How do Categorical Duplicates Affect ML? A New Benchmark and Empirical Analyses

Vraj Shah
University of California, San Diego
vps002@eng.ucsd.edu

Thomas Parashos
California State University,Northridge
thomas.parashos.589@my.csun.edu

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 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 three popular downstream classifiers: Logistic Regression (`LR`), Random Forest (`RF`), and Multi-layer Perceptron (`MLP`). We finally use simulation studies to validate our observations and disentangle multiple confounders that impact ML. We find that `LR` and *Similarity* encoding for *Categoricals* are more robust to duplicates than *One-hot* encoding on the two high-capacity classifiers, `RF` and `MLP`. We provide actionable takeaways that can potentially help both 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 works 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.

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 [4], Microsoft's AutomatedML [7], and H2O Driverless AI [5] 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 [36, 41]. 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 [30, 74]. 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.

Table 1: Customers data used for ML Churn Prediction.

CustId	*Name*	*Age*	*Gender*	*State*	*Title*	*Contract*	*Churn*
101	John	42	Male	California	sr. Scientist	monthly	'Y'
102	Jerry	29	Mail	CA	snr scientist	Month-to -month	'N'

Consider a simplified dataset to be used for a common ML classification task, *Customers Churn prediction* 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 can potentially dilute the signal strength that one can extract from a *Categorical* column to be used for ML. Thus, an ML practitioner would often deduplicate categories before ML. 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 [2]. This can involve non-trivial amount of deduplication effort as duplicates can arise as misspellings, abbreviations, and synonyms, even within the same column. Note that deduplication of the *Categorical* column is different from entity deduplication, which involves disambiguating entities given the full tuples from two tables [45, 75]. 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? How do duplicates impact commonly used ML classifiers? 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 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., # training examples), *Categorical* encoding, and ML model. Figure 1 details these confounders. Considering this, we make 3 component contributions to covers our goals. (1) We produce a labeled data to study how real-world duplicates arise. This

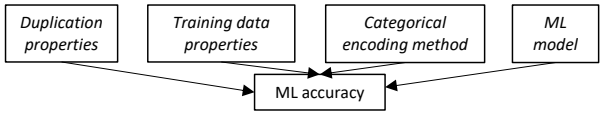

Figure 1: High-level summarization of confounders impacting ML accuracy in presence of duplicates.

helps us set up the duplication properties in simulation study. (2) We phenomenalise the impact on ML by empirically benchmarking with real-world data containing duplicates in presence of multiple 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. Hand-Labeled Data. To understand the behavior of *Categorical* duplicates, we first ask several questions: *How do duplicates manifest themselves in real-world columns? Do they happen often? How much can the domain size be reduced with deduplication? We address them by creating the first labeled data where true entities within a Categorical column are annotated with duplicates.* Our data 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 data is two-fold. (1) Get an estimate of several properties of duplicates such as their cardinalities and occurrences. This enables us to control the synthetic duplication process that is real-world representative in simulation study. (2) Provide a methodological approach to help address critical open questions such as automating *Categorical* deduplication itself. For instance, this can benefit 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 [25].

2. Downstream Benchmark Suite. We create a benchmark suite of 14 real-world datasets to empirically benchmark the impact of duplicates. We make the following choices with each confounder. (1) We choose three popular classifiers from the entire spectrum of bias-variance tradeoff: Logistic Regression (`LR`), a low-capacity model with high bias and Random Forest (`RF`), a high-capacity model with high variance at the two ends. We choose a high-capacity multi-layered perceptron with VC dimension lower than `RF` somewhere in between them [66]. `LR` and `RF` are also the two most popular classifiers among ML practitioners, as per Kaggle data science survey [69]. (2) We focus on three easy to understand *Categorical* encoding schemes, *One-hot*, *String*-based (applicable for tree learners), and *Similarity*. We choose the first two since they are already popular in practice, as per study on OpenML workflows [49]. We choose *Similarity* as it offers duplicates with a feature vector representation that is similar to that of their true entities. In contrast, *One-hot* leads to duplicates with feature vectors orthogonal to their true entities. There exists a large stock of encoding methods for representing *Categoricals* [38]. We leave studying them to future work for tractability sake. We make empirical observations in the two extremes of (3) Data regime in terms of the training examples per category. (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 simulation study to intricately study how different confounders affect ML accuracy. We disentangle the structural model parameters impacting the bias-variance tradeoff by fixing the model capacity. We embed two different true distributions and introduce duplicates 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.

Relationship to Prior Work. Entity Matching (EM) solutions [45, 52, 75] 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, 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.* We leave it to future work to automate this, including potentially extending existing row-level deduplication works.

Note that a *Categorical* column for an ML model assumes values from a finite closed domain. Thus, non-generalizable open domain person names, custom processable addresses, or even semantically rich textual descriptions used in public datasets [45, 46] and string matching literature [27, 50] are not *Categorical*; thus, they are beyond the scope of this work. Although incidental *Categorical* duplicates do arise in prior datasets [27, 45, 51], we posit that we need a systematic benchmark to characterize and understand their impact on ML accuracy that prior works do not focus on. A prior work evaluated the impact of many data cleaning steps on ML [51]. Our work is along the same direction, but they did not specifically explore *Categorical* deduplication and its causal confounders that matter for accuracy. We deep dive into *Categorical* deduplication to offer empirical rigor and understand the importance of task scientifically, rather than a coarse-grained study of cleaning for ML.

Empirical Evaluation. An empirical comparison of our downstream benchmark reveals that deduplication can often improve the ML accuracy significantly, e.g., the impact with duplicates on `LR`, `RF`, and `MLP` using *One-hot* is significant (more than 1% accuracy) on 6, 11, and 8 datasets (out of 14) respectively. Thus, `LR` gets impacted much less than the high-capacity `RF` and `MLP`. 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. Moreover, we provide explanations and insights into how ML models with different biases behave with 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 adverse impact of duplicates than high-capacity `RF` and `MLP` as it overfits less. Moreover, *Similarity* is more robust than other encodings to tolerate duplicates, thereby diminishing the utility of deduplication. We also expose a critical shortcoming of *One-hot* and *String* encodings, when duplicates arising in the deployment but not during training can lead to significant degradation in ML performance.

Table 2: Notations used in this paper.

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	A set of non-empty sets of duplicate values for each entity in ED; |D| = |ED|

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 the 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 duplicates. Moreover, our work opens up several new research directions at the intersection of ML theory, data management, and ML system design. Finally, we open source our entire benchmark including our labeled data, downstream suite, and simulation framework [1].

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 a large labeled dataset, specifically for *Categorical* duplicates where the entities are annotated. We present several insights that characterizes how duplicates exhibit themselves.
2. **Empirical benchmarking to understand the significance of deduplication on ML.** Our curated downstream benchmark containing "in-the-wild" datasets enables us to point out cases where ML may or may not benefit with deduplication.
3. **Characterization of confounders with simulation study.** Our comprehensive synthetic study can disentangle and explain the impact of confounders on how duplicates affect ML.
4. **Utility of our study.** We present the first in-depth scientific empirical study to systematically charactertize when and why *Categorical*-level deduplication can help/not help ML. We present several practical insights for practitioners. We identify open questions for further research 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.

2 PRELIMINARIES

2.1 Assumptions and Scope

We focus on the ML classification setting over structured data. We call the ML model to be trained over the data as the "downstream model." This refers to the ML model that is built on the table after deduplication has been done. 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: A simplified example to illustrate our notions with *State* column categories.

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

2.2 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 [53] and ML literature [39] 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 [53]. 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 $\{t^p : \mathcal{A} \rightarrow \bigcup_{l=1}^{n} dom(A_l), p = 1...|r|\}$, 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 the set of columns $\mathcal{A}$ to a feature vector to train an ML model. We explain this further in Section 4.1. We now reuse and adapt terminologies from existing database [29, 53] and ML literature [39] together for terms that we need for the rest of the paper. Table 2 lists the notations and Table 3 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 [53], 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 $M : C \rightarrow E$. 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 $occ : Z \rightarrow [0, 100]$. 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 $E_j = M(Z_1) = M(Z_2) = ... = M(Z_{|Z|})$ and $|Z| > 1$. 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 $occ(Z_n)$ respectively. Without the 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). *Category Deduplication* is the task of mapping categories from set C to an entity from set E with the mapping function M. 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 (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 predictive power of the column A_l for the downstream task.

3 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.

3.1 Data Sources

We constructed a large real-world dataset of 9921 columns manually annotated with a standardized 9-class vocabulary of ML feature types [64]. The classes include feature types such as *Numeric*, *Categorical*, *Datetime*, *Sentence*, and *Not-Generalizable* (e.g., primary keys). Using this, we obtain 217 raw CSV files that contain at least one string *Categorical* column. Overall, we find 1248 such columns.

