Are Key-Foreign Key Joins Safe to Avoid when Learning High-Capacity Classifiers?

Vraj Shah[1] Arun Kumar[1] Xiaojin Zhu[2]
[1]University of California, San Diego [2]University of Wisconsin-Madison
{vps002, arunkk}@eng.ucsd.edu, jerryzhu@cs.wisc.edu

ABSTRACT

Machine learning (ML) over relational data is a booming area of data management. While there is a lot of work on scalable and fast ML systems, little work has addressed the pains of *sourcing* data for ML tasks. Real-world relational databases typically have many tables (often, dozens) and data scientists often struggle to even obtain all tables for joins before ML. In this context, Kumar et al. showed recently that key-foreign key dependencies (KFKDs) between tables often lets us avoid such joins without significantly affecting prediction accuracy–an idea they called "avoiding joins safely." While initially controversial, this idea has since been used by multiple companies to reduce the burden of data sourcing for ML. But their work applied only to linear classifiers. In this work, we verify if their results hold for three popular high-capacity classifiers: decision trees, non-linear SVMs, and ANNs. We conduct an extensive experimental study using both real-world datasets and simulations to analyze the effects of avoiding KFK joins on such models. Our results show that these high-capacity classifiers are surprisingly and counter-intuitively more robust to avoiding KFK joins compared to linear classifiers, refuting an intuition from the prior work's analysis. We explain this behavior intuitively and identify open questions at the intersection of data management and ML theoretical research. All of our code and datasets are available for download from `http://cseweb.ucsd.edu/~arunkk/hamlet`.

1. INTRODUCTION

The data management community has long studied how to integrate ML with data systems [18, 12, 53]), how to scale ML [6, 32], and how to use database ideas to improve ML tasks [26, 27]. However, little work has tackled the pains of *sourcing* data for ML tasks in the first place, especially, how fundamental data properties affect end-to-end ML workflows [5]. In particular, applications often have many tables connected by database dependencies such as *key-foreign key dependencies* (KFKDs) [37]. Thus, given an ML task, data scientists almost always *join* multiple tables to obtain more *features* [29]. But conversations with data scientists at many enterprise and Web companies revealed that even this simple process of procuring tables is often painful in practice, since different tables are often "owned" by different teams with different access restrictions. This slows down the ML analytics lifecycle [23]. Recent reports of Google's production ML systems also show that features that yield marginal benefits incur high "technical debt" that decreases code mangeability and increases costs [42, 35].

In this context, Kumar et al. [30] showed that one can often omit an entire table by exploiting KFKDs in the *schema* ("avoid the join"), but do so without significantly reducing ML accuracy ("safely"). The basis for this dramatic capability is that a KFK join creates a *functional dependency* (FD) between the *foreign key* and the features brought in by the join, which we call "foreign features."[1]

Example (based on [30]). Consider a common classification task: predicting *customer churn*. The data scientist starts with the main table for training (simplified for exposition): `Customers` (`CustomerID`, `Churn`, `Gender`, `Age`, `Employer`). `Churn` is the *target*, while `Gender`, `Age`, and `Employer` are features. So far, this is a standard classification task. She then notices the table `Employers` (`Employer`, `State`, `Revenue`) in her database with extra features about customers' employers. `Customers.Employer` is thus a foreign key feature connecting these tables. She joins the tables to bring in the foreign features (about employers) because she has a hunch that customers employed by rich companies in coastal states might be less likely to churn. She then tries various classifiers, e.g., logistic regression or decision trees.

Using learning theory, [30] revealed a dichotomy in how safe it is to avoid a KFK join, which we summarize next. Essentially, ML error has two main components, *bias* and *variance*; informally, bias quantifies how complex the ML model is, while variance quantifies how tied the trained model is to the given training dataset [44]. Intuitively, more complex models have lower bias and higher variance; this is known as the *bias-variance trade-off*. Cases with high-variance are colloquially called *overfitting* [33]. Avoiding a KFK join is unlikely to raise the bias but likely to raise the variance, since foreign keys typically have larger domains than foreign features. *In simple terms, avoiding joins might cause extra overfitting.* But this extra overfitting subsides with more training examples, a behavior that was formally quantified using the powerful ML notion of *VC dimension*, which indicates the complexity of an ML model. Using this notion, [30] defined a new quantity, the *tuple ratio*, which is the ratio of the numbers of tuples of the tables being joined (customers and employers in our example). As the tuple ratio goes up, it becomes safer to avoid the join. Users can then configure a VC dimension-specific threshold based on their error

[1]While KFKDs are not the same as FDs [45], assuming features have "closed" domains, they behave essentially as FDs in the output of the join [30].

tolerance. For simple classifiers with VC dimensions *linear* in the number of features (e.g., logistic regression and Naive Bayes), this threshold is as low as 20. This idea was empirically validated with multiple real-world datasets.

While initially controversial, the idea of avoiding joins safely has been adopted by many data scientists, including at Facebook, LogicBlox, and MakeMyTrip [1]. Since the tuple ratio only needs the foreign table's cardinality rather than the table itself, data scientists can easily decide if they want to avoid the join or procure the extra table. However, the results in [30] had a major caveat–they applied only to linear classifiers. In fact, their VC dimension-based analysis suggested that the tuple ratio thresholds might be too high for high-capacity non-linear classifiers, potentially rendering this idea inapplicable to such classifiers in practice.

In this paper, we perform a comprehensive empirical and simulation study and analysis to verify (or refute) the applicability of the idea of avoiding joins safely to three popular "high-capacity" (i.e., with large or even infinite VC dimensions) classifiers: decision trees, SVMs, and ANNs.

Such complex classifiers are known to be prone to overfitting [33]. Thus, the natural expectation is that avoiding a KFK join might cause more overfitting and raise the tuple ratio threshold compared to linear models (i.e., $\gg 20$). Surprisingly, our results show the *exact opposite*! We start by rerunning the experiments from [30] for such models; we also generalize the problem slightly to allow non-categorical features. Irrespective of which model is used, the same set of joins usually turn out to be safe to avoid. Furthermore, on the datasets that had joins that were not safe to avoid, the decrease in accuracy caused by avoiding said joins (unsafely) was lower for the high-capacity classifiers. In other words, *our work refutes an intuition from the VC dimension-based analysis of [30] and shows that these popular high-capacity classifiers are counter-intuitively comparably or more robust to avoiding KFK joins than linear classifiers, not less.*

To understand the above surprising behavior in depth, we conduct a Monte Carlo-style simulation study to stress test how safe it is to avoid a join. We use decision trees, since they were the most robust to avoiding joins. We generate data for a two-table KFK join and embed various "true" distributions for the target. This includes a known "worst-case" scenario for avoiding joins for linear classifiers (i.e., errors blow up) [30]. We vary different properties of the data and the true distribution: numbers of features and training examples, noise, foreign key domain size, and skew. In very few cases does avoiding the join cause the error to rise beyond 1%. Indeed, the only scenario with much higher overfitting was when the tuple ratio was less than 3; this scenario arose in only 1 of the 7 real datasets. These results are in stark contrast to the results for linear classifiers.

Our counter-intuitive results raise new research questions at the intersection of data management and ML theory. There is a need to formalize the effects of KFKDs/FDs on the behavior of decision trees, SVMs, and ANNs. As a first step, we analyze and intuitively explain the behavior of decision trees and SVMs. Other open questions include the implications of more general database dependencies on the behavior of such models and the implications of all database dependencies for other ML tasks such as regression and clustering. We believe that solving these fundamental questions could lead to new ML analytics systems functionalities that make it easier to use ML for data analytics.

Finally, we observed two new practical bottlenecks caused by foreign key features, especially for decision trees. First, the sheer size of their domains makes it hard to interpret and visualize the trees. Second, some foreign key values may not have any training examples even if they are known to be in the domain. We adapt standard techniques to resolve these bottlenecks and verify their effectiveness empirically.

Overall, the contributions of this paper are as follows:

- To the best of our knowledge, this is the first paper to analyze the effects of avoiding KFK joins on three popular high-capacity classifiers: decision trees, SVMs, and ANNs. We present a comprehensive empirical study that refutes an intuition from prior work and shows that these classifiers are counter-intuitively more robust to avoiding joins than linear classifiers.
- We conduct an in-depth simulation study with a decision tree to assess the effects of various data properties on how safe it is to avoid a KFK join.
- We present an intuitive analysis to explain the behavior of decision trees and SVMs when joins are avoided. We identify open questions for research at the intersection of data management and ML theory.
- We resolve two new practical bottlenecks with foreign key features by adapting standard techniques.

Outline. Section 2 presents the notation, background, assumptions, and scope. Section 3 presents results on the real data. Section 4 presents the simulation study. Section 5 presents our analysis of the results and identifies open questions. Section 6 verifies the techniques to make foreign key features more practical. We discuss related prior work in Section 7 and conclude in Section 8.

2. PRELIMINARIES AND BACKGROUND

2.1 Notation

We focus on the standard star schema KFK join setting, which is ubiquitous in many applications, including retail, insurance, Web security, and recommendation systems [37, 30, 29]. The fact table, which has the target variable, is denoted $\mathbf{S}$. It has the schema $\mathbf{S}(\underline{SID}, Y, \mathbf{X}_S, FK_1, \ldots, FK_q)$. A dimension table is denoted $\mathbf{R}_i$ ($i = 1$ to q) and it has the schema $\mathbf{R}_i(\underline{RID_i}, \mathbf{X}_{R_i})$. Y is the target variable (class label), $\mathbf{X}_S$ and $\mathbf{X}_{R_i}$ are vectors (sequences) of features, RID_i is the primary key of $\mathbf{R}_i$, while FK_i is a foreign key feature that refers to $\mathbf{R}_i$. We call $\mathbf{X}_S$ *home* features and $\mathbf{X}_{R_i}$ *foreign* features. Let $\mathbf{T}$ be the output of the star join that constructs the full training dataset by *concatenating* the features from all base tables. In general, its schema is $\mathbf{T}(\underline{SID}, Y, \mathbf{X}_S, FK_1, \ldots, FK_q, \mathbf{X}_{R_1}, \ldots, \mathbf{X}_{R_q})$. In contrast to our setting, traditional ML formulations do not distinguish between home features, foreign keys, and foreign features. The number of tuples in $\mathbf{S}$ (resp. $\mathbf{R}_i$) is denoted n_S (resp. n_{R_i}); the number of features in $\mathbf{X}_S$ (resp. $\mathbf{X}_{R_i}$) is denoted d_S (resp. d_{R_i}). Without loss of generality, we assume that the join is not selective, which means n_S is also the number of tuples in $\mathbf{T}$. $\mathcal{D}_{FK_i}$ denotes the domain of FK_i and by definition, $|\mathcal{D}_{FK_i}| = n_{R_i}$. We call $\frac{n_S}{n_{R_i}}$ the *tuple ratio* for $\mathbf{R}_i$. If $q = 1$ (only one foreign table), we drop the subscript in the notation and use $\mathbf{R}$, FK, and n_R; for simplicity of exposition, we will assume $q = 1$ and use this notation.

