RESEARCH-ARTICLE

Leakage in data mining: Formulation, detection, and avoidance

SHACHAR KAUFMAN, Tel Aviv University, Tel Aviv-Yafo, Tel Aviv District, Israel

SAHARON ROSSET, Tel Aviv University, Tel Aviv-Yafo, Tel Aviv District, Israel

CLAUDIA PERLICH

ORI STITELMAN

Published: 18 December 2012
Accepted: 01 May 2012
Revised: 01 March 2012
Received: 01 November 2011

Leakage in Data Mining: Formulation, Detection, and Avoidance

SHACHAR KAUFMAN and SAHARON ROSSET, Tel Aviv University
CLAUDIA PERLICH and ORI STITELMAN, m6d

Deemed “one of the top ten data mining mistakes”, leakage is the introduction of information about the data mining target that should not be legitimately available to mine from. In addition to our own industry experience with real-life projects, controversies around several major public data mining competitions held recently such as the INFORMS 2010 Data Mining Challenge and the IJCNN 2011 Social Network Challenge are evidence that this issue is as relevant today as it has ever been. While acknowledging the importance and prevalence of leakage in both synthetic competitions and real-life data mining projects, existing literature has largely left this idea unexplored. What little has been said turns out not to be broad enough to cover more complex cases of leakage, such as those where the classical independently and identically distributed (i.i.d.) assumption is violated, that have been recently documented. In our new approach, these cases and others are explained by explicitly defining modeling goals and analyzing the broader framework of the data mining problem. The resulting definition enables us to derive general methodology for dealing with the issue. We show that it is possible to avoid leakage with a simple specific approach to data management followed by what we call a learn-predict separation, and present several ways of detecting leakage when the modeler has no control over how the data have been collected. We also offer an alternative point of view on leakage that is based on causal graph modeling concepts.

Categories and Subject Descriptors: H.2.8 [**Database Management**]: Database Applications—*Data mining*

General Terms: Theory, Algorithms

Additional Key Words and Phrases: Data mining, leakage, predictive modeling

ACM Reference Format:
Kaufman, S., Rosset, S., Perlich, C., and Stitelman, O. 2012. Leakage in data mining: Formulation, detection, and avoidance. ACM Trans. Knowl. Discov. Data. 6, 4, Article 15 (December 2012), 21 pages.
DOI = 10.1145/2382577.2382579 http://doi.acm.org/10.1145/2382577.2382579

1. INTRODUCTION

Deemed “one of the top ten data mining mistakes” [Nisbet et al. 2009], leakage in data mining (henceforth, *leakage*) is the introduction of information about the target of a data mining problem that should not be legitimately available to mine from. A trivial example of leakage would be a model that uses the target itself as an input, thus concluding, for example, that “it rains on rainy days”. In practice, the introduction of illegitimate information is unintentional, and facilitated by the data collection, aggregation, and preparation process. It is usually subtle and indirect, making it hard to detect and eliminate. Leakage is undesirable as it may lead a *modeler*, someone trying to solve the problem, to learn a suboptimal solution, which would in fact be

S. Kaufman and S. Rosset were partially supported by grant 1227/09 from the Israeli Science Foundation, and S. Kaufman was supported by a fellowship from the Edmond J. Safra Center for Bioinformatics at Tel-Aviv University.
Authors’ addresses: S. Kaufman and S. Rosset, Tel Aviv University, P.O. Box 39040, Tel Aviv 69978, Israel; C. Perlich and O. Stitelman, m6d.

DOI 10.1145/2382577.2382579 http://doi.acm.org/10.1145/2382577.2382579

outperformed in deployment by a leakage-free model that could have otherwise been built. At the very least leakage leads to overestimation of the model's performance. A *client* for whom the modeling is undertaken is likely to discover the sad truth about the model when performance in deployment is found to be systematically worse than the estimate promised by the modeler. Even then, identifying leakage as the reason might be highly nontrivial.

Existing literature mentions leakage and acknowledges its importance and prevalence in both synthetic competitions and real-life data mining projects (e.g., Nisbet et al. [2009] and Kohavi et al. [2000]). However these discussions lack several key ingredients. First, they do not present a general and clear theory of what constitutes leakage. Second, these sources do not suggest practical methodologies for leakage detection and avoidance that modelers could apply to their own statistical learning problems. This gap in theory and methodology could be the reason that several recent major data mining competitions such as the KDD Cup 2008, or the INFORMS 2010 Data Mining Challenge, though judiciously organized by capable individuals, suffered severe leakage. In many cases, attempts to fix leakage resulted in the introduction of new leakage, which is more subtle and thus harder to deal with. Other competitions such as KDD Cup 2007 and IJCNN 2011 Social Network Challenge were affected by a second form of leakage that is specific to competitions. Leakage from available external sources undermined the organizers' implicit true goal of encouraging submissions that would actually be useful for the domain. These cases, in addition to our own experience with leakage in the industry and as competitors in and organizers of data mining challenges, are examined in more detail in Section 2. We also revisit them in later sections to provide a more concrete setting for our discussion.

The major contribution of this article, aside from raising awareness to an important issue which we believe is often overlooked, is a proposal in Section 3 for a formal definition of leakage. This definition covers both the common case of leaking features and more complex scenarios that have been encountered in predictive modeling competitions. We use this formulation to facilitate leakage avoidance in Section 4, and suggest in Section 5 methodology for detecting leakage when there is limited or no control over how the data have been collected. This methodology should be particularly useful for practitioners in predictive modeling, as well as for prospective competition organizers. Finally, Section 7 takes a different approach and reviews the discussion of leakage from a causal perspective, formalizing certain types of leakage in terms of causal inference.

2. LEAKAGE IN THE KDD LITERATURE

The subject of leakage has been visited by several data mining textbooks as well as a few scientific papers. Most of the papers we refer to are related to data mining competitions rather than real applications. This is probably due to practitioners either missing or locating and fixing leakage issues without reporting it; we shall return to this point at the end of the section. The following is a short chronological review, collecting examples to be used later as case studies for our proposed definition of leakage.

In the context of predictive modeling, Pyle [1999, 2003, 2009] refers to the phenomenon which we call here leakage as *Anachronisms* (something that is out of place in time), and says that "too good to be true" performance is "a dead giveaway" of its existence. The author suggests turning to *exploratory data analysis* in order to find and eliminate leakage sources. We will also discuss detection approaches in Section 5. Nisbet et al. [2009] refer to the issue as "leaks from the future" and claim it is "one of the top 10 data mining mistakes". They repeat the same basic insights, but also do not suggest a general definition or methodology to correct or prevent leakage. These titles provide a handful of elementary but common examples of leakage. Two representative ones are the following.

—An "account number" feature, for the problem of predicting whether a potential customer would open an account at a bank. Obviously, assignment of such an account number is only done after an account has been opened.
—An "interviewer name" feature, in a cellular company churn prediction problem. While the information "who interviewed the client when they churned" appears innocent enough, it turns out that a specific salesperson was assigned to take over cases where customers had already notified they intend to churn.

Kohavi et al. [2000] describe the introduction of leaks in data mining competitions as giveaway attributes that predict the target because they are downstream in the data collection process. The authors give an example in the domain of retail Web site data analytics where for each page viewed the target is whether the user would leave or stay to view another page. A leaking attribute is the "session length", which is the total number of pages viewed by the user during this visit to the Web site. This attribute is added to each page-view record at the end of the session. A solution is to replace this attribute with "page number in session" which describes the session length up to the current page, where prediction is required.

