Leakage in Data Mining: Formulation, Detection, and Avoidance

Shachar Kaufman
School of Electrical Engineering
Tel-Aviv University
69978 Tel-Aviv, Israel
shachark@post.tau.ac.il

Saharon Rosset
School of Mathematical Sciences
Tel-Aviv University
69978 Tel-Aviv, Israel
saharon@post.tau.ac.il

Claudia Perlich
Media6Degrees
37 East 18th Street, 9th floor
New York, NY 10003
claudia@media6degrees.com

ABSTRACT

Deemed "one of the top ten data mining mistakes", leakage is essentially the introduction of information about the data mining target, which 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 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.

Categories and Subject Descriptors

H.2.8 [Database Management]: Database Applications – *Data mining*. I.5.2 [Pattern Recognition]: Design Methodology – *Classifier design and evaluation*.

General Terms

Theory, Algorithms.

Keywords

Data mining, Leakage, Statistical inference, Predictive modeling.

1. INTRODUCTION

Deemed "one of the top ten data mining mistakes" [7], leakage in data mining (henceforth, *leakage*) is essentially the introduction of information about the target of a data mining problem, which 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 this illegitimate information is unintentional, and facilitated by the data collection, aggregation and preparation process. It is usually subtle and indirect, making it very 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 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, which we survey in Section 2, mentions leakage and acknowledges its importance and prevalence in both synthetic competitions and real-life data mining projects [e.g. 2, 7]. 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 inference problems. This gap in theory and methodology could be the reason that several major data mining competitions held recently such as KDD-Cup 2008, or the INFORMS 2010 Data Mining Challenge, though judiciously organized by capable individuals, suffered from severe leakage. In many cases, attempts to fix leakage resulted in the introduction of new leakage which is even 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 which 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 also in Section 2. We revisit them in later sections to provide a more concrete setting for our discussion.

The major contribution of this paper, that is, 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 we have limited or no

KDD'11, August 21 – 24, 2011, San Diego, California, USA.
Copyright 2011 ACM 978-1-4503-0813-7/11/08…$10.00.

control over how the data have been collected. This methodology should be particularly useful for practitioners in predictive modeling problems, as well as for prospective competition organizers.

2. LEAKAGE IN THE KDD LITERATURE

The subject of leakage has been visited by several data mining textbooks as well as a few papers. Most of the papers we refer to are related to KDD-Cup competitions, probably due to authors of works outside of competitions locating and fixing leakage issues without reporting the process. We shall give a short chronological review here while collecting examples to be used later as case studies for our proposed definition of leakage.

Pyle [9, 10, 11] refers to the phenomenon which we call here leakage, in the context of predictive modeling, 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, which we will also discuss in Section 5. Nisbet *et al.* [7] 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 and prevent leakage. These titles provide a handful of elementary but common examples of leakage. Two representative ones are: (i) 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. (ii) 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.* [2] 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 website data analytics where for each page viewed the prediction 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 website. 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.* [3] 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 paper by the authors [4] 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.* [11] discuss leakage encountered in the 2007 KDD-Cup competition. In that year's contest there were two related challenges concerning movie viewers' reviews from the famous Netflix database. The first challenge, "Who Reviewed What", was to predict whether each user would give a review for each title in 2006, given data up to 2005. The second challenge, "How Many Reviews", was to predict the number of reviews each title would receive in 2006, also using data given up to 2005. For the first challenge, a test set with actual reviews from 2006 was provided. Although disjoint sets of titles were used to construct the data sets for these two challenges, Rosset *et al.*'s winning submission managed to use the test set for the first problem as the target in a supervised-learning modeling approach for the second problem. This was possible due to a combination of two facts. First, up to a scaling factor and noise, the expected number of user/review pairs in the first problem's test set in which a title appears is equal to the total number of reviews which that titled received in 2006. This is exactly the target for the second problem, only on different titles. Second, the titles are similar enough to share statistical properties, so from the available dynamics for the first group of titles one can infer the dynamics of the second group's. 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 [7, 13]. 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 (ignored by most competitors) has tremendous and unexpected predictive power. They hypothesize that multiple clinical study, institution or equipment sources were used 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. 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.* [13], 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 websites. This turned out to be leakage since the website content used for training had been sampled at the point in time where the potential customer has already become a customer, and where the website contained traces of the IBM products purchased, such as the word "Websphere" (*e.g.* in a press release about the purchase or a specific product feature the client uses).