2.2 Background: ML Terms and Concepts

We intuitively explain the ML terms, concepts, models, and theory relevant to this work and refer the interested reader to [17, 33, 44] for a deeper background.

Basics. We focus on *classification* models, which need a *training dataset* with *labeled* examples to learn the *parameters* of the model. Examples include logistic regression, support vector machines (SVMs), decision trees, and artificial neural networks (ANNs). Most ML models assume that the examples are independently and identically distributed (IID) samples from an underlying (hidden) data distribution.[2] A trained model's prediction *error* (or *accuracy*) is measured using a *test dataset* not used for training. Popular testing methodologies include *holdout* validation and (nested) *k-fold cross-validation* (CV). In the former, the labeled dataset is split three-ways: one for training, one for validation (say, to tune *hyper-parameters*), and one for final testing. In the latter, the labeled dataset is partitioned into k folds, with $k-1$ folds used for training (and validation) and the last fold used for testing; k error estimates are obtained by cycling through each fold for testing and averaged.

Models. Logistic regression and linear SVM classify examples using a hyperplane; thus, they are called *linear* classifiers. Naive Bayes models the probability of the label by estimating the conditional probability distribution of each feature and multiplying them all; it can be viewed as a linear classifier [33]. 1-NN simply picks the training example nearest to a given example for prediction. Kernel SVM implicitly transforms feature vectors to a different representation and obtains the so-called "support vectors," which are examples that help separate classes. An ANN applies multiple layers of complex non-linear transformations to feature vectors to separate the classes. Finally, a decision tree learns a disjunction of conjunctive predicates to predict classes.

Theory. The set of prediction functions learnable by a model is called its *hypothesis space*. The test error has three components: *bias* (*approximation error*), *variance* (*estimation error*), and noise [44]. Informally, bias quantifies the error of a hypothetical "best" function in the hypothesis space; it is related to the *capacity* of a model (how many prediction functions it can represent), while variance quantifies the error of the actual prediction function obtained after training relative to the hypothetical best function. Typically, a more complex model (say, with more parameters) has a lower bias but higher variance; this is the *bias-variance trade-off*. A classifier's capacity can be quantified using the Vapnik-Chervonenkis (VC) Dimension [44]. Intuitively, the VC dimension is the largest number of training examples the model can classify perfectly regardless of the training label distribution–a capability called "shattering." For example, logistic regression in 2 dimensions (features) can shatter at most 3 examples due to the "XOR problem" [47]. In general, given d features, its VC dimension is $d+1$. Decision trees, RBF-SVMs, and ANNs typically have large (even infinite) VC dimensions [44]; we call such models *high-capacity classifiers*. High-capacity classifiers often tend to have higher variance than simpler *linear models* (with VC dimensions linear in d), an issue colloquially called *overfitting*.

Feature Selection. Feature selection methods are meta-algorithms that are almost always used with an ML algorithm to improve accuracy and/or interpretability. There are three main types: (1) *Wrappers*: Also called subset selection, these methods use the ML algorithm as a black-box to search through different feature subsets and pick the one with the lowest error. Since optimal subset selection is NP-Hard, various heuristic wrappers are popular in practice, including sequential greedy search [16]. (2) *Filters*: These methods assign a score to each feature (e.g., correlation coefficient) and the top k features are selected. (3) *Embedded*: These methods "wire" feature selection into the ML algorithm, e.g., L1 or L2 regularization for logistic regression. Typically, feature selection alters the bias-variance balance by tolerating a small increase in bias for a larger decrease in variance and thus, reducing errors overall.

[2]Complex models known as statistical relational models avoid the IID assumption and handle correlated examples [15]. Such models are beyond the scope of this paper.

2.3 Background: Avoiding KFK Joins Safely

It was shown in [30] that the FD $FK \rightarrow X_R$ has an interesting and surprising implication for the bias-variance tradeoff: avoiding the KFK join (i.e., omitting X_R) is unlikely to increase the bias because the hypothesis space of almost any classifier does not shrink when X_R is avoided, but in the context feature selection, avoiding the join could result in much higher variance. The latter is because $|\mathcal{D}_{FK}|$ is usually much larger than the domains of the features in X_R. For instance, `State` in our example only has 50 values but `Employer` could have millions. This dichotomy led to the idea of *avoiding joins safely*: avoid it only if the variance is unlikely to increase much. To enable this, [30] introduced a simple *decision rule* with a user-settable threshold based on their error tolerance. The decision rule adapts a standard bound on variance from the ML literature that grows with the VC dimension and shrinks with n_S and it was simplified for linear models to enable thresholding directly on the tuple ratio $(n_S/|\mathcal{D}_{FK}|)$. Thus, as the tuple ratio goes up, there will be less overfitting, since there are more training examples relative to the model's capacity. For linear classifiers, a threshold of 20 ensured the extra overfitting was marginal. But since high-capacity classifiers are usually more prone to overfitting, this approach suggests that the tuple ratio threshold might have to be higher for such classifiers.

2.4 Assumptions and Scope

For the sake of tractability, we adopt some assumptions from [30], but also drop some others to generalize the problem. In particular, we drop the assumption that all features are categorical (finite discrete set); we allow numeric features. We focus on classification and retain the assumption that FK is not a (primary) key in $\mathbf{S}$; otherwise, it will *not* be "generalizable," i.e., all future examples will have values never seen before. In our example, `CustomerID` is not generalizable but the foreign key `Employer` is. We also retain the assumption that all feature domains are fully known during training; this is a standard assumptions in ML [33, 30]. Handling unseen feature values is called the "cold start" issue in ML [40]. In practice, cold start is often resolved by temporarily mapping new values to a known "Others" placeholder. As ML models are periodically retrained, feature domains are expanded with such new information. In particular, we assume $\mathcal{D}_{FK}$ is the same as the set of $\mathbf{R}.RID$ values (new FK_i values are mapped to "Others"). Our goal is *not* to create new ML or feature selection algorithms, nor is to ascertain which algorithm yields the best accuracy or runtime. We aim to expose and analyze how KFKDs/FDs

Table 1: Dataset statistics. q is the number of dimension tables. n_S is the total number of labeled examples; since we use nested cross-validation, the tuple ratio listed is $0.6 \times n_S/n_R$. N/A means that dimension table can never be avoided, since its foreign key feature is not generalizable.

Dataset	(n_S, d_S)	q	(n_R, d_R)	Tuple Ratio
Expedia	942142, 1	2	11939, 8	47.4
			37021, 14	N/A
Movies	1000209, 0	2	6040, 4	99.4
			3706, 21	162
Yelp	215879, 0	2	11535, 32	11.2
			43873, 6	3
Walmart	421570, 1	2	2340, 9	108.1
			45, 2	5620.9
LastFM	343747, 0	2	4099, 7	50
			50000, 4	4.1
Books	253120, 0	2	27876, 2	5.5
			49972, 4	3
Flights	66548, 20	3	540, 5	74
			3167, 6	12.6
			3170, 6	12.6

enable us to dramatically discard foreign features a priori when learning some popular high-capacity classifiers.

3. EMPIRICAL STUDY WITH REAL DATA

We present results for 10 classifiers, including 7 high-capacity ones (CART decision tree with gini, information gain, and gain ratio; SVM with RBF and quadratic kernels; multi-layer perceptron ANN; 1-nearest neighbor), and 3 linear classifiers (Naive Bayes with backward selection, logistic regression with L1 regularization, and linear SVM). We also tried a few other feature selection techniques for the linear classifiers: Naive Bayes with forward selection and filter methods and logistic regression L2 regularization. Since these additional linear classifiers did not provide any new insights, we omit them due to space constraints.

3.1 Datasets

We take the seven real datasets from [30]; these are originally from Kaggle, GroupLens, `openflights.org`, `mtg.upf.edu/node/1671`, and `last.fm`. Two datasets have binary targets (Flights and Expedia); the others have multi-class ordinal targets. However, to generalize the scope of the problem studied, we retain numeric features rather than discretize them as in [30]. The dataset statistics are provided in Table 1. We briefly describe the task for each dataset and explain what the foreign features are. More details about their schemas, including the list of all features are already in the public domain (listed in [30]). *All of our datasets, scripts, and code are available for download on our project webpage*[3] *to make reproducibility easier.*

Walmart: predict if department-wise sales will be high using past sales (fact table) joined with stores and weather/economic indicators.

Flights: predict if a route is codeshared by using other routes (fact table) joined with airlines, source, and destination airports.

[3] `http://cseweb.ucsd.edu/~arunkk/hamlet`

Yelp: predict if a business will be rated highly using past ratings (fact table) joined with users and businesses.

MovieLens: predict if a movie will be rated highly using past ratings (fact table) joined with users and movies.

Expedia: predict if a hotel will be ranked highly using past search listings (fact table) joined with hotels and search events; one foreign key, viz., the search ID, has an "open" domain, i.e., past values will not be seen in the future, which makes it unusable as a feature.

LastFM: predict if a song will be played often using past play information (fact table) joined with users and artists.

Books: predict if a book will be rated highly using past ratings (fact table) joined with readers and books.

3.2 Methodology

We perform 10-fold nested cross-validation, with a random third of the examples in the training folds being used for validation during feature selection and/or hyper-parameter tuning). We compare two approaches for each classifier: *JoinAll*, which uses all features from all base tables (the current widespread practice), and *NoJoin*, which avoids all foreign features a priori (the approach we study). For additional insights, we also include a third approach for decision trees: *NoFK*, which uses all features except the foreign keys. We used the popular R packages "rpart" for the decision trees (but for gain ratio, we used "CORElearn") and "e1071" for the SVMs. For the ANNs, we used the popular Python library Keras on TensorFlow. For Naive Bayes, we used the code from [30], while for logistic regression with L1 regularization, we used the popular R package "glmnet". We used a standard grid search for hyper-parameter tuning, with the grids described in detail below.