Subsequent work by Kohavi et al. [2003] presents the common business analysis problem of characterizing big spenders among customers. The authors explain that this problem is prone to leakage since immediate triggers of the target (e.g., a large purchase or purchase of a diamond) or consequences of the target (e.g., paying a lot of tax) are usually available in collected data and need to be manually identified and removed. To show how correcting for leakage can become an involved process, the authors also discuss the more complex situation where removing the information "total purchase in jewelry" caused information of "no purchases in any department" to become fictitiously predictive. This is because each customer found in the database is there in the first place due to some purchase, and if this purchase is not in any department (still available), it has to be jewelry (which has been removed). They suggest defining analytical questions that should suffer less from leaks, such as characterizing a "migrator" (a user who is a light spender but will become a heavy one) instead of characterizing the "heavy spender". The idea is that it is better to ask analytical questions that have a clear temporal "cause and effect" structure. Of course leaks are still possible, but much harder to introduce by accident and much easier to identify. We return to this idea in Section 3. A later article by the authors [Kohavi et al. 2004] reiterates the previous discussion, and adds the example of the "use of free shipping", where a leak is introduced when free shipping is provided as a special offer with large purchases.

Rosset et al. [2007] discuss leakage encountered in the 2007 KDD Cup competition. In that year's contest there were two related challenges concerning derivatives of the famous Netflix Prize dataset. The training dataset provided by the organizers consisted of more than 100 million ratings (a rating is on a scale from 1 to 5 integral stars) from over 480 thousand randomly chosen, anonymous users on nearly 18 thousand movie titles. The ratings in the training dataset were given by the users between the years 1998 and 2005. Naturally, not all users rated all movies, and in fact the user/ratings matrix is sparse. The first challenge, "Who Rated What", was to predict which users rated which movies during 2006. Only a subset of the movies in the dataset was of interest in this challenge, denoted S_1. For this task, a test set of (user, movie) pairs was given to competitors (with movies from S_1 only), and they were asked to predict for which of these pairs the user actually gave a rating to the movie during 2006. The second challenge, "How Many Ratings", was to predict the total number of ratings that

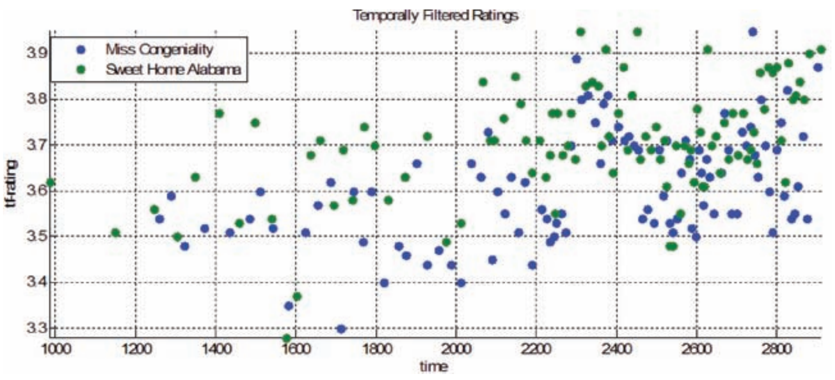

Fig. 1. Different movies in the Netflix database share dynamic properties due to shared causal ancestors such as changes to the review form format, improvements to the Netflix recommender system, advertisement campaigns, and premieres.

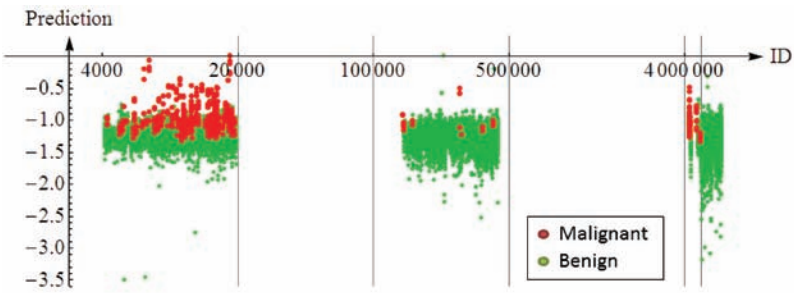

Fig. 2. In the KDD Cup 2008 breast cancer prediction challenge, innocent looking "patient ids" were apparently a trace of the various medical sources joined to create the data. As some sources (e.g., clinical studies) selected only or mostly patients who have cancer, the patient id is a leak for predicting cancer.

were given during 2006 to each of the movies. Here, too, only a subset of the movies was of interest, denoted S_2. The movie sets S_1 and S_2 were disjoint, however, Rosset et al.'s winning submission managed to use the information from the test set provided with the first challenge for improving prediction in the second challenge. From this test set, the authors were able to infer approximately the total number of ratings that each movie in S_1 received during 2006. This involved a complex process taking advantage of the way (user, movie) pairs were sampled to the "Who Rated What" test set. Given these (approximate) numbers of 2006 ratings for each movie in S_1, the authors then built a standard supervised learning model describing dependence of 2006 activity on history prior to 2006, and applied the model to the movies in S_2, to predict their 2006 numbers. This was extremely advantageous because the nature of the rating process changed over time, and the winners essentially inferred the dynamics of movies in S_2 based on the almost-observed dynamics in S_1 (to see why this is a sensible thing to do see Figure 1). We shall revisit this complex example in Section 3, where this case will motivate us to extend our definition of leakage beyond leaking features.

Two medical data mining contests held the following year and which also exhibited leakage are discussed in Perlich et al. [2008] and Rosset et al. [2010]. KDD Cup 2008 dealt with cancer detection from mammography data. Analyzing the data for this competition, the authors point out that the "Patient ID" feature (dismissed as irrelevant by most competitors) has tremendous and unexpected predictive power (see Figure 2). They hypothesize that multiple clinical studies, institutions, or equipment sources were joined to compile the data, and that some of these sources were assigned their population with prior knowledge of the patient's condition. Leakage was thus facilitated by assigning consecutive patient IDs for data from each source, that is, the merge was done without obfuscating the source. In other words, patient IDs are a trace of the assignment of the patient to one of several sources of competition data; since some of these sources (e.g., clinical studies) selected an imbalanced population of sick/healthy patients, this is a leak for a model that is required to *predict* cancer.

The INFORMS Data Mining Challenge 2008 competition, held the same year, addressed the problem of pneumonia diagnosis based on patient information from hospital records. The target was originally embedded as a special value of one or more features in the data given to competitors. The organizers removed these values, however, it was possible to identify traces of such removal, constituting the source of leakage in this example (e.g., a record with all condition codes missing, similarly to Kohavi's jewelry example).

Also in the recent work by Rosset et al. [2010], the concept of identifying and harnessing leakage has been openly addressed as one of three key aspects for winning data mining competitions. This work provides the intuitive definition of leakage as "the unintentional introduction of predictive information about the target by the data collection, aggregation and preparation process". The authors mention that leakage might be the cause of many failures of data mining applications, and give the illustrative example of predicting people who are likely to be sick by looking at how many work days they would end up missing. They also describe a real-life business intelligence project at IBM where potential customers for certain products were identified, among other things, based on keywords found on their Web sites. This turned out to be leakage since the Web site content used for training had been sampled at the point in time where the potential customer has already become a customer, and where the Web site contained traces of the IBM products purchased, such as the product name "Websphere" (e.g., in a press release about the purchase or a specific product feature the client is using).

Kaggle,[1] a Web-based platform for public modeling competitions, hosted a contest in 2010 with the goal to predict HIV progression given the virus DNA sequence and additional patient information. It soon became clear that the data was partially sorted by lexicographical order of the DNA sequences. The training set consisted of the beginning of the sorted list whereas the test set was taken from the end of the list. This already is highly suboptimal for modeling, but to make matters worse, the last third of the training set somehow had no positives whatsoever. Instead, the test set was announced to have a higher positive rate than the training set. Looking at placement of test records according to the DNA lexicographical order, the beginning of the test set was in the range of that last third of the training set which contained no positives. It was rather easy to figure out that all those test set records were surely positives, and in addition to flagging them as positive in the submission, competitors could add them to the training set as additional training examples to improve prediction for all other test instances.

The INFORMS and IJCNN competitions held in late 2010 and early 2011 are fresh examples of how leakage continues to plague predictive modeling problems and competitions in particular. The INFORMS 2010 Data Mining Challenge required participants to develop a model that predicts stock price movements, over a fixed one-hour horizon, at five-minute intervals. Competitors were provided with intraday trading data showing stock prices, sectoral data, economic data, experts' predictions and indices. These data were segmented to a training database, on which participants were expected to build their predictive models, and a test database that was used by

[1]www.kaggle.com.

