Advances in Data Analysis and Classification (2020) 14:801–819
https://doi.org/10.1007/s11634-020-00418-3

REGULAR ARTICLE

A comparison of instance-level counterfactual explanation algorithms for behavioral and textual data: SEDC, LIME-C and SHAP-C

Yanou Ramon[1] · **David Martens**[1] · **Foster Provost**[2] · **Theodoros Evgeniou**[3]

Received: 1 July 2019 / Revised: 13 July 2020 / Accepted: 18 August 2020 / Published online: 2 September 2020

Abstract
Predictive systems based on high-dimensional behavioral and textual data have serious comprehensibility and transparency issues: linear models require investigating thousands of coefficients, while the opaqueness of nonlinear models makes things worse. Counterfactual explanations are becoming increasingly popular for generating insight into model predictions. This study aligns the recently proposed linear interpretable model-agnostic explainer and Shapley additive explanations with the notion of counterfactual explanations, and empirically compares the effectiveness and efficiency of these novel algorithms against a model-agnostic heuristic search algorithm for finding evidence counterfactuals using 13 behavioral and textual data sets. We show that different search methods have different strengths, and importantly, that there is much room for future research.

Keywords Comparative study · Counterfactual explanations · Instance-level explanations · Explainable artificial intelligence · Explanation algorithms · Binary classification · Behavioral data · Textual data

Mathematics Subject Classification 90C27 (Combinatorial optimization) · 90C59 (Approximation methods and heuristics in MP) · 68T01 (General topics in AI)

1 Introduction

The proliferation of big data architectures has resulted in many applications having an increasingly large impact on business and society (Junqué de Fortuny et al. 2013). We

✉ Yanou Ramon
yanou.ramon@uantwerpen.be

1 Department of Engineering Management, University of Antwerp, Prinsstraat 13, Antwerp, Belgium

2 Department of Technology, Operations and Statistics, Stern School of Business, New York, NY, USA

3 Decision Sciences and Technology Management, INSEAD, Fontainebleau, France

focus on two sorts of big data. The first is behavioral data, defined as data that capture human behavior through the actions and interactions of people (Shmueli 2017), which can be used for various predictive purposes. For instance, what you "Like" on Facebook is predictive of your openness and many other personality traits (Kosinski et al. 2013, 2017), while the accounts you pay to or webpages you visit are predictive features for product interest (Martens et al. 2016) and creditworthiness (De Cnudde et al. 2019b). The second sort of big data is textual data. Text classification is ubiquitous in business and government (Provost and Fawcett 2013). Example applications are automatic identification of spam emails (Attenberg et al. 2009), objectionable web content detection (Martens and Provost 2014) and legal document classification (Chhatwal et al. 2018), just to name a few.

Mining behavior and text data can result in highly accurate classification models (Junqué de Fortuny et al. 2013; Provost et al. 2015), but also in very complex model structures. The complexity arises from either the learning technique (e.g., deep learning) or the data, or both. Behavioral and textual data are typically high-dimensional and sparse. Let us consider an example that we will refer back to. We want to predict the gender of users based on the movies they have viewed. A user having watched a movie is represented by a binary feature for each movie, which results in an enormous feature space. However, each user only has watched a small number of movies, which results in an extremely sparse data matrix (almost all elements are zero). A user having watched a movie is represented by a corresponding non-zero value for that binary feature, and we refer to such features as "active". In other words, because of the sparsity, the number of active features m' of a user (the movies someone watched) is much smaller than the dimensionality m of the full feature space (*all* possible movies someone could watch). Because of these data characteristics, even intrinsically interpretable linear models are difficult to interpret because there are many thousands of features, each with their own linear coefficient; further, the features that will be brought to bear for prediction are different for every individual. Applying nonlinear techniques normally renders the reasons for a particular prediction completely opaque.

The importance of understanding individual classification decisions is well-argued in the literature (Gregor and Benbasat 1999; Freitas 2014; Martens and Provost 2014; Goodman and Flaxman 2016; Doshi-Velez and Kim 2017; Ras et al. 2018; Lipton 2018). Explanations for model predictions are often necessary for users to trust and improve the model (Gregor and Benbasat 1999). In some domains, like medical diagnosis and credit scoring, it even is a legal requirement (Martens and Provost 2014; Gregor and Benbasat 1999; Martens et al. 2007) (e.g., why was my loan application rejected?). According to Doshi-Velez and Kim (2017), the demand for interpretable models stems from a mismatch between "formal" objectives (e.g., minimal prediction error) and "ethical" objectives (e.g., privacy), which can only be validated when a certain level of interpretability is achieved.

