected using "Free Text" customer surveys. She wants to build and periodically retrain ML pipelines such that they are most robust to likely duplicates. She prefers an ML pipeline which is not necessarily the most accurate but the one that is most reliable. She wants to minimize any adverse impact from *Categorical* duplicates.

To build such an ML pipeline, she wants to choose from different *Categorical* encoding schemes and ML models that are popular on tabular data [77]. Moreover, she has an intuition that many *Categoricals* such as *MostCommonCrime_Zipcode* are not relevant for the target and cleaning them may not be in the best interest of her time. She would like to prioritize her efforts towards cleaning *Categoricals* that are more likely to impact ML. In addition, she has the resources to collect even more training data on customers, but doesn't know if more training data would necessarily translate to a more robust pipeline. Overall, she is fraught with several questions: *How do Categorical duplicates impact the behavior of popular ML classifiers and encodings? Would the effort towards cleaning duplicates be less worthwhile for a non-relevant column as opposed to a column that is relevant for the task? Would collecting more training data help in mitigating the impact of Categorical duplicates?*

Towards answering these questions, we take a step towards empirically assessing the significance of category deduplication on ML. We first identify the important confounders that matter for ML practitioners and study how they affect ML. We hand label a large dataset of real-world *Categorical* columns with duplicates to understand how they occur. We then make empirical observations of the impact of deduplication with different confounders in the real-world. We finally use synthetic study to validate observed phenomenon and intricately study how each confounder impact ML. We first summarize the confounders we study and then explain our three-part contributions towards building an in-depth understanding of the importance of category deduplication for ML.

We focus this study in the context of a *Categorical* column that has duplicates. As the domain size of column shrinks with deduplication, it can influence the following confounders impacting ML (as Figure 1 shows): (1) Feature vector from *Categorical* encoding.

(2) Hypothesis space denoting a set of all prediction functions from feature space to label space that the ML model can represent. (3) Data regime in terms of the number of training examples per unique category in the column. (4) Column relevancy as a measure of the importance of column for the downstream task. Admittedly, there can exist other complex confounders such as skew in class labels with different distributions and conditional duplication properties given the class label. However, in this work, we focus on the above confounders because of their importance for ML practitioners. We leave studying the other confounders to future work.

1. Our Hand-Labeled Data. We create the first large labeled data where true entities within a *Categorical* column are annotated with duplicate categories. This helps us understand the observed properties of *Categorical* duplicates and how they manifest themselves in real-world columns. Our data includes 1262 string *Categorical* columns from 231 raw CSV files. The labeling process took us about 150 man-hours across 6 months. The utility of our labeled data is two-fold. (1) Configure duplication parameter ranges and skew distributions in simulation study. (2) Presents a crucial artifact for researchers to automate the task of *Categorical* deduplication itself and even to objectively evaluate the accuracy of in-house automated mechanisms by AutoML platform developers. In fact, one such labeled data for ML feature type inference task lead to objective benchmarking of existing AutoML tools and even lead to more accurate supervised ML approaches to automate the task [72]. We dive into this part in Section 5.

2. Downstream Benchmark Suite. We use 14 real-world datasets to empirically study the impact of *Categorical* duplicates. We choose these datasets such that they capture different kinds of duplication and also represents the different regimes in the confounder spectrum (explained in Section 6.2). We choose three popular ML classifiers from the spectrum of bias-variance tradeoff to showcase both high-bias and high-variance scenarios. We choose four *Categorical* encoding schemes to showcase how different ways of encoding the feature space impact the behavior of duplicates on ML. We leave in-depth discussion of this component in Section 6.

3. Synthetic Study. We perform a Monte Carlo-style simulation study to achieve two objectives. (1) Confirm the validity of the observations we make with downstream benchmark suite. (2) Disentangle and characterize the effect of duplicates with multiple confounders individually to make the impact interpretable. We embed a true distribution and vary the confounders one at a time while fixing the rest to study their impact on ML accuracy along with how they trend. Although we use hand-labeled data to inform duplication parameter values, our simulation study is not entirely dependent on it. One can very well fix arbitrary duplication parameter values, although that doesn't change the trends and conclusions that we derive. Section 7 explains this in depth.

4 PRELIMINARIES

4.1 Assumptions and Scope

We focus on the ML classification setting over tabular data. We call the ML model to be trained over the data as the "downstream model." Note that our goal is not to study the upstream deduplication process itself, which is handled manually in the paper. We leave designing automated upstream deduplication mechanisms to future work. We focus on understanding how duplicates manifest themselves in 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* 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.

Table 3: Notations used in this paper with a simplified example to illustrate our notions with *State* column categories.

Symbol	Meaning
C	Set of category values in the column A_l
E	Set of unique real-world entities referred by categories from C
ED	Subset of real-world entities that have at least 1 duplicate; ED $\subseteq E$
occ(Z)	Sum of occurrences of all categories present in set Z; Z $\subseteq$ C
D	Set of non-empty sets of duplicate values for each entity in ED; |D| = |ED|

Category set C_i ($1 \le i \le \|C\|$)		Occurrence of Category (occ({C_i}))	Entity set E_j ($1 \le j \le \|E\|$)	
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

4.2 Definitions

We present terms and notations needed to study the effect of *Categorical* duplicates in the context of implications for ML accuracy. We first draw upon notations from a mix of both database theory [62] and ML literature [49] for known concepts. A relational table is defined by schema $R(A_1, A_2, ..., A_n, Y)$ with a relation (instance) r. We use $\mathcal{A}$ to denote a set of columns $\{A_1, A_2, ..., A_n\}$ and Y is the target column for prediction. Note that, formally, a column is referred to as an attribute [62]. Let $A_l (l \in [1, n])$ be a *Categorical* column with a domain $dom(A_l) \subseteq \mathcal{L}$, where $\mathcal{L}$ is the set of strings with finite length. A relation r is defined over $\mathcal{A}$ as a set of mappings with $\{t^p : \mathcal{A} \to \bigcup_{l=1}^{n} dom(A_l), p = 1...|r|\}$, where for each tuple $t^p \in r$, $t^p(A_l) \in dom(A_l)$, $|r|$ is the number of examples in the the table.

Note that *Categorical* strings are not directly consumable by most ML models. Thus, an encoding scheme is required to transform $\mathcal{A}$ to a feature vector to train an ML model. We explain this further in Section 6.1. We now reuse and adapt terminologies from existing database [39, 62] and ML literature [49] together for terms that we need for the rest of the paper. Table 3 lists the notations and explains the terms used with an example. For simplicity of exposition, we focus on one *Categorical* column with duplicates, $A_l \in \mathcal{A}$.

Definition (Category). A Category set $C^l = \{C^l_1, C^l_2, ..., C^l_{|C^l|}\}$ contains all unique domain values occurring in the column A_l. Note that C^l is also referred to as the active domain of A_l relative to relation r [62], i.e., C^l=$adom(A_l, r)$=$\{c \in dom(A_l) \mid \exists t^p \in r, t^p(A_l) = c\}$. We drop the superscript (C^l) and simplify the active domain operation with C only to make it succinct for follow up set algebra. 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 $E \subseteq C$ represents a subset of *Categories* that conceptually refer to different real-world objects. A category from set C can be uniquely mapped to an entity from set E. Let the mapping function be denoted by $M : C \longrightarrow E$. In Table 3, 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 (ties broken lexicographically) in the column to be the true entity. There exist multiple categories representing the same entity, i.e., $M(C_1)$=$M(C_2)$=$M(C_3)$=E_1={*New York*}.

