New York University - Stern School of Business
Working paper CBA-15-02

Dimensionality Reduction via Matrix Factorization for Predictive Modeling from Large, Sparse Behavioral Data

Jessica Clark and Foster Provost

Department of Information, Operations and Management Sciences

NYU Stern School of Business

May 19, 2015

Abstract

Matrix factorization is a popular technique for engineering features for use in predictive models; it is viewed as a key part of the predictive analytics process and is used in many different domain areas. The purpose of this paper is to investigate matrix-factorization-based dimensionality reduction as a design artifact in predictive analytics. With the rise in availability of large amounts of sparse behavioral data, this investigation comes at a time when traditional techniques must be reevaluated. Our contribution is based on two lines of inquiry: we survey the literature on dimensionality reduction in predictive analytics, and we undertake an experimental evaluation comparing using dimensionality reduction versus not using dimensionality reduction for predictive modeling from large, sparse behavioral data.

Our survey of the dimensionality reduction literature reveals that, despite mixed empirical evidence as to the benefit of computing dimensionality reduction, it is frequently applied in predictive modeling research and application without either comparing to a model built using the full feature set or utilizing state-of-the-art predictive modeling techniques for complexity control. This presents a concern, as the survey reveals complexity control as one of the main reasons for employing dimensionality reduction. This lack of comparison is troubling in light of our empirical results. We experimentally evaluate the efficacy of dimensionality reduction in the context of a collection of predictive modeling problems from a large-scale published study. We find that utilizing dimensionality reduction improves predictive performance only under certain, rather narrow, conditions. Specifically, under default regularization (complexity control) settings dimensionality reduction helps for the more difficult predictive problems (where the predictive performance of a model built using the original feature set is relatively lower), but it actually decreases the performance on the easier problems. More surprisingly, employing state-of-the-art methods for selecting regularization parameters actually eliminates any advantage that dimensionality reduction has! Since the value of building accurate predictive models for business analytics applications has been well-established, the resulting guidelines for the application of dimensionality reduction should lead to better research and managerial decisions.

1 Introduction

Many businesses now collect the vast amounts of high-dimensional, sparse behavioral data that their users generate. For instance, banks know the merchants where their customers spend money; social network sites like Facebook can track all of the items that their users demonstrate that they are interested in; and online ad exchanges rely on browser page-visitation history to serve up more relevant advertisements to users. The established value of such massive, high-dimensional data (Junqué de Fortuny et al. 2013) leads to more interest in utilizing it. Under the belief that this kind and amount of data may necessitate a fundamentally different set of methods, it is essential to reevaluate the tools that have traditionally been used for analytics for smaller data with different characteristics. There a few different ways to approach design science research in Information Systems (Hevner et al. 2004). It can either develop, contextualize, and evaluate a new tool or method, or it can look more deeply at an existing artifact that is broadly utilized but may be misunderstood or misused. This paper takes the second approach, examining the viability of a widely used technique, dimensionality reduction (DR) based on matrix factorization, as a tool to assist in predictive modeling problems, in particular modeling large, sparse behavioral data.

In the business analytics literature for predictive modeling, dimensionality reduction is widely accepted as a beneficial step in preprocessing and exploring the feature space. This can be seen in business analytics applications ranging from spam detection (Cai et al. 2013) to estimation of click volume on web ads (Lifshits and Nowotka 2007) to customer churn prediction (Coussement and Van den Poel 2008). A quote from a recent research essay about the role of predictive analytics in Information Systems research is illustrative: "Due to the often large number of predictors, reducing the dimension can help reduce sampling variance (even at the cost of increasing bias), and in turn increase predictive accuracy" (Shmueli and Koppius 2011). The authors subsequently codify dimensionality reduction as one of their steps for building a predictive model; however, they do not specify when one should or should not apply it. Consider another recent quote from an article in *Management Science*: "Given the large number of potential demographics available, data dimension reduction is an important factor in building a predictive model that is easy to interpret, cost effective, and generalizes well" (Kim et al. 2005).

First, it is necessary to clarify what we mean in this paper by predictive modeling. We specifically

refer to using models induced from data to predict a predefined target variable.[1] The relative success of these models is assessed by computing some measure of generalization accuracy (on held-out data). Thus, as opposed to models whose purpose is to estimate the degree to which various covariates cause changes in the target variable, here we will focus solely on doing a good job of predicting the value of the target. Predictive modeling is important both for research and for practice. The aforementioned study by Shmueli and Koppius explains the variety of ways in which predictive modeling can be valuable for scientific research. For many business applications, such as those listed above, it's useful to have an accurate prediction for some phenomenon in order to take actions more cost-effectively.

It is also necessary to distinguish the use of DR in papers that seek to establish a causal relationship among variables. DR can be used for model, method, or instrument validation, as in Mishra et al. (2007), to increase the interpretability of models for end users (Kiang and Kumar 2001), or for exploratory data analysis (McKnight et al. 2002). Statistically, it can be used to overcome multicollinearity in regressors (Greene 2003). These inference problems are related, and may have overlapping concerns with predictive modeling; however, we set aside such problems as being outside this paper's scope, in which we focus on improving predictive (generalization) performance.

Other than predictive modeling, we see three main (related) domains where DR is widely and successfully applied. In order to process images, features must first be extracted from the individual pixels. Principal components analysis and other forms of dimensionality reduction are frequently employed to accomplish this (Turk and Pentland 1991). Image processing is different from most business analytics problems in the structure of the data, which is generally dense, as well as having the characteristic that any individual feature (pixel) carries very little information. In the field of information retrieval, latent semantic indexing reduces the dimensionality of a term $\times$ document matrix representing a corpus of text documents to overcome the issues of polysemy and synonymy (Deerwester et al. 1990). IR generally can have arbitrary and unforeseen target variables, as opposed to the predefined target variables that we are normally interested in with traditional predictive modeling and which are the focus of this paper. In collaborative filtering, a matrix of user $\times$ item ratings is commonly stochastically factored into two lower-rank matrices, then reconstructed

[1] *Predict* here can refer to an estimation of the value of any variable for which we don't know the value, not just events that may happen in the future (Provost and Fawcett 2013).

to predict ratings for the items that each user has not yet rated. This technique was famously used to win the Netflix prize (Bell et al. 2007). CF is also a different problem from traditional predictive modeling—here, we are interested in predicting all ratings that a user has not supplied, or alternatively finding *some* good recommendations from this set. It is possible to force the CF problem into a predictive modeling structure, with every item as a different target variable; however it is a special case as for each individual predictive problem the features and target are generally drawn from the same domain.

All of these problems have characteristics in common. The original data potentially have more features (dimensions) than instances. Additionally, these are all domains where we might think that there exist latent regularities underlying the data—e.g., image features in image processing, concepts in the case of LSI, movie genres in the case of the Netflix prize. Perhaps because of its success in these three areas, DR is frequently applied in standard predictive modeling problems, sometimes without a comparison to modeling with the original feature set. The question of whether or not to reduce dimensionality can be viewed as a design decision akin to the choice of classifier, and in that light it is not surprising that DR is a good idea in some contexts and not in others.

A key goal of this paper is to survey the DR and predictive modeling literature and examine if (and when) DR is generally "good" for predictive modeling. That is, when the dimensionality of the feature set is reduced (using standard techniques), is predictive performance better than when using the original feature set? Our paper differs from other surveys, notably van der Maaten et al. (2009), in both the depth of our investigation and the type of data we are interested in. Previous investigations into dimensionality reduction generally do not consider the sparse behavioral data now common in business applications, nor data sets with more than $10,000$ instances. A second key contribution of the paper is the presentation of a novel comparison in the context of regularization, which we will see is crucial for comparing predictive modeling with and without DR.

We begin by identifying exactly what "dimensionality reduction" refers to as well as its role within the predictive modeling process. Additionally, there are several important subquestions that we must address in order to fully understand DR as a design artifact. First, we lay out the theoretical reasons why one might think that DR would be a useful technique for feature extraction for predictive modeling. For a given problem, it is important to understand why it may be desirable to apply DR. This will lead to better understanding of the tradeoffs that may be inherent in pursuing

DR versus using other tools. We then survey the literature for work that utilizes DR for feature extraction for predictive modeling, and assess the work in the context of these theoretical reasons and also examine whether they report improvements in performance.

To illustrate the lack of clarity when deciding to apply DR or not, consider the intriguing recent paper by Kosinski et al. in PNAS (Kosinski et al. 2013). The authors demonstrate that it is possible to predict important and sensitive personal traits of individuals using only the items that they have Liked on Facebook. Users can indicate support for a wide variety of content items, including media, brands, Facebook content, people, and statements by Liking them. A key part of the authors' methodology involves applying exactly the methodology we study: conduct DR on the user-Like matrix before building logistic regression models using the resulting reduced-dimensional features to predict the desired traits.

Consider as a second example a paper published in the most recent proceedings of the top data science conference: it uses both website text and network characteristics to identify sites that sell counterfeit and black-market goods such as pharmaceuticals (Der et al. 2014). Again, they apply exactly the procedue we study (and that is prescribed in the aforementioned papers). The original feature count for describing the websites is about 65,000; after applying DR the authors find that most of the variance in the features can be accounted for using only 72 of the reduced dimensions. They then build predictive models on the resulting reduced-dimension space to classify sites, reporting high precision and recall.

Indeed, following the discussion above, both of these data sets initially appear to be ideal candidates for dimensionality reduction: there are more features than instances, the instances are largely sparse, and the data intuitively seem like they ought to contain latent information that could be captured via dimensionality reduction. However, neither of these papers shows whether the DR actually improved the predictive performance; indeed, for all we know, the predictive performance might have been better without the added complexity of the DR.

As this combination—building a predictive model on the dimensions resulting from DR via matrix factorization—has become common both in prescriptions in the literature and in predictive modeling practice, it is important for us to examine it closely from a design science perspective. We present an experimental comparison across a suite of real-world data sets of the sort now collected by many businesses. The comparison demonstrates that contrary to what one would

infer from the literature, better predictive performance can often be attained by simply not using the DR component of the modeling process (i.e., building a model on the full feature set rather than the reduced-dimensional feature set). Thus, the added complexity of the DR stage in many analytics information systems is unnecessary for improving predictive performance, and indeed may be reducing performance.[2] Our main goal is to better understand the use of dimensionality reduction for predictive modeling, but our results could also serve as a cautionary tale to businesses and researchers demonstrating the necessity of considering carefully whether or not to apply each component of an analytical system.

