Fennell et al. *EPJ Data Science* (2019) 8:23
https://doi.org/10.1140/epjds/s13688-019-0201-0

EPJ Data Science
a SpringerOpen Journal

REGULAR ARTICLE Open Access

Predicting and explaining behavioral data with structured feature space decomposition

Peter G. Fennell[1], Zhiya Zuo[1,2] and Kristina Lerman[1*]

*Correspondence: lerman@isi.edu
[1]USC Information Sciences Institute, Marina del Rey, USA
Full list of author information is available at the end of the article

Abstract

Modeling human behavioral data is challenging due to its scale, sparseness (few observations per individual), heterogeneity (differently behaving individuals), and class imbalance (few observations of the outcome of interest). An additional challenge is learning an interpretable model that not only accurately predicts outcomes, but also identifies important factors associated with a given behavior. To address these challenges, we describe a statistical approach to modeling behavioral data called the structured sum-of-squares decomposition (S3D). The algorithm, which is inspired by decision trees, selects important features that collectively explain the variation of the outcome, quantifies correlations between the features, and bins the subspace of important features into smaller, more homogeneous blocks that correspond to similarly-behaving subgroups within the population. This partitioned subspace allows us to predict and analyze the behavior of the outcome variable both statistically and visually, giving a medium to examine the effect of various features and to create explainable predictions. We apply S3D to learn models of online activity from large-scale data collected from diverse sites, such as Stack Exchange, Khan Academy, Twitter, Duolingo, and Digg. We show that S3D creates parsimonious models that can predict outcomes in the held-out data at levels comparable to state-of-the-art approaches, but in addition, produces interpretable models that provide insights into behaviors. This is important for informing strategies aimed at changing behavior, designing social systems, but also for explaining predictions, a critical step towards minimizing algorithmic bias.

Keywords: Computational social science; Empirical studies; Online social networks; Human behavior; Feature selection

1 Introduction

Explanation and prediction are complementary goals of computational social science [1]. The former focuses on identifying factors that explain human behavior, for example, by using regression to estimate parameters of theoretically-motivated models from data. Insights gleaned from such interpretable models have been used to inform the design of social platforms [2] and intervention strategies that steer human behavior in a desired direction [3]. In recent years, prediction has become the de-facto standard for evaluating learned models of social data [4]. This trend, partly driven by the dramatic growth of behavioral data and the success of machine learning algorithms, such as decision trees and

support vector machines, emphasizes ability to accurately predict unseen cases (out-of-sample or held out data) over learning interpretable models [5, 6].

Explainability, however, has become an important aspect of computational modeling. Increasingly, applications of machine learning in commercial, educational, and even judicial settings (e.g., [7]) are subject to regulation and to scrutiny for adverse effects such as biases and discrimination. For example, the US Equal Credit Opportunity Act requires creditors to provide an explanation to an applicant when their models recommend rejecting the applicant for credit, and such explanations require model interpretability. Meanwhile, controversy around discrimination in algorithmic decisions (e.g., see [8]) have further highlighted the need for transparent models whose recommendations can be understood, in contrast to black box algorithms where the step from input data to output decision is opaque. As we come to rely on algorithms to make decisions big and small, the need for algorithmic transparency and the ability to explain machine predictions become ever more acute.

Interpreting algorithms and making sense of behavioral data, however, has proven challenging. Behavioral data is usually massive, containing records of many individuals, each with a large number of potentially highly correlated features. However, the data is also sparse (with only a few observations available per individual) and unbalanced (few examples of the behavior within each class). Yet another challenge is heterogeneity: data is composed of subgroups that vary widely in their behavior. For example, the vast bulk of social media users have very few followers and post a few messages, but a few users have millions of followers or are extremely prolific posters. Ignoring heterogeneity can lead analysts to wrong conclusions due to statistical paradoxes [9, 10].

Machine learning, data science, and social science communities have proposed a number of approaches to learning explainable models from data. Popular among these are regression methods and decision trees, and their ensemble variants, such as random forests and boosting methods. However, while these approaches address one set of challenges, they often trip over the remaining ones. Regression models (e.g., Ridge, Lasso, Elastic Net), while offering interpretability, are limited by their specified functional form and fail to capture relationships in data that do not adhere to this form, and thus can be ineffective at adequately representing the data. Tree-based methods, on the other hand, are very effective at capturing non-linear and unbalanced data, but have limited interpretability. While they offer a measure of feature importance, the relationship between the outcome and features is less transparent, as it requires navigating the depths of many trees, potentially with the same features appearing at different levels.

Motivated by the need for algorithms that perform strongly at both joint goals of prediction and explanation, we propose *Structured Sum-of-Squares Decomposition* (S3D) algorithm, a mathematically principled method for learning interpretable statistical models of behavioral data. The algorithm, which is a variant of decision trees [11], builds a single tree-like structure that is both highly *interpretable* and can be used for out-of-sample *prediction*. In addition, the learned models can be used to *visualize* data.

S3D works as follows: given a set of features and a binary or a continuous-valued outcome Y, S3D identifies a subset of m important features that are orthogonal in their relationship with the outcome and collectively explain the largest amount of the variation in the outcome. In addition to these *selected* features, S3D identifies correlations between all features, thus providing important insights into interactions between features that were

not selected by the model. Similar to decision trees, the S3D algorithm recursively bins the m-dimensional space defined by the selected features into smaller, more homogeneous subgroups or bins, where the outcome variable exhibits little variation within each bin but significant variation between bins. However, in contrast to decision trees, it does so in a structured way, by minimizing variation in the outcome *conditioned* on the existing partition. The decomposition effectively creates an approximation of the (potentially non-linear) functional relationship between Y and the features, while the structured nature of the decomposition gives the model interpretability and also helps reduce overfitting. The resulting model is parsimonious. Indeed, S3D is a low complexity model with only two hyperparameters, but despite its low complexity we show that it is a highly performant predictive tool.

To demonstrate the utility of the proposed method, we apply it to model a variety of datasets, from benchmarks to large-scale heterogeneous behavioral data that we collect from social platforms, including Twitter, Digg, Khan Academy, Duolingo, and Stack Exchange. Across datasets, the performance of S3D is competitive to existing state-of-the-art methods on both classification and regression tasks, while it also offers several advantages. We highlight these advantages by showing how S3D reveals the important factors in explaining and predicting behaviors on the question answering site Stack Exchange and social network Digg. Qualitatively, S3D allows for visualizing the relationship between the outcome and features, and quantifies their importance via prediction task. Surprisingly, despite high heterogeneity of these relationships in many datasets, just a few important features identified by S3D can predict held-out data with remarkable accuracy.

The paper is laid out as follows. In Sect. 2 we examine related work. In Sect. 3 we introduce the S3D model and describe the algorithm for training the model from data. Section 4 is the results section where we apply S3D to various datasets, comparing its *accuracy* to state-of-the-art models while also using its *interpretability* to examine and understand data. We conclude in Sect. 5. We expect that S3D can be a high utility tool to interpretable modelling in both the scientific and commercial domains, and so have made our code open source to the community at large [12].

2 Related work

An empirical study of socio-behavioral data typically begins with a researcher selecting a small set of explanatory variables—perhaps those identified by Principal Component Analysis (PCA), factor analysis, or a more complex method—and a functional form of the relationship between these and the outcome variable, and performs regression analysis to find the coefficients of the explanatory variables. S3D is a novel approach to predictive analytics that combines the necessities of data modeling in a single method: dimensionality reduction, predictive power and iterpretability.

Like PCA, S3D reduces the dimensionality of the data. However, while PCA is an unsupervised technique that creates a smaller set of components that explain the largest amount of variation between the explanatory variables; S3D is a supervised method. It recursively selects features that explain the largest amount of variation between the explanatory variables conditioned on previously selected features (i.e., features at higher levels of the tree), and thus creates a subset of features that are orthogonal to each other with respect to the outcome variable. This smaller set of features accounts for the explainable

variation of the outcome variable. Indeed different outcomes variables can result in different subsets of features, in contrast to unsupervised methods. Another factor that differentiates PCA from S3D is that PCA-learned components are in the derived eigenspace of the features, and thus, they are not directly interpretable within domain knowledge. In contrast, S3D selects important features directly in the same space, which aids interpretation.

S3D also builds a predictive model of data. Social science, machine learning and data science communities have developed a variety of solutions for this task. However, while these solutions address one set of challenges, they often trip over the remaining ones. Linear models, such as linear and logistic regression, have been the mainstay of the computational social sciences community for decades, due to their ability to provide interpretable models of data. However, linear models have a number of drawbacks that limit their utility. They are constrained by their functional form and cannot capture non-linear relationships and complex interactions between features. Furthermore, the interpretability of the coefficients of a linear model can be very limited in data with correlated features. Normally, a researcher examines the effect of a feature on the outcome variable through inspection of the linear coefficient of the feature in the model, but such analysis assumes independence of the features (so that other features can remain constant given a change in the feature). Thus, such analysis with correlated features is not good practice and can lead to misleading conclusions. A standard avenue for improving a linear model is through penalized regression, such as Lasso, Ridge and Elastic Net [13] to reduce the dimensionality of the feature space and to prevent overfitting [14]. These methods select important features for explaining the data, but are not guaranteed to pick features that are uncorrelated. Furthermore, such approaches can also set the coefficients of the remaining features to zero, which makes it impossible to learn how those features affect behavior. In contrast, S3D explicitly learns relationships between correlated features from data and can express these relationships through a network, giving insights into the interdependency between selected and remaining features. Unlike linear models, it can also learn non-linear relationships in data.

Machine learning methods are often significantly more accurate than simpler linear models. While deep learning has attracted attention recently due to its performance on highly complex tasks in natural language processing, computer vision and robotics, tree-based algorithms typically perform as well, if not better, on traditional tasks involving tabular data. These algorithms are based on decision trees, such as CART [11], BART [15] or MARS [16], which work by partitioning data into non overlapping bins to minimize the variance of response variable, or some other cost function, within these subsets. Decision trees are typically prone to overfitting (e.g., early stopping and pruning) and so are ensembles to create optimal models. Ensemble approaches include boosting models such XGBoost or LightGBM, and bagging models the most well known of which is random forest models [17]. Random forest models, for example, learn a model as an average of individual decision trees trained on subsets of the data, and averaging in this way reduces overfitting and optimizes performance on held-out test sets.

Decision trees are very effective at learning from non-linear and unbalanced data, and ensemble tree models are popular due to their strong predictive performance. However, the interpretability of these methods is mainly limited to feature importance, which quantifies the importance of each feature by its contribution to minimizing the loss function.

While a decision hierarchy may be considered interpretable [18], many factors can complicate interpretability, such as the depth of the tree and the fact that the same feature can appear at multiple levels. As a result, one does not look too deeply in the tree—generally, at the features at the first level only. In contrast, S3D does not reuse features at different levels of the hierarchy, which lends itself naturally to visualization. As we show in this paper, the aerial views of the important feature space serve to provide not just interpretability, but insight into data.

