To Join or Not to Join?
Thinking Twice about Joins before Feature Selection

Arun Kumar   Jeffrey Naughton   Jignesh M. Patel   Xiaojin Zhu

Department of Computer Sciences,
University of Wisconsin-Madison
{arun, naughton, jignesh, jerryzhu}@cs.wisc.edu

ABSTRACT

Closer integration of machine learning (ML) with data processing is a booming area in both the data management industry and academia. Almost all ML toolkits assume that the input is a single table, but many datasets are not stored as single tables due to normalization. Thus, analysts often perform key-foreign key joins to obtain features from all base tables and apply a feature selection method, either explicitly or implicitly, with the aim of improving accuracy. In this work, we show that the features brought in by such joins can often be ignored without affecting ML accuracy significantly, i.e., we can "avoid joins safely." We identify the core technical issue that could cause accuracy to decrease in some cases and analyze this issue theoretically. Using simulations, we validate our analysis and measure the effects of various properties of normalized data on accuracy. We apply our analysis to design easy-to-understand decision rules to predict when it is safe to avoid joins in order to help analysts exploit this runtime-accuracy trade-off. Experiments with multiple real normalized datasets show that our rules are able to accurately predict when joins can be avoided safely, and in some cases, this led to significant reductions in the runtime of some popular feature selection methods.

1. INTRODUCTION

The increasing use of machine learning (ML) in data-driven applications [2, 4] has led to a growing interest in more closely integrating ML and data processing [3, 16, 19, 23, 25, 47]. However, almost all ML toolkits assume that the input to an ML model is a single table even though many real-world datasets are stored as multiple tables connected by primary key-foreign key (KFK) dependencies. Thus, analysts typically perform KFK joins before ML to construct a single table that collects features from all base tables and then apply a feature selection method (either explicitly or implicitly [17]) over the entire set of features. Feature selection helps improve ML accuracy and is widely considered crucial for ML-based analytics [1, 17, 23, 46]. While this process certainly "works," it can be both painful and wasteful because the increase in the number of features might make it harder for analysts to explore the data and also increases the runtime of ML and feature selection methods. In some cases, the joins might also be expensive and introduce data redundancy, causing even more efficiency issues [29].

In this work, we help mitigate the above issues by studying a rather radical question: *Is a KFK join even needed for ML?* In other words, is it possible to ignore all "foreign" features (the features from the table referred to by the foreign key) in the first place without significantly reducing ML accuracy? We call this process "avoiding the join." At first, this seems preposterous: how can we be confident that ignoring some features is unlikely to reduce accuracy significantly without even running the feature selection method over the data (which requires the join)? The key turns out to be a rather simple observation: the KFK dependencies present in the *schema* enable us to avoid joins. Simply put, in an information theoretic sense [17], a foreign key encodes "all information" about all the foreign features brought in by a KFK join, which allows us to use it as a "representative" for the foreign features. Thus, this observation seems to make things stunningly simple: ignore all KFK joins and use foreign keys as representatives of foreign features!

Alas, the real world is not as simple as described above. Unfortunately, the information theoretic perspective is not sufficiently perspicacious to fully answer our core question of when it is safe to avoid a join. The finite nature of training datasets in the real world makes it necessary to analyze our problem using the standard ML notions of bias and variance [40]. This requires detailed, yet subtle, theoretical analysis (which we perform) of the effects of KFK joins on ML. It turns out that foreign features, which are safe to ignore from an information theoretic perspective, could be indispensable when both bias and variance are considered. This brings us back to square one with our conundrum: *given a KFK join, how to tell if it is safe to avoid or not?*

Answering the above core question could yield at least four benefits. First, it can help improve the performance of ML tasks without losing much accuracy. Second, in applications in which analysts explore features by being in the loop [27, 46], having fewer tables and features might make exploration easier. Third, it might help reduce the costs of data acquisition in applications where new tables (e.g., weather data) are purchased and joined with existing data. If we can show that such joins might not really help accuracy, analysts can reconsider such purchases. Fourth, in some applications, analysts have dozens of tables in the input and

SIGMOD'16, June 26-July 01, 2016, San Francisco, CA, USA
© 2016 ACM. ISBN 978-1-4503-3531-7/16/06...$15.00
DOI: http://dx.doi.org/10.1145/2882903.2882952

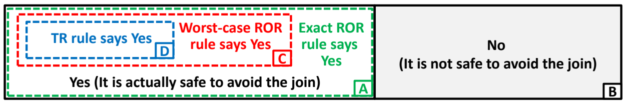

TR rule says Yes
D
Worst-case ROR rule says Yes
C
Exact ROR rule says Yes
Yes (It is actually safe to avoid the join)
A
No
(It is not safe to avoid the join)
B

Figure 1: Illustrating the relationship between the decision rules to tell which joins are "safe to avoid."

prefer to join only a few "most helpful" tables (colloquially called *source selection*). Answering our question might help analysts assess which tables matter less for accuracy.

In this paper, we show that it is possible to answer our core question by designing practical heuristics that are motivated by our theoretical understanding. Thus, apart from establishing new connections between joins and ML, our work can help make feature selection over normalized data easier and faster. Indeed, the data management community is increasingly recognizing the need for more of such formal and systems support to make feature selection easier and faster [7, 27, 36, 46]. Ideally, we desire a *decision rule* that can help answer the question for analysts. Apart from being effective and concurring with our formal analysis, we desire that any such rule be *simple* (making it easy for analysts to understand and implement), *generic* (not tied too closely to a specific ML model), *flexible* (tunable based on what error is tolerable), and *fast*. These additional desiderata are motivated by practical real-world systems-oriented concerns.

Figure 1 illustrates our situation in a rough but intuitive manner. The whole box is the set of KFK joins that gather features. (Section 2.1 makes our assumptions precise.) Box A is the set of joins that are "safe to avoid," i.e., avoiding them is unlikely to blow up the test error (Section 4.2 makes this precise), while box B is the rest. Our goal is to characterize boxes A and B, and develop a decision rule to tell if a given join belongs to box A. We first perform a simulation study (using Naive Bayes as an example) to measure how different properties of the base tables affect the test error. We apply our theoretical and simulation results to devise an intuitive definition of box A and design a decision rule for it that exploits a powerful ML notion called the *VC dimension* [43]. Applying a standard theoretical result from ML, we define a heuristic quantity called the *Risk Of Representation* (ROR) that intuitively captures the increased risk of the test error being higher than the train error by avoiding the join. Using an appropriate threshold on the ROR yields a decision rule that can tell if a join is "safe to avoid" or not; this is how boxes A and B in Figure 1 are defined. Sadly, it is *impossible* in general to compute the ROR a priori, i.e., without performing the very feature selection computations we are trying to avoid. To resolve this quandary, we derive an upper bound on the ROR (we call it *worst-case* ROR) that is computable a priori. It yields a more *conservative* decision rule, i.e., it might wrongly predict that a join is not safe to avoid even though it actually is. Figure 1 illustrates this relationship: box C is contained in box A. The intersection of box A with the complement of box C is the set of missed opportunities for the worst-case ROR rule.

The worst-case ROR rule still requires us to inspect the foreign features (without having to do the join, of course). This motivates us to design an even simpler rule that does not even require us to look at the foreign features. We define a quantity we call the *tuple ratio* (TR) that only depends on the number of training examples and the size of the foreign key's domain. The TR is a conservative simplification of the worst-case ROR. Thus, the TR rule might miss even more opportunities for avoiding joins. Figure 1 illustrates this relationship: box D is contained in box C.

In the rest of this paper, we develop the precise theoretical machinery needed to explain our problem, characterize the effects of KFK joins on ML, and explain how we design our decision rules. Furthermore, since both of these rules are conservative, it is important to know how they perform on real data. Thus, we perform an empirical analysis with seven real-world datasets from diverse application domains: retail, hospitality, transportation, e-commerce, etc. We combine a few popular feature selection methods and popular ML classifiers. We find that there are indeed many cases on real datasets where joins are safe to avoid, and both of our rules work surprisingly well: out of 14 joins in total across all 7 datasets, both of our rules correctly classified 7 joins as safe to avoid and 3 joins as not safe to avoid but deemed 4 joins as not safe to avoid even though avoiding them did not blow up the test errors (note that our decision rules are conservative). Overall, our decision rules improved the performance of the feature selection methods significantly in many cases, including by over 10x in some cases.
In summary, our work makes the following contributions:

- To the best of our knowledge, this is the first paper to study the problem of formally characterizing the effects of KFK joins on ML classifiers and feature selection to help predict when joins can be avoided safely.
- We perform a simulation study using Naive Bayes as an example in order to measure how different properties of the normalized data affect ML error.
- We apply our theoretical and simulation results to design simple decision rules that can predict a priori if it is perhaps safe to avoid a given join.
- We perform an extensive empirical analysis using real-world datasets to validate that there are cases where avoiding joins does not increase ML error significantly and that our rules can accurately predict such cases.

Outline. Section 2 explains our problem setup and gives some background. Section 3 presents an in-depth theoretical analysis of the effects of joins on ML and feature selection. Readers more interested in the practical implications can skip to Section 4, which presents our simulation study and also explains how we design our decision rules. Section 5 presents our empirical validation with real data. Section 6 presents the related work. We conclude in Section 7.

2. PRELIMINARIES AND BACKGROUND

2.1 Setup, Example, and Assumptions

We focus on the same schema setting as [29] and adopt their terminology. The main table with the entities to model is called the *entity table*, denoted $\mathbf{S}$. There are k other tables called *attribute tables*, denoted $\mathbf{R}_i$, for $i = 1$ *to* k (if $k = 1$, we drop the subscript). The schema of $\mathbf{R}_i$ is $\mathbf{R}_i(\underline{RID_i}, \mathbf{X}_{R_i})$, where RID_i is its primary key and $\mathbf{X}_{R_i}$ is a vector (sequence) of features. We abuse the notation slightly to also treat $\mathbf{X}$ as a *set* since the order among features in $\mathbf{X}$ is immaterial in our setting. The schema of $\mathbf{S}$ is $\mathbf{S}(\underline{SID}, Y, \mathbf{X}_S, FK_1, \ldots, FK_k)$, where Y is the target for learning, $\mathbf{X}_S$ is a vector of features, and FK_i is a foreign key that refers to $\mathbf{R}_i$. Let $\mathbf{T}$ denote the output of the equi-join: $\mathbf{T} \leftarrow \pi(\mathbf{R} \bowtie_{RID=FK} \mathbf{S})$. In general, its schema is $\mathbf{T}(\underline{SID}, Y, \mathbf{X}_S, FK_1, \ldots, FK_k, \mathbf{X}_{R_1}, \ldots, \mathbf{X}_{R_k})$.

Example (based on [29]). Consider an insurance analyst predicting customer churn, i.e., will a customer leave the company (cancel their policy)? She decides to build a classifier using the table `Customers` (<u>`CustomerID`</u>, `Churn`, `Gender`, `Age`, `EmployerID`). The `EmployerID` feature is the ID of a customer's employer and is a foreign key referring to another table with data about companies and other organizations that potentially employ the customers: `Employers` (<u>`EmployerID`</u>, `Country`, `Revenue`). Thus, $\mathbf{S}$ is `Customers`, $\mathbf{R}$ is `Employers`, Y is `Churn`, FK is $\mathbf{S}$.`EmployerID`, RID is $\mathbf{R}$.`EmployerID`, $\mathbf{X}_S$ is {`Age`, `Gender`}, and $\mathbf{X}_R$ is {`Country`, `Revenue`}. She joins the tables to bring in $\mathbf{X}_R$ because she has a hunch that customers employed by rich corporations in rich countries are unlikely to churn.

We focus on the case in which all features (including Y) are *nominal*, i.e., each feature has a finite discrete domain.[1] Thus, we focus on *classification*. We assume that the foreign keys are not keys of $\mathbf{S}$ (e.g., `EmployerID` is clearly not a key of `Customers`). We also assume that the domains of all features in $\mathbf{X}_S$, $\mathbf{X}_R$, and all FK_i are "closed with respect to the prediction task" and the domain of FK_i is the same as the set of RID_i values in $\mathbf{R}_i$ (and there are no missing/NULL values)[2]. We explain the closed domain assumption. In ML, recommendation systems assume that the foreign keys of the ratings table, e.g., `MovieID` and `UserID` have closed domains, say, to enable matrix factorization models [24]. A movie might have several past ratings. So, `MovieID` can help predict future ratings. Note that closed domain does *not* mean new `MovieID` values can never occur! It means that analysts build models using only the movies seen so far but revise their feature domains and update ML models periodically (say, monthly) to absorb movies added recently. In between these revisions, the domain of `MovieID` is considered closed. `EmployerID` in our example plays exactly the same role: many customers (past and future) might have the same employer. Thus, it is reasonable to use `EmployerID` as a feature. Handling new movies (or employers) is a well-known problem called *cold-start* [39]. It is closely related to the *referential integrity constraint* for foreign keys in databases [35]. In practice, a common way to handle it is to have a special "Others" record in `Employers` as a placeholder for new employers seen in between revisions. Cold-start is orthogonal to our problem; we leave it to future work.

Overall, each feature in $\mathbf{X}_S$, $\mathbf{X}_{R_i}$, and FK_i is a discrete random variable with a known finite domain. We also assume that the foreign keys are not skewed (we relax this in a discussion in the appendix). We discuss the effects of KFK joins on ML classifiers and feature selection in general, but later we use Naive Bayes as an example.[3] Naive Bayes is a popular classifier with diverse applications ranging from spam detection to medical diagnosis [31,33]. It is also easy to understand and use; it does not require expensive iterative optimization or "black magic" for tuning hyper-parameters.

We emphasize that our goal is *not* to design new ML models or new feature selection methods, nor is to study which feature selection method or ML model yields the highest accuracy. Rather, our goal is to understand the theoretical and practical implications of ubiquitous database dependencies, viz., KFK dependencies (KFKDs) and functional dependencies (FDs) on ML. In this paper, we use the phrase "the join is safe to avoid" to mean that $\mathbf{X}_R$ can be dropped before feature selection without significantly affecting the test error of the subset obtained after feature selection. We make this notion more precise later (Section 4.2).