Definition (Occurrence). We define Occurrence (or percentage Occurrence) of category C_i as percentage of times C_i represents E_j in the column. For instance, whenever real-world *New York* entity occurs, 30% and 10% of the times *NY* and *new york* represents them respectively. *New York* is referred to as the entity since it occurs more than *NY* and *new york*. We define the *Occurrence* function as $occ : Z \longrightarrow [0, 100]$. The input Z is a subset $Z \subseteq C$ such that all categories of the subset map to a unique entity E_j ($j \in [1, |E|]$), i.e., $E_j = M(Z_1)$=$M(Z_2)$=...=$M(Z_{|Z|})$. The output is the sum of occurrence values for all categories present in the input set which is a real number in $[0, 100]$. $occ(Z) = occ(Z_1)$+...+$occ(Z_{|Z|})$, e.g., $occ(\{C1\})$ = 60, $occ(\{C2, C3\})$ = 40, and $occ(\{C1, C4\})$ = Undefined.

Definition (Duplicate). There exist a duplicate for E_j whenever E_j=$M(Z_1)$=$M(Z_2)$=...=$M(Z_{|Z|})$, $|Z|$>1. Whenever E_j occurs, the % times it is represented by Z_1, Z_2, and Z_n are $occ(Z_1)$,$occ(Z_2)$, and $occ(Z_n)$ resp. Without loss of generality, we assume that $occ(Z_1)$ >=$occ(Z_2)$>=...>=$occ(Z_{|Z|})$. Since Z_1 most frequently represents the entity (ties broken lexicographically), the other categories Z_2, ..., Z_n are referred to as duplicates of the entity E_j. We define $ED \subseteq E$ as the subset of the entities that contain at least one duplicate, i.e., $\exists Z \subseteq C$ s.t. $|Z|$>1 and $M(Z_1)$=...=$M(Z_{|Z|})$ = ED_j($j \in [1, |ED|]$). We define a duplicate set D_k($k \in [1, |ED|]$) for every entity in ED such that D_k={$Z_2, Z_3, ..., Z_{|Z|}$} represents a set of duplicate values, e.g., ED_1=*California*, D_1={*Ca*} and ED_2= *New York*, D_2={*new york, NY*}.

Definition (Category Deduplication). This is the task of mapping categories from C to an entity from E with mapping function M. The new column after the assignment is called the *deduplicated* column. Set C and E of the *deduplicated* column are identical.

Definition (Column Relevancy). Let $Acc(\mathcal{A})$ be the % classification accuracy obtained by the ML model with a set of columns $\mathcal{A}$ to be used as features in the input. *Relevancy* of a column $A_l \in \mathcal{A}$ is defined as $Acc(\mathcal{A}) - Acc(\mathcal{A} - \{A_l\})$. This quantifies the absolute predictive power of column A_l for the downstream task.

5 OUR HAND-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, ED, D and their occurrences 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.

Table 4: Duplication types w/ examples from our labeled data

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"
		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"
6	Synonyms	Gender	"Female" , "Woman"
		Venue	"Festival Theatre", "Festival Theater"
7	Presence of Extra Information	City	"Houston" , "Houston TX" , "Houston TX 77055"
8	Different grammar	Colour	"triColor" , "tricolored"
		Venue	"Auditorium" , "TheAuditorium"

5.1 Data Sources

We constructed a large real-world dataset of 9921 columns from 1240 data files with diverse application domains such as retail, healthcare, finance, etc., and they were sourced from Kaggle and UCI ML repository [72]. Columns were manually annotated with a standardized 9-class vocabulary of ML feature types. The classes include feature types such as *Numeric*, *Categorical*, *Datetime*, *Sentence*, and *Not-Generalizable* (e.g., primary keys). Using this, we obtain just the string *Categorical* columns. In addition, we collect more such columns and data files using open-source data portals from Chicago city [17], New York [20] and California state [13], Pittsburgh health [15], mental illness project data portal [27], and also real data surveys from FiveThirtyEight [24], and EveryDay-Data [22]. Note that we use 14 data files exclusively from these sources for empirical benchmarking on real downstream tasks in Section 6. Overall, we find 231 raw CSV files with at least one string *Categorical* column. We find a total of 1262 such columns.

Current Limitation. We sourced the *Categorical* columns by leveraging our previous dataset [72]. The raw files were collected from sources such as Kaggle and UCI ML repo where the data file may have been subjected to some pre-processing. However, this is the best we can do from academic research standpoint given legal constraints: acquire large public datasets using public APIs, annotate them, and make them available to the community. It is hard to acquire truly raw data files from several enterprises and make them public due to legal constraints. Also, we do not make any general claims about the manifestation of duplicates across the universe of the datasets. This would require doing a comprehensive analysis of datasets from all sources including that from enterprises and other organizations. However, this does not diminish the utility of our empirical analyses as both the downstream benchmark suite and our synthetic study are independently useful.

5.2 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 1262 *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 enumerate its category set with the count of times each category appears in it. Before scanning the category set, we sort

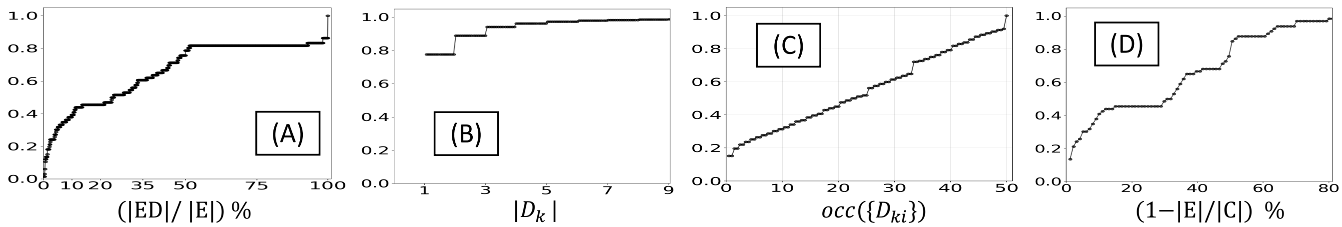

Figure 2: Cumulative distribution function (CDF) over all *Categorical* columns with at least one duplicate on (A) % entities that have at least one duplicate. (B) Duplicate set sizes over all $k \in [1, |ED|]$. The maximum duplicate set size is 148. (C) Duplicate set occurrences over all $k \in [1, |ED|], i \in [1, |D_k|]$. (D) % reduction in domain size with category deduplication.

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. We construct E, ED, and D sets, along with their occurrences for all the columns. *The entire labeling process took us roughly 150 man-hours across 6 months and 3 people.*

5.3 Types of Duplicates and Data Statistics

We find that there exist *eight* types of duplication. 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 stop-words, 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 ED_1= New York and $D_1 = \{$*new-york., NY*$\}$, "*new-york.*" has both *Type 1* and *4* duplication, and the entity *New York* has duplicates with duplication of *Type 1*, *3*, and *4*.

We annotated 67060 entities across all 1262 string *Categorical* columns. We find that almost 5% of those entities have the presence of at least one duplicate with a total of 5584 duplicates. Overall, 66 columns from 47 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 ($|ED|/|E|$). (2) Duplicate set size for all entities of the column (set D). (3) Duplicate occurrences $occ(\{D_{ki}\}), k \in [1, |ED|], i \in [1, |D|]$. Figure 2 plots the CDF of different parameters that characterizes duplication over our labeled data.

We now briefly summarize the presented results. We find that whenever duplicates arise in the column, they can occur quite often. Almost 19% of columns that have duplicates have them in all of their entities! Also, 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. Moreover, the occurrence of duplicates approximately follows a uniform distribution, i.e., all occurrence values up to 50% are roughly equally likely. We present stats on duplication types with takeaways in technical report [73].

6 DOWNSTREAM BENCHMARK

We now empirically study the impact of category duplicates on the downstream ML tasks. Note that our focus is not to compare and evaluate category deduplication methods. We curate a benchmark suite of 14 real-world datasets, each containing a column with duplicates. We use this to empirically evaluate and compare three *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 classifiers.