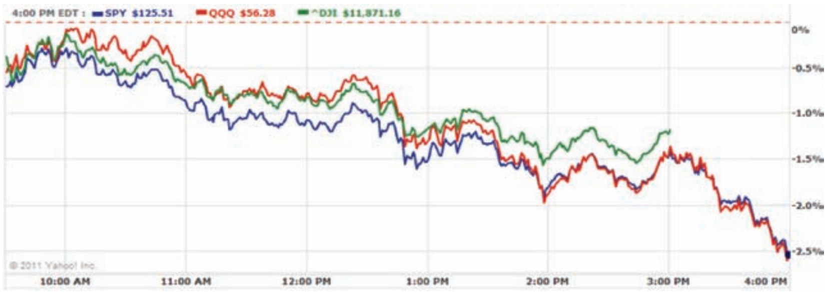

Fig. 3. INFORMS 2010 financial prediction competition. The test set contained "future" data for time series other than the target's. Due to cointegration in this domain (a statistical temporal connection between the target and the other series, as defined by Engle and Granger [1987]), this was a serious leak.

the organizers to evaluate submissions. The surprising results were that about 30 participating groups achieved more than 0.9 AUC, with the best model surpassing 0.99 AUC. Had these models been legitimate they would have indeed made a "big impact on the finance industry" as the organizers had hoped, not to mention making their operators very wealthy individuals. Unfortunately, however, it became clear that although some steps had been taken to prevent competitors from "looking up the answers" (the underlying target stock's identity was not revealed, and the test set did not include the variable being predicted), it was still possible to build models that rely on data from the future. Having data from the future for the explanatory variables, some of which are highly correlated with the target (more precisely, cointegrated [Engle and Granger 1987], e.g., a second stock within the same sector as the target stock) was the true driver of success for these models (see Figure 3). Having access to publicly available stock data such as Yahoo/Google Finance, which allows finding at least good candidates for the identity of the target stock, consequently revealing all test values, could also be used (but apparently no competitor had undertaken this). The organizers held two rankings of competitors, one where future information was allowed and another where it was forbidden, however, in the end they had to admit that verifying future information was not used was impossible, and that it was probable that all models were tainted, as all modelers had been exposed to the test set.

The IJCNN 2011 Social Network Challenge presented participants with anonymized 7,237,983 edges from an undisclosed online social network and asked to predict which of an additional set of 8,960 potential edges are in fact realized on the network as well. The winners have recently reported [Narayanan et al. 2011] they had been able to recognize, through sophisticated analysis, that the social network in question was Flickr and then to deanonymize the majority of the data. This allowed them to use edges available from the online Flickr network to correctly predict over 60% of edges that were identified, while the rest had to be handled classically using legitimate prediction. Similarly to other cases that have been mentioned, these rogue solutions are sometimes so elegant and insightful that they carry merit in their own right. The problem is that they do not answer the original question presented by the organizers.

Finally, it has recently come to our attention that the 2011 KDD Cup also suffered leakage. This time the target (of the "second track" of the competition) was classification of songs to either "rated highly by user" or "not rated by user". The massive dataset was taken from Yahoo Music, and for this track the organizers removed timestamps from all ratings in the data. The team which took second place noticed that the organizers had not reshuffled the data, leaving each user's ratings sorted chronologically. Because temporal effects have a significant impact on ratings (an impact the organizers were trying to eliminate hoping for new insights to be introduced by competitors), this insight was instrumental in that team's solution [Laiy et al. 2011]. It is worth mentioning that this is a relatively rare example where a model contains leaks only within the competition, while it is perfectly acceptable in the relevant domain, because timestamps are in fact typically available for online rating prediction or rating classification.

One observation that should be apparent by now based on the provided examples is the relative frequency with which the problem occurs in competitions. Why is leakage so common here as compared to the apparently much less frequent industry examples?

The first question is whether indeed leakage is more common in competitions or whether we are facing a sample selection bias. In competitions we have thousands of capable data scientists actively looking for anything that can give them an edge. Conversely, most projects have only teams of 2–3 people working on one dataset. In addition, in projects the motivation to explicitly look for leakage is limited. Finally, there is close to no incentive for publishing the occurrence of leakage in the framework of projects whereas winning entries in competitions are typically published in the corresponding workshops. This reinforces our position, that while documented much less often in practical applications of data mining, leakage is no less likely to appear here than in the competitions. And while it may not be exploited to the fullest, the true performance of the model matters much more if it is ultimately to be used in production (e.g., commercially). On the other hand, there are notable differences between projects and competitions that could indeed affect the relative probability of leakage. While they both share the same type of data, often a similar data generation process, and on some level the same objective—build a well-performing predictive model—there often are relevant differences.

(1) Competitions have a near-exclusive focus on predictive algorithm performance on a very specific test set. So organizers may have less incentive to worry about leakage and the true usability of the model, which may lead to willingness to incur the risk of leakage.
(2) The degree of separation between defining the actual task, preparing the data, and executing the actual modeling is notably different. In the context of a competition, the modeler has no influence on the task or the data preparation. In well-executed projects one would expect the modeler to be involved deeply in all of the aforesaid and as a result able to prevent some forms of leakage.
(3) The level of domain knowledge of the competition modeler is typically lower and in addition, due to concerns of privacy, many contests are in the habit of meticulously sterilizing the data. This makes leakage harder to identify (but not necessarily less likely to exist) during modeling.
(4) Competitions typically have higher data requirements and some degree of flexibility on the selection of both task and data. Projects are often forced to start a pilot on notably unsuitable data and there is typically some pressure to answer the question even if by objective standards this should not be done, including due concerns about leakage.
(5) One major risk for leakage in projects is the skill level of the participants. Most competition organizers are very experienced modelers and are in the habit of thinking about concepts like leakage. Leaving the data preparation to some inexperienced modeler, "data technician", or database expert is bound to lead to leakage. Also, many large organizations have physical or conceptual institutional "fire-walls" that do not allow even the skilled modeler to directly interact with the data.

Points 1 through 3 are in favor of leakage being a more common problem in competitions, whereas the latter points are arguments for leakage being more common on real projects (or at least their pilot studies).

Overall, the issue of leakage has been observed in various contexts and problem domains, with a natural focus on predictive modeling. It is also evident that this issue remains highly relevant today. However, none of the discussions that we could find has addressed the issue in a general way, or suggested methodology for handling it. In the following section we make our attempt to derive a formal definition of leakage.

3. FORMULATION

3.1. Preliminaries and Legitimacy

In our discussion of leakage we shall define the roles of client and modeler as in Section 1, and consider the standard framework of supervised learning and its generalizations, where we can discuss examples, targets, and features. We assume the reader is familiar with these concepts. For a complete reference see Hastie et al. [2009]. Let us just lay out our notation and say that in our framework we receive from an axiomatic data preparation stage an *ordered set* of multivariate *observations* $\mathcal{W} = (\mathcal{X}, \mathcal{y})$. $\mathcal{y}$ is the outcome or target ordered set with individual elements y. Similarly, $\mathcal{X}$ and X are the feature-vector ordered set and element, respectively. Components of feature vectors are individual features, denoted χ (ordered set) and x (element). Target and feature-vector elements y and X pertaining to the same element of $\mathcal{W}$ are said to be $\mathcal{W}$*-associated*. The modeler's goal is to infer the value of a target element, from its associated feature-vector element and from a separate group of observations, called the training examples $\mathbf{W_{tr}}$. The solution to this problem is a *model* $\hat{\mathcal{y}} = \mathbb{M}(\mathcal{X}, \mathbf{W_{tr}})$. We say that the model's *observational inputs* for inferring $\hat{\mathcal{y}}$ are $\mathcal{X}$ and $\mathbf{W_{tr}}$, and this relation between the various entities in the framework is the base for our discussion.