Decision Trees: There are two hyper-parameters to tune: *minsplit* and *cp*. *minsplit* is the number of observations that must exist in a node for a split to be attempted. Any split that does not improve the fit by a factor of *cp* is pruned off. The grid is set as follows: *minsplit* $\in \{1, 10, 100, 10^3\}$ and *cp* $\in \{10^{-4}, 10^{-3}, 0.01, 0.1, 0\}$

RBF-SVM: There are two hyper-parameters: C and γ. C controls the cost of misclassification. $\gamma > 0$ controls the bandwidth in the Gaussian kernel; given points x_i and x_j, $k(x_i, x_j) = \exp(-\gamma \cdot \|x_i - x_j\|^2)$. The grid is set as follows: $C \in \{10^{-1}, 1, 10, 100, 10^3\}$ and $\gamma \in \{10^{-4}, 10^{-3}, 0.01, 0.1, 1, 10\}$. On *Movies* and *Expedia*, we perform an extra fine tuning step with $\gamma \in \{2^{-7}, 2^{-6}, \ldots, 2^3\}$ to improve accuracy.

Quadratic-SVM: We tune the same hyper-parameters C and γ for the polynomial kernel of degree 2: $k(x_i, x_j) = (-\gamma\ x_i^T \cdot x_j)^{degree}$. We use the same grid as RBF-SVM.

Linear-SVM: We tune the C hyper-parameter for the linear kernel: $k(x_i, x_j) = x_i^T \cdot x_j$, $C \in \{10^{-1}, 1, 10, 100, 10^3\}$.

ANN: The multi-layer perceptron architecture comprises of 2 hidden units with 256 and 64 neurons respectively. Rectified linear unit (ReLU) is used as the activation function. In order to allow penalties on layer parameters, we do L_2 regularization, with the regularization parameter tuned using the following grid axis: $\{10^{-4}, 10^{-3}, 10^{-2}\}$. We choose the popular Adam stochastic gradient optimization algorithm [24] with the learning rate tuned using the following grid axis: $\{10^{-3}, 10^{-2}, 10^{-1}\}$. The other hyper-parameters of the Adam algorithm used the default values.

Table 2: 10-fold CV errors for the decision trees and 1-NN. We compare the accuracy of *JoinAll* and *NoJoin* within each model. For *Expedia* and *Flights*, we use the zero-one error; for the other datasets, we use the RMSE. The bold font marks the cases where the error of *NoJoin* is at least 0.01 higher than *JoinAll*.

Dataset	Decision Tree									1NN	
	Gini			Information			Gain Ratio				
	JoinAll	*NoJoin*	*NoFK*	*JoinAll*	*NoJoin*	*NoFK*	*JoinAll*	*NoJoin*	*NoFK*	*JoinAll*	*NoJoin*
Expedia	0.2456	0.2524	0.2653	0.2457	0.2500	0.2643	0.2462	0.2464	0.2751	0.2743	0.2685
Movies	1.0263	1.0250	1.0544	1.0320	1.0327	1.0548	1.0345	1.0361	1.0627	1.0958	1.0658
Yelp	**1.2929**	**1.3096**	1.2242	**1.3062**	**1.3259**	1.2257	**1.2452**	**1.2634**	1.2252	1.3567	1.2951
Walmart	0.7829	0.7867	0.8358	0.7839	0.7877	0.8354	0.7893	0.7899	0.8312	0.7938	0.7645
LastFM	0.9431	0.9463	1.1299	0.9445	0.9447	1.1320	0.9547	0.9552	1.1341	1.0463	1.0451
Books	0.9771	0.9797	1.0293	0.9763	0.9845	1.0312	1.0091	1.0093	1.0661	1.0936	1.0857
Flights	0.1388	0.1442	0.2030	0.1440	0.1474	0.1977	0.1419	0.1385	0.1707	0.1128	0.1087

Table 3: 10-fold CV errors of the SVMs, ANN, Naive Bayes, and logistic regression from the same experiments as Table 2.

Dataset	SVM						ANN		Naïve Bayes		Logistic Regression	
	Linear		Polynomial		RBF				BFS		L1	
	JoinAll	*NoJoin*	*JoinAll*	*NoJoin*	*JoinAll*	*NoJoin*	*JoinAll*	*NoJoin*	*JoinAll*	*NoJoin*	*JoinAll*	*NoJoin*
Expedia	0.2131	0.2155	0.2075	0.2129	0.2049	0.2105	0.1896	0.1912	0.2423	0.2450	0.2134	0.2176
Movies	1.0337	1.0342	1.0147	1.0149	0.9855	0.9856	0.9754	0.9755	1.0678	1.0742	1.0350	1.0413
Yelp	**1.1950**	**1.2060**	**1.1553**	**1.1662**	**1.1263**	**1.1456**	1.1965	1.2052	**1.1145**	**1.1842**	**1.1250**	**1.1502**
Walmart	0.8448	0.8460	0.7942	0.7948	0.7651	0.7656	0.7354	0.7355	0.8821	0.8851	0.8240	0.8243
LastFM	**1.0953**	**1.1149**	1.0038	1.0081	0.9643	0.9691	**1.0102**	**1.0255**	**0.9645**	**0.9758**	0.9838	0.9869
Books	1.1239	1.1262	1.0234	1.0245	0.9839	0.9856	0.9588	0.9585	1.0667	1.0674	**0.9719**	**0.9831**
Flights	0.1229	0.1274	0.1013	0.1065	0.0752	0.0802	0.0651	0.0688	0.1313	0.1353	0.1205	0.1230

Logistic Regression: The glmnet package performs automatic hyper-parameter tuning for the L1 regularizer, as well as the optimization algorithm. However, it has three parameters to specify a desired convergence threshold and a limit on the execution time: *nlambda*, which we set to 100, *maxit*, which we set to 10000, and *thresh*, which we set to 0.001.

Tables 2 and 3 present the 10-fold cross-validation errors of all models on all datasets.

3.3 Results

Accuracy

Our first and most important observation is that for almost all the datasets (*Yelp* being the exception) and for all three split criteria, the error of the decision tree is comparable (a gap of within 0.01) between *NoJoin* and *JoinAll*. The trend is virtually the same for the RBF-SVM and ANN as well. We also observe that the trend is almost the same for the linear models, albeit less robustly so. Thus, regardless of whether our classifier is linear or higher capacity, the relative behavior of *NoJoin* vis-a-vis *JoinAll* is virtually the same. These results represent our key counter-intuitive finding: joins are no less safe to avoid with the high-capacity classifiers than with the linear classifiers. The absolute errors of the high-capacity classifiers is mostly lower than the linear models; this is expected but orthogonal to our focus. Interestingly, on *Yelp*, in which both joins are known to be *not* safe to avoid for linear models [30], *NoJoin* correctly sees a large rise in error against *JoinAll*–almost 0.07 for Naive Bayes. But the rise is smaller for some high-capacity classifiers, e.g., RBF-SVM, Gini decision tree, and ANN all see a rise less than 0.03. Thus, these high-capacity classifiers are counter-intuitively *more* robust than linear classifiers to avoiding joins.

Table 4: Robustness results for discarding individual dimension tables with a Gini decision tree.

Dataset	Expedia	Movies	Yelp	Walmart	LastFM	Books
NoR1	0.2456	1.0261	1.2918	0.7846	0.9434	0.9772
NoR2	X	1.0252	1.3077	0.7844	0.9465	0.9869
JoinAll	0.2452	1.0263	1.2929	0.7829	0.9431	0.9771
NoJoins	0.2524	1.0250	1.3096	0.7867	0.9463	0.9797

Flights : $NoR1 : 0.1387$ $NoR2 : 0.1392$ $NoR3 : 0.1404$
$NoR1, R2 : 0.1377$ $NoR1, R3 : 0.1379$ $NoR2, R3 : 0.1426$

We also see that *NoFK* often has much higher errors than *JoinAll* and *NoJoin*. Thus, foreign key features are useful even for high-capacity classifiers; it is known for linear classifiers that dropping foreign keys causes bias to shoot up [30]. Interestingly, on *Yelp*, which has very low tuple ratios, *NoFK* has much lower errors than *JoinAll* and *NoJoin*.

To understand the above results more deeply, we conduct a "robustness" experiment by discarding dimension tables

one at a time: Table 4 presents these results for the Gini decision tree. We see that the errors with dropping dimension tables one (or even two) at a time are all still within 0.01 of *NoJoin* in all cases, except for *Yelp*. Even on *Yelp*, the error increases significantly only when $\mathbf{R}_2$ (users table) is dropped, not $\mathbf{R}_1$. As Table 1 shows, the tuple ratio for $\mathbf{R}_2$ is only 3, while that for $\mathbf{R}_1$ is 11.2. Interestingly, the tuple ratio is similarly low (3) for $\mathbf{R}_2$ in *Books* but *NoJoin* error is not much higher. Thus, the tuple ratio is only a *conservative* indicator: it can tell if an error is likely to rise but the error may not actually rise in some cases. Almost every other dimension table can safely be discarded. The results were similar for ANN on *Yelp* and for the RBF-SVM on *Yelp*, *LastFM*, and *Books*; we skip these for brevity.

Overall, out of 14 dimension tables across the 7 datasets, we are able to safely avoid (with a tolerance of 0.01) 13 for decision trees and ANN with a tuple ratio threshold of about 3. For RBF-SVM, we were able to safely avoid 11 dimension tables with a tuple ratio threshold being of about 6. These are in stark contrast to the more modest results reported with the linear classifiers in [30]: only 7 of the dimension tables could be safely avoided, that too with a tuple ratio threshold of 20. *Thus, we see that the decision trees and ANN need six times fewer training examples and the RBF-SVM needs three times fewer training examples than linear classifiers to avoid significant extra overfitting when avoiding KFK joins.* These results are counter-intuitive because such complex classifiers are known to need more (not less) training examples to avoid extra overfitting.

For an interesting comparison that we use later in Section 5, we also show the results for 1-NN (from "RWeka" in R) in Table 2. Surprisingly, this "braindead" classifier has significantly *lower* errors with *NoJoin* than *JoinAll* for most datasets! We discuss this behavior further in Section 5.

Hypothesis Tests. The cross-validation errors suggest that *NoJoin* is not significantly worse than *JoinAll* for most datasets, especially those with high tuple ratios. We now validate if the error differences are indeed statistically significant for a given error tolerance. We perform a one-tailed t-test with the ten folds' asymmetric error differences between *NoJoin* and *JoinAll* for each model on each dataset. We set the tolerance (ϵ) to both 0 and 0.01. The null hypothesis is that the error difference is not significantly higher than ϵ. Figure 5 lists the number of models for which the null hypothesis was *not rejected* for the standard $\alpha = 0.05$ confidence level and the recently recommended stricter level of $\alpha = 0.005$ [2]. We see that except on *Yelp*, which has very low tuple ratios, *NoJoin* is *not* statistically significantly worse than *JoinAll* for most models (both linear and higher capacity), especially for $\epsilon = 0.01$ but also for $\epsilon = 0$ in many cases. For example, logistic regression on *Movies* and *Yelp* has p-values of 0.97 and 0.000026 respectively for $\epsilon = 0.01$. Since the p-value for *Movies* is greater than the α levels, the null hypothesis is retained. But for *Yelp*, the null hypothesis is rejected as the p-value is far below the α levels. Due to space constraints, we skip the other p-values here but have released all the detailed results on our project webpage.