Table 4: Duplication types with examples from our hand-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" , "Gold/White"
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"

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 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, ED, and D, along with their occurrences for all the columns. *The entire labeling process took roughly 120 man-hours across 5 months.*

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 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.*

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

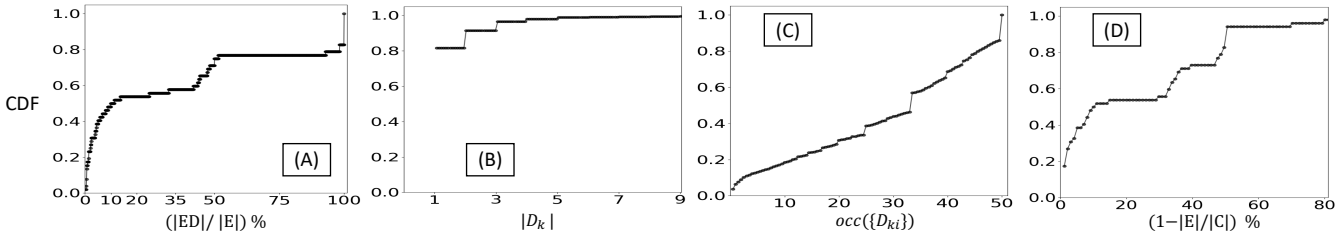

Figure 2: CDF over all *Categorical* columns with at least 1 duplicate on (A) percentage of entities that have at least one duplicate. (B) Duplicate set sizes over all $k \in [1, |ED|]$. The maximum duplicate set size is 148. (C) Duplicate set occurrences over all $k \in [1, |ED|], i \in [1, |D_k|]$.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

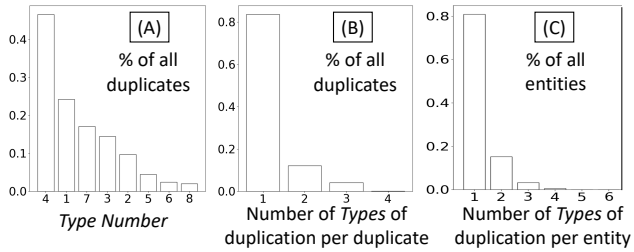

Figure 3: Histogram plots to illustrate how duplication types (from Table 4) arise across all columns in all files. x and y denote x-axis and y-axis respectively. (A) y% duplicates have duplication of at least one x *Type Number*. (B) y% duplicates have x different duplication types within. (C) y% entities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

have at least one duplicate ($|ED|/|E|$). (2) Duplicate set size for all entities present in the column (set D). (3) Duplicate occurrences $occ(\{D_{ki}\}), k \in [1, |ED|], i \in [1, |D|]$. 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.

(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, 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 same. Additionally, 1% of all entities 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 column, they can occur quite often.

(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. Moreover, the occurrence of duplicates approximately follows a uniform distribution, i.e., all occurrence values up to 50% are roughly equally likely. Finally, $|ED|/|E|$ 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 are the most common, while *Synonyms* and *Grammar* issues are 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 choose three popular downstream classifiers used among the ML practitioners as per Kaggle data science survey [69]: `LR`, `RF`, and a two-layered MLP. These popular models also present representative choices from the bias-variance tradeoff spectrum [39]: high bias and low variance approach with `LR` and low bias and high variance approaches with `RF` and `MLP`. Note that `MLP`'s VC dimension is in between the `LR` and `RF` [66].

We encode the *Categorical* columns with three commonly used encoding schemes: (1) *One-hot* encoding (*OHE*). (2) *String* encoding (*StrE*) [63] (3) *Similarity* encoding (*SimE*) [28]. *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 duplicates in *four* of our 14 downstream datasets. We present the statistics for the rest of the datasets in the technical report [65]. $|r|$, $|\mathcal{A}|$, and $|Y|$ are the total number of examples, columns, and target classes in the dataset respectively. $|rC|$ denotes the number of training examples per category of the set C. P is the fraction of $|E|$ that has at least 1 duplicate being mapped to "*Others*" category in the validation set for *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 ($1-|E|/|C|<0.25$), moderate ($1-|E|/|C|>0.25$ & <0.50), and high ($1-|E|/|C|>0.50$) respectively. 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 deduplication as category set size has shrunk.

Datasets	|r|	|$\mathcal{A}$|	|Y|	Amount of Duplication					Data Regime		P %
				$\frac{\|ED\|}{\|E\|}$ %	median $\|D_k\|$	median occ($\{D_{ki}\}$)	|C|	1- $\frac{\|E\|}{\|C\|}$ %	|rC|	|rC| after dedup (Increase w.r.t Raw)	
Midwest Survey	2778	29	9	33.1	2	4	1008	64	2.5	6.5 (2.6x)	23.6
Relocated Vehicles	3263	20	4	33.2	1	20	1097	35.8	2.5	3.8 (1.5x)	14.9
San Francisco	148654	13	2	10.7	1	25	2159	9.8	46.3	50.9 (1.1x)	3.2
Building Violations	22012	17	6	51	2	4.8	270	63	53.7	145 (2.7x)	4.4

$Sim(.)$ is a similarity metric defined as the dice-coefficient over n-gram (n ranges from 2 to 4) strings [26]. This feature vector can be computed even for any new categories arising in test set which are unseen during training for *SimE*.

4.2 Real Datasets and Labeling

We collect 14 datasets from real open-source portals such as Chicago city, New York and California state, Pittsburgh health, mental illness project, and real data surveys from FiveThirtyEight, EveryDayData, and Kaggle. Specifically, we obtain the following datasets: Midwest Survey [17], Mental Health [19], EU IT [21], Wifi [23], Relocated Vehicles [14], Health Sciences [12], Salaries [16], TSM Habitat [11], Building Violations [13], Etailing [24], Mid or Feed [22], Halloween [18], San Francisco [20], and Utility [15]. Each dataset has a column with duplicates which we manually deduplicate. We leave automating the upstream deduplication task with a learning-based approach using our labeled data to future work.

Table 11 presents the statistics with different confounders that can potentially impact ML performance over four of our datasets. There are four data-dependent confounders that can potentially impact ML accuracy. (1) Three parameters characterizing duplicates: $|ED|/|E|$, $|D_k|$, $occ(D_k)$. We use the quantity % reduction in domain size with deduplication ($1-|E|/|C|$) to summarize the amount of duplication. (2) Data regime relative to the complexity of the prediction task. We simplify it as the number of training examples per category value ($|rC|$). We ensure that our selected datasets sufficiently represent different ranges of values (high vs. low measured relatively) in each confounder spectrums. This allows us to make empirical observations by assessing their significance in Section 4.4. We use simulation study (Section 5) to disentangle the impact individually with all confounders. We hope our work inspires more data benchmark standardization in this space with industry involvement.

4.3 Methodology

We partition each dataset into an 80:20 split of train and test set. We perform 5-fold cross-validation and use a fourth of the examples in the train set for hyper-parameter search. 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. The MLP architecture comprises of 2 hidden units with 100 neurons each and is L2 regularized. Due to space constraints, we present the entire grids for hyper-parameter tuning in the technical report [65].

4.4 Results

Table 6 and 7 shows the end-to-end comparison of the downstream ML models built with different encoding schemes in terms 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 (*Deduped*) lead to an 11.5% lift in accuracy relative to the *Raw*. Table 8 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 9. 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 RF with *OHE* 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 2% and up to 9.5% for MLP.

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

O3. Deduped helps RF using *OHE* the most, *StrE* the second most, and *SimE* the least (see Table 8). Interestingly, the median lifts in accuracies due to deduplication with *SimE* are just 0.4 and 0.5 on RF and MLP respectively. Overall, *SimE* improves the ML performance 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.

O4. Deduplication reduces the overfitting gap for all models (from Table 9). Thus, deduplication can reduce the variance component of

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.

Dataset	Logistic Regression (LR)						Random Forest (RF)							
	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 7: Downstream accuracy comparison of (high-capacity) MLP with different *Categorical* encoding on *Raw* vs. *Deduped*.

Dataset	Relevancy *OHE* on *Raw*	OHE		SimE	
		Raw	*Deduped*	*Raw*	*Deduped*
Midwest Survey	16.4	54.7	+9.5	63.4	+3.8
Mental Health	-3.8	42.4	+2	43.2	-0.4
Relocated Vehicles	21.9	83.6	+3.6	89.6	+0
Health Sciences	-4	55.1	+4.9	56.4	+1.8
Salaries	-5.6	22	+0.5	19.9	+5.4
TSM Habitat	0.1	50.7	-2.7	50.7	-2.7
EU IT	6.9	13.4	-2.4	6.8	+5
Halloween	-1.5	41.9	+4.2	43	+0.8
Utility	16.1	65.1	+2.3	73.2	+2.5
Mid or Feed	-0.5	34	+2	32.7	+0.2
Wifi	-6.3	52.6	+2.1	48.4	+3.2
Etailing	0.2	40.2	-3	37.2	+0
San Francisco	26.9	86	+0.1	86.1	-0.1
Building Violations	0.1	97.2	+0	97.4	+0

Table 8: Summary statistics to illustrate the impact of deduplication on ML models using different encoding schemes with 14 downstream datasets. ∗ and † denote two and one cases where both encoding schemes perform the best resp.

	LR		Random Forest (RF)			MLP	
	OHE	*SimE*	*OHE*	*StrE*	*SimE*	*OHE*	*SimE*
% lift in accuracy with Deduped							
Mean	1.5	1	2.4	1.7	0.7	1.7	1.4
Median	0.6	0.4	1.6	1.2	0.4	2	0.5
75th percentile	1.6	1.6	2.4	1.5	2.7	3.3	3
Max	9.4	8.4	11.5	10	4.4	9.5	5.4
# downstream datasets where							
>1% lift in accuracy on *Deduped*	6	5	11	8	6	8	6
Best performing encoding on *Raw*	6*	10*	5	3	6	6†	9†
Best performing encodin on *Deduped*	5*	11*	5	3	6	8*	8*

the test error and improve the generalization ability of the classifiers. Since RF and MLP are more prone to overfitting than LR, their lifts in accuracies with *Deduped* are more significant. 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*. We observe this across the three downstream classifiers.