The latest 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. The data were segmented to a training database, on which participants were expected to build their predictive models, and a test database which was used by 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've 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 cointegrated with the target (*e.g.* a second stock within the same sector as the target stock), and 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) was the true driver of success for these models. 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 [3] they had been able to recognize, through sophisticated analysis, that the social network in question was Flickr and then to de-anonymize the majority of the data. This allowed them to use edges available from the online Flickr network to correctly predict over 60% of edges which 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.

Clearly, then, the issue of leakage has been observed in various contexts and problem domains, with a natural focus on predictive modeling. 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 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 statistical inference 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 [1]. Let us just lay out our notation and say that in our framework we receive from an axiomatic data preparation stage a multivariate random *process* $\mathcal{W} = (\mathcal{X}, \mathcal{Y})$. $\mathcal{Y}$ is the outcome or target generating process with samples y target *instances*. Values or *realizations* of the random variable y are denoted $\boldsymbol{y}$ (in bold). Similarly, $\mathcal{X}$, X and $\boldsymbol{X}$ are the feature-vector generating process, an instance and realization. For individual feature generating processes, instances and realizations we use $x \in \mathcal{X}$, $x \in X$ and $\boldsymbol{x} \in \boldsymbol{X}$. Specific instances x_0 and y_0 taken from the same instance of $\mathcal{W}$ are said to be $\mathcal{W}$-related. The modeler's goal is to statistically infer a target instance, from its associated feature-vector instance in $\mathcal{W}$ and from a separate group of samples of $\mathcal{W}$, called the training examples $\boldsymbol{W_{tr}}$. The solution to this problem is a model $\hat{y} = \mathbb{M}(X, \boldsymbol{W_{tr}})$. We say that the model's *observational inputs* for predicting y are X and $\boldsymbol{W_{tr}}$, and this relation between the various elements in the framework is the base for our discussion.

Models containing leaks are a subclass of the broader concept of illegitimate or unacceptable models. At this level, l*egitimacy*, 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.

However our focus here is on leakage, which is a specific form of illegitimacy that is 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 random variable. We say a second random variable v is u-legitimate if v is observable to the client for the purpose of inferring u. In this case we write $v \in legit\{u\}$.

A fully concrete meaning of 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 the target itself must never be used for inference:

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

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 instance y if one or more of its observational inputs are y-illegitimate. We say that the model *inherits* the illegitimacy property from the features and training examples it uses. The discussion proceeds along these two possible sources of leakage for a model: 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 legitimacy to the case of random processes: Let $\mathcal{u}$ be some random process. We say a second random process $\mathcal{v}$ is $\mathcal{u}$-legitimate if, for every pair of instances of $\mathcal{u}$ and $\mathcal{v}$, u and v respectively, which are $\mathcal{W}$-related, v is u-legitimate. We use the same notation as we did for random variables in 3.1, and write that $\mathcal{v} \in legit\{\mathcal{u}\}$.

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

$$\forall x \in \mathcal{X}, x \in legit\{\mathcal{Y}\}. \tag{2}$$

That is, any feature made available by the data preparation process is deemed legitimate by the precise formulation of the modeling problem at hand, instance by instance w.r.t. 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 $\mathcal{y}$, made scalar for the sake of simplicity, are random processes over some time axis t (not necessarily physical time). Prediction is required by the client for the target process $\mathcal{y}$ at times $t_{\mathcal{y}}$, and their $\mathcal{W}$-related feature process $\mathcal{x}$ is observable to the client at times t_x. We then have:

$$legit\{\mathcal{y}\} \subseteq \{x \in \mathcal{X} | t_x < t_{\mathcal{y}}\}. \tag{3}$$

Such a rule should be read: Any legitimate feature w.r.t. the target process is a member of the right hand side set of features. In this case the right hand side is the set of all features whose every instance is observed earlier than its $\mathcal{W}$-related target instance. We are assuming with this notation that $\mathcal{X}$ contains all possible features, and use "⊆" to express that additional legitimacy constraints might also apply (otherwise "=" could be used).

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 $t_{\mathcal{y}}$ as in (4) below 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 $t_{\mathcal{y}}$ makes the problem uninteresting. Such is the case for the "heavy spender" example from [3]. 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".

$$legit\{\mathcal{y}\} \subseteq \{x \in \mathcal{X} | t_x < t_{\mathcal{y}} - \tau\}. \tag{4}$$

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

$$legit\{\mathcal{y}\} \subseteq \{x \in \mathcal{X} | t_{\mathcal{y}} - \tau < t_x < t_{\mathcal{y}}\}. \tag{5}$$

We can think of a requirement to use exactly n features from a specified pool $\mathcal{X}_p$ of preselected features:

$$legit\{\mathcal{y}\} \subseteq \{[x_1, \ldots, x_n] | \forall k \; x_k \in \mathcal{X}_p, k \; unique\}, \tag{6}$$

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:

$$legit\{\mathcal{y}\} \subseteq \mathcal{X}_p. \tag{7}$$

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

$$legit\{\mathcal{y}\} \subseteq \{x \in \mathcal{X}_p | t_x < t_{\mathcal{y}}\}. \tag{8}$$

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 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 [8, 13] 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.