6.1 Models and Encodings

We choose three popular classifiers used commonly among the ML practitioners as per Kaggle data science survey [77]: Logistic Regression (LR), Random Forest (RF), and an artificial neural network (ANN). These models also present representative choices from the bias-variance tradeoff spectrum [49]: high bias and low variance approach with LR and low bias and high variance approaches with RF and ANN. LR has a low-capacity, while RF has high-capcity and infinite VC dimension as it can represent any function on the data [74]. We use ANN architecture with 2 hidden units, each with 100 neurons. Although there is no magic number for ANN architecture size, the above network already offers a very large VC-dimension and a high-capacity [36]. We use the synthetic study (Section 7) with two extremes in the ANN's bias spectrum to empirically assess the different capacities of ANN on downstream ML.

We encode *Categorical* columns with four popular schemes: *One-hot* (*OHE*), *String* (*StrE*) [70], *Similarity* (*SimE*) [38], and a pre-trained Transformer-based embeddings (*TransformerE*) [61]. *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. With a category set C^l (for A_l) closed during training, *OHE* sets feature vector X_l^p=$[1(t^p(A_l)$=$C_1^l), ..., 1(t^p(A_l)$=$C_{|C^l|}^l)]$, where $1(.)$ is the indicator function and p=$1..|r|$. RF with *OHE* performs binary splits on the data. RF 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 LR, 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. The feature vector for category set C^l is given as X_l^p=$[Sim(t^p(A_l), C_1^l), ..., Sim(t^p(A_l), C_{|C|}^l)]$, where

Table 5: Statistics of the column containing *Categorical* duplicates in our 14 downstream datasets. $|r|$, $|\mathcal{A}|$, and $|Y|$ are the total number of examples, columns, and target classes in the data resp. Duplication types numbering correspond to Table 4. $|rC|$ denotes the number of training examples per category of the set C. 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 ($1 - |E|/|C| < 0.25$), moderate ($1 - |E|/|C| > 0.25$ & < 0.50), and high ($1 - |E|/|C| > 0.50$) resp. We use the following thresholds with the same colors to better interpret the data regime: low ($|rC| < 5$), moderate ($|rC| > 5$ & < 25), and high ($|rC| > 25$). Note that the data regime moves up with category deduplication as category set size has shrunk. We present more fine-grained statistics for the datasets in the tech report [73].

Datasets	|r|	|A|	$\|Y\|$	Duplication Types								Amount of Duplication			Data Regime	
				1	2	3	4	5	6	7	8	$\frac{\|ED\|}{\|E\|}$ %	|C|	1- $\frac{\|E\|}{\|C\|}$ %	*Raw* |rC|	*Truth* |rC|
Midwest Survey	2778	29	9	X	X	X	X	X		X	X	33	1008	64	2.5	6.5
Mental Health	1260	27	5	X	X	X			X	X		40	49	69	23.2	81.2
Relocated Vehicles	3263	20	4	X	X	X	X			X		33	1097	36	2.5	3.8
Health Sciences	238	101	4	X		X			X			36	56	61	3.6	8.3
Salaries	1655	18	8		X		X			X		24	647	29	1.8	2.2
TSM Habitat	2823	48	19	X			X		X	X		11	912	11	2.6	2.9
EU IT	1253	23	5	X	X	X	X	X	X			24	256	35	3.9	5.9
Halloween	292	55	6	X			X			X	X	31	163	51	1.5	3
Utility	4574	13	95	X			X			X		38	199	31	16.2	24.3
Mid or Feed	1006	78	5	X		X	X		X			21	37	62	20.6	59.7
Wifi	98	9	2	X			X				X	30	69	52	1.3	2.5
Etailing	439	44	5	X	X	X	X			X		47	71	68	5.3	14.3
San Francisco	148654	13	2	X		X						11	2159	10	46.3	50.9
Building Violations	22012	17	6		X	X				X		51	270	63	53.7	145

$Sim(.)$ is a similarity metric defined as the dice-coefficient over n-gram (n ranges from 2 to 4) strings [34]. *TransformerE* uses a pre-trained BERT base transformer model to obtain embeddings as features [61]. The feature vector can be computed even for any new categories arising in test set which are unseen during training for both *TransformerE* and *SimE*.

6.2 Datasets used for Analyses

We choose 14 datasets from Section 5.1 such that we not only represent the different duplication types but also span the spectrum of different confounder combinations. Table 5 presents the statistics over our datasets. We use the quantity % reduction in domain size with deduplication (1-$|E|/|C|$) to summarize the magnitude of duplication. We use the data regime notion to denote the number of training examples per category value of the column with duplicates ($|rC|$). We ensure that our selected datasets sufficiently represent different ranges of values (high vs. low measured relatively) in both confounder spectrum. For instance, a dataset that involves a high amount of duplication coupled with high- and low-data regimes such as *Building Violation* and *Midwest Survey* respectively. We will later see in Section 6.4 that the former dataset is robust to duplicates even with almost 51% of their column's entity diluted with duplicates, while the latter is not. This enables us to make specific observations on the role of different confounders, which we validate and disentangle using our simulation study in the Section 7.

Specifically, we obtain the following data files: Midwest Survey[25], Wifi[32], Mental Health[28], EU IT[30], Relocated Vehicles[19], Utility [21], Health Sciences[16], Salaries[23], TSM Habitat[14], Building Violations [18], Etailing[33], Mid or Feed[31], Halloween[26], and San Francisco[29]. Each dataset has a column with *Categorical* duplicates which we manually deduplicated in Section 5.2. We do not claim that these 14 datasets are representative of the percentage one can encounter in practice. Our goal with the downstream benchmark is not to make universal claims about the impact of *Categorical* duplicates on just the commonly encountered datasets. Instead, we select them plainly to showcase different confounder settings and study the behavior of duplicates in those settings. The benchmark suite helps us point out the cases where deduplication matters. This coupled with synthetic study only serves as a guide that can help ML practitioners and AutoML platform developers glean insights. We hope our work inspires more data benchmark standardization in this space with industry involvement.

6.3 Methodology

We partition each dataset into an 80:20 split of train and test. We perform 5-fold cross-validation and use a fourth of the examples in the train set for hyper-parameter search. We use scikit-learn[68], H2o[7], SimilarityEncoder[10], and FlairNLP[1] packages to employ *OHE*, *StrE*, *SimE*, and *TransformerE* resp. We tune the regularization parameter for LR. We tune the number of trees and their maximum depth for RF with values for each ranging from 5 to 100. ANN is L2 regularized and tuned. Due to space constraints, we present the entire grids for hyper-parameter tuning in technical report [73].

6.4 Results

6.4.1 ***Results comparing the ML impact with and without Categorical duplicates.*** Table 6 shows the comparison of downstream ML models built with different encoding schemes in terms

Table 6: Classification accuracy comparison of ML models with different *Categorical* encodings on *Raw* (column has *Categorical* duplicates intact) vs. *Truth* (column has been deduplicated with truth). Accuracy results for *Truth* are shown relative to *Raw* as delta lift/drop in % accuracy. Green, blue, and red colors denote cases where the *Truth* accuracy relative to *Raw* is significantly higher, comparable, and significantly lower (error tolerance of 1%) respectively. *TRel* denotes the true Relevancy of the column that has been deduplicated.