O6. Deduplication increases the column *Relevancy* for all models, i.e., the column becomes more predictive for the downstream tasks after deduplication. Note that the amount of increase in column *Relevancy* with *Deduped* also quantifies the accuracy lift with *Deduped*.

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

O8. High-data regime is robust to the impact of duplicates than the low-data regime, regardless of the amount of duplication. Even a high amount of duplication has a negligible impact in the high-data

Table 9: Delta drop in % overfitting gap in accuracy with *OHE*. The overfitting gap for *Deduped* is shown relative to the *Raw*.

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

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 deduplication on any of the ML models.

We also rerun our downstream benchmark suite using additional evaluation 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 additional metrics.* Thus, we defer their results to a technical report [65]. Overall, there exist *six* confounders that can potentially impact the downstream ML: $|ED|/|E|$, $|D_k|$, $occ(D_k)$, $|rC|$, column *Relevancy*, and 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 the *six* confounders that impact ML. Thus, we use the simulation study in the next section to disentangle and study them separately.

5 IN-DEPTH SIMULATION STUDY

We now use a Monte Carlo-style simulation study to 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 arise in the test set.

Encoding Schemes. We focus this study in the context of *OHE* and *StrE*. *SimE* require the categories to be semantically meaningful strings. We find from Table 4 and Figure 3 that an entity can have duplication of multiple types. Constructing a fine-grained simulator that generates semantically meaningful duplicates while preserving the same ground truth entity is non-trivial and intricate from the language standpoint. Thus, we leave designing an apt simulation mechanism for *SimE* to future work.

Downstream Models. The structural model parameters such as the number of tree estimators and maximum tree depth for RF and the specific MLP 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 MLP'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 MLP comprising of two hidden units with 5 neurons each (denoted as LoCapMLP), 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 4.3) over the synthetically generated data described in Section 5.1. We again use a high-capacity MLP comprising of two hidden units with 100 neurons each (HiCapMLP).

Setup. There is one relational table with Y being boolean (domain size is 2). We include three *Categorical* columns in the table and set $|\mathcal{A}|$ 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 $P(\mathcal{A}, Y)$ and sample examples in an independently and identically distributed manner. We study two different simulation scenarios: AllA and Hyperplane. These scenarios represent two opposite extremes of how RF and hyperplane-based classifiers fits the data. AllA represents a complex joint distribution where all features obtained from $\mathcal{A}$ determine Y. Although a high-capacity model such as RF can recover this with rule-based splits, LR can extremely underfit. Hyperplane represents a distribution where the data is simply separable with a hyperplane. This distribution is well-suited for LR and MLP (where each neuron defines a hyperplane), but represents a bad-case scenario for RF that requires many numbers of splits to recover the true concept. We sample $|r|$ 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 all models obtained from these 1000 runs.

5.1 Scenario AllA

Data generating process. We create a true distribution that maps all features obtained from $\mathcal{A}$ to Y. 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 dataset in the following fashion. (1) Fix fraction of entities to be diluted with duplicates, e.g., $|ED|/|E|$=30 (2) Form set ED (set of entities that are to be diluted with duplicates) by sampling uniformly randomly $|ED|$ categories from $|E|$ (E_1 to $E_{|E|}$), e.g., ED=$\{E_3, E_5, E_8\}$. (3) For every entity in set ED, fix the duplicate set size $|D_k|, \forall 1 \leq k \leq |ED|$, e.g., $|D_k|$=1, $\forall 1 \leq k \leq 3$. We assume that all entities have identical duplicate set sizes. We relax this assumption

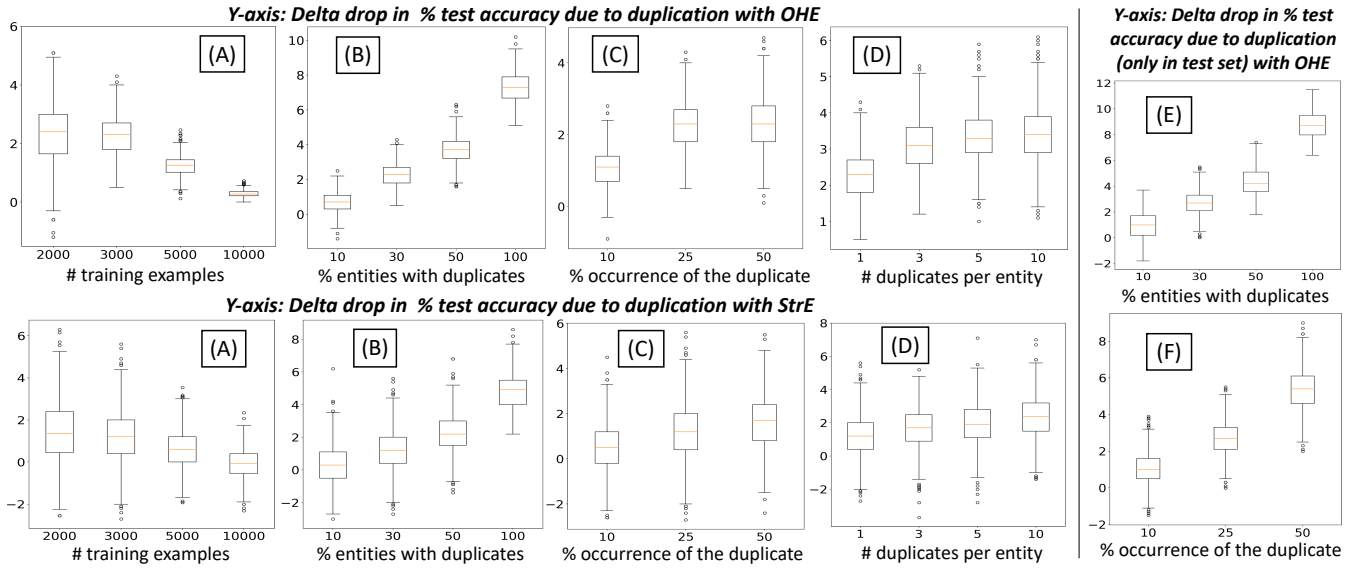

Figure 4: `AllA` scenario results for `HiCapRF` with *OHE* and *StrE*. (A-D) Duplicates present in train, validation, and test set. (E-F) Only test set is diluted with duplicates. (A) Vary $|r|_t$ (# training examples) while fixing $(|ED|/|E|, occ(D_k), |D_k|)$=(30, 25, 1) (B) Vary $|ED|/|E|$ while fixing $(|r|_t, occ(D_k), |D_k|)$=(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) and (F) are same as (B) and (C) resp.

in Section 5.1.3. (4) Given $|D_k|$, we form the set D by introducing duplicates, e.g., D_1={E_3-$duplicate_1$}, D_2={E_5-$duplicate_1$}, D_3={E_8-$duplicate_1$}. (5) Fix $occ(D_k), 1 \leq k \leq |ED|, 1 \leq i \leq |D_k|$. For every duplicate value d in D, set occurrence $occ(d)$=$occ(D_k)/|D_k|$, 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 finally obtain the test accuracy of the ML models on the dirty dataset.

Results. We use our labeled data to pick appropriate values for the confounders such that we can showcase an average and worst-case scenario. We vary all confounders one at a time while fixing the rest. We confirm the trends and observations made with *italics*.

5.1.1. Varying the data regime. Figure 14 (A) presents the delta drop in %accuracy with duplication relative to the ground truth clean dataset 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), the presence of duplicates can 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 accuracy 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`, `LoCapMLP` *and* `HiCapMLP`. *Thus, a high-data regime can tolerate duplicates by remaining more agnostic to the model biases.* Moreover, increasing the amount of duplication for a high data regime ($|r|_t$ = 10k) has a marginal impact on the accuracy. *Thus, even high duplication has a marginal impact in the high-data regime.* We present the corresponding accuracy plots of the effects of duplicates with data regime changes on all the other classifiers in the technical report [65].

5.1.2. Varying parameters controlling the amount of duplication. Figure 14 (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 $|D_k|$.* As $|ED|/|E|$, $occ(D_k)$, and $|D_k|$ are each increased 5 folds, the magnitude increase in delta drop accuracy with duplicates using *OHE* are 5.3x, 2.1x, and 1.4x respectively. In contrast, the same magnitude increases using *StrE* are 7.3x, 3.4x, and 1.6x respectively. Thus, among the three duplication 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 14 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 5 presents how a key confounder ($|ED|/|E|$) affects other studied classifiers. We find that all high-bias models behave similarly as they show a marginal drop in accuracy even when all the entities are diluted with duplicates. In contrast, `HiCapMLP` 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 [65].