2 General Background and Notation

We start by assuming that we have n data instances, each composed of the value for a binary target variable that we would like to predict, as well as the values for d attributes arranged in a feature vector. Each instance u has a positive association with at least one feature i. Assume that this association is binary. Therefore, the data can be represented as an $n \times d$ binary matrix A such that $A_{ui} = 1$ when u is associated with i and 0 otherwise, plus a target vector $\mathbf{y}$ where $y_u \in \{0, 1\}$.

To give a motivating example, an instance might be a Facebook user where each attribute represents one item that she might Like. Facebook Likes are defined in the aforementioned study by Kosinski et al. as "a mechanism used by Facebook users to express their positive association with (or "Like") online content, such as photos, friends' status updates, Facebook pages of products, sports, musicians, books, restaurants, or popular Web sites." We can represent each user's feature vector as being 1 where the user has Liked the corresponding item and 0 otherwise. We want to predict demographic traits (such as gender) or psychographic traits (such as intelligence) using this Like information. To predict, say, gender, A and $\mathbf{y}$ would thus be defined as

$$A_{ui} = \begin{cases} 1 & \text{if } u \text{ Liked } i \\ 0 & \text{else} \end{cases}, \mathbf{y}_u = \begin{cases} 1 & \text{if } u \text{ is female} \\ 0 & \text{else} \end{cases}$$

Let us further assume that we want to apply dimensionality reduction to our problem. We will

[2] While the predictive performances in cited papers are very strong, and thus we are not calling into question the conclusions of their studies, they may be improved with the methodological change of not reducing the dimensionality of their feature space at all.

further explain some of the reasons why this might be considered desirable below. There are many ways to reduce the dimensionality of our feature data. A review and comparison of some of these techniques can be found in van der Maaten et al. (2009) and others. Because our intention here is not to rehash comparisons of dimensionality reduction techniques that have been explored elsewhere, going forward we will specifically be referring to matrix factorization methods of dimensionality reduction, and Singular Value Decomposition (SVD) in particular. SVD is massively popular as a form of preprocessing and feature extraction in predictive modeling and classification problems. To the extent that dimensionality reduction is used in Information Systems research, this is usually the method of choice (Kim et al. 2005). It has been used for dimensionality reduction across many domains, including text classification (Yang 1995), facial image recognition (Turk and Pentland 1991), network security (Xu and Wang 2005), and neuroscience (López et al. 2011). The Facebook Likes study introduced above used SVD.[3] Due to the general ubiquity of SVD's use in predictive modeling research and practice, let's assume going forward unless otherwise stated that when computing dimensionality reduction (DR), it is accomplished via SVD.

2.1 Dimensionality Reduction via Singular Value Decomposition

SVD computes matrices L, S, and R whose product equals our data matrix A. The SVD of A is

$$A = LSR. \tag{1}$$

This decomposition has useful properties. The columns of L (known as left-singular vectors) and rows of R (right-singular vectors) are orthonormal. S is a diagonal matrix, and the columns of L and rows of R are usually sorted in decreasing order of the corresponding values in S, which represent the amount of sample variance in the original data accounted for by the corresponding

[3] van der Maaten et al. (2009) surveyed numerous linear and nonlinear dimensionality reduction techniques across a variety of classification problems and found that using principal components analysis (PCA) features as input yielded the best classification accuracy of all of the dimensionality reduction techniques. The authors also found that with non-sparse, medium-sized data (up to 10,000 instance), not reducing the dimensionality of the features actually yielded the best performance in the majority of cases—a finding that we will explore in greater detail below. Note that PCA is SVD where the sample mean of each feature is 0, which typically is achieved via the obvious preprocessing step. Subtracting the sample mean from all features destroys the sparsity of a data set, which is critical for the large, sparse data that we explore in this paper. For example, the Facebook dataset we describe below would go from having about 35 million nonzero entries to having about 37 billion nonzero entries—several orders of magnitude larger than the famous Netflix Prize dataset. For this reason, the sample mean normally is not subtracted out for sparse data, technically yielding SVD rather than PCA. Often in practice the terms are used interchangeably.

singular vectors. In many practical applications, most of the sample variance is accounted for by the first few singular vectors; therefore, it is common to choose some $k < n, d$ and compute the truncated SVD which consists of the first k singular vectors:

$$A \approx L_k S_k R_k. \tag{2}$$

Importantly, the resulting $n \times k$ matrix L_k can be viewed as representing some latent structure in the data. The k columns of L_k are sometimes known as the "latent factors" that underly the original data; here we will refer to them as the SVD components. Each instance will have a representation in this new k-dimensional space captured by L_k. The corresponding rows of the $k \times d$ matrix R_k are sometimes referred to as the "loadings" of the components, representing how strongly each data instance is associated with each component. Once the SVD has been computed, the resulting latent factors may be used as features in a predictive model.[4]

2.2 Choosing the Number of SVD Components

In this subsection we discuss choosing k, the number of components to select when computing dimensionality reduction. There is substantial evidence that utility peaks with a certain number of dimensions.[5] In fact, "selection of number of dimensions" is mentioned as being important in almost every paper that utilizes dimensionality reduction for predictive modeling, which we discuss below. There is no a priori way of discovering the optimal number of dimensions. Furthermore, certain recent papers have indicated that for large-scale problems, the optimal number of dimensions may be very large, larger than previously considered. This is one area where large, sparse behavioral data may cause differences from what is traditionally understood in DR application.[6]

The first method for finding the optimal number for k uses an operational criterion: that is, picking k that optimizes predictive performance on held-out data. This is what Deerwester et al. (1990) do, demonstrating that the precision of their information retrieval system increases from .2 to about .5 and plateaus at about $k = 100$. In an updated version of that paper (Dumais 1992), they extend the number of components to 1100 and find that after about 100 dimensions performance

[4] See (Hastie et al. 2009) for additional technical details.

[5] For some examples, see Deegalla and Bostrom (2006), Hu et al. (2007), and Coussement and Van den Poel (2008).

[6] Examples where performance continues to improve past a small number for k can be seen in Sarwar et al. (2000a), Raeder et al. (2013), and (Koren et al. 2009).

begins to degrade. Another way of choosing k is by ranking the singular values (elements of S) and picking the top few based on visual inspection, for instance, by looking for a "knee" in the singular values (Hoff 2007), (Burl et al. 1994).[7] It is very important, however, to consider the troublesome statistical multiple comparisons problem (Jensen and Cohen 2000) that is inherent to choosing the "best" k after the fact—similarly to how we would avoid choosing other modeling components by observing their performance on the testing data.

One could choose k by doing some sort of (nested) cross-validation. For instance, Owen and Perry (2009) show a method for holding out data, computing SVD on the non-held-out data, and selecting k so as to minimize the reconstruction error between the held-out data and its SVD approximation. As with the selection of other modeling hyperparameters, it is important to ensure that the cross-validation used to select the parameters does not use the data held out for the ultimate estimation of generalization performance, as with nested cross-validation (Provost and Fawcett 2013). Because of the need for such care to avoid issues with multiple comparisons, finding the "optimal" number of dimensions empirically can be computation (and time) intensive. We will return to this below in the experimental comparison.

Finally, k can be chosen such that the resulting latent dimensions account for a predetermined portion of the total variance in the data. This is done by Der et al. (2014), who select sufficient factors to account for 99% of the variance.

3 Dimensionality Reduction in Predictive Modeling Survey

Practitioners and researchers should understand the proper theoretical or empirical rationale behind why they are choosing to utilize a particular modeling technique, including dimensionality reduction, for a particular modeling application. To that end, in this section we present the results of a survey of what has been studied and written on the use of dimensionality reduction for predictive modeling. The survey additionally reveals that there is a substantial amount of predictive modeling work which employs dimensionality reduction absent any explicit rationale (theoretical or empirical) for doing so. Our empirical results in Section 5 further support the necessity for establishing such a rationale and for careful comparison.

[7] Burl et al. (1994) also use visual inspection of the latent factors themselves, given that the data are images.

3.1 Why Dimensionality Reduction?

As mentioned above, in this paper we will consider dimensionality reduction to be beneficial if a predictive model built using the reduced space as features has superior predictive performance to a model built using the original features. The literature provides a strong theoretical basis for why we think this might happen.

Famously, dimensionality reduction can be employed to overcome the "Curse of Dimensionality" (Bellman 1961), which can manifest itself in numerous ways. In particular, traditional statistical modeling techniques break down in high-dimensional spaces. Techniques which depend on measuring similarity of instances fail since no two instances appear similar in high-dimensional space (Domingos 2012). Some algorithms have been developed that overcome this "Curse" (Gionis et al. 1999); however, SVD can also be applied to the data. DR is thus utilized to avoid overfitting in these cases. This reasoning is commonly employed in papers that use such data.[8]

The second theoretical foundation for why DR may help in predictive modeling draws on the bias/variance tradeoff in predictive error (Friedman 1997). That is, that by reducing dimensionality, we reduce training variance at the cost of introducing some bias. The hope is that the overall error will be diminished (Shmueli and Koppius 2011). Feature selection and regularization improve predictive performance by trying to exploit this tradeoff (Friedman 1997), (Scharf 1991), (Domingos 2012), (Kim et al. 2005). (Feature engineering more generally may also try to reduce bias, usually at the expense of increased variance.) Related is the issue of collinearity of variables: high-dimensional data often have collinear variables, introducing variance in regression (as compared to having no collinear variables). Since the resulting components from SVD are orthogonal, they are by construction not collinear. Thus SVD contributes to reducing the variance of the estimator (Mandel 1982).

There are other reasons to compute dimensionality reduction besides improving predictive performance. While we do not consider accomplishing these tasks within the scope of this paper, they are well-explored in the literature and worth mentioning in the service of understanding dimensionality reduction as a design artifact. They are also frequently mentioned in papers where the main goal is predictive accuracy, whether as a side benefit or an additional objective.

[8] See (López et al. 2011), (West et al. 2001), and (Burl et al. 1994) for a few examples.

Delineating among these reasons is crucial since it may be necessary to trade off among goals. Being knowledgeable and cautious about these tradeoffs will hopefully lead to better use of DR.

DR can be used as a tool to compress large data sets, whether to save space or time; however, SVD and other model-based methods are time consuming to compute.[9] Relatedly, SVD is also used simply to improve the "manageability" of the feature space, as in Thorleuchter et al. (2012). Unfortunately, in the context of the data on which this paper focuses, if the original matrix A is sparse, the resulting L and R matrices from SVD will not necessarily be sparse, potentially dramatically increasing the space required to store the features. While there do exist sparse DR techniques, they are generally slower to compute than SVD and don't come with the same guarantees since they are not computed using convex optimization (Zou et al. 2006). Because of these disadvantages, it is helpful to focus on where SVD still can be useful for compression for predictive modeling. SVD can be quite helpful for saving training time if the same compressed feature set will be reused numerous times for different predictive problems. For instance, in our domain of predicting personal traits, we can compute the SVD of the Facebook Likes matrix once as a preprocess, then reuse the reduced matrix to learn models to predict each trait.