Dataset	Random Forest							ANN						Logistic Regression			
	OHE			*StrE*		*SimE*		*OHE*		*SimE*		TransformerE		*OHE*		*SimE*	
	TRel	*Raw*	*Truth*	*Raw*	*Truth*	*Raw*	*Truth*	*Raw*	*Truth*	*Raw*	*Truth*	*Raw*	*Truth*	*Raw*	*Truth*	*Raw*	*Truth*
Midwest Survey	16.1	49.1	+11.5	59.2	+10	64.9	+4.4	54.7	+9.5	63.4	+3.8	56.8	+8.5	57.2	+9.4	66.7	+2.1
Mental Health	1.3	47.9	+1.1	47.8	-0.1	47.4	-1.7	42.4	+2	43.2	-0.4	45.1	-0.7	46.9	+1.3	46.3	+0.6
Relocated Vehicles	9.1	72.5	+3	81.3	+4.1	88.3	-0.1	83.6	+3.6	89.6	+0	77	+1.6	82.9	+4	88.4	+0.4
Health Sciences	0.4	53.3	+2.2	61.8	+0	60	-2.7	55.1	+4.9	56.4	+1.8	57.3	+0.4	58.7	+0.9	60	+1.8
Salaries	0.7	64.7	+1.7	69.6	+1.3	94.6	+0.4	22	+0.5	19.9	+5.4	25	+3.8	30.4	+0.2	32.4	-1.3
TSM Habitat	5.2	71.2	+0.4	84.1	+1.4	71.2	+0.4	50.7	-2.7	50.7	-2.7	35.3	+0	50.7	+0	50.7	+0
EU IT	3.3	41.2	+1.2	43.6	-0.6	47.8	+4	13.4	-2.4	6.8	+5	9.9	+1.5	29.1	+0	29.1	+0
Halloween	-0.4	40	+1.5	36.2	+1.5	34.7	-4.9	41.9	+4.2	43	+0.8	41.5	+0	42.6	+3.4	49.8	+1.1
Utility	8.1	58.8	+1.4	46.3	+1.2	43.2	+1.4	65.1	+2.3	73.2	+2.5	82.1	-0.2	42.4	-0.2	43	+0.3
Mid or Feed	1.5	40.2	+2.5	35.7	-0.2	36.2	+1.8	34	+2	32.7	+0.2	33.5	+0.1	40.5	+1.7	41.5	-1.2
Wifi	4.2	60	+5.3	57.9	+4.2	50.5	+3.2	52.6	+2.1	48.4	+3.2	61.1	-0.9	64.2	+1.1	58.9	+8.4
Etailing	-0.5	40	+2	44.5	+1.1	38.2	+3	40.2	-3	37.2	+0	36.6	-0.7	41.1	-0.5	38.9	+1.8
San Francisco	24.4	83.4	+0.1	83.9	-0.3	86	+0	86	+0.1	86.1	-0.1	85.6	+0.2	86	-0.1	85.5	+0
Building Violations	-0.1	97.5	-0.1	97.3	+0.1	97.6	+0	97.2	+0	97.4	+0	97.6	-0.6	91.6	+0	91.9	+0

Table 7: Summary statistics over 14 downstream datasets.

Models with Encoding Schemes		*%accuracy lift w/ Truth vs Raw*			*#datasets w/ >1% accuracy lift with Truth vs. Raw*
		Mean	Median	Max	
LR	OHE	1.5	0.6	9.4	6
	SimE	1	0.4	8.4	5
RF	OHE	2.4	1.6	11.5	11
	StrE	1.7	1.2	10	8
	SimE	0.7	0.4	4.4	6
ANN	OHE	1.7	2	9.5	8
	SimE	1.4	0.5	5.4	6
	TransformerE	0.9	0.1	8.5	4

Table 8: Comparisons of overfitting gap with *OHE*. The drop in overfitting gap for *Truth* is shown relative to the *Raw*.

Dataset	RF		ANN		LR	
	Raw	*Truth*	*Raw*	*Truth*	*Raw*	*Truth*
Midwest Survey	50.7	-14.2	45.1	-10.4	24.4	-9.4
Mental Health	42.3	-7.2	26.7	-0.2	11.7	-3.5
Relocated Vehicles	27.3	-3.1	16.4	-3.6	17	-4.1
Health Sciences	35	-8.1	44.9	-4.9	9.3	-5.9
Salaries	34.6	-1	1.4	-0.5	1.9	+0.2
TSM Habitat	28	-0	0.1	+0.5	1.9	-0
EU IT	53.1	-6.6	1.4	+0.9	1.2	-0
Halloween	50.9	-5.8	58.1	-4.2	38.3	-3.5
Utility	41.2	-1.4	26.1	-3	0.7	-0.3
Mid or Feed	58.4	-1.1	66	-2	34.2	-12.8
Wifi	26.2	+1.3	47.4	-2.1	11.1	-2.1
Etailing	54.4	-1.6	59.7	+2.9	41.2	-7.7
San Francisco	-0.2	-0	1.1	-0.1	0.5	-0
Building Violations	1.8	-0.1	1.1	-0.2	0.2	+0.1

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

O1. There exist several downstream cases where *Truth* improves the ML accuracy over *Raw* for any encoding scheme. For instance, the delta accuracy increase with *Truth* on RF with *OHE* is of median 1.6% and up to 11.5% compared to *Raw* (across 14 datasets). Moreover, the delta accuracy increase is of median 2% and up to 9.5% for ANN.

O2. Delta increases in accuracies with *Truth* are typically higher with RF and ANN than LR. The median delta increases in accuracy with RF and ANN using *OHE* are 1.6 and 2, compared to 0.6 for LR. Thus, LR is more robust to duplicates than the high-capacity models.

O3. Truth helps RF using *OHE* the most, *StrE* the second most, and *SimE* and *TransformerE* the least (see Table 7). Interestingly, the median lifts in accuracies due to deduplication with *SimE* are just 0.4 and 0.5 on RF and ANN respectively. Overall, *SimE* improves the ML performance with *Truth* in just ~40% downstream cases. This is because, *SimE* considers morphological variations between the category strings and maps a duplicate to a similar feature vector as the true entity. So, duplicates are often located close to their true entities in the feature space. Thus, any further lift in accuracy due to deduplication is marginal. Although, *TransformerE* is robust to duplicates on most datasets, we find that it often fails to capture misspellings and abbreviation kind duplicates, e.g., *Salaries*.

Table 9: Summary statitics comparing the ML impact with *Standardized* vs. *Raw* and *Standardized* vs. *Truth*.

Stats over 14 Downstream Datasets	Accuracy lift w/ *Standardized* vs. *Raw*				Accuracy lift w/ *Truth vs. Standardized*			
	OHE + RF	OHE + ANN	OHE + LR	TransformerE + ANN	OHE + RF	OHE + ANN	OHE + LR	TransformerE + ANN
Median	0.3	0.1	0.1	0	1	1.8	1.1	0.7
Max	6.3	3.3	3.8	17	7.4	6.3	5.7	9
#dataset w/ >1% accuracy lift	3	4	3	6	7	9	8	6

O4. Deduplication reduces the overfitting gap for all models (from Table 8), thereby improving their generalization ability. Since RF and ANN are more prone to overfitting than LR, their accuracy lifts with *Truth* are more significant.

O5. If the magnitude of overfitting gap on *Raw* is insignificant ($< 1\%$), the amount of possible reduction in overfitting with *Truth* 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*. We observe this across the three classifiers.

O6. Category deduplication increases the column *Relevancy* for all models, i.e., the column becomes more predictive for the downstream tasks after category deduplication. Note that the magnitude of accuracy lift with *Truth* quantifies the increase in column *Relevancy* with *Truth*, as per definition in Section 4.2.

O7. The accuracy lifts with *Truth* on all the models are more significant when the column has high *Relevancy* unless there exist a high-data regime with a large number of training examples per category. Thus, if a column has already high *Relevancy* on *Raw*, it may be worthwhile conservatively to deduplicate, e.g., *Relocated Vehicles* and *Midwest Survey.*

O8. High-data regime is robust to the impact of *Categorical* duplicates than low-data regime, regardless of the amount of duplication. Even a high amount of duplication 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. We do not see any lift in accuracy with category deduplication on any of the ML models.