In summary, S3D combines the performance of high accuracy tree-based methods with the interpretability of regression. It carries out dimensionality reduction and feature selection as part of the model fitting stage, and can fit a variety of implicit functional relationships between the variables, not just the linear relationship. S3D's predictions of binary outcomes are not as sensitive to parameter choices as BART's [19]. It has much lower complexity than tree-based methods which typically employ tens or hundreds of trees; indeed S3D also has the advantage of having only two parameters to tune, as opposed to twelve for a typical random forest. S3D adds structure to the feature importance offered by random forests, showing which subset of features are sufficient for explaining variation in outcomes. Finally, S3D is fully transparent and allows for visualizing the data and predictions. S3D addresses interpretability by identifying features that are important to explaining the outcome variable, along with the relationship between these features and the outcome. This feature identification can help inform mathematical models of human behavior. Such model-based approaches have been used, for example, to predict popularity of online content [20], the productivity of scientists [21], and the size of epidemics [22].

3 Method

We present the Structured Sum-of-Squares Decomposition algorithm (S3D), a variant of the classification and regression trees (CART) [11, 15, 16], that takes as input a set of features $\mathbf{X} = \{X_j\}_{j=1}^{M}$ and an outcome variable Y and selects a smaller set of m important features that collectively best explain the outcome. The method bins the values of each important feature X_j^S to decompose the m-dimensional selected feature space into smaller non-overlapping blocks, such that Y exhibits significant variation between blocks but little variation within each block. These blocks allow us to approximate the functional relationship $Y = f(\mathbf{X})$ as a multidimensional step function over all blocks in the m-dimensional selected feature space, thus capturing non-linear relationships between features and the outcome.

Our method chooses features recursively in a greedy manner, so that features chosen at each step explain most of the variation in Y *conditioned* on the features chosen at the previous steps. Features that are correlated will explain much of the same variation in Y, and our approach of successively choosing features based on how much remaining variation in Y they explain results in a set $\mathbf{X}^S = \{X_j^S\}_{j=1}^{m}$ of important features that are orthogonal in their relationships with Y. By decomposing data recursively, we create a parsimonious model that not only quantifies relationships between the features and the outcome variable Y, but also quantifies relationships among the features themselves. Figure 1 illustrates the algorithm. The left panel shows the data, projected onto the space defined by two features X_1 and X_2. As the first step, S3D bins the values of each feature and selects one that explains the largest variation of the outcome Y, here X_1 (middle panel). Next, it bins the values of each unselected feature, to identify one that explains most of the remaining variation, here feature X_2. The resulting two-dimensional partition of the feature space also

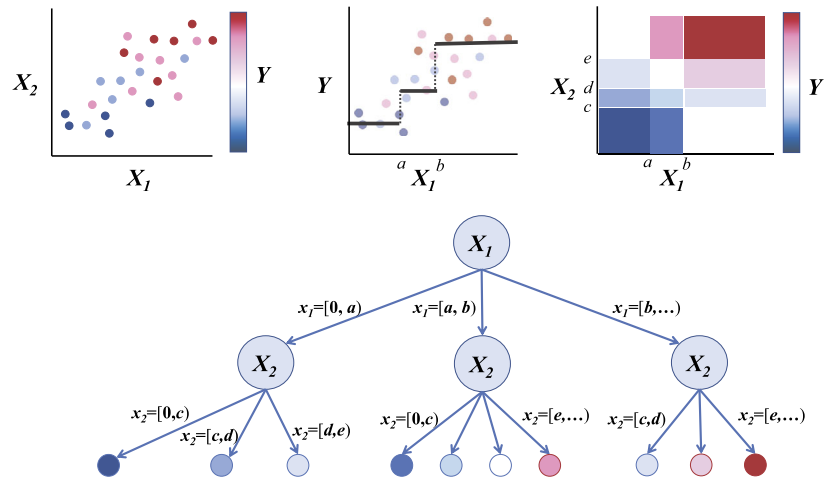

Figure 1 Illustration of S3D learning a model of data. S3D bins the values of each feature (left panel) to select one that explains the largest variation of the outcome Y (middle panel). Next, it bins each unselected feature to identify one that explains the largest amount of the remaining variation. The aerial view of the partition of the important feature space (right panel) represents a visualization of the data, which is also represented by a tree (bottom panel). S3D continues partition the space using features not previously selected, until there is no appreciable variation left to explain

serves as visualization of the data (right panel). The decomposition of the data according to the selected features can also be represented by a tree (bottom). The algorithm continues in this manner until there is no remaining variation left to explain.

Our model is able to achieve performance comparable to state-of-the-art machine learning algorithms on prediction tasks, while offering advantages over those methods: our algorithm uses only two tuning parameters, can represent non-linear relationships between variables, and creates an interpretable model that is amenable to analysis and produces insights into behavior that merely predictive models do not give. Below, we described the key steps of the algorithm.

3.1 Structured feature space decomposition

A key concept used to describe variation in observations $\{y_i\}_{i=1}^N$ of a random variable Y is the total sum of squares SST, which is defined as $SST = \sum_{i=1}^N (y_i - \bar{y})^2$, where $\bar{y} = \sum_{i=1}^N y_i/N$ is the sample mean of the observations. The total sum of squares is intrinsically related to variation in Y; indeed the sample variance $\hat{\sigma}^2$ of Y can be directly obtained from this quantity as $\hat{\sigma}^2 = SST/(N-1)$.

Given a feature X_j, one method of quantifying its importance, i.e., how much variation in Y can be explained by X_j is as follows: (1) partition X_j into a collection P_{X_j} of non-overlapping bins (Fig. 1), (2) compute the number of data points N_p and the average value $\bar{y}_p$ of Y in each bin $p \in P_{X_j}$, and (3) decompose the total sum of squares of Y as

$$\sum_{i=1}^{N}(y_i - \bar{y})^2 = \sum_{p \in P_{X_j}} N_p(\bar{y}_p - \bar{y})^2 + \sum_{p \in P_{X_j}} \sum_{i=1}^{N_p}(y_{p,i} - \bar{y}_p)^2, \tag{1}$$

where $y_{p,i}$ here is the i'th data point in bin p. The first sum on the right hand side of Eq. (1) is the explained sum of squares (or sum of squares between groups), a weighted average

of squared differences between global ($\bar{y}$) and local ($\bar{y}_p$) averages that measures how much Y varies between different bins p of X_j. The second sum is the residual sum of squares (or sum of squares within groups), which measures how much variation in Y remains within each bin p. The R^2 coefficient of determination is then the proportion of the explained sum of squares to the total sum of squares, given by

$$R^2 = \frac{\sum_{p \in P_{X_j}} N_p (\bar{y}_p - \bar{y})^2}{SST}. \tag{2}$$

The R^2 measure takes values between zero and one, with large values of R^2 indicating a larger proportion of the variation of Y explained by X_j. This method of approximating the functional relationship between Y and X_j as a step function with bins, or groups P_{X_j} and corresponding values $\bar{y}_p$, allows us to quantify the variation in Y explained by X_j through the R^2 of the corresponding step function as given by Eq. (2).

3.1.1 Binning values of a feature

We now introduce a method to systematically *learn* the binning P_{X_j} of the feature X_j which will be central to our algorithm. Given the data, we can split the domain of the feature X_j at the value s into two bins: $X_j \leq s$ and $X_j > s$. From Eq. (2), we see that the proportion of variation in Y explained by such a split is

$$R^2(s; X_j) = \frac{N_{X_j \leq s}(\bar{y}_{X_j \leq s} - \bar{y})^2 + N_{X_j > s}(\bar{y}_{X_j > s} - \bar{y})^2}{SST}, \tag{3}$$

where $N_{X_j \leq s}$ and $\bar{y}_{X_j \leq s}$ are the number of data points and average value of Y in the bin $X_j \leq s$, and vice versa for $N_{X_j > s}$ and $\bar{y}_{X_j > s}$. $R^2(s; X_j)$ can be computed for each possible value of s in the domain of X_j, and we can choose the optimal split s_1 as the split s that maximizes $R^2(s; X_j)$ of Eq. (3). Choosing s_1, and binning the domain of X_j into $P_{X_j} = \{[\min(X_j), s_1], (s_1, \max(X_j))\}$, we can again find the next best split s_2 to optimize the improvement in R^2. In general, having chosen n splits $\{s_u\}_{u=1}^n$ with a resulting partition P_{X_j} of $n+1$ bins, the next best split s_{n+1} can be chosen as the split s that maximizes the improvement in R^2 as given by

$$\begin{aligned} \Delta R^2(s | P_{X_j}; X_j) = \frac{1}{SST} \big(& N_{p(s)|X_j \leq s} (\bar{y}_{p(s)|X_j \leq s})^2 \\ & + N_{p(s)|X_j > s} (\bar{y}_{p(s)|X_j > s})^2 - N_{p(s)} (\bar{y}_{p(s)})^2 \big), \end{aligned} \tag{4}$$

where $p(s)$ in Eq. (4) is the bin in P_{X_j} that contains the point s and $p(s)|X_j \leq s$ (resp. $p(s)|X_j > s$) is the restriction of that bin to points $X_j \leq s$ (resp. $X_j > s$). In this manner, we recursively split the domain of X_j to create a partition of the feature.

However, splitting in this manner can continue indefinitely, resulting in a model that is too fine-grained and thus overfits the data. To prevent overfitting, we need a stopping criterion. To this end we introduce a loss function $L(P_{X_j})$ that penalizes the size $|P_{X_j}|$ of the partition, i.e., the number of bins:

$$L(P_{X_j}) = 1 - R^2(P_{X_j}) + \lambda |P_{X_j}|. \tag{5}$$

The parameter λ controls how fine-grained the bins are: smaller values of λ allow for more finer bins, and vice versa. The loss function of Eq. (5) reaches a minimum when the change in $R^2(P_{X_j})$ from adding an extra split to P_{X_j} is less than λ—at this point we stop splitting and return the binning P_{X_j}.

Having formed the binning P_{X_j} of X_j with splits $\{s_u\}_{u=1}^n$, the total score $R^2(X_j)$ can be calculated from Eq. (2), or by summing $R^2(s_1; X_j)$ from Eq. (3) along with the ΔR^2 terms in Eq. (4) for each of the splits $\{s_u\}_{u=2}^n$. Completing this procedure for all features gives a measure of *feature importance*, i.e., how much variation in Y each feature alone explains, and ranking these features allows us to choose the most important feature X_{C_1} that explains the largest amount of the variation in Y.

3.1.2 Selecting additional features

After choosing the most important feature, we search the rest of the features for one that explains most of the remaining variance in Y, then the third feature, and so on. Here, we describe the procedure for finding the next best feature having already chosen l features $\mathbf{X}^S = \{X_1^S, \ldots, X_l^S\}$ with a corresponding binning $\mathcal{P}^S = P_1^S \times \cdots \times P_l^S$ of the chosen feature space, where $\times$ here is the cartesian product. In this case, a total $R^2(\mathcal{P}^S) = \sum_{p\in\mathcal{P}^S} N_p(\bar{y}_p - \bar{y})^2/SST$ of the variation in Y has been explained, and we now look for the feature that best explains the remaining variation $1 - R^2(\mathcal{P}^S)$.