Runtimes

A key benefit of avoiding joins safely is that ML runtimes (including feature selection) could be significantly lowered for linear models [30]. We now check if this holds for high-capacity classifiers as well by comparing the end-to-end execution times (training, validation with grid search, and testing). Due to space constraints, we only report Gini metric for decision trees and RBF kernel for SVMs; these were also the most robust to avoiding joins. All experiments (except for ANN) were run on CloudLab [39]; we use a custom OpenStack profile running Ubuntu 14.10 with 40 Intel Xeon cores and 160GB of RAM. The ANN experiments were run on a commodity laptop with Nvidia GeForce GTX 1050 GPU, 16GB RAM and Windows 10. We used R version 3.2.2 and TensorFlow version 1.1.0. Figure 1 presents the results.

Table 5: Results of the hypothesis tests.

Dataset	eps = 0		eps = 0.01	
	α = 0.05	α = 0.005	α = 0.05	α = 0.005
Expedia	3	5	10	10
Movies	8	8	10	10
Yelp	1	1	4	5
Walmart	7	8	10	10
LastFM	4	6	9	9
Books	6	7	10	10
Flights	2	4	10	10

For the high-capacity classifiers, we saw an average speedup of about 2x for *NoJoin* over *JoinAll*. The highest speedup was on the *Movies*: 3.6x for the decision tree and 6.2x for the RBF-SVM. As for the ANN, *LastFM* reported the largest speedup of 2.5x. The speedup for the linear classifiers were more significant, e.g., over 80x for Naive Bayes on on *Movies* and and about 20x for logistic regression on *LastFM*. These results corroborate the orders of magnitude speedup reported in [30].

4. IN-DEPTH SIMULATION STUDY

We now dive deeper into the behavior of the decision trees using a simulation study in which we vary the underlying "true" data distribution and sampling datasets of different dimensions. We focus on a two-table join for simplicity. We use the decision tree, since it exhibited the maximum robustness to avoiding KFK joins on the real data. Our study comprehensively "stress tests" this robustness. Note that our methodology is generic enough to be applicable to any other classifier too, since we only use generic notions of error and *net variance* as defined in [30].

Setup and Data Synthesis. There is one dimension table $\mathbf{R}$ ($q = 1$), and all of $\mathbf{X}_S$, $\mathbf{X}_R$, and Y are boolean (domain size 2). We control the "true" distribution $P(Y, \mathbf{X})$ and sample labeled examples in an IID manner from it. We study two different scenarios for what features are used to (probabilistically) determine Y: `OneXr` and `XSXR`. These scenarios represent opposite extremes for how likely the (test) error is likely to shoot up when $\mathbf{X}_R$ is discarded and FK is used as a representative [30]. In `OneXr`, a lone feature $X_r \in \mathbf{X}_R$ determines Y; the rest of $\mathbf{X}_R$ and $\mathbf{X}_S$ are random noise (but note that FK will not be noise because it functionally determines X_r). In `XSXR`, all features in $\mathbf{X}_S$ and $\mathbf{X}_R$ determine Y. Intuitively, `OneXr` is the worst-case scenario for discarding $\mathbf{X}_R$ because X_r is typically far more succinct than FK, which we expect to translate to less possibility of overfitting with *NoJoin*. Note that if we use FK directly in

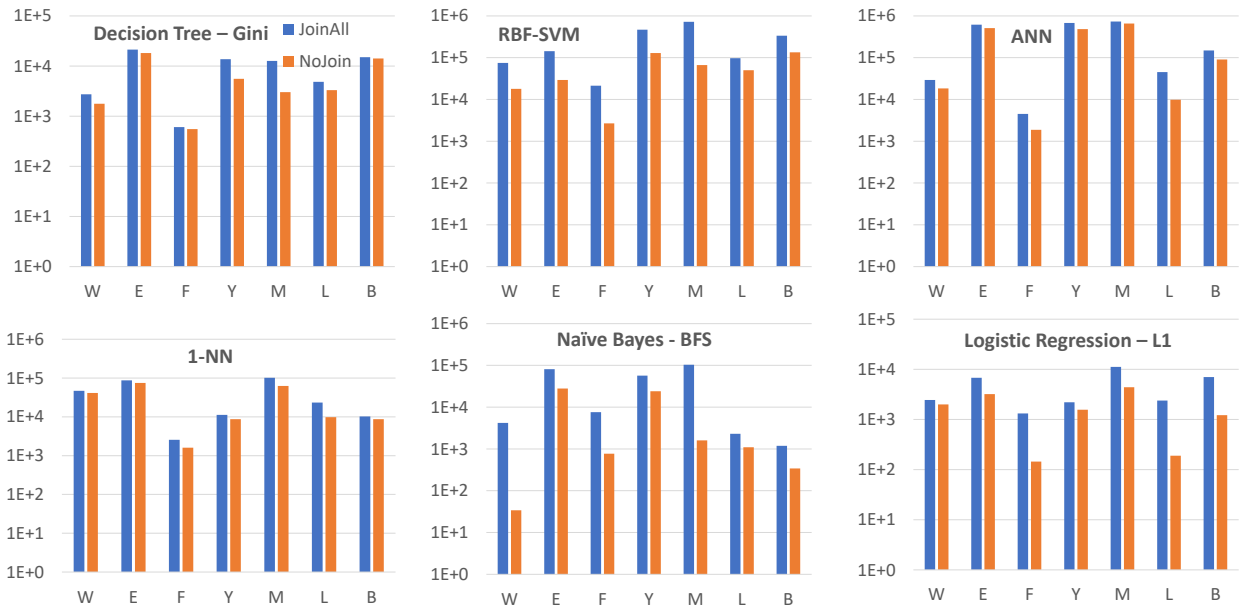


Figure 1: End-to-end runtimes on the real-world datasets: Walmart (W), Expedia (E), Flights (F), Yelp (Y), Movies (M), LastFM (L) and Books (B).

P, $\mathbf{X}_R$ can be more easily discarded because FK conveys more information anyway; so, we skip such a scenario.

The following data parameters are varied one at a time: number of training examples (n_S), size of foreign key domain ($|\mathcal{D}_{FK}| = n_R$), number of features in $\mathbf{X}_R$ (d_R), and number of features in $\mathbf{X}_S$ (d_S). We also sample $\frac{n_S}{4}$ examples each for the validation set (for hyper-parameter tuning) and the holdout test set (final indicator of error). We generate 100 different training datasets and measure the average test error and average net variance (as defined in [10]) based on the different models obtained from these 100 runs.

4.1 Scenario OneXr

The "true" distribution is set as follows: $P(Y = 0|X_r = 0) = P(Y = 1|X_r = 1) = p$, where p is called the probability skew parameter that controls the noise (also called Bayes error [17]). The exact procedure for sampling examples is as follows: (1) Construct tuples of $\mathbf{R}$ by sampling $\mathbf{X}_R$ values randomly (each feature value is an independent coin toss). (2) Construct the tuples of $\mathbf{S}$ by sampling $\mathbf{X}_S$ values randomly (independent coin tosses). (3) Assign FK values to $\mathbf{S}$ tuples uniformly randomly from $\mathcal{D}_{FK}$. (4) Assign Y values to $\mathbf{S}$ tuples by looking up into their respective X_r value (implicit join on $FK = RID$) and sampling from the above conditional distribution.

We compare *JoinAll*, *NoJoin*, and *NoFK*; we include *NoFK* for a lower bound on errors, since we know FK does not directly determine Y (although indirectly it does).[4] Figure 2 presents the results for the test errors for varying each relevant data and distribution parameter, one at a time.

Interestingly, regardless of the parameter varied, in almost all cases, *NoJoin* and *JoinAll* have almost identical errors (close to the Bayes error)! From inspecting the actual decision trees learned in these two settings, we found that in almost all cases, FK was used repeatedly for partitioning; seldom was a feature from $\mathbf{X}_R$, including X_r, used. This suggests that FK can indeed act as a good representative of $\mathbf{X}_R$ even in this extreme case. In contrast to these results, [30] found that for linear models, the errors of *NoJoin* shot up compared to *JoinAll* (a gap of nearly 0.05) as the tuple ratio starts falling below 20. In stark contrast, as Figure 2(B) shows, even for a tuple ratio of just 3, *NoJoin* and *JoinAll* have similar errors with the decision tree. This corroborates the results seen for the decision tree on the real datasets (Table 2). When n_S becomes very low or when $|\mathcal{D}_{FK}|$ becomes very high, the absolute errors of *JoinAll* and *NoJoin* increase compared to *NoFK*. This suggests that when the tuple ratio is very low, *NoFK* is perhaps worth trying too. This is similar to the behavior seen on *Yelp*. Overall, *NoJoin* exhibits similar behavior as *JoinAll* in most cases.

We also ran this scenario for the RBF-SVM (and 1-NN); the trends were similar, except for the value of the tuple ratio at which *NoJoin* deviates from *JoinAll*. Figure 3 presents the results for the experiment in which we increase $|\mathcal{D}_{FK}| = n_R$, while fixing everything else, similar to Figure 2(B) for the decision tree. We see that for the RBF-SVM, the error deviates when the tuple ratio falls below roughly 6. This corroborates its behavior on the real datasets (Table 3). The 1-NN, as expected, is far less stable and the deviation starts even at a tuple ratio of 100. As Figure 4 confirms, the deviation in error for the RBF-SVM is due to the net variance, which helps quantify the extra overfitting. This is akin to the extra overfitting reported in [30] using the plots of the net variance. Intriguingly, the 1-NN sees its net variance exhibit non-monotonic behavior; this is likely an artifact of its unstable behavior, since fewer and fewer training examples will match on FK as n_R keeps rising.

Finally, we also ran this scenario with a skew in $P(FK)$, which makes it less safe to avoid the join for linear classifiers [30]. But our simulations with a decision tree show that it is robust even to foreign key skew in terms of how safe it

[4] In general though, *NoFK* could have much higher errors if FK is part of the true distribution; indeed, *NoFK* had much higher errors on many real datasets (Table 2).