Methods such as feature hashing (Shi et al. 2009) and random projections (Achlioptas 2001) provide ways of randomly selecting or combining the original data. They are designed to reduce the size of data sets, are fast to compute, and sometimes come with theoretical bounds on the amount of information lost due to compression (Shi et al. 2009), (Achlioptas 2001). Predictive performance-wise, the resulting compressed feature sets can also be experimentally compared with feature sets based on SVD (Fradkin and Madigan 2003) or with the original feature set (Weinberger et al. 2009). These methods generally are not employed to improve predictive accuracy.

Another reason to compute DR on a feature space is to detect interpretable latent structure in data. This can be important if it is necessary to explain the features in the model, perhaps to a domain expert or manager. This reasoning is similarly employed in causal inference papers, as in Kiang and Kumar (2001). Additionally, the latent factors can be an end goal in and of themselves. Thus, DR can be seen as a way of preprocessing the data to visually inspect it, as described in Westad et al. (2003) and Hubert et al. (2000). There are plenty of DR methods that were designed with interpretability of the latent factors as a goal, including sparse PCA (Zou et al.

[9] SVD's computation time is $\mathcal{O}(n^3)$.

2006), non-negative matrix factorization (Seung and Lee 2001), and Latent Dirichlet Allocation (Blei et al. 2003).

3.2 Dimensionality Reduction for Other Problems

Dimensionality reduction is very commonly used for image recognition. It is first necessary to note that image data is very different from that found in many business applications and in particular the sparse, high-dimensional data that motivates this paper. First, images are not sparse. Therefore, computing the SVD of image data actually does reduce storage space, instead of destroying sparsity. Second, images are represented as vectors of individual pixels. Unlike in the type of business domains we focus on here, one individual pixel value is generally uninformative. Consider in contrast the nature of the features in the Facebook Likes problem. An individual Like could intuitively give considerable information about the target variable. Similarly, a URL that a user clicked on or a merchant to which she sent money may conceivably be informative about a business or behavioral target variable. The color value of any individual pixel is unlikely to tell us anything about any target variable - for instance, whether the image is of a cat or a dog.[10] Therefore, some form of feature extraction is usually necessary to create usable features from pixel data.

Historically, many applications of DR to images are facial recognition problems—comparing an image of a person against a database of known persons to determine which person it is. Turk and Pentland (1991) present a foundational paper that describes how to use dimensionality reduction (via PCA) to accomplish this, arguing that this is an efficient way to learn faces as well as a nice compact representation. They compute the PCA of the database of known faces, then project any new faces into the "eigenface" space. They then compare new faces against the existing faces to find the most similar one. This is originally based on ideas presented by Sirovich and Kirby (1987). Fisherfaces is a related technique, presented as a supervised alternative to PCA (Belhumeur et al. 1997). PCA and SVD are also used frequently in image classification problems for biological data. The nature of the data sources such as brain scans leads to not many instances being available. These papers often justify their use of DR by pointing out that there are many more features than images. Examples include breast cancer status prediction (West et al. 2001), tumor classification

[10] One exception is skin tone for classifying images of adult content, but still there it is the amount of skin tone across all pixels; one pixel alone may not be informative.

(Antoniadis et al. 2003), and early detection of Alzheimer's using brain scans (López et al. 2011).

Another domain in which DR is commonly employed is Information Retrieval. The goal of information retrieval is to be able to find the most relevant documents in a corpus to any query. Both documents and queries consist of terms, so queries can be thought of as mini-documents.[11] The setting for information retrieval is slightly different from that in predictive modeling. Instead of one target variable being the focus of prediction, there are potentially infinite prediction tasks, since a user can presumably submit any query to an IR system. Latent Semantic Indexing (LSI) (Deerwester et al. 1990) reduces the dimensionality of a corpus of text documents via SVD for IR purposes. The resulting SVD components are interpreted as "concepts." Representing documents and queries in terms of their underlying concepts helps to overcome natural issues present in text representations, namely synonymy (multiple words with the same meaning) and polysemy (one word with multiple meanings). This famous technique has been used widely for information retrieval (Manning et al. 2008). There have been studies comparing SVD to other methods of dimensionality reduction for IR purposes (Kumar 2009) as well as modifying and improving upon LSI (Hofmann 1999), (Blei et al. 2003). LSI and its relatives have been shown to be demonstrably more effective for information retrieval than direct term matching (using unreduced data).

It makes intuitive sense that preprocessing techniques that work well in information retrieval would also apply to large-scale predictive modeling using sparse data, despite the structural differences in the problems. Text data is certainly large and sparse, and document collections can be large. Further, information retrieval is very similar to predictive modeling in nature. Therefore, we might expect that applying the same technique to the feature space for predictive modeling would be effective. Indeed, LSI has been extensively used for text classification.[12]

Collaborative filtering is closely related to both predictive modeling and information retrieval. Recommender systems utilize users' past ratings on various items such as movies or books to recommend new items to the users, often by predicting ratings on unrated items. Collaborative filtering systems accomplish this by recommending items rated highly by users with similar ratings histories (Adomavicius and Tuzhilin 2005). The general collaborative filtering (CF) problem is often cast as: predict the rating that a given user u will assign to an item i, then recommend items with

[11] Note that we use the words "document" and "terms" loosely since this may be applied in other, non-text domains.
[12] See the citations in Sebastiani (2002).

high predicted ratings. Examples include Netflix's movie recommender system and Amazon's book (and other item) recommender system. This can be seen as a matrix completion problem—given a matrix containing known entries r_{ui} where user u has rated item i, try to predict how the rest of the matrix will be filled in.

As explained by Koren (2010), there are two main approaches to collaborative filtering problems: neighborhood methods and latent factor models. Latent factor models use matrix factorization or other forms of dimensionality reduction to fill in missing items in the user-item matrix. There are several recognized ways to use matrix factorization in CF. First, the actual SVD of the matrix can be computed. Because SVD cannot be computed on an incomplete matrix, sometimes the entries are filled in with averages, as in Sarwar et al. (2000b). The other option is to stochastically compute left and right matrices entry-by-entry optimizing a regularized objective function that minimizes the difference between the reconstructed matrix and the original data. Either way, the resulting decomposed low-dimensional matrices are then multiplied together to get predictions for the previously unfilled spots (Koren et al. 2009). Typically the quality of the predictions is measured as the MSE between the true and estimated values of selected held-out cells of the data matrix.

Matrix factorization techniques in CF are generally accepted as yielding good results but being time and memory intensive (Bobadilla et al. 2013). This is supported by an extensive line of research, starting with Billsus and Pazzani (1998), who introduced using SVD for CF by proposing using a subset of the singular vectors as training examples for predicting users' movie ratings. Many more papers that use this technique to attain good performance on CF tasks followed, such as in Sarwar et al. (2000b), Mobasher et al. (2004), Vozalis and Margaritis (2005), Takács et al. (2009), Vozalis and Margaritis (2007), Paterek (2007), and Cacheda et al. (2011). Perhaps the most famous example is the SVD-based technique used as part of the algorithm that won the Netflix Prize (Koren 2009). There are also a few papers where SVD was demonstrated to be inferior to non-DR techniques on particular data sets or settings, such as Huang et al. (2007) and Sun et al. (2005).

SVD's demonstrated good performance on CF and IR problems suggests that we should be able to successfully apply SVD to predictive modeling problems where the data looks like CF or IR data—that is, data that are large and sparse (as in our experimental study, below). However, the reported performance of DR techniques for feature extraction in predictive modeling has been mixed.

3.3 (Mixed) Empirical Evidence from Predictive Modeling

As we see in the papers discussed so far, there is the perception that doing dimensionality reduction is generally a good idea for predictive modeling. This perception perhaps is based in part on the success of matrix factorization in image processing, information retrieval, and collaborative filtering; however, these settings are not identical to predictive modeling.

There have been some papers in which a model built on a reduced-dimensionality feature set has been shown to achieve better generalization performance than a model built using the original features. Several of these papers have the objective of demonstrating the usefulness of one or more particular dimensionality reduction methods. Specifically, the dimensionality reduction is introduced: to reduce the noisiness of data (Ahn et al. 2007); to uncover latent meaning in the data (Whitman 2003), (Blei et al. 2003); to overcome poor features (as in images) (Subasi and Ismail Gursoy 2010); to combat polysemy and synonymy, as IR (Karypis and Han 2000); or to speed up the modeling (Fruergaard et al. 2013). Other papers mention that a comparison has been undertaken but do not explicitly state the results (West et al. 2001).

Note, however, that none of these papers mentions that the predictive modeling used state-of-the-art methods for complexity control, as discussed above in Section 1. Without good complexity control, improved generalization performance may simply be due to the dimensionality reduction acting as a regularization mechanism—and being compared to poorly regularized modeling. We explore this further below in Section 4.

On the other hand, there are papers that demonstrate dimensionality reduction for predictive modeling performing worse than not using DR, such as Raeder et al. (2013) and Xu and Wang (2005). Others report that DR resulted in mediocre performance (Guyon et al. 2009). Most tellingly, however, is the aforementioned survey by van der Maaten et al. of linear and nonlinear dimensionality reduction techniques. This survey utilized both "natural" and "synthetic" data sets.[13] They found that on the majority of their "natural" datasets, classification error was not improved by doing any form of dimensionality reduction on the feature space. However, they hedge their bets by explaining this as likely being due to an incorrect choice of k. Symmetrically with the concern noted

[13]Their results differ from ours, below, in that (i) they do not use sparse behavioral data, (ii) their datasets are limited in size to fewer than 10,000 instances, and (iii) they do not use state-of-the-art methods to choose the size k of the reduced space, a key limitation.

in the last paragraph about complexity control, none of the papers showing that DR is worse use state-of-the-art methods to choose the size k of the resultant reduced space.

Beyond this mixed evidence, we find many papers that apply dimensionality reduction to their feature sets without explicitly mentioning a comparison to models built on the original feature sets. It is important to note that there are reasons not to just do DR automatically, even ignoring the effect on predictive performance. Computing SVD is very time consuming and computationally expensive. It also can obscure interpretability of the original features (latent factors are not guaranteed to be interpretable!). While some of these papers do give rationale for computing the SVD, it is not certain that they are building the best models that they can (and therefore obtaining the strongest results possible) by ignoring this comparison.