6.4.2 ***Results with additional evaluation metrics.*** We rerun our downstream benchmark suite with metrics such as macro/micro average of precision, recall, and F1-score. *We find that none of the empirical conclusions made with diagonal accuracy change even with these metrics.* Thus, we defer their results and discussion to techical report [73]. Beyond our observations, there exists a non-trivial interaction of the confounders impacting ML. We disentangle and study them separately in the next section.

6.4.3 ***Results when standardizing the Categorical column that has duplicates.*** In this section, we explore if *Categorical* duplicates are trivial enough to simply disregard for the downstream ML with a common standardization process. To consolidate a *Categorical* domain that has duplicates, we use a set of rules that are commonly used to standardize strings with NLTK [9]: Lower-casing strings, removing special chars, lemmatization to capture morphological variations, trimming excess white spaces, and removing stopwords. We call the data we get as a result, *Standardized* and use the same methodology as Section 6.3 to build ML models. Table 9 presents the comparison of ML accuracy with *Standardized* relative to *Raw* and *Truth* using *OHE* and *TransformerE.* We find that the median accuracy lifts with *Standardized* vs. *Raw* are marginal with all the models. Moreover, the accuracy lifts with *Truth* relative to *Standardized* are significantly higher for many datasets. Even with embedding-based method like *TransformerE* on the standardized column, the downstream ML accuracy suffer (compared to *Truth*) significantly up to 9% on 6 datasets. Thus, accurately consolidating duplicates would likely require either complex heuristics, more custom processing, or even domain knowledge about the data.

Note that the discussion of how *Categorical* duplicates should be handled is completely orthogonal to our focus. Although our goal is not to explore *Categorical* deduplication approaches, here, we simply present a setting where the problem can not be trivialized. Regardless, this does not diminishes the utility of our work on understanding how *Categorical* duplicates affect downstream ML.

7 IN-DEPTH SIMULATION STUDY

We now dive deeper into the impact of each confounder on the downstream ML. 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 commonly used encoding schemes when unseen duplicates during training arise in the test.

7.1 Models and Encodings

The structural model parameters such as the number of tree estimators and maximum tree depth for RF and the specific ANN architecture can largely impact the bias-variance tradeoff. Thus, we fix them to disentangle their impact and better illustrate our findings by presenting two extremes of RF's and ANN's bias spectrum. We use high-bias models such as shallow decision tree with a restricted tree depth of 5 (denoted as ShallowDT), a low-capacity ANN comprising of two hidden units with 5 neurons each (denoted as LoCapANN), and also LR. In addition, we use low-bias high-capacity RF with the number of tree estimators and maximum tree depth being fixed to 50 (denoted as HiCapRF). These values represent the median best-fit parameters obtained by performing a grid search (with the grids being same as Section 6.3) over the synthetically generated data described in Section 7.2. We again use a high-capacity ANN comprising of two hidden units with 100 neurons each (HiCapANN).

We focus this study in the context of *OHE* and *StrE*. *SimE* and *TransformE* require the categories to be semantically meaningful strings. An entity can have duplication of multiple types. Constructing a fine-grained simulator that generates semantically meaningful duplicates while preserving the same true entity is non-trivial and intricate from the language standpoint. We leave designing an apt simulation mechanism for *SimE* and *TransformE* to future.

7.2 Setup and Data Synthesis

There is one relational table with Y being boolean. We include 3 *Categorical* columns in the table and set $|\mathcal{A}|$=3. We set entity set size of every column, $|E|$=10 (all columns have a domain size of 10).

Data generating process. We set up a "true" distribution $P(\mathcal{A}, Y)$ and sample examples in an independently and identically distributed

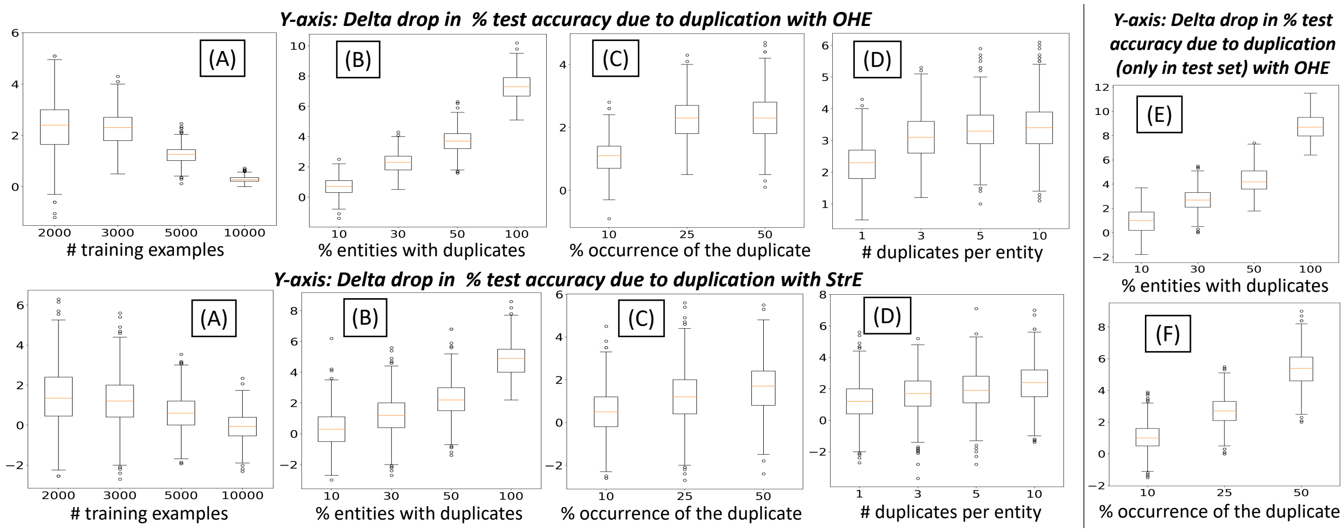

Figure 3: Simulation results for `HiCapRF` with *OHE* and *StrE*. (A-D) Duplicates are present in train, validation, and test set. (E-F) Only test set is diluted with duplicates. (A) Vary $|r|_t$ (# training examples) while fixing $(|ED|/|E|, occ(D_k), |D_k|)$=(30, 25, 1) **(B) Vary** $|ED|/|E|$ **while fixing** $(|r|_t, occ(D_k), |D_k|)$=(3000, 25, 1) **(C) Vary** $occ(D_k)$ **while fixing** $(|r|_t, |ED|/|E|, |D_k|)$=(3000, 30, 1) **(D) Vary** $|D_k|$ **while fixing** $(|ED|/|E|, |r|_t, occ(D_k))$=(30, 3000, 25)**, for all** $k \in [1, |ED|]$**. Parameter settings of (E) & (F) are same as (B) & (C) resp.**

manner. We study a complex joint distribution where all features obtained from $\mathcal{A}$ determine Y. We sample $|r|$ number of total examples, where the examples for training, validation, and test are 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 all models obtained from these 1000 runs.

The exact sampling process is as follows. (1) Construct a conditional probability table (CPT) with entries for all possible values of $\mathcal{A}$ from 1 to $|E|$. We then assign $P(Y=0|\mathcal{A})$ to either 0 or 1 with a random coin toss. (2) Construct $|r|$ tuples of $\mathcal{A}$ by sampling values uniform randomly from $|E|$. (3) We assign Y values to tuples of $\mathcal{A}$ 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 classification accuracy of the ML models on the test split.

