Data Mining and Knowledge Discovery (2019) 33:871–916
https://doi.org/10.1007/s10618-019-00616-4

Unsupervised dimensionality reduction versus supervised regularization for classification from sparse data

Jessica Clark[1] · Foster Provost[2]

Received: 7 November 2017 / Accepted: 21 January 2019 / Published online: 20 February 2019

Abstract
Unsupervised matrix-factorization-based dimensionality reduction (DR) techniques are popularly used for feature engineering with the goal of improving the generalization performance of predictive models, especially with massive, sparse feature sets. Often DR is employed for the same purpose as supervised regularization and other forms of complexity control: exploiting a bias/variance tradeoff to mitigate overfitting. Contradicting this practice, there is consensus among existing expert guidelines that supervised regularization is a superior way to improve predictive performance. However, these guidelines are not always followed for this sort of data, and it is not unusual to find DR used with no comparison to modeling with the full feature set. Further, the existing literature does not take into account that DR and supervised regularization are often used in conjunction. We experimentally compare binary classification performance using DR features versus the original features under numerous conditions: using a total of 97 binary classification tasks, 6 classifiers, 3 DR techniques, and 4 evaluation metrics. Crucially, we also experiment using varied methodologies to tune and evaluate various key hyperparameters. We find a very clear, but nuanced result. Using state-of-the-art hyperparameter-selection methods, applying DR does not add value beyond supervised regularization, and can often diminish performance. However, if regularization is not done well (e.g., one just uses the default regularization parameter), DR does have relatively better performance—but these approaches result in lower performance overall. These latter results provide an explanation for why practitioners may be continuing to use DR without undertaking the necessary comparison to using the original features. However, this practice seems generally wrongheaded in light of the main results, if the goal is to maximize generalization performance.

Keywords Dimensionality reduction · Binary classification · Sparse data · Experimental comparison · Data mining

Responsible editor: Pauli Miettinen.

Extended author information available on the last page of the article

1 Introduction

Predictive modeling from sparse, binary data sets often uses unsupervised matrix-factorization-based (MF-based) dimensionality reduction (DR) as a preprocessing step. There seems to be confusion in the research, educational, and practitioner communities as to whether using DR in such a way is beneficial. This paper experimentally evaluates whether and under what conditions using unsupervised DR improves the generalization performance of binary classifiers trained using massive, sparse feature sets.

To the best of our knowledge, this is the first work that comprehensively evaluates whether or not DR is an advantageous preprocessing step to improve predictive modeling performance in the context of state-of-the-art complexity control techniques. Past work in this area has not considered the extent to which different model selection methodologies may affect DR's performance relative to the original feature set. Furthermore, past work has also not conducted experiments using massive, sparse data—where DR is commonly applied. The intended audience for this research is not necessarily researchers developing new machine learning algorithms (although our findings are certainly relevant to that community), but more broadly, anyone who leverages predictive modeling for their research or work.

Our main contribution is an experimental comparison of models trained using reduced-dimensionality features obtained using unsupervised MF-based DR versus using the full (unreduced) feature set. The experiments in this study primarily consist of a deep dive into one predictive test bed, comprising 21 binary classification tasks on the same high-dimensional, sparse feature set. Focusing on one feature set and related tasks allows us to control for possible variations in the selection of the target task. As a generality check, we follow up with a set of experiments on 15 additional data sets, comprising a total of 76 additional classification tasks. While our results mainly focus on binary classification, we touch on multiclass classification and numerical regression and see intriguing preliminary results.

The experiments vary the methodology used to select and evaluate key complexity parameters. Namely, to compare DR-feature versus full-feature performance for each set of experiments we use either (a) nested cross-validation to select from a set of possible parameter values or (b) standard cross-validation with a default parameter value. Furthermore, the experiments use three unsupervised matrix-factorization-based DR methods (Singular Value Decomposition, Non-negative Matrix Factorization, and Latent Dirichlet Allocation), six classification algorithms (L_2-regularized logistic regression, Support Vector Machines, L_1-regularized logistic regression, Classification Trees, Random Forests, and k-Nearest Neighbors), and are compared using four different evaluation metrics: AUC, H-Measure, AUCH, and the lift over random selection at the top 5% of test instances.

Via our literature review, we establish that DR is commonly used in applied data mining contexts, but seems to be somewhat misunderstood, and therefore can be used incorrectly. Specifically, modelers often fail to undertake an experimental comparison of modeling using dimensionality-reduced features versus the original full feature set. Furthermore, existing guidelines (as encoded in data mining and machine learning textbooks and courses) treat DR and supervised regularization as competing methods;

however, these two methods for mitigating overfitting could be (and often are) used in conjunction.

As evidence of the lack of clarity on this issue, consider the influential 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 employs the process this paper studies: conduct DR (via Singular Value Decomposition) on the user-Like matrix before building logistic regression models using the resulting reduced-dimensional features to predict the desired traits. As a second example, consider a paper published in the proceedings of the ACM SIGKDD 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 procedure we study. 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.

Both of these data sets initially might appear to be ideal candidates for DR. The dimensionality is very high, the instances are largely sparse, and the data intuitively seem like they ought to contain latent information that could be captured via dimensionality reduction; therefore, there would seem to be a large potential benefit to reducing the sampling variance. However, neither of these papers reports whether using DR actually improves the predictive performance; indeed, for all we know, the predictive performance might have been better without the complexity added by using DR. A natural question is whether the same (or better) performance could have been achieved with supervised regularization alone, which would thus simplify the predictive modeling process (cf., Chen et al. 2017).[1]

Our experiments show that the method that clearly gives the best generalization performance overall does not use DR, and for this method unsupervised DR tends to reduce generalization performance. Therefore, using DR without assessing whether it adds value is a mistake.

Our results also provide a viable explanation for why researchers and practitioners may include DR. The primary results are based on conducting state-of-the-art procedures for selecting the hyperparameters for regularization. If one uses less-sophisticated regularization, or none at all, then using DR features may be superior—although the overall performance will usually be worse.

The empirical study yields five specific conclusions about using unsupervised DR for predictive modeling:

1. On these data, unsupervised MF-based DR is generally unnecessary and potentially diminishes the performance of binary classifiers when used in conjunction with state-of-the-art techniques for supervised complexity control.

[1] The predictive performances in cited papers are very strong, and thus we are not calling into question the conclusions of their studies.

2. Unsupervised DR *can* be beneficial when (a) supervised regularization is not used at all or (b) regularization strength is not selected using state-of-the-art techniques. In general, the harder the binary classification problem, the more using DR will improve performance; however, DR tends to reduce performance for easier problems.
3. DR has a greater relative advantage when used in conjunction with classifiers that leverage some form of internal feature selection; unfortunately, such classifiers tend to perform worse overall. Furthermore, DR has the greatest relative advantage when used in conjunction with k-Nearest Neighbors classifiers, but these classifiers perform quite poorly.
4. Of the three unsupervised DR methods we experiment with, SVD is generally the best-performing, and so if it is necessary or desirable to utilize DR for other reasons (for instance, to decrease training time over multiple, related tasks), SVD is a solid default method to use.
5. Of the 24 feature set/classifier combinations in our experiments, L_2-regularized logistic regression models trained on the original, full feature sets had the best performance hands down. This combination is thus a strong baseline for use with large, sparse, binary data.

A broader lesson is that variations in the methodology used for tuning and evaluating models and modeling parameters can make a big difference in terms of predictive performance. State-of-the-art methods for tuning and evaluating models such as nested cross-validation and grid search are a better use of time and computation resources than adding unnecessary complexity via a DR step. Predictive modelers should keep in mind the principle of Occam's Razor and perform lesion studies (Langley 2000) in order to ensure that predictive modeling systems are no more complex than they need to be. Introducing extraneous components will at best increase processing time, and at worst can result in reduced performance.

We mostly limit this experimental comparison to the very large, sparse data now common in many applications. Similarly to the motivating examples listed above, such data might seem to be ideally suited for DR's use. Further, the main focus of the paper is on binary classification (rather than numerical regression or multi-class classification). Since the primary contribution of the study is the lesson that one should carefully consider whether to apply DR (especially in the context of regularization), depth is more important than breadth. We are *not* claiming that DR will never help—but that we can show an important class of problems where it does not seem to and that researchers and practitioners don't seem to understand that well.

The rest of this paper proceeds as follows. In Sect. 2, we refine the definition of DR for the purposes of this research. We also examine the existing guidelines for use of DR in predictive modeling and review papers from the applied data mining literature that use DR as part of their methodologies. Section 3 describes our experimental settings and the rationale behind our experimental design. Section 4 shows the results of our experiments comparing binary classification performance using full feature versus DR feature sets. Finally, Sect. 5 discusses the implications of our results, and shows some intriguing results that suggest next steps for research.

2 Background and literature review

Dimensionality reduction, in the very broadest sense, can be defined as a class of methods that reduce the size of the feature set used for modeling. In this paper we will consider DR 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 this might happen.

Famously, DR 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. DR is thus employed to avoid overfitting in predictive problems from such data.[2] Guidelines for use of DR in predictive modeling are generally building on this principle.

The second, related 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). DR and other forms of complexity control improve predictive performance by trying to exploit this tradeoff (Friedman 1997; Scharf 1991; Domingos 2012; Kim et al. 2005).

Besides improving predictive performance, DR can be used to compress large data sets, whether to save space or time (Thorleuchter et al. 2012). On the other hand, if the original feature matrices are very sparse, standard DR methods such as Singular Value Decomposition (SVD) are time consuming to compute and generally result in dense matrices which are likely to be larger in terms of storage space. DR can also be used for visualization of latent structure in the data (Westad et al. 2003). Consideration of these tasks is outside the scope of this paper. However, it is worth considering a possible trade-off in predictive performance if DR is employed to reduce processing time, for instance.

There are two ways of accomplishing DR (Guyon and Elisseeff 2003; Liu and Motoda 1998): feature selection and feature engineering. There have been numerous reviews and comparisons of DR techniques (Guyon and Elisseeff 2003; Liu and Motoda 1998), including some that focus chiefly on feature engineering techniques (Van der Maaten et al. 2009) and others that focus on feature selection methods (Blum and Langley 1997; Saeys et al. 2007; Forman 2003). We focus on evaluating the performance of unsupervised matrix-factorization-based feature engineering techniques such as Singular Value Decomposition (SVD) in particular for two main reasons.

First, it is a common practice to use both SVD and some form of supervised feature selection such as L_1 regularization in conjunction. To the best of our knowledge, there is not a systematic comparison of SVD features versus full features in the context of supervised regularization. Second, MF has a long history of use in predictive modeling by industry and research practitioners across many domains, including text classification (Yang 1995), facial image recognition (Turk and Pentland 1991), network security (Xu and Wang 2005), neuroscience (López et al. 2011), and social data analysis (Kosinski et al. 2013). Crucially for applied data mining researchers and

[2] See López et al. (2011), West et al. (2001), and Burl et al. (1998) for a few examples.

industry practitioners, popular software packages such as Python and MatLab contain off-the-shelf implementations of MF techniques for large, sparse data sets that can be seamlessly integrated within a predictive modeling pipeline.

2.1 Predictive modeling setting and notation

The training data $\mathcal{D}$ comprise 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 $\mathbf{x}_i$. Each instance makes up one row of the data matrix $\mathbf{X}$, and each feature corresponds to one column. Each instance i has a nonzero association with at least one attribute. Assume that this association is binary:

$$\begin{aligned} \mathcal{D} &= \{(\mathbf{x}_i, y_i)\} \\ \mathbf{x}_i &= \{x_{i1}, x_{i2}, \ldots, x_{id}\} \\ x_{ij}, y_i &\in \{0, 1\} \end{aligned}$$

The goal is to use a binary classifier such as logistic regression to learn functions $f : \mathbf{x}_i \to [0, 1]$ that score instances with an estimate of class membership probability.[3] Our primary results evaluate each f by how well the resulting scores rank a set of held-out instances $\mathcal{D}_h$, measured using the Area Under the ROC Curve (AUC); however, we check for robustness by also measuring the H-measure, AUCH, and lift.

Matrix factorization (MF) splits the feature matrix $\mathbf{X}$ into two matrices $\mathbf{L}$ and $\mathbf{R}$ such that $\mathbf{X} = \mathbf{LR}$. MF-based DR results in smaller matrices $\mathbf{L}_k \subset \mathbb{R}^{n\times k}$ and $\mathbf{R}_k \subset \mathbb{R}^{k\times d}$ such that

$$\mathbf{X} \approx \mathbf{L_k R_k} \tag{1}$$

The k columns of $\mathbf{L}_k$ can be viewed as "latent factors" underlying the original data. Each instance will have a representation in the new k-dimensional space captured by $\mathbf{L}_k$. The corresponding rows of the $k \times d$ matrix $\mathbf{R}_k$ are sometimes referred to as the "loadings" of the components, representing the relationship between the new features and the original features. Once the matrix factorization has been computed, the resulting latent factors may be used as a new set of features in a predictive model. Thus, we will also learn functions $g : \mathbf{l}_{k,i} \to [0, 1]$. Section 3.3 discusses the specific MF techniques we use and their implementations. Our experiments will compare full feature data and DR performance.

2.2 Existing recommendations for use of DR

The importance of DR as a preprocessing technique for predictive modeling is underscored by its inclusion in data mining and machine learning textbooks (James et al. 2013; Hand et al. 2001; Cios et al. 2007; Tan et al. 2006; Izenman 2008; Friedman et al. 2001). DR is seen as useful for visualizing and compressing data, and potentially beneficial for predictive performance by reducing overfitting via exploiting a bias/variance

[3] Some methods, such as Support Vector Machines (SVMs), provide a score that doesn't necessarily map to the interval [0, 1].

tradeoff. Because supervised regularization is closely related to DR in its purpose and results, such books compare the two techniques. Most textbooks therefore come with a strong caveat that because DR components are not generated in a supervised fashion, unsupervised DR may not be the best regularization technique—supervised regularization likely would be a better choice.

The recommendations from this body of literature range from generally optimistic (Tan et al. 2006; James et al. 2013) to much more guarded (Hand et al. 2001; Cios et al. 2007; Izenman 2008; Friedman et al. 2001). Andrew Ng sums up the caution towards using SVD for predictive modeling in his hugely popular MOOC on Machine Learning:

> So some people think of PCA[4] as a way to prevent over-fitting. But just to emphasize this is a bad application of PCA and I do not recommend doing this. And it's not that this method works badly. If you want to use this method to reduce the dimensional data, to try to prevent over-fitting, it might actually work OK. But this just is not a good way to address over-fitting and instead, if you're worried about over-fitting, there is a much better way to address it, to use regularization instead of using PCA to reduce the dimension of the data.[5]

However, in discussing DR for predictive modeling, there are a few elements missing from the settings considered by the textbooks. Primarily, they assume that DR and regularization are *competing* techniques; however, it is unusual in state-of-the-art predictive modeling contexts to *not* use some form of supervised complexity control for regression or classification. Therefore, researchers and practitioners are left asking, for instance: what if we were to use regularization on top of DR? Additionally, they usually discuss PCR (principal components regression) but do not delve into alternative predictive tasks (such as binary classification), induction algorithms, or DR methods that use components in a similar way. Furthermore, to the best of our knowledge, past literature has not explored whether DR might be more beneficial for classification than regression or vice versa. Finally, the motivating examples chosen to demonstrate PCR use relatively small data sets.