It is easy to see how conditions (2) and (3) similarly apply to the account number and interviewer name examples from [10], the session length of [2] (while the corrected "page number in session" is fine), the immediate and indirect triggers described in [3, 4], the remaining competitions described in [8, 13], and the website based features used by IBM and discussed in [13]. However not all examples fall under condition (2).

Let us examine the case mentioned earlier of KDD-Cup 2007 as discussed in [11]. While clearly taking advantage of information from reviews given to titles during 2006 (the mere fact of using data from the future is proof, but we can also see it in action by the presence of measurable leakage – the fact that this model performed significantly better both in internal tests and the final competition), the final delivered model $\mathbb{M}$ does not include any illegitimate feature[1]. 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 $\mathcal{y}_t$ for $t = [101, 102, \ldots, 200]$, clearly a hopeless task. Suppose further that for the purpose of predicting $\mathcal{y}_t$, t itself is a legitimate feature but otherwise, as in (3), only past information is deemed legitimate – so obviously we cannot cheat. Now consider a model trained on examples $\boldsymbol{W_{tr}}$ taken from $t = [1, 2, \ldots, 200]$. The proposed model is $\widehat{y_t} = \mathbb{M}(\mathrm{t}, \boldsymbol{W_{tr}})$, a table containing for each $\boldsymbol{t}$ the target's realized value $\boldsymbol{y_t}$. Strictly speaking, the only

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

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 $y \in Y_{ev}$,

$$\forall X \in X_{tr}, X \in legit\{y\} \wedge \forall \tilde{y} \in Y_{tr}, \tilde{y} \in legit\{y\} \qquad (9)$$

where Y_{ev} is the set of evaluation[2] target instances, and Y_{tr}, X_{tr} are the sets of training targets and feature-vectors respectively whose realizations make up the set of training examples $\boldsymbol{W_{tr}}$.

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 instance the model is called to generate a prediction for.

For modeling problems where the usual i.i.d. instances assumption is valid, and when without loss of generality considering all information specific to the instance 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 non-stationarity, a.k.a. concept-drift [15], and more specifically the case when samples of the target (or, within a Bayesian framework, the target/feature) are not mutually independent, condition (9) cannot be reduced to condition (2). Such is the case of KDD-Cup 2007. Available information about the number of reviews given to a group of titles for the "who reviewed what" task is not statistically independent of the number of reviews given to the second group of titles which is the target in the "how many ratings" task. The reason for this is that these reviews are all given by the same population of users over the same period in 2006, and thus are 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, dependence is bound to occur.

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 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: (i) selecting or designing features that will have predictive power in deployment, but don't show this power on training examples, (ii) algorithm or parametric model selection, and (iii) meta-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* [5].

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.

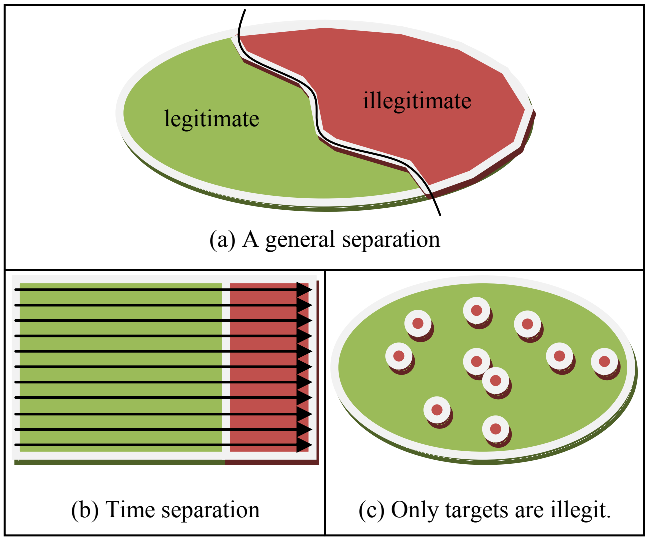
As a final point on legitimacy, let us mention that once it has been clearly defined for a problem, the major challenge becomes preparing the data in such a way that ensures models built on this data would be leakage free. Alternatively, when we do not have full control over data collection or when it is 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.