Given a remaining feature X_j, we bin the domain of X_j similarly to how we binned it when choosing the first feature. The first split s_1 of X_j is chosen as the value s that maximizes the improvement in R^2, given by

$$\Delta R^2\big(s|\mathcal{P}^S; X_j\big) = \frac{1}{SST}\sum_{p\in\mathcal{P}^S}\big(N_{p|X_j\le s}(\bar{y}_{p|X_j\le s})^2 + N_{p|X_j>s}(\bar{y}_{X_j>s})^2 - N_p(\bar{y}_p)^2\big), \tag{6}$$

where $p|X_j \le s$ (resp. $p|X_j > s$) is the set of data points in $p \in \mathcal{P}^S$ for which $X_j \le s$ (resp. $X_j > s$). In general, given n splits and a corresponding partition P_{X_j} of X_j, the $n+1$'st split is chosen as the value s that maximizes

$$\Delta R^2\big(s|P_{X_j}, \mathcal{P}^S; X_j\big) = \frac{1}{SST}\sum_{p\in\mathcal{P}^S\times P_{X_j}|s\in p}\big(N_{p|X_j\le s}(\bar{y}_{p|X_j\le s})^2 + N_{p|X_j>s}(\bar{y}_{X_j>s})^2 - N_p(\bar{y}_p)^2\big), \tag{7}$$

with the sum in Eq. (7) taken over all elements p of $\mathcal{P}^S \times P_{X_j}$ that contain the point s. The loss function in the general setting is

$$L\big(P_{X_j}|\mathcal{P}^S\big) = \frac{1 - R^2(\mathcal{P}^S \times P_{X_j})}{1 - R^2(\mathcal{P}^S)} + \lambda|P_{X_j}|, \tag{8}$$

where the denominator of the fractional first term in Eq. (8) normalizes this term to be between zero and one, is the case in Eq. (5). This normalization is necessary because as we progress through the algorithm, subsequent features may explain less of the variance of Y (as features are chosen hierarchically), and so changes in $1 - R^2(\mathcal{P}^S \times P_{X_j})$ by splitting X_j are smaller. The effect of this would be a coarser binning of the feature, and so the

normalization ensures that this is not the case and that the feature is binned consistently at each stage of the algorithm. Again, the partition P_{X_j} of X_j is chosen that minimizes the loss function of Eq. (8), and the R^2 improvement is calculated for this feature as

$$\Delta R^2\left(P_{X_j}|\mathcal{P}^S\right) = R^2\left(\mathcal{P}^S \times P_{X_j}\right) - R^2\left(\mathcal{P}^S\right). \tag{9}$$

This procedure is repeated for all remaining features X_j to select the feature X_{l+1} with the maximal R^2 improvement.

This process of binning features, calculating their importance by the improvement ΔR^2 and choosing the one with the largest improvement continues until no further variation in Y can be explained or until an alternative stopping condition (such as a pre-specified maximum number of steps) is met. Our algorithm learns a hierarchy of important features that explain the variation in Y and a binning or partition $\mathcal{P}^S$ with corresponding values $\{\bar{y}_p\}_{p\in\mathcal{P}^S}$ approximating the functional relationship between the outcome and the features. Note that, when binning a feature, once the vectors $\{\{N_{p|X_j\leq s}\}_{p\in\mathcal{P}^S}\}_s$ and $\{\{\bar{y}_{p|X_j\leq s}\}_{p\in\mathcal{P}^S}\}_s$ have been constructed (an operation that takes $\mathcal{O}(N)$ time), the binning is independent of N, and instead depends on the number of unique values of the feature (as required to calculate the optimal splits s at each step). The fact that S3D scales linearly with the number of data points N allows us to apply the algorithm to large datasets, such as the Twitter and Digg (see Sect. 4).

3.1.3 Hyperparameters

The S3D model has two hyperparameters: (*1*) λ that controls granularity of feature binning; (*2*) k that specifies the maximum number of features to use for prediction. The hyperparameter k is analogous to the maximum tree depth in decision tree methods. Both hyperparameters are important to prevent overfitting — left unrestricted, the algorithm can learn too fine-grained a model that fails to generalize to unseen data. We note that it is possible to stop early in the training phase by restricting the maximum number of features to select. In other words, the algorithm can stop when the number of selected features reaches a predefined value. Nonetheless, it is recommended *not* to lay any limit during the training phase but rather tune k in the validation step.

To tune the hyperparameters, we train S3D for various values of λ, in each case letting the algorithm continue until there is no further improvement in R^2. This results in a model with m_λ selected features and partition $\mathcal{P}^{S_\lambda} = \{P_1^{S_\lambda}, \ldots, P_{m_\lambda}^{S_\lambda}\}$. Then, for $k \in [1, m_\lambda]$, we evaluate the predictive performance of the model using only the top k selected features and the sub-partition $\{P_1^{S_\lambda}, \ldots, P_k^{S_\lambda}\}$ (Fig. 2). Performance is measured on held-out tuning data using a specified metric. The optimal hyperparameters (λ, k) are those that achieve the best performance on held-out data.

3.2 Applications of the learned model

Given a dataset, S3D learns an ordered set of important, orthogonal features $\mathbf{X}^S$, a partitioning $\mathcal{P}^S$ of the selected feature space with corresponding $\bar{y}_p$ and N_p values for each bin or block $p \in \mathcal{P}^S$, and ΔR^2 for each remaining variable at each step of the algorithm. This decomposition serves as a parsimonious model of data and can be used for feature selection, feature correlation, prediction and analysis as described below.

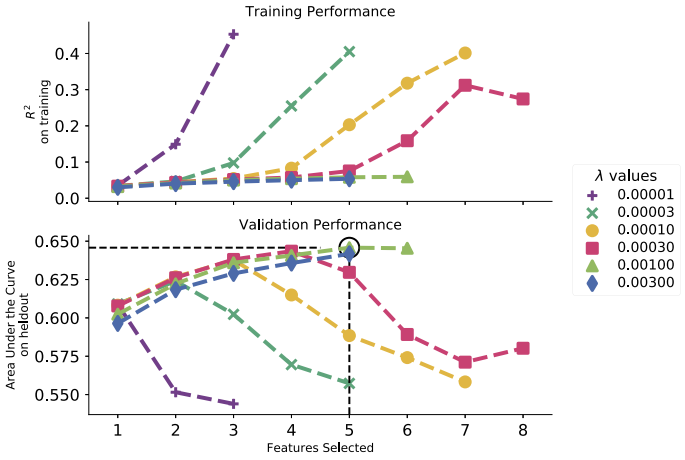

Figure 2 Hyperparameter tuning on *Stack Exchange* data. Top: R^2 in the training set as a function of λ and k; Bottom: AUC on the held-out data. Note that for illustration, we show the trajectory for only one of five splits. Different "best" hyperparameters may exist for different splits

3.2.1 *Feature selection and correlations*

The ordered set $\mathbf{X}^S$ of important, orthogonal features allows us to quantify feature importance in heterogeneous behavioral data. The top-ranked features explain the largest amount of variation in the outcome variable, while each successive feature explains most of the remaining variation that is not explained by the features that were already selected.

Aside from the selected features $\mathbf{X}^S$, S3D provides insights into features that are *not* selected by the algorithm, quantifying variation that they explain in the outcome variable that is made redundant through the selected variables. This is calculated in the following manner. At a given step l of the algorithm, feature X_l^S is selected as the best feature with an R^2 improvement of $\Delta R^2(P_{X_l^S}|\mathcal{P}^{S(l-1)})$ (Eq. (9)), where $\mathcal{P}^{S(l-1)}$ is the partition prior to step l. Meanwhile, a different remaining feature X_j has an R^2 improvement of $\Delta R^2(P_{X_j}|\mathcal{P}^{S(l-1)})$. At the next stage of the algorithm, given X_l^S has been selected, X_j will have an R^2 improvement of $\Delta R^2(P_{X_j}|\mathcal{P}^{S(l-1)} \times P_{X_l^S})$, and thus the variation in X_j that is made redundant through the selection of X_l^S is the difference between these two ΔR^2:

$$a_{X_j,X_l^S} = \Delta R^2\left(P_{X_j}|\mathcal{P}^{S(l-1)}\right) - \Delta R^2\left(P_{X_j}|\mathcal{P}^{S(l-1)} \times P_{X_l^S}\right). \tag{10}$$

The coefficients a_{X_j,X_l^S} facilitate our analysis of (a) relationships between the features and (b) the effect of unselected features on the outcome variable (through selected features to which they are correlated). We implement the coefficients as weights in a feature network that is weighted and directed (Fig. 8). This network gives a tool for analysis—unselected features that otherwise can explain much of the variation in the outcome will have heavy links, and the selected features X_l^S to which these links point reveal correlations and through which selected features the unselected feature is made redundant.

3.2.2 *Prediction*

The learned model can be used as a predictive tool for both discrete and continuous-valued outcome variables. Given input data $\mathbf{x}$, the model predicts the expected value $\hat{\mu}$ of Y as $\hat{\mu}|\mathbf{x} = \bar{y}_{p(\mathbf{x})}$, where $p(\mathbf{x})$ is the block in decomposition $\mathcal{P}^S$ to which $\mathbf{x}$ belongs. For

continuous-valued outcome variables, this predicted expected value $\hat{\mu}$ will be the prediction of the outcome of Y, i.e., $\hat{y}|\mathbf{x} = \bar{y}_{p(\mathbf{x})}$. For discrete-valued outcomes, the expected value has to be thresholded to predict an outcome class. For binary outcomes $Y \in \{0, 1\}$, $\hat{\mu}|\mathbf{x} = \bar{y}_{p(\mathbf{x})}$ is the maximum likelihood estimate of the probability of the outcome $Y = 1$ in the block $p(\mathbf{x})$, and thus our model specifies that the outcome $Y = 1|\mathbf{x}$ will occur with probability $\bar{y}_{p(\mathbf{x})}$. By choosing an appropriate *discrimination threshold* θ, our model then makes the prediction $\hat{y}$ as

$$\hat{y}|\mathbf{x} = \begin{cases} 1, & \bar{y}_{p(\mathbf{x})} \geq \theta, \\ 0, & \bar{y}_{p(\mathbf{x})} < \theta. \end{cases} \tag{11}$$

Unbalanced data Two of the datasets that we study, Digg and Twitter, are highly unbalanced—their outcome variables (whether the user *adopts* a meme) are binary, and the proportions of positive outcomes in the data are 0.0025 and 0.0007 respectively. Using the standard discrimination threshold $\theta = 0.5$ results in predicting an insufficient number of ones. To address this issue, we choose the discrimination threshold based on the training data, picking the largest value θ such that the number of predicted positive examples in the training data is greater than or equal to the actual number of positive examples in the training set. This threshold is then used for prediction on the held-out tuning data, as well as on the test data. Note that, for these two datasets, we also alter the discrimination threshold for the regression and random forest models in the same manner.

3.2.3 Analysis and model interpretation

One of the more interesting contributions of S3D is its potential for model exploration. By selecting features sequentially, we create a model where typically lower amounts of variation are explained at successive levels, and so a visual analysis of the first few important dimensions of the model allows us to understand the effects of the important features on data and predictions. The expectations $\bar{y}_p$ for each block $p \in \mathcal{P}^S$ facilitate this exploration, approximating the relationship $Y = f(\mathbf{X})$ between outcome and features. Furthermore, for binary data, the predicted outcomes obtained by thresholding the expectations show how predictions change as a function of the features, which allows for explaining predictions and visually exploring the data.