2.2 Background: Feature Selection

Feature selection methods are almost always used along with an ML classifier to help improve accuracy [17]. While our work is orthogonal to feature selection methods, we briefly discuss a few popular ones for concreteness sake. At a high level, there are three types of feature selection methods: *wrappers*, *filters*, and *embedded* methods [17,21].

A wrapper uses the classifier as a "black-box" and heuristically searches the space of feature subsets to obtain a more accurate subset. Sequential greedy search is a popular wrapper; it has two variants: *forward selection* and *backward selection*. Given a feature set $\mathbf{X}$, forward (resp. backward) selection computes the *error* of an ML model for different subsets of $\mathbf{X}$ of increasing (resp. decreasing) size starting with the empty set (resp. full set $\mathbf{X}$) by adding (resp. eliminating) one feature at a time. The error can be the holdout validation error or the k-fold cross-validation error. For our purposes, the simpler holdout method described in [18] suffices: the labeled data is split 50%:25%:25% with the first part used for training, the second part used for the validation error during greedy search, and the last part used for the holdout test error, which is the final indicator of the chosen subset's accuracy. Filters apply a specified scoring function to each feature $F \in \mathbf{X}$ using the labeled data but independent of any classifier. The top-k features are then chosen, with k picked either manually or tuned automatically using the validation error for a given classifier (we use the latter). Popular scoring functions include *mutual information* $I(F;Y)$ and *information gain ratio* $IGR(F;Y)$. Intuitively, $I(F;Y)$ tells us how much the knowledge of F reduces the *entropy* of Y, while $IGR(F;Y)$ normalizes it by the feature's entropy. Embedded methods are "wired" in to the classifier. A common example is L1 or L2 norm *regularization* for linear and logistic regression. These methods perform implicit feature selection by modifying the regression coefficients directly instead of searching for subsets, e.g., L1 norm makes some coefficients vanish, which is akin to dropping the corresponding features [18].

3. EFFECTS OF KFK JOINS ON ML

We start with a brief information theoretic analysis, and then dig deeper to establish the formal connections between KFKDs and the bias-variance trade-off in ML. For ease of exposition, we assume only one attribute table $\mathbf{R}$. Readers more interested in the practical aspects can skip to the summary at the end of this section, or to Section 4.

3.1 The Information Theoretic Perspective

A standard theoretical approach to ascertain which features are "useful" is to use the information theoretic notions of feature redundancy and relevancy [17,22]. Thus, we now perform such an analysis to help explain why it might be safe to avoid the join with $\mathbf{R}$, i.e., ignore $\mathbf{X}_R$.

[1] Numeric features are assumed to have been discretized to a finite set of categories, say, using binning [31].

[2] To handle RID_i values absent from FK_i in a given instance of $\mathbf{S}$, we adopt the standard practice of *smoothing* [30].

[3] We present some empirical results and a discussion of some other popular ML models in Section 5 and the appendix.

3.1.1 Feature Redundancy

We start with some intuition. The foreign key FK is also a feature used to predict Y. In our example, it is reasonable to use `EmployerID` to help predict `Churn`. The equi-join condition $\mathbf{S}.FK = \mathbf{R}.RID$ that creates $\mathbf{T}$ causes FK to *functionally determine* all of $\mathbf{X}_R$ in $\mathbf{T}$. It is as if the FD $RID \rightarrow \mathbf{X}_R$ in $\mathbf{R}$ becomes the FD $FK \rightarrow \mathbf{X}_R$ in $\mathbf{T}$.[4] Thus, given FK, $\mathbf{X}_R$ is fixed, i.e., $\mathbf{X}_R$ does *not* provide any more "information" over FK. The notion of feature *redundancy* helps capture such behavior formally [22, 45]. Its rigorous definition is given in the appendix. This yields our first, albeit simple, result (let $\mathbf{X} \equiv \mathbf{X}_S \cup \{FK\} \cup \mathbf{X}_R$).

PROPOSITION 3.1. *In $\boldsymbol{T}$, all $F \in \boldsymbol{X}_R$ are redundant.*

The proof is in the appendix. This result extends trivially to multiple attribute tables. In fact, it extends to a more general set of FDs, as we show in the appendix. In the ML literature, the redundancy of a feature is often considered an indication that the feature should be dropped. In fact, many feature selection methods explicitly search for such redundancy in order to remove the redundant features [17,22,45]. However, they try to detect the presence of feature redundancy approximately based on the dataset *instance*. Our scenario is stronger because Proposition 3.1 *guarantees* the existence of redundant features. This motivates us to consider the seemingly "radical" step of avoiding these redundant features, i.e., avoiding the join with $\mathbf{R}$.

3.1.2 Feature Relevancy

A redundant feature might sometimes be more "useful" in predicting Y; this is captured using the formal notion of feature *relevancy*. This leads to the classical redundancy-relevancy trade-off in ML [17]. We provide some intuition. Suppose, in our example, that customers of rich corporations never churn and they are the only ones who do not churn; then `Revenue` is perhaps the most relevant feature, even though it is redundant. Thus, we would like to know if it is possible for some $F \in \mathbf{X}_R$ to be *more* relevant than FK. Feature relevancy is often formalized using scores such as the mutual information $I(F;Y)$ (a rigorous definition is given in the appendix) or the information gain ratio $IGR(F;Y)$. A feature with a higher score is considered to be more relevant [17, 45]. However, when we apply these two popular notions of feature relevancy to our setting, we realize that our information theoretic analysis hits a wall.

THEOREM 3.1. $\forall F \in \boldsymbol{X}_R, I(F;Y) \leq I(FK;Y)$

PROPOSITION 3.2. *It is possible for a feature $F \in \boldsymbol{X}_R$ to have higher $IGR(F;Y)$ than $IGR(FK;Y)$.*

The proofs are in the appendix. Basically, we get near-opposite conclusions depending on the score! If we use mutual information, features in $\mathbf{X}_R$ are no more relevant than FK. Coupled with the earlier fact that $\mathbf{X}_R$ is redundant, this suggests strongly that the join is not too useful and that we might as well stick with using FK as a "representative" of $\mathbf{X}_R$. However, if we use information gain ratio, a feature in $\mathbf{X}_R$ could be *more* relevant than FK. This suggests that we should bring in $\mathbf{X}_R$ by performing the join and let the feature selection method ascertain which features to use. We explain this strange behavior intuitively.

The domain of FK is likely to be much larger than any feature in $\mathbf{X}_R$. For example, there are fewer than two hundred countries in the world, but there are millions of employers. Thus, `EmployerID` might have a much larger domain than `Country` in `Employers`. Mutual information tends to prefer features with larger domains, but information gain ratio resolves this issue by penalizing features with larger domains [17, 31]. This brings us back to square one with our original conundrum! It seems the information theoretic analysis is insufficient to help us precisely answer our core question of when it is safe to avoid a join. Thus, we now dive deeper into our problem by analyzing the effects of KFK joins on ML and feature selection from the perspective of the bias-variance trade-off, which lies at the heart of ML.

3.2 The Join Strikes Back: KFK Joins and the Bias-Variance Trade-off

We now expose a "danger" in avoiding the join (using FK as a representative for $\mathbf{X}_R$). Our intuition is simple: information theoretic arguments are generally applicable to asymptotic cases, but in the real world training datasets are *finite*. Thus, we need to look to statistical learning theory to understand precisely how ML error is affected. We start with an intuitive explanation of some standard concepts.

The expected *test error* (i.e., error on an unseen labeled example) can be decomposed into three components: *bias* (a.k.a *approximation error*), *variance* (a.k.a. *estimation error*), and *noise* [18, 40]. The noise is an inevitable component that is independent of the ML model. The bias captures the lowest possible error by any model instance in the class of ML models considered. For example, using Naive Bayes introduces a bias compared to learning the expensive joint distribution since it assumes conditional independence [33]. The variance captures the error introduced by the fact that the training dataset is only a random finite sample from the underlying data distribution, i.e., it formalizes the "instability" of the model with respect to the training data. For example, if we train Naive Bayes on two different training samples, their test errors are likely to differ. And if we provide fewer training examples to the ML model, its test error is likely to increase due to higher variance. In colloquial terms, a scenario with high variance is called *overfitting* [18].

The crux of the argument is as follows: *using FK as a representative is likely to yield a model with higher variance than a model obtained by including $\boldsymbol{X}_R$ for consideration.* Thus, the chance of getting a higher test error might increase if we avoid the join. Perhaps surprisingly, this holds true irrespective of the number of features in $\mathbf{X}_R$! Before explaining why, we observe that there is no contradiction with Section 3.1; the information theoretic analysis deals primarily with bias, not variance. We explain more below.

Relationship between Hypothesis Spaces. To help formalize our argument, we first explain the relationship between the classes of models built using FK and $\mathbf{X}_R$. This is important to understand because a model with a larger hypothesis space might have higher variance.

As before, let $\mathbf{X} \equiv \mathbf{X}_S \cup \{FK\} \cup \mathbf{X}_R$. For simplicity of exposition, let $\mathcal{D}_Y = \{0, 1\}$. Also, since our primary goal is to understand the effects of the FD $FK \rightarrow \mathbf{X}_R$, we set $\mathbf{X}_S = \phi$ (empty) for ease of exposition. A learned ML

[4] KFKDs differ from FDs [5], but we can treat the dependencies in $\mathbf{T}$ as FDs since we had assumed that there are no `NULL` values and that all feature domains are *closed*.

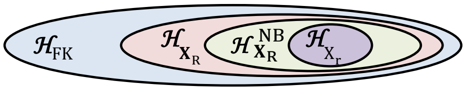


Figure 2: Relationship between hypothesis spaces.

model instance is just a prediction function $f : \mathcal{D}_{\mathbf{X}} \to \{0, 1\}$. The universe of all possible prediction functions based on $\mathbf{X}$ is denoted $\mathcal{H}_{\mathbf{X}} = \{f | f : \mathcal{D}_{\mathbf{X}} \to \{0, 1\}\}$. A given class of ML models, e.g., Naive Bayes, can only learn a subset of $\mathcal{H}_{\mathbf{X}}$ owing to its bias. This subset of functions that it can learn is called the *hypothesis space* of the ML model. Let $\mathcal{H}^{NB}_{\mathbf{X}}$ denote the hypothesis space of Naive Bayes models on $\mathbf{X}$. Given $\mathbf{Z} \subseteq \mathbf{X}$, we define the *restriction* of $\mathcal{H}_{\mathbf{X}}$ to $\mathbf{Z}$ as follows: $\mathcal{H}_{\mathbf{Z}} = \{f | f \in \mathcal{H}_{\mathbf{X}} \wedge \forall \mathbf{u}, \mathbf{v} \in \mathcal{D}_{\mathbf{X}}, \mathbf{u}_{|\mathbf{Z}} = \mathbf{v}_{|\mathbf{Z}} \implies f(\mathbf{u}) = f(\mathbf{v})\}$. Here, $\mathbf{u}_{|\mathbf{Z}}$ denotes the projection of $\mathbf{u}$ to only the features in $\mathbf{Z}$. We now establish the relationship between the various hypothesis spaces. Figure 2 depicts it pictorially.

Proposition 3.3. $\mathcal{H}_{\mathbf{X}} = \mathcal{H}_{FK} \supseteq \mathcal{H}_{\mathbf{X}_R}$

The proof is in the appendix. Note that the first part ($\mathcal{H}_{\mathbf{X}} = \mathcal{H}_{FK}$) is essentially the learning theory equivalent of Proposition 3.1. The second part might seem counter-intuitive because even if there are, say, a million features in $\mathbf{X}_R$ but only a hundred FK values, using FK instead of $\mathbf{X}_R$ is still more "powerful" than using $\mathbf{X}_R$ and dropping FK. But the intuition is simple: since $\mathbf{R}$ is fixed in our setting, we can only ever observe at most $|\mathcal{D}_{FK}|$ distinct values of $\mathbf{X}_R$, even if $\Pi_{F \in \mathbf{X}_R} |\mathcal{D}_F| \gg |\mathcal{D}_{FK}|$. Hence, using $\mathbf{X}_R$ instead of FK might increase the bias. For example, suppose that customers employed by Profit University and Charity Inc. churn and they are the only ones who churn. Then it is *impossible* in general to learn this concept correctly if `EmployerID` is excluded. Note that $\mathcal{H}^{NB}_{\mathbf{X}_R} \subseteq \mathcal{H}_{\mathbf{X}_R}$. Finally, $\forall X_r \in \mathbf{X}_R$, $\mathcal{H}^{NB}_{\mathbf{X}_R} \supseteq \mathcal{H}_{X_r} = \mathcal{H}^{NB}_{X_r}$ (and $\mathcal{H}^{NB}_{FK} = \mathcal{H}_{FK}$), since Naive Bayes has no bias if there is only one feature.

The relationship between the size of the hypothesis space and the variance is formalized in ML using the powerful notion of the *VC dimension* [40, 43]. We use this notion to complete the formalization of our argument.

VC Dimension. Due to space constraints, we only give an intuitive explanation and an example here. Intuitively, the VC dimension captures the ability of a classifier to assign the true class labels to a set of labeled data points of a given cardinality–a capability known as "shattering." For example, consider a linear classifier in 2-D. It is easy to see that it can shatter any set of 3 points (distinct and non-collinear). But it cannot shatter a set of 4 points due to the XOR problem [12]. Thus, the VC dimension of a 2-D linear classifier is 3. For finite hypothesis spaces (as in our case), the VC dimension is a direct indicator of the size of the hypothesis space [40]. A standard result from ML bounds the difference between the test and train errors (this difference is solely due to variance) as a function of the VC dimension (v) and the number of training examples (n):

Theorem 3.2. (From [40], p. 51) *For every $\delta \in (0, 1)$, with probability at least $1 - \delta$ over the choice of the training dataset, and for $n > v$, we have:*

$$|Test\ error - Train\ error| \leq \frac{4+\sqrt{vlog(2en/v)}}{\delta\sqrt{2n}}$$

Thus, a higher VC dimension means a looser bound on the variance and, possibly, higher variance. The VC dimension usually increases with the number of features. For example, it is linear in the number of features for "linear" classifiers such as logistic regression and also Naive Bayes [43]. But note that we had assumed that all features are nominal in our setting. Thus, we recode the features to numeric space using the standard binary vector representation, i.e., a feature F is converted to a 0/1 vector with $|\mathcal{D}_F| - 1$ dimensions (the last category is represented as a zero vector). With this recoding, the VC dimension of Naive Bayes (or logistic regression) on a set $\mathbf{X}$ of nominal features is $1 + \sum_{F \in \mathbf{X}} (|\mathcal{D}_F| - 1)$. If we use FK alone, the maximum VC dimension for any classifier is $|\mathcal{D}_{FK}|$, which is matched by almost all popular classifiers such as Naive Bayes. However, as per the argument for Figure 2, the VC dimension of any classifier on $\mathbf{X}_R$ is at most the number of distinct values of $\mathbf{X}_R$ in the given table $\mathbf{R}$, say, r. Since RID is the primary key of $\mathbf{R}$, we have $|\mathcal{D}_{FK}| \geq r$. Thus, the VC dimension is likely to be higher if FK is used as a representative for $\mathbf{X}_R$.