In the case of the breast cancer detection problem, for example, we have one observation per patient (a "patient record"). Target elements $y_i = \mathcal{y}(i)$ take the binary value "does patient i have cancer?" coded as either 0 for "no" or 1 for "yes". Feature-vector elements X_i could be random vectors describing the "patient id", an integer isomorphic to i, as the first component of the vector and 1000 pixel grey-level values taken from a mammography image as the remaining components of the vector. Elements X_i and y_j taken from the same patient record $i = j$ are $\mathcal{W}$-associated. A fixed set of patient records constitute our training examples $\mathbf{W_{tr}}$, the data given by the organizers to competitors.

Models containing leaks are a special case of the broader class of illegitimate or unacceptable models. At this level, *legitimacy*, which is a key concept in our formulation of leakage, is completely abstract. Every modeling problem sets its own rules for what constitutes a legitimate or acceptable solution and different problems, even if using the same data, may have wildly different views on legitimacy. For example, a solution could be considered illegitimate if it is too complex; say if it uses too many features or if it is not linear in its features, or even if it is too expensive to implement.

However, our focus here is on leakage, which is a specific form of illegitimacy that is, with respect to a target element, an intrinsic property of the observational inputs of a model. This form of illegitimacy remains partly abstract, but could be further defined as follows: Let u be some observable. We say a second observable v is *u leakage legitimate* if v is observable to the client for the purpose of inferring u. In this case we write $v \in \mathit{legit}\{u\}$.

A fully concrete meaning of leakage legitimacy is built-in to any specific inference problem. The trivial legitimacy rule, going back to the first example of leakage given in Section 1, is that a target element must never be used for inferring itself:

$$y \notin \mathit{legit}\{y\}. \tag{1}$$

We could use this rule if we wanted to disqualify the winning submission to the IJCNN 2011 Social Network Challenge, for it, however cleverly, eventually uses some of the targets themselves for inference. This condition should be abided by all problems, and we refrain from explicitly mentioning it for the remaining examples we shall discuss.

Naturally, a model contains leaks with respect to a target element y if its observational inputs are y-illegitimate. This is a challenging construct to work with since legitimacy is defined on the inputs as a whole. Fortunately, however, for many problems the legitimacy definition is separable which means that it can be further broken down to a legitimacy condition on subsets of inputs that may be considered separately. The discussion that follows focuses on cases where legitimacy is separable down to the level of individual observables. We proceed along the two types of observational inputs in our framework: features and training examples.

3.2. Leaking Features

We begin with the more common case of *leaking features*. First we must extend our abstract definition of leakage legitimacy to entire ordered sets: Let u be some ordered set of observables. We say a second ordered set v is *u-legitimate* if, for every pair of elements of u and v, u and v respectively, which are $\mathcal{W}$-associated, v is u-legitimate. We use the same notation as we did for individual observables in Section 3.1, and write that $v \in legit\{u\}$.

Leaking features are then covered by a simple condition for the absence of leakage:

$$\forall \chi \text{ component of } \mathcal{X},\ \chi \in legit\{\mathcal{Y}\}. \tag{2}$$

That is, any feature made available by the data preparation process is deemed legitimate by the precise formulation of the modeling problem at hand, element by element with respect to its matching target.

The prevailing example for this type of leakage is what we call the *no-time-machine* requirement. In the context of predictive modeling, it is implicitly required that a legitimate model only build on features with information from a time earlier (or sometimes, no later) than that of the target. Formally, $\mathcal{X}$ and $\mathcal{Y}$, are defined over some time axis t (not necessarily physical time). Prediction is required by the client for a target element y at time $t\{y\}$. Each feature x (one of the components of X) associated with an observation is unobservable to the client until $t\{x\}$ from then on it is observable. Let $t\{\mathcal{Y}\}$ denote the ordered set resulting from element-wise application of the operator $t\{y\}$ on the ordered set $\mathcal{Y}$. Similarly define the ordered set $t\{\chi\}$. We then have:

$$t\{\chi\} < t\{\mathcal{Y}\} \Leftrightarrow \chi \in legit\{\mathcal{Y}\}. \tag{3}$$

Such a rule should be read as: A legitimate feature is an ordered set whose every element is observable to the client earlier than its $\mathcal{W}$-associated target element.

Note that the different definitions of the "timestamping" operator t for features and targets is crucial. A good example for its necessity is leakage in the financial world, which relates to the date when information becomes public, and thus observable to the client using a hypothetical financial model (assuming the client is not a rogue inside trader). Specifically, stock-price prediction models would be highly "successful" should they use quarterly data assuming they are available a day after the quarter ends, whereas in reality they are usually publicly known only about three weeks later. We therefore define leakage legitimacy in the predictive modeling case using the concept of observability time of the features and prediction time of the target.

While the simple no-time-machine requirement is indeed the most common case, one could think of additional scenarios which are still covered by condition (2). A simple extension is to require features to be observable a sufficient period of time prior to $t\{y\}$

as in (4) to follow in order to preclude any information that is an immediate trigger of the target. One reason why this might be necessary is that sometimes it is too limiting to think of the target as pertaining to a point-in-time, only to a rough interval. Using data observable close to $t\{y\}$ makes the problem uninteresting. Such is the case for the "heavy spender" example from Kohavi and Parekh [2003]. With legitimacy defined as (3) (or as (4) when $\tau = 0$) a model may be built that uses the purchase of a diamond to conclude that the customer is a big spender but with τ sufficiently large this is not allowed. This transforms the problem from identification of "heavy spenders" to the suggested identification of "migrators".

$$t\{\chi\} < t\{y\} - \tau \Leftrightarrow \chi \in legit\{y\}. \tag{4}$$

Another example, using the same predictive modeling notation, is a memory limitation, where a model may not use information *older* than a time relative to that of the target:

$$t\{y\} - \tau < t\{\chi\} < t\{y\} \Leftrightarrow \chi \in legit\{y\}. \tag{5}$$

Beyond predictive modeling, we can think of a requirement to use exactly m features from the universe of possible features $\mathcal{X}$:

$$legit\{y\} = \{\{\chi_1, \ldots, \chi_m\} | \forall k : \chi_k \text{ unique component of } \mathcal{X}\}, \tag{6}$$

and so on. In fact, there is a variant of example (6) which is very common: only the features $\mathcal{X}_p$ selected for a specific provided dataset are considered legitimate. Sometimes this rule allows free use of the entire set:

$$legit\{y\} = \mathcal{P}(\mathcal{X}_p), \tag{7}$$

where $\mathcal{P}(A)$ stands for the power set of A.

Usually, however, this rule is combined with (3) to give:

$$legit\{y\} = \left\{S | S \in \mathcal{P}(\mathcal{X}_p), \forall \chi \in S : t\{\chi\} < t\{y\}\right\}. \tag{8}$$

Most documented cases of leakage mentioned in Section 2 are covered by condition (2) in conjunction with a no-time-machine requirement as in (3). For instance, in another view of the trivial example of predicting rainy days, the target is an illegitimate feature since its value is not observable to the client when the prediction is required (say, the previous day). As another example, the pneumonia detection database in the INFORMS 2008 challenge discussed in Perlich et al. [2008] and Rosset et al. [2010] implies that a certain combination of missing diagnosis code and some other features is highly informative of the target. However, this feature is illegitimate, as the patient's condition is still being studied when prediction is required.

It is easy to see how conditions (2) and (3) similarly apply to the account number and interviewer name examples from Pyle [2003], the session length of Kohavi et al. [2000] (while the corrected "page number in session" is fine), the immediate and indirect triggers described in Kohavi and Parekh [2003] and Kohavi et al. [2004], the remaining competitions described in Perlich et al. [2008] and Rosset et al. [2010], and the Website-based features used by IBM and discussed in Rosset et al. [2010]. However, not all examples fall under condition (2).

Let us examine the case mentioned earlier of KDD Cup 2007 as discussed in Rosset et al. [2007], in which predictions were required at the end of 2005. While clearly taking advantage of leaks (the mere fact of using data from 2006 is proof, but we can also see it in action by the presence of measurable leakage; the fact that this model performed significantly better than a similar model built only on given data, both in internal tests and the final competition), the final delivered model $\mathbb{M}$ does not include any illegitimate feature.[2] To understand what has transpired, we must address the issue of leakage in training examples.