3.3 Comparison to state-of-the-art

We compare S3D to linear and logistic regression (with *Lasso* and *Elastic Net* [13] regularization), random forests (*RF*) [17], and support vector machines with linear kernel (*Linear SVM*) [23], using 5-fold cross validation (CV; Section S1.2). The Scikit Learn [24] implementation of the random forest model used in our experiments is based on the CART algorithm. We also compare S3D's performance to models with similar complexity. First, we use random forest to rank all the features and retrain the model on the top-k (and top-$2k$) ranked features, where k is the number of important features selected by S3D. The resulting models are called $RF(k)$ and $RF(2k)$ respectively. Second, we investigate the efficacy of using S3D for *supervised feature selection*. Specifically, we train random forests using only the k important features chosen by S3D, which we refer to as *RF-S3D* model.

These algorithms are ideal benchmarks for S3D, with the ability to provide feature importance scores, and therefore interpretability, to the trained models. Further, we emphasize that S3D can produce sparse models that can do both feature selection and prediction.

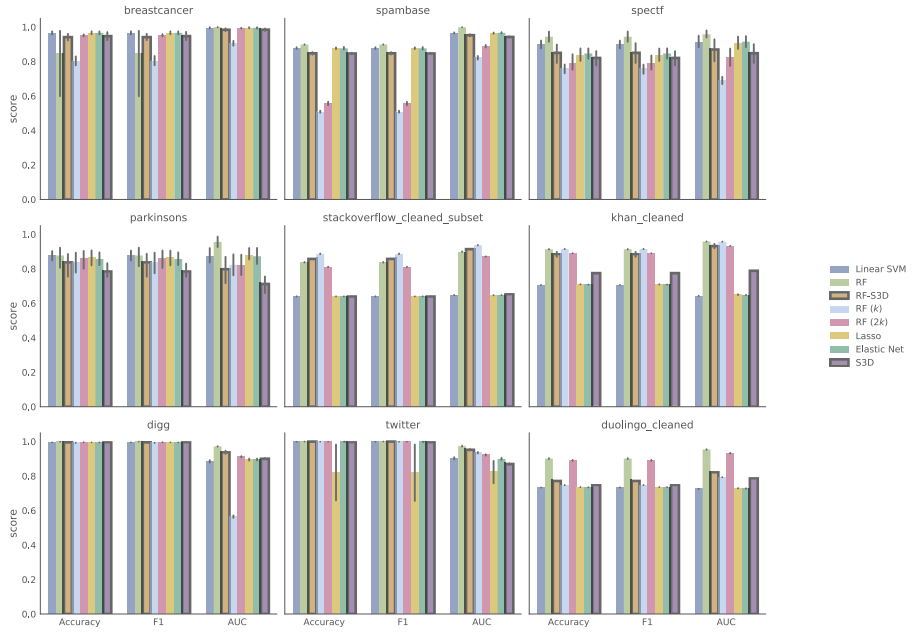

Figure 3 Classification performance on 5-fold cross validation in nine datasets. Error bars here indicate one standard deviation. A higher bar (greater value) means better performance

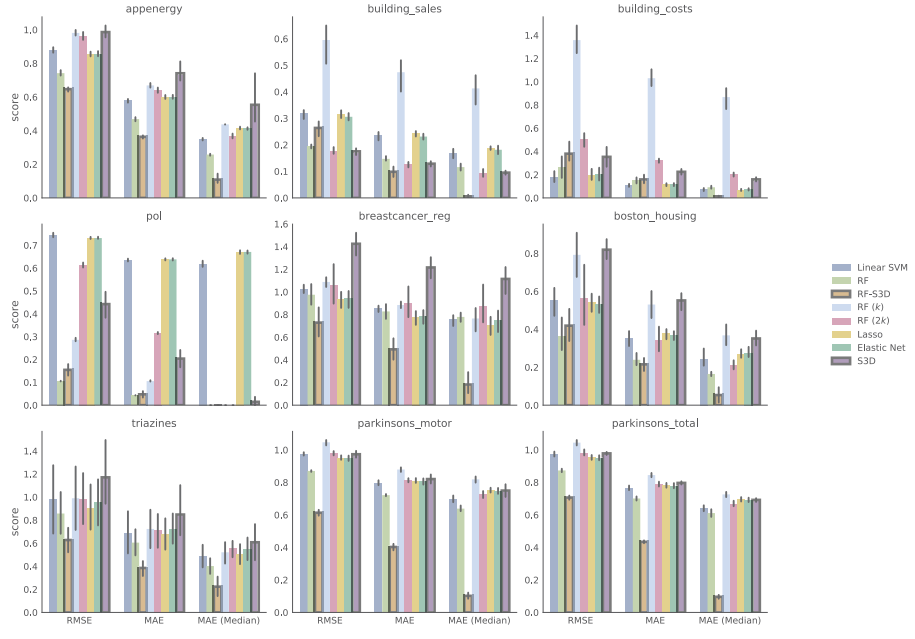

Figure 4 Regression performance on 5-fold cross validation in nine datasets. Error bars here indicate one standard deviation. A higher bar (greater value) means *worse* performance

As shown in Sect. 4.2, S3D performs similarly to logistic regression in both classification and regression (Figs. 3 and 4), but uses fewer features (Fig. 5). Finally, both random forests [25] and linear SVM [26] are proven to be adequately powerful in prediction tasks while being relatively simple to train.

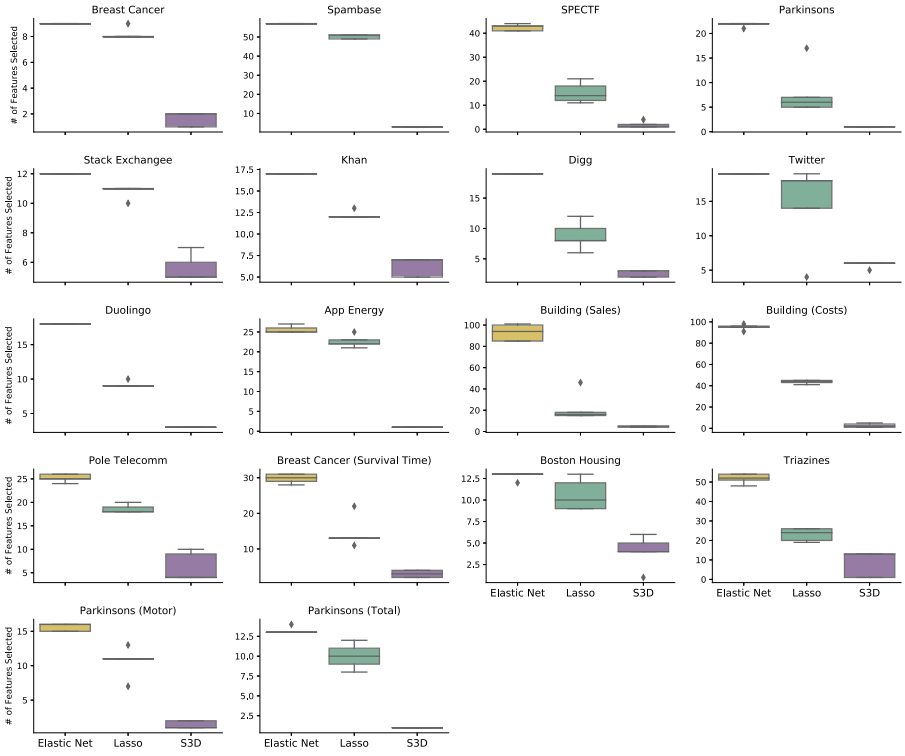

Figure 5 Number of features selected by elastic net and lasso regression, as well as S3D, across outer CV for all datasets. The five points in each box correspond to the number of features selected in each *outer* fold

We partition data into five equal size folds,[a] each of which is rotated as a hold-out set for testing. For classification (except random forests), we standardize feature values by centering and scaling variance to one. For regression, both features and target values are standardized. Standardization of test set is based on training data. In each run, we train and tune the models on four folds, where three are used for training and one for validation. Finally we evaluate the performance of the tuned models on the remaining test fold. The final evaluation is therefore the average performance across each of the five folds. In other words, we tune the hyperparameters with 4-fold CV (hereafter *inner CV*) and evaluate the performance of the optimal models with 5-fold CV (hereafter *outer CV*). For classification tasks, we evaluate performance using (1) accuracy (the percentage of correctly classified data points), (2) $F1$ score (the harmonic mean of precision, the percentage of predicted ones that are correctly classified, and recall, the percentage of actual ones that are correctly classified), and (3) area under the curve (AUC). For regression tasks, we employ (1) root mean squared error (RMSE), (2) mean absolute error (MAE-Mean), (3) median absolute error (MAE-Median). Note that for classification tasks, higher values of the metrics imply better performance. For regression tasks, lower values of the metrics imply better performance. During the inner CV phase (i.e., hyperparameter tuning), we optimize classification performance on AUC scores and regression on RMSE scores.

For Lasso regression we tune the strength of $l1$ regularization. For elastic net regression, we tune the strengths of both $l1$ and $l2$ regularizations. For random forests, we tune (1) the number of features to randomly select for each decision tree, (2) the minimum number of samples required to make a prediction, (3) whether or not to bootstrap when sampling data

for each decision tree, and (4) criterion of the quality of a split (choices are Gini impurity or information gain for classification; MAE or MSE for regression). For linear SVM, we tune the penalty parameter for regularization.

4 Results

We apply S3D[b] to benchmark datasets from the UCI Machine Learning Repository [27] and from Luís Torgo's personal website.[c] In addition, we include five large-scale behavioral datasets, as described in the following paragraphs. Table 1 lists all *18* datasets used in both classification and regression tasks, along with their statistics. See Additional file 1 Section S1.1 for more detailed description of the benchmark data and data preprocessing.

Behavioral data came from various social platforms:

- *Stack Exchange.* The Q&A platform Stack Exchange enables users to ask and answer questions. Askers can also *accept* one of the answers as the best answer. This enables us to measure answerer performance by whether their answer was accepted as the best answer or not. The data we analyze includes a random sample of all answers posted on Stack Exchange from 08/2009 until 09/2014 that preserves the class distribution. Each record corresponds to an answer and contains a binary outcome variable $Y \in \{0, 1\}$ (one indicates the answer was accepted, and zero otherwise), along with 14 features. These features include answer-based features, such as the length of the answer, measured in the number of *words*, *lines of code* and *hyperlinks* to Web content the answer contains, the *number of other answers* the question already has, the answer's *readability* score, a numeric index giving the level of education needed to easily comprehend the answer. Other features include the answerer's *reputation*, how long the answerer has been registered (*signup duration* in months) and a percentile rank (*signup percentile*), the number of answers they have previously written, time since the previous answer, the number of answers written by the answerer in his or her current session, and *answer's position* within the session, i.e., whether it was the first, second, third, etc. answer the user wrote during the same session.