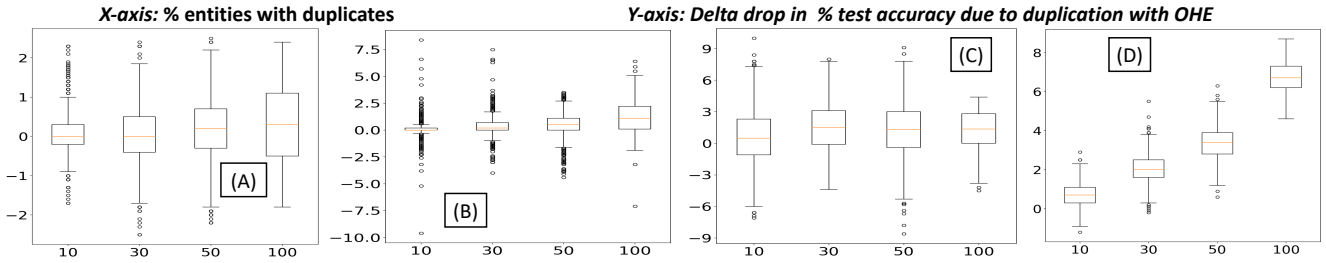

Figure 5: `AllA` results with *OHE* for (A) `LR` (B) `ShallowDT` (C) `LoCapMLP` (D) `HiCapMLP` with the same setup as Figure 4(B).

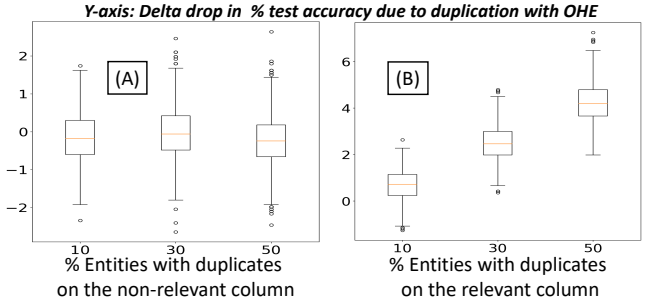

Figure 6: `AllA` results on `HiCapRF`. We set $|\mathcal{A}| = 4$ and vary $|ED|/|E|$, while fixing $(|r|_t, occ(D_k), |D_k|)$=(5000, 25, 1). Duplicates introduced on the column with (A) non-positive *Relevancy* (noisy column) (B) high *Relevancy* (predictive column).

5.1.3. 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 in the same set D_k are more likely to occur than others. 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 in Section 5.1.1 and 5.1.2 change or get invalidated with this new setup.* Due to space constraints, we present the accuracy plots in tech report [65].

5.1.4. 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 the downstream 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 14 (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 14 (B-C)).* This simply 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 used all the columns in the dataset as part of 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. Thus, we have the presence of both high and low *Relevancy* columns in the dataset. We 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 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 even observe the same trend with HiCapMLP.*

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, $1<=i<=|\mathcal{A}|$. Each weight W_i with cardinality $|E|$ is randomly sampled from a list of non-zero integers ($[-5, 5]\backslash\{0\}$) without replacement. Note that the integer weights are chosen only 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 $|r|$ tuples of $\mathcal{A}$ by sampling values uniformly randomly from $|E|$. Thus, with fixed weights, the hyperplane over *One-hot* encoded example feature vectors is given by $\sum_{i=1}^{i=|\mathcal{A}|} W_i \cdot A_i = 0$. (3) Examples for which $\sum_{i=1}^{i=|\mathcal{A}|} W_i \cdot A_i >= 0$ are labeled positive (Y=0) and 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 accuracy due to duplicates with all models using *OHE*. We find that all high-bias approaches are again robust to the presence of duplicates even when all entities are diluted with duplicates. Interestingly, `HiCapMLP` exhibits only marginal impact with duplicates. In contrast, duplicates affect `HiCapRF` significantly, especially in a high-duplication regime. We explain this interesting behavior in Section 5.3. We vary other confounders such as the other duplication parameters, the fraction of entities being mapped to "*Others*," and column *Relevancy*. We confirm the same trends that we saw with all models in `AllA` scenario, except with `HiCapMLP` which behaves similar to `LR` than `HiCapRF`. We present all the plots in the technical report [65].

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 models with the overfitting gap. Figure 8 presents the overfitting gap results of all classifiers with *OHE* on the `AllA` scenario. We find that the delta drop in accuracy (Figure 14) closely follows the increase in

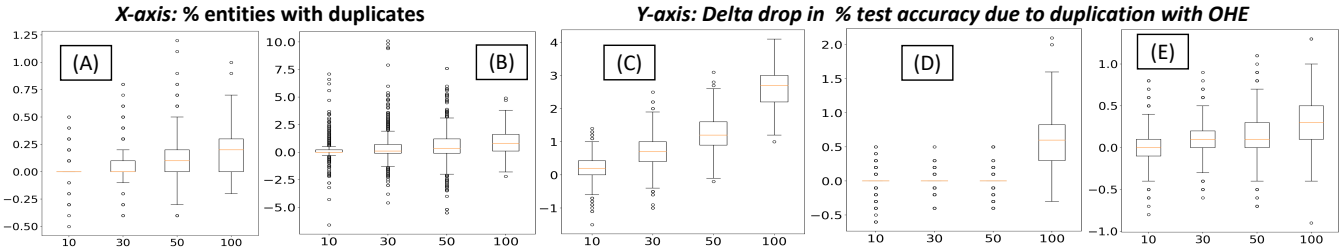

Figure 7: `Hyperplane` setting results on (A) `LR` (B) `ShallowDT` (C) `HiCapRF` (D) `LoCapMLP` (E) `HiCapMLP` (same setup as Figure 5).

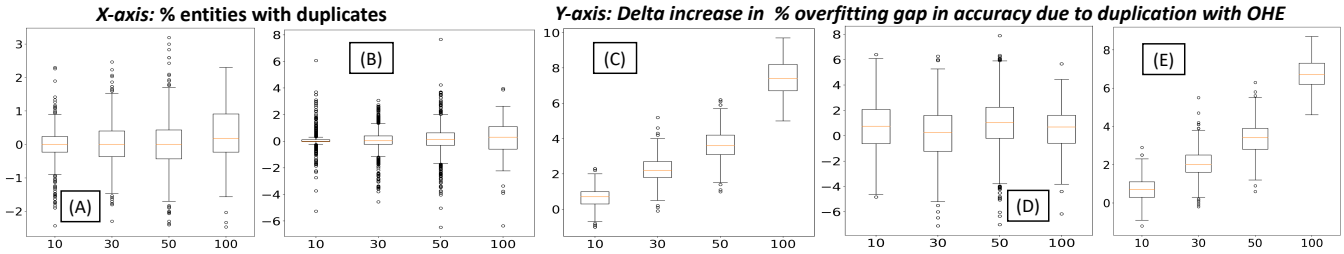

Figure 8: `AllA` setting results on (A) `LR` (B) `ShallowDT` (C) `HiCapRF` (D) `LoCapMLP` (E) `HiCapMLP` (with the same setup as Figure 4(B)).

the overfitting gap due to duplicates with both high-capacity models, `HiCapRF` and `HiCapMLP`. 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 the high-capacity models, which are prone to overfitting.* However, as the number of training examples rises, the amount of overfitting subsides. This explains our trends in the high-data regime.

Now with the `Hyperplane` setting, `LR` exhibits no amount of extra overfitting with duplicates. This is because the VC dimension of the `LR` is linear in the number of features. As the dimensionality of the feature space expands with duplicates, the 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 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.* Also with `MLP`, each neuron functioning as a hyperplane classifier can easily recover the true hyperplane. Thus, even `HiCapMLP` doesn't overfit as much on the `Hyperplane` setting. In contrast, `HiCapRF` 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 `HiCapRF` in the `Hyperplane` setting compared to `AllA`. *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.*

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. `LR` is less prone to overfitting than `RF` and `MLP` when 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* inherently exploits the presence of similarly valued duplicates in the *Categoricals*. This makes it significantly more robust from duplicates compared to *OHE* and *StrE*. Moreover, unseen duplicates that arise during the deployment phase can degrade ML performance

with *OHE* or *StrE*. Overall, *Similarity* encoding and/or a Logistic Regression 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 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. Simulate duplications in your data with the synthetic suite. Our synthetic simulation suite provides an empirical methodological infrastructure for understanding the category deduplication effect in presence of different confounders. 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, existing open source AutoML tools such as AutoGluon [34] and TransmogrifAI [9] do not 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. *Our labeled dataset presents a learning-based approach to automate deduplication. Moreover, this will lead to an objective assessment of the accuracy of automation.* Capturing semantic-level characteristics of the categories with either designing features or with deep learning models like Siamese neural network [56] is an important avenue for future research.

2. Define new benchmark tasks. In an AutoML production setting with millions of features, one cannot possibly perform deduplication over all columns. Given a cleaning budget 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 classifier to help identify them, but how does a column-level featurization look like? Our labeled data can be leveraged at a column-level to design such a coarse-grained classifier.

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 [71]? 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 [51] analyses the impact of many data cleaning steps on downstream ML tasks. However, they do not specifically cover *Categorical* deduplication. Although they study string-level inconsistencies within a column with four real datasets, they are not all *Categorical*. They focus on deriving a broad perspective of many cleaning steps such as this. In contrast, we offer empirical depth with confounder characterization to study the impact of *Categorical* duplicates on ML. AutoML benchmark [36] 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 standpoint is not their focus. We performed an objective benchmarking of a specific ML data prep step, namely the feature type inference task [64]. We build upon our open-sourced datasets but we study a completely different problem.