3.3. Leakage in Training Examples

Let us first consider the following synthetic but illustrative example. Suppose we are trying to predict the level of a white noise process y_t for $t \in \{101, 102, \ldots, 200\}$, clearly a hopeless task. Suppose further that for the purpose of predicting y_t, t itself is a legitimate feature but otherwise, as in (3), only past information is deemed legitimate. Now consider a model trained on examples $\mathbf{W_{tr}}$ taken from $t \in \{1, 2, \ldots, 200\}$. The proposed model is $\hat{y}_t = \mathbb{M}(t, \mathbf{W_{tr}})$, a table containing for each t the target's realized value y_t. Strictly speaking, the only feature used by this model, t, is legitimate. Hence the model has no leakage as defined by condition (2), however, it clearly has perfect prediction performance for the evaluation set in the example. We would naturally like to capture this case under a complete definition of leakage for this problem.

In order to tackle this case, we suggest adding to (2) the following condition for the absence of leakage: For all $\mathbf{y} \in \mathbf{Y_{ev}}$,

$$\forall X \in \mathbf{X_{tr}}, X \in legit\{y\} \wedge \forall \tilde{y} \in \mathbf{Y_{tr}}, \tilde{y} \in legit\{y\}, \tag{9}$$

where $\mathbf{Y_{ev}}$ is the set of *evaluation*[3] target elements, and $\mathbf{Y_{tr}}, \mathbf{X_{tr}}$ are the sets of training targets and feature vectors, respectively, whose realizations make up the set of training examples $\mathbf{W_{tr}}$.

One way of interpreting this condition is to think of the information presented for training as constant features embedded into the model, and added to every feature-vector element the model is called to generate a prediction for. This reflects the fact that leakage is not just a property of features, but can be a property of training examples.

For modeling problems where the usual "i.i.d. elements" assumption is valid, and when without loss of generality considering all information specific to the element being predicted as features rather than examples, condition (9) simply reduces to condition (2) since irrelevant observations can always be considered legitimate. In contrast, when dealing with problems exhibiting nonstationarity, otherwise known as concept drift [Widmer and Kubat 1996], and more specifically the case when samples of the target are not mutually independent, condition (9) cannot be reduced to condition (2). Such is the case of KDD Cup 2007. Information about the number of ratings given to the first group of movies is not statistically independent of the number of ratings given to the second group of movies. The former was provided with the "Who Rated What" task, while the latter is the target in the "How Many Ratings" task. The reason for this dependence is that both movie groups receive ratings given by the same population of users over the same period in 2006. Ratings are thus mutually affected by shared causal ancestors such as viewing and participation trends (e.g., promotions, similar media or event that gets a lot of exposure, and so on). Without proper conditioning on these shared ancestors we have potential dependence, and because most of these ancestors are unobservable, and difficult to find observable proxies for, controlling for them is unrealistic.

3.4. Discussion

It is worth noting that leakage in training examples is not limited to the explicit use of illegitimate examples in the training process. A more dangerous way in which

[2]In fact the use of external sources that are not rolled-back to 2005, such as using IMDB data current to the time of modeling (2007), is simple leakage just like, say, in the IBM example. However, this is not the major source of leakage in this example.

[3]We use the term evaluation as it could play the classic role of either validation or testing.

illegitimate examples may creep in and introduce leakage is through *design decisions*. Suppose, for example, that we have access to illegitimate data about the deployment population, but there is no evidence in training data to support this knowledge. This might prompt us to use a certain modeling approach that otherwise contains no leakage in training examples but is still illegitimate. Examples could be:

—selecting or designing features that will have predictive power in deployment, but do not show this power on training examples,
—algorithm or parametric model selection, and
—tuning-parameter value choices.

This form of leakage is perhaps the most dangerous as an evaluator may not be able to identify it even when she knows what she is looking for. The exact same design could have been brought on by theoretic rationale, in which case it would have been completely legitimate. In some domains such as time-series prediction, where typically only a single history measuring the phenomenon of interest is available for analysis, this form of leakage is endemic and commonly known as data snooping/dredging [Lo and MacKinlay 1990] (a situation where researchers do not form a hypothesis in advance of examining data).

Another important point is that our notion of leakage is intimately linked to the idea of making predictions with respect to a particular target. In that narrow sense, there is no room for leakage in unsupervised learning where there is no notion of a target (e.g., as defined in Hastie et al. [2009]). However, one has to carefully consider the purpose of unsupervised efforts. Often, the results are being utilized in a later stage supervised task and at that point leakage becomes very relevant and even harder to detect. So in some unsupervised learning scenarios the same issues we discuss here are also relevant. For example, suppose patient mammography data has “malignant/benign” labels but those are simply not specified in the data, and we resort to clustering to find some proxy of the lost labels. Of course if illegitimate features such as the target itself were to creep into the data (e.g., they are in fact present but not marked as such) we would have leaks in our model.

Regarding concretization of legitimacy for a new problem: Arguably, more often than not the modeler might find it very challenging to define, together with the client, a complete set of such legitimacy guidelines prior to any modeling work being undertaken, and specifically prior to performing preliminary evaluation. Nevertheless it should usually be rather easy to provide a coarse definition of legitimacy for the problem, and a good place to start is to consider model use cases. The specification of any modeling problem is really incomplete without laying out these ground rules of what constitutes a legitimate model.

As a final point on legitimacy, let us mention that once it has been clearly defined for a problem, the major challenge becomes preparing the data in such a way that ensures models built on this data would be leakage free. Alternatively, when we do not have full control over data collection or when they are simply given to us, a methodology for detecting when a large number of seemingly innocent pieces of information are in fact plagued with leakage is required. This shall be the focus of the following two sections.

4. AVOIDANCE

4.1. Methodology

Our suggested methodology for avoiding leakage is a two-stage process of tagging every observation with *legitimacy tags* during collection and then observing what we call a *learn-predict separation*. We shall now describe these stages and then provide some examples.

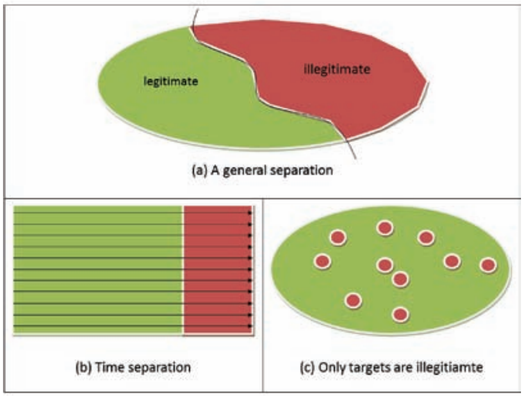

Fig. 4. An illustration of learn-predict separation.

At the most basic level suitable for handling the more general case of leakage in training examples, legitimacy tags (or hints) are ancillary data attached to every pair $(\mathbf{x}, \mathbf{y})$ of observational input instance $\mathbf{x}$ and target instance $\mathbf{y}$, sufficient for answering the question "is $\mathbf{x}$ legitimate for inferring $\mathbf{y}$?" under the problem's definition of legitimacy. With this tagged version of the database it is possible, for every example being studied, to roll back the state of the world to a legitimate decision state, eliminating any confusion that may arise from only considering the original raw data.

In the learn-predict separation paradigm (illustrated in Figure 4) the modeler uses the raw but tagged data to construct training examples in such a way that: (i) for each target instance, only those observational inputs which are purely legitimate for predicting it are included as features, and (ii) only observational inputs which are purely legitimate with all evaluation targets may serve as examples. This way, by construction, we directly take care of the two types of leakage that make up our formulation, respectively leakage in features (2) and in training examples (9). To completely prevent leakage by design decisions, the modeler has to be careful not to even get exposed to information beyond the separation point, for this we can only prescribe self-control.