Table 1 Datasets Used for Performance Comparison

Prediction task	Dataset	# of samples	# of features
Classification	Breast Cancer (Original) [28]	683	9
	Spambase	4601	57
	SPECTF [29]	267	44
	Parkinsons [30]	195	22
	Stack Exchange	1,026,225	12
	Khan	680,551	17
	Digg	1,000,000	19
	Twitter	5,000,000	19
	Duolingo	767,718	18
Regression	App Energy [31]	19,735	27
	Building (Sales) [32]	372	103
	Building (Costs) [32]	372	103
	Pole Telecommunication [33]	15,000	48
	Breast Cancer (Prognostic) [28]	194	32
	Boson Housing [34]	506	13
	Triazines [35]	186	60
	Parkinsons (Motor) [36]	5875	16
	Parkinsons (Total) [36]	5875	16

- *Khan Academy*. The online educational platform Khan Academy enables users learn a subject then practice what they learned through a sequence of questions on the subject. We study performance during the practice stage by looking at whether users answered the questions correctly on their first attempt ($Y = 1$) or not ($Y = 0$). We study an anonymized sample of questions answered by adult Khan Academy users over a period from 07/2012 to 02/2014. For each question a user answers we have 19 features: as with Stack Exchange, these include answer-based, user-based, and other temporal features. The features include the amount of time it takes the user to answer the question, (*solve_time*), the number of attempts the user made to answer the question, time since the user's previous answer (*time_since_prev_ans*), the number of questions the user answered during the current session, (*session_length*), and the answer's *position within the session*. Additional features include temporal attributes such as the *hour* of the day, *day* of the week, *month*, etc. that the question was answered; user-based features, such as the month user *signed up* for Khan Academy, the number of *first_five* questions user answered correctly without hints, time between user's first and last answer, (*signup_duration*), the numbers of *all questions* user ever attempted to answer, and the number of *all attempts* made on all questions, and other features, such as how long this user has currently been studying.
- *Duolingo*. The online language learning platform Duolingo is accessed through an app on a mobile device. Users are encouraged to use the app in short bursts during breaks and commutes. The data[d] was made available as part of a previous study [2]. The data contains a 2-week sample (02/28/2013 through 03/12/2013) of ongoing activity of users learning a language. All users in this data started lessons before the beginning of the data collection period. We focus on 45K users who completed at least five lessons. The median number of lessons was 7, although some had as many as 639 lessons. Performance on a lesson is defined as $Y = 1$ if the user got *all* the words in the lesson correct; otherwise, it is $Y = 0$. Features describing the user include how many lessons and sessions the user completed, how many perfect lessons the user had, the month and day of the lesson, etc.
- *Digg*. The social news platform Digg allows users to post news stories, which their followers can like or "digg." When a user diggs a story, that story is broadcast to his or her followers, a mechanism that allows for the diffusion of contents through the Digg social network. A further characteristic of Digg is its front page—stories that are popular are promoted to the front page and thus become visible to *every* Digg user. We study a dataset that tracks the diffusion of 3,500 popular Digg stories from their submissions by a single user to their eventual promotion to and residence on the Digg front page. We study information diffusion on Digg by examining whether or not ($Y \in \{0, 1\}$) users "digg" (i.e., adopt) a story following exposure of the story from their friends, and thus share that information with their followers. The features associated with adoption include user-based features, such as *indegree* and *outdegree* (number of followers and followees of the users), node *activity* (how often the user posts), *information received* (the rate at which the user receives information from all followees); dynamics-related features such as the number of times the user was exposed to the story, and story-related features, such as its *global popularity* in the previous hour, and diurnal-features, including the *hour* of the day and *day* of the

week. Through this data, we can study the factors that are important in explaining the spread of information in this social system.

- *Twitter.* On the online social network Twitter, users can post information, which is then broadcast to their followers, i.e., the other Twitter users that follow that user. This dataset tracks the spread of 65,000 unique URLs through the Twitter social network during one month in 2010. Similarly to *Digg*, we can study social influence and information diffusion by examining whether ($Y = 1$) or not ($Y = 0$) a user posts a URL after being exposed to it when one of his or her friends posts. The features associated with each exposure event are the same as those for *Digg*.

We compare the average predictive performance across 5 holdout sets of S3D to Lasso regression, elastic net regression, random forests, and linear SVM. We show that S3D can achieve competitive performance with the benchmark algorithms with a smaller set of features. Finally, we explore our tuned S3D models and demonstrate their utility to understanding human behaviors in Sect. g. Relevant datasets and codes[e] can be used to replicate the following results.

4.1 Tuning hyperparameters

An essential part of training a statistical model is hyperparameter tuning—in the case of S3D, selecting the parameters λ and k. This procedure is illustrated for Stack Exchange data in Fig. 2, where we show the total R^2 at each step of the algorithm for various values of λ, as well as the AUCs at these steps computed on the held-out tuning data. Overly small values of λ perform quite poorly on the held-out data, as they produce very fine-grained bins that overfit the data. Larger values of λ avoid being too fine-grained—the R^2 on the training set increases initially but diverges again as extra features selected in additional steps overfit the data (as shown through the decreasing performance on the held-out data; Fig. 2 bottom). Parameter k (x-axis in Fig. 2) controls the number of steps of S3D, thus picking the optimal model between underfitting and overfitting. Supplementary file *s3d_hyperparameter_df.csv*[f] reports the best hyperparamters for all datasets, across the 4-fold inner CV processes.

4.2 Prediction performance

Figures 3 and 4 report performance on the outer CV for all datasets S3D, random forests, linear SVM, and logistic regressions (both Lasso and elastic net). Overall S3D achieves predictive performance comparable to other state-of-the-art machine learning methods.

In most cases, S3D's performance is similar to that of logistic regression and linear SVM. Its performance relative to random forest is especially remarkable considering the difference in complexity of the models. S3D uses a subset of features and a simple m-dimensional hypercube to make predictions, in stark contrast to random forests, which use all features and learn many decision trees. In contrast, S3D selects a small set of features, producing more parsimonious models as compared to Lasso and elastic net regression (Fig. 5).

S3D is especially useful as a feature selection method. Using just the few features selected by S3D to train a random forest leads to highly competitive performance on the regression task for many datasets (RF-S3D bars in Fig. 4). Remarkably, its performance is often better than that of the random forest trained on the full feature set. This is likely because features selected by S3D are uncorrelated with each other; while correlations among features used by the random forest reduce performance.

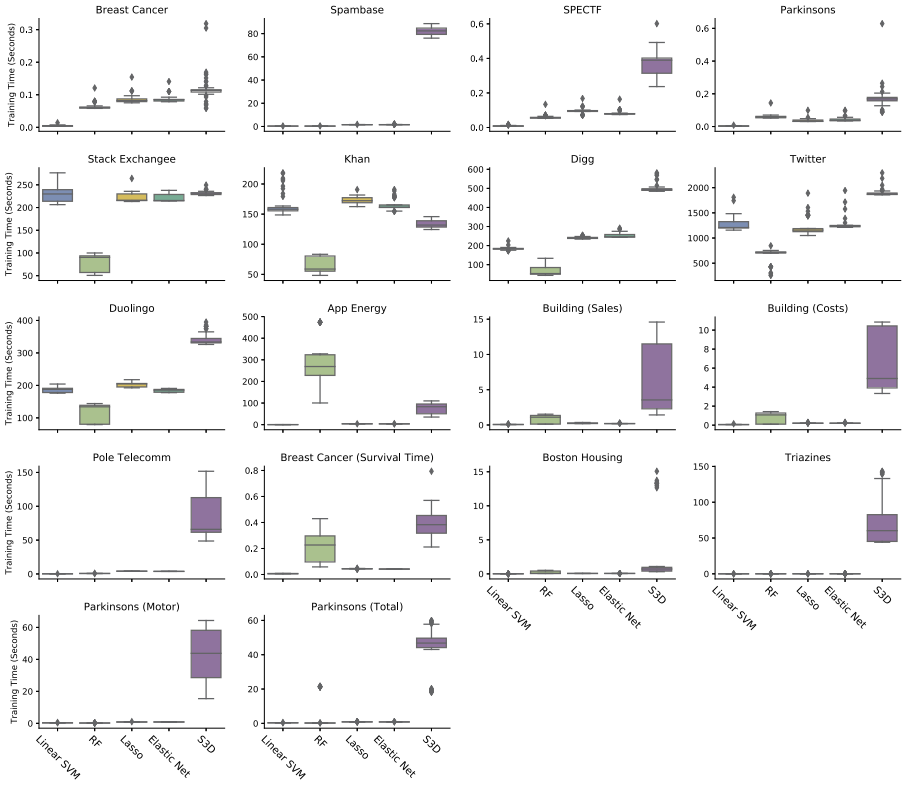

Figure 6 Boxplots of training time (seconds) of all five algorithms

Finally, we show that the runtime of S3D is competitive to the other four algorithms (Fig. 6). For each dataset, all models were trained using the best parameters found in inner CV and full training sets over each split (recall that there are five splits) repeated ten times. In other words, there is no cross validation in the evaluation of runtime, but only in training with the optimal parameters selected in the previous performance evaluation steps. Therefore, each box in Fig. 6 shows the distribution of training time over 50 runs. Note that the Python package Scikit Learn [24] is used to implement logistic regression, random forest, and linear SVM, therefore producing superb runtime performance, as it is highly optimized. We believe that the implementation of S3D can be further improved. For instance, the timing of S3D includes reading the input file, whereas the other four methods do not require this. Furthermore, Fig. 6 only reflects training time with one set of hyperparameters for each model. While random forests manifest outstanding efficiency, it is worth noting that the large amount of hyperparameters (in this study, we searched for four; there are at least four more) will inevitably lead to undesirably long hours of grid search. On the other hand, S3D only needs two (λ and k), which substantially reduce user effort in hyperparameter tuning.

4.3 Analyzing human behavior with S3D

In this section, we present a detailed description of applying S3D to understand human behaviors using *Stack Exchange* data.[g] We additionally included *Digg* in Sect. 4.3.2 to demonstrate visualizations of the learned S3D model. Specifically, we used the best hyperparameters during cross validation: $\lambda = 0.001$ and $k = 5$ for *Stack Exchange*; $\lambda = 0.001$ and $k = 3$

for *Digg*. In order to provide the most comprehensive explanation of the data, we applied S3D to *all available data* using hyperparameters that were seen most often during cross validation.

4.3.1 Feature selection and correlations

For the large scale behavioral data, S3D selects a subset of features that collectively explain the largest amount of the variation in the outcome variable. It also quantifies correlations between selected features to unselected ones. In the following, we describe selected features and examine the effects of the unselected ones.