In the Context of Feature Selection. The above variance-based argument gets stronger when we consider the fact that we might *not* retain all of $\mathbf{X}_R$ *after* feature selection. Consider an extreme scenario: suppose the "true" distribution can be succintly described using a lone feature $X_r \in \mathbf{X}_R$. In our churn example, this represents a case where all customers with employers based in "The Shire" churn and they are the only ones who churn (X_r is `Country`). Suppose an "oracle" told us to only use X_r. Clearly, $\mathcal{H}_{X_r}$ is likely to be much smaller than $\mathcal{H}_{FK}$, as illustrated in Figure 2. Thus, the variance for a model based on FK is likely to be higher than a model based on X_r, as per Theorem 3.2. For the opposite extreme, i.e., the true concept needs all of $\mathbf{X}_R$, the gap with $\mathcal{H}_{FK}$ might decrease, but it might still be large.

Alas, in the real world, we do not have oracles to tell us which features are part of the true distribution; it could be none, some, or all features in $\mathbf{X}_R$. What we do have instead of an oracle is a feature selection method, although it does the job approximately using a finite labeled sample. For example, in the above extreme scenario, if we input $\{FK, X_r\}$ to a feature selection method, it is likely to output $\{X_r\}$ precisely because a model based on $\{X_r\}$ is likely to have lower variance than one based on $\{FK\}$ or $\{FK, X_r\}$. By avoiding the join, we shut the door on such possibilities and force the model to work only with FK. Thus, overall, even though FK can act as a representative for $\mathbf{X}_R$, it is probably safer to give the entire set $\mathbf{X}$ to the feature selection method and let it figure out the subset to use. Finally, note that we had assumed $\mathbf{X}_S$ is empty for the above discussion because it is orthogonal to our core issue. If $\mathbf{X}_S$ is not empty, all the hypothesis spaces shown in Figure 2 will blow up, but their relative relationships, and hence the above arguments about the variance, will remain unaffected.

Summary. Our analysis reveals a dichotomy in the accuracy effects of avoiding a KFK join for ML and feature selection: avoiding the join and using FK as a representative of $\mathbf{X}_R$ might not increase the bias, but the variance (compared after feature selection) might increase significantly.

4. PREDICTING A PRIORI IF IT IS SAFE TO AVOID A KFK JOIN

Given our understanding of the dichotomy in the effects of joins, we now focus on answering our core question: how

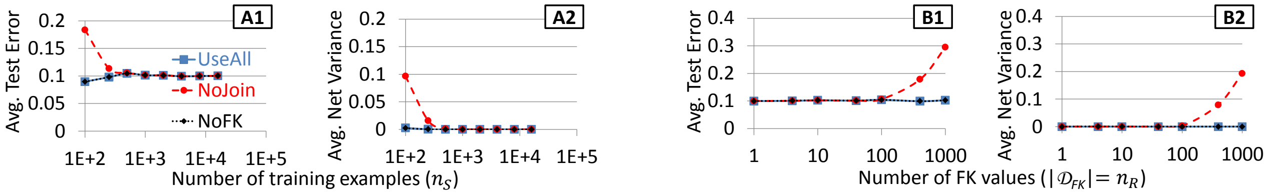


Figure 3: Simulation results for the scenario in which only a single $X_r \in \mathbf{X}_R$ is part of the true distribution, which has $P(Y = 0|X_r = 0) = P(Y = 1|X_r = 1) = p$. For these results, we set $p = 0.1$ (varying this probability did not change the overall trends). (A) Vary n_S while fixing $(d_S, d_R, |\mathcal{D}_{FK}|) = (2, 4, 40)$. (B) Vary $|\mathcal{D}_{FK}|$ $(= n_R)$ while fixing $(n_S, d_S, d_R) = (1000, 4, 4)$.

to predict a priori if a join with $\mathbf{R}$ is safe to avoid. We start with a simulation study using "controlled" datasets to validate our theoretical analysis and measure precisely how the error varies as we vary different properties of the normalized data. We then explain our decision rules and how we use our simulation measurements to tune the rules. All the plots in Section 4 are based on our synthetic datasets.

4.1 Simulation Study

We perform a Monte Carlo-style study. For the sake of tractability and depth of understanding, we use Naive Bayes as the classifier in this section. But note that our methodology is generic enough to be applicable to any classifier because we measure the accuracy only using standard notions of error, bias, and variance for a generic ML classifier.

Data Synthesis. We sample the labeled examples in an independently and identically distributed manner from a controlled true distribution $P(Y, \mathbf{X})$. Different scenarios are possible based on what features in $\mathbf{X}$ are used: it could be any or all features in $\mathbf{X}_S$, FK, and/or $\mathbf{X}_R$. Our primary goal is to understand the effects of the FD $FK \rightarrow \mathbf{X}_R$ and explain the danger in avoiding the join. Thus, we focus on a key scenario that intuitively represents the worst-case scenario for avoiding the join: the true distribution is succintly captured using a lone feature $X_r \in \mathbf{X}_R$. This corresponds to the example in Section 3.2 in which all customers with employers based in "The Shire" churn and they are the only ones who churn. In line with Proposition 3.3, we expect FK to play an indirect role in predicting Y in both scenarios. All other features are random noise. We also studied two other representative scenarios: one in which all of $\mathbf{X}_S$ and $\mathbf{X}_R$ are part of the true distribution, and one in which only $\mathbf{X}_S$ and FK are. Since these two did not yield any major additional insights, we present them in the appendix.

Simulation Setup. There is one attribute table $\mathbf{R}$ $(k = 1)$, and all of $\mathbf{X}_S$, $\mathbf{X}_R$, and Y are boolean (i.e., domain size 2). The following parameters are varied one at a time: number of features in $\mathbf{X}_S$ (d_S), number of features in $\mathbf{X}_R$ (d_R), $|\mathcal{D}_{FK}|$ $(= n_R)$, and total number of training examples (n_S). We also sample $\frac{n_S}{4}$ examples for the test set. We generate 100 different training datasets and measure the test error and the variance based on the different models obtained from these 100 runs. This whole process was repeated 100 times with different seeds for the pseudo-random number generator [38]. Thus, we have $10,000$ runs in total for one combination of the parameters studied. While the test error and variance were defined intuitively in Section 3.2, we now define them formally (based on [11]).

Definitions. We start with the formal definition of error (based on Domingos and Pazzani [13]).

Definition 4.1. Zero-one loss. *Let t be the true class label of a given example with $\boldsymbol{X} = \boldsymbol{x}$, while $c_M(\boldsymbol{x})$ be the class predicted by a classifier M on $\boldsymbol{x}$. The* zero-one loss *(of M on $\boldsymbol{x}$), denoted $L(t, c_M(\boldsymbol{x}))$, is defined as $L(t, c_M(\boldsymbol{x})) = \mathbf{1}_{t=c_M(\boldsymbol{x})}$, where $\mathbf{1}$ is the indicator function.*

Definition 4.2. *The* local error *(in short, the* error*) of M on $\boldsymbol{x}$, denoted $E_M(\boldsymbol{x})$, is defined as the expected value of the zero-one loss, where the expectation is over the values of Y, given $\boldsymbol{X} = \boldsymbol{x}$, i.e., $E_M(\boldsymbol{x}) = \sum_{y \in \mathcal{D}_Y} P(Y = y|\boldsymbol{X} = \boldsymbol{x}) L_M(y, c_M(\boldsymbol{x}))$.*

Since M depends on the training data, we compute the expectation of the error over different training datasets, say, by averaging over a given finite collection of training datasets $\mathcal{S}$ (typically, all of the same size). Note that $|\mathcal{S}| = 100$ for our Monte Carlo runs. The *expected error* of a classifier (across true distributions and $\mathcal{S}$) on $\mathbf{x}$ is decomposed as follows:

$$E[L(t, c_M(\mathbf{x}))] = B(\mathbf{x}) + (1 - 2B(\mathbf{x}))V(\mathbf{x}) + cN(\mathbf{x}) \quad (1)$$

Here, $B(\mathbf{x})$ is the bias, $V(\mathbf{x})$ is the variance, and $N(\mathbf{x})$ is the noise. The quantity $(1 - 2B(\mathbf{x}))V(\mathbf{x})$ is called the *net variance*, which is needed to capture the opposing effects of the variance on biased and unbiased predictions [11]. The *main prediction* on $\mathbf{x}$, given $\mathcal{S}$, is defined as the mode among the multi-set of predictions that result from learning M over each training dataset in $\mathcal{S}$. The main prediction is denoted y_m, while a single prediction based on some dataset in $\mathcal{S}$ is denoted y. The bias is defined as the zero-one loss of the main prediction, i.e., $B(\mathbf{x}) = L(t, y_m)$. The variance is defined as the average loss with respect to the main prediction, i.e., $V(\mathbf{x}) = E_D[L(y_m, y)]$. The *average bias* (resp. *average net variance* and *average test error*) is the average of the bias (resp. net variance and test error) over the entire set of test examples. Our goal is to understand how the average test error and the average net variance are affected by avoiding a join. Due to space constraints, we only discuss the key results here and present the others in the appendix.

Results. We compare three classes of models: *UseAll*, which uses all of $\mathbf{X}_S$, FK, and $\mathbf{X}_R$, *NoJoin*, which omits $\mathbf{X}_R$, and *NoFK*, which omits FK. Figure 3 plots the average test error and average net variance against n_S as well as $|\mathcal{D}_{FK}|$.

At a high level, Figure 3 validates our theoretical results on the dichotomy in the effects of avoiding the join. Both *UseAll* and *NoFK* use X_r, which enables them to achieve the lowest errors. When n_S is large, *NoJoin* matches their errors even though it does not use X_r. This confirms our arguments in Section 3.1 about FK acting as a representative of X_r. However, when n_S drops, the error of $NoJoin$ increases, and as Figure 3(A2) shows, this is due to the increase in the net variance. Figure 3(B) drills into why that happens: for a fixed n_S, a higher $|\mathcal{D}_{FK}|$ yields a higher error for $NoJoin$, again because of higher net variance. This confirms our arguments in Section 3.2 about the danger of using FK as a representative due to the increase in the variance. Due to space constraints, we discuss other parameters and the other two simulation scenarios in the appendix.

4.2 Towards a Decision Rule

We now precisely define what we mean by "a join is safe to avoid" and devise intuitive decision rules to predict such cases. While our definition and decision rules are heuristic, they are based on our theoretical and simulation-based insights and they satisfy all the desiderata listed in Section 1. Our key guiding principle is *conservatism*: it is fine to not avoid a join that could have been avoided in hindsight (a missed opportunity), but we do not want to avoid a join that should not be avoided (i.e., the error blows up if it is avoided). This is reasonable since the feature selection method is there to figure out if features in $\mathbf{X}_R$ are not helpful, albeit with poorer performance. Designing a rule to avoid joins safely is challenging mainly because it needs to balance this subtle performance-accuracy trade-off correctly.

The Risk of Representation. We start be defining a heuristic quantity based on the *increase* in the error bound given by Theorem 3.2. Intuitively, it quantifies the "extra risk" caused by avoiding the join. We compare the bounds for a hypothetical "best" model that uses some subset of $\mathbf{X}_R$ instead of FK (join performed) against one that uses FK instead (join avoided). A subset of $\mathbf{X}_S$ might be used by both. We call this quantity the *Risk of Representation* (ROR):

$$ROR = \frac{\sqrt{v_{Yes}log(\frac{2en}{v_{Yes}})} - \sqrt{v_{No}log(\frac{2en}{v_{No}})}}{\delta\sqrt{2n}} + \Delta bias$$

In the above, v_{Yes} is the VC dimension of a classifier that uses FK as a representative and avoids the join, while v_{No} is for one that does not avoid the join (the appendix conveniently lists all the extra notation used in Section 4.2). We first define them and then explain the intuition behind their definition. For simplicity's sake, we restrict ourselves to models such as Naive Bayes and logistic regression that have VC dimension linear in the number of features, but discuss some other classifiers later.[5] We are given $\mathbf{X} = \mathbf{X}_S \cup \{FK\} \cup \mathbf{X}_R$. Suppose an "oracle" told us that $\mathbf{U}_S \subseteq \mathbf{X}_S$ and $\mathbf{U}_R \subseteq \mathbf{X}_R$ are the only features in the true distribution. We only consider the case where $\mathbf{U}_R$ is nonempty.[6] Thus, ideally, $v_{Yes} = \sum_{F \in \mathbf{U}_S}(|\mathcal{D}_F| - 1) + |\mathcal{D}_{FK}|$. Denote $\sum_{F \in \mathbf{U}_S}(|\mathcal{D}_F| - 1)$ by q_S; this is the sum of the number of unique values of all features in $\mathbf{U}_S$. Let q_R denote the number of unique values of $\mathbf{U}_R$ (taken jointly; not as individual features) in $\mathbf{R}$. In general, v_{No} does not have a closed form expression since it depends on $\mathbf{R}$, but we have $q_S < v_{No} \leq q_S + q_R$. Thus, $v_{No} \leq v_{Yes}$. Once again, since we do not have oracles in the real world, we will not know $\mathbf{U}_S$ and $\mathbf{U}_R$ a priori. Thus, it is *impossible* to compute the ROR exactly in general.[7] Furthermore, Theorem 3.2 only deals with the variance, not the bias. Thus, we denote the difference in bias using $\Delta bias$ in the ROR. Given this definition of the ROR, we now precisely state what we mean by the join with $\mathbf{R}$ is "safe to avoid."