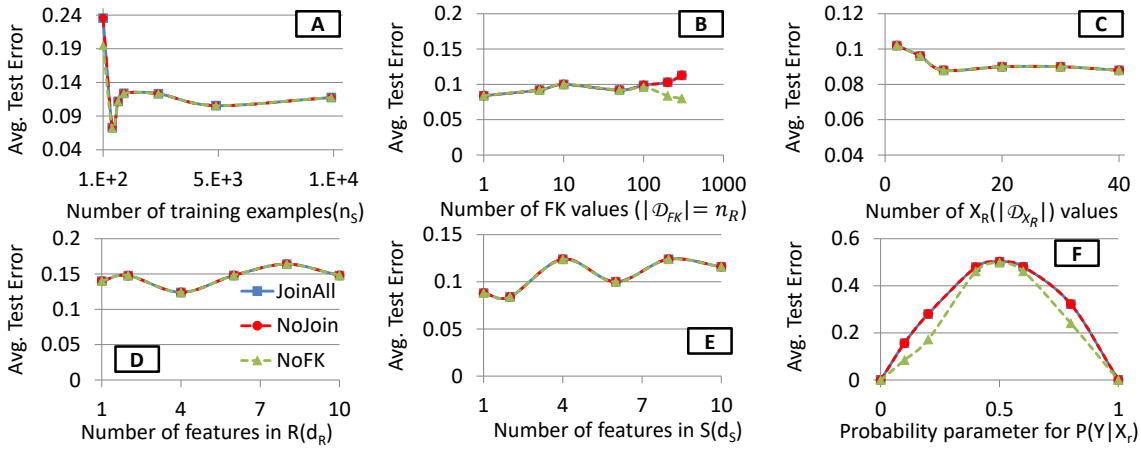


Figure 2: Simulation results for Scenario `OneXr`. For all plots except (E), we fix $p = 0.1$. Note that $n_R \equiv |\mathcal{D}_{FK}|$. (A) Vary n_S, while fixing $(n_R, d_S, d_R) = (40, 4, 4)$. (B) Vary n_R, while fixing $(n_S, d_S, d_R) = (1000, 4, 4)$. (C) Vary d_S, while fixing $(n_S, n_R, d_R) = (1000, 40, 4)$. (D) Vary d_R, while fixing $(n_R, d_S, d_R) = (1000, 40, 4)$. (E) Vary p, while fixing $(n_S, n_R, d_S, d_R) = (1000, 40, 4, 4)$. (F) Vary $|\mathcal{D}_{X_r}|$, while fixing $(n_S, n_R, d_S, d_R) = (1000, 40, 4, 4)$; all other features in $\mathbf{X}_R$ and $\mathbf{X}_S$ are binary.

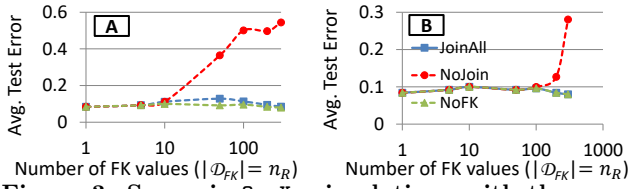


Figure 3: Scenario `OneXr` simulations with the same setup as Figure 2(B), except for (A) 1-NN and (B) RBF-SVM.

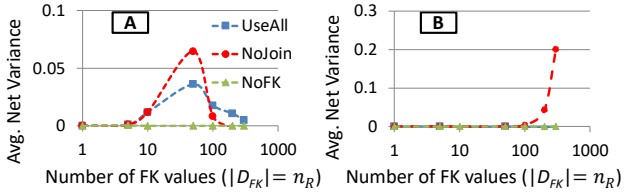


Figure 4: Average net variance in the scenario `OneXr` for (A) 1-NN and (B) RBF-SVM.

is to avoid the join. Due to space constraints, we present these results in the technical report [43].

4.2 Scenario XSXR

Unlike `OneXr`, we now create a true distribution that maps $\mathbf{X} \equiv [\mathbf{X}_S, \mathbf{X}_R]$ to Y without any noise (Bayes error). The exact procedure for sampling examples is as follows: (1) Construct a true probability table (TPT) with entries for all possible values of $[\mathbf{X}_S, \mathbf{X}_R]$ and assign a random probability to each entry such that the total probability is 1. (2) For each entry in the TPT, pick a Y value randomly and append the TPT entry; this ensures $H(Y|\mathbf{X}) = 0$. (3) Marginalize the TPT to obtain $P(\mathbf{X}_R)$ and from it, sample $n_R = \mathcal{D}_{FK}$ tuples for $\mathbf{R}$ along with an associated sequential RID value. (4) In the original TPT, push the probability of each entry to 0 if its $\mathbf{X}_R$ values did not get picked for $\mathbf{R}$ in step 3. (5) Renormalize the TPT so that the total probability is 1 and sample n_S examples (Y values do not change) and construct $\mathbf{S}$. (6) For each tuple in $\mathbf{S}$, pick its FK value uniformly randomly from the subset of RID values that map to its $\mathbf{X}_R$ value in $\mathbf{R}$ (an implicit join). We again compare *JoinAll*, *NoJoin*, and *NoFK*. Figure 5 presents the results.

Once again, we see that *NoJoin* and *JoinAll* exhibit similar errors in almost all cases, with the largest gap being 0.017 in Figure 5(C). Interestingly, even when the tuple ratio is close to 1, the gap between *NoJoin* and *JoinAll* does not widen much. Figure 5(B)) shows that as $|\mathcal{D}_{FK}|$ increases, *NoFK* remains at low overall errors, unlike both *JoinAll* and *NoJoin*. But as we increase d_R or d_S, the gap between *JoinAll*/*NoJoin* and *NoFK* narrows because even *NoFK* does not have enough training examples. Of course, all gaps virtually disappear as the number of training examples increases, as shown by Figure 5(A). Overall, *NoJoin* again exhibits similar behavior as *JoinAll*.

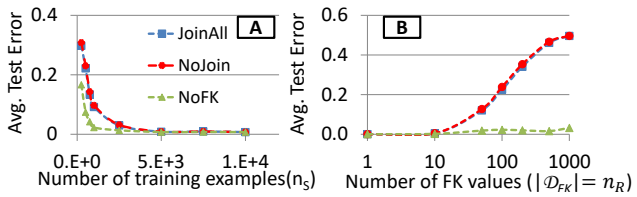
4.3 Scenario RepOneXr

We now present results for a new simulation scenario that is a slight twist on `OneXr`: the tuples of $\mathbf{R}$ are constructed by replicating the value of X_r sampled for a tuple to create all the other features in X_R. That is, X_R of an example is just the same value repeated d_R times. Note that the FD $FK \rightarrow X_R$ implies there are at least as many unique FK values as $\mathbf{X}_R$ values. Thus, by increasing $|\mathcal{D}_{FK}|$ relative to d_R, we hope to increase the chance of the model getting "confused" with *NoJoin*. Our goal is to see if this widens the gap between *JoinAll* and *NoJoin*.

Figure 6 presents the results for the two experiments on decision trees where (A) has a high tuple ratio of 25 and (B) has a low tuple ratio of 5. Once again, *JoinAll* and *NoJoin* exhibit similar errors in both the cases. The same experiment's results with RBF-SVM and 1-NN are shown in Figure 7) and Figure 8 respectively. For the RBF-SVM, *NoJoin* has higher errors at the tuple ratio of 5 but not 25, while for the 1-NN, *NoJoin* has higher errors in both cases.

5. ANALYSIS AND OPEN QUESTIONS

5.1 Explaining the Results



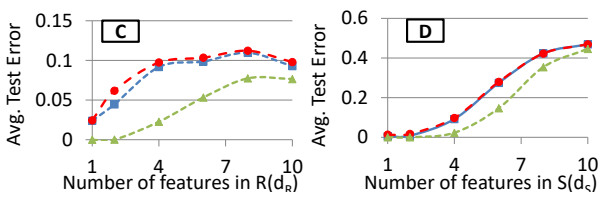


Figure 5: Simulation results for Scenario XSXR. The parameter values varied/fixed are the same as in Figure 2 (A)-(D).

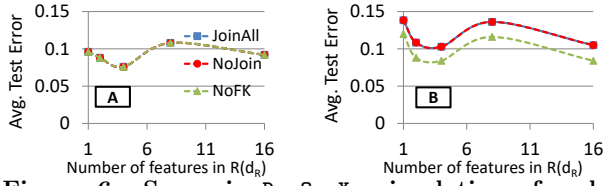


Figure 6: Scenario RepOneXr simulations for decision tree. (A) Vary d_R while fixing $(n_S, n_R, d_S) = (1000, 40, 4)$. (B) Vary d_R while fixing $(n_S, n_R, d_S) = (1000, 200, 4)$.

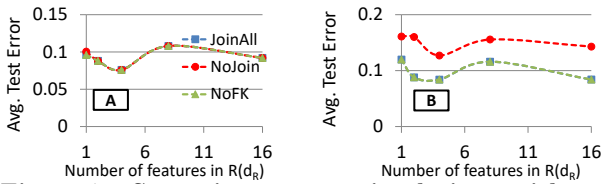


Figure 7: Scenario RepOneXr simulations with same setup as Figure 6, except for RBF-SVM.

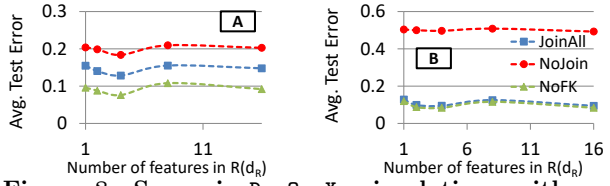


Figure 8: Scenario RepOneXr simulations with same setup as Figure 6, except for 1-NN.

We now intuitively explain the surprising behavior of decision trees and RBF-SVMs with *NoJoin* vis-a-vis *JoinAll*. We first ask: Does *NoJoin* compromise the "generalization error"? The generalization error is the difference of the test and train errors. Tables 6 and 7 list the train errors (averaged across the 10 folds). *JoinAll* and *NoJoin* are remarkably close for the decision trees (except for *Yelp*, of course). The absolute generalization errors are often high, e.g., train error is almost 0 on *Flights* with RBF-SVMs but test errors are about 0.08, but this is orthogonal to our focus–we only note that *NoJoin* does not increase this generalization error significantly. The same is true for all the decision trees. *Thus, avoiding the KFK joins safely did not significantly affect the generalization errors the high-capacity classifiers.*

Returning to 1-NN, Table 2 showed that it has similar errors as RBF-SVM on some datasets. We now explain why this comparison is useful: RBF-SVM behaves similar to 1-NN in some cases when FK is used (both *JoinAll* and *NoJoin*). But this does not necessarily hurt its test accuracy. Note that FK is represented using the standard one-hot encoding for RBF-SVM and 1-NN. So, FK can contribute to a maximum distance of 2 in a (squared) Euclidean distance between two examples x_i and x_j. But since $\mathbf{X}_R$ is functionally dependent on FK, if $x_i.FK = x_j.FK$, then $x_i.\mathbf{X}_R = x_j.\mathbf{X}_R$. So, if $x_i.FK = x_j.FK$, the only contributor to the distance is $\mathbf{X}_S$. But in many of the datasets, since $\mathbf{X}_S$ is empty ($d_S = 0$), FK becomes the sole determiner of the distances for *NoJoin*. This is akin to sheer *memorization* of a feature's large domain. Since we operate on features with finite domains, test examples will also have FK from that domain. Thus, memorizing FK does not hurt generalization. While this seems similar to how deep neural networks excel at sheer memorization but still offer good test accuracy [52], the models in our setting are not necessarily memorizing all features but rather only FK. A similar explanation holds for the decision tree. If $\mathbf{X}_S$ is not empty, then it will likely play a major role in the distance computations and our setting becomes more similar to the traditional single-table learning setting (no FDs).