Various approaches have been proposed for explaining model predictions (Craven 1996; Martens and Provost 2014; Ribeiro et al. 2016; Lundberg and Lee 2017; Lipton 2018; Wachter et al. 2018), varying in scope and flexibility. The scope indicates whether the method generates global explanations (for the entire feature/instance space) or instance-level explanations (for a single prediction) (Martens and Provost 2014), whereas the flexibility indicates whether the approach is model-specific or

model-agnostic. Much research focuses on model-specific explanation techniques tailored to a specific type of prediction models such as deep learning models (Samek et al. 2015; Arras et al. 2017) or random forests (Breiman 2001). In contrast, model-agnostic methods explain model predictions of any prediction model. This increases flexibility; however, often it results in substantially more computational effort (Martens and Provost 2014; Arras et al. 2017).

For this paper, we focus on the increasingly popular notion of "counterfactual" explanations (Martens and Provost 2014; Provost 2014; Chen et al. 2017; Wachter et al. 2018; Nguyen 2018; Sokol and Flach 2019). A counterfactual explanation of a model-based system's decision for a particular instance is defined as a set of "evidence" of the instance without which the system would not have made that decision. In our setting of behavioral and textual data, this evidence corresponds to a set of active features of the instance where changing these feature values to zero would lead the system not to have made that decision. Ideally, this set is minimal, meaning that the predicted class only changes when all features that are part of the counterfactual explanation are removed (feature values set to zero). Note that minimality is not always guaranteed and depends on the algorithm that is used to compute counterfactuals.

Counterfactuals have been argued to be crucial for explaining predictions on the instance-level as they pinpoint the features that led to the decision (Sokol and Flach 2019) and make the decision actionable (Chen et al. 2017; Wachter et al. 2018; Fernandez et al. 2019). In our running movie example, if we want an explanation of why a user called, say, Sam was predicted to be "male", we want to know which movies were critical for the model's decision. A counterfactual explanation shows a set of movies such that removing them from Sam's movie list would lead the predicted class to no longer be "male" (see Fig. 1a). In the context of textual and behavioral data, removing the feature from the instance is equivalent to setting the original feature value to zero or "removing the evidence". In the remaining of this study, we will consider counterfactuals based on the removal of evidence that is present in the data—for example, words that appear in a document or items that an individual has "Liked" on Facebook. These correspond to "active" features—those that are present in a sparse representation, or those that are non-zero in a traditional feature-vector representation.

In this study, we are interested in finding minimum-sized counterfactual explanations. A possible approach to find the counterfactual is to conduct a complete search through the entire space of feature combinations, starting with one feature and incrementally increasing the number of features until an explanation is found. However, this strategy scales up exponentially with the number of features, making it impracticable for high-dimensional feature spaces (Martens and Provost 2014). Consequently, there is a need for an algorithm that computes counterfactuals with a good trade-off between effectiveness (selecting the most important features for the explanation) and efficiency (computation time).

Martens and Provost (2014) proposed a heuristic best-first search for Evidence Counterfactuals (*SEDC*),[1] which is able to counterfactually explain predictions of any

[1] The original paper presented the framework for counterfactual explanations, subsequently referred to as Evidence Counterfactuals (Provost 2014; Moeyersoms et al. 2016; Chen et al. 2017). The paper discussed several methods for finding such explanations. We evaluate the heuristic best-first search SEDC algorithm here.

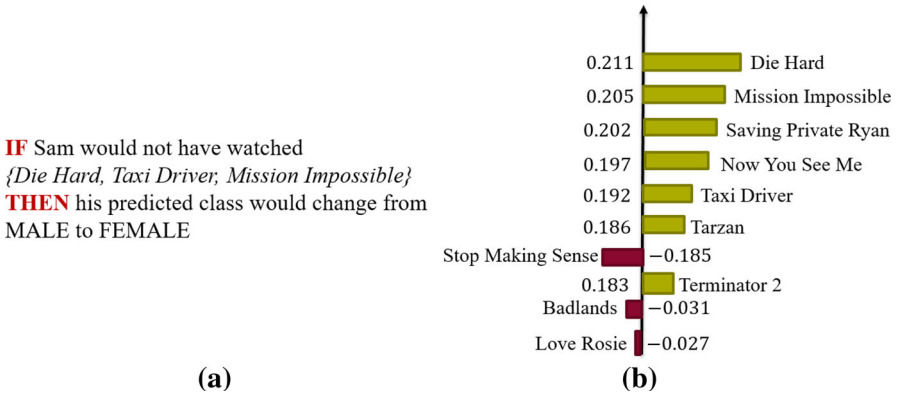

Fig. 1 Example outputs of two different types of instance-level explanations that show why Sam was classified as "male" based on his movie viewing data: a counterfactual explanation (**a**) and an additive feature attribution explanation (**b**)

classification model with sparse features. To the best of our knowledge, *SEDC* is the only existing model-agnostic explanation algorithm for counterfactuals which is able to deal with behavioral and textual data. For this reason, we use this algorithm as a benchmark in this study. The proposed algorithm by Wachter et al. (2018) cannot deal with many binary variables—a common representation for explanations for behavioral and textual data—which eliminates their algorithm from this study.