Definition 4.3. *Given a failure probability δ and a bound $\epsilon > 0$, we say the join with $\boldsymbol{R}$ is* (δ, ϵ)-safe to avoid *iff the ROR with the given δ is no larger than ϵ.*

The ROR Rule. While the ROR intuitively captures the risk of avoiding the join and provides us a threshold-based decision rule, we immediately hit another wall: it is *impossible* in general to compute $\Delta bias$ a priori without knowing $\mathbf{U}_S$ and $\mathbf{U}_R$. Thus, prima facie, using the ROR directly for a decision rule seems to be a hopeless idea to pursue! We resolve this quandary using a simple observation: we do not really need the exact ROR but only a "good enough" indicator of the risk of using FK as a representative. Thus, drawing upon our guiding principle of conservatism, we upper bound the ROR and create a more conservative decision rule. We explain the derivation step by step. Assume $n > v_{Yes}$. First, Proposition 3.3 showed that dropping $\mathbf{X}_R$ a priori does not increase the bias but dropping FK might, which means $\Delta bias \leq 0$. Hence, we ignore $\Delta bias$ entirely:

$$ROR \leq \frac{\sqrt{v_{Yes}log(\frac{2en}{v_{Yes}})} - \sqrt{v_{No}log(\frac{2en}{v_{No}})}}{\delta\sqrt{2n}}$$

Second, we substitute the values of some variables in the above inequality. Denote $q_{No} = v_{No} - q_S$ (the slack in the earlier inequality $q_S < v_{No}$). The inequality now becomes:

$$ROR \leq \frac{1}{\delta\sqrt{2n}}[\sqrt{(q_S + |\mathcal{D}_{FK}|)log(2en/(q_S + |\mathcal{D}_{FK}|))}$$
$$- \sqrt{(q_S + q_{No})log(2en/(q_S + q_{No}))}]$$

Third, we observe that since $|\mathcal{D}_{FK}| \geq q_R \geq q_{No}$, the RHS above is a non-increasing function of q_S that is maximum when $q_S = 0$. This lets us eliminate $\mathbf{U}_S$ from the picture:

$$ROR \leq \frac{1}{\delta\sqrt{2n}}(\sqrt{|\mathcal{D}_{FK}|log(2en/|\mathcal{D}_{FK}|)} - \sqrt{q_{No}log(2en/q_{No}})$$

Fourth, let q_R^* denote the minimum possible number of unique values of $\mathbf{U}_R$ in $\mathbf{R}$, i.e., $q_R^* = min_{F \in \mathbf{X}_R}|\mathcal{D}_F|$. Now, we observe that since $q_{No} \leq |\mathcal{D}_{FK}| \leq n$, the RHS above is a non-increasing function of q_{No} that is maximum when $q_{No} = q_R^*$. Thus, finally, we get the following inequality:

$$ROR \leq \frac{1}{\delta\sqrt{2n}}(\sqrt{|\mathcal{D}_{FK}|log(2en/|\mathcal{D}_{FK}|)} - \sqrt{q_R^* log(2en/q_R^*})$$

Intuitively, the above represents the worst-case scenario in which $\mathbf{U}_S$ is empty and $\mathbf{U}_R = \{argmin_{F \in \mathbf{X}_R}|\mathcal{D}_F|\}$. Thus, we call this bound the *worst-case* ROR, and its "gap" with the exact ROR could be large (Figure 1). Henceforth, we use the term "ROR" to refer to this worst-case upper bound. The ROR rule uses a threshold: given a parameter ρ, avoid the join if ROR $\leq \rho$.[8] This rule is conservative because given a join that is (δ, ρ)-safe to avoid, there might be some $\rho' < \rho$ such that the join is actually also (δ, ρ')-safe to avoid.

[5] The upper bound derivation is similar for classifiers with more complex VC dimensions, e.g., the joint distribution. We leave a deeper formal analysis to future work.

[6] If $\mathbf{U}_R$ is empty, $\mathbf{R}$ is trivially useless.

[7] A feature selection method can ascertain $\mathbf{U}_S$ and $\mathbf{U}_R$ approximately, but our goal is to avoid this computation.

[8] We set the failure probability δ to be always 0.1, but it can also be folded into ρ since $ROR \propto \frac{1}{\delta}$.

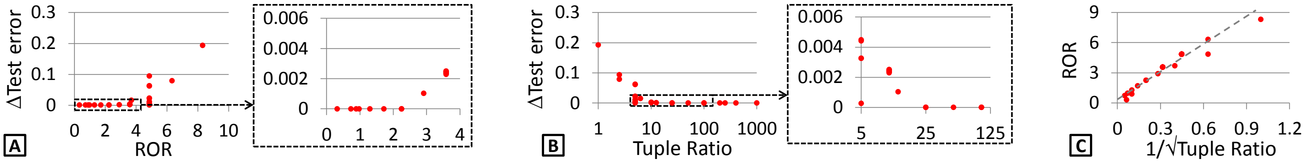


Figure 4: Scatter plots based on all the results of the simulation experiments referred to by Figure 3. (A) Increase in test error caused by avoiding the join (denoted "ΔTest error") against ROR. (B) ΔTest error against tuple ratio. (C) ROR against inverse square root of tuple ratio.

The TR Rule. The ROR rule still requires us to look at $\mathbf{X}_R$ to ascertain the features' domains and obtain q_R^*. This motivates us to consider an even simpler rule that depends only on n_S (number of training examples in $\mathbf{S}$) and $|\mathcal{D}_{FK}|$ ($= n_R$, by definition). We call $\frac{n_S}{n_R}$ the *tuple ratio* (TR). The key advantages of the TR over the ROR are that this is easier to understand and implement and that this does not even require us to look at $\mathbf{X}_R$, i.e., this enables us to ignore the join without even looking at $\mathbf{R}$. We now explain the relationship between the TR and the ROR.

When $|\mathcal{D}_{FK}| \gg q_R^*$, the ROR can be approximated as follows: $ROR \approx \frac{\sqrt{|\mathcal{D}_{FK}| log(2en/|\mathcal{D}_{FK}|)}}{\delta\sqrt{2n}}$. Since $n \equiv n_S$, we also have $ROR \approx (1/\sqrt{n_S/n_R})(\frac{\sqrt{log(2en_S/n_R)}}{\delta\sqrt{2}})$, which is approximately linear in $(1/\sqrt{TR})$ for a reasonably large TR. Thus, the TR is a conservative simplification of the ROR. The TR rule applies a threshold on the TR to predict if it is safe to avoid join: given a parameter τ, the join is avoided if TR $\geq \tau$. Note that since the TR rule is more conservative, it might lead to more missed opportunities than the ROR rule (Figure 1). We now explain why this "gap" arises. The key reason is that the TR cannot distinguish between scenarios where $|\mathcal{D}_{FK}| \gg q_R^*$ and where $|\mathcal{D}_{FK}|$ is comparable to q_R^*, as illustrated by Figure 5. When $|\mathcal{D}_{FK}| \gg q_R^*$, the ROR is high, which means the join may not be safe to avoid. But when $|\mathcal{D}_{FK}| \approx q_R^*$, the ROR is low, which means the join may be safe to avoid. The TR is oblivious to this finer distinction enabled by the ROR. But in practice, we expect that this extra capability of the ROR might not be too significant since it only matters if *all* features in $\mathbf{X}_R$ have domain sizes comparable to FK. Such an extreme situation is perhaps unlikely in the real world. In fact, as we explain later in Section 5.2.2, the ROR rule and the TR rule yielded identical results for join avoidance on all our real datasets.

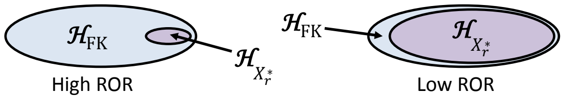


Figure 5: When $q_R^* = |\mathcal{D}_{X_r^*}| \ll |\mathcal{D}_{FK}|$, the ROR is high. When $q_R^* \approx |\mathcal{D}_{FK}|$, the ROR is low. The TR rule cannot distinguish between these two scenarios.

Tuning the Thresholds. We now explain how to tune the thresholds of our decision rules (ρ for ROR and τ for TR). For our purposes, we define a "significant increase" in test error as an absolute increase of 0.001. This might be too strict (or lenient) based on the application. Our goal here is only to demonstrate the feasibility of tuning our rules. Applications willing to tolerate a higher (or lower) error can easily retune the rules based on our methodology.

Figure 4(A) shows a scatter plot of the (asymmetric) test error difference between *NoJoin* and *UseAll* based on our diverse set of simulation results (varying n_S, $|\mathcal{D}_{FK}|$, d_R, etc.) for the first scenario (a lone $X_r \in \mathbf{X}_R$ is part of the true distribution). We see that as the ROR increases, the test error difference increases, which confirms that the ROR is an indicator of the test error difference. In fact, for sufficiently small values of the ROR, the test error is practically unchanged. The zoomed-in portion of Figure 4(A) suggests that a threshold of $\rho = 2.5$ is reasonable. Figure 4(B) shows the same errors against the TR. We see that as the TR increases, the test error difference decreases. For sufficiently large values of the TR, the test error is practically unchanged. Thus, even the more conservative TR can be a surprisingly good indicator of the test error difference. The zoomed-in portion of Figure 4(B) suggests a threshold of $\tau = 20$ is reasonable. Finally, Figure 4(C) confirms the relationship between the ROR and the TR: the ROR is approximately linear in $1/\sqrt{TR}$ (Pearson correlation coefficient ≈ 0.97).

These thresholds need to be tuned only *once per ML model* (more precisely, *once per VC dimension expression*). Thus, they are qualitatively different from (hyper-)parameters in ML that need to be tuned *once per dataset instance* using cross-validation. The above threshold values can be directly used in practice for models such as Naive Bayes and logistic regression. In fact, they worked unmodified for *all* the real datasets in our experiments for both Naive Bayes and logistic regression! If the error tolerance is changed, one can use our simulation results to get new thresholds. For an ML model with a completely different VC dimension expression, our simulations have to be repeated with that ML model and new thresholds obtained in a manner similar to the above.

Multiple Attribute Tables. It is trivial to extend the TR rule to multiple $\mathbf{R}_i$: avoid the join with $\mathbf{R}_i$ if $\frac{n_S}{n_{R_i}} \geq \tau$. As for the ROR rule, since $\mathbf{U}_S$ was eliminated when deriving the worst-case ROR, we can ignore the other foreign keys in $\mathbf{S}$. Thus, we can avoid the join with $\mathbf{R}_i$ if the ROR computed using FK_i and $\min_{F \in \mathbf{X}_{R_i}} |\mathcal{D}_F|$ is $\leq \rho$. Making join avoidance decisions for multiple $\mathbf{R}_i$ jointly, rather than independently as we do, might yield less conservative (albeit more complex) decision rules. We leave this to future work.

Multi-Class Case. The VC dimension makes sense only when Y has two classes, which might limit the applicability of the ROR rule. In the ML literature, there are various generalizations of the VC dimension for multi-class classifiers, e.g., the Natarajan dimension or the graph dimension [40]. Intuitively, they generalize the notion of the power of the classifier by also considering the number of classes. However, it is known that these more general dimensions are bounded (for "linear" classifiers such as Naive Bayes or lo-

Dataset	#Y	(n_S, d_S)	k	k'	$(n_{R_i}, d_{R_i}), i = 1\ to\ k$
Walmart	7	(421570, 1)	2	2	(2340, 9), (45, 2)
Expedia	2	(942142, 6)	2	1	(11939, 8), (37021, 14)
Flights	2	(66548, 20)	3	3	(540, 5), (3182, 6), (3182, 6)
Yelp	5	(215879, 0)	2	2	(11537, 32), (43873, 6)
MovieLens1M	5	(1000209, 0)	2	2	(3706, 21), (6040, 4)
LastFM	5	(343747, 0)	2	2	(4999, 7), (50000, 4)
BookCrossing	5	(253120, 0)	2	2	(49972, 4), (27876, 2)

Figure 6: Dataset statistics. #Y is the number of target classes. k is the number of attribute tables. k' is the number of foreign keys with closed domains.

gistic regression) by a log-linear factor in the product of the total number of feature values (sum of the domain sizes for nominal features) and the number of classes [10]. Hence, intuitively, the ROR rule might be a stricter condition than needed for the multi-class case, which is in line with our guiding principle of conservatism for avoiding joins. We leave a deeper analysis of the multi-class case to future work.

5. EXPERIMENTS ON REAL DATA

Our goal here is three-fold: (1) verify that there are cases where avoiding joins does not increase error significantly, (2) verify that our rules can accurately predict those cases, and (3) analyze the robustness and sensitivity of our rules.

Datasets. Standard sources such as the UCI ML repository did not have datasets with known KFKDs/FDs. Thus, we obtained real datasets from other sources: Walmart, Expedia, and Yelp are from the data science contest portal Kaggle (www.kaggle.com); MovieLens1M and BookCrossing are from GroupLens (grouplens.org); Flights is from openflights.org; LastFM is from mtg.upf.edu/node/1671 and last.fm. Figure 6 provides the dataset statistics. We describe each dataset and the prediction task. We used a standard unsupervised binning technique (equal-length histograms) for numeric features. Links to the data and our scripts for data preparation will be made available on our project website. To the best of our knowledge, this is the first paper to gather and clean so many normalized real datasets for ML. We hope our efforts help further research on this topic.

Walmart. Predict department-wise sales levels by joining data about past sales with data about stores and weather/economic indicators. **S** is Sales (SalesLevel, IndicatorID, StoreID, Dept), while Y is SalesLevel. $\mathbf{R}_1$ is Indicators (IndicatorID, TempAvg, TempStdev, CPIAvg, CPIStdev, FuelPriceAvg, FuelPriceStdev, UnempRateAvg, UnempRateStdev, IsHoliday). $\mathbf{R}_2$ is Stores (StoreID, Type, Size). Both foreign keys (StoreID and IndicatorID) have closed domains with respect to the prediction task.