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 (x, y) of observational input instance x and target instance y, sufficient for answering the question "is x legitimate for inferring 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



(a) A general separation

(b) Time separation

(c) Only targets are illegit.

Figure 1. An illustration of learn-predict separation.

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

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 1) 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 time-stamps with sufficient precision. Legitimacy tagging is implemented by time-stamping every observation. Learn-predict separation is implemented by a cut at some point in time that segments training from evaluation examples. This is what has been coined in [13] *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 time-stamped and not stored separately, and at best whole records end up with one time-stamp. 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 time-stamping, causing observations taken before *and* after the target's time-stamp 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 Reviews" task in itself was (as far as we are aware) well time-stamped 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 Reviewed What" task contained leakage was due to an external source of leakage, an issue related with data mining competitions which we shall discuss next.

4.2 External Leakage in Competitions

Our account of leakage avoidance, especially in light of our recurring references to data mining competitions in this paper, 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 Reviews" task, the INFORMS 2010 financial forecasting challenge, and the IJCNN 2011 Social Network Challenge[3].

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.

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 paper. 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 which left traces and the online original source for competitors to find.

5. DETECTION

Often the modeler doesn't have 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 [14] 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

[3] 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).

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.

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 high entropy 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 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.

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 over-fitting, tangible concept-drift, issues with the design of the field-test such a 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 above, 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 [4], 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 you are still left with the 'no purchase in any department' signature. Another example for this is 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 [8]).

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 [16].

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, perhaps impossible, and that modeling with the remaining leakage is often the preferred alternative to futile leakage removal efforts.

7. CONCLUSION

It should be clear by now 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 paper 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 paper 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.

Many cases of leakage happen when in selecting 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 vs. free shipping). In other cases, the modeler is well aware of the implications of his selection, but falters when facing the tradeoff 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 case studies and suggested methodology described in this paper can help save projects and competitions from falling in the leakage trap and allow them to encourage models and modeling approaches that would be relevant in their domains.

8. REFERENCES

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

[2] Kohavi, R., Brodley, C., Frasca, B., Mason, L., and Zheng, Z. 2000. KDD-cup 2000 organizers' report: peeling the onion. *ACM SIGKDD Explorations Newsletter*. 2(2).

[3] Kohavi, R. and Parekh, R. 2003. Ten supplementary analyses to improve e-commerce web sites. In *Proceedings of the Fifth WEBKDD Workshop*.

[4] Kohavi, R., Mason L., Parekh, R. and Zheng Z. 2004. Lessons and challenges from mining retail e-commerce data. *Machine Learning*. 57(1-2).

[5] Lo, A.W. and MacKinlay A.C. 1990. Data-snooping biases in tests of financial asset pricing models. *Review of Financial Studies*. 3(3) 431-467.

[6] Narayanan, A., Shi, E., and Rubinstein, B. 2011. Link Prediction by De-anonymization: How We Won the Kaggle Social Network Challenge. *Proceedings of the 2011 International Joint Conference on Neural Networks (IJCNN). Preprint.*

[7] Nisbet, R., Elder, J. and Miner, G. 2009. *Handbook of Statistical Analysis and Data Mining Applications*. Academic Press.

[8] Perlich C., Melville P., Liu Y., Swirszcz G., Lawrence R., Rosset S. 2008. Breast cancer identification: KDD cup winner's report. *SIGKDD Explorations Newsletter*. 10(2) 39-42.

[9] Pyle, D. 1999. *Data Preparation for Data Mining*. Morgan Kaufmann Publishers.

[10] Pyle, D. 2003. *Business Modeling and Data Mining*. Morgan Kaufmann Publishers.

[11] Pyle, D. 2009. *Data Mining: Know it All*. Ch. 9. Morgan Kaufmann Publishers.

[12] 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 Explorations Newsletter*. 9(2).

[13] Rosset, S., Perlich, C., Swirszcz, G., Liu, Y., and Prem, M. 2010. Medical data mining: lessons from winning two competitions. *Data Mining and Knowledge Discovery*. 20(3) 439-468.

[14] Tukey, J. 1977. *Exploratory Data Analysis*. Addison-Wesley.

[15] Widmer, G. and Kubat, M. 1996. Learning in the presence of concept drift and hidden contexts. *Machine Learning*. 23(1).

[16] Xie, J. and Coggeshall, S. 2010. Prediction of transfers to tertiary care and hospital mortality: A gradient boosting decision tree approach. *Statistical Analysis and Data Mining*, 3: 253–258.