Given that there is some consensus among educational materials for applied data miners, it would be surprising if DR ever improved predictive performance. Yet, practitioners use it, as we have observed in the literature (as cited above), through interactions with industry data scientists, as well as with our own students. To add additional anecdotal support that there is confusion about the value of DR for predictive modeling, we surveyed online forums that are commonly used by data scientists: Stack Exchange, Quora, and Reddit. Advice given on such sites runs the gamut from highly skeptical ("The only situation I can think of where accuracy is actually increased with dimensionality reduction is when you have extremely noisy variables."[6]) to guarded ("there is no

[4] Principal Components Analysis (PCA) is closely related to SVD. See Section 3.3. Often in practice the terms PCA and SVD are used interchangeably.

[5] https://www.coursera.org/learn/machine-learning/lecture/RBqQl/advice-for-applying-pca, accessed August 18, 2018.

[6] https://www.reddit.com/r/datascience/comments/55woqt/selecting_features_for_dimensionality_reduction/, accessed August 18, 2018.

guarantee that this will work and there are often better regularization approaches"[7]) to highly positive ("Performing a singular value decomposition in order to use the derived scores in a classifier has a positive influence in a classifier's overall performance in most cases"[8]). The range of responses—which contradict each other as well as the advice given in textbooks—supports our claim that there is no "go-to" resource that clarifies the question of whether and when to use DR for predictive modeling.

As a further example, dimensionality reduction is widely accepted as a beneficial step in preprocessing and exploring the feature space in the business analytics literature. Shmueli and Koppius (2011) provide a guide to utilizing predictive modeling in business research. DR is a prescribed part of their process because, "reducing the dimension can help reduce sampling variance (even at the cost of increasing bias), and in turn increase predictive accuracy." The authors subsequently codify dimensionality reduction as one of their steps for building a predictive model.

2.3 Mixed performance of DR in predictive modeling literature

Moving beyond the statements in textbooks, empirical comparisons of reduced-dimensionality feature sets versus original features have shown mixed results. 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 DR methods. Specifically, DR 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 in information retrieval (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).

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), Xu and Wang (2005), and Pechenizkiy et al. (2004). Others report that DR resulted in mediocre performance (Guyon et al. 2009). Most tellingly, however, is a survey by Van der Maaten et al. (2009) of linear and nonlinear dimensionality reduction techniques. This survey utilized both "natural" and "synthetic" data sets.[9] They found that on the majority of their "natural" data sets, classification error was not improved by doing any form of dimensionality reduction on the feature space. However, they hedge their conclusion by explaining this as likely being due to an incorrect choice of the number of latent factors to include as features ("*k*").

[7] https://stats.stackexchange.com/questions/141864/how-can-top-principal-components-retain-the-predictive-power-on-a-dependent-vari, accessed August 18, 2018.

[8] https://stats.stackexchange.com/questions/142557/is-dimensionality-reduction-almost-always-useful-for-classification, accessed August 18, 2018.

[9] Their results differ from ours, below, in that (i) they do not use sparse data, (ii) their data sets 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.

Beyond this mixed evidence, there are numerous 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. The various methods to compute DR are very time consuming and computationally expensive. DR 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 using DR, 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 DR and no-DR 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 DR to their feature sets (Kosinski et al. 2013; Cai et al. 2013; Der et al. 2014; Arulogun et al. 2012).

Thus we see that some practitioners favor the use of DR, not only in the online advice, but in practice in published papers—where DR features frequently are used without a comparison against the full feature set. The results we present below suggest that if maximizing generalization performance is a goal, then using DR features without a comparison to the full feature set is a mistake. However, we also show results that provide one clear explanation as to why practitioners may make this mistake: if you don't use state-of-the-art methods for choosing the degree of regularization with the full feature set, then you often will not see the advantage over using the DR features.

3 Experimental test-bed

Broadly, the experiments in this paper explore the relative performance of DR features versus original features. We evaluate performance on a suite of binary prediction tasks using large, sparse feature sets. A key dimension of our experiments is that we vary the methodology used to tune parameters and report performance. The core results focus on two parameters: the number of DR components k used as features, and the supervised regularization parameter C. We vary the method for choosing each of these: either a single default value, or task-specific grid search using nested cross-validation.

Nested cross-validation adds an inner loop of cross-validation to each of the folds of standard cross-validation (Provost and Fawcett 2013). In the inner loop we do 3-fold cross-validation to estimate the performance of each parameter combination in our grid. In the outer loop, a new model is trained using the parameter combination that yielded the best performance 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. Figure 1 describes the basic experimental procedure.

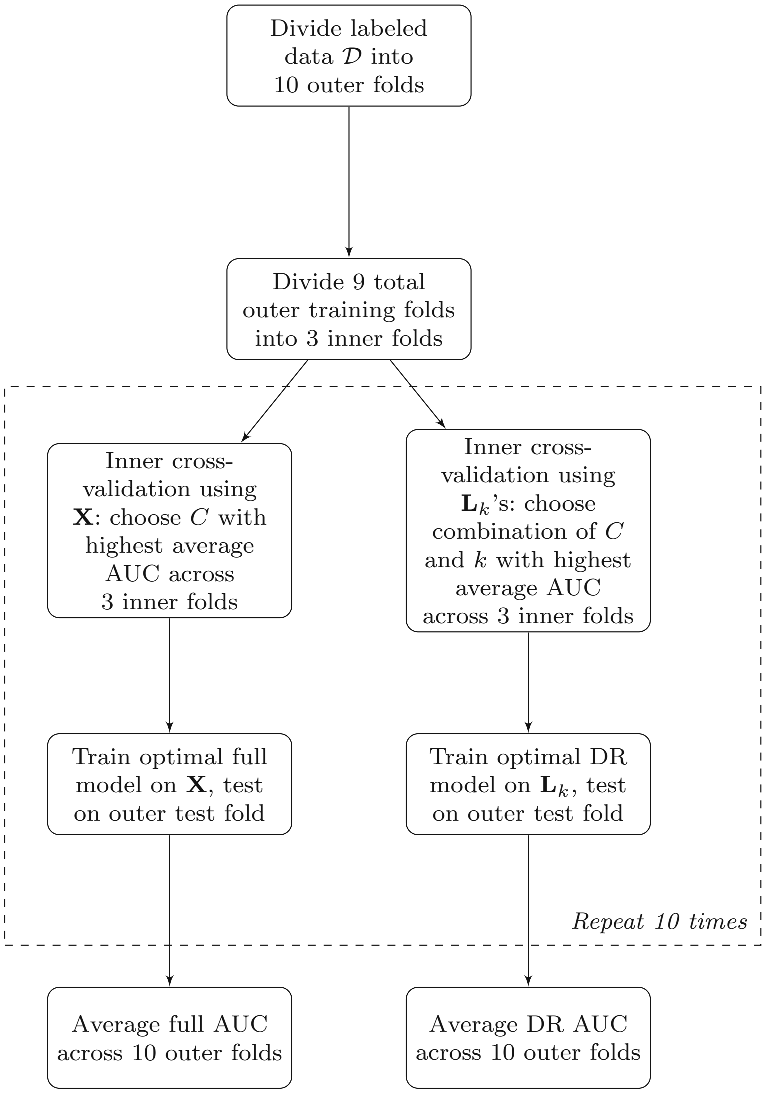

Divide labeled data $\mathcal{D}$ into 10 outer folds
Divide 9 total outer training folds into 3 inner folds
Inner cross-validation using $\mathbf{X}$: choose C with highest average AUC across 3 inner folds
Inner cross-validation using $\mathbf{L}_k$'s: choose combination of C and k with highest average AUC across 3 inner folds
Train optimal full model on $\mathbf{X}$, test on outer test fold
Train optimal DR model on $\mathbf{L}_k$, test on outer test fold
Repeat 10 times
Average full AUC across 10 outer folds
Average DR AUC across 10 outer folds

Fig. 1 Experimental procedure for comparing full to DR performance, using nested cross-validation for unbiased parameter selection and evaluation

Table 1 summarizes the combinations of parameter selection methodologies that we experimented with. All experiments use 10-fold cross-validation for evaluation (we note specifically where 3-fold nested cross-validation has been added in). Our primary results measure binary classification performance using AUC, which we further validate using three other evaluation metrics: H measure (Hand 2009), AUCH (Hand 2009), and lift. The remainder of this section goes into more detail on the various feature sets, predictive tasks, DR techniques, and binary classifiers used in the experiments.

Table 1 Modeling test-bed settings, varying model-tuning methodology

		k	
		Chosen using NCV	$k = 100$
C	Chosen using NCV	✓	
	$C = 1$	✓	✓
	No regularization		✓

NCV nested cross-validation

3.1 Data sets

Our main experiments utilize data collected by Kosinski et al. (2013). 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 data that is increasingly available, and which can provide markedly better predictive performance than traditional data (Martens et al. 2016). While the main results are a deep dive into many data sets based on one feature set, Sect. 4.6 shows results for 76 additional tasks drawn from 15 other large, sparse data sets.

The researchers collected data via a Facebook application that allowed users to take personality tests and also reported the users' Likes and profile information. Using this data, it is possible to formulate a user-Like matrix $\mathbf{X}$ such that $\mathbf{X}_{ij} = 1$ 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. The user-Like matrix we use comprises $n = 211{,}018$ 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 non-zero). The average user Liked 195 items and the average item was Liked 166 times.[10]

The second component of the data is a set of 21 binary and numeric prediction tasks, drawn from the app users' personality test results, profile information, and survey items. We replicate the authors' predictions for all of the binary target variables. These include "single versus in relationship", "parents together at 21", "smokes cigarettes", "drinks alcohol", "uses drugs", "Caucasian versus African American", "Christianity versus Islam", "Democrat versus Republican", "gay", "lesbian", and "gender". Counts of the number of users and items for which each binary target has label values can be found in Table 5 of Appendix A. We followed the labeling procedure undertaken by Kosinski et al. for all target variables except "Caucasian versus African American," which was labeled by visual inspection of profile pictures not available to us; we instead 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 the target was set to 1, and 0 otherwise. These numerical variables are "satisfaction with life", "intelligence", "emotional stability", "agree-

[10] We store this resulting matrix $\mathbf{X}$ as a SciPy sparse matrix (Jones et al. 2001) in Python.

ableness", "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 $\geq$ 30; others were set to be the median or top-quartile value for that variable to ensure variety in base rates. More information on the various numerical variables and their thresholds can be found in Table 6 in Appendix A.

An important note is that although the 21 predictive tasks all rely on the same testbed of features, there is wide variation in important data set characteristics across the tasks. Namely, some of the tasks have far more features than instances, and for some the feature matrix is roughly square. There is also variation in the base rate (percent of positive instances). Tables 5 and 6 in Appendix A contain more information about the shapes and base rates of the various feature matrices.

Finally, we follow up this study by studying 15 additional (publicly available) data sets comprising 76 total predictive tasks, many of which were collected from the UC Irvine Machine Learning Repository (Dheeru and Karra Taniskidou 2017). Some of the feature sets did not initially comprise binary features, and some of the default tasks were not binary classification. Just as before, we binarized features or target variables where necessary by using a threshold. These data sets also vary in their shapes, sizes, sparsity levels, and base rates. A listing of the sizes of these data sets, the number of nonzero elements, and the base rates can be found in Appendix A, Table 7. Descriptions of the tasks are given here:

1. *Book* Predicting users' ages based on books rated (Ziegler et al. 2005).
2. *Movies* Predicting users' genders based on movies rated.[11]
3. *Ta-Feng* Predicting users' ages based on products purchased.[12]
4. *URL* Predicting whether a URL is malicious based on website features (Ma et al. 2009).
5. *Flickr* Predicting volume of comments on Flickr pictures based on which users have "favorited" them (Cha et al. 2009).
6. *Movielens* Predicting the age and gender of Movielens users, given the movies they've rated, and predicting whether a movie belongs to each of 18 genres, given the users who have rated it (20 total tasks) (Harper and Konstan 2016).[13]
7. *CiteSeer* Predicting whether "J. Lee" is the author of a particular scientific paper based on the other authors of the paper.[14]
8. *Daily and sports activities* Predicting whether a subject is performing each of 19 activities based on physical sensor outputs (19 prediction tasks) (Altun et al. 2010; Barshan and Yüksek 2014; Altun and Barshan 2010).[15]
9. *DeliciousMIL* Predicting whether a web page from del.icio.us is tagged with each of 20 tags using the page text (20 prediction tasks) (Soleimani and Miller 2016).[16]

[11] http://webscope.sandbox.yahoo.com/, accessed August 18, 2018.

[12] http://recsyswiki.com/wiki/Grocery_shopping_datasets, accessed August 18, 2018.

[13] https://grouplens.org/datasets/movielens/20m/, accessed August 18, 2018.

[14] http://komarix.org/ac/ds/#spardat, accessed August 18, 2018.

[15] https://archive.ics.uci.edu/ml/datasets/daily+and+sports+activities, accessed August 18, 2018.

[16] https://archive.ics.uci.edu/ml/datasets/DeliciousMIL%3A+A+Data+Set+for+Multi-Label+Multi-Instance+Learning+with+Instance+Labels#, accessed August 18, 2018.

10 *E-commerce* Predicting an online shopper's gender based on the products they've looked at.[17]
11. *Farm Ads* Predicting whether a farm-animal-themed website owner approves of a particular ad, given the ad creative and website text.[18]
12. *Gisette* Predicting whether a handwritten digit is a 4 or a 9 given pixel values and higher-order features (Guyon et al. 2005).[19]
13. *IMDB* Predicting whether the voice actor Mel Blanc was involved in a particular film or television program, based on the other cast and crew members associated with the project.[20]
14. *p53 mutants* Predicting whether a p53 protein molecule is active or inactive (i.e. cancerous), based on physical features of the molecule (Danziger et al. 2009, 2007, 2006).[21]
15. *Reuters* Predicting whether a Reuters document belongs to each of six categories, given the text of the document (six prediction tasks) (Amini et al. 2009).[22]

3.2 Predictive methods and complexity control

Our experiments evaluate the performance of binary classifiers trained using the scikit-learn package in Python (Pedregosa et al. 2011). The core results use L_2-regularized logistic regression (L_2-LR). This method has been found to have superior performance on large, sparse data such as these in a recent independent benchmarking study (De Cnudde et al. 2017). Training LR (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 2006). In practice, training an L_2-LR model involves finding weights $\mathbf{w}$ that minimize:[23]

$$C \sum_{i=1}^{n} \log\left(1 + \exp\left(-y_i \mathbf{w}^T \mathbf{x}_i\right)\right) + ||\mathbf{w}||_2 \tag{2}$$

In Eq. (2), C is the regularization parameter, which controls the trade-off between minimizing the logistic loss and the amount of regularization. Our experimental conditions consider both scikit-learn's default ($C = 1$) and state-of-the-art procedures for evaluating and selecting the optimal parameter value. Specifically,

[17] https://knowledgepit.fedcsis.org/contest/view.php?id=107, accessed August 18, 2018.

[18] https://archive.ics.uci.edu/ml/datasets/Farm+Ads, accessed August 18, 2018.

[19] https://archive.ics.uci.edu/ml/datasets/Gisette, accessed August 18, 2018.

[20] http://komarix.org/ac/ds/#spardat, accessed August 18, 2018.