As an example, in the common no-time-machine case where legitimacy is defined by (3), legitimacy tags are timestamps with sufficient precision. Legitimacy tagging is implemented by timestamping every observation. Learn-predict separation is implemented by a cut at some point in time that segments training from evaluation examples (see Figure 5). This is what has been coined in Rosset et al. [2010] as *prediction about the future*. Interestingly enough, this common case does not sit well with the equally common way databases are organized. Updates to database records are usually not timestamped and not stored separately, and at best whole records end up with one timestamp. Records are then translated into examples, and this loss of information is often the source of all evil that allows leakage to find its way into predictive models.

The original data for the INFORMS 2008 Data Mining Challenge lacked proper timestamping, causing observations taken before and after the target's timestamp to end up as components of examples. This made time separation impossible, and models built on this data did not perform prediction about the future. On the other hand, the data for KDD Cup 2007's "How Many Ratings" task in itself was (as far as we are aware) well timestamped and separated. Training data provided to competitors was sampled prior to 2006, while test data was sampled after and including 2006, and was not given. The fact that training data exposed by the organizers for the separate "Who

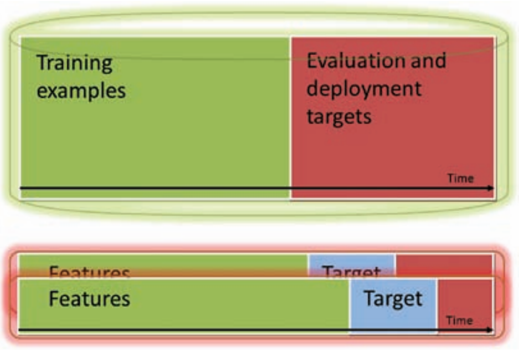

Fig. 5. For the predictive modeling case, where there is a time-line along which all observations may be serialized, separation is implemented by a cut at some point in time that segments training from evaluation examples, and features from their targets.

Rated What" task contained leakage was due to an external source of leakage, an issue related with data mining competitions that we shall discuss next.

It is worth noting that legitimacy tags are a special case of data provenance [Buneman et al. 2001]. Provenance describes where the data came from and the process by which they arrived at a database, and is essential to disambiguate the data and enable reuse. Here we are focused only on the ability to disambiguate the data at a level which enables avoiding leaks, but questions of quality or accuracy could be just as important in many domains.

4.2. External Leakage in Competitions

Our account of leakage avoidance, especially in light of our recurring references to data mining competitions in this article, would be incomplete without mentioning the case of *external leakage*. This happens when some data source other than what is simply given by the client (organizer) for the purpose of performing inference, contains leakage and is accessible to modelers (competitors). Examples for this kind of leakage include the KDD Cup 2007 "How Many Ratings" task, the INFORMS 2010 financial forecasting challenge, and the IJCNN 2011 Social Network Challenge.[4]

In these cases, it would seem that even a perfect application of the suggested avoidance methodology breaks down by considering the additional source of data. Indeed, separation only prevents leakage from the data actually separated. The fact that other data are even considered is indeed a competition issue, or in some cases an issue of a project organized like a competition (i.e., projects within large organizations, outsourcing or government issued projects). Sometimes this issue stems from a lack of an auditing process for submissions, however, most of the time, it is introduced to the playground on purpose.

Competition organizers, and some project clients, have an ulterior conflict of interest. On the one hand they do not want competitors to cheat and use illegitimate data. On the other hand they would welcome insightful competitors suggesting new ideas for sources of information. This is a common situation, but the two desires or tasks are often conflicting: when one admits not knowing which sources could be used, one also admits she can't provide an air-tight definition of what she accepts as legitimate. She may be able to say something about legitimacy in her problem, but would intentionally leave room for competitors to maneuver.

[4]Although it is entirely possible that internal leakage was also present in these cases (e.g., forum discussions regarding the IJCNN 2011 competition on http://www.kaggle.com).

The solution to this conflict is to separate the task of suggesting broader legitimacy definitions for a problem from the modeling task that fixes the current understanding of legitimacy. Competitions should just choose one task, or have two separate challenges: one to suggest better data, and one to predict with the given data only. The two tasks require different approaches to competition organization, a thorough account of which is beyond the scope of this article. One approach for the first task that we will mention is *live prediction*.

When the legitimacy definition for a data mining problem is isomorphic to the no-time-machine legitimacy definition (3) of predictive modeling, we can sometimes take advantage of the fact that a learn-predict separation over time is physically impossible to circumvent. We can then ask competitors to literally predict targets in the future (that is, a time after submission date) with whatever sources of data they think might be relevant, and they will not be able to cheat in this respect. For instance, the IJCNN Social Network Challenge could have asked to predict new edges in the network graph a month in advance, instead of synthetically removing edges from an existing network that left traces and the online original source for competitors to find.

5. DETECTION

Often the modeler has no control over the data collection process. When the data are not properly tagged, the modeler cannot pursue a learn-predict separation as in the previous section. One important question is how to detect leakage when it happens in given data, as the ability to detect that there is a problem can help mitigate its effects. In the context of our formulation from Section 3, detecting leakage boils down to pointing out how conditions (2) or (9) fail to hold for the dataset in question. A brute-force solution to this task is often infeasible because datasets will always be too large. We propose the following methods for filtering leakage candidates.

Exploratory Data Analysis (EDA) can be a powerful tool for identifying leakage. EDA [Tukey 1977] is the good practice of getting more intimate with the raw data, examining it through basic and interpretable visualization or statistical tools. Prejudice free and methodological, this kind of examination can expose leakage as patterns in the data that are surprising. In the INFORMS 2008 breast cancer example, for instance, the fact that the "patient id" is so strongly correlated with the target is surprising, if we expect ids to be given with little or no knowledge of the patient's diagnosis, for instance, on an arrival time basis. Of course some surprising facts revealed by the data through basic analysis could be legitimate, for the same breast cancer example it might be the case that family doctors direct their patients to specific diagnosis paths (which issue patient IDs) based on their initial diagnosis, which is a legitimate piece of information. Generally, however, as most worthy problems are highly nontrivial, it is reasonable that only few surprising candidates would require closer examination to validate their legitimacy. On the very practical side, a good starting point for EDA is to look for any form of unexpected data properties. Common giveaways are found in identifiers, matching (or inconsistent matching) of identifiers (i.e., sample selection biases), surprises in distributions (spikes in densities of continuous values), and finally suspicious order in supposedly random data.

Initial EDA is not the only stage of modeling where surprising behavior can expose leakage. The "IBM Websphere" example discussed in Section 1 is an excellent example that shows how the surprising behavior of a feature in the fitted model, in this case a rare value (the word "Websphere"), becomes apparent only after the model has been built. Another approach related to critical examination of modeling results comes from observing overall surprising model performance. In many cases we can come to expect, from our own experience or from prior/competing documented results, a certain level of performance for the problem at hand. A substantial divergence from this expected performance is surprising and merits testing the most informative observations that the model is based on more closely for legitimacy. The results of many participants in the INFORMS 2010 financial forecasting challenge are an example of this case because they contradict prior evidence about the efficiency of the stock market.

Recognizing the issues of leakage in competitions, Kaggle recently deployed a similar semiautomated EDA approach that, prior to the beginning of a data mining contest, builds a model using random forests and estimates variable importance across all features to create a list of the n most predictive variables. This list is submitted to a domain expert for legitimacy assessment. This approach can be viewed as a hybrid solution where legitimacy hints are not initially required and only validated upon request.

Finally, perhaps the best approach, but possibly also the one most expensive to implement, is early in-the-field testing of initial models. Any substantial leakage would be reflected as a difference between estimated and realized out-of-sample performance. However, this is in fact a sanity check of the model's generalization capability, and while this would work well for many cases, other issues can make it challenging or even impossible to isolate the cause of such performance discrepancy as leakage: classical overfitting, tangible concept drift, issues with the design of the field-test such as sampling bias, and so on.