Duplication process. We introduce duplicates in a column $A_l \in \mathcal{A}$ of the clean data as follows. (1) Fix fraction of entities to be diluted with duplicates, e.g., $|ED|/|E|$=0.3 (2) Form set ED (entities to be diluted with duplicates) by sampling uniformly randomly $|ED|$ categories from E, e.g., ED=$\{E_3, E_5, E_8\}$. (3) For every entity in ED, fix duplicate set size $|D_k|, k \in [1, |ED|]$, e.g., $|D_k|$=1, $k \in [1, 3]$. We assume that all entities have identical duplicate set sizes. We relax this assumption in Section 7.3.5. (4) Given $|D_k|$, we form the set D by introducing duplicates, e.g., D_1=$\{E_3\text{-}duplicate_1\}$, D_2=$\{E_5\text{-}duplicate_1\}$, D_3=$\{E_8\text{-}duplicate_1\}$. (5) Fix $occ(D_k), k \in [1, |ED|]$. For every duplicate value d in D, set occurrence $occ(d)$=$occ(D_k)/|D_k|$, i.e., assume that all the duplicates representing an entity are equally likely to occur. We relax this assumption in Section 7.3.5. (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 finally obtain the test accuracy of the ML models on the dirty dataset. We use our labeled data to configure apt duplication parameter values such that we can showcase an average and worst-case scenario.

7.3 Results

We vary all confounders one at a time while fixing the rest. We confirm the trends and observations made with *italics.*

7.3.1 ***Varying the data regime.*** Figure 3 (A) presents the delta drop in %accuracy with duplication relative to the ground truth on `HiCapRF` as the number of training examples ($|r|_t$) are varied with both *OHE* and *StrE*. We find that with the rise in $|r|_t$, the delta drop in accuracy decreases. With just 3 training examples per CPT entry ($|r|_t = 3k$ and total entries in CPT=$1k$), duplicates cause a drop of median 2.3% and up to 4.3% accuracy with *OHE*. With 10 training examples, the median and max drops in accuracies due to duplicates with *OHE* are 0.3% and 0.7% 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 encoding and also all the other classifiers:* `LR`, `ShallowDT`, `LoCapANN` *and* `HiCapANN`. *Thus, a high-data regime can tolerate duplicates by remaining more agnostic to the model biases.* Increasing the amount of duplication for a high data regime ($|r|_t$=$10k$) has a marginal impact on accuracy. *Thus, even high duplication has a marginal impact in the high-data regime.* We present the corresponding accuracy plots of the impact of duplicates with data regime changes on the other classifiers in tech report [73].

7.3.2 ***Varying parameters controlling the amount of duplication.*** Figure 3 (B-D) shows how different duplication parameters influence `HiCapRF`. We notice a clear trend: *the drop in accuracy with* `HiCapRF` *rises with the increase in any of the three duplication controlling parameters,* $|ED|/|E|$*,* $occ(D_k)$*, and* $|D_k|$. Among the

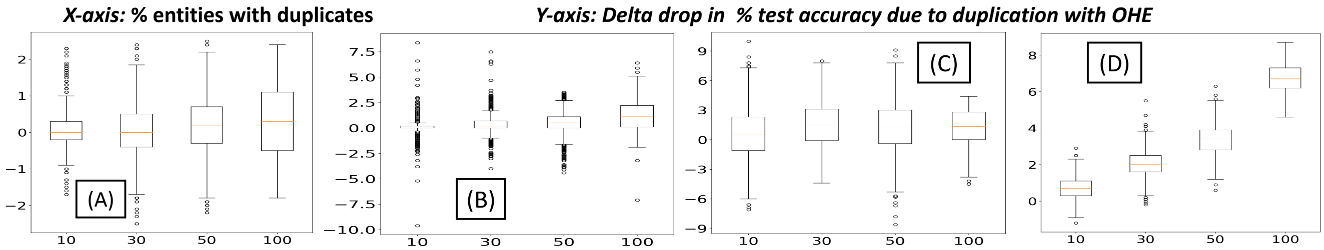

Figure 4: Simulation results with *OHE* for (A) `LR` (B) `ShallowDT` (C) `LoCapANN` (D) `HiCapANN` with the same setup as Figure 3(B).

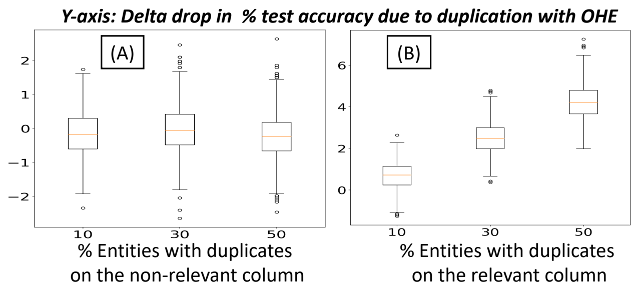

Figure 5: `HiCapRF` results. Vary $|ED|/|E|$, while fixing $(|\mathcal{A}|, |r|_t, occ(D_k), |D_k|)$=$(4, 5000, 25, 1)$. Duplicates introduced on the column with (A) non-positive *Relevancy* (noisy column) (B) high *Relevancy* (predictive column).

three parameters, $|ED|/|E|$ has the most drastic effect on `HiCapRF`. The effects of the increase in $|D_k|$ are less pronounced because all other parameters including $occ(D_k)$ are kept fixed. Thus, there exist more duplicates for the same occurrence. *Interestingly, we find from Figure 3 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.*

Figure 4 presents how a key confounder ($|ED|/|E|$) affects other classifiers. We find that all high-bias models behave similarly as they show a marginal drop in accuracy even when all entities are diluted with duplicates. In contrast, `HiCapANN` exhibits similar behavior as `HiCapRF` when $|ED|/|E|$ is increased. Note that the absolute accuracies of the high-bias approaches are lower than that of high-capacity ones. *Overall, both high-capacity classifiers are more susceptible to the adverse performance impact of duplicates than the high-bias approaches.* We notice the same trend as other confounders ($occ(D_k)$ and $|D_k|$) are varied. We present the corresponding accuracy plots with other confounders in tech report [73].

7.3.3 ***Varying properties of duplicates being mapped to "Others."*** We study how duplicates that do not arise in the train set but are present in the test set (say, during deployment) can impact ML. We modify and repeat our duplication process on just the test set while keeping the train set intact. We introduce just one duplicate in the test set that gets mapped to "*Others*." Figure 3 (E-F) presents the results on `HiCapRF` with *OHE* where $|ED|/|E|$ and $occ(D_k)$ are varied. *We find that the delta drop in accuracies with all parameters are even more higher than the corresponding delta drops when both train and test set were duplicated (Figure 3 (B-C)).* This simply suggests that the presence of unwarranted duplicates during the test can cause downstream ML to suffer significantly.

7.3.4 ***Varying column 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. Thus, we have the presence of both high and low *Relevancy* columns. We introduce duplicates in both types of columns one at a time. Figure 5 present results. *We find that duplication on a highly relevant column has a significant adverse impact on HiCapRF performance. In contrast, the impact is negligible when duplicates are introduced over the noisy column. Even increasing the amount of duplication creates no impact with the low relevancy column. We observe the same trend with HiCapANN.*

7.3.5 ***Introducing skewness in the duplication parameters.*** Until now, we assumed that all entities in ED have identical duplicate set sizes $|D_k|$ and all duplicates in D_k are equally likely to occur. From our labeled data, we find that most entities have small duplicate set sizes and only a few entities have many duplicates. Also, some duplicates are more likely to occur than others in D_k. Thus, we relax these two assumptions and include distributions in $|D_k|$ and $occ(D_k)$ that can better represent the duplication process. We alter our duplication process and approximate $|D_k|$ with a long-tail Zipfian distribution and $occ(D_k)$ with a Needle-and-Thread distribution, varying the skew amount one at a time. *Overall, we find that none of our takeaways change or get invalidated with this setup.* We present the accuracy plots in tech report [73].

7.4 Explanations and Takeaways

We now intuitively explain the behavior of ML classifiers in presence of duplicates with the synthetic study. We check the generalization ability of the ML models with the overfitting gap. Figure 6 presents the overfitting gap results of all classifiers. We find that the delta accuracy drop (Figure 3) closely follows the increase in the overfitting gap due to duplicates with both high-capacity models, `HiCapRF` and `HiCapANN`. 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 high-capacity models, which are prone to overfitting.* However, as the number of training examples rises, the overfitting subsides. This explains our trends in the high-data regime.