AutoML Platforms. Several AutoML tools allow users to perform automated ML algorithm and hyper-parameter search without covering any data prep tasks [35, 59, 70]. Other AutoML platforms that offer (or claim to offer) end-to-end ML support such as Salesforce TransmogrifAI [9], Google AutoML Tables [4], and Amazon AutoGluon [34] 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 contributions from community to automate the deduplication task, including potentially extending AutoML with deduplication processor in the optimization process [35, 57, 58]. Moreover, we believe that our downstream benchmark, our analysis with simulation studies, and takeaways are all valuable to improve AutoML platforms.

Data Prep and Cleaning for ML. There exists numerous data prep tools such as rule-based tools [42], exploratory data analysis-based libraries [62], visual interfaces [10], and program synthesis-based tools [40, 43] to reduce user's manual grunt work effort and allow them to productively prepare their data for ML. In addition, Auto-insight generator tools [31, 32, 72] 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 [47, 48]. 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 [60, 67, 68], learning-based [45, 52, 55, 75], semi-supervised [44], unsupervised [73], and active-learning [54] approaches. Our focus is on analyzing the impact of category duplicates on 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. Moreover, active learning-based string matching solution [27] is truly complementary as it can enhance our benchmark by reducing the labeling effort for users.

REFERENCES

[1] Accessed May 29, 2022. Github Repository for studying the impact of Cleaning Category Duplicates on ML, https://github.com/anon-categdups/CategDedupRepo.

[2] Accessed May 29, 2022. 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 May 29, 2022. Google AutoML Tables Data Prep User Guidelines, https://cloud.google.com/automl-tables/docs/data-best-practices.

[4] Accessed May 29, 2022. Google Cloud AutoML, https://cloud.google.com/automl/.

[5] Accessed May 29, 2022. H2O Driverless AI, https://www.h2o.ai/products/h2o-driverless-ai/.

[6] Accessed May 29, 2022. H2o.AI, https://www.h2o.ai/.

[7] Accessed May 29, 2022. Microsoft AutoML, https://azure.microsoft.com/en-us/services/machine-learning/automatedml/.

[8] Accessed May 29, 2022. Similarity Encoder Library, https://github.com/dirty-cat/dirty_cat.

[9] Accessed May 29, 2022. TransmogrifAI: Automated Machine Learning for Structured Data, https://transmogrif.ai/.

[10] Accessed May 29, 2022. Trifacta: Data Wrangling Tools & Software, https://www.trifacta.com/.

[11] Accessed May 29, 2022. *https://data.ca.gov/dataset/tsm-habitat-rapid-assessment-survey-2016-ds28271.*

[12] Accessed May 29, 2022. *https://datacatalog.hsls.pitt.edu/dataset/77.*

[13] Accessed May 29, 2022. *https://data.cityofchicago.org/Buildings/Vacant-and-Abandoned-Buildings-Violations/kc9i-wq85.*

[14] Accessed May 29, 2022. *https://data.cityofchicago.org/Transportation/Relocated-Vehicles/5k2z-suxx.*

[15] Accessed May 29, 2022. *https://data.ny.gov/Energy-Environment/Utility-Company-Customer-Service-Response-Index-CS/w3b5-8aqf.*

[16] Accessed May 29, 2022. *https://everydaydata.co/2017/02/07/hacker-news-part-one.html.*

[17] Accessed May 29, 2022. *https://github.com/fivethirtyeight/data/tree/master/region-survey.*

[18] Accessed May 29, 2022. *https://maxcandocia.com/article/2018/Oct/22/trick-or-treating-ages/.*

[19] Accessed May 29, 2022. *https://osmihelp.org/research.*

[20] Accessed May 29, 2022. *https://transparentcalifornia.com/salaries/san-francisco/.*

[21] Accessed May 29, 2022. *https://www.asdcode.de/2021/01/it-salary-survey-december-2020.html.*

[22] Accessed May 29, 2022. *https://www.kaggle.com/definitelyliliput/rawscores.*

[23] Accessed May 29, 2022. *https://www.kaggle.com/mlomuscio/wifi-study.*

[24] Accessed May 29, 2022. *https://www.kaggle.com/pushpaltayal/etailing-customer-survey-in-india.*

[25] 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

[26] 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

[27] 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

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

[29] 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

[30] 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

[31] 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

[32] 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

[33] 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

[34] 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

[35] 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

[36] 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).

[37] Margherita Grandini, Enrico Bagli, and Giorgio Visani. 2020. Metrics for multi-class classification: an overview. *arXiv preprint arXiv:2008.05756.*

[38] John T. Hancock and Taghi M. Khoshgoftaar. 2020. Survey on categorical data for neural networks. *J. Big Data* 7, 1 (2020), 28. https://doi.org/10.1186/s40537-020-00305-w

[39] 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

[40] 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.

[41] 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

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

[43] 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.

[44] 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

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

[46] Hanna Köpcke, Andreas Thor, and Erhard Rahm. 2010. Evaluation of entity resolution approaches on real-world match problems. *Proc. VLDB Endow.* 3, 1 (2010), 484–493. https://doi.org/10.14778/1920841.1920904

[47] 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

[48] 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

[49] Angela Lee, Doris Xin, Doris Lee, and Aditya G. Parameswaran. 2020. Demystifying a Dark Art: Understanding Real-World Machine Learning Model Development. *CoRR* abs/2005.01520 (2020). arXiv:2005.01520 https://arxiv.org/abs/2005.01520

[50] 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

[51] 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).

[52] 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

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

[54] 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

[55] 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

[56] 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

[57] 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

[58] 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.

[59] 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

[60] 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

[61] 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.

[62] 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.*

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

[64] 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.

[65] Vraj Shah, Thomas Parashos, and Arun Kumar. Accessed May 29, 2022. *How do Categorical Duplicates Affect ML? A New Benchmark and Empirical Analyses (Technical Report).* https://adalabucsd.github.io/papers/TR_2022_CategDedup.pdf.

[66] 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

[67] 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

[68] 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

[69] Survey. Accessed May 29, 2022. 2021 State of Data Science and Machine Learning. https://www.kaggle.com/kaggle-survey-2021.

[70] 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.

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

[72] 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

[73] 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

[74] 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

[75] 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

APPENDIX

A EXISTING EM DATASETS

Labeled datasets used in the entity deduplication literature such as Magellan [45] involve duplicates at a tuple-level. But this is an orthogonal problem to category deduplication. Tuple-level duplicates do not necessarily imply duplication in the *Categorical* strings. Likewise, duplication in a *Categorical* column may not lead to row-level duplicates. We present three pairs of duplicate records from three different datasets of the Magellan data repository in Table 10. Note that generic open domain attributes such as author 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. Moreover, *Title* in Citations datasets has rich semantic information and is typically used as a *Text* or *Sentence* type feature. A *Categorical* feature assumes mutually exclusive values from a known finite domain set. We find that almost all of the Magellan datasets involve duplication in non-*Categorical* features such as generic person names, company names, addresses, and textual values with rich semantic information. Thus, they are not relevant for us to study category deduplication. We focus exclusively on the *Categorical* features and curate the first labeled dataset of entities annotated with duplicates within a *Categorical* column.

B DOWNSTREAM BENCHMARK

B.1 Datasets

Table 11 presents the statistics with different confounders that can potentially impact the ML performance over all 14 downstream datasets. For the downstream benchmark, we downloaded as many datasets as possible from public sources and looked for columns with duplicates in them. We then made sure that our selected datasets sufficiently represents different extremes in the confounder

Table 10: Three pairs of duplicate tuples from three different datasets of the Magellan data repository [45].

Dataset Name	Left Tables			Right Tables		
Restraunt	**Address**	**Phone number**	**Name**	**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
Ebooks	**Author**	**Title**	**Price**	**Author**	**Title**	**Price**
	John D.T. White	101 Things You May Not Have Known About the US Masters	5.99	John White	101 Things You May Not Have Known About the US Masters	5.49
Citations	**Author**	**Entry Type**	**Title**	**Author**	**Entry Type**	**Title**
	David A. Cohn and Zoubin Ghahramani and Michael I. Jordan	article	Active Learning with Statistical Models	Cohn, David A and Ghahramani, Zoubin and Jordan, Michael I	article	Active learning with statistical models

Table 11: Statistics of the column containing duplicates in our downstream benchmark datasets. $|r|$, $|\mathcal{A}|$, and $|Y|$ are the total number of examples, number of columns, and number of target classes in the given dataset respectively. $|rC|$ 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 ($1-|E|/|C|$<0.25), moderate ($1-|E|/|C|$ >0.25 & <0.50), and high ($1-|E|/|C|$ >0.50) respectively. 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 deduplication as category set size has shrunk.

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

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. While the former dataset is robust to duplicates even with almost 51% of their column's entity diluted with duplicates, 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. We do not claim that the 14 downstream datasets are highly 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 datasets plainly to showcase different confounder settings and study the behavior of duplicates in those settings. The benchmark suite helps us lay out the confounders that matter. This coupled with synthetic study only serves as a guidebook that can help ML practitioners and AutoML platform developers glean insights.