[21] https://archive.ics.uci.edu/ml/datasets/p53+Mutants, accessed August 18, 2018.

[22] https://archive.ics.uci.edu/ml/datasets/Reuters+RCV1+RCV2+Multilingual,+Multiview+Text+Categorization+Test+collection, accessed August 18, 2018.

[23] Equation 2 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.

we performed nested cross-validation paired with a grid search choosing C from {.001, .003, .009, .027, .081, .243, .729, 2.187, 6.561, 19.683}.

We also include results for additional classifiers: linear classifiers (SVMs and L_1-regularized logistic regression), tree-structured classifiers (decision trees and random forests), and k-Nearest Neighbors. The experiments using L_1-LR and SVMs also use a grid search choosing C from {.001, .003, .009, .027, .081, .243, .729, 2.187, 6.561, 19.683}. There are alternative ways of controlling complexity in tree induction; our experiments using trees and random forests models use the heuristic of setting a minimum number of objects allowed to be in a leaf node,[24] choosing among {2, 4, 8, 16, 32, 64, 128, 256, 512}. Finally, we select the number of neighbors considered in each kNN model among {5, 10, 50, 100, 150, 200, 500, 1000, 2000}.

Finally, Sect. 4.8 shows results for a different predictive task: numerical regression. These results were generated using L_2-regularized linear regression, also known as Ridge Regression. Similarly to logistic regression, ridge regression allows for selection of a regularization parameter λ which controls the complexity of the regression. Here, we select the regularization parameter from {.01, .1, 1, 5, 10, 15, 20, 25, 30}. Performance is reported in terms of Pearson's r.

Our main results are computed using the popular AUC (Area Under the ROC Curve) metric. To ensure that these results generalize, we also compute three additional performance metrics: the H measure (Hand 2009), the AUCH (area under the convex hull of an ROC curve) (Hand 2009), and the lift over random selection at the top 5% of test instances (which we will call lift at 5% going forward). Because we do not have use cases specified for the learned models, we have mainly chosen metrics that are not dependent on a particular classification threshold, such as classification accuracy.

3.3 Dimensionality reduction methods

SVD[25] is the historical method of choice for researchers and practitioners who use matrix-factorization-based DR for predictive modeling across a wide variety of domains, including text classification (Yang 1995), facial image recognition (Turk and Pentland 1991), network security (Xu and Wang 2005), neuroscience (López et al. 2011), and social data analysis (Kosinski et al. 2013).[26] Implementations are

[24] We choose this alternative over others, such as specifying a maximum depth of the tree, following the observation of Provost and Fawcett (2013) that "A nice property of controlling complexity in this way is that tree induction will automatically grow the tree branches that have a lot of data and cut short branches that have fewer data—thereby automatically adapting the model based on the data distribution" (Chapter 5).

[25] Principal Components Analysis (PCA) is closely related to SVD: the features are preprocessed so that the sample mean of each is 0. 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 this reason, the sample mean normally is not subtracted out for sparse data, technically yielding SVD rather than PCA. Often in practice the terms PCA and SVD are used interchangeably.

[26] 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

widely available in commonly used data mining software packages such as Python, MatLab, and R.

SVD is popular for a reason: it has highly desirable properties. The SVD components (sometimes referred to as latent factors) are usually sorted in decreasing order of the amount of sample variance that they account for in the original data matrix. 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. Importantly, the resulting truncated SVD matrices can be viewed as representing some latent structure in the data. Additionally, SVD is computed by optimizing a convex objective function; the solution is unique and equivalent to the eigenvectors of the data matrix. Thus, a further benefit of using SVD is that the decomposition can be computed once for the maximum desired k and the resulting components selected in decreasing order of their value. Because of these advantages as well as SVD's popularity, the main experiments in this paper focus on SVD as the DR method.[27] We compute the SVD of the user-Like matrix using SciPy's sparse SVD package for Python.

However, SVD is far from the only DR method used by researchers or practitioners, even within the class of unsupervised MF-based DR techniques. These methods do not necessarily come with the same desirable guarantees as SVD, though. Other methods generally do not have a unique solution; therefore the resulting components may be different depending on how the optimization algorithm is initialized. Further, the number of components must be specified *a priori*: computing factorizations with different numbers of components will result in completely different components. Additionally, these components will not be sorted in any order of relevance; all components may account for equal variance in the data (or be equally relevant to the target variable). Finally, and crucially for one important audience of this research, other methods are not as widely implemented in popular off-the-shelf software packages, especially for sparse data structures. Therefore, for this simple reason, they are far less useful to those who simply want to utilize DR to improve predictive performance, rather than do research on the use of DR.

Of course, every alternative DR method does come with special advantages, and some have been found to perform well in certain practical contexts (Xing and Girolami 2007; Shahnaz et al. 2006). To ensure that our results are not dependent on the choice of DR method, we also explore additional methods. Non-negative Matrix Factorization (NMF) requires that the resulting components must be non-negative. This constraint usually results in sparse (and therefore, potentially interpretable) components, and was originally developed for image processing applications (Lee and Seung 1999). Latent Dirichlet Allocation (LDA) is a generative probabilistic model for matrix factorization that may capture a more realistic picture of the relationships in the data than SVD does. It was originally developed for text modeling (Blei et al. 2003). Both have shown promising predictive performance in certain contexts (Shahnaz et al. 2006; Bíró et al.

dimensionality of the features actually yielded the best performance in the majority of cases—a finding that this paper explores in greater detail, with sparse data.

[27] When the DR method is PCA and it is used in conjunction with linear regression, this process is sometimes referred to as Principal Components Regression. Many more technical details are given by James et al. (2013).

2008). We compute the NMF and LDA decompositions of our feature matrices using Python's scikit-learn package with default settings (Pedregosa et al. 2011).

3.3.1 Dimensionality reduction parameter

An important computational aspect of using DR in predictive modeling is choosing the optimal value for k, the number of components that will be used as features in the model. There is substantial evidence that utility peaks with a certain number of dimensions.[28] In fact, "selection of number of dimensions" is mentioned as being important in almost every paper that utilizes dimensionality reduction for predictive modeling.

There are numerous ways to estimate the optimal value for k. First: use an operational criterion; that is, pick the k that optimizes predictive performance on held-out data. Another way is to rank the singular values (elements of S) and pick the top few based on visual inspection, for instance, by looking for a "knee" in the singular values (Hoff 2007; Burl et al. 1998).[29] 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. 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.

As with the regularization parameter C, we also experiment with different methodologies to select k. Kosinski et al. state that they utilize the top $k = 100$ components as features and so that is what we use as the default in our predictive models, for all three DR methodologies. Given the ubiquity of mentions of careful selection of k in past DR literature, we also experiment using nested cross-validation to choose k, selecting from {50, 100, 200, 500, 1000}.

4 Experimental results

This section presents results comparing learning with DR features versus learning with the full feature set across different domains and data sets, different modeling algorithms, and different evaluation criteria.

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

[29] Burl et al. (1998) also use visual inspection of the latent factors themselves, given that the data are images.

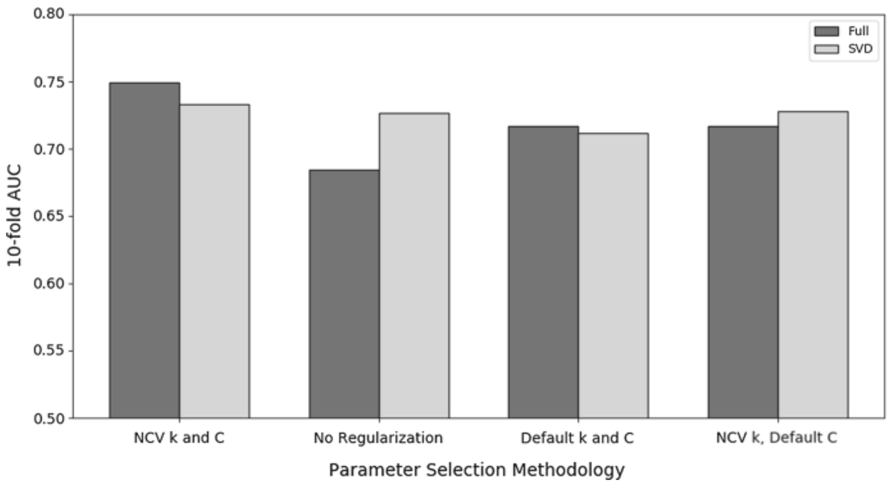


Fig. 2 Mean binary classification performance (AUC) across 21 tasks for SVD features versus full features, for each of the four parameter selection methodologies in our experiments. Left-to-right: using nested cross-validation (NCV) to set both C and k; no regularization and default k; default values for both k and C; and default C but selecting k via NCV. Note that although using the state-of-the-art regularization with the full feature set is the best method overall, if one did not undertake the NCV grid search, then one might draw a different conclusion

4.1 Main results

Let us first focus on the suite of Facebook user-Like prediction problems described in Sect. 3. We compare DR-feature predictive performance versus full-feature predictive performance on the 21 classification tasks. Each set of experiments represents a different set of modeling design choices. The core results vary the methodology used to select the number of SVD components k and supervised regularization parameter C: either default settings, or parameters chosen using nested cross-validation and grid search.

Figure 2 compares the average SVD and full-feature performance across tasks for each modeling methodology. These results show several noteworthy things. First, using the original, full feature set yields the best performance overall—better than any use of DR—but only when using what we would consider to be "best practices" for predictive modeling. In particular, you have to choose the regularization parameter carefully via a state-of-the-art (S.O.T.A.) method such as nested cross-validation. Just choosing the default regularization parameter is not sufficient. In fact, as we will see, this best-practice method beats using DR on every single task. Second, if you are not regularizing using a S.O.T.A. method, using the DR features may indeed outperform using the full feature set: in two of the other settings, SVD actually is better in aggregate (the full-feature performance is still slightly better than SVD when modeling using default parameter settings).

Table 2 presents the average time spent training classifiers and predicting estimates on test data for four of the prediction methodologies, assuming 10-fold cross validation with 3-inner-fold nested cross validation for the 21 Facebook tasks. While the

Table 2 Average number of seconds to run 10 outer folds, for each task, across various parameter selection methodologies

Task name	Full: Default	SVD: Default	Full: NCV	SVD: NCV K
Parents	2.9	0.1	61.4	5.6
Race	4.1	0.1	90.8	8.5
Religion	4.2	0.1	107.0	13.3
Uses drugs	4.7	0.1	83.9	8.8
Drinker	8.3	0.2	135.2	11.2
Smoker	8.4	0.1	130.2	12.6
IQ	13.2	0.3	161.4	25.1
Satisfaction with life	14.5	0.2	220.3	22.6
Politics	36.8	0.6	504.0	61.5
Gay	73.2	2.8	721.0	148.4
Lesbian	96.4	1.5	1424.4	156.0
Network density	182.8	2.4	2548.2	295.6
Gender	1225.5	25.8	19,615.0	3945.8
Age	1465.8	21.1	20,249.6	2441.3
Agreeableness	1823.5	23.8	28,561.6	1943.1
Number of friends	1834.5	20.4	18,993.9	2355.0
Conscientiousness	1864.0	21.7	22,932.0	1752.8
Extraversion	1901.4	18.2	20,684.9	1506.6
Relationship	2115.7	21.3	23,571.6	2748.0
Openness	2124.1	26.2	21,613.2	1867.0
Emotional stability	2377.6	22.6	29,231.8	2244.8

Computing the SVD as a preprocess took 6138 s, which affects the overall run times for the SVD approach

best performance overall is obtained when modeling with the full feature set (using NCV to conduct parameter selection), the shortened training/prediction time associated with SVD models means they are a compelling alternative if prediction time is more of a concern than performance. However, note that computing the SVD of this feature set took 6138 s. Thus, using SVD only saves time for tasks that are more time consuming overall, or when it is computed once and then reused for many predictive tasks.

The remainder of this section delves more deeply into these results, and conducts further experiments using different classifiers, DR techniques, data sets, and alternative predictive tasks.

4.2 Problem-specific parameter selection

Let's look deeper into the comparison of L_2-LR's predictive performance using the full feature set and using SVD features, with both regularization parameters and k chosen using nested cross-validation—i.e., the leftmost pair of bars in Fig. 2. The complete

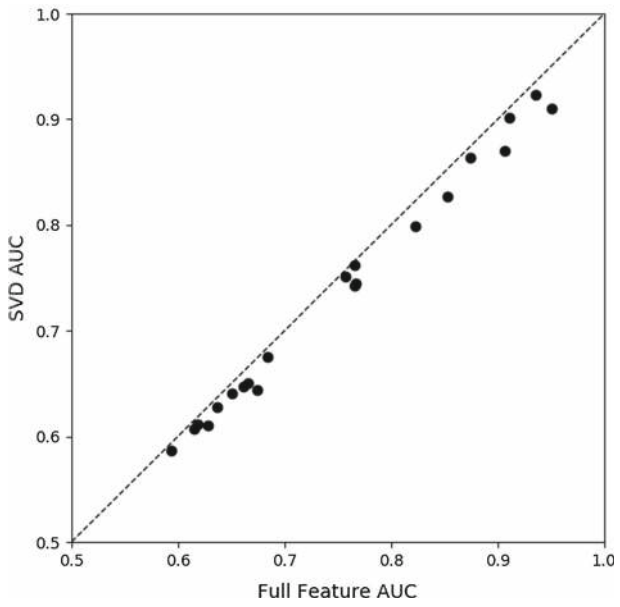


Fig. 3 SVD versus full feature performance with regularization settings (and k) selected through nested cross-validation. Each point represents the (Full Feature AUC, SVD AUC) pair for one classification task. Points falling below the $y = x$ line have better full feature performance, and vice versa. SVD at best gives no advantage and in many cases significantly diminishes performance

grid search covers every combination of C and k (50 total possible combinations for the SVD approach). Figure 3 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. Points above the $y = x$ line on the plot are cases where using SVD improves performance; points below are where using SVD decreases performance.

Figure 3 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 for every task! 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 −0.013, z = −4.015, which is significant for $p < .01$. Thus, we reject the null hypothesis—SVD does indeed hurt predictive performance.

This result is further illustrated in Fig. 4, which shows the extent to which using nested cross-validation to select C improves predictive performance over the default parameter setting (for C) when modeling using the full feature set. Carefully selecting regularization parameters never diminishes performance, which is not the case for SVD (as we will show in the next subsection). It is both safer and more effective to use time and computational resources for careful selection of regularization parameters rather than computation of SVD.