We find that `LR` exhibits no amount of extra overfitting with duplicates. This is because the VC dimension of `LR` is linear in the number of features. As the dimensionality of the feature space expands with duplicates, VC dimension of `LR` 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

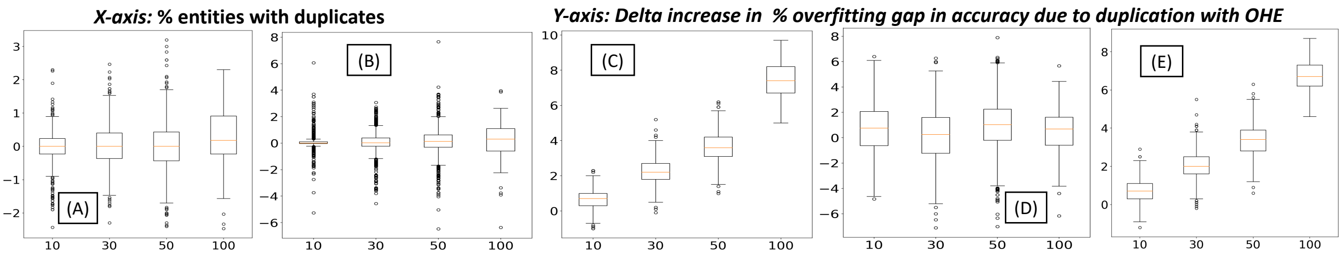

Figure 6: Simulation results on (A) `LR` (B) `ShallowDT` (C) `HiCapRF` (D) `LoCapANN` (E) `HiCapANN` (with the same setup as Figure 3(B)).

check the weights of the hyperplane learned with `LR` 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* `LR` *can learn the true concept by completely ignoring the extra dimensions. Thus, the variance does not rise.* `HiCapRF` with *OHE* makes many binary splits on the data to recover the true concept, causing the tree to fully grow to the restricted height. Chances of further overfitting with duplicates are reduced with a limited height. *This 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.*

8 DISCUSSION

8.1 Public Release

We release a public repository on GitHub with our entire benchmark suite [2]. This includes our entire labeled dataset of 1262 *Categorical* columns along with entities in them annotated with corresponding duplicates and their raw CSV files. We also release the code to run downstream and benchmark suites.

8.2 Utility of our Labeled Data

Besides the utility of our labeled data for empirically benchmarking the impact of *Categorical* duplicates in Section 6, it also serve as a critical artifact to enable researchers in addressing many open questions. We highlight two important research directions below.

a. Design accurate methods for category deduplication. Although *Categorical* duplicates can often impact ML accuracy substantially, many existing open source AutoML tools such as AutoGluon [45] and TransmogrifAI [11] do not support an automated deduplication workflow. Cleaning duplicates manually or using ad hoc rules can be slow and error-prone for many users, especially non-technical lay users who were promised an end-to-end automation of the entire ML workflow. *Our labeled dataset will lead to an objective assessment of the accuracy of automation of different deduplication approaches. Moreover, this will serve towards bulding supervised learning-based approach to automate the category deduplication task itself.* In fact, one such hand-labeled data lead to highly accurate supervised ML approaches and even outperformed the existing industrial-strength tools for ML feature type inference [72].

b. Theoretical quantification. Our empirical study suggests that *Categorical* 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 [79]? Can we set up a decision rule to formally characterize when catgeory deduplication would be needed? Our labeled data can be a key enabler to empirically validate the theory.

8.3 Takeaways for ML Practitioners

We find that the presence of *Categorical* 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 ML with each confounder individually. However, our empirical analyses 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 oblivious to many confounders. While some of the presented insights have remained folklore for practitioners, our work presents the first systematic scientific study and put this on an empirically rigorous footing. We now give general guidelines and actionable insights to help them prioritise their category deduplication effort and also potentially design better end-to-end automation pipelines.

a. Make ML workflows less susceptible to the adverse performance impact of *Categorical* duplicates. `LR` is less prone to overfitting than `RF` and `ANN` when *Categorical* duplicates arise. This is because, as duplicates increase feature dimensionality of *Categoricals*, `LR` can completely ignore the extra dimensions of duplicates by setting their weights close to 0, making them overfit less. Also, *StrE* is relatively more robust than *OHE* when using `RF`. Moreover, *SimE* and *TransformerE* are significantly more robust from *Categorical* duplicates compared to *OHE* and *StrE*. Moreover, unseen *Categorical* duplicates that arise during the deployment phase can degrade ML performance with *OHE* or *StrE*. Overall, *Similarity* encoding and Logistic Regression or *Transformer* embedding 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 *Categorical* duplicates. Moreover, the impact of *Categorical* 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 runtime.

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

REFERENCES

[1] Accessed July 1, 2023. *FlairNLP: Framework for state-of-the-art NLP, https://github.com/flairnlp/flair*.

[2] Accessed July 1, 2023. *Github Repository for studying the impact of Cleaning Category Duplicates on ML, https://github.com/anon-categdups/CategDedupRepo.*

[3] Accessed July 1, 2023. 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

[4] Accessed July 1, 2023. Google AutoML Tables Data Prep User Guidelines, https://cloud.google.com/automl-tables/docs/data-best-practices.

[5] Accessed July 1, 2023. *Google Cloud AutoML, https://cloud.google.com/automl/*.

[6] Accessed July 1, 2023. *H2O Driverless AI, https://www.h2o.ai/products/h2o-driverless-ai/*.

[7] Accessed July 1, 2023. H2o.AI, https://www.h2o.ai/.

[8] Accessed July 1, 2023. *Microsoft AutoML, https://azure.microsoft.com/en-us/services/machine-learning/automatedml/*.

[9] Accessed July 1, 2023. Processing Raw Data with NLTK, https://www.nltk.org/book/ch03.html.

[10] Accessed July 1, 2023. *Similarity Encoder Library, https://github.com/dirty-cat/dirty_cat*.

[11] Accessed July 1, 2023. TransmogrifAI: Automated Machine Learning for Structured Data, https://transmogrif.ai/.

[12] Accessed July 1, 2023. Trifacta: Data Wrangling Tools & Software, https://www.trifacta.com/.

[13] Accessed July 1, 2023. *https://data.ca.gov/*.

[14] Accessed July 1, 2023. *https://data.ca.gov/dataset/tsm-habitat-rapid-assessment-survey-2016-ds28271.*

[15] Accessed July 1, 2023. *https://datacatalog.hsls.pitt.edu/*.

[16] Accessed July 1, 2023. *https://datacatalog.hsls.pitt.edu/dataset/77*.

[17] Accessed July 1, 2023. *https://data.cityofchicago.org/*.

[18] Accessed July 1, 2023. *https://data.cityofchicago.org/Buildings/Vacant-and-Abandoned-Buildings-Violations/kc9i-wq85.*

[19] Accessed July 1, 2023. *https://data.cityofchicago.org/Transportation/Relocated-Vehicles/5k2z-suxx*.

[20] Accessed July 1, 2023. *https://data.ny.gov/*.

[21] Accessed July 1, 2023. *https://data.ny.gov/Energy-Environment/Utility-Company-Customer-Service-Response-Index-CS/w3b5-8aqf* .

[22] Accessed July 1, 2023. *https://everydaydata.co/*.

[23] Accessed July 1, 2023. *https://everydaydata.co/2017/02/07/hacker-news-part-one.html*.

[24] Accessed July 1, 2023. *https://github.com/fivethirtyeight/data/*.

[25] Accessed July 1, 2023. *https://github.com/fivethirtyeight/data/tree/master/region-survey*.