Expedia. Predict if a hotel will be ranked highly by joining data about past search listings with data about hotels and search events. **S** is Listings (Position, HotelID, SearchID, Score1, Score2, LogHistoricalPrice, PriceUSD, PromoFlag, OrigDestDistance). Y is Position. $\mathbf{R}_1$ is Hotels (HotelID, Country, Stars, ReviewScore, BookingUSDAvg, BookingUSDStdev, BookingCount, BrandBool, ClickCount). $\mathbf{R}_2$ is Searches (SearchID, Year, Month, WeekOfYear, TimeOfDay, VisitorCountry, SearchDest, LengthOfStay, ChildrenCount, AdultsCount, RoomCount, SiteID, BookingWindow, SatNightBool, RandomBool). HotelID has a closed domain with respect to the prediction task, while SearchID does not.

Flights. Predict if a route is codeshared by joining data about routes with data about airlines, source, and destination airports. **S** is Routes (CodeShare, AirlineID, SrcAirportID, DestAirportID, Equipment1, . . . , Equipment20). Y is CodeShare. $\mathbf{R}_1$ is Airlines (AirlineID, AirCountry, Active, NameWords, NameHasAir, NameHasAirlines). $\mathbf{R}_2$ is SrcAirports (SrcAirportID, SrcCity, SrcCountry, SrcDST, SrcTimeZone, SrcLongitude, SrcLatitude). $\mathbf{R}_3$ is DestAirports (DestAirportID, DestCity, DestCountry, DestTimeZone, DestDST, DestLongitude, DestLatitude). All foreign keys have closed domains with respect to the prediction task.

Yelp. Predict business ratings by joining data about past ratings with data about users and businesses. **S** is Ratings (Stars, UserID, BusinessID). Y is Stars. $\mathbf{R}_1$ is Businesses (BusinessID, BusinessStars, BusinessReviewCount, Latitude, Longitude, City, State, WeekdayCheckins1, . . . , WeekdayCheckins5, WeekendCheckins1, . . . , WeekendCheckins5, Category1, . . . Category15, IsOpen). $\mathbf{R}_2$ is Users (UserID, Gender, UserStars, UserReviewCount, VotesUseful, VotesFunny, VotesCool). Both foreign keys have closed domains with respect to the prediction task.

MovieLens1M. Predict movie ratings by joining data about past ratings with data about users and movies. **S** is Ratings (Stars, UserID, MovieID). Y is Stars. $\mathbf{R}_1$ is Movies (MovieID, NameWords, NameHasParentheses, Year, Genre1, . . . , Genre18). $\mathbf{R}_2$ is Users (UserID, Gender, Age, Zipcode, Occupation). Both foreign keys have closed domains with respect to the prediction task.

LastFM. Predict music play levels by joining data about past play levels with data about users and artists. **S** is Plays (PlayLevel, UserID, ArtistID), Y is PlayLevel, $\mathbf{R}_1$ is Artists (ArtistID, Listens, Scrobbles, Genre1, . . . , Genre5), and $\mathbf{R}_2$ is Users (UserID, Gender, Age, Country, JoinYear). Both foreign keys have closed domains with respect to the prediction task.

BookCrossing. Predict book ratings by joining data about past ratings with data about readers and books. **S** is Ratings (Stars, UserID, BookID). Y is Stars. $\mathbf{R}_1$ is Users (UserID, Age, Country). $\mathbf{R}_2$ is Books (BookID, Year, Publisher, NumTitleWords, NumAuthorWords). Both foreign keys have closed domains with respect to the prediction task.

Experimental Setup. All experiments were run on CloudLab, which offers free and exclusive access to physical compute nodes for research [37]. We use their default ARM64 OpenStack Juno profile with Ubuntu 14.10. It provides an HP Proliant server with 8 ARMv8 cores, 64 GB RAM, and 100 GB disk. Our code is written in R (version 3.1.1), and all datasets fit in memory as R data frames.

5.1 End-to-end Error and Runtime

We compare two approaches: *JoinAll*, which joins all base tables, and *JoinOpt*, which joins only those base tables predicted by the TR rule to be not safe to avoid (the ROR rule gave identical results). For each approach, we pair Naive Bayes with one of four popular feature selection methods: two wrappers (forward selection, and backward selection) and two filters (mutual information-based and information gain ratio-based). We compare only the runtimes of feature selection and exclude the time taken to join the tables. This can work against *JoinOpt*, but as such the joins took $< 1\%$ of the total runtime in almost all our results. As mentioned

A	Walmart		Expedia		Flights		Yelp		MovieLens1M		LastFM		BookCrossing	
Error Metric	**RMSE**		**Zero-one**		**Zero-one**		**RMSE**		**RMSE**		**RMSE**		**RMSE**	
Approach	JoinAll	**JoinOpt**	JoinAll	**JoinOpt**	JoinAll	**JoinOpt**	JoinAll	**JoinOpt**	JoinAll	**JoinOpt**	JoinAll	**JoinOpt**	JoinAll	**JoinOpt**
# Tables in i/p	3	1	3	2	4	3	3	3	3	1	3	2	3	3
# Features in i/p	14	3	29	21	40	35	40	40	27	2	13	6	8	8
Error with full set	0.9850	0.8927	0.2820	0.2417	0.2445	0.1699	1.3279	1.3279	1.1671	1.0671	1.2344	1.2145	1.4678	1.4678
Forward Selection														
Validation error	0.8927	0.8927	0.2338	0.2338	0.1390	0.1390	1.1361	1.1361	1.0670	1.0671	1.0247	1.0247	1.4313	1.4313
# Features in o/p	3	3	7	7	13	13	9	9	3	2	1	1	2	2
Holdout test error	0.8910	0.8910	0.2336	0.2336	0.1359	0.1359	1.1317	1.1317	1.0687	1.0685	1.0248	1.0248	1.4295	1.4295
Backward Selection														
Holdout test error	0.8961	0.8910	0.2337	0.2337	0.1390	0.1354	1.1330	1.1330	1.0693	1.0685	1.0475	1.0248	1.4422	1.4422
Mutual Information-based Filter														
Holdout test error	0.8910	0.8910	0.2339	0.2339	0.1610	0.1610	1.1676	1.1676	1.0685	1.0685	1.0248	1.0248	1.4327	1.4327
Information Gain Ratio-based Filter														
Holdout test error	0.9014	0.8910	0.2401	0.2352	0.1476	0.1476	1.1423	1.1423	1.0692	1.0685	1.0248	1.0248	1.4327	1.4327

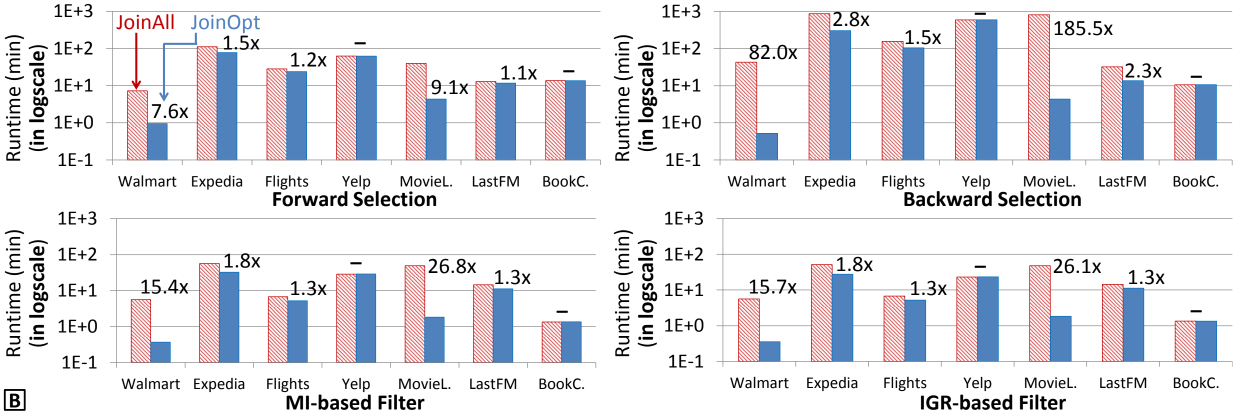


Figure 7: End-to-end results on real data. (A) Error after feature selection. (B) Runtime of feature selection.

before, we use the standard holdout validation method with the entity table (**S**) split randomly into 50%:25%:25% for training, validation, and final holdout testing. For the filter methods, the number of features filtered after ranking was actually tuned using holdout validation as a "wrapper."

In order to make the comparison more meaningful, the error metric used to report the error of the learned model depends on the type of the target and the number of classes. Specifically, the zero-one error is used for Expedia and Flights, which have binary targets, while the root mean squared error (RMSE) is used for the other datasets, which have multi-class ordinal targets. Note that our goal is to check if *JoinOpt* avoided any joins, and if so, whether its error is much higher than *JoinAll*. Figure 7 presents the results.

Errors and Output Features. The results in Figure 7(A) validate our key claim: *JoinOpt* did avoid some joins, and in all the cases where it did, the holdout test error did not increase significantly. For example, *JoinOpt* avoided both joins in both Walmart and MovieLens1M ("#Tables in input") without any increase in error. On Expedia, Flights, and LastFM, only one join each was avoided by *JoinOpt*, and the error did not increase much here either. Finally, on Yelp and BookCrossing, none of the joins were predicted to be safe to avoid. Furthermore, the trends are the same irrespective of the feature selection method used. In general, sequential greedy search had lower errors than the filter methods, which is consistent with the literature [17]. Surprisingly, in 12 of the 20 results (4 methods × 5 datasets; Yelp and BookCrossing excluded), *JoinOpt* and *JoinAll* had *identical* errors! This is because the output feature sets were identical even though the input for *JoinOpt* was smaller. For example, both *JoinAll* and *JoinOpt* had selected the same 3 features on Walmart for both forward selection and MI-based filter: {IndicatorID, StoreID, Dept}. Thus, none of the foreign features seem to matter for accuracy here. Similarly, on Expedia, the outputs were identical for forward selection: {HotelID, Score2, RandomBool, BookingWindow, Year, ChildrenCount, SatNightBool}, and backward selection but with 12 features. Thus, the HotelID sufficed and the hotel's features were not needed. On LastFM, for all methods except backward selection, both *JoinAll* and *JoinOpt* returned only {UserID}. It seems even ArtistID does not help (not just the artist's features), but our rules are not meant to detect this. Due to space constraints, we provide all other output features in the appendix.

In 3 of the 20 results, *JoinOpt* had almost the same error as *JoinAll* but with a different output feature set. For example, on MovieLens1M, for forward selection *JoinOpt* gave {UserID, MovieID}, while *JoinAll* also included a movie genre feature. More surprisingly, the error was actually significantly *lower* for *JoinOpt* in 5 of the 20 results (e.g., backward selection on Walmart and LastFM). This lower error is a serendipity caused by the variability introduced by the heuristic nature of the feature selection method. Since these feature selection methods are not globally optimal, *JoinAll*, which uses all the redundant features, seems to face a higher risk of being stuck at a poor local optimal. For example, for backward selection on Walmart, *JoinAll* dropped IndicatorID but retained many store and weather features even though it was less helpful for accuracy.

Runtime. The runtime speedup depends on the ratio of the number of features used by *JoinAll* against *JoinOpt*: the

A

Walmart

	JoinAll	No R1	No R2	*NoJoins*
FS	0.8910	0.8910	0.8910	0.8910
BS	0.8961	0.8910	0.8910	0.8910

MovieLens1M

	JoinAll	No R1	No R2	*NoJoins*
FS	1.0687	1.0685	1.0687	1.0685
BS	1.0693	1.0685	1.0699	1.0685

Yelp

	JoinAll	No R1	No R2	NoJoins
FS	1.1317	1.2350	1.2052	1.3322
BS	1.1330	1.2145	1.2446	1.3322

LastFM

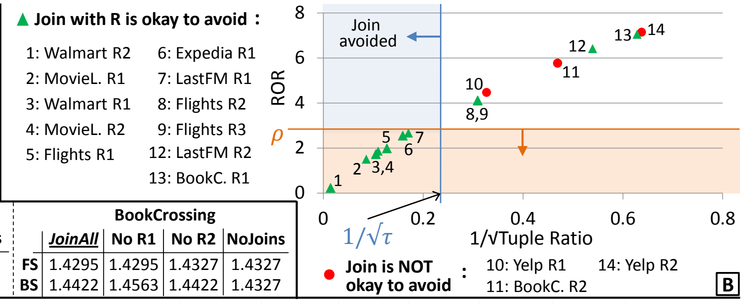
	JoinAll	*No R1*	No R2	NoJoins
FS	1.0248	1.0248	1.0248	1.0248
BS	1.0475	1.0248	1.0456	1.0248

Flights

	JoinAll	*No R1*	No R2	No R3	No R1,R2	No R1,R3	No R2,R3	NoJoins
FS	0.1359	0.1359	0.1359	0.1359	0.1359	0.1359	0.1359	0.1359
BS	0.1390	0.1354	0.1363	0.1363	0.1354	0.1387	0.1363	0.1354

BookCrossing

	JoinAll	No R1	No R2	NoJoins
FS	1.4295	1.4295	1.4327	1.4327
BS	1.4422	1.4563	1.4422	1.4327



FS	Walmart	Expedia	Flights	Yelp	MovieL.	LastFM	BookC.
JoinOpt	0.8910	0.2336	0.1359	1.1317	1.0685	1.0248	1.4295
JoinAllNoFK	0.9477	0.2800	0.1688	1.1317	1.1940	1.5375	1.5347

BS	Walmart	Expedia	Flights	Yelp	MovieL.	LastFM	BookC.
JoinOpt	0.8910	0.2337	0.1354	1.1330	1.0685	1.0248	1.4422
JoinAllNoFK	0.9455	0.2760	0.1817	1.1381	1.1806	1.5863	1.5347

C

Figure 8: (A) Robustness. Holdout test errors after Forward Selection (FS) and Backward Selection (BS). The "plan" chosen by JoinOpt is highlighted, e.g., NoJoins on Walmart. (B) Sensitivity. We set $\rho = 2.5$ and $\tau = 20$. An attribute table is deemed "okay to avoid" if the increase in error was within 0.001 with either FS or BS. (C) Holdout test errors of JoinOpt and JoinAllNoFK, which drops all foreign keys a priori.

more features avoided, the higher the speedup. Hence, the speedup depends on the number of features in the base tables, as Figure 7(B) shows. Since *JoinOpt* avoided both joins on MovieLens1M and Walmart, the speedups were high: 186x and 82x resp. for backward selection; 26x and 15x resp. for the filter methods. But on Expedia, the ratio of the number of features used is lower (≈ 1.4), which yielded more modest speedups of between 1.5x to 2.8x. Similarly, the speedups are lower on Flights and LastFM. It is noteworthy that these datasets cover the entire spectrum in terms of how many joins can be avoided: Walmart and MovieLens1M on one end; Yelp and BookCrossing on the other.