A fundamental problem with the methods for leakage detection suggested in this section is that they all require some degree of domain knowledge: For EDA one needs to know if a good predictor is reasonable; comparison of model performance to alternative models or prior state-of-art models requires knowledge of the previous results; and the setup for early in-the-field evaluation is obviously very involved. The fact that these methods still rely on domain knowledge places an emphasis on leakage avoidance during data collection, where we have more control over the data.

6. (NOT) FIXING LEAKAGE

Once we have detected leakage, what should we do about it? In the best-case scenario, one might be able to take a step back, get access to raw data with intact legitimacy tags, and use a learn-predict separation to reconstruct a leakage-free version of the problem. The second-best scenario happens when intact data is not available but the modeler can afford to fix the data collection process and postpone the project until leakage-free data become available. In the final scenario, one just has to make do with that which is available.

Because of structural constraints at work, leakage can be somewhat localized in samples. This is true in both INFORMS 2008 and INFORMS 2009 competitions mentioned before, and also in the "IBM Websphere" example. When the model is used in the field, by definition all observations are legitimate and there can be no active leaks. So to the extent that most training examples are also leakage free, the model may perform worse in deployment than in the pilot evaluation, but would still be better than random guessing and possibly competitive with models built with no leakage. This is good news as it means that, for some problems, living with leakage without attempting to fix it could work.

What happens when we do try to fix leakage? Without explicit legitimacy tags in the data, it is often impossible to figure out the legitimacy of specific observations and/or features even if it is obvious that leakage has occurred. It may be possible to partly plug the leak but not to seal it completely, and it is not uncommon that an attempt to fix leakage only makes it worse.

Usually, where there is one leaking feature, there are more. Removing the "obvious" leaks that are detected may exacerbate the effect of undetected ones. In the e-commerce example from Kohavi et al. [2004], one might envision to simply remove the obvious "free shipping" field, however, this kind of *feature removal* succeeds only in very few and simple scenarios to completely eradicate leaks. In particular, in this example one is still left with the “no purchase in any department” signature. Another example for this is the KDD Cup 2008 breast cancer prediction competition, where the patient ID contained an obvious leak. It is by no means obvious that removing this feature would leave a leakage-free dataset, however. Assuming different ID ranges correspond to different health care facilities (in different geographical locations, with different equipment), there may be additional traces of this in the data. If, for instance, the imaging equipment’s grey scale is slightly different and in particular grey levels are higher in the location with high cancer rate, the model without ID could pick up this leaking signal from the remaining data, and the performance estimate would still be optimistic (the winners show evidence of this in their report [Perlich et al. 2008]).

Similar arguments can be made about *feature modification* performed in INFORMS 2008 in an attempt to plug obvious leaks, which clearly created others; and instance removal in organization of INFORMS 2009, which also left some unintended traces [Xie and Coggeshall 2010].

In summary, further research into general methodology for leakage correction is indeed required. Lacking such methodology, our experience is that fully fixing leakage without learn-predict separation is typically very hard, and perhaps impossible.

7. LEAKAGE AND CAUSALITY

In this section we take a look at how research in causality, and the counterfactual framework, allow us to gain insights about leakage. Directed Acyclic Graphs (DAGs) are commonly used to represent causal, or counterfactual, relationships between variables [Pearl 1995; Robins 1997, 2009]. Rather than attempting to explain various instances of leakage in terms of causality we will look at a couple of general leakage situations and explain how phenomena commonly understood and expressed through DAGs can aid our understanding of certain general leakage problems.

First, it should be intuitively clear that causality and leakage might be connected. In the majority of our examples, the leak changed the role of cause and effect. The “free shipping” was an effect of the “big spending”. Even if the majority of useful predictive models are not causal, this stream of literature can help us to address leakage, because it forces the modeler to be more specific on legitimate dependencies.

Before explaining the utility we can derive from DAGs as a representation of a causal structure, let us define them. Each graph is represented as a set of nodes and edges. Each node represents a random variable and each edge represents a relationship between a pair of nodes. In DAGs these edges have direction indicating that a particular node is a cause of a subsequent node. Figure 6 presents an example of such a DAG. A path in the graph is a sequence of edges. Thus, the sequence of edges (V, Z), (Z, X), (X, Y) is a path in the DAG depicted in Figure 6. A directed path is a path where each arrow points from one node to the subsequent node in the path. DAGs in particular contain no loops in their paths, that is, the directed paths in the graph do not cycle back into the same nodes. These graphs are called acyclic and are the types of graphs we consider here. Within acyclic graphs we can define nodes that precede a particular node as ancestors of the node and nodes that come after, or are caused by the node, are defined as descendants. This genealogy type terminology is extended to define direct descendants as children and direct ancestors as parents. For example, Y and Z are parents of X and V; Z, and Y are the ancestors of X. Likewise X is a child of Z and Y.

Typically these DAGs are generated based off of temporal cues and subject matter knowledge of experts in the field. We recognize that many times in competitions contestants are not privy to the subject matter knowledge needed to generate a useful graph. However, the temporal cues in many cases are available and many times these cues may be used to generate graphs and thereby enforcing temporal notions of legitimacy.

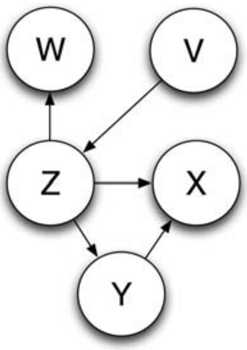

Fig. 6. Example of a DAG.

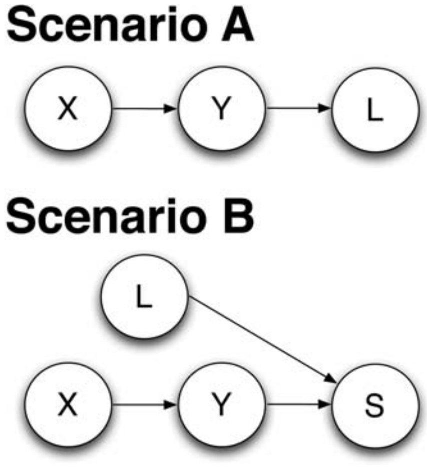

Fig. 7. Leakage scenarios expressed through DAGs.

The two graphs displayed in Figure 7 represent two scenarios that may be used to describe leakage. In each graph X represents a set of ancestors of the target variable Y and L represents a set of nodes or variables that may introduce leakage. S in scenario B is a sampling variable that will be explained in detail shortly. For simplicity we will view each node in these graphs as a single random variable; however, in practice they can represent a set of variables.

Scenario A describes the typical temporal problem that has already been described in the leakage literature and is just recast here in terms of graph vocabulary for completeness. Thus, $legit\{Y\}$ may be expressed according to the graph as the set of all ancestors of Y. So using Y itself or any of Y's descendants in predicting Y is a source of leakage. Thus Y and its descendants are not part of the set $legit\{Y\}$. This is like the example earlier where the "account number" had predictive power because it is caused by the observations setting up a bank account, the target variable of interest. It can be said that "account number" is a descendent of opening an account at a bank and thus it is not part of $legit$\{Opening A Bank Account\}.

The second graph, B, represents a more interesting form of leakage that is more difficult to discern absent the use of a graph. This is a form of selection bias that is commonly referred to as "stratifying on a collider" bias in the epidemiology and causal community. It has also been explored as a type of selection bias in Smith and Elkan

[2007] and was referred to as "arbitrary bias". A collider is a common descendent of two nodes in a DAG. This is represented by S in scenario 2. It is well-documented that stratifying on a collider causes bias and ultimately leads to unreliable models and conclusions. A nice example of this bias is introduced in Pearl [2009, page 17].

Many times temporal cues are not enough to construct a DAG. For example, in KDD Cup 2008 which dealt with breast cancer detection, temporal cues alone would suggest that ID could be used as a predictor of breast cancer because it is assigned before diagnosis. Using a variable such as the ID in the breast study a practitioner may assume they are adjusting for a variable that preceded the outcome variable in time and thus should have no problem with leakage. At worst they would assume that variable would have no predictive value and have no harmful affect on future predictions. One could reasonably assume a DAG similar to Scenario A where ID is an ancestor of Y. This is because we know ID generation preceded the diagnosis of the breast cancer. So it is obvious that temporal cues alone are not enough to distinguish what is in $legit\{Y\}$ from what is not because we observed the ID variable leaking information about the outcome.