Table 12: Comparison of downstream models in terms of Macro F1 score with different *Categorical* encoding schemes on *Raw* (column with duplicates) vs. *Deduped* (deduplicated column) data. Results for *Deduped* are shown relative to the *Raw* as delta drop in % F1 score. Green, blue, and red colors denote cases where the *Deduped* F1 score relative to *Raw* is significantly higher, comparable, and significantly lower (error tolerance of 1%) respectively.

Dataset	Logistic Regression				Random Forest						MLP			
	OHE		*SimE*		*OHE*		*StrE*		*SimE*		*OHE*		*SimE*	
	Raw	*Deduped*	*Raw*	*Deduped*	*Raw*	*Deduped*	*Raw*	*Deduped*	*Raw*	*Deduped*	*Raw*	*Deduped*	*Raw*	*Deduped*
Midwest Survey	55.7	+10.3	65.4	+2.7	44.9	+12.6	56.5	+11.7	63.4	+5	54.3	+9.7	63.3	+3.7
Mental Health	42	-0.6	40.1	+0.8	40.3	+0	39.3	-1.3	38	+0.8	39.3	+2.7	41.1	+0.5
Relocated Vehicles	82.8	+4	88.4	+0.4	71.6	+3.5	81.3	+3.7	88.3	-0.1	83.6	+3.6	89.5	+0
Health Sciences	56.1	+0.7	57.4	+2.2	51.5	+3.3	59.1	-0.5	59.2	-3.7	54.7	+5.4	56.6	+1.8
Salaries	27.4	+1.3	30.5	-2	57.6	+2.1	64.5	+1.9	93.8	+0.4	15.9	+3.4	14.7	+8.7
TSM Habitat	34.1	+0	34.1	+0	68.5	+0	82.2	+1.6	85.7	+0.6	24.6	+4	19.1	+12.3
EU IT	16.1	+0	16.1	+0	33.6	+1.1	36.8	+0.2	43.1	+2.7	9.3	-2.2	4.2	+4.8
Halloween	37.1	+3.8	45.7	+0.5	34.2	+3	33.2	+1.7	31.1	-4.9	38.8	+4.1	40.9	-0.6
Utility	37	+0.1	38.5	+0.3	58.2	+1.5	44.9	+1.4	41.8	+1.7	65.2	+2	73.4	+2.2
Mid or Feed	37.2	+0.2	39.1	-3.6	35.2	+1.8	26.6	-0.3	26.1	+2.6	33	+1.9	31.2	+0.4
Wifi	54.9	+1.5	50.7	+8.2	52.7	+8.5	54.3	-4.5	50.2	+1.9	51.6	+2.3	48.3	+2.6
Etailing	37.2	-2.3	37.5	-0.1	33.3	+3	36.3	+1.4	32.9	+3.1	39.4	-3.1	36.7	+0
San Francisco	85.9	-0.1	85.6	-0.1	83	+0.3	83.3	+0.3	86.1	-0.1	86	+0.1	86	+0.1
Building Violations	89.4	+0	89.3	+0.1	97.5	-0.1	97	+0.1	97.4	+0.1	97.5	+0.1	97.2	+0.4

Table 13: Delta drop in % macro F1 overfitting gap of the ML models with *OHE*. The overfitting gap for *Deduped* is shown relative to the *Raw*.

Dataset	LR		Random Forest		MLP	
	Raw	*Deduped*	*Raw*	*Deduped*	*Raw*	*Deduped*
Midwest Survey	25.3	-10	55	-15.8	45.5	-10.6
Mental Health	12.5	-3.4	49.8	-7.9	28.9	+1.5
Relocated Vehicles	17	-4.1	28.2	-3.7	16.4	-3.6
Health Sciences	7.5	-6.5	35.1	-8.6	45.3	-5.4
Salaries	2.3	-0.8	41.6	-1.4	0.8	+0.6
TSM Habitat	2.1	-0	30.6	-4.8	1.3	+0.2
EU IT	0.1	-0	60	-6.8	1	+1.1
Halloween	40.4	-3.8	56.2	-7.2	61.2	-4.1
Utility	0.1	-0.9	41.8	-1.5	26	-2.9
Mid or Feed	35.7	-12.4	63.3	-0.4	67	-1.9
Wifi	11.9	-2.6	31.8	-4.8	46.8	+0.8
Etailing	43.3	-6.7	60.6	-2.3	60.6	+3
San Francisco	0.6	-0.2	0.1	+0.1	1.1	-0.1
Building Violations	0.1	+0.1	1.8	+0.1	1.2	-0.2

B.2 Methodology

Experimental Setup. We use CloudLab [33] with custom OpenStack profile running Ubuntu 18.04, 10 Intel Xeon cores, and 160GB of RAM. We use python scikit-learn [61], H2O [6], and SimilarityEncoder [8] packages to implement *OHE*, *StrE*, and *SimE* respectively. We use a standard grid search for hyper-parameter tuning, with the grids described in detail below.

Logistic Regression: There is only one regularization parameter to tune: *C*. Larger the value of C, lower is the regularization strength, hence increasing the complexity of the model. The grid for *C* is set as $\{10^{-3}, 10^{-2}, 10^{-1}, 1, 10, 100, 10^{3}\}$.

Random Forest: There are two hyper-parameters to tune: *NumEstimator* and *MaxDepth*. *NumEstimator* is the number of trees in the forest. *MaxDepth* is the maximum depth of the tree. The grid is set as follows: *NumEstimator* $\in \{5, 25, 50, 75, 100\}$ and *MaxDepth* $\in \{5, 10, 25, 50, 100\}$.

MLP: The multi-layer perceptron architecture comprises of 2 hidden units with 100 neurons each. We do L_2 regularization with the regularization parameter tuned using the following grid axis: $\{10^{-4}, 10^{-3}, 10^{-2}\}$.

B.3 Results with Additional Evaluation Metrics

We check if using additional evaluation metrics such as F1 score, precision, and recall alter any empirical observations or conclusions in Section 4.4 of the paper. Note that the micro average of precision, recall, and F1-score is identical to the accuracy of multi-class classification [37], which we already reported in Tables 6 and 7. Thus, we use the macro average of precision, recall, and F1-score [37] as evaluation metrics and rerun our downstream benchmark suite (Section 4). We check if evaluating with macro F1 score alters the

Table 14: Summary statistics over 14 downstream datasets in terms of Macro F1-score, precision, and recall to showcase the impact of deduplication on ML models using different encoding schemes.

F1 score	LR		Random Forest			MLP	
	OHE	*SimE*	*OHE*	*StrE*	*SimE*	*OHE*	*SimE*
	% lift in accuracy with Deduped						
Mean	1.4	0.7	2.9	1.4	0.7	2.4	2.6
Median	0.2	0.2	2	1.4	0.7	2.5	1.2
75th percentile	1.5	0.7	3.2	1.9	2.5	3.9	3.4
Max	10.3	8.2	12.6	11.7	5	9.7	12.3
	# downstream datasets where						
>1% lift wrt metric on *Deduped*	5	3	10	8	6	10	7

Precision	LR		Random Forest			MLP	
	OHE	*SimE*	*OHE*	*StrE*	*SimE*	*OHE*	*SimE*
	% lift in accuracy with Deduped						
Mean	1.8	1.1	4.2	1.2	0.7	1.7	1.9
Median	0.9	0.3	3.2	1	0.5	1.9	1.7
75th percentile	2.9	2.1	6.2	1.9	2.8	3.4	2.8
Max	10.3	8.1	14.7	11.9	5.8	9.8	9.8
	# downstream datasets where						
>1% lift wrt metric on *Deduped*	7	4	10	7	6	9	8

Recall	LR		Random Forest			MLP	
	OHE	*SimE*	*OHE*	*StrE*	*SimE*	*OHE*	*SimE*
	% lift in accuracy with Deduped						
Mean	1.6	0.9	2.4	1.9	0.7	2.3	2.7
Median	0.9	0.4	1.6	1.3	0.6	2.1	1.3
75th percentile	1.6	1.1	2.4	2.1	2.7	4.1	3.7
Max	9.4	8.4	10.8	10	4.4	9.5	15
	# downstream datasets where						
>1% lift wrt metric on *Deduped*	7	4	10	9	6	9	7

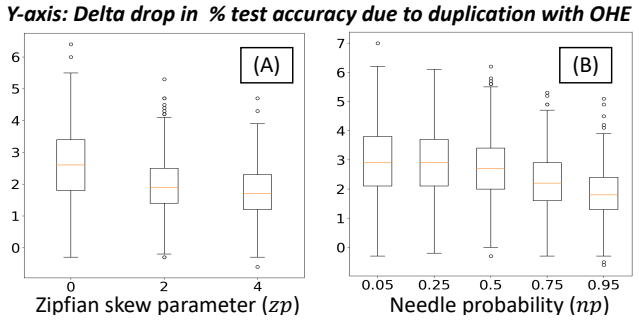

Figure 9: Effects of skew parameters on `AllA` simulation scenario for Random Forest with *OHE*. $|A|$ is preset to 3. (A) Vary Zipfian skew parameter zp of $|G_k|$, while fixing $(|r|_t, |ED|/|E|) = (3000, 30)$. (B) Vary needle probability parameter np of $occ(G_k)$, while fixing $(|r|_t, |ED|/|E|, zp) = (3000, 30, 2)$.

conclusion made with % diagonal classification accuracy as the metric in regard to the varied confounder.