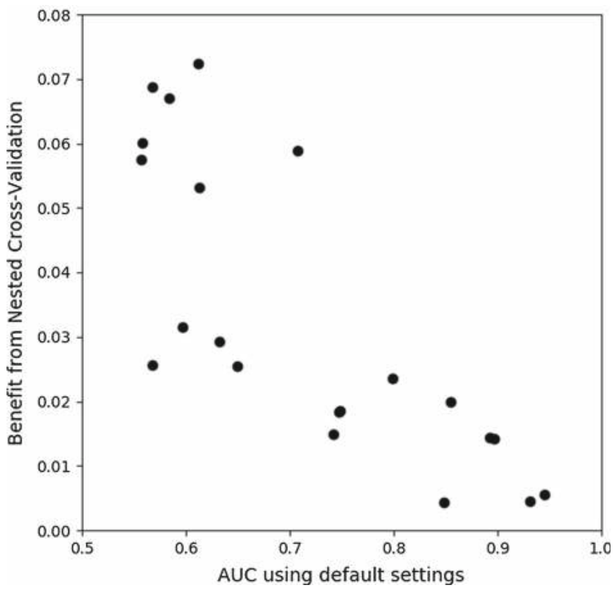


Fig. 4 Improvement over default settings when using nested cross-validation to select regularization parameters, when modeling with the original feature set. Each point represents the (default settings performance, benefit from using NCV) pair for one classification task. While utilizing SVD results in diminished performance for every task (see Fig. 3), carefully choosing regularization parameters only improves performance

4.2.1 Concatenated feature sets

A question that surprisingly, to our knowledge, has not been raised in prior research is: why not use both feature sets? To further investigate whether SVD adds some predictive benefit when modeling, we conducted additional experiments using feature sets constructed by concatenating the full feature sets and SVD feature sets. To the best of our knowledge, past research has not experimented with such feature sets. We constructed new matrices $\mathbf{A}_k$ from the full feature sets $\mathbf{X}$ and SVD features $\mathbf{L}_k$[30] such that

$$\mathbf{A}_k = [\mathbf{X} \quad \mathbf{L}_k] \tag{3}$$

Once again, both k and C were selected using nested cross-validation and grid search. Figure 5 compares the performance of modeling with the concatenated $\mathbf{A}_k$ matrices (with k chosen using nested-cross validation as above) to the performance with the original full feature set. Performance using the concatenated feature set is statistically significantly better than the original feature set: the median difference is .0001, and the Wilcoxon signed rank test yields z = 2.83, which is significant for $p = .05$. However, we do note most tasks have difference close to zero between full and SVD feature sets.

[30] We normalized each SVD component in $\mathbf{L}_k$ to have minimum value of zero and maximum value of 1 so that each feature in $\mathbf{A}_k$ took on the same range of possible values.

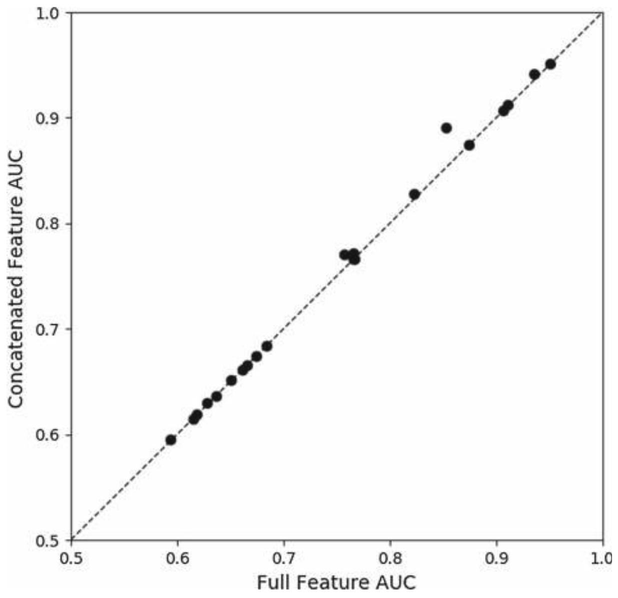


Fig. 5 Comparing the performance of the two feature sets concatenated (full features + SVD features) to the full feature performance. Each point represents the (full feature AUC, Full + SVD AUC) pair for one classification task. Points falling below the $y = x$ line have better full feature performance, and vice versa. Adding the SVD features to the full features is beneficial, although this is largely due to one task

4.3 Alternative parameter selection methodologies

This section provides evidence that there *are* model selection and evaluation settings that result in relatively better performance for SVD versus full-features. These results could explain why SVD remains a popular technique in the applied data mining literature, despite the guidelines that supervised regularization is superior for complexity control. However, recall the result summarized in Fig. 2: these settings result in lower overall average performance.

4.3.1 No regularization

The main rationale for using SVD to improve predictive performance, as we discuss above (and many others have noted), is that the reduced-dimensional data set will be less “noisy”, thereby resulting in a lower-variance estimator (at the possible cost of introducing some bias). This is also the rationale for using regularization in supervised classification algorithms (James et al. 2013; Hand et al. 2001; Cios et al. 2007; Tan et al. 2006; Izenman 2008; Friedman et al. 2001). Because of the lack of complexity control, we would expect that classifiers without supervised regularization would overfit, and therefore perform poorly.

Figure 6 compares SVD performance to full-feature performance when regularization is not employed in the model training process. In this setting, models trained using the SVD features are significantly better than those trained using the original full

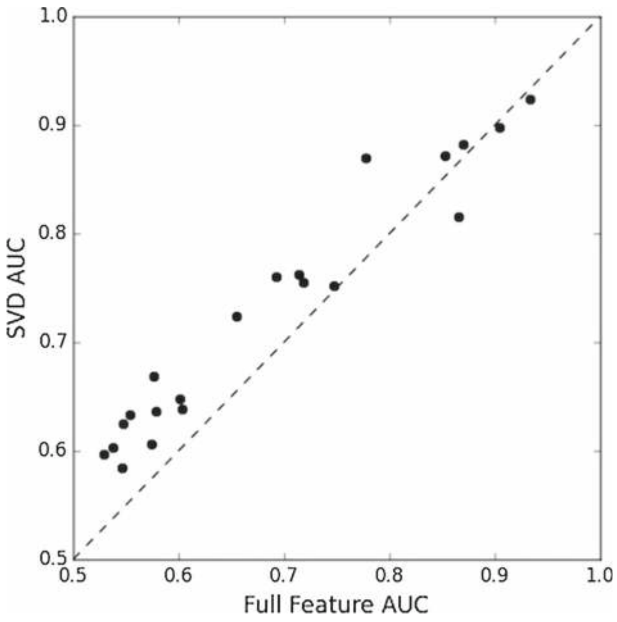


Fig. 6 SVD versus full feature performance, using no supervised regularization. Each point represents the (full feature AUC, SVD AUC) pair for one classification task. Points falling below the $y = x$ line have better full feature performance, and vice versa. In this case, SVD has significantly better performance

feature sets; the median improvement seen from using SVD features is .047 and the z-score from a Wilcoxon signed rank test is 3.424 (significant for $p < .01$). Note that the benefit from SVD seems to be most concentrated for those cases where the AUC using the full feature set was lower; we discuss this further in the next subsection.

4.3.2 Default regularization and fixed *k*

As mentioned above, DR and regularization are not necessarily mutually exclusive. The two methods for avoiding overfitting can be, and frequently are, used in conjunction. Therefore, data miners who want to avoid the time-consuming process of conducting a complete grid search to tune both k and C might consider using default values for both, rather than not using regularization at all. This next set of experiments compares performance using scikit-learn's default regularization settings for L_2-LR (i.e. $C = 1$) and $k = 100$. Similarly to Figs. 3 and 6, Fig. 7 compares full feature performance on the x-axis versus SVD performance on the y-axis.

Again, SVD does indeed improve performance for many of the targets. For these pairs of points, the median difference is 0.008 and the resulting z score from applying the Wilcoxon signed rank test is -0.122, which means that we fail to reject the null hypothesis at $p = .01$, .05., and .1. These results imply that overall, SVD neither helps nor hurts predictive accuracy. However, also note that there appear to be several regimes within the distribution. Points toward the far top-right of the plot are pre-

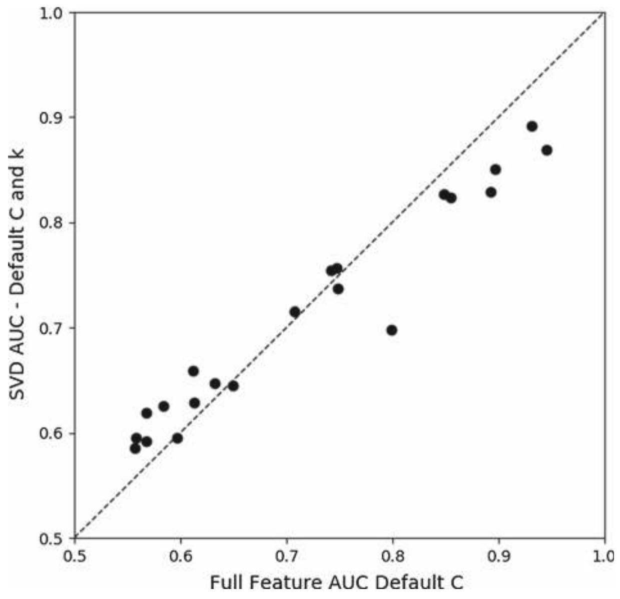


Fig. 7 Comparing SVD versus full feature set using LR with default k and regularization parameter C. Each point represents the (full feature AUC, SVD AUC) pair for one classification task. Points falling below the $y = x$ line have better full feature performance, and vice versa. SVD neither helps nor hurts in general. The benefit of SVD seems to relate to the difficulty of the predictive modeling problem

dictive problems where SVD hurt more than helped and vice versa for points 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 Fig. 8 bear this idea out. Figure 8 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 between the two sets of points. This implies that for easier problems, caution should be exercised when applying 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. This is perhaps because the "more difficult" tasks suffer due to increased overfitting; SVD in these cases reduces variance sufficiently to improve generalization performance. It is worth reiterating that here—using the default regularization parameter—the generalization performance is systematically worse than using state-of-the-art regularization, as shown earlier.

4.3.3 Problem-specific k, fixed C

As is discussed above, correctly choosing k (the number of SVD components used as features in predictive modeling) is crucial. Given that almost all prior work mentions the importance of setting this parameter well, one might justifiably wonder whether

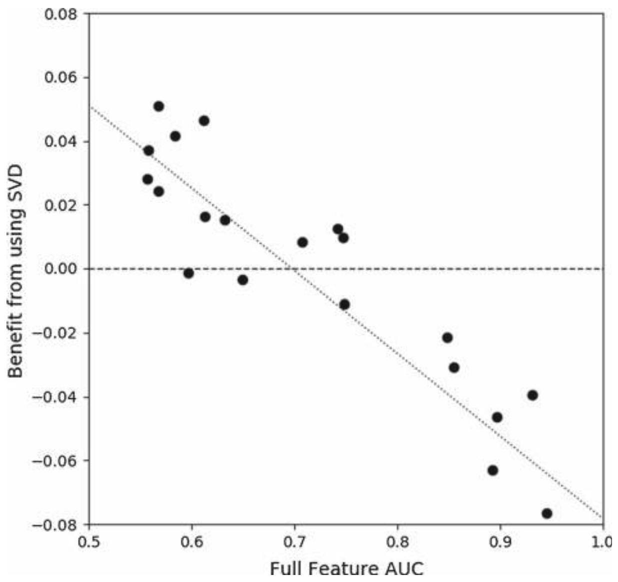


Fig. 8 Comparing the amount of benefit from using SVD to the difficulty of the original predictive problem (full feature AUC). Points falling above the $y = 0$ line are those where SVD had positive benefit, and vice-versa. Each point represents one classification task, using default settings for k and C. There is a strong relationship between the difficulty of the original predictive problem and the benefit from using SVD: easier predictive problems (with high AUC on the full-feature task) have diminished predictive performance from using SVD; SVD benefits harder predictive problems

the last results 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.

Figure 9 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. The median benefit from using SVD here (over full-feature performance with the default C) is 0.012, and the z-score is 1.686 (significant at $p = .10$).

Also note that there still appear to be two regimes within the tasks shown in Fig. 10. 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). Even when choosing k using state-of-the-art predictive modeling techniques, SVD does not necessarily help with predictive performance, especially on easier tasks. The lesson here should thus be that utilizing SVD necessitates also selecting k through nested cross-validation (although such a selection does add to the total computation time).

While the literature definitely leans to the recommendation that DR is a worse option than supervised regularization when it comes to complexity control, these results show that it depends on how one performs the regularization. When using less-sophisticated methods for selecting regularization hyperparameters, DR does indeed provide benefit.

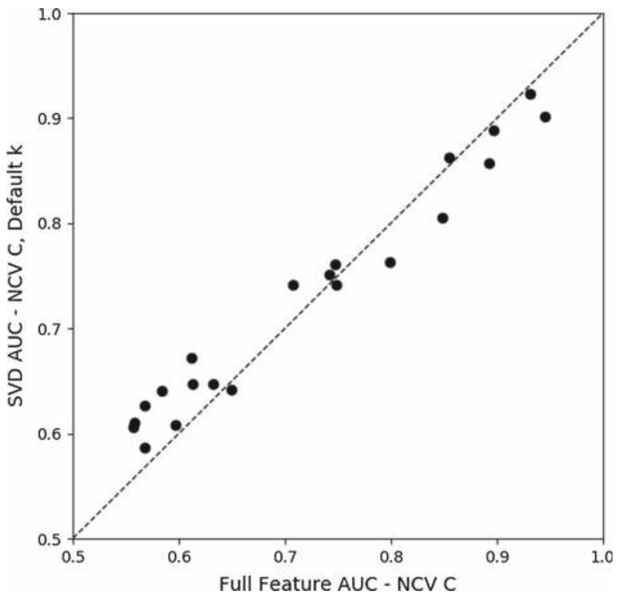


Fig. 9 Comparing SVD versus full feature set using default regularization settings and k selected using nested cross-validation. Each point represents the (full feature AUC, SVD AUC) pair for one classification task. Points falling below the $y = x$ line have better full feature performance, and vice versa. When using default regularization, SVD performs slightly better relative to full-feature performance than in Fig. 7.

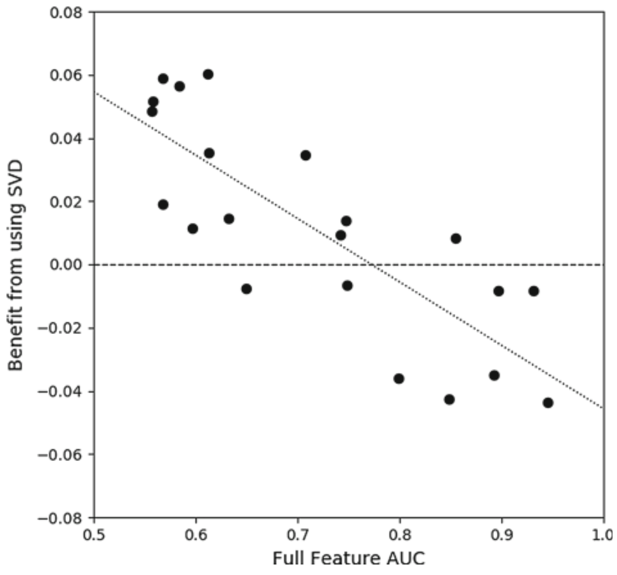


Fig. 10 Relating the amount of benefit from using SVD to the difficulty of the original predictive problem (full feature AUC). Points falling above the $y = 0$ line are those where SVD had positive benefit, and vice-versa. Each point represents one classification task, using default regularization parameter and k selected using nested cross-validation. Again, harder predictive problems here have positive benefit from using SVD; this is not necessarily the case for easier problems

The upshot is that SVD may well appear better to someone who is not using nested cross-validation to perform a full grid search to tune crucial modeling parameters! The remainder of this section replicates the experiments using nested cross-validation to select C and k, but with different classifiers, DR techniques, and data sets.