In the literature, other instance-level explanation types have been proposed for high-dimensional data sources, such as additive feature attribution explanations (Ribeiro et al. 2016; Lundberg and Lee 2017). In our movie running example, additive feature attribution explanations would show an ordered list of important movies and their corresponding importance weights—specifically, importance for this particular model decision (see Fig. 1b). Such algorithms generate an importance-ranked list of features, i.e., coefficients of a linear model, for a single instance. The idea of developing hybrid methods which connect counterfactuals with additive feature attribution explanations stems from the following reasoning: if these importance-rankings of features are sufficiently accurate, it may be possible to compute counterfactuals from them: starting from the highest-ranked feature, we remove features until the predicted class changes.[2] One novelty of this study resides in the idea that these importance rankings may be an "intelligent" starting point for searching for counterfactuals. The resulting algorithm for computing counterfactuals may be better than the existing *SEDC* algorithm. For this reason, we empirically compare the counterfactual explanation algorithms to help researchers and practitioners better understand which method is most suitable when facing behavioral or textual data.

This paper's main contributions are fourfold: (1) we propose two novel model-agnostic explanation algorithms, creating them via the combination of counterfactual explanations and additive feature attribution methods (*LIME-C* and *SHAP-C*); (2) we

[2] Fernandez et al. (2019) show that having a high importance weight (from SHAP) is neither necessary nor sufficient for a feature to be part of a counterfactual explanation. Therefore, we should be clear that this is an alternative heuristic approach.

define quantitative evaluation criteria that proxy the effectiveness and efficiency of these algorithms; (3) we perform an in-depth evaluation of the explanation quality of *LIME-C* and *SHAP-C* when applied to high-dimensional behavioral and textual data and benchmark their performance against the *SEDC* algorithm, and lastly, (4) we propose changes to the model-agnostic methods for generating counterfactuals, and discuss research directions stemming out of our findings.

2 Counterfactual explanation algorithms

To our knowledge, counterfactual explanations were first used to explain document classifications (Martens and Provost 2014), and since that time have been applied more broadly (Provost 2014; Chen et al. 2017; Wachter et al. 2018; Nguyen 2018; Sokol and Flach 2019). Martens and Provost (2014) define an explanation in counterfactual terms as a minimal set of active features such that, when removing these features from the instance, the predicted class changes.[3] For instance, in Fig. 1a, the movies *"Die Hard"*, *"Taxi Driver"* and *"Mission Impossible"* explain why Sam was classified as "male".

Consider instance $\mathbf{x} = (x_1, \ldots, x_m)$ and the feature indices $I_{\mathbf{x}} = \{1, \ldots, m\}$ for $m \in \mathbb{N}$. Let $I_A \subseteq I_{\mathbf{x}}$ represent the indices of the active features of $\mathbf{x}$ such that $\forall j \in I_A$: $x_j \neq 0, \forall j \notin I_A$: $x_j = 0$. Let $I \subseteq I_A$ be a subset of the indices of active features, then a perturbed instance $\mathbf{z}_I$ of instance $\mathbf{x}$ (hereafter referred to as $\mathbf{z}$) is defined as $\forall l \in I$: $z_l = x_l, \forall l \notin I$: $z_l = 0$. Let C be a trained classifier that is a function from instances to k classes. Instance $\mathbf{x}$ is classified by classifier C: $\mathbf{x} \rightarrow \{0, \ldots, k\}$ as class c. In this study, we define a counterfactual explanation as follows:

Definition 1 A counterfactual explanation for instance $\mathbf{x}$'s classification is a set of active features with indices $E \subseteq I_A$ such that removing all features with indices E from the instance $\mathbf{x}$ leads C to produce another classification. The perturbed instance $\mathbf{z}_I$ with $I = I_A \backslash E$ denotes the result of removing the features with indices E from instance $\mathbf{x}$. Further, a counterfactual explanation is minimal in the sense that removing any subset of E does not yield a change in class. Specifically:

A set of features with indices E is a counterfactual explanation for $C(\mathbf{x}) \Leftrightarrow$

1. $E \subseteq I_A$ (the features are active for instance $\mathbf{x}$),
2. $C(\mathbf{z}_{I_A \backslash E}) \neq c$ (the class changes), and
3. $\neg\exists\ E' \subset E$: $C(\mathbf{z}_{I_A \backslash E'}) \neq c$ (E is minimal).