We give a detailed description of the features selected at each step (Fig. 7) and the resulting feature network (Fig. 8). Figure 7 visually ranks the features by showing the amount of variation explained by each feature at every step of the algorithm. The features selected at each step are outlined in black. The first and most important feature selected is *the number of answers provided before this question*. This feature, for one thing, indicates how active a user is in the community. For another, it implicitly reflects a user's capability. Interestingly, there is obvious dependencies between the number of previous answers and (1) *reputation*, (2) *signup duration/percentile*, and (3) *code lines*. Given the amount of previous answers in the model, the contribution of these features decreases dramatically. The second feature S3D selects is *signup percentile*, which measures answerers' "age" on Stack Exchange as a percentile rank. Intuitively, the longer a user stays in the system, the more likely they can accumulate their reputation and capability to produce a "good" answer. It is noteworthy that *signup duration* and *percentile* share the exact amount of explained variation, which echoes the fact that the Spearman correlation between them is **1**. Following user tenure, the number of *lines of codes* is selected as the third most important feature, followed by the *number of words* and *URLs*, which all, to some extent, manifest how informative an answer is. Note that the variation explained by the features *number of words* and

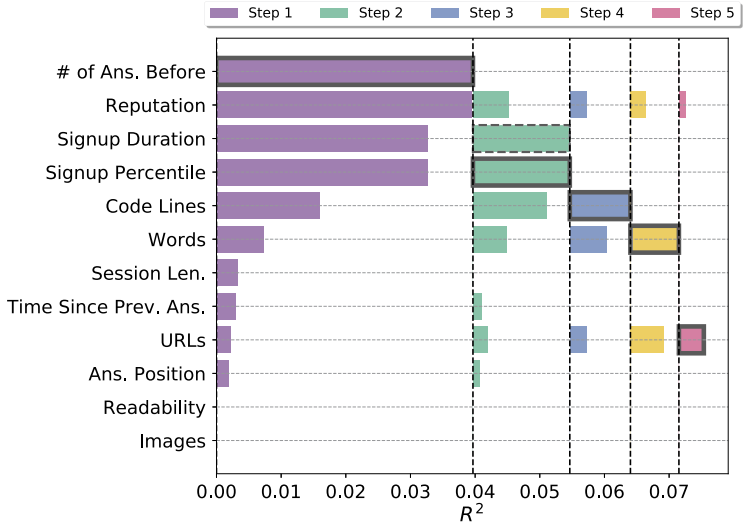

Figure 7 The amount of variation in the Stack Exchange outcome variable *performance* explained by each feature at each step of the S3D algorithm. The feature that explains most of the remaining variation at each step is highlighted here with a solid black rectangle surrounding its bar, whereas dashed rectangles indicate that the corresponding feature has the same amount of contribution (i.e., ΔR^2) but not selected

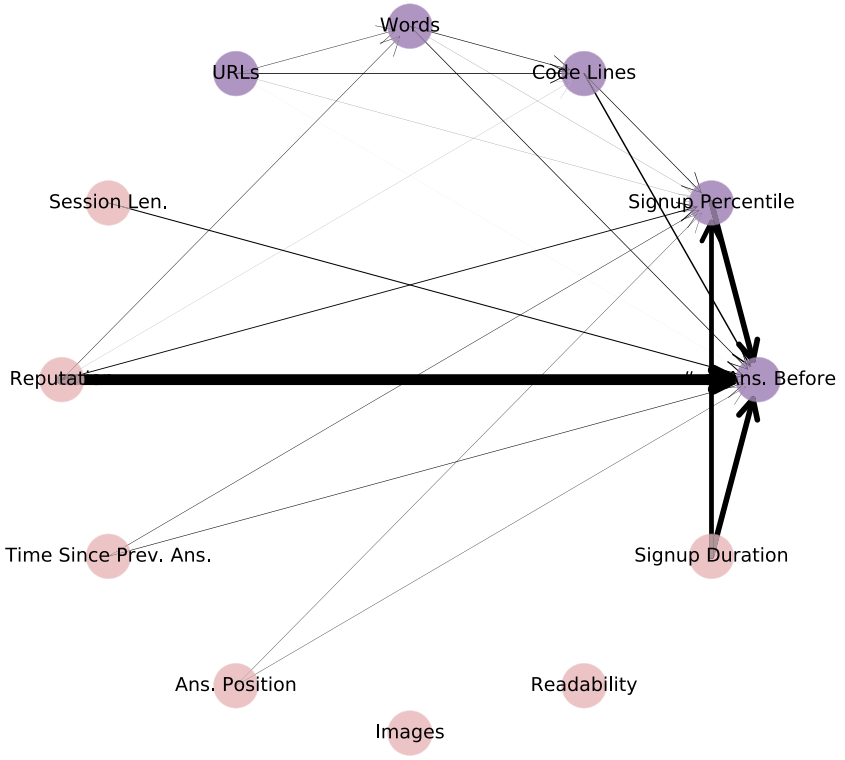

Figure 8 The feature network for Stack Exchange, showing the variation in the outcome from the features that have not been selected (pink) through the selected features (purple). Edge width is proportional to weights described in Sect. 3.2.1—the thicker a link between two features is, the more correlated they are

URLs exceeds the variation explained by these features in the first step, leading to an interesting implication that there may exist an *interaction* effect. In particular, given answers with the same *number of code lines* and by answerers who signed up in similar time period and shared similar activeness, the *number of words* and *URLs* will contribute more to the final acceptance probability. The ability to identify moderation effects among variables, in fact, is a fascinating characteristic of S3D when analyzing heterogeneous behavioral data.

With $R^2 = 0.075$, the five selected features collectively explain the largest amount of variation in whether an answer is accepted by the asker as the best answer to his or her question. The unselected features have been made redundant by the selected features. Such redundancies can be represented as a directed and weighted network through the coefficients of Equation (10), as shown in Fig. 8. Specifically, links between selected features (purple) the unselected (pink) features show the variation in the outcome explained by the pink node can be explained by the purple node. The network visualizes the correlations and the significance of unselected features. While some of the correlations are obvious, such as those between the *number of answers*, user *reputation*, and tenure length (i.e., *signup duration*/*percentile*), others are less evident. For example, there are links from *reputation* to the number of *words*, and *code lines*, implying that reputable users may provide more detailed answers containing informative clues such as references to related webpages and sample codes. Although relatively weak, the link from answer position pointing towards the number of answers a user provided before and signup percentile alludes that senior users may tend to be more active and engaging in the community, therefore being

Table 2 Features found to be important by random forest in Stack Exchange and Digg, along with their relative weights

Stack exchange		Digg	
feature	*weight*	*feature*	*weight*
Signup Percentile	0.148	Activity	0.258
Signup Duration	0.143	Time Since Last Tweet	0.152
Reputation	0.130	Outdegree	0.068
# Ans. Before	0.125	Info Received	0.067
Time Since Prev Ans.	0.117	Indegree	0.058
Words	0.114	Meme Pop. (Current)	0.035
Readability	0.101	Meme Age	0.033
Code Lines	0.066	Neighb Indegree	0.032
Session Len.	0.023	Neighb Activity	0.032
URLs	0.022	Meme Pop. (Recent)	0.032
Ans. Position	0.012	Neighb Info Received	0.031
Images	0.000	Neighb Outdegree	0.030
		Order	0.030
		Inv. Exposure Rate	0.029
		Time Last/Second to Last Exposures	0.028
		Time Last/First Exposures	0.026
		# Exposures	0.024
		Hour	0.023
		Day	0.013

early answerers to many questions. The feature network, in this manner, not only lets us analyze which unselected features are explanatory of an outcome variable, but to which selected features they are correlated and are made redundant, providing a tool to suggest further exploration of correlations within the data.

For comparison, features found to be important by the random forest algorithm are shown in Table 2, along with their weights. While the top two features selected by S3D for Stack Exchange are also highly ranked by random forest, the latter considers other features to be more important than the number of *words*, *code lines* and *URLs*, which S3D selected as important features. Random forest ranks highly features like *Signup Duration* and *Reputation*, which are highly correlated with existing features *Signup Percentile* and *# Ans. Written by User Before*. These correlated features are not useful for prediction, and may actually hurt performance. This is the reason why some feature selection algorithms, such as minimum redundancy method [37], filter out redundant, highly correlated features.

In the Khan Academy dataset, S3D selects as important features: (1) the *time* it takes the user to solve the problem; (2) the *number of problems* that the user has solved on the first attempt without hints; (3) *time since previous problem*; (4) the number of *first five problems* solved correctly on the first attempt; (5) *index of the session* among all of that user's sessions; (6) *index of the problem* within its session. It is noteworthy that the second and fourth features here are analogous to *signup duration* and *reputation* on Stack Exchange, as the number of problems that a user solves correctly on their first attempt is a combination of both skill and tenure.

For the Duolingo language learning platform, S3D picks similar skill-based features: (1) the number of *all lessons* completed perfectly; (2) the number of *prior lessons* completed; (3) the number of *distinct words* in the lesson. Similarly, the first feature here is equivalent to the second and fourth feature selected in Khan Academy, which quantifies both skill and tenure.

In the case of Digg social network, to explain whether a user will "digg" (or "like") a story recommended by friends, S3D selects as important features: (1) user *activity* (how many stories this user recommended); (2) the amount of *information received* by this user from the people she follows; (3) current popularity of this story, and (4) user in-degree. The first two features describe how a user processes and receives information, while the third one reflects how "viral" a story is, and fourth features is how many people the user follows. The features selected by S3D are also ranked highly by random forest (Table 2); although, features highly correlated with these, such as *time since last tweet*, which is correlated with user *activity*, and *indegree*, which is correlated with the amount of *information received*, are ranked equally high. Since their effect is already captured by the selected features, they are not needed in the model.

For Twitter, S3D selects (1) the amount of *information received* by this user; (2) *in-degree* (i.e., the number of followers and thus popularity) of friends; (3) the number of *times* this user has been exposed to this meme; (4) user *activity*; (5) the age of this tweet; (6) user's *out-degree* (followees). S3D identifies the information received by the user as an important feature for both Digg and Twitter, which highlights important role that cognitive load plays in information spread online [38]. On the other hand, the differences such as the lesser importance of user activity and greater importance of a user's friends in Twitter suggests interesting disparities in the manner of information diffusion on these two platforms.

4.3.2 Model analysis

S3D is a promising tool for data exploration. By iteratively selecting features and measuring the amount of outcome variation they explain (Fig. 7), we can visualize the important dimensions of the model to fully understand both effects of important features and the corresponding predictions. See Additional file 1 for a detailed step-by-step illustration.

For Stack Exchange, S3D selects five important features. We visualize the model with the first four features in Fig. 9, that unfolds the $m = 4$ hypercube learned by the model. It shows how the expectation (top plot) and the corresponding prediction (bottom plot) that the answer will be accepted as best answer, vary as a function of the four selected features. The prediction threshold selection is described in Sect. 3.2.2. Each row of plots in Fig. 9 corresponds to a single bin of the first selected feature *number of answers before*, while each column corresponds to bins of the second feature *signup percentile rank*. Individual plots vary according to the third and fourth features *code lines* and *words*. It is quite evident that variation in the outcome (i.e., Fig. 9 top plot) is greater between plots than within plots, a result of the fact that features are picked *successively* to explain such variation. These plots show the collective effects of these four features: acceptance *increases* with the user's experience (*number of answers before* feature) and tenure (*signup percentile rank*). Furthermore, longer answers with more *words* and *lines of code* are more likely to be accepted as best answers. Another interesting pattern emerges when the number of answers provided before is above 3587: the acceptance rate rises when *signup percentile* goes down. In other words, given a high level of user engagement in the community, newer users tend to produce answers that have higher chances of being accepted. On the other hand, more senior users tend to have a higher probability of having their answers accepted, when the number of previous answers is lower.