4.4 SVD with other classifiers

We find that the choice of classifier has a significant effect on the relative benefit of SVD versus full features for predictive modeling. This section replicates the experiments of Sect. 4.2 using four additional classifiers: Support Vector Machines, L_1-LR, classification trees, and random forests. We further present additional results using kNN classifiers in Sect. 4.6.

SVM classifiers (with linear kernel) result in similar performance as in Sect. 4.2, as shown in Fig. 11a: the median benefit from using SVD is -0.01, and the z-score from a Wilcoxon signed rank test is -1.99 (significant at $p = .05$). SVMs are similar to L_2-LR in that both are linear models with L_2 regularization (SVMs are trained using hinge loss instead of logistic loss). This potentially explains why the relative performance of DR and full features follow the same pattern for these two classifiers.

The effective difference between L_2-LR and L_1-LR is that L_1-LR implicitly selects a subset of the total features by driving some or even most features' coefficients to zero, resulting in sparse coefficient vectors. Figure 11b plots a comparison of full-feature performance versus SVD performance using L_1-LR for both, using nested cross-validation to search over both k and C. Here, SVD performs significantly better than the full-feature models! The median benefit from using SVD is 0.007, and the z-score from a Wilcoxon signed rank test is 1.895 (significant at $p = .1$).

Similarly, Fig. 12a, b compare full-feature performance versus SVD performance using classification tree and random forest models, respectively. For trees, the median benefit from using SVD is .012, $z = 1.616$. For random forests, the median benefit from using SVD is -0.002, $z = -0.261$. Neither of these differences are significant at $p = .1$.

On the whole, L_2-LR and SVM are the two classifiers for which modeling using the full feature set is significantly better than using SVD features. L_2-LR is also the best classifier overall, with a mean AUC across the 21 tasks of .749. SVM has the second best overall mean AUC at 0.739.

On the other hand, using SVD features performs relatively better than full features for classifiers that implicitly or explicitly incorporate some form of feature selection: L_1-LR, classification trees, and random forests. The relatively better performance of SVD in the context of feature-selecting models makes some sense. Due to the sparse nature of the original features, a limited subset of them is not likely to include all information available to train the best model. Because the SVD features each include information from potentially all of the original features, perhaps the necessary information is included even though the modeling algorithm selects a subset of the SVD components. However, note both that the overall performance of such classifiers is lower overall, and that the difference between the two feature sets is not necessarily statistically significant.

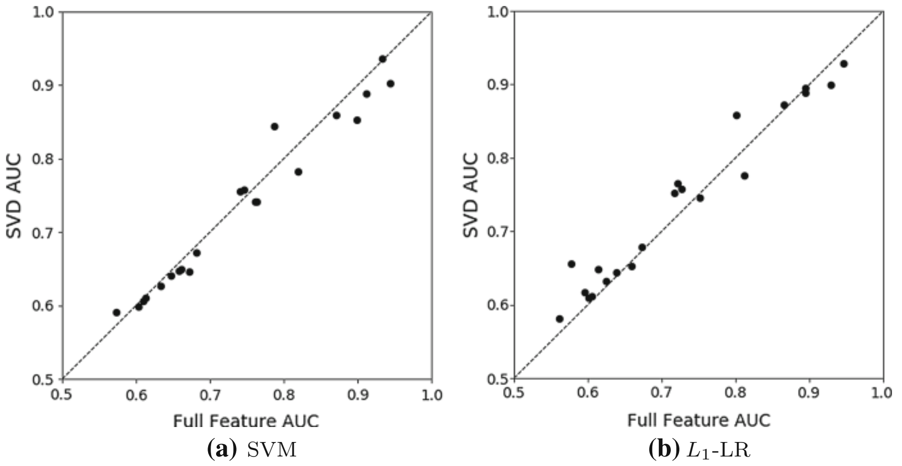


Fig. 11 SVD versus full feature performance using SVMs and L_1-LR, with k and C chosen using nested cross-validation. Each point represents the (full feature AUC, SVD AUC) pair for one classification task. Points falling below the $y = x$ line have better full feature performance, and vice versa. SVD is significantly worse for SVMs. On the other hand, SVD does provide substantial benefit for L_1-LR models: the median benefit from using SVD is 0.007, $z = 1.895$

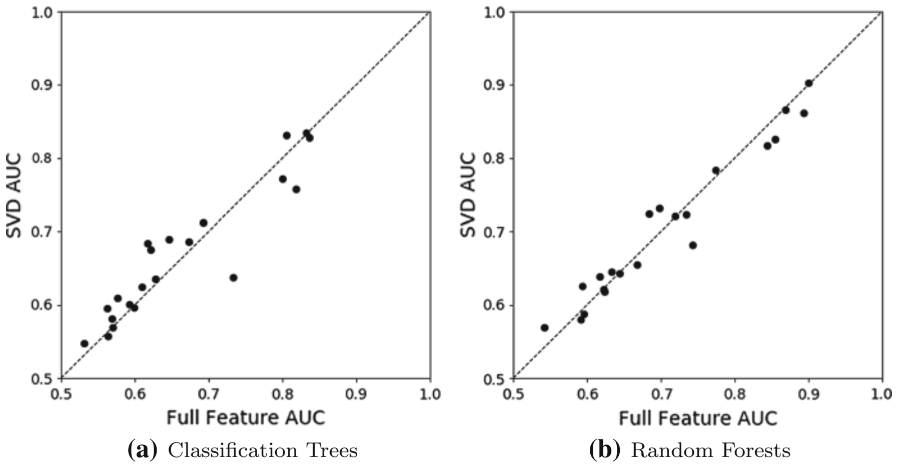


Fig. 12 SVD versus full feature performance using Classification Trees and Random Forests, with k and C chosen using nested cross-validation. Each point represents the (full feature AUC, SVD AUC) pair for one classification task. Points falling below the $y = x$ line have better full feature performance, and vice-versa. The differences between SVD and full feature performance are not statistically significant for these classifiers

4.5 Other DR methods

As discussed in Sect. 3.3, SVD is not the only MF-based DR algorithm. We now present results for NMF (Non-negative Matrix Factorization) and LDA (Latent Dirichlet Allocation).

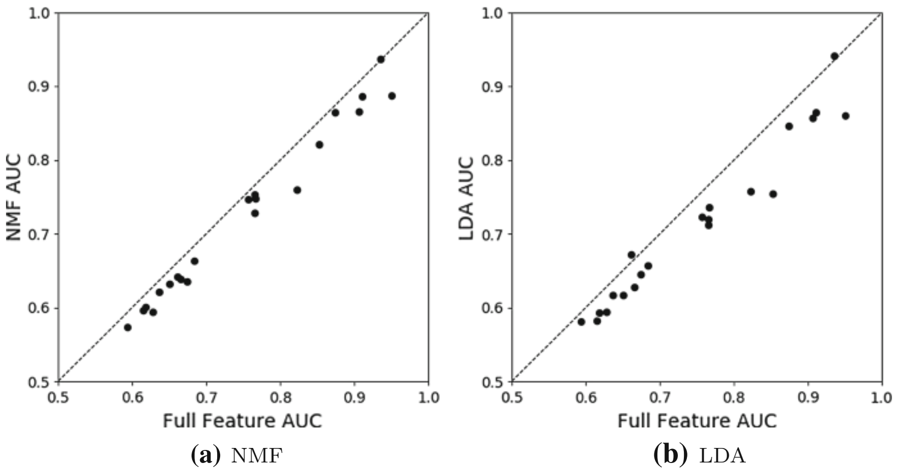


Fig. 13 DR versus full feature performance using NMF and LDA, with k and C chosen using nested cross-validation. Each point represents the (full feature AUC, DR AUC) pair for one classification task. Points falling below the $y = x$ line have better full feature performance, and vice versa. Similarly to the SVD features, using the DR features when modeling using L_2-LR yields significantly worse performance than using the full features

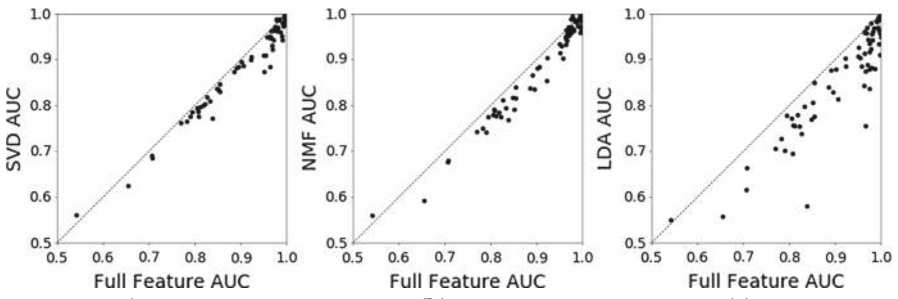
Figure 13a, b replicates the experiments from Sect. 4.2, with NMF and LDA replacing SVD. The results are very similar to those found using SVD. For NMF, the median benefit from using NMF is -0.020 and the Wilcoxon signed-rank test $z = -3.980$ (so, the full feature performance is better at $p = .01$). LDA also produces qualitatively similar results: the median difference between DR and full features is -0.033, $z = -3.910$ (the full features are significantly better at $p = .01$).

Note that the SVD performance is on average the highest out of the three DR techniques (using L_2-LR, the respective averages are .733 for SVD, .724 for NMF, and .712 for LDA). This is perhaps because of the advantages listed in Sect. 3.3; additionally, NMF and LDA come with many more parameters that could be tuned other than k. These methods may be improved with further tuning; of course, this would add to the total computation time.

This section's experiments with the alternative DR methods NMF and LDA not only strengthen our prior results regarding DR's performance relative to the original features, they also provide additional insights. Specifically, we still can't conclude that DR features perform better than original features; this additional preprocessing step frequently degrades predictive performance. Further, none of the DR methods perform better than L_2-LR on the full feature set. NMF and LDA, if anything, perform worse than SVD and thus the additional complexity they introduce (they are more time-consuming to compute and have more parameters to tune) is unnecessary.

4.6 Additional data sets and kNN

To ensure the above results were not simply due to the choice of data sets, we also replicated the same experiments on 15 additional large, sparse data sets, comprising



(a) SVD **(b)** NMF **(c)** LDA

Fig. 14 Plots of DR versus full performance for 15 additional data sets comprising 76 binary prediction tasks, using L_2-LR

76 total binary prediction tasks. Figure 14 compares the performance between DR and full features on the additional tasks, using L_2-LR as the classifier. In summary, the aggregate conclusions we draw are similar to those from the 21 Facebook data sets. Appendix C contains additional plots of results for these additional data sets using SVD, NMF, and LDA, and the remaining classifiers.

Off-the-shelf implementations of k-Nearest Neighbors models are generally not suitable for use with very large data sets. This is because the prediction time increases with the square of the number of instances. Indeed, training time for some of the largest feature sets that we studied exceeded 30 hours per fold. Therefore, because this method is not commonly used with the type of large data sets that we study, we do not include a comparison in our core results. However, we do include a comparison using all but the largest data sets/prediction tasks (including tasks from the additional data sets). The results are shown in Fig. 15.

The results are striking in that this is the only classifier for which the DR features are unequivocally superior to the results generated with the full feature sets. The median difference between the SVD and full features is .040, $z = 7.46$ (significant at $p < .01$); between NMF and full features is .027, $z = 6.68$ (significant at $p < .01$); and between LDA and full features is .042, $z = 5.11$ (significant at $p < .01$). This is not necessarily surprising, as kNN models are especially susceptible to the "curse of dimensionality": in very high-dimensional space, no two instances look similar. On the other hand, even the DR-feature kNN predictions perform worse than the best of the other classifiers.

Figure 16 summarizes the overall mean performance of all of the classifiers on the extended collection of data sets. (To facilitate a fair comparison with kNN, we include only the 77 smallest binary prediction tasks.) These results illuminate two key findings. First, L_2-LR on the full feature set dominates all of the methods tried so far. Feature-selection-oriented methods in general do not perform as well. Second, when used in conjunction with L_1-LR or random forest modeling, DR has better performance relative to the full feature set than it does when used with L_2-LR. We can't confidently conclude that using the DR features is *worse* than using the full feature set (although, neither can we conclude that it is better). Thus, *if* you are going

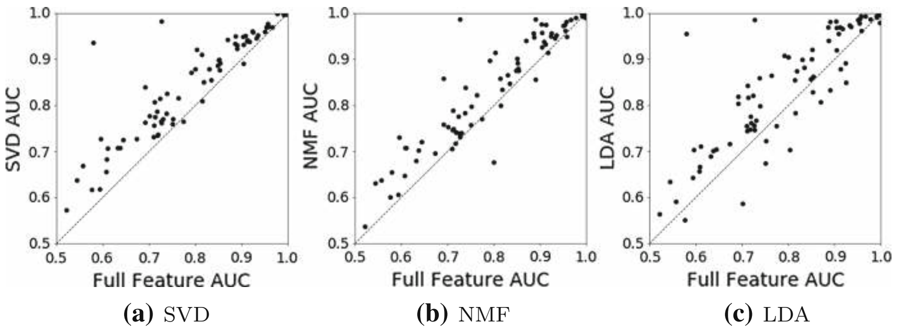


Fig. 15 DR versus full performance using kNN, for all but the largest Facebook and additional tasks (77 total tasks)

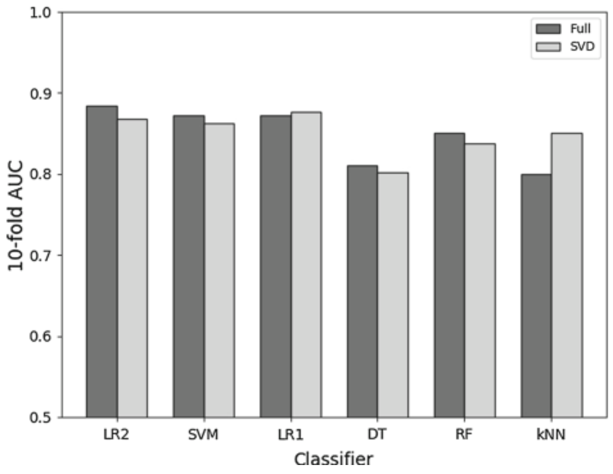


Fig. 16 Mean performance of SVD and full feature sets across the 77 smallest tasks for each classifier. L_2-LR with the full feature set has the best overall performance, with SVM/full feature and L_1-LR/SVD also in the top three. Overall, decision tree and kNN models have the worst performance

to choose a feature-selection-based learning algorithm for some other reason, you may well want to consider using DR features.