While these latter papers do not mention an empirical comparison between SVD and no-SVD feature sets, some do provide a rationale. The reasons are generally the same that we have encountered before: to reduce the noise in the data (Hu et al. 2007); to make the dimensionality of the space more tractable (Thorleuchter et al. 2012), (Coussement and Van den Poel 2008); or to reduce the computational expense of modeling (Tremblay et al. 2009). Sometimes DR is employed in the context of "side benefits" such as aiding in interpretation of a model (Khan et al. 2007) or speeding up prediction time (Lifshits and Nowotka 2007). Others do not provide either empirical or theoretical justification for applying SVD to their feature sets (Kosinski et al. 2013), (Cai et al. 2013), (Der et al. 2014), (Arulogun et al. 2012).

4 Experimental Setup

So far, we have shown that there is mixed evidence in the literature regarding the efficacy of dimensionality reduction for improving performance in predictive modeling. We have also shown that it is a reasonably popular data pre-processing technique, to the extent that numerous papers utilize it without any empirical justification. Here, we carry out an empirical comparison on one collection of predictive modeling problems based on sparse behavioral data, found in the aforementioned paper by Kosinski et al. The thrust of their research is that private traits *can* be predicted better than at random using Facebook Likes. Our intention here is not to critique these existing (strong) results, but to use the domain as a testbed of multiple, real predictive modeling

problems using the sort of sparse behavioral data that is increasingly seen in business analytics applications. In the process, we demonstrate that the results can be strengthened by not employing DR at all, as DR broadly tends to reduce predictive performance. Additionally, for research where results depend on prediction quality, the implication is that caution should be exercised when considering applying dimensionality reduction, since it may decrease predictive performance.

4.1 The Data

We recap the details of data collection from the Kosinski et al. paper. They collect data via a Facebook application that allows users to take personality tests and also makes the users' Likes and profile information available to the researchers. Thus, they can formulate a user-Like matrix A such that $A_{ij} = 1$ if user i Liked item j. They have generously made the resulting user-Like matrix as well as the personality test results and anonymized profile information available to other researchers.

In their paper, the user-Like matrix includes all users who have Liked at least 20 items and all Likes that have been Liked by at least 2 users, yielding $n = 58,466$ users and $d = 55,814$ Likes. The data have been incrementally updated since publication of their results. Therefore, our data have larger n and d than what is in their paper, specifically $n = 211,018$ users and $d = 179,605$ Likes. The corresponding matrix contains roughly 35 million unique user-Like pairs, resulting in a matrix with 99.91% sparsity (meaning .09% of the entries in the matrix are filled). The average user Liked 195 items and the average Like was Liked 166 times.[14]

We replicate the authors' predictions for all of the binary target variables. These include "single vs. in relationship", "parents together at 21", "smokes cigarettes", "drinks alcohol", "uses drugs", "Caucasian vs. African American", "Christianity vs. Islam", "Democrat vs. Republican", "gay", "lesbian", and "gender". Counts for the number of labeled users and Likes for each binary target can be found in Appendix A, Table 1. We followed the labeling procedure undertaken by Kosinski et al. for all target variables except "Caucasian vs. African American," which was labeled by visual inspection of profile pictures which were not available to us; therefore we used values from a different survey item.

Additionally, we included predictions for the numerical variables that they report. In order to provide a consistent comparison, we binarized all variables by setting a threshold, above which

[14]We store this resulting matrix A as a SciPy sparse matrix (Jones et al. 2001–) in Python.

the target was set to 1, and 0 otherwise. These numerical variables are "satisfaction with life", "intelligence", "emotional stability", "agreeableness", "extraversion", "conscientiousness", "openness", "density of friendship network", "number of Facebook friends", and "age". Some of the thresholds for binarization were set based on an intuitive value, such as age ≥ 30; others were set to be the median or top-quartile value for that variable to ensure some variety in base rates. More information on the various numerical variables and their thresholds can be found in Appendix A, Table 2.

4.2 Dimensionality Reduction

Kosinski et al. reduce the dimensionality of the user-Like matrix using SVD. They state that they utilize the top $k = 100$ components as features for most of the predictive models and $k = 30$ for some of the variables for which there is less data. This selection is based on an operational criterion: visual inspection of predictive accuracy vs. number of components for a few target variables flattens out around 100. Since we have more labeled instances than they did we use $k = 100$ as our default choice for k in our predictive models. We perform additional experiments that go one step farther and treat k as a model parameter to be selected using held-out data; see Section 4.3 for more details.

We compute the SVD of the user-Like matrix using SciPy's sparse SVD package for Python. There are two slightly different options for this computation. Only a subset of the set of users have labels for any task. It is possible to filter A separately for labeled users for each target variable, then compute the SVD of the resulting sub-matrix of A. The other option is to compute the SVD of the entire A matrix, then filter the resulting factors for each target variable's labeled values. We use the second option, as it computes the reduced space using the maximum amount of data.

While the authors do not state whether they compute SVD transductively or inductively (Zelikovitz and Marquez 2005), they do say that "SVD components were used to build models that predict users' individual traits and preferences." The implication is that they directly used the components as inputs in their regressions, and so that is what we also do for our experiments.

4.3 Prediction and Evaluation

Kosinski et al. use logistic regression with 10-fold cross-validation to predict the binary target variables, using the top $k = 100$ SVD components as inputs. They report the area under the ROC curve (AUC) as their measure of predictive performance, which is equivalent to the probability that

their model will rank any positive instance higher than any negative instance. To replicate their results, we do the same, training and evaluating our models using the Logistic Regression package in Python's scikit-learn (Pedregosa et al. 2011). We also train and evaluate models built on the original user-Like matrix A. To benchmark our results, we compared our SVD model results to the Kosinski et al. results. This comparison can be found in Table 1. The AUC's don't match up exactly, which is to be expected due to the updated data; however, they are very close. Since the goal of these experiments was to do a systematic comparison of SVD features to unreduced features, minor deviations from results presented in the original paper should not matter.

Training logistic regression (and other types of models) with no regularization can result in overfitting (Provost and Fawcett 2013), and—importantly for this study—the degree of regularization can greatly impact predictive performance (Bishop et al. 2006). In practice, training a regularlized logistic regression model involves finding the set of weights $\mathbf{w}$ that minimizes:[15]

$$C \sum_{i=1}^{n} \log\left(1 + \exp\left(-y_i \mathbf{w}^T \mathbf{x}_i\right)\right) + ||\mathbf{w}|| \tag{3}$$

The second term in equation (3) is the regularization term: it penalizes the magnitude of a norm of the weight vector. Either the L_1 or L_2 norm may be used. These two types of regularization are known as L_1 and L_2 regularization, respectively. C is the regularization parameter, which controls the trade-off between minimizing the logistic loss and the amount of regularization. Thus, higher values of C imply less regularization and lower values of C yield more regularization.

The default setting in scikit-learn is L_2 regularization with $C = 1$. Since Kosinski et al. do not discuss regularization in their paper, we use this default value to validate the performance of our SVD models and for our first set of experiments. Again, since our experimental results match those of Kosinski et al., this seems to be a reasonable choice.

Careful regularization can dramatically change the results of a regression; therefore, we also experimented with using a state-of-the-art procedures for selecting the optimal type of regularization and parameter. Specifically, we performed nested cross-validation paired with a grid search to

[15] Equation 3 is often written as

$$\sum_{i=1}^{n} \log\left(1 + \exp\left(-y_i \mathbf{w}^T \mathbf{x}_i\right)\right) + \lambda ||\mathbf{w}||,$$

where λ is the regularization parameter; however, here we adopt the convention taken by scikit-learn.

properly optimize and evaluate the effect of regularization on the performance. Nested cross-validation adds an inner loop of cross-validation to each of the folds of standard cross-validation. In the inner-loop we do 3-fold cross-validation to estimate the performance of each parameter combination in our grid. We try both L_1 and L_2 regularization, with C chosen from $\{.01, .1, 1, 10\}$. In the outer loop, a new model is trained using the parameter combination that yielded the highest AUC from the inner loop and then is tested on the test data. Thus, the parameter selection is part of the training process and the test data are not compromised (Provost and Fawcett 2013).

The research cited in Section 2.2 emphasizes the importance of choosing k (the number of SVD components to utilize as features in predictive modeling) carefully in order to maximize predictive accuracy. Common methods of doing so are: using an operational criterion such as predictive accuracy, visual inspection of the relative magnitudes of the singular values, using cross-validation to minimize reconstruction error on held-out data, and selecting a threshold for the amount of variance in the original data to be accounted for. As we discuss in Section 3.3, SVD is commonly used in predictive modeling applications. Nearly all of these papers choose some k; however, none of them appear to have treated k as a modeling parameter and used nested cross-validation for selection.

Because of k's importance to the success of SVD, we add two further sets of experiments. In order to avoid the problem of multiple comparisons and treat the choice of k as part of the training process, we (A) use nested-cross validation to select k from $\{2, 10, 25, 50, 100, 200, 500, 1000, 1500, 2000\}$ within each fold, while holding the regularization parameters constant to the default setting. Then, for completeness, we (B) select k and the regularization parameters in a huge round of nested cross-validation, which performs the full grid search over all combinations of regularization types, regularization parameters, and values for k and selects the best combination based on three-fold cross validation within each of the ten outer folds.

5 Experimental Results

This section shows the results of experiments on the suite of Facebook user-Like prediction problems described in Section 4.1, using the methodology and settings from Sections 4.2 and 4.3. We compare SVD-feature predictive performance versus full-feature predictive performance on the 21 predictive tasks with varied settings: default regularization (hyper)parameters, regularization

parameters chosen using nested cross-validation, k chosen using nested cross-validation, and k plus the regularization parameters chosen using nested cross-validation.

Using SVD does not demonstrate universally superior performance; in fact, the conditions under which it shows an improvement over the full-feature performance are narrowly constrained. The results in this section should be useful to both researchers and practitioners as they demonstrate that undertaking such a comparison is necessary to achieve optimal predictive performance, and that parameter choice can have dramatic effect when using SVD for predictive modeling.

5.1 Default Regularization and Fixed k

The first set of experiments compare performance using scikit-learn's default regularization settings for logistic regression. Kosinski et al. do not discuss regularization in their paper; here, performance on the $k = 100$ SVD feature set using the default regularization is roughly comparable to theirs. To illustrate the overall extent to which SVD affects predictive performance, Figure 1 plots the full-feature performance on the x-axis and the SVD performance on the y-axis. Each point represents one prediction problem's AUC pair. For clarity, a $y = x$ line is also on the plot. Points above the line are cases where SVD improves performance; points below are where SVD decreases performance.