Note that, for behavioral and textual data, removing features corresponds to setting the (original) feature values to zero.

We implemented the model-agnostic *SEDC* heuristic search algorithm, presented by Martens and Provost (2014), which conducts a best-first search strategy. For explaining individual predictions of linear classification models, *SEDC* is optimal in the sense

[3] Such explanations have been called Evidence Counterfactuals, referring to the feature evidence that leads the classifier to make its classification (Provost 2014; Chen et al. 2017); we will adopt this terminology to differentiate such explanations from the additive feature attribution explanations described next.

that it always finds a minimum-sized feature set that changes the predicted class (formal proof can be found in Martens and Provost (2014)). For explaining nonlinear model predictions, optimality is not guaranteed because the algorithm cuts off its search after a limit has been reached (Martens and Provost 2014). Also, because of the search cut-off, the explanations may not be minimal, i.e., a subset of the explanation set may also be a counterfactual. We further limit *SEDC*'s search by stopping after the first explanation has been found. As the empirical results below show, this means that the method is very fast; we leave assessing the full effectiveness vs. time tradeoff to future work.

Additive feature attribution methods use an explanation model g as an interpretable approximation of the trained classification model C (with corresponding scoring function f_c: $\mathbf{x} \rightarrow \mathbb{R}$) in the neighborhood of an instance $\mathbf{x}$. Two recently proposed model-agnostic methods are the linear interpretable model-agnostic explainer (LIME) (Ribeiro et al. 2016) and Shapley additive explanations (SHAP) (Lundberg and Lee 2017). In the context of text and behavior, the explanation model g is a linear function of binary variables that indicate whether the feature is "active" (original value) or "removed" (zero). Consider again the instance $\mathbf{x} = (x_1, \ldots, x_m)$ that has m' active features (note that the full feature dimension m can be much larger). The additive feature attribution explanation of instance $\mathbf{x}$ can be represented as a linear model:

$$g(\mathbf{x}') = \phi_0 + \sum_{j=1}^{m} \phi_j x'_j \tag{1}$$

where $x'_j \in \{0, 1\}$ is the binary representation of x_j (where x'_j is 1 if x_j is non-zero, else it equals 0), m is the number of features of instance $\mathbf{x}$, and $\phi_0, \phi_j \in \mathbb{R}$. For SHAP, the weights retrieved from the model also represent the (approximate) Shapley values, which have theoretically attractive properties (see Lundberg and Lee (2017) for more details). The main differences between LIME and SHAP are (1) how they generate the sample of perturbed instances, (2) the distance function π and (3) the complexity control of the explanation.

Suppose we want to explain the instance $\mathbf{x}$. Both LIME and SHAP first map the instance to a binary representation $\mathbf{x}' = (x'_1, \ldots, x'_m)$ using a mapping function $h(\mathbf{x}) = \mathbf{x}'$. Next, perturbed instances are generated from $\mathbf{x}'$, and each perturbed instance $\mathbf{z}'$ is assigned a distance weight $\pi_{\mathbf{x}'}(\mathbf{z}')$. LIME generates perturbed instances by sampling $\tilde{n}$ instances and randomly removing active features from $\mathbf{x}'$. Each perturbed instance $\mathbf{z}'$ is then mapped onto the original feature space to obtain the predicted score using the scoring function $f_c(\mathbf{z})$, which is then used as a label for training the explanation model g. Each perturbed sample is assigned a corresponding weight. For textual data, LIME uses the cosine distance as the distance function to measure the similarity between $\mathbf{x}'$ and $\mathbf{z}'$, which seems a suitable choice for behavioral data as well.[4] SHAP starts

[4] The cosine distance is defined as $cosine(\mathbf{x}', \mathbf{z}') = \frac{\mathbf{x}' \cdot \mathbf{z}'}{||\mathbf{x}'|| \cdot ||\mathbf{z}'||}$ and measures how similar two data instances are irrespective of their size i.e., the number of active features. This seems a suitable choice for behavioral and textual data instances, which can vary a lot in size (e.g., documents with varying lengths, users with different number of movies watched or Facebook pages "Liked", etc.).

by estimating distance weights for different subset sizes. A subset size is the number of non-zero elements of a perturbed instance $\mathbf{z}'$. For each subset size s, a distance weight is estimated.[5] Then, the method samples $\tilde{n}$ perturbed instances from the subset spaces, starting from the smallest (and largest) subsets. LIME trains the explanation model by using ℓ_2-regularized linear regression and controls the complexity even more by allowing exactly K features in the explanation. SHAP trains the model using ℓ_1-regularized linear regression.