4.7 Binary prediction summary

This paper has presented a thorough investigation of DR for binary classification on 21 data sets representing predictive modeling tasks drawing from very similar sets of features, as well as a broader investigation across different data sets of varied sizes, shapes, sparsity levels, and base rates. Table 3 summarizes the performance of the various DR methods and modeling algorithms across the entire suite of 97 predictive tasks. In summary, L_2-LR and SVM models have the best performance overall and they perform significantly better when using the original, full feature set. The classifiers that perform implicit feature selection—L_1-LR, decision trees, and random forests—have mixed results between DR and full-feature modeling. We also

Table 3 Summary of DR versus full features, for various modeling algorithms and DR methods

	DR method	Median DR benefit	z-score	DR better	Full better
L_2-LR	SVD	– 0.012	– 7.862(***)		✓
	NMF	– 0.019	– 7.738(***)		✓
	LDA	– 0.038	– 8.282(***)		✓
SVM	SVD	– 0.005	– 3.165(***)		✓
	NMF	– 0.022	– 5.514(***)		✓
	LDA	– 0.037	– 7.716(***)		✓
L_1-LR	SVD	0.002	2.639(***)	✓	
	NMF	– 0.004	– 3.587(***)		✓
	LDA	– 0.023	– 6.471(***)		✓
Tree	SVD	– 0.001	– 1.225		
	NMF	0.019	7.378(***)	✓	
	LDA	0.011	2.977(***)	✓	
RF	SVD	– 0.010	– 3.557(***)		✓
	NMF	0.004	4.482(***)	✓	
	LDA	– 0.012	– 3.888(***)		✓
kNN	SVD	0.081	7.248(***)	✓	
	NMF	0.062	6.238(***)	✓	
	LDA	0.071	5.238(***)	✓	

Note that the kNN statistics here are based on a subset of the 97 tasks

note that L_1-LR with SVD features is among the top classification methods that we experimented with. Finally, kNN sees the greatest relative benefit when modeling with DR features; however, recall from Fig. 16 that kNN's classifications are not competitive (and furthermore, kNN is not really suited to very large data sets). DR has marginal benefit at best when predictive performance is the goal, and is very often worse. We therefore conclude that it should be avoided in order to keep the predictive modeling process as simple as possible, unless there is strong evidence that it is beneficial for one's particular task. Also note that these results are similar when using other evaluation metrics. Tables 8, 9, and 10 in Appendix B replicate Table 3 using the H-measure, AUCH, and lift at 5%, respectively.

Thus, we are able to do a complete comparison of all predictive modeling methods across all of the tasks (21 Facebook modeling tasks and 76 from the other data sets). Figure 17 shows the L_2-LR performance using the full feature set versus performance for each of the other combinations of DR type and classifier. These results emphasize that using the full feature set yields superior performance, especially for L_2-LR. Thus, we can recommend L_2-LR as the default baseline method of choice when doing binary classification with large, sparse data. This concurs with recent independent results experimenting with alternative classification techniques (but not DR) for similar data to those we use here (De Cnudde et al. 2017), as well as with prior observations from practice (e.g., Raeder et al. 2013; Dalessandro et al. 2014).

Table 4 shows the average performance for each feature set/classifier pair, across four evaluation metrics and 97 classification tasks (because we did not run kNN for

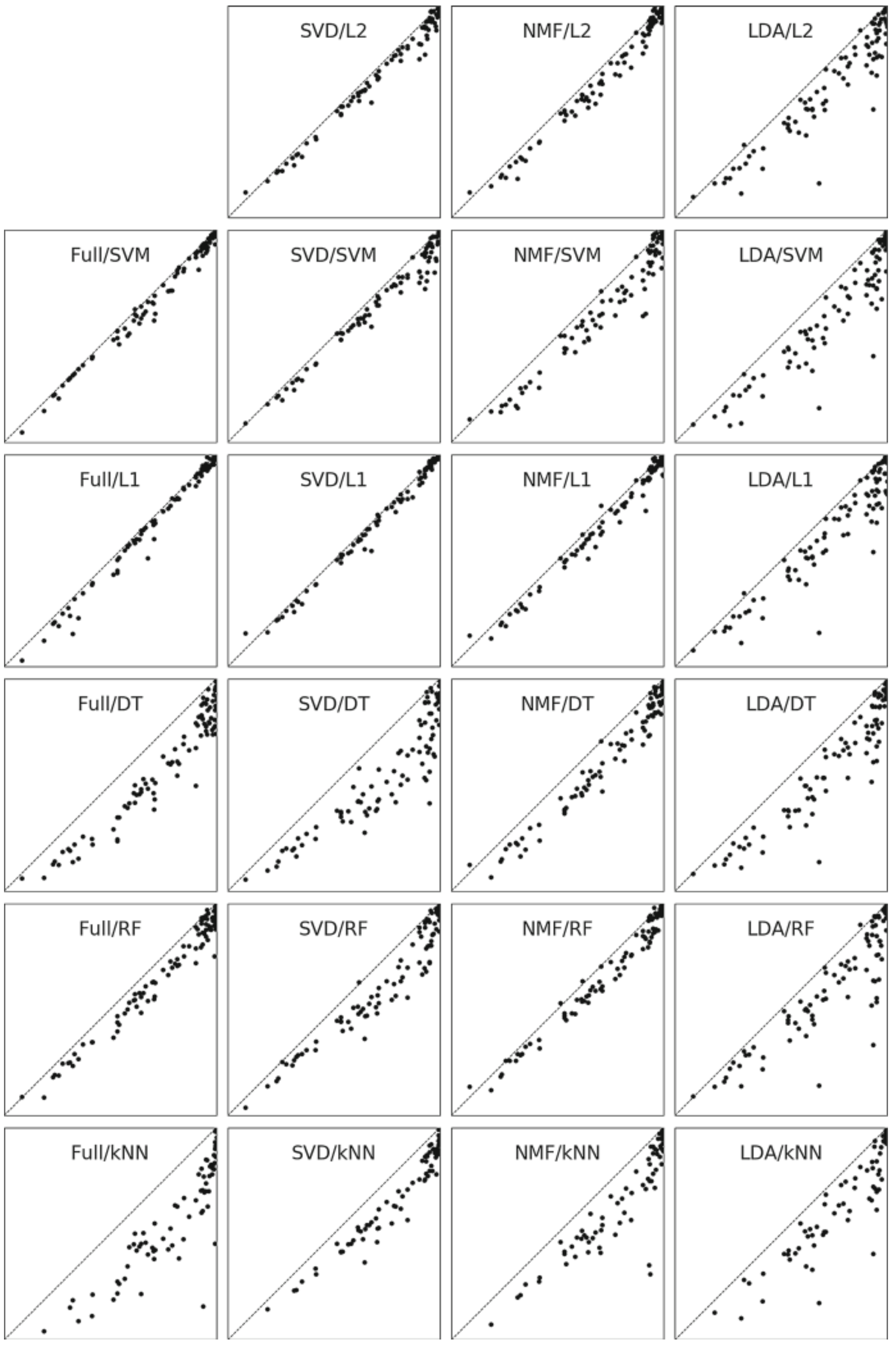


Fig. 17 Each chart plots L_2-regularized logistic regression performance on the x-axis versus performance using each of the other feature set/modeling algorithm combinations on the y-axis. Points falling below the $y = x$ line are where L_2-LR on the full feature set is better. Clearly this method is superior to every other combination of feature set and modeling algorithm in terms of predictive performance. Note that the kNN statistics here are based on a subset of the 97 tasks

all tasks, we have excluded it from this table). A result that immediately stands out is that for each metric L_2-LR has the best performance in the majority of tasks; no other method even comes close. This method also has the highest average AUC across all tasks (.88), and the lowest average rank among the 20 feature/method combinations (4.15). If one were to want to use DR for another reason besides improving predictive performance, among the DR feature sets, SVD shows the best performance, especially when used with L_1-LR; it had the best performance in 11 classification tasks, the second highest mean AUC (.88) and the second-lowest average rank (of an SVD method, 10.44). These results are roughly equivalent across all four evaluation metrics and therefore were not due to a lucky choice of metric. Note that presenting results in this fashion stands the risk of a severe multiple comparisons problem (not in favor of the winner): comparing all other methods to L_2-LR means that at least one of those other methods may be superior by chance. Even so, the results are quite striking.

These two sets of results enable us to make some conclusions that should prove useful for those wishing to use DR for binary classification. For the best performance overall, use L_2-LR with the full feature set and state-of-the-art selection of the regularization parameters. If it is desirable to use unsupervised matrix-factorization-based DR for any of the other benefits mentioned in Sect. 2 (such as potential data set compression or reduced modeling time across many tasks with the same feature set), SVD is the superior method, and it performs best if L_1-LR is the modeling algorithm used. Similarly, if you need to use feature selection, L_1-LR is the best feature-selection-based modeling algorithm on these data, and it works best when paired with features generated by SVD (rather than the full feature set). kNN and decision tree classifiers should be avoided as they have the lowest overall performance.

Furthermore, we emphasize that these results were generated using nested cross-validation to do a grid search over both the regularization parameter C (or other complexity parameters in the cases of classification trees, random forests, and kNN) and the number of DR components k. If NCV were not employed, we would have made different conclusions, as shown earlier.

4.8 Other tasks: numerical regression and multiclass classification

It is also worth investigating the efficacy of SVD for numerical regression and multiclass classification. Fortunately, the Facebook data set has 10 numerical target variables (which we had binarized for the previous sections), so we can study numerical regression with the same overall feature set as for the paper's main experiments. Although we have not done as deep of an investigation on these tasks, these more limited analyses suggest intriguing avenues for future research.

Figure 18 shows Pearson's r score for full features versus SVD resulting from numerical modeling via Ridge Regression for the 10 predictive tasks. The median benefit from SVD is .048 (and, in fact, SVD performs better than the full feature set for all but one target variable), so it appears that SVD does provide benefit for numerical regression. *This not only is in contrast to the results for classification, but also is counter to the guidance provided in the textbooks!*

Table 4 Rank comparison across all methods, for all target variables and feature sets

	AUC			H measure			AUCH			Lift at 5%		
	Wins	Mean	Mean rank	Wins	Mean	Mean rank	Wins	Mean	Mean rank	Wins	Mean	Mean rank
Full/L2LR	**58**	**0.88**	**4.15**	**58**	**0.61**	**3.85**	**61**	**0.89**	**3.97**	**43**	**8.49**	**4.76**
Full/SVM	4	0.87	8.54	7	0.58	9.47	4	0.88	8.96	10	8.11	9.91
Full/L1LR	7	0.87	5.30	6	0.58	5.11	5	0.88	5.24	16	8.23	8.67
Full/DT	0	0.81	7.74	0	0.44	7.86	0	0.82	7.53	0	6.16	8.55
Full/RF	1	0.85	8.41	1	0.51	7.56	1	0.86	8.30	5	6.98	8.35
SVD/L2LR	0	0.87	9.48	1	0.55	9.32	0	0.88	9.28	0	7.63	10.04
SVD/SVM	0	0.86	11.24	0	0.54	10.27	0	0.87	11.08	2	7.52	10.63
SVD/L1LR	11	0.88	10.44	11	0.58	10.93	11	0.89	10.29	6	8.16	10.10
SVD/DT	0	0.80	10.25	0	0.41	10.26	0	0.81	10.59	2	5.88	10.17
SVD/RF	0	0.84	11.31	1	0.50	12.09	0	0.85	11.23	4	7.03	9.70
NMF/L2LR	2	0.86	12.21	0	0.55	11.89	3	0.87	12.30	2	7.63	9.41
NMF/SVM	2	0.85	12.95	2	0.52	13.61	2	0.86	13.74	0	7.41	12.43
NMF/L1LR	3	0.87	12.65	6	0.56	12.21	3	0.88	13.01	2	7.87	10.81
NMF/DT	0	0.84	13.09	0	0.49	12.41	0	0.84	12.60	1	7.15	10.49
NMF/RF	7	0.86	12.69	2	0.54	12.59	5	0.87	12.67	2	7.57	11.39
LDA/L2LR	0	0.84	13.51	0	0.49	13.78	0	0.85	13.78	0	6.39	12.31
LDA/SVM	0	0.83	9.03	0	0.47	9.51	0	0.84	8.93	0	6.30	11.97
LDA/L1LR	1	0.84	13.70	2	0.49	13.28	1	0.85	12.67	0	6.40	12.75
LDA/DT	0	0.82	11.82	0	0.45	12.93	0	0.83	12.15	0	6.42	14.65
LDA/RF	1	0.83	11.48	0	0.47	11.09	1	0.84	11.69	2	6.61	12.89

The feature set/classifier combination with the winning performance in each category is highlighted in bold
L_2-LR using the full feature set is the best method for the majority of tasks (58 out of 97), and has the best mean AUC and mean rank across tasks. These results are relatively consistent across binary evaluation metrics

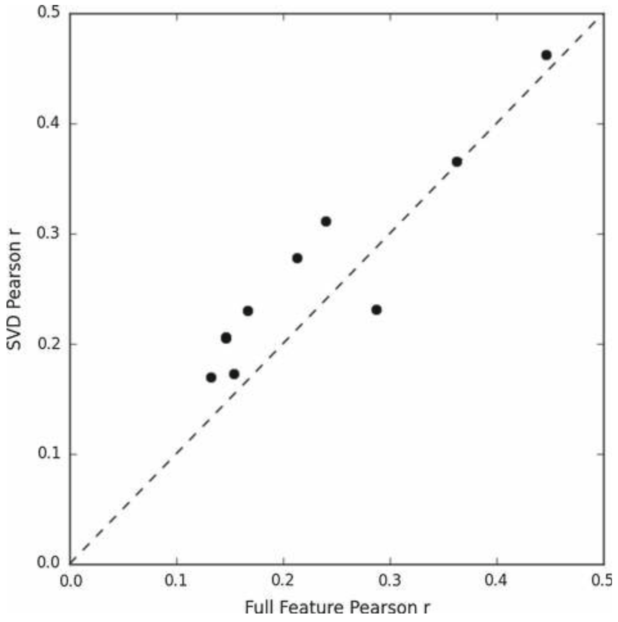


Fig. 18 Comparing Pearson's r score for numerical regression using SVD versus full features. Each point represents the (full feature r score, SVD r score) pair for one classification task. Points falling below the $y = x$ line have better full feature performance, and vice versa. SVD provides positive benefit for all but one of the ten numerical target variables

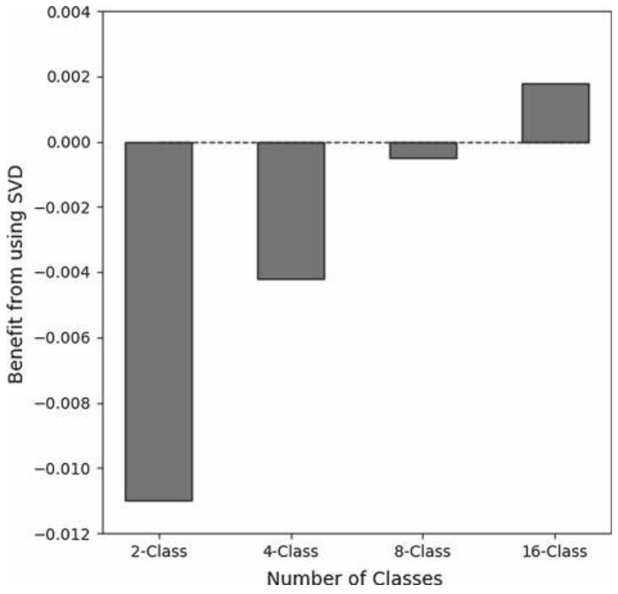


Fig. 19 Mean improvement in performance by using SVD for various numbers of classes. The amount of improvement that can be gained by using SVD increases with the number of classes in the data

In addition to numerical regression, the numerical target variables allow us to perform ordinal multiclass classification by grouping instances into roughly equal-sized classes based on their numerical value. Just as the results in prior sections grouped the instances into two classes, we grouped them into four, eight, and sixteen classes for the results below. The performance reported is the average AUC using one-versus-rest classification for each of the classes (with L_2-LR using nested cross-validation to select C and k). Figure 19 summarizes the average difference in performance across tasks between DR and full features versus the number of classes in the data.