The first thing to notice is that SVD does indeed improve performance for many of the targets. The Wilcoxon signed rank test (Wilcoxon et al. 1963) tests for the null hypothesis that pairs of data points drawn from different populations have a median difference of zero. For these pairs of points, the median difference is .98% and the resulting z score from applying the test is .88, which means that we fail to reject the null hypothesis at $p = .01, .05.$, and .1. This implies that overall, SVD neither helps nor hurts predictive accuracy. However, also note that there appear to be several regimes within the distribution. Instances toward the far top-right of the plot are actually hurt more than helped by SVD and vice-versa for instances at the bottom-left.

Delving more deeply into these results, notice that SVD tends to decrease performance for "easier" problems, that is, those for which the AUC using either feature set is relatively higher. The results in Figure 2 bear this idea out. Figure 2 plots the difficulty of the predictive problem versus the degree to which SVD improves predictive accuracy (SVD AUC - full-feature AUC). There is a strong linear relationship: a regression slope test between these two variables yields a slope of $-.23$, $t = -5.71$, $p < .01$. This implies that for easier problems, caution should be exercised when applying

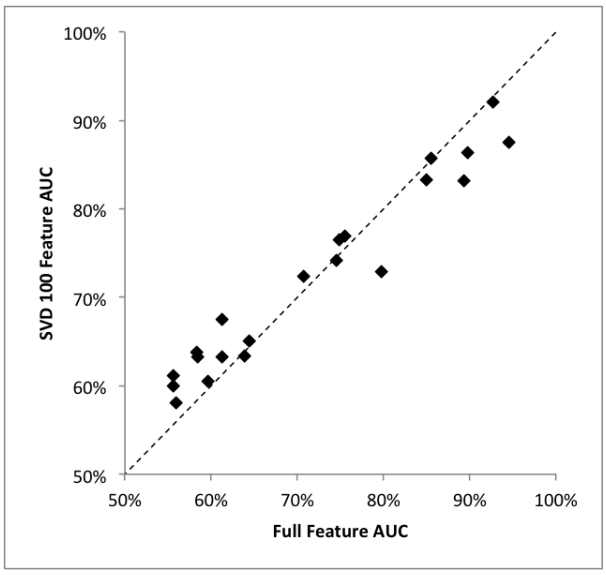

Figure 1: Comparing SVD versus full feature set using logistic regression with default regularization settings. SVD neither helps nor hurts in general. The benefit of SVD seems to relate to the difficulty of the predictive modeling problem.

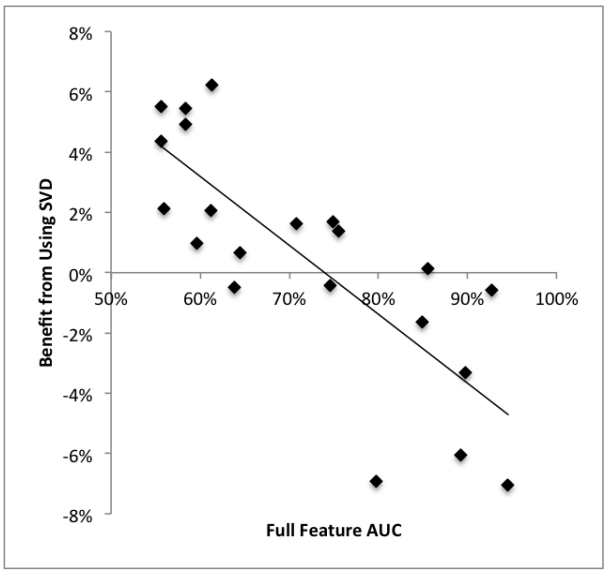

Figure 2: The difficulty of the original predictive problem strongly relates to the amount of benefit from using SVD. When using default regularization, easier predictive problems (where AUC on the full-feature task is high) have diminished predictive performance from using SVD; harder predictive problems have positive benefit.

SVD to the feature set, as it may decrease performance; however, for more difficult problems, SVD can improve predictive accuracy when using default regularization settings.

5.2 Problem-specific Regularization

While the prior result is itself useful, it draws to mind the intuition that SVD is implicitly performing regularization as Shmueli and Koppius noted; that it can"reduce sampling variance (even at the cost of increasing bias), and in turn increase predictive accuracy" (Shmueli and Koppius 2011). If one wants better "regularization," it makes sense to move beyond the default settings for the predictive modeling. Thus, let us now repeat the experiments utilizing problem-specific regularization. As mentioned i9n Section 4.3, logistic regression's performance can be dramatically affected by the choice of type of regularization and regularization parameter. The state-of-the-art (S.O.T.A.) practice for selecting these parameters is nested cross-validation.

Figure 3 plots the full-feature performance vs. the SVD-feature performance; however, this time all models were built using nested cross-validation to select parameters and evaluate the results.

The figure shows that using S.O.T.A. predictive modeling, there is no clear benefit to using SVD at all. In fact, SVD decreases predictive performance in the majority of cases! Applying the Wilcoxon signed-rank test, we obtain that the median difference is $-.67\%$, $z = -2.98$, $p < .01$. Thus, we reject the null hypothesis—SVD does indeed hurt predictive performance.

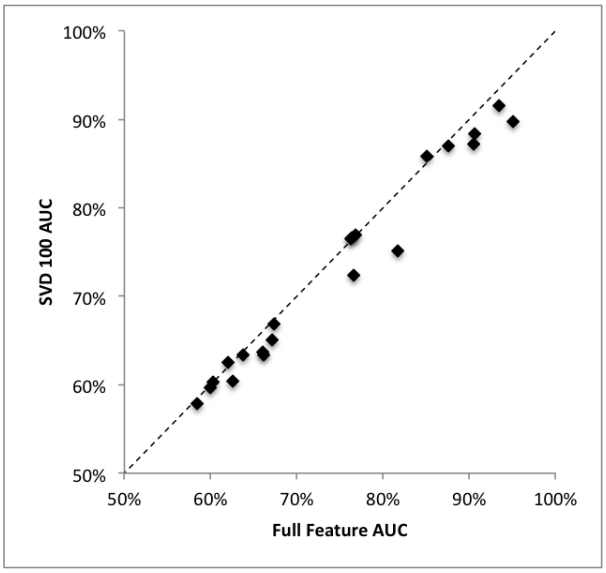

Figure 3: Comparing SVD versus full feature set with regularization settings selected through nested cross-validation. When using task-specific regularization settings, SVD at best gives little advantage and in many cases significantly diminishes predictive performance. This result applies across all problems, regardless of the difficulty of the predictive task.

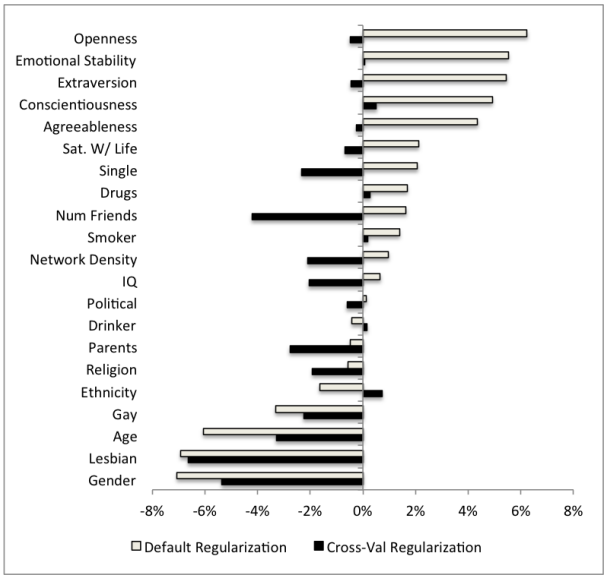

Figure 4: Summarizing benefit from SVD using default versus nested cross-validated regularization. Note that the benefit in predictive quality due to using SVD either vanishes or becomes negative in nearly every case.

Additionally, there is no longer a clear pattern as to which target variables' predictions will be improved or weakened by using SVD. Harder predictive problems do not necessarily benefit more from SVD when regularization parameters are selected using S.O.T.A. methods. To summarize the impact that proper regularization has on the directionality of the comparison, see Figure 4, which shows the difference between SVD performance and full-feature performance for each task. Notice how the directionality of many results changes dramatically with careful regularization. Even when it doesn't, the magnitude of the benefit from using SVD is dramatically reduced.

These results provide additional context around the findings from Section 5.1. SVD is itself a form of regularization. The easier predictive problems likely needed less regularization to perform

well. For the harder predictive problems, providing stronger regularization directly through the logistic regression objective function makes up for this need, thereby removing the advantage that was observed for SVD on these tasks.

5.3 Problem-specific Number of SVD Components

As is discussed above, correctly choosing k (the number of SVD components used as features in predictive modeling) is crucial. Given the near-ubiquity of mentions of the importance of this parameter, one might justifiably wonder whether the results in the last two subsections were simply due to a non-optimal choice for k. Thus, this section reports on a set of experiments using nested cross-validation to select k. Here we hold the type of regularization and C constant at their default values, and as in Section 5.1 compare to logistic regression with default-regularization. (Thus, these results should be contrasted with those in Section 5.1.)

Figure 5 again plots the full-feature performance versus the SVD-feature performance, using nested cross-validation to select k within every fold for every predictive task instead of $k = 100$ for all SVD tests (as above). As would be expected, the results for SVD are slightly better this time—that is, in most cases, treating k as a hyperparameter and selecting it carefully yields better generalization performance.

Also note that there still appear to be two regimes within the tasks shown in Figure 5. Once again, more difficult predictive problems appear to be more likely to be improved with use of SVD, while the opposite is true of easier predictive problems (those for which the full-feature AUC is high). This idea is again borne out by plotting the benefit from using SVD (SVD AUC - full-feature AUC), as in Figure 6, which exhibits a strong linear relationship between difficulty of the predictive problem and amount of SVD benefit. Even when choosing k carefully, using state-of-the-art predictive modeling techniques, SVD does not necessarily help with predictive performance, especially on easier tasks. Of course, these results—while illuminating—are subject to the symmetric criticism that the regularization parameters were not chosen well.