Note that neither LIME nor SHAP produce non-trivial counterfactual explanations natively. However, it is straighforward to produce variants of the algorithm that do. Specifically, we can apply the efficient search algorithm for counterfactuals for linear models (Martens and Provost 2014), which we refer to as *lin-SEDC*,[6] to the importance-ranked lists generated by LIME and SHAP. We refer to these novel algorithms as *LIME-C* and *SHAP-C* where *C* stands for "Counterfactual". The (general) pseudo-code of a hybrid algorithm of additive feature attribution explanations and counterfactuals is shown in Algorithm 1. In a first step, an additive feature attribution explanation is generated, without regularizing the linear explanation model g. For LIME, this means that the complexity parameter K is set to the number of active features m', whereas for SHAP, no regularization is used. From this step, a linear model with the binary representation of the features is obtained (original value versus zero), or equivalently, we retrieve an importance-ranked list of features.

In a second step, the linear algorithm for finding counterfactuals (*lin-SEDC*) (Martens and Provost 2014) is applied to the ranked list to efficiently generate a counterfactual, if possible. In more detail: the (active) features of the linear explanation model are ranked by their estimated coefficients (from high to low coefficient). Then, in a first iteration, the feature that is ranked at the top is removed from the instance, or equivalently, its value is set to zero. If this results in a class change, a counterfactual explanation is found. If not, the set of two top-ranked features is checked for being a counterfactual explanation. If not, the set of three top-ranked features are removed from the instance, and so on, until a counterfactual explanation has been found. Both *LIME-C* and *SHAP-C* rely on random sampling to generate counterfactuals, and thus, are stochastic explanation algorithms. This is in contrast to *SEDC*, which always results in the same search tree path for finding explanations when re-running the algorithm. Moreover, note that there is no guarantee that the counterfactuals from the hybrid algorithms are always minimal.

[5] The distance function of SHAP is defined as $\pi_{\mathbf{x}'}(\mathbf{z}') = \frac{(m'-1)}{(m' \; choose \; s) s (m'-s)}$ where $(m' \; choose \; s) = \frac{m'!}{s!(m'-s)!}$. The number of active features of $\mathbf{x}'$ is represented by m' and the subset size s refers to the number of non-zero elements in perturbed instance $\mathbf{z}'$.

[6] See https://github.com/yramon/edc/tree/master/LinearEDC for open-source code (see Martens and Provost (2014) for more details on the algorithm).

Algorithm 1 Additive Feature Attribution + Counterfactuals

Input:
$\mathbf{x} = (x_1,\ldots,x_m)$ % Instance to explain with m features
$I_{\mathbf{x}} = (1,\ldots,m)$ % Indices of m features of $\mathbf{x}$
$I_A \subseteq I_{\mathbf{x}}$ % Indices of the m' active features of $\mathbf{x}$
C: $\mathbf{x} \rightarrow \{0,\ldots,k\}$ % Classifier C that maps instances to k classes, with scoring function f_c
$h(\mathbf{x}) : \mathbf{x} \rightarrow \mathbf{x}' = (x'_1,\ldots,x'_m)$ % Function h maps instance to binary representation
$maxf$ % Maximum number of features in counterfactual explanation
$maxtime$ % Maximum computation time in minutes

Output:
Feature indices of the counterfactual explanation, E % Set of feature indices E that counterfactually explains the prediction of $\mathbf{x}$
Step 1: AFA($\mathbf{x}$, h, f_c) % Additive Feature Attribution without complexity control
Output:
$\Phi = (\phi_1,\ldots,\phi_m)$ % Estimated coefficients of all features in explanation model g
$\Phi_A = (\phi_j)_{\forall j \in I_A}$ % Estimated coefficients of active features in explanation model g
t % Time elapsed in minutes
Step 2: lin-SEDC($\Phi_A, maxf, maxtime, t, C$) % *lin-SEDC* algorithm
Sort coefficients $\phi_{A,j}$ in Φ_A in descending order as $1\ldots m'$
Sort j in I_A according to the sorted coefficients vector Φ_A
$c = C(\mathbf{x})$ % Class predicted by the binary classifier
$E = \{\}$ % Initialize set of indices for counterfactual explanation
$t_{total} = t$
j=1
while $c_{new} = c$ **&** $j \leq maxf$ **&** $t_{total} \leq maxtime$ **&** $\phi_{A,j} \geq 0$ **do**
$E = E \cup \{I_{A,j}\}$ % Add the j-th element of I_A to E
$c_{new} = C(\mathbf{z}_{I_A \setminus E})$ % Class predicted if features in E are removed from $\mathbf{x}$
$j = j + 1$ % Add extra iteration
$t_{total} = t_{total} + t_{elapsed}$ % Add extra time
end while
if $c_{new} = c$ **then**
$E = \{\}$ % No counterfactual explanation is found
end if