Overall, we find that none of the empirical conclusions made with diagonal accuracy in the paper change even with these additional evaluation metrics. Table 12 presents the comparison of downstream models with different *Categorical* encoding schemes in terms of macro F1 scores. Table 14 (similar to Table 8) showcases the aggregate statistics over all evaluation metrics. Finally, we present the generalization performance of the ML classifiers with the overfitting gap, difference between train and validation macro F1 scores in Table 13 here (similar to Table 9). We confirm the validity of all observations *O1-O8* (Section 4.4 of the paper) made with the downstream benchmark suite with the additional evaluation metrics. As an example, we still find that the delta increases in macro F1-score, precision, and recall with *Deduped* are higher with Random Forest and Multi-layer Perceptron (MLP) than Logistic Regression. Moreover, we again find that *Similarity* encoding is more robust than other encoding schemes to tolerate duplicates. Note that the focus of this work is on interpreting the impact on ML with features having different duplication properties and not on disentangling the impact at a per-class basis.

C SIMULATION STUDY RESULTS

C.1 Scenario AllA

C.1.1 Varying the data regime and the parameters that control the amount of duplication. Figure 12 shows the delta drop in classification accuracy with duplication relative to the ground truth clean dataset on all models with *OHE* as the number of training examples and the duplication confounders are varied. Figure 13 shows the results as the confounders are varied for `HiCapRF` with *StrE*. We again note that a high-data regime is more robust to duplication than a low-data regime for both encoding schemes.All the high-bias approaches are more robust to duplication than the high-capacity models. Also, duplication confounders can have significant adverse performance effects on high-capacity classifiers

C.1.2 Skewness in the duplication parameters. We now alter our duplication process to capture skewness in $|D_k|$ and $occ(D_k)$, varying one at a time. Figure 9 presents the results. We find that the delta drop in % accuracy due to duplicates remains significant regardless of the amount of skew in $|D_k|$ and $occ(D_k)$. With the Zipfian skew in $|D_k|$, the delta drop is highest at uniform distribution in $|D_k|$ (no skew setting) and marginally decreases as the skewness parameter is increased. On the other hand, when a needle-and-thread skew in $occ(D_k)$ is present, one duplicate from set D_k has a probability mass np (needle parameter). The remaining 1-np probability mass is uniformly distributed over the rest of the duplicates in D_k. We find that the delta drop due to duplicates decreases while still remaining significant when one duplicate value is more likely to occur than the rest (as np is increased). *Overall, the overarching conclusion from this analysis is that none of our results or takeaways change or get invalidated with this new setup.*

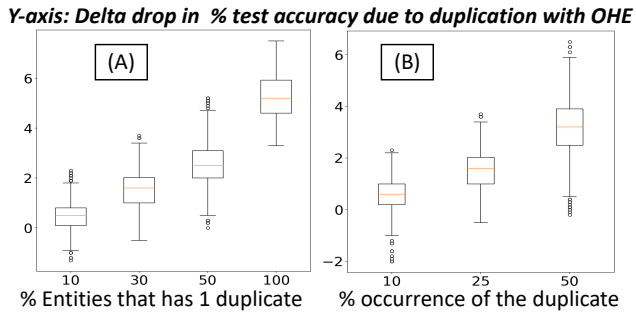

Figure 10: `Hyperplane` scenario results on *HiCapRF* with *OHE*. Only test set is diluted with duplicates. (A) Vary $|ED|/|E|$, while fixing $(|r|_t, occ(D_k), |D_k|) = (3000, 25, 1)$. (B) Vary $occ(D_k)$, while fixing $(|r|_t, |ED|/|E|, |D_k|) = (3000, 30, 1)$.

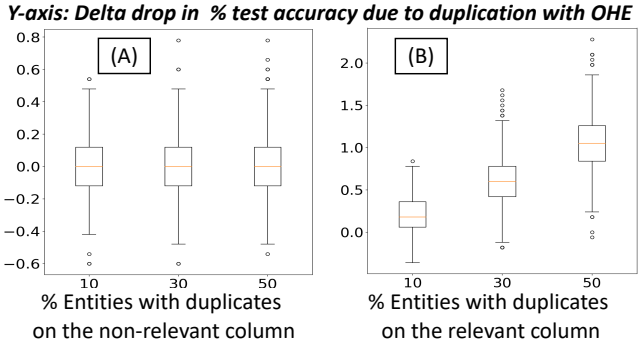

Figure 11: `Hyperplane` results on `HiCapRF`. We set $|\mathcal{A}| = 4$ and vary $|ED|/|E|$, while fixing $(|r|_t, occ(D_k), |D_k|)$=$(5000, 25, 1)$. Duplicates introduced on the column with (A) non-positive *Relevancy* (noisy column) (B) high *Relevancy* (predictive column).

C.2 Scenario Hyperplane

C.2.1 Varying the data regime and the parameters that control the amount of duplication. Figure 15 presents the delta drop in classification accuracy due to duplicates with all models using *OHE*. We again note that as the number of available training examples is increased, the delta drop in accuracy due to duplicates decreases for `HiCapRF`. Raising the other duplication parameters such as $|ED|/|E|, occ(D_k), |D_k|$ also increases the adverse performance impact of duplicates on `HiCapRF`. Interestingly, we find that `HiCapMLP` exhibits only a marginal amount of overfitting on the `Hyperplane` simulation scenario. Thus, we do not see any impact due to duplicates on `HiCapMLP` and also on all the high-bias approaches.

C.2.2 Varying properties of duplicates being mapped to "Others" and column Relevancy. We now repeat the simulation scenario presented in Section 5.1.3 but with `Hyperplane` setting, i.e. the true distribution is given by a hyperplane that separates the classes. Figure 10 presents the results when only the test set is diluted with duplicates. We again note that the presence of duplicates in the test set impacts *HiCapRF* significantly. Figure 11 presents the `Hyperplane` simulation results when we have the presence of both high and low relevancy columns in the dataset (setup same as Section 5.1.5). We again find that the duplication on a noisy column has a marginal impact on `HiCapRF`, while duplicates the on relevant column affect it significantly.

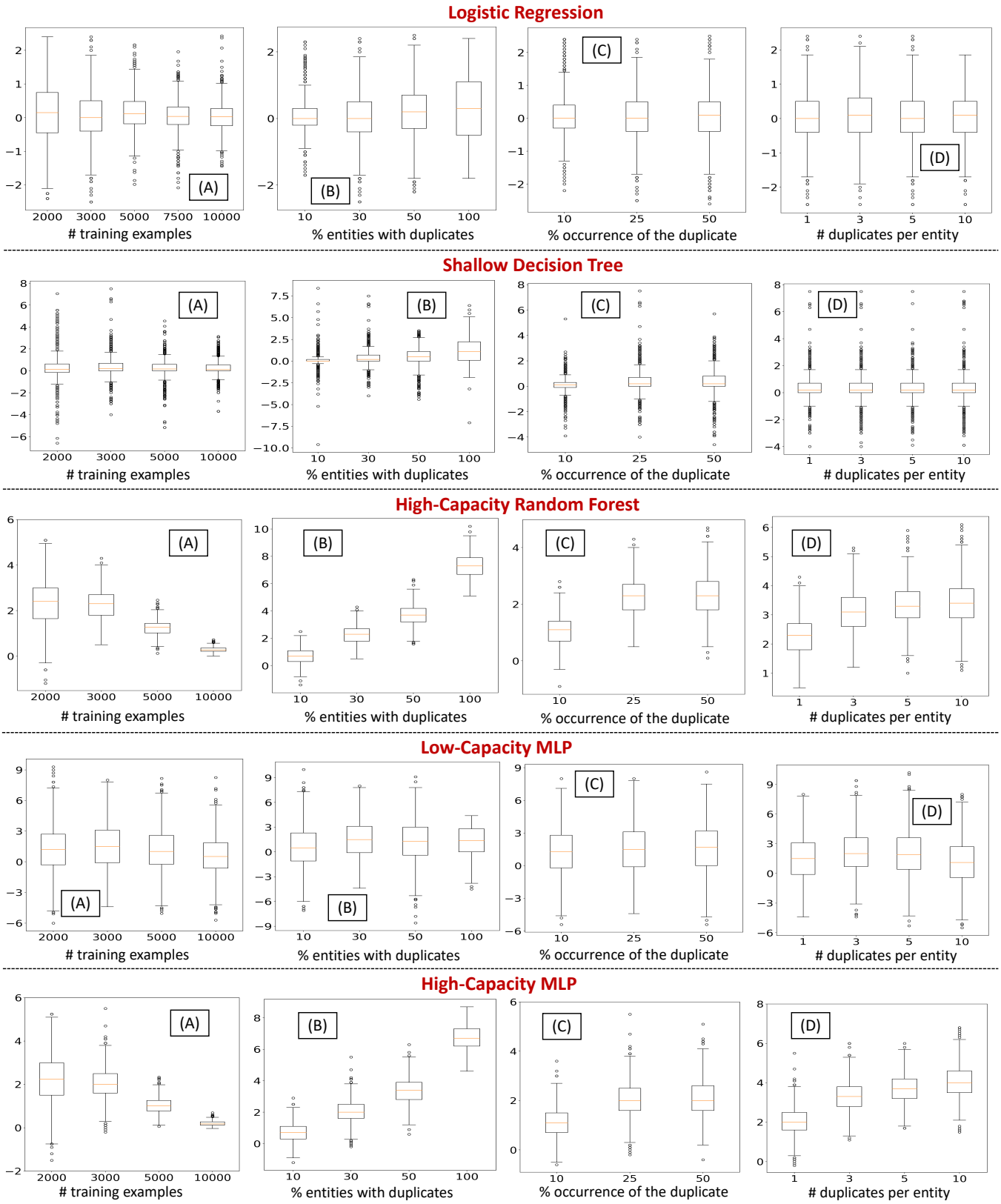

Figure 12: `AllA` simulation scenario results for `LR`, `ShallowDT`, `HiCapRF`, `LoCapMLP`, and `HiCapMLP` with *OHE*. $|X| = 3$. (A) Vary $|r|_t$ (# training examples) , while fixing $(|ED|/|E|, occ(D_k), |D_k|) = (30, 25, 1)$. (B) Vary $|ED|/|E|$, while fixing $(|r|_t, occ(D_k), |D_k|) = (3000, 25, 1)$. (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 $k \in [1, |ED|]$.