5.4 Problem-specific Grid Search

Finally, to do a full fair comparison of SVD versus full-feature performance, the experiments in this subsection compare predictive performance using SVD with both regularization parameters and k

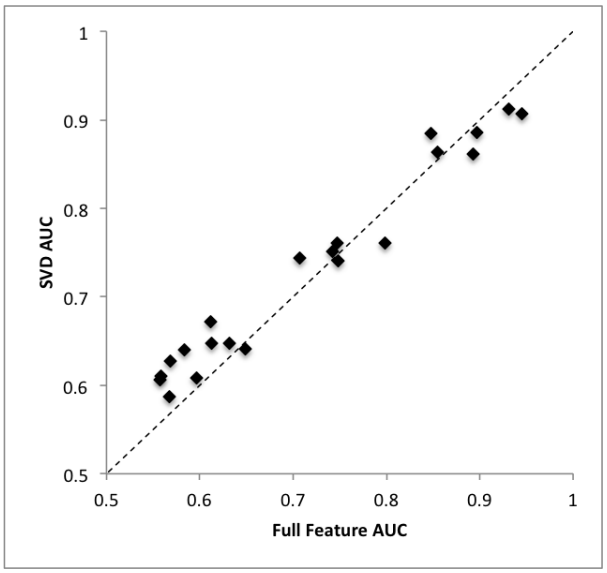

Figure 5: Comparing SVD versus full feature set using default regularization settings and k selected using nested cross-validation. SVD neither helps nor hurts in general (although here appears to perform slightly better relative to full-feature performance than in Figure 1). Again, the benefit of SVD seems to relate to the difficulty of the predictive modeling problem.

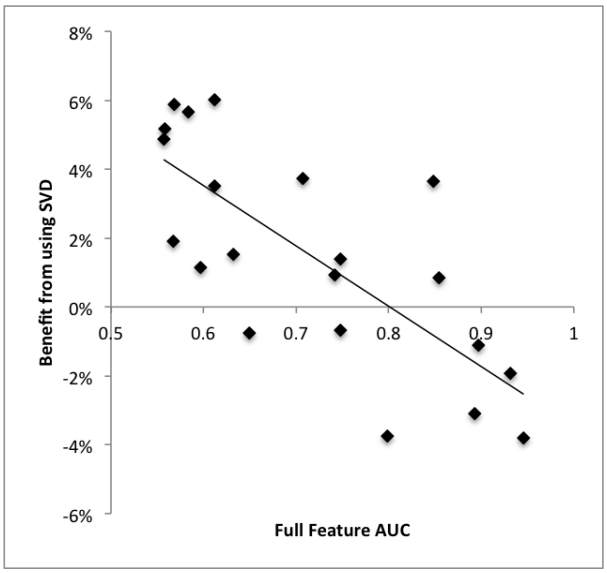

Figure 6: Relating the difficulty of the original predictive problem to the amount of benefit from using SVD, given task-specific choice of k. Again, harder predictive problems here have positive benefit from using SVD; this is not necessarily the case for easier problems.

chosen using nested cross-validation over a complete grid search covering every combination of type of regularization, C, and k (80 total possible combinations).

Figure 7 summarizes the improvements due to SVD under three sets of conditions: optimizing over k, optimizing over regularization settings, or optimizing over both (full grid search). Qualitatively, it appears that optimizing over both sets of parameters at the same time yields the best improvements over default settings in the plurality of cases. Selecting k in isolation helps somewhat more than selecting regularization parameters in isolation; however, doing either one is better than doing neither.

Figure 8 plots SVD AUC with S.O.T.A. selection of regularization and k versus well-regularized full-feature AUC. As in Section 5.2, using SVD decreases predictive performance on the majority of tasks, even when optimizing over values of k. The Wilcoxon signed-rank test yields that the median difference is $-.90\%$, $z = -2.46$, $p = .014$. Thus, we reject the null hypothesis—SVD indeed hurts predictive performance in the context of thorough regularization, even when k is selected carefully.

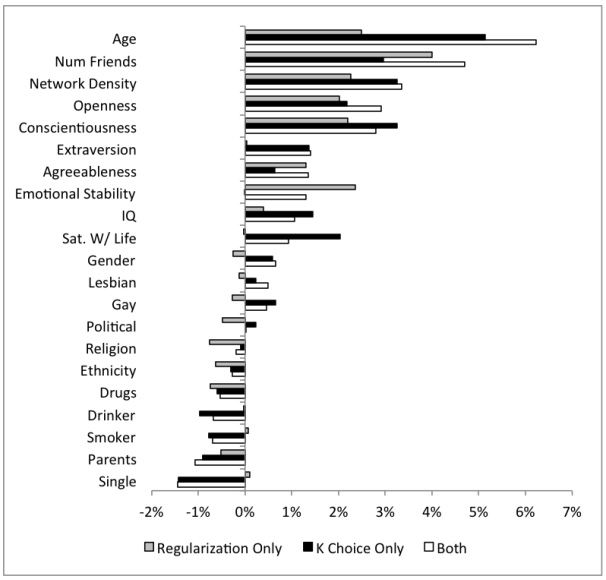

Figure 7: Improvement over default settings obtained by optimizing over regularization parameters, over k, and over both sets of parameters by predictive task. Note that in many cases optimizing over both yields the best improvements, but this is not uniformly the case.

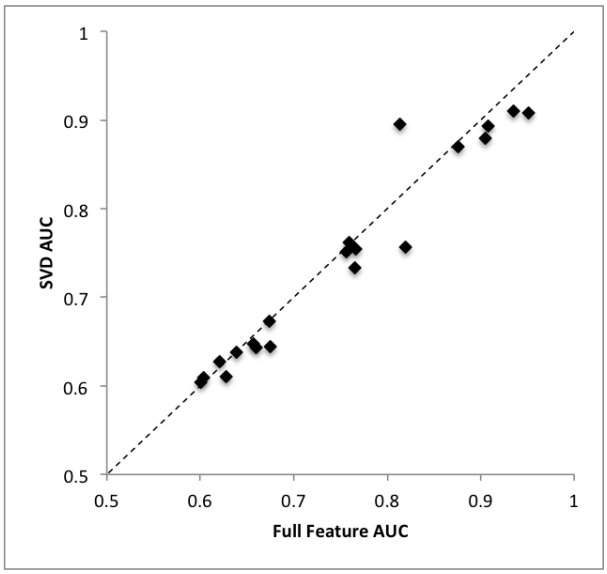

Figure 8: Comparing predictive performance using SVD versus full feature set with regularization settings (as well as k for the SVD models) selected through nested cross-validation. Again, when using task-specific regularization settings rather than default, with only one exception SVD at best gives little advantage and in many cases substantially diminishes predictive performance. This result applies across all problems, regardless of the difficulty of the predictive task.

6 Discussion and Implications

This research results in four main implications stemming from the survey of DR in predictive modeling and related domains and the experimental evaluation. The implications all apply to both researchers and practitioners (people who want to do predictive modeling "well" in practice). All of these findings reflect the importance of being clear about *why* one is using dimensionality reduction.

First, we clarify dimensionality reduction's place in the set of information systems artifacts currently called "predictive analytics." Second, we find that in evaluating predictive systems that incorporate DR, existing research often does not perform lesion studies (Langley 2000) to assess whether this non-trivial component (the DR) actually is improving predictive performance. This is particularly concerning in light of the experimental evaluation that we present, which demonstrates that DR can significantly *reduce* predictive performance. Third, looking more deeply, the arguments for using DR to improve predictive accuracy are identical to those for using

regularization. However, regularization is a more direct form of complexity control, in that it is taken into account explicitly in the optimization of the modeling objective function. Strikingly, based on our survey, research using DR for engineering features in predictive modeling never uses the DR features in the context of regularization. Our empirical results demonstrate that doing so is necessary, as the direct regularization (i) provides better complexity control overall, (ii) in almost all cases obviates the improvements in predictive performance achieved via DR, and (iii) improves the predictive performance of systems that use DR—which is important if DR is being used for a reason other than optimizing predictive performance. Finally, our findings reflect on existing research that utilizes DR for predictive modeling. Let's briefly elaborate on these implications.

One of the tenets of design science research is that design artifacts should not have superfluous components (Hevner et al. 2004), (Langley 2000). Our survey and experiments analyze dimensionality reduction's place in the predictive modeling process artifact. The survey reveals that many research papers incorporate DR in their process without either presenting a theoretical rationale for doing so or conducting an empirical evaluation of the value of doing so. A non-trivial component such as DR should have a demonstrably positive effect on the efficacy of conducting predictive modeling. If it does not, then it it is unnecessary and should not be included as part of the design artifact.

Crucially, this emphasizes that researchers and practitioners must be clear as to what purpose they are employing DR. SVD and related techniques have many side benefits such as increased tractability (smaller feature space) and potentially sped-up prediction time. While these are sometimes cited as the reasons why researchers have used DR, the survey in Section 3 purposely drew on papers for which good predictive (generalization) performance was the end goal. Without any theoretical justification or comparison, these papers risk presenting to the reader an unnecessarily complicated predictive modeling system. The result may be that subsequent researchers and practictioners act based on prior published results, without having considered that the prior results may present systems with superfluous components. The subsequent work then is subject to similar criticism. The experimental results from Section 5 show that, even without the state-of-the-art setting of hyperparameters, the SVD features only improve performance on a limited set of tasks (those which were harder to begin with using the full feature set), and they tended to result in decreased performance on the easier problems. This would suggest a possible revision to Shmueli and Koppius's process (Shmueli and Koppius 2011): removing DR as a component.

The survey of DR's use in other fields such as information retrieval and collaborative filtering reveals generally positive results. Such results may be one reason why researchers tend to utilize DR in predictive modeling, especially on tasks where the data appear similar to IR and CF data: large, sparse, and drawn from a behavioral domain. However, while the data on such predictive modeling tasks may be structurally similar to that in IR and CF, the predictive set-up is different. Researchers and practitioners should be aware of both the potential tradeoff in performance when utilizing DR to pursue an alternate goal and the difference between standard predictive modeling and other domains. One should not assume that DR will work: the comparison between DR and full feature sets should be done. Our survey reveals that this key step is frequently not done, and our experiments reveal that it is necessary. Under standard conditions (with default regularization settings), we see that predictive performance may decrease by using SVD, especially on easier tasks.

Moreover, the arguments for using DR to improve predictive accuracy are essentially the same as those for using regularization in predictive modeling. Reducing the dimensionality of the feature space reduces the variance of the modeling at the cost of introducing some predictive bias. Ditto for regularization. Trading off variance for bias potentially can improve overall predictive accuracy (but it also can hurt, of course). Thus, since DR and regularization seek to accomplish the same goal, why not simply perform careful, direct regularization instead of using DR? Further, why not do both? In the survey, we found that not only do researchers never compare DR to a full feature set which has had regularization settings carefully chosen, they also never consider DR in the context of careful regularization—that is, using nested cross-validation or some other nested holdout design to select regularization settings. The experiments from this paper show that in general, careful regularization shows a bigger improvement in predictive performance than using DR. Even when careful regularization is used for both feature sets (DR features and full features), the full feature set performs statistically significantly better. However, the results also show that if DR were to be used for another motivating reason, careful regularization still helps.