3 Experimental setup

3.1 Data sets and models

Our experimental data comprise 10 behavioral and 3 textual data sets. All data are public, except the *Facebook* and *Fraud* data. The classification tasks are binary and vary from gender prediction to sentiment analysis. Table 1 summarizes the characteristics of the data. All data have high-dimensional feature spaces with up to hundreds of thousands of features. *Movielens_1m, Movielens_100k, KDD2015, Airline* and *Twitter* have lower-dimensional feature spaces compared to the other data sets. For all data sets, the "class-of-interest" is the minority class. A large class imbalance is present for the *Fraud* data. Also, *20news* has a large imbalance compared to the other data sets. Relatively balanced data are *Facebook*, *TaFeng* and *LibimSeTi* (imbalance values b larger than 30%). The large sparsity values p for all data indicate that the number of active features is very small compared to the total number of possible active features.

Table 1 also shows the number of test instances per data set and the average number of active features $\overline{m}'$, which is very different between the data sets. *Ecommerce* and *Flickr* have very small instances (only 2 to 3 active features), in contrast to other data such as *Movielens_1m* with instances having over 150 active features.

For the behavioral data, we build ℓ_2-regularized Logistic Regression (ℓ_2-LR) models and multi-layer perceptrons (MLP). Logistic Regression has proven to be the best-performing shallow model for big behavioral data (De Cnudde et al. 2019a), while a follow-up study demonstrated a (modest) performance improvement by deep learning models (De Cnudde et al. 2019c). For the textual data, we build bag-of-words support vector machines (SVM) with linear and RBF kernel, because they are well-established to be successful for text mining applications (Joachims 1998; Martens and Provost 2014). For preprocessing text, we remove stopwords and lemmatize tokens using the *NLTK* package in *Python*, and then, use TF-IDF[7] vectorization (Joachims 1998; Martens and Provost 2014). We use 80% of the data for training the models and 20% as test set. For ℓ_2-LR and SVM, we fine-tune the regularization parameter using a holdout set (25% of training data). For MLP, we use the best parameter configuration found by De Cnudde et al. (2019a). We build models using the *Scikit-learn* library. To make classifications, we sort the test instances by decreasing predicted scores and classify the k% top-ranked instances as positive, such that the fraction of test instances classified as positive equals the fraction of positives in the training data.

3.2 Explanations

For the experiments, we generate counterfactuals for the positively predicted test instances and we set the maximum size of the counterfactual explanation to 30, in line with questions as to the utility of explanations sets that are too large (Sokol and Flach 2019; Martens and Provost 2014). As a second algorithmic choice, we set the maximum time to compute an explanation to 5 min. For *SEDC*, we set the maximum number of iterations to 50 and we use our own *Python* implementation.[8] For *LIME-C*,[9] we use LimeText explainer[10] for generating the importance-ranked list. We set the complexity parameter K equal to the number of active features (Ribeiro et al. 2016) and we set the number of perturbed samples $\tilde{n}$ equal to 5,000 (Ribeiro et al. 2016; Nguyen 2018). Next, we compute counterfactuals from the ranked feature list using *lin-SEDC* (Martens and Provost 2014). For *SHAP-C*,[11] we first compute the linear model using the model-agnostic implementation KernelExplainer[12] with a single reference value (zero), default ℓ_1-regularization and the identity link function. Similar to *LIME-*

[7] TF-IDF is short for term frequency and inverse document frequency.

[8] See https://github.com/yramon/edc for open-source code.

[9] See https://github.com/yramon/LimeCounterfactual for open-source code.

[10] See https://github.com/marcotcr/lime. Currently, no implementation exists for behavioral data, where a single reference value of zero is used. For this reason, we artificially generated text data from the behavioral features and use the *CountVectorizer*.

[11] See https://github.com/yramon/ShapCounterfactual for open-source code.

[12] See https://github.com/slundberg/shap. We used version 0.29.3 for the experiments.