Summary. In all 28 results, *JoinOpt* had either identical or almost the same error as *JoinAll* but was often significantly faster, thus validating our core claim.

5.2 Drill-down on Real Data

5.2.1 Robustness of Join Avoidance Decisions

JoinOpt avoids only the joins predicted by the TR rule as being safe to avoid. We would like to understand the robustness of its decisions, i.e., what if we had avoided a different subset of the joins? Due to space constraints, we focus only on sequential greedy search. Figure 8(A) presents the results. Expedia is absent because it has only one foreign key with a closed domain; so Figure 7 suffices for it.

On Walmart and MovieLens1M, it was safe to avoid both joins. Our rule predicted this correctly. At the opposite end, avoiding either join in Yelp and BookCrossing blows up the error. Our rule predicted this correctly as well. This shows the need for decision rules such as ours: the state-of-the-art *JoinAll* misses the speedup opportunities on Walmart and MovieLens1M, while its naive opposite *NoJoins* causes the error to blow up on Yelp and BookCrossing. On Flights, our rule predicted that Airlines is safe to avoid but not the other two attribute tables. Yet, it turned out that even the other two could have been avoided. This is an instance of the "missed opportunities" we had anticipated: recall that our rules are conservative (Figure 1). LastFM and BookCrossing also have an attribute table each that was deemed not safe to avoid, but it turned out that they could have been avoided. However, the results for *JoinAll* suggest that the features from those tables were not useful anyway; such opportunities are beyond the scope of this work.

5.2.2 Sensitivity to Thresholds

The TR rule uses $\tau = 20$ based on our simulation results. Similarly, we had picked $\rho = 2.5$ for the ROR rule. We now validate the sensitivity of the rules to the threshold settings by comparing them with the actual TR and ROR values on the real data. Figure 8(B) shows the results.

We see that the ROR is almost linear in the inverse square root of the TR even on the real data. Many attribute tables of Walmart, MovieLens1M, Flights, and Expedia lie below the thresholds chosen for either rule. They were all correctly predicted to be safe to avoid. Both attribute tables of Yelp and one of BookCrossing were correctly predicted to be not safe to avoid. The others fall in between and include some "missed opportunities." But note that there is no case in which avoiding an attribute table that was deemed safe to avoid by the TR rule caused the error to blow up.

Finally, we also tried a higher error tolerance of 0.01 instead of 0.001. This yielded new thresholds $\tau = 10$ and $\rho = 4.2$ based on Figure 4. This setting correctly predicted that two more joins could be avoided (both on Flights).

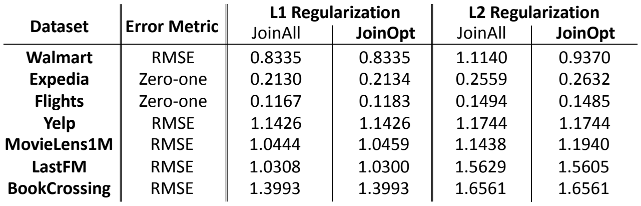
5.2.3 What if Foreign Keys Are Dropped?

Analysts sometimes judge foreign keys as being too "uninterpretable" and simply drop them. To assess the impact of this choice, we compare JoinOpt (whose accuracy is similar to JoinAll) with JoinAllNoFK, which is similar to JoinAll, but drops all foreign keys a priori. Figure 8(C) shows the results. In 6 of the 7 datasets, dropping foreign keys proved to be catastrophic for accuracy for both forward and backward selection. As we explained in our theoretical analysis in Section 3.2 (Figure 2), this is primarily because dropping foreign keys (JoinAllNoFK) might cause the bias to blow up drastically, while JoinOpt does not increase the bias.

5.3 Other ML Classifiers

It is natural to wonder if the trends observed on Naive Bayes would translate to other ML models. Note that the theoretical results and arguments in Section 3 apply to ML classifiers in general. Nevertheless, we now discuss another popular classifier: logistic regression. We also consider the popular Tree-Augmented Naive Bayes model (TAN) but, due to space constraints, we discuss TAN in the appendix.

Unlike Naive Bayes, the most popular feature selection method for logistic regression is the embedded method of



Dataset	Error Metric	L1 Regularization		L2 Regularization	
		JoinAll	**JoinOpt**	JoinAll	**JoinOpt**
Walmart	RMSE	0.8335	0.8335	1.1140	0.9370
Expedia	Zero-one	0.2130	0.2134	0.2559	0.2632
Flights	Zero-one	0.1167	0.1183	0.1494	0.1485
Yelp	RMSE	1.1426	1.1426	1.1744	1.1744
MovieLens1M	RMSE	1.0444	1.0459	1.1438	1.1940
LastFM	RMSE	1.0308	1.0300	1.5629	1.5605
BookCrossing	RMSE	1.3993	1.3993	1.6561	1.6561

Figure 9: Holdout test errors for logistic regression with regularization for the same setup as Figure 7.

regularization that constrains the L1 or L2 norm of the coefficient vector [17,18]. We consider both and use the well-tuned implementation of logistic regression from the popular *glmnet* library in R [14]. Figure 9 presents the results. We see that the errors of *JoinOpt* are comparable to those of *JoinAll* with L1. Thus, the trends are the same as for Naive Bayes, which agrees with our theoretical results. Interestingly, L2 errors are significantly higher than L1 errors. This is an artefact of the data existing in a sparse feature space for which L1 is known to be usually better than L2 [18].

5.4 Discussion: Implications for Analysts

Our work presents at least three major practical implications. These are based on our conversations with analysts at multiple settings–a telecom company, a Web company, and an analytics vendor–about our results. First, analysts often join all tables almost by instinct. Our work shows that this might lead to much poorer performance without much accuracy gain. Avoiding joins that are safe to avoid can speed up the exploratory process of comparing feature sets and ML models. Our rules, especially the easy-to-understand TR rule, help with this task. Second, we found many cases where avoiding some joins led to a counter-intuitive *increase* in accuracy. Thus, at the least, it might help to try both *JoinOpt* and *JoinAll*. Third, analysts often *drop* all foreign key features (even if they have closed domains, e.g., `StoreID` in Walmart) since they subjectively deem such features as too "uninterpretable." Our work shows that this ad hoc step could seriously hurt accuracy. This finding helps analysts be better informed of the precise consequences of such a step.

Finally, most of the burden of feature engineering (designing and choosing features) for ML today falls on analysts [12,20]. But the database community is increasingly recognizing the need to provide more systems support for feature engineering, e.g., Columbus [23,46] provides declarative feature selection operations along with a performance "optimizer." We think it is possible for such systems to integrate our decision rules for avoiding joins either as new optimizations or as "suggestions" for analysts.

6. RELATED WORK

We now explain how our problem and ideas relate to prior work in both the database and ML literature.

Database Dependencies. Given a set of FDs, a relation can be decomposed into BCNF (in most cases) [5,8]. Our work deals with the opposite scenario of joins resulting in a relation with FDs. This scenario was also studied in [29], but their goal was to avoid the materialization of the joins to improve ML performance. Our work focuses on the more fundamental question of whether such joins are even needed for ML in the first place from an accuracy perspective. There are database dependencies that are more general than FDs [5]. We think it is interesting future work to explore the implications of these more general database dependencies for ML and feature selection.

Graphical ML Models. The connection between embedded multi-valued dependencies (EMVDs) and probabilistic graphical models in ML was first shown by [34] and studied further by [44]. FDs are stronger constraints than EMVDs and cause feature redundancy. We perform a theoretical and empirical analysis of the effects of such redundancy in terms of its implications for avoiding joins.

Feature Selection. There is a large body of work in the ML and data mining literature whose focus is designing new feature selection methods to improve ML accuracy [17,18]. Our focus is *not* on designing new feature selection methods but on understanding the effects of joins on feature selection. Our work is orthogonal to the feature selection method used. The redundancy-relevancy trade-off is also well-studied in the ML literature [17,22,45]. There is also some prior work on inferring approximate FDs from the data and using them as part of feature selection [42]. However, all these methods generally focus on approximately estimating redundancy and relevancy using the dataset *instance* [17]. In contrast, our results are based on the *schema*, which enables us to safely avoid features *without even looking at instances*, that is, using just the metadata. To the best of our knowledge, no feature selection method has this radical capability. A technique to "bias" the input and reduce the number of features was proposed by FOCUS [6]. At a high level, our rules are akin to a "bias" in the input. But FOCUS still requires expensive computations on the instance, unlike our rules.

Analytics Systems. There is growing interest in both industry and academia to more closely integrate ML with data processing [3,9,16,19,25,26,47]. In this context, there is a growing recognition that feature engineering is one of the most critical bottlenecks for ML [7,27,29,36,46]. Our work helps make it easier for analysts to apply feature selection over normalized data. We hope our work spurs more research in this important direction.

7. CONCLUSION AND FUTURE WORK

In this era of "big data," it is becoming almost "big dogma" that more features and data are somehow *always* better for ML. Our work makes a contrarian case that in some situations "less is more." Specifically, using theoretical, simulation, and empirical analyses, we show that in many cases, which can be predicted, features obtained using key-foreign key joins might not improve accuracy much, but degrade performance. Our work opens up new connections between data management, ML, and feature selection, and it raises new fundamental research questions at their intersection.

First, it is an open question as to how ML accuracy is affected by other database dependencies and how we can better exploit schema information to simplify feature engineering. Extending our results to numeric features and classifiers with infinite VC dimensions is another avenue. Finally, from conversations with analysts about our results, we learned that they were interested in our TR rule because it helps them quickly decide which tables to start with (often choosing from among over a dozen tables) for deeper analytics. This suggests the whole process of source selection and feature engineering might have more open questions.

8. REFERENCES

[1] Feature Selection and Dimension Reduction Techniques in SAS. nesug.org/Proceedings/nesug11/sa/sa08.pdf.

[2] Gartner Report on Analytics. gartner.com/it/page.jsp?id=1971516.

[3] Oracle R Enterprise.

[4] SAS Report on Analytics. sas.com/reg/wp/corp/23876.

[5] S. Abiteboul et al. *Foundations of Databases*. Addison-Wesley, 1995.

[6] H. Almuallim and T. G. Dietterich. Efficient Algorithms for Identifying Relevant Features. Technical report, 1992.

[7] M. Anderson et al. Brainwash: A Data System for Feature Engineering. In *CIDR*, 2013.

[8] C. Beeri and P. A. Bernstein. Computational Problems Related to the Design of Normal Form Relational Schemas. *TODS*, 4(1), Mar. 1979.

[9] Z. Cai et al. Simulation of Database-valued Markov Chains Using SimSQL. In *SIGMOD*, 2013.

[10] A. Daniely et al. Multiclass Learning Approaches: A Theoretical Comparison with Implications. In *NIPS*, 2012.

[11] P. Domingos. A Unified Bias-Variance Decomposition and its Applications. In *ICML*, 2000.

[12] P. Domingos. A Few Useful Things to Know About Machine Learning. *CACM*, 55(10), Oct. 2012.

[13] P. Domingos and M. Pazzani. On the Optimality of the Simple Bayesian Classifier under Zero-One Loss. *Machine Learning*, 29(2-3):103–130, 1997.

[14] J. H. Friedman et al. Regularization Paths for Generalized Linear Models via Coordinate Descent. *Journal of Statistical Software*, 33(1), 2010.

[15] N. Friedman, D. Geiger, and M. Goldszmidt. Bayesian Network Classifiers. *Machine Learning*, 29(2-3):131–163, Nov. 1997.

[16] A. Ghoting et al. SystemML: Declarative Machine Learning on MapReduce. In *ICDE*, 2011.

[17] I. Guyon, S. Gunn, M. Nikravesh, and L. A. Zadeh. *Feature Extraction: Foundations and Applications*. New York: Springer-Verlag, 2001.

[18] T. Hastie et al. *The Elements of Statistical Learning: Data mining, Inference, and Prediction*. Springer-Verlag, 2001.

[19] J. Hellerstein et al. The MADlib Analytics Library or MAD Skills, the SQL. In *VLDB*, 2012.

[20] S. Kandel et al. Enterprise Data Analysis and Visualization: An Interview Study. In *IEEE VAST*, 2012.

[21] R. Kohavi and G. H. John. Wrappers for Feature Subset Selection. *Artif. Intell.*, 97(1-2), Dec. 1997.

[22] D. Koller and M. Sahami. Toward Optimal Feature Selection. In *ICML*, 1995.

[23] P. Konda et al. Feature Selection in Enterprise Analytics: A Demonstration using an R-based Data Analytics System. In *VLDB*, 2013.

[24] Y. Koren et al. Matrix Factorization Techniques for Recommender Systems. *IEEE Computer*, 42(8), Aug. 2009.

[25] T. Kraska et al. MLbase: A Distributed Machine-learning System. In *CIDR*, 2013.

[26] A. Kumar et al. Hazy: Making it Easier to Build and Maintain Big-data Analytics. *CACM*, 56(3):40–49, March 2013.

[27] A. Kumar et al. Model Selection Management Systems: The Next Frontier of Advanced Analytics. *ACM SIGMOD Record*, Dec. 2015.

[28] A. Kumar et al. To Join or Not to Join? Thinking Twice about Joins before Feature Selection. *UW-Madison CS Tech. Rep. TR1828*, 2015.