Relatedly, k, the number of SVD components that are used as features, must be carefully chosen in order to maximize predictive performance. This fact is stated in nearly every paper that utilizes SVD or some other form of DR; however, in our survey we find no instance where k is chosen using state-of-the-art modeling techniques such as nested cross-validation. Rather, k is either chosen using some heuristic or selected by building models trying different values for k and measuring

performance on some held-out data—and then choosing k on the hold-out data! This embeds a serious multiple comparisons problem (Jensen and Cohen 2000) and thus any associated results cannot be relied upon due to the potential for unchecked overfitting. Our experiments show that carefully selecting either or both of these parameters (k and regularization settings) using nested cross-validation can make a big difference in terms of both predictive performance and in selecting which feature set works better.

This paper's research most closely resembles the aforementioned van der Maaten et al. survey of dimensionality reduction techniques, but our implications also apply to any research or project that uses dimensionality reduction. We show that the superior performance of "no DR" is *not* simply due to the incorrect choice of k. Additionally, unlike any study of DR before, we evaluate and compare in the context of careful regularization. Our results show that any work demonstrating the superiority of a particular method of DR may be questionable if it is not done in the context of regularization. Also, the reader should question whether any paper that selects a particular value for k has done the selection appropriately (i.e., in a principled fashion but not based on performance on the holdout data). While many existing results could be made stronger, many may in fact change—especially any results that purport to show the superiority of one method of DR.

There are some important limitations to our research. The scope of one paper is not able to include all of the different things that may affect performance. We did not examine some options: the way in which the dimensions may be used (choice of model); type of dimensionality reduction (focusing on standard linear DR through SVD, which is the most popular and was found to outperform other methods in (van der Maaten et al. 2009)); or the various ways to use DR (factors-as-features vs. matrix reconstruction, as is used in collaborative filtering). Additionally, we limited ourselves to one (albeit large) collection of data sets to compare DR versus no-DR. This was primarily because we sought to replicate and hopefully improve results that have been published by top-notch researchers in a very prestigious venue, but also because this enabled us to do a comparison across predictive modeling problems which all used the same original features (one of the key types of applications for which DR provides value).

7 Conclusion

This paper thoroughly explores dimensionality reduction as design artifact in predictive modeling. We have satisfied the seven guidelines for design science research from the aforementioned Hevner et al. paper by establishing DR as an artifact; demonstrating the relevance of this artifact to business problems; evaluating it in the context of its effect on predictive accuracy; contributing novel knowledge of the conditions under which DR performs well; using rigor in our exploration and evaluation of DR; performing a thorough search of the space of design possibilities that may affect DR's performance; and communicating the technical and managerial implications of our results.

We show that it is vital to be clear regarding the *purpose* for using DR, whether it is to improve predictive accuracy, to generate features that are tractable for use in an econometric model, to understand the latent structure in the data, or to reduce computational overhead. We also demonstrate the necessity of understanding the domain of the problem and the use scenario for the resulting predictive model. Confusing the domain of the predictive problem could lead to confusion regarding how and why to apply DR. Our results regarding DR's performance given default regularization settings as well as carefully selected regularization settings give clear and actionable guidelines for when to use DR in a predictive modeling problem.

This research yields some clear directions for future work. Primarily, we would like to more thoroughly explore the statistical relationship between regularization and dimensionality reduction. Theoretically, the two methods are accomplishing the same goal: exploiting a tradeoff between error due to bias and variance in order to diminish overfitting models. Positioning DR in this light reveals it as one more tool in the data scientist's arsenal for improving model performance, and it may be possible to prove theoretical results regarding situations in which DR is useful, or how SVD relates to L_1 and L_2 regularization. Additionally, various DR techniques have been developed to themselves avoid overfitting, such as sparse PCA and regularized SVD; however, in practice these methods are not used as commonly as straightforward SVD. An exploration of these methods in the context of regularized predictive models would also be beneficial.

It is our hope that this research will inspire researchers and practitioners in IS and beyond to be careful in their use of SVD or other forms of dimensionality reduction for predictive modeling in research or business problems. In particular, we were surprised at the degree to which utilizing

Task	K's n	Our n	Our d	Base Rate	K's AUC	Our AUC
Single vs. In Relationship	46,027	162,980	179,605	47%	.67	.63
Parents together at 21	766	2,088	84,813	50%	.6	.63
Smokes Cigarettes	1,211	3,690	118,643	25%	.73	.77
Drinks Alcohol	1,196	3,667	118,604	51%	.7	.74
Uses drugs	856	2,711	104,869	17%	.65	.77
Caucasian vs. African American	7,000	2,645	100,506	95%	.95	.83
Christianity vs. Islam	18,833	3,625	105,023	95%	.82	.92
Democrat vs. Republican	9,752	12,936	147,759	59%	.85	.86
Gay		25,813	167,307	5%	.88	.86
Lesbian		30,087	173,375	3%	.75	.73
Gender	57,505	210,004	179,605	61%	.93	.87

Table 1: Data details for binary target variables

nested cross-validation for model parameter selection changed the directionality of the results. This further emphasizes the vast range of decisions that can be made when doing predictive modeling in order to improve accuracy. While it is never possible to do a full grid search over every possibility, our results here help to illuminate the consequences of one set of choices.

A Tables

Table 1 shows the amount of data available at the time of publication of Kosinski et al. (2013) versus the number of users and Likes currently available for each binary target variable and the base rates for each of these variables. We also include the AUC obtained by Kosinski et al. versus the AUC we obtained when running our experiments with default regularization and k settings.

Table 2 shows the number of users and Likes for which there is labeled data for each numerical target variable, the thresholds for denoting positive versus negative instances, and the base rates.

References

Dimitris Achlioptas. Database-friendly random projections. In *Proceedings of the twentieth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 274–281. ACM, 2001.

Gediminas Adomavicius and Alexander Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *Knowledge and Data Engineering, IEEE Transactions on*, 17(6):734–749, 2005.

Target	n	d	Threshold	Base Rate
Satisfaction with Life	6,512	141,815	5	37%
Intelligence	6,129	137,558	115	49%
Emotional Stability	173,109	179,443	2.75	54%
Agreeableness	173,109	179,443	3.6	50%
Extraversion	173,109	179,443	3.56	50%
Conscientiousness	173,109	179,443	3.5	52%
Openness	173,109	179,443	4	51%
Network Density	46,265	178,914	.05	25%
Number of Friends	171,789	179,402	334	25%
Age	185,692	179,605	30	24%

Table 2: Data details for numeric target variables

Hyunchul Ahn, Eunsup Choi, and Ingoo Han. Extracting underlying meaningful features and canceling noise using independent component analysis for direct marketing. *Expert Systems with Applications*, 33(1): 181–191, 2007.

Anestis Antoniadis, Sophie Lambert-Lacroix, and Frédérique Leblanc. Effective dimension reduction methods for tumor classification using gene expression data. *Bioinformatics*, 19(5):563–570, 2003.

OT Arulogun, EO Omidiora, MA Waheed, OA Fakolujo, OM Olaniyi, et al. On the classification of gasoline-fuelled engine exhaust fume related faults using electronic nose and principal component analysis. *Computing, Information Systems, Development Informatics and Allied Research Journal*, 3(2), 2012.

Peter N. Belhumeur, João P Hespanha, and David Kriegman. Eigenfaces vs. fisherfaces: Recognition using class specific linear projection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 19 (7):711–720, 1997.

Robert M Bell, Yehuda Koren, and Chris Volinsky. The bellkor solution to the netflix prize, 2007.

Richard Bellman. *Adaptive control processes: a guided tour*, volume 4. Princeton university press Princeton, 1961.

Daniel Billsus and Michael J Pazzani. Learning collaborative information filters. In *ICML*, volume 98, pages 46–54, 1998.

Christopher M Bishop et al. *Pattern recognition and machine learning*, volume 1. springer New York, 2006.

David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. *the Journal of machine Learning research*, 3:993–1022, 2003.

Jesús Bobadilla, Fernando Ortega, Antonio Hernando, and Abraham Gutiérrez. Recommender systems survey. *Knowledge-Based Systems*, 46:109–132, 2013.

MC Burl, UM Fayyad, P Perona, P Smyth, and MP Burl. Automating the hunt for volcanoes on venus. In

Computer Vision and Pattern Recognition, 1994. Proceedings CVPR'94., 1994 IEEE Computer Society Conference on, pages 302–309. IEEE, 1994.

Fidel Cacheda, Víctor Carneiro, Diego Fernández, and Vreixo Formoso. Comparison of collaborative filtering algorithms: Limitations of current techniques and proposals for scalable, high-performance recommender systems. *ACM Transactions on the Web (TWEB)*, 5(1):2, 2011.

Jinye Cai, Pingping Xu, Huiyu Tang, and Lin Sun. An improved selective ensemble method for spam filtering. In *Communication Technology (ICCT), 2013 15th IEEE International Conference on*, pages 743–747. IEEE, 2013.

Kristof Coussement and Dirk Van den Poel. Integrating the voice of customers through call center emails into a decision support system for churn prediction. *Information & Management*, 45(3):164–174, 2008.

Sampath Deegalla and Henrik Bostrom. Reducing high-dimensional data by principal component analysis vs. random projection for nearest neighbor classification. In *Machine Learning and Applications, 2006. ICMLA'06. 5th International Conference on*, pages 245–250. IEEE, 2006.

Scott C. Deerwester, Susan T Dumais, Thomas K. Landauer, George W. Furnas, and Richard A. Harshman. Indexing by latent semantic analysis. *JASIS*, 41(6):391–407, 1990.

Matthew F Der, Lawrence K Saul, Stefan Savage, and Geoffrey M Voelker. Knock it off: profiling the online storefronts of counterfeit merchandise. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1759–1768. ACM, 2014.

Pedro Domingos. A few useful things to know about machine learning. *Communications of the ACM*, 55(10): 78–87, 2012.

Susan Dumais. Enhancing performance in latent semantic indexing (lsi) retrieval, 1992.

Dmitriy Fradkin and David Madigan. Experiments with random projections for machine learning. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 517–522. ACM, 2003.

Jerome H Friedman. On bias, variance, 0/1loss, and the curse-of-dimensionality. *Data mining and knowledge discovery*, 1(1):55–77, 1997.

Bjarne Ørum Fruergaard, Toke Jansen Hansen, and Lars Kai Hansen. Dimensionality reduction for click-through rate prediction: Dense versus sparse representation. *arXiv preprint arXiv:1311.6976*, 2013.

Aristides Gionis, Piotr Indyk, Rajeev Motwani, et al. Similarity search in high dimensions via hashing. In *VLDB*, volume 99, pages 518–529, 1999.