Table 1 Data characteristics of the data sets (references are placed in brackets): data type (B: behavioral, T: textual), target variable, number of instances and features, the class imbalance b of the target, the sparsity p and test set for linear and nonlinear models (percentage of instances predicted as positive are placed in brackets)

Dataset	Type	Target	Instances	Features	b (%)	p (%)	Test set (%)	$\overline{m}'_{lin}$	$\overline{m}'_{nonlin}$
Flickr* (Cha et al. 2009)	B	Comments	1,00,000	1,90,991	36.91	99.99	20,000 (20)	2.02	2.96
Ecommerce* (PKDD'15 Data Mining Competition)	B	Gender	15,000	21,880	21.98	99.99	3,000 (15)	2.60	2.67
Airline* (Crowdflower)	T	Sentiment	14,640	5,183	16.14	99.82	2,928 (15)	7.81	8.21
Twitter (Crowdflower)	T	Topic	6,090	4,569	9.15	99.74	1,218 (10)	9.52	9.35
Fraud* (n/a)	B	Fraudulent	858,131	1,07,345	6.4e−5	99.99	171,627 (1)	11.83	14.09
YahooMovies* (Yahoo Webscope Program)	B	Gender	7,642	11,915	28.87	99.76	1,529 (20)	25.24	25.00
TaFeng* (Hsu et al. 2004)	B	Age	31,640	23,719	45.23	99.90	6,328 (15)	44.32	37.24
KDD2015* (KDD cup 2015)	B	Dropout	1,20,542	4,835	20.71	99.67	24,109 (20)	49.01	46.40
20news (Lang 1995)	T	Atheism	18,846	41,356	4.24	99.84	3,770 (5)	67.96	62.77
Movielens_100k (Harper et al. 2015)	B	Gender	943	1,682	28.95	93.69	189 (25)	68.73	73.42
Facebook* (Chen et al. 2017)	B	Gender	3,86,321	1,22,924	44.57	99.94	77,265 (30)	83.03	84.55
Movielens_1m* (Harper et al. 2015)	B	Gender	6,040	3,706	28.29	95.53	1,208 (25)	168.46	153.46
LibimSeTi* (Brozovsky and Petricek 2007)	B	Gender	1,37,806	1,66,353	44.53	99.93	27,562 (30)	229.16	226.97

We use 20% of the data as test set. A * indicates that the number of positively predicted test instances used for the experiments was a random subset of 300. The average number of active features $\overline{m}'_{lin}$ and $\overline{m}'_{nonlin}$ are measured over the positively predicted test instances of the linear and nonlinear model respectively. We sort the data by increasing values of $\overline{m}'_{lin}$

C, we set the size of the neighborhood $\tilde{n}$ equal to 5000. Here as well, counterfactuals are computed from the importance-ranked list using *lin-SEDC*.

3.3 Evaluation criteria

We define the following set of performance metrics for evaluating counterfactual explanations generated by the three different algorithms:

1. *Effectiveness*
   - *Switching point*: number of features that need to be removed before the classification changes. The switching point equals the size of the counterfactual explanation.
   - *Percentage explained*: fraction of positively predicted instances for which a counterfactual explanation smaller than 30 features is found.
2. *Efficiency*
   - *Computation time*: number of seconds it takes to generate an explanation.

To compare effectiveness of the different algorithms, we need a common definition for assessing (counterfactual) explanations. Feature-ranking explanations were tied to the notion of the counterfactual implicitly by Nguyen (2018), who introduces the notion of the switching point, which is the number of features that need to be removed (or set zero)—when traversing the ranked list—before the prediction switches to another class. (This is essentially the procedure of *lin-SEDC*.) The switching point was originally introduced as a proxy for the method's ability to rank features from high to low relative importance (Arras et al. 2017; Nguyen 2018); it also gives us a straightforward method for turning the feature-ranked explanations into counterfactual explanations. (For explanations already represented as counterfactuals, such as those produced by *SEDC*, the switching point simply equals the number of features in the explanation.) Measuring the switching point is important, because in cases where the prediction is not the default prediction, simply selecting all the features would produce a class change, but would be a trivial "explanation". All else being equal, for a better importance-ranked list one would not have to choose as many features to create a counterfactual explanation, resulting in a lower switching point. We do not allow counterfactuals to be larger than 30; therefore, the switching points also will be no larger than 30. In the experiments, we also compute a random explainer for estimating the switching point, against which we benchmark our counterfactual algorithms. It randomly selects a feature and sets it to zero. If the class changes, then a switching point is found. If not, it verifies whether the predicted score at least decreased. If not, it selects another random feature. If yes, then it selects a new, random feature and evaluates whether leaving out these two features together results in a class change. This is repeated until the random algorithm finds a switching point.