[29] A. Kumar, J. Naughton, and J. Patel. Learning Generalized Linear Models Over Normalized Data. In *SIGMOD*, 2015.

[30] C. D. Manning, P. Raghavan, and H. Schütze. *Introduction to Information Retrieval*. Cambridge University Press, 2008.

[31] T. M. Mitchell. *Machine Learning*. McGraw Hill, 1997.

[32] A. Pavlo et al. Skew-aware Automatic Database Partitioning in Shared-nothing, Parallel OLTP Systems. In *SIGMOD*, 2012.

[33] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann Publishers Inc., 1988.

[34] J. Pearl and T. Verma. The Logic of Representing Dependencies by Directed Graphs. In *AAAI*, 1987.

[35] R. Ramakrishnan and J. Gehrke. *Database Management Systems*. McGraw-Hill, Inc., New York, NY, USA, 2003.

[36] C. Ré et al. Feature Engineering for Knowledge Base Construction. *Data Engineering Bulletin*, 2014.

[37] R. Ricci, E. Eide, and the CloudLab Team. Introducing CloudLab: Scientific Infrastructure for Advancing Cloud Architectures and Applications. *;login:*, 39(6), 2014.

[38] I. Rish et al. An Analysis of Data Characteristics that Affect Naive Bayes Performance. In *ICML*, 2001.

[39] A. I. Schein et al. Methods and Metrics for Cold-start Recommendations. In *SIGIR*, 2002.

[40] S. Shalev-Shwartz and S. Ben-David. *Understanding Machine Learning: From Theory to Algorithms*. Cambridge University Press, 2014.

[41] A. Silberschatz et al. *Database Systems Concepts*. McGraw-Hill, Inc., 2006.

[42] O. Uncu and I. Turksen. A Novel Feature Selection Approach: Combining Feature Wrappers and Filters. *Information Sciences*, 177(2), 2007.

[43] V. N. Vapnik. *The Nature of Statistical Learning Theory*. Springer-Verlag New York, Inc., 1995.

[44] S. K. M. Wong et al. A Method for Implementing a Probabilistic Model as a Relational Database. In *UAI*, 1995.

[45] L. Yu and H. Liu. Efficient Feature Selection via Analysis of Relevance and Redundancy. *JMLR*, 5, Dec. 2004.

[46] C. Zhang et al. Materialization Optimizations for Feature Selection Workloads. In *SIGMOD*, 2014.

[47] Y. Zhang et al. I/O-Efficient Statistical Computing with RIOT. In *ICDE*, 2010.

APPENDIX

A. ACKNOWLEDGMENTS

We thank David DeWitt, AnHai Doan, Robert McCann, and David Page for their feedback about this work. We thank Dan Olteanu for directing us to some of the datasets. We thank Pradap Konda and Wentao Wu for their feedback on an earlier version of this paper. This work is supported by a research grant from the Microsoft Jim Gray Systems Lab. All views expressed in this work are that of the authors and do not necessarily reflect any views of Microsoft.

B. EXTENDED BACKGROUND

DEFINITION B.1. Mutual information. *Given two random variables A and B with domains $\mathcal{D}_A$ and $\mathcal{D}_B$, their* mutual information *is $I(A;B) = H(B) - H(B|A)$, where $H(B)$ is the entropy of B. Thus, we have:*

$$I(A;B) = \textstyle\sum_{a\in\mathcal{D}_A}\sum_{b\in\mathcal{D}_B} P(a,b) log \frac{P(a,b)}{P(a)P(b)}$$

DEFINITION B.2. Weak relevance. *A feature $F \in \boldsymbol{X}$ is* weakly relevant *iff $P(Y|\boldsymbol{X}) = P(Y|\boldsymbol{X} - \{F\})$ and $\exists \boldsymbol{Z} \subseteq \boldsymbol{X} - \{F\}$ s.t. $P(Y|\boldsymbol{Z}, F) \neq P(Y|\boldsymbol{Z})$.*

DEFINITION B.3. Markov blanket. *Given a feature $F \in \boldsymbol{X}$, let $\boldsymbol{M}_F \subseteq \boldsymbol{X} - \{F\}$; $\boldsymbol{M}_F$ is a* Markov Blanket *for F iff $P(Y, \boldsymbol{X} - \{F\} - \boldsymbol{M}_F|\boldsymbol{M}_F, F) = P(Y, \boldsymbol{X} - \{F\} - \boldsymbol{M}_F|\boldsymbol{M}_F)$.*

DEFINITION B.4. Redundant feature. *A feature $F \in \boldsymbol{X}$ is* redundant *iff it is* weakly relevant *and it has a* Markov blanket *in $\boldsymbol{X}$.*

C. PROOFS

PROPOSITION C.1. *In $\boldsymbol{T}$, all $F \in \boldsymbol{X}_R$ are redundant.*

PROOF. Consider an arbitrary feature $F \in \mathbf{X}_R$. We start by showing that F is weakly relevant. To show this, we need to prove that $P(Y|\mathbf{X}) = P(Y|\mathbf{X} - \{F\})$ and $\exists \mathbf{Z} \subseteq \mathbf{X} - \{F\}$ *s.t.* $P(Y|\mathbf{Z}, F) \neq P(Y|\mathbf{Z})$. For the first part, we observe that due to the FD $FK \rightarrow \mathbf{X}_R$, fixing FK automatically fixes $\mathbf{X}_R$ (and hence F). Thus, fixing $\mathbf{X}$ automatically fixes $\mathbf{X} - \{F\}$ and vice versa, which implies that $P(Y|\mathbf{X}) = P(Y|\mathbf{X} - \{F\})$. As for the second part, we only need to produce one instance. Choose $\mathbf{Z} = \phi$, which means we need to show that it is possible to have $P(Y|F) \neq P(Y)$. It is clearly trivial to produce an instance satisfying this inequality. Thus, F is weakly relevant. Next, we show that F has a Markov Blanket $\mathbf{M}_F \subseteq \mathbf{X} - \{F\}$. In fact, we have $\mathbf{M}_F = \{FK\}$. This is because the FD $FK \rightarrow \mathbf{X}_R$ causes F to be fixed when FK is fixed, which implies $\mathbf{M}_F \cup \{F\}$ is fixed when $\mathbf{M}_F$ is fixed and vice versa. Hence, $P(Y, \mathbf{X} - \{F\} - \mathbf{M}_F|\mathbf{M}_F, F) = P(Y, \mathbf{X} - \{F\} - \mathbf{M}_F|\mathbf{M}_F)$. Thus, overall, F is a redundant feature. Since F was arbitrary, all features in $\mathbf{X}_R$ are redundant. □

Next, we extend the previous result to a more general set of FDs. We start with a definition.

Definition C.1. *A set of FDs $\mathcal{Q}$ over $\boldsymbol{X}$ is* acyclic *iff the digraph on $\boldsymbol{X}$ created as follows is acyclic: include an edge from feature X_i to X_j if there is an FD in $\mathcal{Q}$ in which X_i is in the determinant set and X_j is in the dependent set.*

Corollary C.1. *Given a table $\boldsymbol{T}(\underline{ID}, Y, \boldsymbol{X})$ with a canonical acyclic set of FDs $\mathcal{Q}$ on the features $\boldsymbol{X}$, a feature that appears in the dependent set of an FD in $\mathcal{Q}$ is redundant.*

Proof. The proof is a direct extension of the proof for Theorem C.1, and we only present the line of reasoning here. We convert $\mathbf{T}$ into a relational schema in Boyce-Codd Normal Form (BCNF) using standard techniques that take $\mathcal{Q}$ as an input [41]. Since ID is the primary key of $\mathbf{T}$, both ID and Y will be present in the same table after the normalization (call it the "main table"). Now, features that occur on the right-hand side of an FD will occur in a separate table whose key will be the features on the left-hand side of that FD. And there will be a KFKD between a feature (or features) in the main table and the keys of the other tables in a manner similar to how FK refers to RID. Thus all features that occur on the right-hand side of an FD provide no more information than the features in the main table in the same way that $\mathbf{X}_R$ provides no more information than FK in Theorem C.1. Hence, any feature that occurs in the dependent set of an FD in $\mathcal{Q}$ is redundant. □

Theorem C.2. $\forall F \in \boldsymbol{X}_R, I(F;Y) \leq I(FK;Y)$

Proof. Let the FD $FK \rightarrow \mathbf{X}_R$ be represented by a collection of functions of the form $f_F : \mathcal{D}_{FK} \rightarrow \mathcal{D}_F$ for each $F \in \mathbf{X}_R$. Our goal is to show that $I(FK;Y) \geq I(F;Y), \forall F \in \mathbf{X}_R$.

Consider any $F \in \mathbf{X}_R$. We have the following:

$$I(F;Y) = \sum_{x,y} P(F=x, Y=y)\ log \frac{P(F=x, Y=y)}{P(F=x)P(Y=y)}$$

$$I(FK;Y) = \sum_{z,y} P(FK=z, Y=y)\ log \frac{P(FK=z, Y=y)}{P(FK=z)P(Y=y)}$$

Due to the FD $FK \rightarrow \mathbf{X}_R$, the following equalities hold:

$$P(F=x) = \sum_{z:f_F(z)=x} P(FK=z)$$

$$P(F=x, Y=y) = \sum_{z:f_F(z)=x} P(FK=z, Y=y)$$

Since, all the quantities involved are non-negative, we can apply the *log-sum inequality*, which is stated as follows.

Definition C.2. *Given non-negative numbers $a_1, \ldots, a_n$ and $b_1, \ldots, b_n$, with $a = \sum a_i$ and $b = \sum b_i$, the following inequality holds, and is known as the* log-sum inequality*:*

$$\sum_{i=1}^{n} a_i log(\frac{a_i}{b_i}) \geq alog(\frac{a}{b})$$

In our setting, fixing (x,y), we have $a = P(F=x, Y=y)$ and a_is are $P(FK=z, Y=y)$, for each $z : f_F(z) = x$. Similarly, $b = P(F=x)P(Y=y)$ and b_is are $P(FK=z)P(Y=y)$, for each $z : f_F(z) = x$. Thus, we have the following inequality:

$$\sum_{z:f_F(z)=x} P(FK=z, Y=y) log \frac{P(FK=z, Y=y)}{P(FK=z)P(Y=y)} \geq$$
$$P(F=x, Y=y) log \frac{P(F=x, Y=y)}{P(F=x)P(Y=y)}$$

Since this is true for all values of (x,y), summing all the inequalities gives us $I(FK;Y) \geq I(F;Y)$. □

Proposition C.2. *It is possible for a feature $F \in \boldsymbol{X}_R$ to have higher $IGR(F;Y)$ than $IGR(FK;Y)$.*

Proof. It is trivial to construct such an instance. Thus, we omit the proof here. □

Proposition C.3. $\mathcal{H}_{\boldsymbol{X}} = \mathcal{H}_{FK} \supseteq \mathcal{H}_{\boldsymbol{X}_R}$

Proof. Recall that we had assumed $\mathbf{X}_S = \phi$. Thus, $\mathbf{X} \equiv \{FK\} \cup \mathbf{X}_R$. We first prove the first part. By definition, given $\mathbf{Z} \subseteq \mathbf{X}$, we have: $\mathcal{H}_{\mathbf{Z}} = \{f | f \in \mathcal{H}_{\mathbf{X}} \wedge \forall \mathbf{u}, \mathbf{v} \in \mathcal{D}_{\mathbf{X}}, \mathbf{u}_{|\mathbf{Z}} = \mathbf{v}_{|\mathbf{Z}} \implies f(\mathbf{u}) = f(\mathbf{v})\}$. It is easy to see that the FD $FK \rightarrow \mathbf{X}_R$ automatically ensures that this condition is true for $\mathbf{Z} = \{FK\}$. This is because $\forall \mathbf{u}, \mathbf{v} \in \mathcal{D}_{\mathbf{X}}$ s.t. $\mathbf{u}_{|FK} = \mathbf{v}_{|FK}$, the FD implies $\mathbf{u}_{|\mathbf{X}_R} = \mathbf{v}_{|\mathbf{X}_R}$, which in turn implies $\mathbf{u} = \mathbf{v}$, and hence, $f(\mathbf{u}) = f(\mathbf{v})$. Thus, $\mathcal{H}_{\mathbf{X}} = \mathcal{H}_{FK}$.