William H Greene. *Econometric analysis*. Pearson Education India, 2003.

Isabelle Guyon, Vincent Lemaire, Marc Boullé, Gideon Dror, and David Vogel. Analysis of the kdd cup 2009: Fast scoring on a large orange customer database. 2009.

Trevor Hastie, Robert Tibshirani, Jerome Friedman, T Hastie, J Friedman, and R Tibshirani. *The elements of statistical learning*, volume 2. Springer, 2009.

Alan R Hevner, Salvatore T March, Jinsoo Park, and Sudha Ram. Design science in information systems research. *MIS quarterly*, 28(1):75–105, 2004.

Peter D Hoff. Model averaging and dimension selection for the singular value decomposition. *Journal of the American Statistical Association*, 102(478), 2007.

Thomas Hofmann. Probabilistic latent semantic indexing. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 50–57. ACM, 1999.

Jian Hu, Hua-Jun Zeng, Hua Li, Cheng Niu, and Zheng Chen. Demographic prediction based on user's browsing behavior. In *Proceedings of the 16th international conference on World Wide Web*, pages 151–160. ACM, 2007.

Zan Huang, Daniel Zeng, and Hsinchun Chen. A comparison of collaborative-filtering recommendation algorithms for e-commerce. *IEEE Intelligent Systems*, 22(5):68–78, 2007.

Lawrence Hubert, Jacqueline Meulman, and Willem Heiser. Two purposes for matrix factorization: A historical appraisal. *SIAM review*, 42(1):68–82, 2000.

David D Jensen and Paul R Cohen. Multiple comparisons in induction algorithms. *Machine Learning*, 38(3): 309–338, 2000.

Eric Jones, Travis Oliphant, Pearu Peterson, et al. SciPy: Open source scientific tools for Python, 2001–. URL `http://www.scipy.org/`. [Online; accessed 2014-10-26].

Enric Junqué de Fortuny, David Martens, and Foster Provost. Predictive modeling with big data: Is bigger really better? *Big Data*, 1(4):215–226, 2013.

George Karypis and Eui-Hong Sam Han. Fast supervised dimensionality reduction algorithm with applications to document categorization & retrieval. In *Proceedings of the ninth international conference on Information and knowledge management*, pages 12–19. ACM, 2000.

Rehan M Khan, Chung-Hay Luk, Adeen Flinker, Amit Aggarwal, Hadas Lapid, Rafi Haddad, and Noam Sobel. Predicting odor pleasantness from odorant structure: pleasantness as a reflection of the physical world. *The Journal of Neuroscience*, 27(37):10015–10023, 2007.

Melody Y Kiang and Ajith Kumar. An evaluation of self-organizing map networks as a robust alternative to factor analysis in data mining applications. *Information Systems Research*, 12(2):177–194, 2001.

YongSeog Kim, W Nick Street, Gary J Russell, and Filippo Menczer. Customer targeting: A neural network approach guided by genetic algorithms. *Management Science*, 51(2):264–276, 2005.

Yehuda Koren. The bellkor solution to the netflix grand prize. *Netflix prize documentation*, 81, 2009.

Yehuda Koren. Factor in the neighbors: Scalable and accurate collaborative filtering. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 4(1):1, 2010.

Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009.

Michal Kosinski, David Stillwell, and Thore Graepel. Private traits and attributes are predictable from digital records of human behavior. *Proceedings of the National Academy of Sciences*, 110(15):5802–5805, 2013.

Aswani Ch Kumar. Analysis of unsupervised dimensionality reduction techniques. *Computer Science and Information Systems/ComSIS*, 6(2):217–227, 2009.

Pat Langley. Crafting papers on machine learning. In *ICML*, pages 1207–1216, 2000.

Yury Lifshits and Dirk Nowotka. Estimation of the click volume by large scale regression analysis. In *Computer Science–Theory and Applications*, pages 216–226. Springer, 2007.

Míriam López, Javier Ramírez, Juan Manuel Górriz, Ignacio Álvarez, Diego Salas-Gonzalez, Fermín Segovia, Rosa Chaves, Pablo Padilla, and Manuel Gómez-Río. Principal component analysis-based techniques and supervised classification schemes for the early detection of alzheimer's disease. *Neurocomputing*, 74 (8):1260–1271, 2011.

John Mandel. Use of the singular value decomposition in regression analysis. *The American Statistician*, 36 (1):15–24, 1982.

Christopher D Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to information retrieval*, volume 1. Cambridge university press Cambridge, 2008.

D Harrison McKnight, Vivek Choudhury, and Charles Kacmar. Developing and validating trust measures for e-commerce: an integrative typology. *Information systems research*, 13(3):334–359, 2002.

Abhay Nath Mishra, Prabhudev Konana, and Anitesh Barua. Antecedents and consequences of internet use in procurement: an empirical investigation of us manufacturing firms. *Information Systems Research*, 18(1):103–120, 2007.

Bamshad Mobasher, Xin Jin, and Yanzan Zhou. Semantically enhanced collaborative filtering on the web. In *Web Mining: From Web to Semantic Web*, pages 57–76. Springer, 2004.

Art B Owen and Patrick O Perry. Bi-cross-validation of the svd and the nonnegative matrix factorization. *The Annals of Applied Statistics*, pages 564–594, 2009.

Arkadiusz Paterek. Improving regularized singular value decomposition for collaborative filtering. In *Proceedings of KDD cup and workshop*, volume 2007, pages 5–8, 2007.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

Foster Provost and Tom Fawcett. *Data Science for Business: What you need to know about data mining and data-analytic thinking.* " O'Reilly Media, Inc.", 2013.

Troy Raeder, Claudia Perlich, Brian Dalessandro, Ori Stitelman, and Foster Provost. Scalable supervised dimensionality reduction using clustering. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1213–1221. ACM, 2013.

Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. Analysis of recommendation algorithms for e-commerce. In *Proceedings of the 2nd ACM conference on Electronic commerce*, pages 158–167. ACM, 2000a.

Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. Application of dimensionality reduction in recommender system-a case study. Technical report, DTIC Document, 2000b.

Louis L Scharf. The svd and reduced rank signal processing. *Signal processing*, 25(2):113–133, 1991.

Fabrizio Sebastiani. Machine learning in automated text categorization. *ACM computing surveys (CSUR)*, 34(1):1–47, 2002.

D Seung and L Lee. Algorithms for non-negative matrix factorization. *Advances in neural information processing systems*, 13:556–562, 2001.

Qinfeng Shi, James Petterson, Gideon Dror, John Langford, Alex Smola, and SVN Vishwanathan. Hash kernels for structured data. *The Journal of Machine Learning Research*, 10:2615–2637, 2009.

Galit Shmueli and Otto R Koppius. Predictive analytics in information systems research. *MIS Quarterly*, 35 (3):553–572, 2011.

Lawrence Sirovich and Michael Kirby. Low-dimensional procedure for the characterization of human faces. *JOSA A*, 4(3):519–524, 1987.

Abdulhamit Subasi and M Ismail Gursoy. Eeg signal classification using pca, ica, lda and support vector machines. *Expert Systems with Applications*, 37(12):8659–8666, 2010.

Xiaohua Sun, Fansheng Kong, and Song Ye. A comparison of several algorithms for collaborative filtering in startup stage. In *Networking, Sensing and Control, 2005. Proceedings. 2005 IEEE*, pages 25–28. IEEE, 2005.

Gábor Takács, István Pilászy, Bottyán Németh, and Domonkos Tikk. Scalable collaborative filtering approaches for large recommender systems. *The Journal of Machine Learning Research*, 10:623–656, 2009.

Dirk Thorleuchter, Dirk Van den Poel, and Anita Prinzie. Analyzing existing customers websites to improve the customer acquisition process as well as the profitability prediction in b-to-b marketing. *Expert systems with applications*, 39(3):2597–2605, 2012.

Monica Chiarini Tremblay, Donald J Berndt, Stephen L Luther, Philip R Foulis, and Dustin D French. Identifying fall-related injuries: Text mining the electronic medical record. *Information Technology and Management*, 10(4):253–265, 2009.

Matthew Turk and Alex Pentland. Eigenfaces for recognition. *Journal of cognitive neuroscience*, 3(1):71–86, 1991.

LJP van der Maaten, EO Postma, and HJ van Den Herik. Dimensionality reduction: A comparative review. *Journal of Machine Learning Research*, 10:1–41, 2009.

Manolis G Vozalis and Konstantinos G Margaritis. Applying svd on item-based filtering. In *Intelligent Systems Design and Applications, 2005. ISDA'05. Proceedings. 5th International Conference on*, pages 464–469. IEEE, 2005.

Manolis G Vozalis and Konstantinos G Margaritis. Using svd and demographic data for the enhancement of generalized collaborative filtering. *Information Sciences*, 177(15):3017–3037, 2007.

Kilian Weinberger, Anirban Dasgupta, John Langford, Alex Smola, and Josh Attenberg. Feature hashing for large scale multitask learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 1113–1120. ACM, 2009.

Mike West, Carrie Blanchette, Holly Dressman, Erich Huang, Seiichi Ishida, Rainer Spang, Harry Zuzan, John A Olson, Jeffrey R Marks, and Joseph R Nevins. Predicting the clinical status of human breast cancer by using gene expression profiles. *Proceedings of the National Academy of Sciences*, 98(20): 11462–11467, 2001.

Frank Westad, Margrethe Hersletha, Per Lea, and Harald Martens. Variable selection in pca in sensory descriptive and consumer data. *Food Quality and Preference*, 14(5):463–472, 2003.

Brian Whitman. Semantic rank reduction of music audio. In *Applications of Signal Processing to Audio and Acoustics, 2003 IEEE Workshop on.*, pages 135–138. IEEE, 2003.

Frank Wilcoxon, SK Katti, and Roberta A Wilcox. *Critical values and probability levels for the Wilcoxon rank sum test and the Wilcoxon signed rank test.* American Cyanamid Comp., 1963.

Xin Xu and Xuening Wang. An adaptive network intrusion detection method based on pca and support vector machines. In *Advanced Data Mining and Applications*, pages 696–703. Springer, 2005.

Yiming Yang. Noise reduction in a statistical approach to text categorization. In *Proceedings of the 18th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 256–263. ACM, 1995.

Sarah Zelikovitz and Finella Marquez. Transductive learning for short-text classification problems using latent semantic indexing. *International Journal of Pattern Recognition and Artificial Intelligence*, 19 (02):143–163, 2005.

Hui Zou, Trevor Hastie, and Robert Tibshirani. Sparse principal component analysis. *Journal of computational and graphical statistics*, 15(2):265–286, 2006.