Intriguingly, it appears that SVD provides more benefit as the number of classes in the data increases. Binary classification has the fewest possible values for the target variable to take on. Although ordinal multiclass classification and numerical regression are different (specifically, numerical regression orders the instances while multiclass classification does not consider the ordering of the classes while modeling), they both yield more possible values for the target variable. If one considers that ordinal multiclass classification with an increasing number of classes increasingly resembles numerical regression, then these results show a clear trend in the benefit versus harm of using DR.

5 Conclusion

Unsupervised DR is commonly used with the goal of improving predictive modeling performance. The theoretical motivation to use unsupervised DR is well-understood, but overlaps with the reason for using supervised regularization: both techniques leverage a bias/variance tradeoff. Textbooks largely recommend supervised regularization as the better alternative, but neither practitioners nor researchers seem to follow those recommendations—especially when applying predictive modeling to massive, sparse feature sets. Further, the textbooks that we surveyed largely treat this as an either/or proposition, and do not consider that regularization and DR can be (and usually are) used in conjunction. Indeed, in the applied predictive modeling literature, DR features are often used without mention of a comparison with using the original full set of features. This seeming confusion is particularly troubling in light of our empirical results: that using the full feature set, with state-of-the-art regularization, usually gives better generalization performance.

This paper satisfies the need for a reference on whether and when to use DR for binary classification using large, sparse feature sets. It fills in several holes in the existing literature. Namely, past analyses don't treat DR and regularization as potentially complementary techniques, haven't experimented with different modeling methods, have used much smaller data sets, and have not focused on the high-dimensional, sparse data for which researchers and practitioners tend to gravitate to DR.

The experiments in this paper consist of a comparison on a test-bed comprising 21 varied binary classification tasks using (subsets of) the same massive feature set, as well as 76 supplementary tasks from a total of 15 publicly available data sets. We find that the methodology used to select and evaluate crucial modeling parameters has a substantial effect on the relative performance of DR versus full features. When using

L_2-LR (or SVMs) and a state-of-the-art hyperparameter tuning and evaluation methodology (nested cross-validation) for selecting the regularization parameter, SVD does not improve performance. However, when using less sophisticated model selection and evaluation procedures, SVD can sometimes have better performance, for instance, if no supervised regularization is employed at all, or if default values are used for the regularization parameter and/or the number of SVD components. However, as expected, these methodologies result in worse overall performance. Furthermore, classifiers—like tree induction and L_1-LR—that effectively incorporate feature selection can perform better when using DR features, although these classifiers tend to have worse performance overall on these data sets. kNN classifiers universally improve when using DR features, but again have worse performance overall than the top-performing classifiers. These results are consistent across two DR techniques in addition to SVD—NMF and LDA—and are also consistent across four evaluation metrics: AUC, H measure, AUCH, and lift over random targeting at the top 5% of instances.

In addition to the practical implications directly related to the use of DR, our results help benchmark other modeling choices. L_2-LR with full features and state-of-the-art selection of the regularization parameter is hard to beat for this type of large, sparse data for binary classification (cf., Raeder et al. 2013; Dalessandro et al. 2014; De Cnudde et al. 2017). This suggests that this method should be considered to be the baseline against which to compare when faced with such data. L_1-LR with SVD features is also a strong contender, and is the best option if using DR is desirable for other reasons. SVD is the best-performing DR method of the three that we evaluated.

The results in this paper are largely limited to one sort of data (large, sparse data mainly generated by human behavior), and one type of predictive modeling task (binary classification). This particular combination of data type and task has become quite common in the predictive modeling literature as well as in industry use, and our results are relatively conclusive; however, we do not mean to imply that unsupervised DR will not work for *any* data type or modeling task. In Sect. 4.8, we show preliminary results for a few other predictive tasks where DR does seem to add substantial value: numerical regression and ordinal, multi-class classification. Furthermore, there is an intriguing trend in the results for ordinal multi-class classification: increasing the number of classes leads to increasing benefit from DR. While the results from Sect. 4.8 should not be taken as a comprehensive study due to the more limited nature of the investigation and smaller number of data points, this observation provides a basis for interesting future work.

An additional limitation is that this study focuses on unsupervised matrix-factorization-based DR techniques. Alternative methods include supervised dimensionality reduction techniques such as Linear Discriminant Analysis (Friedman et al. 2001) or neural-network-based approaches such as deep learning or word2vec (Mikolov et al. 2013). We focused on unsupervised DR for reasons mentioned above: using unsupervised DR as a preprocessing step is relatively common, and most other methods are more computationally complex and don't have implementations for sparse feature sets widely available. However, we are not aware of a systematic study such as this one evaluating supervised DR or other methods, and such a study would make another good avenue for future work.

There are a multitude of major and minor variants and modifications that can be made to the predictive modeling process. Ultimately the core lesson of this paper can be summarized as one of basic system design principles: exercise caution when adding complexity via a dimensionality reduction step to the predictive modeling process, even if one feels confident that DR will benefit the performance. We have both established that this principle is frequently violated in the predictive modeling literature, and conducted a comprehensive study showing that this violation is a mistake that leads to weaker results than might otherwise be possible. However, it is important that our results don't cause other researchers to make the same mistake in reverse! We show a few classes of model selection methodologies and predictive tasks—numerical regression and multiclass classification—where DR does add value, further supporting our main message: don't rely on intuition or results from other contexts, but compare design alternatives carefully. Unless there is a very good reason, applied researchers and practitioners should keep the process as simple as possible so as not to inadvertently harm predictive performance or perform unnecessary computation.

Acknowledgements We thank Michal Kosinski, David Stillwell, and Thore Graepel for sharing the Facebook Like data, and all the authors who have made their data and code publicly available, Thayumanav Jayadevan for help processing the data sets, Andre Meyer for a Faculty Fellowship, and Richard Tsai and the NYU Fubon Center for supporting the research environment at NYU Stern.

Appendix A: data set details

Table 5 shows the number of users and Likes currently available for each binary target variable and the base rates for each of the variables provided by Kosinski et al. (2013).

Table 6 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 resulting base rates.

Table 7 summarizes the characteristics of the 15 additional large, sparse data sets included in this paper besides the Facebook data.

Appendix B: results using other evaluation metrics

Table 8 reports the median difference in H measure between DR and full feature performance, for each classifier/DR method pair, averaged across 97 tasks (or 77, in the case of kNN). It also reports the *z*-statistic from a Wilcoxon signed-rank test, the significance level, and whether DR or full features are significantly better.

Table 9 reports the median difference in AUCH between DR and full feature performance, for each classifier/DR method pair, averaged across 97 tasks (or 77, in the case of kNN). It also reports the *z*-statistic from a Wilcoxon signed-rank test, the significance level, and whether DR or full features are significantly better (Fig. 20).

Table 10 reports the median difference in lift over random targeting at the top 5% of instances between DR and full feature performance, for each classifier/DR method

Table 5 Data details for binary target variables

Target	n	d	Base rate (%)
Single versus in relationship	162,980	179,605	47
Parents together at 21	2088	84,813	50
Smokes cigarettes	3690	118,643	25
Drinks alcohol	3667	118,604	51
Uses drugs	2711	104,869	17
Caucasian versus African American	2645	100,506	95
Christianity versus Islam	3625	105,023	95
Democrat versus Republican	12,936	147,759	59
Gay	25,813	167,307	5
Lesbian	30,087	173,375	3
Gender	210,004	179,605	61

Table 6 Data details for numeric target variables

Target	n	d	Threshold	Base rate (%)
Satisfaction with life	6512	141,815	5	37
Intelligence	6129	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 7 Characteristics of additional "Big Data" data sets

Data set	n	d	Base rate	% nonzero entries
Book	285,089	62,107	46%	.0047
Flickr	88,787	100,000	27%	.0035
Movies	11,916	7619	29%	.23
Ta-Feng	23,812	32,266	54%	.10
URL	358,073	100,000	33%	.03
Movielens-users	6040	3706	(Varies)	4.5
Movielens-movies	3706	6040	(Varies)	4.5
CiteSeer	100,000	105,354	0.1%	.001
Daily and sports activities	9120	5625	(Varies)	5
DeliciousMIL	12,234	8520	(Varies)	.1
E-commerce	29,998	36,111	78%	.006
Farm Ads	4143	54,877	53%	.3
Gisette	7000	5000	50%	13.0
IMDB	95,220	100,000	0.5%	.01
p53 mutants	31,420	5407	0.5%	5
Reuters	111,740	21,531	(Varies)	.2

Table 8 Summary of DR versus full features, for various modeling algorithms and DR methods, measured using the H measure

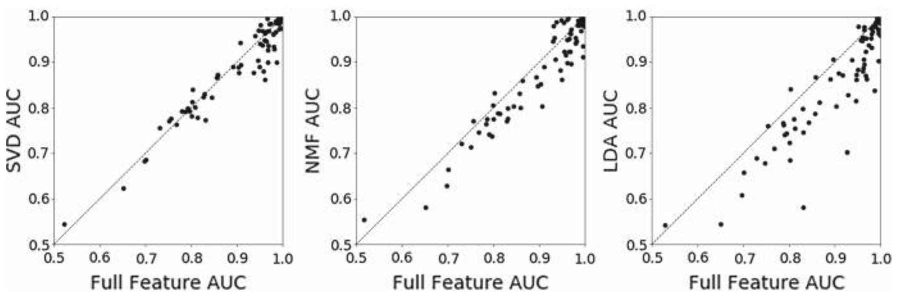
	DR method	Median DR benefit	z-score	DR better	Full better
L_2-LR	SVD	– 0.038	– 8.136(∗∗∗)		✓
	NMF	– 0.044	– 7.994(∗∗∗)		✓
	LDA	– 0.082	– 8.473(∗∗∗)		✓
SVM	SVD	– 0.023	– 4.611(∗∗∗)		✓
	NMF	– 0.046	– 5.896(∗∗∗)		✓
	LDA	– 0.077	– 7.918(∗∗∗)		✓
L_1-LR	SVD	0.000	0.743		
	NMF	– 0.011	– 3.338(∗∗∗)		✓
	LDA	– 0.056	– 6.554(∗∗∗)		✓
Tree	SVD	– 0.016	– 2.233(∗∗)		✓
	NMF	0.023	6.532(∗∗∗)	✓	
	LDA	0.010	1.379		
RF	SVD	– 0.016	– 2.693(∗∗∗)		✓
	NMF	0.011	4.244(∗∗∗)	✓	
	LDA	– 0.034	– 3.373(∗∗∗)		✓
kNN	SVD	0.081	7.248(∗∗∗)	✓	
	NMF	0.062	6.238(∗∗∗)	✓	
	LDA	0.071	5.283(∗∗∗)	✓	

Note that the kNN statistics here are based on a subset of the 97 tasks

Table 9 Summary of DR versus full features, for various modeling algorithms and DR methods, measured using AUCH

	DR method	Median DR benefit	z-score	DR better	Full better
L_2-LR	SVD	– 0.011	– 7.935(∗∗∗)		✓
	NMF	– 0.018	– 7.800(∗∗∗)		✓
	LDA	– 0.034	– 8.337(∗∗∗)		✓
SVM	SVD	– 0.004	– 3.237(∗∗∗)		✓
	NMF	– 0.020	– 5.798(∗∗∗)		✓
	LDA	– 0.033	– 7.659(∗∗∗)		✓
L_1-LR	SVD	0.001	2.700(∗∗∗)	✓	
	NMF	– 0.003	– 3.524(∗∗∗)		✓
	LDA	– 0.021	– 6.345(∗∗∗)		✓
Tree	SVD	– 0.002	– 1.319		
	NMF	0.018	7.389(∗∗∗)	✓	
	LDA	0.013	2.855(∗∗∗)	✓	
RF	SVD	– 0.009	– 3.557(∗∗∗)		✓
	NMF	0.003	4.334(∗∗∗)	✓	
	LDA	– 0.010	– 4.104(∗∗∗)		✓
kNN	SVD	0.081	7.248(∗∗∗)	✓	
	NMF	0.062	6.237(∗∗∗)	✓	
	LDA	0.071	5.283(∗∗∗)	✓	

Note that the kNN statistics here are based on a subset of the 97 tasks



(a) SVD **(b)** NMF **(c)** LDA

Fig. 20 Plots of DR versus full performance for 15 additional data sets comprising 76 binary prediction tasks, using SVMs

pair, averaged across 97 tasks (or 77, in the case of kNN). It also reports the z-statistic from a Wilcoxon signed-rank test, the significance level, and whether DR or full features are significantly better.

Table 10 Summary of DR versus full features, for various modeling algorithms and DR methods, measured using lift at the top 5% of instances

	DR method	Median DR benefit	z-score	DR better	Full better
L_2-LR	SVD	– 0.154	– 4.825(∗∗∗)		✓
	NMF	– 0.105	– 4.386(∗∗∗)		✓
	LDA	– 0.781	– 7.331(∗∗∗)		✓
SVM	SVD	– 0.011	– 2.152(∗∗)		✓
	NMF	– 0.226	– 4.359(∗∗∗)		✓
	LDA	– 0.536	– 7.142(∗∗∗)		✓
L_1-LR	SVD	0.000	– 1.594		
	NMF	– 0.000	– 1.829(∗)		✓
	LDA	– 0.413	– 5.998(∗∗∗)		✓
Tree	SVD	0.000	– 0.980		
	NMF	0.057	2.785(∗∗∗)	✓	
	LDA	0.000	– 0.077		
RF	SVD	0.000	– 1.661(∗)		✓
	NMF	0.000	– 0.727		
	LDA	– 0.085	– 3.691(∗∗∗)		✓
kNN	SVD	0.545	4.942(∗∗∗)	✓	
	NMF	0.333	3.401(∗∗∗)	✓	
	LDA	0.272	2.707(∗∗∗)	✓	

Note that the kNN statistics here are based on a subset of the 97 tasks

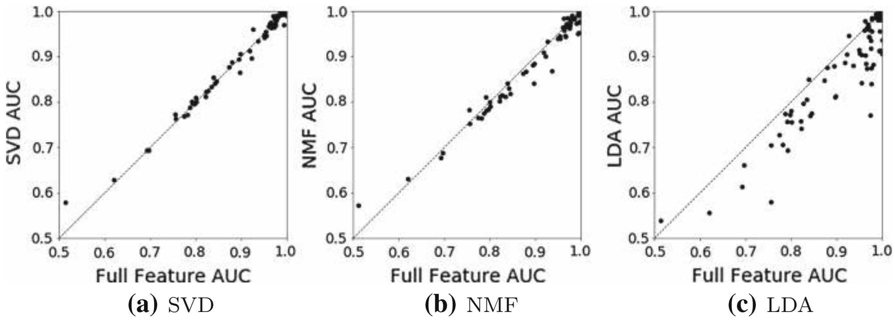
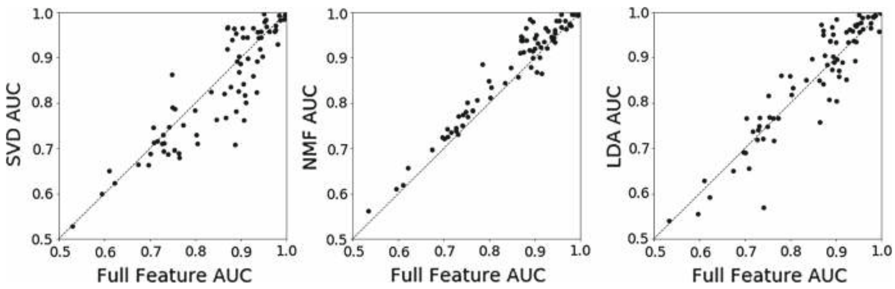


Fig. 21 Plots of DR versus full performance for 15 additional data sets comprising 76 binary prediction tasks, using L_1-LR

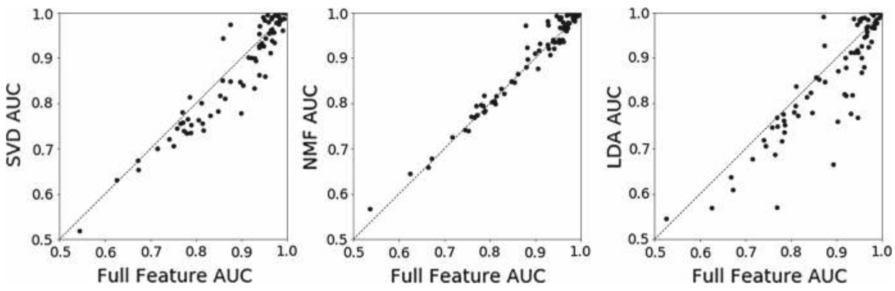
Appendix C: individual results for additional data sets

Each plot in this section shows DR versus full-feature performance on the 76 incremental tasks coming from 15 publicly available data sets, for one classifier/DR method pair (Figs. 21, 22, 23). The results for L_2-LR are in Fig. 14 in Sect. 4.6.