As for the second part, we show that for any arbitrary $f \in \mathcal{H}_{\mathbf{X}_R}$, $\exists g_f \in \mathcal{H}_{FK}$ s.t. $f = g_f$. Note that an $f \in \mathcal{H}_{\mathbf{X}_R}$ satisfies the condition $\forall \mathbf{u}, \mathbf{v} \in \mathcal{D}_{\mathbf{X}}, \mathbf{u}_{|\mathbf{X}_R} = \mathbf{v}_{|\mathbf{X}_R} \implies f(\mathbf{u}) = f(\mathbf{v})$. Similarly, any $g \in \mathcal{H}_{FK}$ satisfies the condition $\forall \mathbf{u}, \mathbf{v} \in \mathcal{D}_{\mathbf{X}}, \mathbf{u}_{|FK} = \mathbf{v}_{|FK} \implies g(\mathbf{u}) = g(\mathbf{v})$. We now pick some $g_f \in \mathcal{H}_{FK}$ that also satisfies the following condition: $\forall \mathbf{u} \in \mathcal{D}_{\mathbf{X}}$, $g_f(\mathbf{u}) = f(\mathbf{u})$. Such a g_f necessarily exists because of three reasons: $\mathcal{H}_{\mathbf{X}_R}$ is defined based only on those values of $\mathbf{X}_R$ that are actually present in $\mathbf{R}$, the FD $FK \rightarrow \mathbf{X}_R$ ensures that there is at least one value of FK that maps to a given value of $\mathbf{X}_R$, and the same FD also ensures (by definition) that $\forall \mathbf{u}, \mathbf{v} \in \mathcal{D}_{\mathbf{X}}, \mathbf{u}_{|FK} = \mathbf{v}_{|FK} \implies \mathbf{u}_{|\mathbf{X}_R} = \mathbf{v}_{|\mathbf{X}_R}$. Thus, overall, $\mathcal{H}_{\mathbf{X}_R} \subseteq \mathcal{H}_{FK}$. Note that the equality arises when there is exactly one value of FK that maps to one value of $\mathbf{X}_R$ in $\mathbf{R}$, i.e., all tuples in $\mathbf{R}$ have distinct values of $\mathbf{X}_R$. □

D. MORE SIMULATION RESULTS

Figure 10 presents the remaining key plots for the simulation scenario in which the true distribution is succintly captured using a lone feature $X_r \in \mathbf{X}_R$. We also studied two other scenarios: one in which all of $\mathbf{X}_R$ and $\mathbf{X}_S$ are part of the true distribution, and another in which only $\mathbf{X}_S$ and

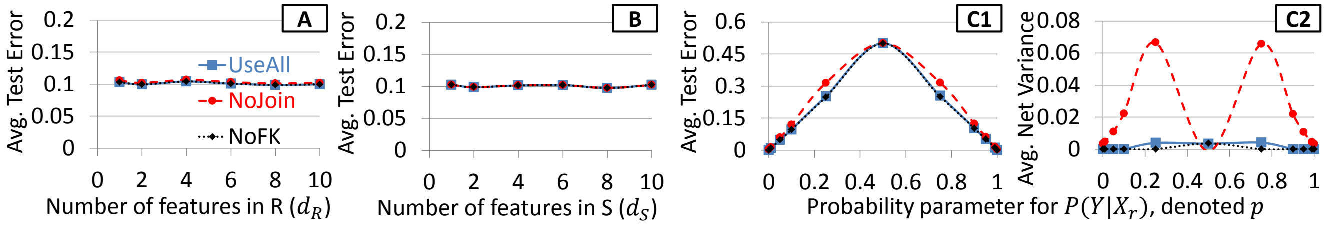


Figure 10: Remaining simulation results for the same scenario as Figure 3. (A) Vary d_R while fixing $(n_S, d_S, |\mathcal{D}_{FK}|, p) = (1000, 4, 100, 0.1)$. (B) Vary d_S while fixing $(n_S, d_R, |\mathcal{D}_{FK}|, p) = (1000, 4, 40, 0.1)$. (C) Vary p while fixing $(n_S, d_S, d_R, |\mathcal{D}_{FK}|) = (1000, 4, 4, 200)$.

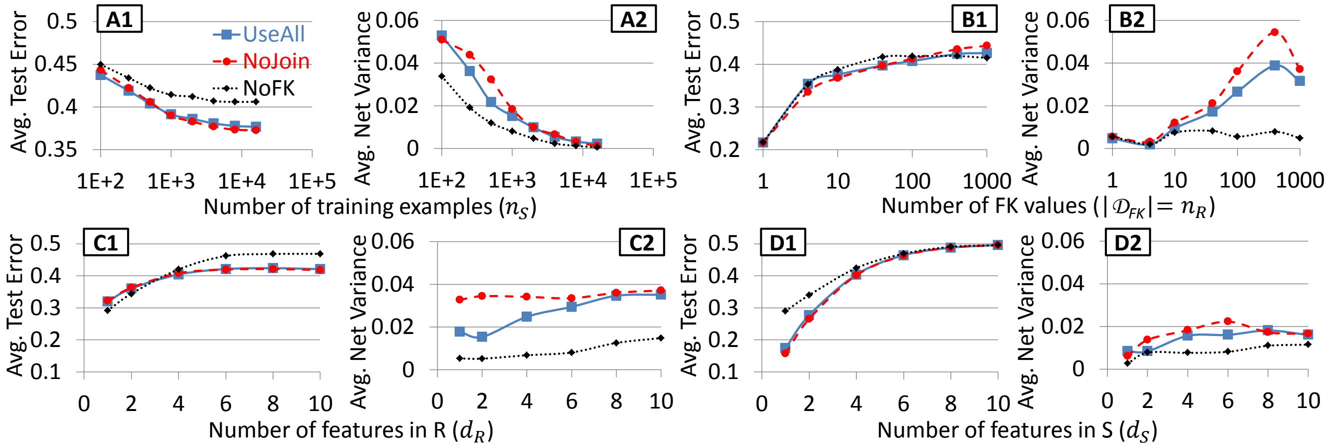


Figure 11: Simulation results for the scenario in which all of $\mathbf{X}_S$ and $\mathbf{X}_R$ are part of the true distribution. (A) Vary n_S while fixing $(d_S, d_R, |\mathcal{D}_{FK}|) = (4, 4, 40)$. (B) Vary $|\mathcal{D}_{FK}|$ while fixing $(n_S, d_S, d_R) = (1000, 4, 4)$. (C) Vary d_R while fixing $(n_S, d_S, |\mathcal{D}_{FK}|) = (1000, 4, 100)$. (D) Vary d_S while fixing $(n_S, d_R, |\mathcal{D}_{FK}|) = (1000, 4, 40)$.

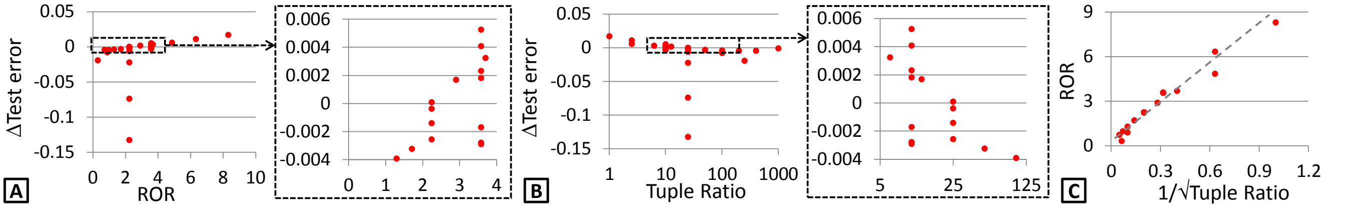


Figure 12: Scatter plots based on all the results of the simulation experiments referred to by Figure 11. (A) Increase in test error caused by avoiding the join (denoted "ΔTest error") against ROR. (B) ΔTest error against tuple ratio. (C) ROR against inverse square root of tuple ratio.

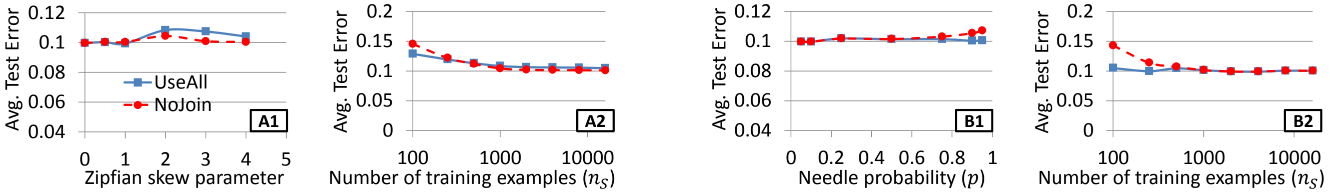


Figure 13: Effects of foreign key skew for the scenario referred to by Figure 3. (A) Benign skew: $P(FK)$ has a Zipfian distribution. We fix $(n_S, n_R, d_S, d_R) = (1000, 40, 4, 4)$, while for (A2), the Zipf skew parameter is set to 2. (B) Malign skew: $P(FK)$ has a needle-and-thread distribution. We fix $(n_S, n_R, d_S, d_R) = (1000, 40, 4, 4)$, while for (A2), the needle probability parameter is set to 0.5.

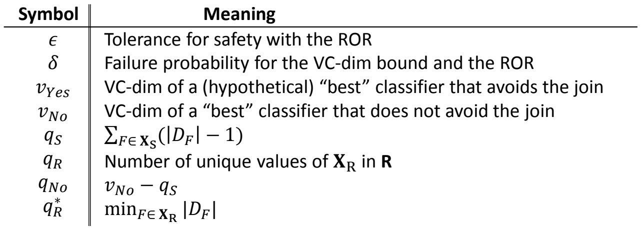
FK are. Figure 11 presents the plots for the former. Since the latter scenario in which $\mathbf{X}_R$ is useless did not reveal any interesting new insights, we skip it for brevity. We also plot the difference in test error between *NoJoin* and *JoinAll* for the scenario in which all of $\mathbf{X}_R$ and $\mathbf{X}_S$ are part of the true distribution in Figure 12. We see that the trends are largely similar to those in Figure 4 and that the same thresholds for ρ and τ work here as well.

Foreign Key Skew. So far, we had assumed that FK values are not skewed. Neither the ROR nor the TR account for skew in $P(FK)$. Foreign key skew is a classical problem in the database literature due to its effects on parallel joins [41] but its effects on ML have not been studied before. We now shed some light on the effects of skew in $P(FK)$ on ML.

We start by observing a key twist to the database-style understanding of skew: skew in $P(FK)$ *per se* is less important than its implications for learning the target. Thus, we classify skew in $P(FK)$ into two types: *benign* and *malign.* Loosely defined, benign skew in $P(FK)$ is that which does not make it much harder to learn the target, while malign skew is the opposite. We give some intuition using the scenario of a lone feature $X_r \in \mathbf{X}_R$ being part of the true distribution. Suppose $P(X_r)$ has no skew (the distribution is based on $\mathbf{T}$, not $\mathbf{R}$). Since multiple FK values might map to the same X_r value, there might still be high skew in $P(FK)$. But what really matters for accuracy is whether $P(Y)$ is skewed as well, and whether the skew in $P(Y)$ "colludes" with the skew in $P(FK)$.

There are three possible cases when there is skew in $P(FK)$: (1) $P(Y)$ is not skewed, (2) $P(Y)$ is skewed (some class label is dominant), and $P(FK)$ is skewed such that low-probability FK values co-occur mostly with high-probability Y values, and (3) $P(Y)$ is skewed, and $P(FK)$ is skewed such that low-probability FK values co-occur mostly with low-probability Y values. Cases (1) and (2) represent benign skew: even though some FK values have low probability, together they might still be able to learn the target concept reasonably well because there might be "enough" training examples for each class label. But case (3) is an instance of malign skew: essentially, FK "diffuses" the already low probability of some Y value(s) into a potentially large number of low-probability FK values. This issue might be less likely if X_r was used instead, i.e., the join was not avoided, since usually $|\mathcal{D}_{X_r}| \ll |\mathcal{D}_{FK}|$. Thus, malign skews in $P(FK)$ might make it less safe to avoid the join.

To verify the above, we performed two more simulation experiments. First, we embed a benign skew in FK using the standard Zipf distribution, which is often used in the database literature [32]. Second, we embed a malign skew in FK using what we call a "needle-and-thread" distribution: one FK value has a probability mass p ("needle" probability) and it is associated with one X_r value (and hence one Y value). The remaining $1-p$ probability mass is uniformly distributed over the remaining $(n_R - 1)$ FK values, all of which are associated with the other X_r value (and hence, the other Y value). Intuitively, this captures the extreme case (3) in which the skew in FK colludes with the skew in Y. Figure 13 presents the results for *UseAll* and *NoJoin*. As expected, the benign skew does not cause the test error of *NoJoin* to increase much, but the malign skew does. Figure 13(A) also suggests that benign skew might sometimes work in favor of *NoJoin* (this is primarily because the bias increased for *UseAll*). But as Figure 13(B1) shows, the test error of *NoJoin* increases when the skew in Y colludes with the skew in FK. However, as Figure 13(B2) shows, this gap closes as the number of training examples increases.



Symbol	Meaning
ϵ	Tolerance for safety with the ROR
δ	Failure probability for the VC-dim bound and the ROR
v_{Yes}	VC-dim of a (hypothetical) "best" classifier that avoids the join
v_{No}	VC-dim of a "best" classifier that does not avoid the join
q_S	$\sum_{F\in \mathbf{X}_S}(\|D_F\| - 1)$
q_R	Number of unique values of $\mathbf{X}_R$ in **R**
q_{No}	$v_{No} - q_S$
q_R^*	$\min_{F\in \mathbf{X}_R} \|D_F\|$

Figure 14: Extra notation used in Section 4.2

Overall, we need to account for malign skews in FK when using the ROR or TR rules for avoiding joins safely. While it is possible to detect malign skews using $H(FK|Y)$, we take a simpler, albeit more conservative, approach. We just check $H(Y)$, and if it is too low (say, below 0.5, which corresponds roughly to a 90%:10% split), we do not avoid the join. This is in line with our guiding principle of conservatism. It also captured all the cases of malign skews in our above simulations. We leave more complex approaches for handling skew to future work.

E. OTHER ML MODELS: TAN

TAN strikes a balance between the efficiency of Naive Bayes and the expressive power of general Bayesian networks [15]. TAN searches for strong conditional dependencies among pairs of features in $\mathbf{X}$ given Y using mutual information to construct a tree of dependencies on $\mathbf{X}$. Surprisingly, TAN might actually be *less* accurate than Naive Bayes on datasets with the KFKDs we study because TAN might not even use $\mathbf{X}_R$. Intuitively, this is because the FD $FK \rightarrow \mathbf{X}_R$ causes all features in $\mathbf{X}_R$ to be dependent on FK in the tree computed by TAN. This leads to $\mathbf{X}_R$ participating only via unhelpful Kronecker delta distributions, viz., $P(\mathbf{X}_R|FK)$. Depending on how structure learning is done, general Bayesian networks could face this issue too. We leave techniques to solve this issue to future work.

F. OUTPUT FEATURE SETS

Due to space constraints, we only provide a summary here. We provide the detailed list of output feature sets in our technical report [28]. We omit Yelp and BookCrossing, since none of the joins were avoided by *JoinOpt* on those two datasets. On Walmart, the output of *JoinOpt* was identical to the respective output of *JoinAll* with forward selection and MI-based filter. On Expedia, the output of *JoinOpt* was identical to the respective output of *JoinAll* with forward selection, backward selection, and MI-based filter. On Flights, the output of *JoinOpt* was identical to the respective output of *JoinAll* with forward selection and MI-based filter. On MovieLens1M, the output of *JoinOpt* was identical to the output of *JoinAll* with MI-based filter. On LastFM, the output of *JoinOpt* was identical to the respective output of *JoinAll* with forward selection, MI-based filter, and IGR-based filter. In almost all the other cases, the outputs of *JoinOpt* and *JoinAll*, while different, differed only by a few features.