We now explain why *NoJoin* deviates from *JoinAll* when the tuple ratio is very low for RBF-SVM. Even if $x_i.FK \neq x_j.FK$, it is possible that $x_i.\mathbf{X}_R = x_j.\mathbf{X}_R$. Suppose the "true" distribution is captured by $\mathbf{X}_R$ (as in OneXr). If the tuple ratio is very low, there might be many FK values but the number of distinct $\mathbf{X}_R$ values might still be small. In this case, given x_i, RBF-SVM (and 1-NN) is more likely to pick an x_j that minimizes the distances on $\mathbf{X}_R$, thus, potentially yielding lower errors. But since *NoJoin* does not have access to $\mathbf{X}_R$, it can only use $\mathbf{X}_S$ and FK. So, if $\mathbf{X}_S$ is mostly noise, the possibility of the model getting "confused" increases. To see why, if there are few other examples that share $x_i.FK$, matching on $\mathbf{X}_S$ becomes more important. Thus, a non-match on FK becomes more likely, which means a non-match on the implicit $\mathbf{X}_R$ becomes more likely, which in turns makes higher errors more likely. But if there are more examples that share $x_i.FK$, then a match on FK is more likely. Thus, as the tuple ratio increases, the gap between *NoJoin* and *JoinAll* decreases, as Figure 3 showed. Internally, RBF-SVM seems more robust to such chance mismatches, since it learns a higher-level relationship between all features compared to 1-NN. Thus, RBF-SVM is more robust to avoiding joins at lower tuples ratios compared to 1-NN.

Finally, the decision tree's internal feature selection and partitioning seems to make it robust to noise from many features. Suppose again the "true" distribution is similar to OneXr. Since FK already encodes all information that $\mathbf{X}_R$ provides, the tree almost always uses FK in its partitioning, often multiple times. This is not necessarily "bad" for test accuracy because test examples share $\mathcal{D}_{FK}$. But when the tuple ratio is extremely low, the chance of $\mathbf{X}_S$ "confusing" the tree against the information FK provides goes up, potentially leading to higher errors with *NoJoin*. *JoinAll* escapes such a confusion due to $\mathbf{X}_R$. If $\mathbf{X}_S$ is empty, then FK will almost surely be used for partitioning. But with very

Table 6: Training errors for the same experiments as Table 2. Bold font marks the cases where the error of *NoJoin* is at least 0.01 higher than *JoinAll*.

Dataset	Decision Tree									1NN	
	Gini			Information			Gain Ratio				
	JoinAll	*NoJoin*	*NoFK*	*JoinAll*	*NoJoin*	*NoFK*	*JoinAll*	*NoJoin*	*NoFK*	*JoinAll*	*NoJoin*
Expedia	0.1425	0.1478	0.1660	0.1442	0.1465	0.1641	0.1455	0.1454	0.1738	0	0
Movies	1.0038	1.0038	1.0476	1.0106	1.0119	1.0545	1.0145	1.0174	1.0551	1.0546	1.0420
Yelp	1.0425	1.0458	1.2149	1.0537	1.0571	1.2290	**1.0650**	**1.0844**	1.2358	1.3050	1.1958
Walmart	0.7339	0.7362	0.8212	0.7362	0.7346	0.8298	0.7373	0.7362	0.8176	0.7045	0.7058
LastFM	0.9153	0.9172	1.0947	0.9153	0.9155	1.1059	0.9247	0.9252	1.1092	1.0140	1.0137
Books	0.8439	0.8440	0.8909	0.8750	0.8750	0.8926	0.8957	0.8953	0.9317	1.0701	1.0540
Flights	0.0004	0.0005	0.0121	0.0008	0.0007	0.0079	0.1247	0.1253	0.1261	0	0

Table 7: Training errors for the same experiments as Table 3. Bold font marks the cases where the error of *NoJoin* is at least 0.01 higher than *JoinAll*.

Dataset	SVM						ANN		Naïve Bayes		Logistic Regression	
	Linear		Polynomial		RBF				BFS		L1	
	JoinAll	*NoJoin*	*JoinAll*	*NoJoin*	*JoinAll*	*NoJoin*	*JoinAll*	*NoJoin*	*JoinAll*	*NoJoin*	*JoinAll*	*NoJoin*
Expedia	0.2038	0.2063	0.1934	0.1955	**0.1913**	**0.2017**	0.1770	0.1774	0.2350	0.2359	0.2057	0.2058
Movies	1.0149	1.0161	0.9851	0.9836	0.9685	0.9718	0.9558	0.9531	1.0347	1.0360	1.0152	1.0172
Yelp	1.1258	1.1239	**1.0838**	**1.0979**	**1.0743**	**1.0927**	1.1759	1.1749	**1.0572**	**1.1081**	**1.1041**	**1.1183**
Walmart	0.8254	0.8261	0.7752	0.7755	0.7456	0.7462	0.6956	0.6951	0.8541	0.8553	0.8028	0.8077
LastFM	**1.0461**	**1.0578**	0.9836	0.9830	0.9451	0.9466	0.9848	0.9852	0.9242	0.9253	0.9447	0.9462
Books	1.0737	1.0761	**1.0053**	**1.0059**	0.9534	0.9557	0.9338	0.9345	1.0235	1.0245	0.9436	0.9469
Flights	0.0874	0.0914	0.0189	0.0190	0	0	0.0451	0.0500	0.1246	0.1285	0.0971	0.1028

few training examples per FK value, the chance of sending it to a wrong partition goes up, leading to higher errors. It turns out that even with just 3 or 4 training examples per FK value, such issues get mitigated. Thus, decision trees seem even more robust to avoiding joins.

5.2 Open Research Questions

While our analysis intuitively explains the behavior of decision trees and RBF-SVMs, there are many open questions for research. Is it possible to quantify the probability of wrong partitioning with a decision tree as a function of the data properties? Is it possible to quantify the probability of mismatched examples being picked by RBF-SVM? Why does the theory of VC dimension predict the opposite of the observed behavior with these models? How do we quantify their generalization if memorization is allowable and what forms of memorization are allowed? Answering these questions would provide deeper insights into the effects of KFKDs/FDs on such classifiers. It could also yield more formal mechanisms to characterize when avoiding joins is feasible beyond just looking at tuple ratios.

There are database dependencies more general than FDs: embedded multi-valued dependencies and join dependencies [45]. How do these dependencies among features affect ML models? There are also conditional FDs, which satisfy FD-like constraints among subsets of rows [45]; how do such data properties affect ML models? Finally, Armstrong's axioms imply that foreign features can be divided into arbitrary subsets before being avoided; this opens up a new trade-off space between avoiding $\mathbf{X}_R$ and using it $\mathbf{X}_R$. How do we quantify this trade-off and exploit it? Answering these questions would open up new connections between data management and ML theory and potentially enable new functionalities for ML analytics systems.

6. MAKING FK FEATURES PRACTICAL

We now discuss two key practical issues caused by a large $|\mathcal{D}_{FK}|$ and study how standard techniques can be adapted to resolve them. Unlike prior work on handling large-domain regular features [8], foreign key features are distinct, since they have coarser-grained side information available in foreign features, which can be exploited.

6.1 Foreign Key Domain Compression

While foreign key features are clearly often useful for *accuracy*, they could make *interpretability* difficult. For example, it is hard to visualize a decision tree that uses a foreign key feature with 1000s of values. Thus, we consider a simple technique from the ML literature to mitigate this issue: *lossy compression*. Essentially, FK with domain $\mathcal{D}_{FK}$ is recoded as $[m]$ (where $m = |\mathcal{D}_{FK}|$). Given a user-specified positive integer "budget" $l \ll m$, we want a mapping $f : [m] \rightarrow [l]$.

A standard unsupervised method to construct f is the *random hashing* trick [48], i.e., randomly map from $[m]$ to $[l]$. We also try a simple supervised method based on filter-based feature selection that we call the *Sort-based* method.

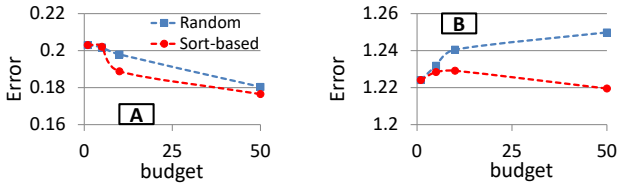


Figure 9: Domain compression. (A) *Flights*. (B) *Yelp*.

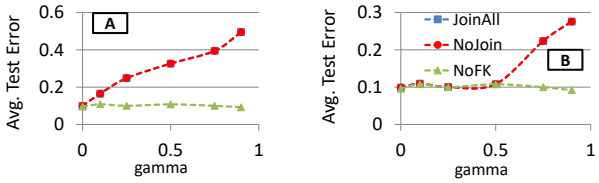


Figure 10: Smoothing. (A) Hashing. (B) $\mathbf{X}_R$-based.

It preserves more of the information contained in FK about Y. It is a greedy approach in which we sort $\mathcal{D}_{FK}$ based on $H(Y|FK = z)$, $z \in \mathcal{D}_{FK}$, compute the differences among adjacent pairs of values, and pick the boundaries corresponding to the top $l-1$ differences (ties broken randomly). This gives us an l-partition of $\mathcal{D}_{FK}$. The intuition is that by grouping FK values that have comparable conditional entropy, $H(Y|f(FK))$ is unlikely to be much higher than $H(Y|FK)$. Note that the lower $H(Y|FK)$ is, the more informative FK is to predict Y.