[26] Accessed July 1, 2023. *https://maxcandocia.com/article/2018/Oct/22/trick-or-treating-ages/*.

[27] Accessed July 1, 2023. *https://osmihelp.org/*.

[28] Accessed July 1, 2023. *https://osmihelp.org/research.*

[29] Accessed July 1, 2023. *https://transparentcalifornia.com/salaries/san-francisco/*.

[30] Accessed July 1, 2023. *https://www.asdcode.de/2021/01/it-salary-survey-december-2020.html*.

[31] Accessed July 1, 2023. *https://www.kaggle.com/definitelyliliput/rawscores.*

[32] Accessed July 1, 2023. *https://www.kaggle.com/mlomuscio/wifi-study*.

[33] Accessed July 1, 2023. *https://www.kaggle.com/pushpaltayal/etailing-customer-survey-in-india.*

[34] 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

[35] Adel Ardalan, Derek Paulsen, Amanpreet Singh Saini, Walter Cai, and AnHai Doan. 2021. Toward Data Cleaning with a Target Accuracy: A Case Study for Value Normalization. *CoRR* abs/2101.05308 (2021). arXiv:2101.05308 https://arxiv.org/abs/2101.05308

[36] Peter L Bartlett, Nick Harvey, Christopher Liaw, and Abbas Mehrabian. 2019. Nearly-tight VC-dimension and pseudodimension bounds for piecewise linear neural networks. *The Journal of Machine Learning Research* 20, 1 (2019), 2285–2301.

[37] Paul Suganthan G. C., Adel Ardalan, AnHai Doan, and Aditya Akella. 2018. Smurf: Self-Service String Matching Using Random Forests. *Proc. VLDB Endow.* 12, 3 (2018), 278–291. https://doi.org/10.14778/3291264.3291272

[38] Patricio Cerda, Gaël Varoquaux, and Balázs Kégl. 2018. Similarity encoding for learning with dirty categorical variables. *Machine Learning* 107, 8 (2018), 1477–1494.

[39] Peter Christen. 2012. *Data Matching - Concepts and Techniques for Record Linkage, Entity Resolution, and Duplicate Detection.* Springer. https://doi.org/10.1007/978-3-642-31164-2

[40] Xu Chu, Ihab F. Ilyas, Sanjay Krishnan, and Jiannan Wang. 2016. Data Cleaning: Overview and Emerging Challenges. 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, 2201–2206. https://doi.org/10.1145/2882903.2912574

[41] 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

[42] Sanjib Das, AnHai Doan, Paul Suganthan G. C., Chaitanya Gokhale, Pradap Konda, Yash Govind, and Derek Paulsen. Accessed July 1, 2023. The Magellan Data Repository. https://sites.google.com/site/anhaidgroup/useful-stuff/the-magellan-data-repository?authuser=0.

[43] Sanjib Das, Paul Suganthan GC, AnHai Doan, Jeffrey F Naughton, Ganesh Krishnan, Rohit Deep, Esteban Arcaute, Vijay Raghavendra, and Youngchoon Park. 2017. Falcon: Scaling up hands-off crowdsourced entity matching to build cloud services. In *Proceedings of the 2017 ACM International Conference on Management of Data*. 1431–1446.

[44] Dong Deng, Wenbo Tao, Ziawasch Abedjan, Ahmed K. Elmagarmid, Ihab F. Ilyas, Guoliang Li, Samuel Madden, Mourad Ouzzani, Michael Stonebraker, and Nan Tang. 2019. Unsupervised String Transformation Learning for Entity Consolidation. In *35th IEEE International Conference on Data Engineering, ICDE 2019, Macao, China, April 8-11, 2019*. IEEE, 196–207. https://doi.org/10.1109/ICDE.2019.00026

[45] 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

[46] 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

[47] 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).

[48] Chaitanya Gokhale, Sanjib Das, AnHai Doan, Jeffrey F Naughton, Narasimhan Rampalli, Jude Shavlik, and Xiaojin Zhu. 2014. Corleone: Hands-off crowdsourcing for entity matching. In *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*. 601–612.

[49] 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

[50] 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.

[51] 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

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

[53] 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.

[54] 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)*, Fabien Gandon, Marta Sabou, Harald Sack, Claudia d'Amato, Philippe Cudré-Mauroux, and Antoine Zimmermann (Eds.), Vol. 9088. Springer, 388–402. https://doi.org/10.1007/978-3-319-18818-8_24

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

[56] 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

[57] 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

[58] Peng Li, Xiang Cheng, Xu Chu, Yeye He, and Surajit Chaudhuri. 2021. Auto-FuzzyJoin: Auto-Program Fuzzy Similarity Joins Without Labeled Examples. In *SIGMOD '21: International Conference on Management of Data, Virtual Event, China, June 20-25, 2021*, Guoliang Li, Zhanhuai Li, Stratos Idreos, and Divesh Srivastava (Eds.). ACM, 1064–1076. https://doi.org/10.1145/3448016.3452824

[59] Peng Li, Xi Rao, Jennifer Blase, Yue Zhang, Xu Chu, and Ce Zhang. 2021. CleanML: A Study for Evaluating the Impact of Data Cleaning on ML Classification Tasks. In *37th IEEE International Conference on Data Engineering, ICDE 2021, Chania, Greece, April 19-22, 2021*. IEEE, 13–24. https://doi.org/10.1109/ICDE51399.2021.00009

[60] 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

[61] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692* (2019).

[62] David Maier. 1983. *The theory of relational databases*. Vol. 11. Computer science press Rockville.

[63] 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

[64] 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

[65] Felix Neutatz, Binger Chen, Ziawasch Abedjan, and Eugene Wu. 2021. From Cleaning before ML to Cleaning for ML. *IEEE Data Eng. Bull.* 44, 1 (2021), 24–41. http://sites.computer.org/debull/A21mar/p24.pdf

[66] Felix Neutatz, Binger Chen, Yazan Alkhatib, Jingwen Ye, and Ziawasch Abedjan. 2022. Data Cleaning and AutoML: Would an optimizer choose to clean? *Datenbank-Spektrum* (2022), 1–10.

[67] 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

[68] 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.

[69] 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.*

[70] J. Ross Quinlan. 1986. Induction of decision trees. *Machine learning* 1, 1 (1986), 81–106.

[71] Theodoros Rekatsinas, Xu Chu, Ihab F. Ilyas, and Christopher Ré. 2017. HoloClean: Holistic Data Repairs with Probabilistic Inference. *Proc. VLDB Endow.* 10, 11 (2017), 1190–1201. https://doi.org/10.14778/3137628.3137631

[72] 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.

[73] Vraj Shah, Thomas Parashos, and Arun Kumar. Accessed July 1, 2023. *How do Categorical Duplicates Affect ML? A New Benchmark and Empirical Analyses (Technical Report)*. https://adalabucsd.github.io/papers/TR_2023_CategDedup.pdf.

[74] Shai Shalev-Shwartz and Shai Ben-David. 2014. *Understanding Machine Learning - From Theory to Algorithms*. Cambridge University Press. http://www.cambridge.org/de/academic/subjects/computer-science/pattern-recognition-and-machine-learning/understanding-machine-learning-theory-algorithms

[75] 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

[76] 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

[77] Survey. Accessed July 1, 2023. 2021 State of Data Science and Machine Learning. https://www.kaggle.com/kaggle-survey-2021.

[78] Nan Tang, Ju Fan, Fangyi Li, Jianhong Tu, Xiaoyong Du, Guoliang Li, Samuel Madden, and Mourad Ouzzani. 2021. RPT: Relational Pre-trained Transformer Is Almost All You Need towards Democratizing Data Preparation. *Proc. VLDB Endow.* 14, 8 (2021), 1254–1261. https://doi.org/10.14778/3457390.3457391

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

[80] 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

[81] 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

[82] 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