(a) SVD **(b)** NMF **(c)** LDA

Fig. 22 Plots of DR versus full performance for 15 additional data sets comprising 76 binary prediction tasks, using Classification Trees



(a) SVD **(b)** NMF **(c)** LDA

Fig. 23 Plots of DR versus full performance for 15 additional data sets comprising 76 binary prediction tasks, using Random Forests

References

Ahn H, Eunsup E, Han I (2007) Extracting underlying meaningful features and canceling noise using independent component analysis for direct marketing. Expert Syst Appl 33(1):181–191

Altun K, Barshan B (2010) Human activity recognition using inertial/magnetic sensor units. In: International workshop on human behavior understanding. Springer, Berlin, pp 38–51

Altun K, Barshan B, Tunçel O (2010) Comparative study on classifying human activities with miniature inertial and magnetic sensors. Pattern Recognit 43(10):3605–3620

Amini M, Usunier N, Goutte C (2009) Learning from multiple partially observed views-an application to multilingual text categorization. In: Advances in neural information processing systems, pp 28–36

Arulogun OT, Omidiora EO, Waheed MA, Fakolujo OA, Olaniyi OM (2012) On the classification of gasoline-fuelled engine exhaust fume related faults using electronic nose and principal component analysis. Comput Inf Syst Dev Inform Allied Res J 3(2):1–8

Barshan B, Yüksek MC (2014) Recognizing daily and sports activities in two open source machine learning environments using body-worn sensor units. Comput J 57(11):1649–1667

Bellman R (1961) Adaptive control processes: a guided tour, vol 4. Princeton University Press, Princeton

Bíró I, Szabó J, Benczúr AA (2008) Latent Dirichlet allocation in web spam filtering. In: Proceedings of the 4th international workshop on adversarial information retrieval on the web, pp 29–32

Bishop CM (2006) Pattern recognition and machine learning, vol 1. Springer, New York

Blei DM, Ng AY, Jordan MI (2003) Latent Dirichlet allocation. J Mach Learn Res 3:993–1022

Blum AL, Langley P (1997) of relevant features and examples in machine learning. Artif Int 97(1):245–271

Burl MC, Asker L, Smyth P, Fayyad U, Perona P, Crumpler L, Aubele J (1998) Learning to recognize volcanoes on Venus. Mach Learn 30(2):165–194

Cai J, Xu P, Tang H, Sun L (2013) An improved selective ensemble method for spam filtering. In: Proceedings of the 2013 15th IEEE international conference on communication technology (ICCT), pp 743–747

Cha M, Mislove A, Gummadi KP (2009) A measurement-driven analysis of information propagation in the Flickr social network. In: Proceedings of the 18th international conference on World Wide Web, pp 721–730

Chen D, Fraiberger SP, Moakler R, Provost F (2017) Enhancing transparency and control when drawing data-driven inferences about individuals. Big Data 5(3):197–212

Cios KJ, Pedrycz W, Swiniarski RW, Kurgan L (2007) Data mining: a knowledge discovery approach. Springer, Berlin 2007

Coussement K, Van den Poel D (2008) Integrating the voice of customers through call center emails into a decision support system for churn prediction. Inf Manag 45(3):164–174

Dalessandro B, Chen D, Raeder T, Perlich C, Williams MH, Provost F (2014) Scalable hands-free transfer learning for online advertising. In: Proceedings of the 20th ACM SIGKDD international conference on knowledge discovery and data mining, pp 721–730

Danziger SA, Swamidass SJ, Zeng J, Dearth LR, Lu Q, Chen JH, Cheng J, Hoang VP, Saigo H, Luo R (2006) Functional census of mutation sequence spaces: the example of p53 cancer rescue mutants. IEEE/ACM Trans on Comput Biol Bioinform (TCBB) 3(2):114–125

Danziger SA, Zeng J, Wang Y, Brachmann RK, Lathrop RH (2007) Choosing where to look next in a mutation sequence space: active learning of informative p53 cancer rescue mutants. Bioinformatics 23(13):i104–i114

Danziger SA, Baronio R, Ho L, Hall L, Salmon K, Hatfield GW, Kaiser P, Lathrop RH (2009) Predicting positive p53 cancer rescue regions using most informative positive (MIP) active learning. PLoS Comput Biol 5(9):e1000498

De Cnudde S, Martens D, Evgeniou T, Provost F (2017) A benchmarking study of classification techniques for behavioral data. Technical report 2017-005, University of Antwerp, Faculty of Applied Economics

Deegalla S, Bostrom H (2006) Reducing high-dimensional data by principal component analysis vs. random projection for nearest neighbor classification. In: Proceedings of the 5th international conference on machine learning and applications (ICMLA), pp 245–250

Der MF, Saul LK, Savage S, Voelker GM (2014) 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, pp 1759–1768

Dheeru D, Karra Taniskidou E (2017) UCI machine learning repository. http://archive.ics.uci.edu/ml. Accessed 18 Aug 2018

Domingos P (2012) A few useful things to know about machine learning. Commun ACM 55(10):78–87

Forman G (2003) An extensive empirical study of feature selection metrics for text classification. J Mach Learn Res 3:1289–1305

Friedman J (1997) On bias, variance, 0/1 loss, and the curse-of-dimensionality. Data Min Knowl Discov 1(1):55–77

Friedman J, Hastie T, Tibshirani R (2001) The elements of statistical learning, vol 1. Springer series in statistics. Springer, Berlin

Fruergaard BO, Hansen TJ, Hansen LK (2013) Dimensionality reduction for click-through rate prediction: dense versus sparse representation. arXiv preprint arXiv:1311.6976

Guyon I, Elisseeff A (2003) An introduction to variable and feature selection. J Mach Learn Res 3:1157–1182

Guyon I, Gunn S, Ben-Hur A, Dror G (2005) Result analysis of the NIPS 2003 feature selection challenge. In: Advances in neural information processing systems, pp 545–552

Guyon I, Lemaire V, Boullé M, Dror G, Vogel D (2009) Analysis of the KDD Cup 2009: fast scoring on a large orange customer database. In: KDD Cup, pp 1–22

Hand DJ (2009) Measuring classifier performance: a coherent alternative to the area under the ROC curve. Mach Learn 77(1):103–123

Hand DJ, Mannila H, Smyth P (2001) Principles of data mining. MIT Press, Cambridge

Harper FM, Konstan JA (2016) The Movielens datasets: history and context. ACM Trans Interact Int Syst (TIIS) 5(4):19

Hoff PD (2007) Model averaging and dimension selection for the singular value decomposition. J Am Stat Assoc 102(478):674–685

Hu J, Zeng HJ, Li H, Niu C, Chen Z (2007) Demographic prediction based on user's browsing behavior. In: Proceedings of the 16th international conference on World Wide Web, pp 151–160

Izenman AJ (2009) Modern multivariate statistical techniques: regression, classification and manifold learning. Springer, Berlin

James G, Witten D, Hastie T, Tibshirani R (2013) An introduction to statistical learning, vol 6. Springer, Berlin

Jensen DD, Cohen PR (2000) Multiple comparisons in induction algorithms. Mach Learn 38(3):309–338

Jones E, Oliphant T, Peterson P et al (2001) SciPy: open source scientific tools for Python. http://www.scipy.org/. Accessed 26 Oct 2014

Karypis G, Han EHS (2000) Fast supervised dimensionality reduction algorithm with applications to document categorization & retrieval. In: Proceedings of the ninth international conference on information and knowledge management, pp 12–19

Khan RM, Luk CH, Flinker A, Aggarwal A, Lapid H, Haddad R, Sobel N (2007) Predicting odor pleasantness from odorant structure: pleasantness as a reflection of the physical world. J Neurosci 27(37):10015–10023

Kim YS, Street WN, Russell GJ, Menczer F (2005) Customer targeting: a neural network approach guided by genetic algorithms. Manag Sci 51(2):264–276

Kosinski M, Stillwell D, Graepel T (2013) Private traits and attributes are predictable from digital records of human behavior. Proc Natl Acad Sci 110(15):5802–5805

Langley P (2000) Crafting papers on machine learning. In: ICML, pp 1207–1216

Lee DD, Seung HS (1999) Learning the parts of objects by non-negative matrix factorization. Nature 401(6755):788–791

Lifshits Y, Nowotka D (2007) Estimation of the click volume by large scale regression analysis. In: Computer science—theory and applications, pp 216–226

Liu H, Motoda H (1998) Feature extraction, construction and selection: a data mining perspective. Springer, Berlin

López M, Ramírez J, Górriz JM, Álvarez I, Salas-Gonzalez D, Segovia F, Chaves R, Padilla P, Gómez-Río M (2011) Principal component analysis-based techniques and supervised classification schemes for the early detection of Alzheimer's disease. Neurocomputing 74(8):1260–1271

Martens D, Provost F, Clark J, Junqué de Fortuny E (2016) Mining massive fine-grained behavior data to improve predictive analytics. MIS Q 40(4):869–888

Ma J, Saul LK, Savage S, Voelker GM (2009) Identifying suspicious URLs: an application of large-scale online learning. In: Proceedings of the 26th annual international conference on machine learning, pp 681–688

Mikolov T, Sutskever I, Chen K, Corrado GS, Dean J (2013) Distributed representations of words and phrases and their compositionality. In: Advances in neural information processing systems, pp 3111-3119

Owen AB, Perry PO (2009) Bi-cross-validation of the SVD and the nonnegative matrix factorization. Ann Appl Stat 3(2):564–594

Pechenizkiy M, Tsymbal A, Puuronen S (2004) PCA-based feature transformation for classification: issues in medical diagnostics. In: Proceedings of the 17th IEEE symposium on computer-based medical system (CBMS 2004), pp 535–540

Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, Blondel M, Prettenhofer P, Weiss R, Dubourg V, Vanderplas J, Passos A, Cournapeau D, Brucher M, Perrot M, Duchesnay E (2011) Scikit-learn: machine learning in Python. J Mach Learn Res 12:2825–2830

Provost F, Fawcett T (2013) Data science for business: what you need to know about data mining and data-analytic thinking. O'Reilly Media Inc., Newton

Raeder T, Perlich C, Dalessandro B, Stitelman O, Provost F (2013) Scalable supervised dimensionality reduction using clustering. In: Proceedings of the 19th ACM SIGKDD international conference on knowledge discovery and data mining, pp 1213–1221

Saeys Y, Inza I, Larrañaga P (2007) A review of feature selection techniques in bioinformatics. Bioinformatics 23(19):2507–2517

Scharf LL (1991) The SVD and reduced rank signal processing. Signal Process 25(2):113–133

Shahnaz F, Berry MW, Pauca VP, Plemmons RJ (2006) Document clustering using nonnegative matrix factorization. Inf Process Manag 42(2):373–386

Shmueli G, Koppius OR (2011) Predictive analytics in information systems research. MIS Q 35(3):553–572

Soleimani H, Miller DJ (2016) Semi-supervised multi-label topic models for document classification and sentence labeling. In: Proceedings of the 25th ACM international conference on information and knowledge management, pp 105–114

Subasi A, Gursoy MI (2010) EEG signal classification using PCA, ICA, LDA and support vector machines. Expert Syst Appl 37(12):8659–8666

Tan PN, Steinbach M, Kumar V (2006) Introduction to data mining. Pearson Education, New Delhi

Thorleuchter D, Van den Poel D, Prinzie A (2012) Analyzing existing customers' websites to improve the customer acquisition process as well as the profitability prediction in B-to-B marketing. Expert Syst Appl 39(3):2597–2605

Tremblay MC, Berndt DJ, Luther SL, Foulis PR, Dustin D, French DD (2009) Identifying fall-related injuries: text mining the electronic medical record. Inf Technol Manag 10(4):253–265

Turk M, Pentland A (1991) Eigenfaces for recognition. J Cognit Neurosci 3(1):71–86

Van der Maaten L, Postma E, Van Den Herik J (2009) Dimensionality reduction: a comparative review. Technical report TiCC TR 2009-005, Tilburg centre for Creative Computing, Tilburg University

West M, Blanchette C, Dressman H, Huang E, Ishida S, Spang R, Zuzan H, Olson JA, Marks JR, Nevins JR (2001) Predicting the clinical status of human breast cancer by using gene expression profiles. Proc Natl Acad Sci 98(20):11462–11467

Westad F, Hersletha M, Lea P, Martens H (2003) Variable selection in PCA in sensory descriptive and consumer data. Food Qual Prefer 14(5):463–472

Whitman B (2003) Semantic rank reduction of music audio. In: 2003 IEEE workshop on applications of signal processing to audio and acoustics, pp 135–138

Wilcoxon F, Katti SK, Wilcox RA (1963) Critical values and probability levels for the Wilcoxon rank sum test and the Wilcoxon signed rank test. In: Institute of Mathematical Statistics (ed) Selected tables in mathematical statistics, vol 1. American Mathematical Society, Providence, pp 171-259

Xing D, Girolami M (2007) Employing latent Dirichlet allocation for fraud detection in telecommunications. Pattern Recognit Lett 28(13):1727–1734

Xu X, Wang X (2005) An adaptive network intrusion detection method based on PCA and support vector machines. In: Advanced data mining and applications, pp 696–703

Yang Y (1995) 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, pp 256–263

Ziegler CN, McNee SM, Konstan JA, Lausen G (2005) Improving recommendation lists through topic diversification. In: Proceedings of the 14th international conference on World Wide Web, pp 22–32

Affiliations

Jessica Clark[1] **· Foster Provost**[2]

✉ Jessica Clark
jclark1@rhsmith.umd.edu

Foster Provost
fprovost@stern.nyu.edu

[1] University of Maryland Robert H. Smith School of Business, 7621 Mowatt Lane, College Park, MD 20742, USA

[2] NYU Stern School of Business, 44 West Fourth Street, New York, NY 10012, USA