We empirically compare the above two heuristics using two real datasets for the Gini decision tree with *NoJoin*. Our methodology is as follows. We use the training partition to construct f and then compress FK for the whole dataset. We then use the validation partition and obtain cross-validation errors as before. For random hashing, we report the average across five runs. Figure 9 presents the results. On *Yelp*, both *Random* and *Sort-based* have comparable errors although *Sort-based* is marginally higher, especially as l increases. But on *Flights*, the gap is larger for some values of l although the gap narrows as the l increases. The test error with the whole $\mathcal{D}_{FK}$ ($l = m$) for *NoJoin* on *Flights* was 0.14 (see Table 2). Thus, it is surprising to see an error of only about 0.18 even with such high domain compression. Even more surprisingly, the test error on *Yelp* goes down after domain compression from 1.31 to about 1.22. Overall, these results suggest that FK domain compression, especially with *Sort-based*, is a promising way to resolve the large-domain issue rather than dropping FK.

6.2 Foreign Key Smoothing

Another issue caused by a large $|\mathcal{D}_{FK}|$ is that some FK values might not arise in the train set but arise in the test set or during deployment. This is not the cold start issue, since all FK values are from within the closed $\mathcal{D}_{FK}$, but rather an issue of there not being enough labeled examples to cover all of $\mathcal{D}_{FK}$ well. Typically, this issue is handled using *smoothing*, e.g., Laplacian smoothing for Naive Bayes by adding a pseudocount of 1 to all frequency counts [33]. While similar techniques have been studied for probability estimation using decision trees [36], to the best of our knowledge, this issue has not been handled in general for classification using decision trees. In fact, popular decision tree packages in R simply crash if this issue arises! Note that SVMs, ANNs, and other numeric feature space-based models do not have this issue, since they use one-hot encoding of FK.

We consider a simple solution approach: smooth by *reassigning* an FK value not seen during training to an FK value that was seen. The reassignment can be done in many ways but for simplicity sake, we consider only two unsupervised methods: *random* reassignment and distances using foreign features ($\mathbf{X}_R$). Note that the latter is only feasible in cases where the dimension tables have been procured; the idea is to use the auxiliary information in $\mathbf{X}_R$ to smooth FK rather than just using *JoinAll*. We smooth using $\mathbf{X}_R$ as follows: given a test example with FK not seen during training, obtain an FK seen during training whose corresponding $\mathbf{X}_R$ feature vector has the minimum distance with the given test example's $\mathbf{X}_R$ (ties broken randomly). The distance measure is just a sum of the l_0 distance for categorical features (count of pairwise mismatches) and l_2 distance for numeric features (Euclidean distance).

The intuition for $\mathbf{X}_R$-based smoothing is that if $\mathbf{X}_R$ is part of the "true" distribution, it may yield lower errors than random smoothing, but if $\mathbf{X}_R$ is just noise, both methods become similar. We empirically compare these methods using the `OneXr` simulation scenario in which a lone feature $X_r \in \mathbf{X}_R$ determines the target (with some Bayes error). We introduce a parameter γ that is the ratio of the number of FK values not seen during training to $|\mathcal{D}_{FK}|$. If $\gamma = 0$, smoothing is not needed; as γ increases, more smoothing is needed. Figure 10 presents the results. We see that $\mathbf{X}_R$-based smoothing yields much lower test errors for both *NoJoin* and *JoinAll*. In fact, the smoothed approaches' errors are comparable to *NoFK* and the Bayes error for low values of γ (< 0.5). As γ gets closer to 1, the errors of $\mathbf{X}_R$-based smoothing also increase but not as much as random smoothing. Overall, these results suggest that one could get "the best of both worlds" in a way: even if foreign features are available, rather for using them always as in *JoinAll*, an often viable alternative is to use them as side information for smoothing foreign key features with *NoJoin*, thus still yielding some of the runtime and usability benefits of *NoJoin*.

6.3 Discussion and Limitations

Our results confirm that it is often safe to avoid KFK joins even for popular high-capacity classifiers. Thus, data scientists can use the tuple ratio rule to easily reduce the burden of data sourcing for such classifiers too, not just linear models. We also showed that it is possible to avoid joins safely regardless of whether features are categorical or numeric. This has a new implication for further theoretical analysis of our results because the analysis in [30] relied on the finiteness of the hypothesis space due to all features being categorical. But an infinite hypothesis space does not preclude a finite VC dimension [44]. Extending the theoretical analysis to our more general setting is an open problem. While we focused on star schemas, our results can be easily extended to snowflake schemas as well due to the transitivity of FDs. Our results also apply to single-table data with an acyclic set of FDs, as noted in [30], since a BCNF decomposition can yield a multi-table scenario.

We recap the limitations and assumptions of our work to help data scientists apply our idea in the right context. We focused only on popular classification models but our results hold for both binary and multi-class targets and both categorical and numeric features. If a foreign key is not

generalizable (e.g., search ID in *Expedia*), it cannot be used directly as a feature and so, its corresponding join should not be avoided. Finally, we leave it to future work to study the interplay of our work with cold start techniques and latency trade-offs during model serving.

7. RELATED WORK

Database Dependencies and ML. Optimizing ML over joins of multiple tables was studied in [29, 41, 38, 28], but their goal was primarily to reduce runtimes without affecting ML accuracy. ML over joins was also studied in [50] but their focus was on devising a new ML algorithm. In contrast, our work studied the more fundamental question of whether KFK joins can be avoided safely for ML classifiers. We first demonstrated the feasibility of avoiding joins safely in [30] for linear models. In this work, we revisit that idea for high-capacity classifiers and also empirically verify mechanisms to make foreign key features more practical. Embedded multi-valued dependencies (EMVDs) are database dependencies that are more general than functional dependencies [3]. The implication of EMVDs for probabilistic conditional independence in Bayesian networks was originally described by [34] and further explored by [49]. However, their use of EMVDs still requires computations over all features in the data instance. In contrast, avoiding joins safely omits entire sets of features for complex ML models *without performing any computations* on the foreign features. There is a large body of work on statistical relational learning (SRL) to handle joins that cause duplicates in the fact table [15]. But as mentioned before, our work focuses on the regular IID setting for which SRL might be an overkill.

Feature Selection. The ML and data mining communities have long studied feature selection methods [16]. Our goal is *not* to design new feature selection methods nor is it compare existing ones. Rather, we study if KFKDs/FDs in the schema let us to avoid entire tables a priori for some popular high-capacity classifiers, i.e., "short-circuiting" feature selection using database schema information to reduce the burden of data sourcing. The trade-off between feature redundancy and relevancy is well-studied [16, 51, 25]. The conventional wisdom is that even a feature that is redundant might be highly relevant and thus, unavoidable in the mix [16]. Our work shows that, perhaps surprisingly, even highly relevant foreign features can be safely discarded in many practical classification tasks for many high-capacity classifiers. There is prior work on exploiting FDs in feature selection; [46] infers approximate FDs using the dataset instance and exploits them during feature selection, FOCUS [4] is an approach to bias the input and reduce the number of features, while [7] proposes a measure called consistency to aid in feature subset search. Our work is orthogonal to these algorithms because they all still require computations over all features, while avoiding joins safely *omits foreign features without even looking at them* and obviously, without performing any computations on them. To the best of our knowledge, no feature selection method exhibits such a dramatic capability. Gini and information gain are known to be biased towards large-domain features in decision trees [8]. Different approaches have been studied to resolve this issue [20]. Our work is orthogonal because we study how KFKDs/FDs enable us to ignore foreign features a priori safely. Even with the gain ratio score that is known to mitigate the bias towards large-domain features, our main findings stand. Unsupervised dimensionality reduction methods such as random hashing and PCA are also popular [17]. Our foreign key domain compression techniques for decision trees are inspired by such methods.

Data Integration. Integrating data and features from various sources for ML often requires applying and adapting data integration techniques [31, 9], e.g., integrating features from different data types in recommendation systems [21], sensor fusion [22], dimensionality reduction during feature fusion [14], and controlling data quality during data fusion [11]. Avoiding joins safely can be seen as one schema-based mechanism to reduce the integration burden by predicting a priori if a source table is unlikely to improve accuracy. It is an open challenge to devise similar mechanisms for other types of data sources, say, using other schema constraints, ontology information, and sampling. There is also a growing interest in making data discovery and other forms of metadata management easier [13, 19]. Our work can be seen as a mechanism to verify the potential utility of some of the discovered data sources using their metadata. We hope our work spurs more research in this direction of exploiting ideas from data integration and data discovery to reduce the data sourcing burden for ML tasks.

8. CONCLUSIONS AND FUTURE WORK

It is high time for the data management community to look beyond just building faster ML systems and help reduce the pains of data sourcing for ML. Understanding how fundamental data properties and schema information can simplify end-to-end ML workflows is one promising avenue in this direction. While the idea of avoiding joins safely has been adopted in practice for linear classifiers, in this comprehensive study, we show that it works as well or better for popular high-capacity classifiers too. This goes against the intuition that high-capacity classifiers are typically more prone to overfitting. We hope that our work spurs discussions and new research on simplifying data sourcing for ML.

As for future work, we plan to formally analyze the effects of KFKDs/FDs on high-capacity classifiers using learning theory. Other interesting avenues include understanding the effects of other database dependencies on ML, including regression and clustering models, and designing an automated "advisor" for data sourcing for ML tasks, especially when there are heterogeneous data types and sources.

9. REFERENCES

[1] Personal communications: Facebook friend recommendation system; LogicBlox retail analytics; MakeMyTrip customer analytics.

[2] Nature News. `https://www.nature.com/articles/d41586-017-02190-5`.

[3] S. Abiteboul, R. Hull, and V. Vianu, editors. *Foundations of Databases: The Logical Level*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1st edition, 1995.

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

[5] M. Anderson, D. Antenucci, V. Bittorf, M. Burgess, M. J. Cafarella, A. Kumar, F. Niu, Y. Park, C. Ré, and C. Zhang. Brainwash: A Data System for Feature Engineering. In *CIDR*, 2013.

[6] M. Boehm, M. W. Dusenberry, D. Eriksson, A. V. Evfimievski, F. M. Manshadi, N. Pansare, B. Reinwald, F. R. Reiss, P. Sen, A. C. Surve, and S. Tatikonda. SystemML: Declarative Machine Learning on Spark. *PVLDB*, 9(13):1425–1436, 2016.

[7] M. Dash, H. Liu, and H. Motoda. Consistency based feature selection. In *Proceedings of the 4th Pacific-Asia Conference on Knowledge Discovery and Data Mining, Current Issues and New Applications*, PAKDK, pages 98–109, London, UK, UK, 2000. Springer-Verlag.

[8] H. Deng, G. Runger, and E. Tuv. Bias of importance measures for multi-valued attributes and solutions. In *Proceedings of the 21st International Conference on Artificial Neural Networks - Volume Part II*, ICANN'11, pages 293–300, Berlin, Heidelberg, 2011. Springer-Verlag.