An alternative DAG for the system may be like that presented in scenario B where ID is in fact not an ancestor of Y. That is, ID has no causal effect on Y. Rather, the sick individuals may have been sampled for a certain ID range and the nonsick from a different ID range. This may be because the positives were sampled from a much smaller range of IDs because finding individuals that do not have cancer is easy and takes a much smaller range of IDs while finding enough negatives, or those people with cancer, requires sampling a much larger range of IDs. S in this situation is a binary variable that indicates if an observation was sampled into the dataset or not. In this case the sampling method itself created the collider. That is, we can think of the binary variable to be sampled or not to be sampled for the final dataset as the common descendent of both having breast cancer, Y, and the patient's ID. Since the determination to sample for the contest dataset was caused by both Y and ID the binary variable of being included in the dataset or not is a collider. Once the sampling is actually done and only those with S equal to 1 are chosen for the dataset, the data are now stratified by the collider, or common descendent, and a relationship is induced between the ID and the outcome, Y. However, this relationship is a byproduct of the artificial sampling procedure and not a real relationship in the greater population. As a result the ID is highly predictive in the data as sampled but has no real-world predictive value. This is the result of the way that the data was preprocessed for the contest that induced a relationship between ID and Y which the machine learning algorithms were able to pick up on. This induced relationship is leakage and in this form is known as the bias induced by stratifying on a collider. That is, by stratifying on a common descendent, S, a relationship is induced between ID and the outcome Y. Had there been no stratification on S no relationship would exist between ID and Y. From a conceptual level, S is a leak since it is an descendent of Y (and thereby illegitimate) even if it never directly appears in the dataset but only assisted in the process of assembling it. The remnants of this leak manifest itself in the collider.

Unfortunately, "accidental" sampling on a collider is probably one of the most common issues in the preparation of competitions as well as project datasets. This typically happens when either one class variable is not entirely well-defined ("everybody else") or exceedingly rare. In those cases, the data for positives are generated in a different process than negatives and a vast collection of variables show spurious signal.

8. CONCLUSION

It should be clear at this point that modeling with leakage is undesirable on many levels: it is a source for poor generalization and overestimation of expected performance.

A rich set of examples from diverse data mining domains given throughout this article add to our own experience to suggest that in the absence of methodology for handling it, leakage could be the cause of many failures of data mining applications.

In this article we have described leakage as an abstract property of the relationship of observational inputs and target instances, and showed how it could be made concrete for various problems. In light of this formulation an approach for preventing leakage during data collection was presented that adds legitimacy tags to each observation. Also suggested were three ways for zooming in on potentially leaking features: EDA, ex-post analysis of modeling results, and early field-testing. Finally, problems with fixing leakage have been discussed as an area where further research is required.

Causal modeling provides an alternative way of thinking about leakage and enforces a methodology prior to estimating a model that will prevent leakage, to the extent that the structure of the designed causal graph is correct. While there is recent work on learning the structure of the causal graph from data automatically, it is recognized as a very hard problem and the methods are far from fully developed. The act of designing the causal graph is still a dominantly human effort and is analogous to annotating legitimacy cues on the level of variables. Causal analysis also hints at sampling as a major source of leakage problems.

Another source of leakage is when in selecting or constructing the target variable from an existing dataset, the modeler neglects to consider the legitimacy definition imposed by this selection, which makes other related variables illegitimate (e.g., large purchases versus free shipping). In other cases, the modeler is well aware of the implications of his selection, but falters when facing the trade-off between removing potentially important predictive information and ensuring no leakage. Most instances of internal leakage in competitions were in fact of this nature and have been created by the organizers despite best attempts to avoid it.

We hope that the notable collection of case studies and suggested methodology described in this article can help to raise awareness to this important issue in data mining, provide initial guidelines that may save projects and competitions from falling in the leakage trap, and ultimately encourage models and modeling approaches that are relevant in their application domains.

REFERENCES

Buneman, P., Khanna, S., and Wang-Chiew, T. 2001. Why and where: A characterization of data provenance. In *Proceedings of the International Conference on Database Theory (ICDT'01)*. 316–330.

Engle, R. and Granger, C. 1987. Co-Integration and error correction: Representation, estimation and testing. *Econometrica 55*, 2, 251–276.

Hastie, T., Tibshirani, R., and Friedman, J. H. 2009. *The Elements of Statistical Learning: Data Mining, Inference and Prediction*. Springer.

Kohavi, R., Brodley, C., Frasca, B., Mason, L., and Zheng, Z. 2000. Kdd-Cup 2000 organizers' report: Peeling the onion. *ACM SIGKDD Explor. Newslett. 2*.

Kohavi, R., Mason, L., Parekh, R., and Zheng, Z. 2004. Lessons and challenges from mining retail e-commerce data. *Mach. Learn. 1,* 2.

Kohavi, R. and Parekh, R. 2003. Ten supplementary analyses to improve e-commerce web sites. In *Proceedings of the 5^{th} WEBKDD Workshop.*

Laiy, S., Xiang, L., Diao, R., Liu, Y., Gu, H., Xu, L., Li, H., Wang, D., Liu, K., Zhao, J., et al. 2011. Hybrid recommendation models for binary user preference prediction problem. http://jmlr.csail.mit.edu/proceedings/papers/v18/lai12a/lai12a.pdf.

Lo, A. and MacKinlay, A. 1990. Data-Snooping biases in tests of financial asset pricing models. *Rev. Finan. Stud. 1,* 431–467.

Narayanan, A., Shi, E., and Rubinstein, B. 2011. Link prediction by deanonymization: How we won the kaggle social network challenge. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN'11).*

NISBET, R., ELDER, J., AND MINER, G. 2009. *Handbook of Statistical Analysis and Data Mining Applications.* Academic Press.

PEARL, J. 1995. Causal diagrams for empirical research. *Biometrika 82*, 4, 669–688.

PEARL, J. 2009. *Causality: Models, Reasoning and Inference* 2$^{\text{nd}}$ Ed. Cambridge University Press, Cambridge, UK.

PERLICH, C., MELVILLE, P., LIU, Y., SWIRSZCZ, G., LAWRENCE, R., AND ROSSET, S. 2008. Breast cancer identification: KDD cup winner's report. *ACM SIGKDD Explor. Newslett. 2,* 39–42.

PYLE, D. 1999. *Data Preparation for Data Mining*. Morgan Kaufmann Publishers.

PYLE, D. 2003. *Business Modeling and Data Mining.* Morgan Kaufmann Publishers.

PYLE, D. 2009. *Data Mining: Know it All*. Morgan Kaufmann Publishers, Chapter 9.

ROBINS, J. 1997. Causal inference from complex longitudinal data. *Lat. Variab. Model. Appl. Causal. 120,* 69–117.

ROSSET, S., PERLICH, C., AND LIU, Y. 2007. Making the most of your data: KDD cup 2007 "how many ratings" winner's report. *ACM SIGKDD Explor. Newslett. 2*.

ROSSET, S., PERLICH, C., SWIRSZCZ, G., LIU, Y., AND MELVILLE, P. 2010. Medical data mining: Lessons from winning two competitions. *Data Min. Knowl. Discov. 3*, 439–468.

SMITH, A. AND ELKAN, C. 2007. Making generative classifiers robust to selection bias. In *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM Press, New York, 657–666.

TUKEY, J. 1977. *Exploratory Data Analysis*. Addison-Wesley.

WIDMER, G. AND KUBAT, M. 1996. Learning in the presence of concept drift and hidden contexts. *Mach. Learn. 1.*

XIE, J. AND COGGESHALL, S. 2010. Prediction of transfers to tertiary care and hospital mortality: A gradient boosting decision tree approach. *Stastist. Anal. Data Min. 3*, 4, 253-258.

Received November 2011; revised March 2012; accepted May 2012