We also examine the S3D model learned for Digg to illustrate its effectiveness on highly unbalanced and heterogeneous data. Here, S3D selects as important features *user activity* (i.e., how often a user posts per day), *information received* (the number of stories, or

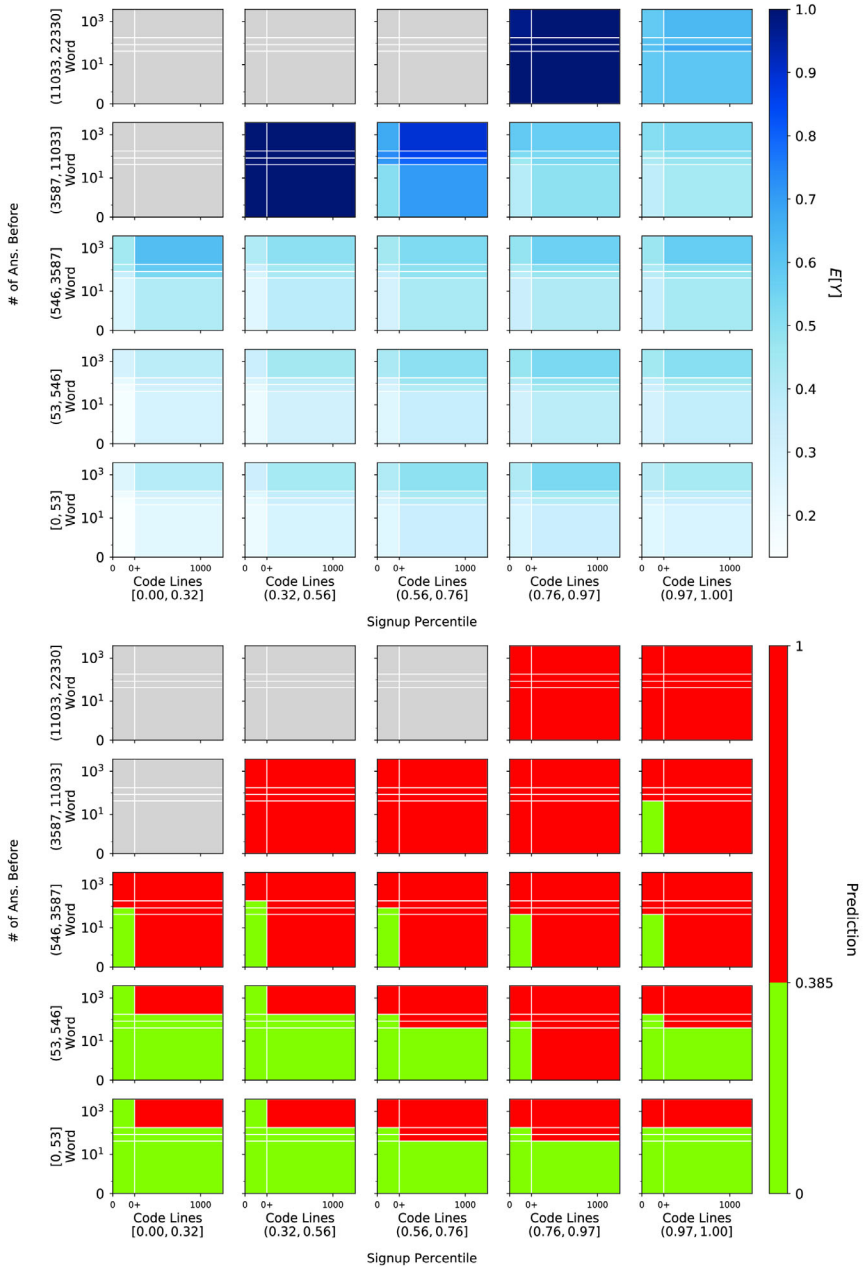

Figure 9 Visualization of the S3D model learned for Stack Exchange, showing the decomposition of feature space defined by the four most important features. These plots represent the partition of the 4-dimensional hypercube, and show how the acceptance probability (top) and corresponding predicted acceptance (bottom; red for label 1 and green for 0) of data points vary within the space. *Gray* areas have no observations

memes, a user's friends recommend), the *current popularity of the story*, i.e., how many users have recommended it, *user in-degree*. The model, presented in Fig. 10, shows extreme heterogeneity in data with values of features and adoption probabilities varying widely, and notable here is S3D's ability to learn appropriate binnings of the features over many orders of magnitude. Specifically, the figure shows that the probability of a user to adopt a story *increases* when he/she is more active in the community (see also Figure S5),

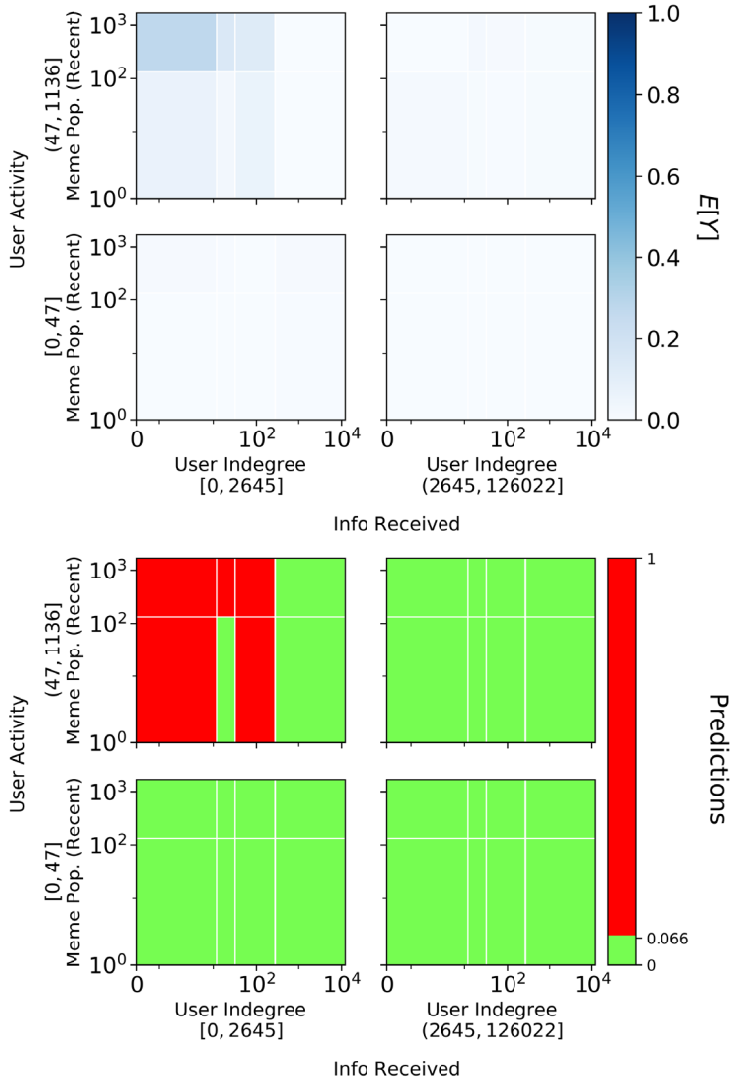

Figure 10 Visualization of the S3D model learned for Digg, showing the decomposition of feature space defined by *four* most important features. Each plot shows adoption probability of a meme within each block in the 3D feature subspace. Each bottom plot shows predicted meme adoptions. *Gray* areas have no observations

but *decreases* as users receive more information from friends (see also Figure S6). Specifically, given relatively low activity (e.g., adopting fewer than 505 stories), those users seeing less information from friends are more likely to adopt a new story (corresponding to higher color intensity on the left hand side in each plot). The highly active users, on the other hand, also tend to receive more information—around 1000 stories per day—from friends. However, they too are less likely to adopt a new story as they receive more information from friends. These two features—*user activity* and *information received*—represent the interplay between information processing and cognitive load. Our analysis highlights the extent of to which information overload, which occurs when users receive more information than they can effectively process, inhibits adoption and spread of memes online [38–40].

The third feature *current popularity* shows the impact of story popularity (i.e., virality or stickiness) on adoption. Our model shows that more popular stories are more likely to be adopted by individuals, as would be expected of viral memes. Striking is the absence of features related to the number or timing of exposures, either as selected features or in the feature network. The exposure effects may be quite subtle or even non-existent. The latter suggests that information on Digg spreads as a simple contagion where the probability of adopting a meme is independent of the number of exposures [41, 42].

The large heterogeneity as a function of basic node features has important implications for the inference of social contagion, because heterogeneity and underlying confounders may distort analysis. A possible approach to such inference is to decompose the feature space, as in Fig. 10, and statistically test the effect of multiple exposures in the resulting homogeneous blocks, an approach that would ensure that the most important factors that best explain the variation in the adoption of information have been conditioned on.

5 Conclusion

We have introduced S3D, a statistical model with low complexity but strong predictive power that offers potential to greatly expand the scope of predictive models and machine learning. S3D provides not only predictive capabilities but also explanation through its comprehensive description of data. Learning from the data in a structured manner, S3D allows us to construct the hierarchy of features or co-variates important in explaining an outcome, and allows us to examine the effect of these features on the outcome variable through visualization of the model in its projected form (e.g., Figs. 9 and 10). This is a positive step towards transparent algorithms that can be examined for bias, which presents a major stumbling block in the development and application of machine learning. Furthermore, S3D has the added benefit of quantifying explained variation in features unselected by the algorithm, as userful component for practitioners who are often concerned with the relationship between specific co-variates and an outcome variable.

We have demonstrated the effectiveness of S3D on a variety of datasets, including benchmarks and real-world behavioral data, where it predicts outcomes in the held-out data at levels comparable to state-of-the-art machine learning algorithms. Its potential for interpreting complex behavioral data through feature ranking, identifying feature correlations and visualization, however, goes beyond these alternate methods. Our approach reveals the important factors in explaining human behavior, such as competition, skill, and answer complexity when analyzing performance on Stack Exchange or essential user attributes such as activity and information load in the social networks Digg and Twitter. Aside from increasing our understanding of social systems, knowledge about what factors affect behavioral outcomes can also help us design of social platforms that improve human performance, including, for example, optimizing learning on educational platforms [2, 43] or fairer judicial decisions [7]. The insights gained from the model can help design effective intervention strategies that change behaviors so as to improve individual and collective well-being. Note that while S3D, as currently described, works with binary or continuous-valued outcomes, it may be possible to extend it also to categorical outcomes.

Moving forward, there are many areas where S3D can make a considerable impact, but we here highlight two that are of considerable interest to the authors. The first is the mathematical modeling of human behavior, where researchers require tractable models that are amendable to analysis but that also have a level of complexity that allow them to accurately capture behaviour [44–46]. Here, S3D can be used as a tool to extract ingredients

(i.e., co-variates) important to such models, and also functional forms that are required in such models. The second is the use of S3D by practitioners to both explain predictions and analyze interventions based on these predictions. Transparency should be a key requirement for algorithms applied to sensitive areas such as predicting recidivism, and our work here shows that simple algorithms, such as S3D, can meet this requirement without sacrificing predictive accuracy. The development of machine learning tools should not be restricted to optimizing one single metric (predictive power), as other ingredients, such as interpretability, can improve how these methods effect society and are perceived thereof.

Additional material

Additional file 1: Supplementary information (PDF 527 kB)

Acknowledgements
Authors are grateful to Raha Moraffa for her help setting up the cross validation framework and useful suggestions for improving performance.

Funding
This material is based upon work supported by the Defense Advanced Research Projects Agency (DARPA) and the Army Research Office (ARO) under Contracts No. W911NF-17-C-0094 and No. W911NF-18-C-0011, and in part by the James S. McDonnell Foundation. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of DARPA and the ARO.