[9] A. Doan, A. Halevy, and Z. Ives. *Principles of Data Integration*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1st edition, 2012.

[10] P. Domingos. A Unified Bias-Variance Decomposition and its Applications. In *Proceedings of 17th International Conference on Machine Learning*, 2000.

[11] X. L. Dong and D. Srivastava. Big data integration. *PVLDB*, 6(11):1188–1189, 2013.

[12] X. Feng, A. Kumar, B. Recht, and C. Ré. Towards a unified architecture for in-rdbms analytics. In *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*, SIGMOD '12, pages 325–336, New York, NY, USA, 2012. ACM.

[13] R. C. Fernandez, Z. Abedjan, S. Madden, and M. Stonebraker. Towards Large-scale Data Discovery: Position Paper. In *Proceedings of the Third International Workshop on Exploratory Search in Databases and the Web*, ExploreDB '16, pages 3–5, New York, NY, USA, 2016. ACM.

[14] Y. Fu, L. Cao, G. Guo, and T. S. Huang. Multiple feature fusion by subspace learning. In *Proceedings of the 2008 International Conference on Content-based Image and Video Retrieval*, CIVR '08, pages 127–134, New York, NY, USA, 2008. ACM.

[15] L. Getoor and B. Taskar. *Introduction to Statistical Relational Learning)*. The MIT Press, 2007.

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

[17] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning: Data mining, Inference, and Prediction*. Springer-Verlag, 2001.

[18] J. M. Hellerstein, C. Ré, F. Schoppmann, D. Z. Wang, E. Fratkin, A. Gorajek, K. S. Ng, C. Welton, X. Feng, K. Li, and A. Kumar. The MADlib Analytics Library or MAD Skills, the SQL. *PVLDB*, 5(12):1700–1711, 2012.

[19] J. M. Hellerstein, V. Sreekanti, J. E. Gonzalez, J. Dalton, A. Dey, S. Nag, K. Ramachandran, S. Arora, A. Bhattacharyya, and S. Das. Ground: A Data Context Service. In *CIDR*, 2017.

[20] T. Hothorn, K. Hornik, and A. Zeileis. Unbiased recursive partitioning: A conditional inference framework. *JOURNAL OF COMPUTATIONAL AND GRAPHICAL STATISTICS*, 15(3):651–674, 2006.

[21] Y. Jing, D. Liu, D. Kislyuk, A. Zhai, J. Xu, J. Donahue, and S. Tavel. Visual search at pinterest. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '15, pages 1889–1898, New York, NY, USA, 2015. ACM.

[22] B. Khaleghi, A. Khamis, F. O. Karray, and S. N. Razavi. Multisensor data fusion: A review of the state-of-the-art. *Information Fusion*, 14(1):28–44, Jan. 2013.

[23] M. Kim, T. Zimmermann, R. DeLine, and A. Begel. Data Scientists in Software Teams: State of the Art and Challenges. *IEEE Transactions on Software Engineering*, PP(99), 2017.

[24] D. P. Kingma and J. Ba. Adam: A Method for Stochastic Optimization. In *3rd International Conference for Learning Representations (ICLR)*, 2015.

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

[26] P. Konda, A. Kumar, C. Ré, and V. Sashikanth. Feature Selection in Enterprise Analytics: A Demonstration using an R-based Data Analytics System. *PVLDB*, 6(12):1306–1309, 2013.

[27] T. Kraska, A. Talwalkar, J. C. Duchi, R. Griffith, M. J. Franklin, and M. I. Jordan. MLbase: A Distributed Machine-learning System. In *CIDR*, 2013.

[28] A. Kumar, M. Jalal, B. Yan, J. Naughton, and J. M. Patel. Demonstration of Santoku: Optimizing Machine Learning over Normalized Data. *PVLDB*, 8(12):1864–1867, 2015.

[29] A. Kumar, J. Naughton, and J. M. Patel. Learning generalized linear models over normalized data. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, SIGMOD '15, pages 1969–1984, New York, NY, USA, 2015. ACM.

[30] A. Kumar, J. Naughton, J. M. Patel, and X. Zhu. To join or not to join?: Thinking twice about joins before feature selection. In *Proceedings of the 2016 International Conference on Management of Data*, SIGMOD '16, pages 19–34, New York, NY, USA, 2016. ACM.

[31] Y. Li and A. Ngom. Data Integration in Machine Learning. In *IEEE International Conference on Bioinformatics and Biomedicine (BTBM)*, 2015.

[32] Y. Low, J. E. Gonzalez, A. Kyrola, D. Bickson, C. E. Guestrin, and J. Hellerstein. GraphLab: A New Framework For Parallel Machine Learning. In *UAI*, 2010.

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

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

[35] N. Polyzotis, S. Roy, S. E. Whang, and M. Zinkevich. Data management challenges in production machine learning. In *Proceedings of the 2017 ACM International Conference on Management of Data*, SIGMOD '17, pages 1723–1726, New York, NY, USA, 2017. ACM.

[36] F. Provost and P. Domingos. Tree Induction for Probability-Based Ranking. *Machine Learning*, 52(3):199–215, 2003.

[37] R. Ramakrishnan and J. Gehrke. *Database Management Systems*. McGraw-Hill, Inc., 2003.

[38] S. Rendle. Scaling Factorization Machines to Relational Data. *PVLDB*, 6(5):337–348, 2013.

[39] R. Ricci, E. Eide, and C. Team. Introducing CloudLab: Scientific Infrastructure for Advancing Cloud Architectures and Applications. *; login:: the magazine of USENIX & SAGE*, 39(6):36–38, 2014.

[40] A. I. Schein, A. Popescul, L. H. Ungar, and D. M. Pennock. Methods and metrics for cold-start recommendations. In *Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '02, pages 253–260, New York, NY, USA, 2002. ACM.

[41] M. Schleich, D. Olteanu, and R. Ciucanu. Learning linear regression models over factorized joins. In *Proceedings of the 2016 International Conference on Management of Data*, SIGMOD '16, pages 3–18, New York, NY, USA, 2016. ACM.

[42] D. Sculley, G. Holt, D. Golovin, E. Davydov, T. Phillips, D. Ebner, V. Chaudhary, M. Young, J.-F. Crespo, and D. Dennison. Machine Learning: The High Interest Credit Card of Technical Debt. In *SE4ML: Software Engineering for Machine Learning (NIPS 2014 Workshop)*, 2014.

[43] V. Shah, A. Kumar, and X. Zhu. Are Key-Foreign Key Joins Safe to Avoid when Learning High-Capacity Classifiers? (Technical Report). `http://cseweb.ucsd.edu/~arunkk/hamlet`.

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

[45] A. Silberschatz, H. Korth, and S. Sudarshan. *Database Systems Concepts*. McGraw-Hill, Inc., New York, NY, USA, 5 edition, 2006.

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

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

[48] K. Weinberger, A. Dasgupta, J. Langford, A. Smola, and J. Attenberg. Feature hashing for large scale multitask learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, pages 1113–1120, New York, NY, USA, 2009. ACM.

[49] S. K. M. Wong, , C. J. Butz, and Y. Xiang. A Method for Implementing a Probabilistic Model as a Relational Database. In *Proceedings of the Eleventh conference on Uncertainty in artificial intelligence*, 1995.

[50] X. Yin, J. Han, J. Yang, and P. S. Yu. Crossmine: Efficient classification across multiple database relations. In *Proceedings of the 2004 European Conference on Constraint-Based Mining and Inductive Databases*, pages 172–195, Berlin, Heidelberg, 2005. Springer-Verlag.

[51] L. Yu and H. Liu. Efficient Feature Selection via Analysis of Relevance and Redundancy. *Journal of Machine Learning Research*, 5, Dec. 2004.

[52] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals. Understanding Deep Learning Requires Rethinking Generalization. In *International Conference on Learning Representations (ICLR)*, 2017.

[53] Y. Zhang, W. Zhang, and J. Yang. I/O-Efficient Statistical Computing with RIOT. In *Data Engineering (ICDE), 2010 IEEE 26th International Conference on*, 2010.

APPENDIX

A. FOREIGN KEY SKEW

The regular `OneXr` scenario samples FK uniformly randomly from $\mathcal{D}_{FK}$ (step 3 in the procedure). We now ask if a *skew* in the distribution of FK values could widen the gap between *JoinAll* and *NoJoin*. To study this scenario, we modify the data generation procedure slightly: in step 3, we sample FK values with a Zipfian skew or a needle-and-thread skew. The Zipfian skew simply uses a Zipfian distribution for $P(FK)$ controlled by the Zipfian skew parameter. The needle-and-thread skew allocates a large probability mass (parameter p) to a single FK value (the "needle") and uniformly distributes the rest of the probability mass to all other FK values (the "thread"). For the linear model case, [30] reported that as the skew parameters increased, the gap widened. Figure 11 presents the results for the decision tree.

Surprisingly, the gap between *NoJoin* and *JoinAll* does not widen significantly no matter how much skew introduced in either the Zipfian or the needle-and-thread case! This result further affirms the remarkable robustness of the decision tree when discarding foreign features. As expected, *NoFK* is better when n_S is low, while overall, *NoJoin* is quite similar to *JoinAll*.

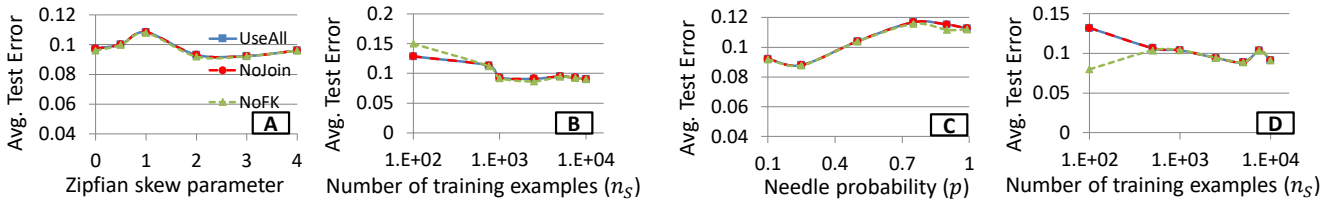


Figure 11: Scenario `OneXr` simulations with skew in $P(FK)$. (A-B) Zipfian skew. (C-D) Needle-and-thread skew. For (A) and (C), we vary the respective skew parameter (Zipfian skew parameter and needle probability), while fixing $(n_S, n_R, d_S, d_R) = (1000, 40, 4, 4)$. For (B) and (D), we vary n_S, while fixing $(n_R, d_S, d_R) = (40, 4, 4)$, the Zipfian skew parameter to 2 for (B), and the needle probability to 0.5 for (D).