Figure 13: `AllA` simulation scenario results for `ShallowDT` and `HiCapRF` with *StrE*. (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 $k \in [1, |ED|]$.

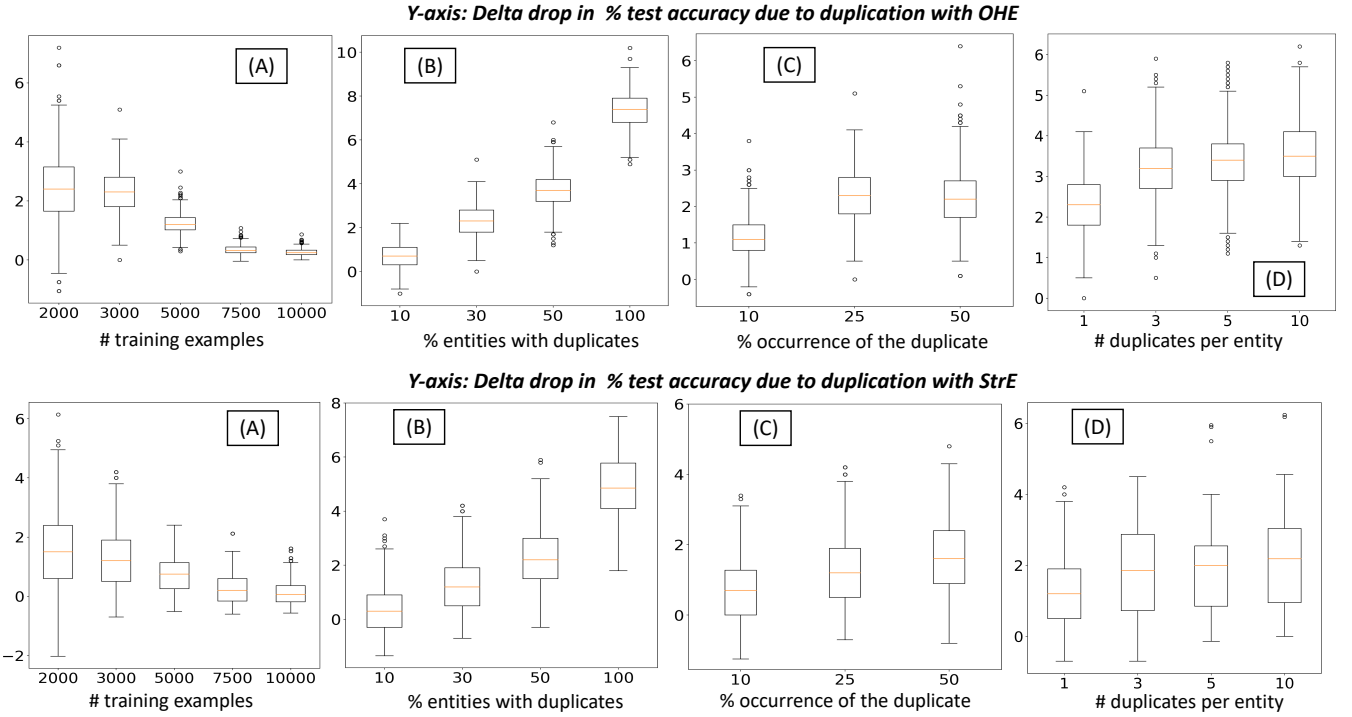

Figure 14: `AllA` simulation scenario results for Random Forest with *OHE* and *StrE* where hyper-parameters are tuned with grid search. $|X| = 3$. (A) Vary $|r|_t$ (# training examples) , while fixing $(|ED|/|E|, occ(D_k), |D_k|) = (30, 25, 1)$. (B) Vary $|ED|/|E|$, while fixing $(|r|_t, occ(D_k), |D_k|) = (3000, 25, 1)$. (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 $k \in [1, |ED|]$.

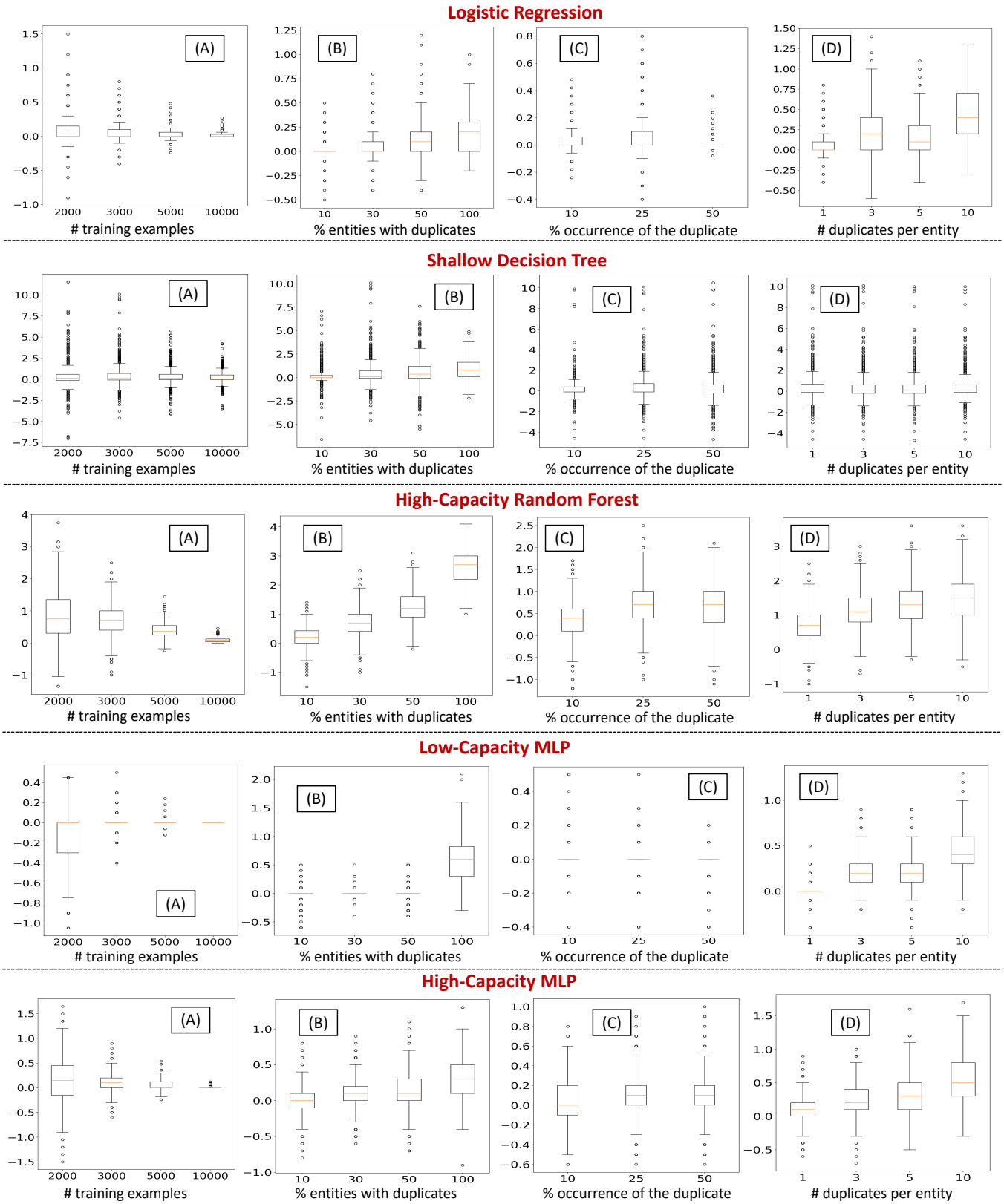

Figure 15: `Hyperplane` simulation scenario results for `LR`, `ShallowDT`, `HiCapRF`, `LoCapMLP`, and `HiCapMLP` with *OHE*. $|X| = 3$. **(A) Vary** $|r|_t$ **(# training examples) , while fixing** $(|ED|/|E|, occ(D_k), |D_k|) = (30, 25, 1)$. **(B) Vary** $|ED|/|E|$**, while fixing** $(|r|_t, occ(D_k), |D_k|) = (3000, 25, 1)$. **(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** $k \in [1, |ED|]$.