Abbreviations
S3D, Structured Sum-of-Squares Decomposition; SST, Total sum of squares; SVM, Support Vector Machine; CV, Cross validation; AUC, Area under the curve; RF, Random Forest; Q&A, Question-answering.

Availability of data and materials
The code and data required for replicating reported results are available here:
https://github.com/peterfennell/S3D/tree/paper-replication

Competing interests
The authors declare that they have no competing interests.

Authors' contributions
The main idea of this paper was proposed by PGF and KL. PGF and KL designed the algorith; PGF implemented the algorithm; ZZ carried out the validation studies. All authors participated in preparing the manuscript. All authors read and approved the final manuscript.

Author details
[1]USC Information Sciences Institute, Marina del Rey, USA. [2]City University of Hong Kong, Kowloon Tong, Hong Kong.

Endnotes
[a] For classification tasks, each fold is made by preserving the ratio of samples in each class (i.e., stratified).
[b] S3D is available as C++ code and Python wrapper: see [12].
[c] https://www.dcc.fc.up.pt/~ltorgo/Regression/DataSets.html
[d] https://github.com/duolingo/halflife-regression
[e] https://github.com/peterfennell/S3D/tree/paper-replication
[f] https://raw.githubusercontent.com/peterfennell/S3D/paper-replication/replicate/s3d_hyperparameter_df.csv
[g] See https://github.com/peterfennell/S3D/blob/paper-replication/replicate/3-visualize-models.ipynb for results on the other four human behavior datasets.

Publisher's Note

Received: 24 September 2018 Accepted: 10 June 2019 Published online: 27 June 2019

References

1. Hofman JM, Sharma A, Watts DJ (2017) Prediction and explanation in social systems. Science 355(6324):486–488. https://doi.org/10.1126/science.aal3856

2. Settles B, Meeder B (2016) A trainable spaced repetition model for language learning. In: Proceedings of the 54th annual meeting of the association for computational linguistics (volume 1: long papers). Association for Computational Linguistics, Stroudsburg, pp 1848–1858. https://doi.org/10.18653/v1/P16-1174
3. Thaler RH, Sunstein CR (2009) Nudge: improving decisions about health, wealth, and happiness, rev and expanded edn. Penguin Books, New York
4. Lazer D, Pentland A, Adamic L, Aral S, Barabasi A-L, Brewer D, Christakis N, Contractor N, Fowler J, Gutmann M, Jebara T, King G, Macy M, Roy D, Van Alstyne M (2009) Computational social science. Science 323(5915):721–723. https://doi.org/10.1126/science.1167742
5. Lipton ZC (2018) The mythos of model interpretability. ACM Queue 16(3):30. https://doi.org/10.1145/3236386.3241340
6. Hofman JM, Sharma A, Watts DJ (2017) Prediction and explanation in social systems. Science 488:486–488. https://doi.org/10.1126/science.aal3856
7. Kleinberg J, Lakkaraju H, Leskovec J, Ludwig J, Mullainathan S (2017) Human decisions and machine predictions. Q J Econ 133(1):237–293. https://doi.org/10.1093/qje/qjx032
8. Dressel J, Farid H (2018) The accuracy, fairness, and limits of predicting recidivism. Sci Adv 4(1):5580
9. Blyth CR (1972) On Simpson's paradox and the sure-thing principle. J Am Stat Assoc 67(338):364–366. https://doi.org/10.1080/01621459.1972.10482387
10. Alipourfard N, Fennell PG, Lerman K (2018) Can you trust the trend: discovering Simpson's paradoxes in social data. In: Proceedings of the eleventh ACM international conference on web search and data mining—WSDM'18. ACM Press, New York, pp 19–27. https://doi.org/10.1145/3159652.3159684. 1801.04385
11. Breiman L, Friedman JH, Olshen RA, Stone CJ (1984) Classification and regression trees, 1st edn. Wadsworth Publishing, New York
12. Fennell PG (2018) GitHub. https://github.com/peterfennell/S3D
13. Zou H, Hastie T (2005) Regularization and variable selection via the elastic net. J R Stat Soc, Ser B, Stat Methodol 67(2):301–320. https://doi.org/10.1111/j.1467-9868.2005.00503.x
14. Hastie T, Tibshirani R, Friedman J (2009) The elements of statistical learning, 2nd edn. Springer series in statistics. Springer, New York. https://doi.org/10.1007/978-0-387-84858-7. 1010.3003
15. Chipman HA, George EI, McCulloch RE (2010) BART: Bayesian additive regression trees. Ann Appl Stat 4(1):266–298. https://doi.org/10.1214/09-AOAS285. 0806.3286
16. Friedman J (1991) Multivariate adaptive regression splines. Ann Stat 19(1):1–67. https://doi.org/10.2307/2241837
17. Breiman L (2001) Random forests. Mach Learn 45(1):5–32. https://doi.org/10.1023/A:1010933404324
18. Moore A, Murdock V, Cai Y, Jones K (2018) Transparent tree ensembles. In: The 41st international ACM SIGIR conference on research & development in information retrieval. SIGIR'18. ACM, New York, pp 1241–1244. https://doi.org/10.1145/3209978.3210151
19. Dorie V, Harada M, Carnegie NB, Hill J (2016) A flexible, interpretable framework for assessing sensitivity to unmeasured confounding. Stat Med 35(20):3453–3470. https://doi.org/10.1002/sim.6973
20. Stoddard G (2015) Popularity dynamics and intrinsic quality in reddit and hacker news. In: Proceedings of the 9th international conference on web and social media, ICWSM 2015, pp 416–425. https://doi.org/10.1145/2740908.2742470. 1501.07860
21. Sinatra R, Wang D, Deville P, Song C, Barabási A-L (2016) Quantifying the evolution of individual scientific impact. Science 354(6312):5239
22. Colizza V, Barrat A, Barthelemy M, Vespignani A (2006) The role of the airline transportation network in the prediction and predictability of global epidemics. Proc Natl Acad Sci 103(7):2015–2020. https://doi.org/10.1073/pnas.0510525103. 0507029v1
23. Boser BE, Guyon IM, Vapnik VN (1992) A training algorithm for optimal margin classifiers. In: Proceedings of the fifth annual workshop on computational learning theory—COLT'92. ACM Press, New York, pp 144–152. https://doi.org/10.1145/130385.130401
24. 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 É (2011) Scikit-learn: machine learning in Python. J Mach Learn Res 12:2825–2830
25. Fernández-Delgado M, Cernadas E, Barro S, Amorim D, Fernández-Delgado A (2014) Do we need hundreds of classifiers to solve real world classification problems? J Mach Learn Res 15:3133–3181
26. Chang Y-W, Hsieh C-J, Chang K-W, Ringgaard M, Lin C-J (2010) Training and testing low-degree polynomial data mappings via linear SVM. J Mach Learn Res 11:1471–1490
27. Dheeru D, Karra Taniskidou E (2017) {UCI} Machine Learning Repository. http://archive.ics.uci.edu/ml
28. Street WN, Wolberg WH, Mangasarian OL (1993) Nuclear feature extraction for breast tumor diagnosis. ISandT/SPIE International Symposium on Electronic Imaging: Science and Technology 1905, 861–870. https://doi.org/10.1117/12.148698
29. Kurgan LA, Cios KJ, Tadeusiewicz R, Ogiela M, Goodenday LS (2001) Knowledge discovery approach to automated cardiac SPECT diagnosis. Artif Intell Med 23(2):149–169. https://doi.org/10.1016/S0933-3657(01)00082-3
30. Little MA, McSharry PE, Roberts SJ, Costello DAE, Moroz IM (2007) Exploiting nonlinear recurrence and fractal scaling properties for voice disorder detection. Biomed Eng Online 6(1):23. https://doi.org/10.1186/1475-925X-6-23. 0707.0086
31. Candanedo LM, Feldheim V, Deramaix D (2017) Data driven prediction models of energy use of appliances in a low-energy house. Energy Build 140:81–97. https://doi.org/10.1016/j.enbuild.2017.01.083
32. Rafiei MH, Adeli H (2016) A novel machine learning model for estimation of sale prices of real estate units. J Constr Eng Manage 142(2):04015066. https://doi.org/10.1061/(ASCE)CO.1943-7862.0001047
33. Weiss SM, Indurkhya N (1995) Rule-based machine learning methods for functional prediction. J Artif Intell Res 3(1995):383–403. https://doi.org/10.1613/jair.199 9512107
34. Harrison D, Rubinfeld DL (1978) Hedonic housing prices and the demand for clean air. J Environ Econ Manag 5(1):81–102. https://doi.org/10.1016/0095-0696(78)90006-2
35. King RD, Hirst JD, Sternberg MJE (1995) Comparison of artificial intelligence methods for modeling pharmaceutical QSARS. Appl Artif Intell 9(2):213–233. https://doi.org/10.1080/08839519508945474

36. Little MA, McSharry PE, Hunter EJ, Spielman J, Ramig LO (2009) Suitability of dysphonia measurements for telemonitoring of Parkinson's disease. IEEE Trans Biomed Eng 56(4):1015–1022. https://doi.org/10.1109/TBME.2008.2005954
37. Auffarth B, López M, Cerquides J (2010) Comparison of redundancy and relevance measures for feature selection in tissue classification of ct images. In: Industrial conference on data mining. Springer, Berlin, pp 248–262
38. Hodas NO, Lerman K (2012) How visibility and divided attention constrain social contagion. In: Proceedings—2012 ASE/IEEE international conference on privacy, security, risk and trust and 2012 ASE/IEEE international conference on social computing, SocialCom/PASSAT 2012. IEEE Comput. Soc., Los Alamitos, pp 249–257. https://doi.org/10.1109/SocialCom-PASSAT.2012.129
39. Ver Steeg G, Ghosh R, Lerman K (2011) What stops social epidemics? In: Proceedings of 5th international conference on weblogs and social, Media
40. Gomez-Rodriguez M, Gummadi KP, Schölkopf B (2014) Quantifying information overload in social media and its impact on social contagions. In: Proceedings of the 8th international conference on weblogs and social media, ICWSM 2014, pp 170–179
41. Centola D, Eguíluz VM, Macy MW (2007) Cascade dynamics of complex propagation. Phys A, Stat Mech Appl 374(1):449–456. https://doi.org/10.1016/j.physa.2006.06.018. 0504165
42. Hodas NO, Lerman K (2014) The simple rules of social contagion. Sci Rep 4(1):4343. https://doi.org/10.1038/srep04343. 1308.5015
43. Rendle S (2012) Factorization machines with libFM. ACM Trans Intell Syst Technol 3(3):1–22. https://doi.org/10.1145/2168752.2168771
44. Watts DJ (2002) A simple model of global cascades on random networks. Proc Natl Acad Sci 99(9):5766–5771. https://doi.org/10.1073/pnas.082090499
45. Fernández-Gracia J, Suchecki K, Ramasco JJ, San Miguel M, Eguíluz VM (2014) Is the voter model a model for voters? Phys Rev Lett 112(15):158701. https://doi.org/10.1103/PhysRevLett.112.158701
46. O'Sullivan DJP, O'Keeffe GJ, Fennell PG, Gleeson JP (2015) Mathematical modeling of complex contagion on clustered networks. Front Phys 3:71. https://doi.org/10.3389/fphy.2015.00071