Information on effectiveness is captured by the percentage explained, which indicates the fraction of instances for which a counterfactual explanation smaller than 30 features is found. More specifically, when the explanation method is not very good at identifying the most relevant features, the method will most likely result in larger

switching points. This will result in fewer instances for which a counterfactual smaller than 30 is found.

Lastly, we also compare the efficiency of available implementations of the explanation algorithms, as finding counterfactual explanations can be a hard computational problem. The computation time is important to a greater or lesser degree depending on the timeliness needs of the application. For example, whether one will compute an explanation on demand for a small number of instances at human-cognition speeds versus one will compute and cache explanations for all predictions in a high-throughput application (e.g., why was I shown this?).

4 Results: effectiveness

Table 2 shows the percentage explained by each of the algorithms. For the linear models, there are very small differences between the methods and *SEDC* is always better than or as good as the other methods. For the *LibimSeTi* data, *LIME-C* and *SHAP-C* find significantly fewer counterfactual explanations than *SEDC*.

For the nonlinear models, however, *SEDC* never produces more explanations than *LIME-C* and *SHAP-C*. *SEDC* has a significantly lower percentage explained than *LIME-C* and/or *SHAP-C* for 5 out of 13 data sets. Since in theory, without an iteration limit, the best-first search will find (all) explanations for every case, this phenomenon is due to the heuristic cut-off of the search at 50 iterations—it does not expand more than 50 feature sets (search nodes). In more detail: for some nonlinear models, removing one feature does not result in a predicted score change for any of the features. Consequently, the algorithm selects a random feature to continue with in the following iteration. The same may happen in the second iteration. These "bad" feature choices are what makes the algorithm need more than 50 iterations to find a counterfactual explanation.

Lastly, *SHAP-C* seems to have difficulties for the *Fraud* data. For *Fraud/nonlin*, only 75% of the test instances are explained. For the non-explained instances, all estimated coefficients (step 1 in Algorithm 1) are zero, so no linear explanation model was generated. We conjecture this is due to the random sampling procedure, which results in a higher number of required instances $\tilde{n}$. When setting the sample size $\tilde{n}$ to 7,000 (instead of 5,000), the percentage explained increases to a maximum of 100, indicating that this is the required number of perturbed samples needed to generate explanations. We conjecture that the "critical number of perturbed samples" increases for highly imbalanced data like *Fraud* and that this is related to the sampling procedure of *SHAP-C*.

Table 3 indicates the median and interquantile range of the switching points.[13] A first observation is that the data sets with large instances, such as *Movielens_1m* and *Facebook*, have a wider range of switching points (large third quantile value) compared to data sets with small instances such as *Flickr* and *Ecommerce*, where the first quantile, the median and the third quantile are equal to 1. We also observe that, for linear models, there are no differences in the median switching point between the

[13] Median and interquantile range reported rather than the mean and standard deviation because the switching point only takes positive values and is right-skewed.

Table 2 Percentage explained (counterfactuals smaller than 30 features)

Dataset	Linear			Nonlinear		
	SEDC (%)	LIME-C (%)	SHAP-C (%)	SEDC (%)	LIME-C (%)	SHAP-C (%)
Flickr	**100**	99.33	**100**	**28.67**	**28.67**	**28.67**
Ecommerce	**100**	97.33	**100**	95.00	96.67	**99.67**
Airline	**100**	**100**	**100**	**100**	**100**	**100**
Twitter	**100**	**100**	**100**	**100**	**100**	**100**
Fraud	**100**	**100**	81.67	**100**	**100**	75
YahooMovies	**100**	**100**	**100**	98.67	**100**	**100**
TaFeng	**100**	**100**	**100**	93.33	**100**	**100**
KDD2015	**100**	**100**	**100**	99.67	**100**	99.67
20news	**100**	99.47	**100**	**100**	98.94	**100**
Movielens_100k	**100**	**100**	**100**	**100**	**100**	**100**
Facebook	**96.67**	95.33	95.00	70.33	**93.67**	90.00
Movielens_1m	**98.67**	**98.67**	**98.67**	89.67	**95.67**	**95.67**
LibimSeTi	**95.67**	91.00	89.33	77.33	**91.33**	89.67
Average	**99.31**	98.55	97.28	88.67	**92.69**	90.64
# Wins	**13**	8	10	6	**11**	9

For stochastic *LIME-C/SHAP-C*, these are average percentages over 5 runs. The best percentages are indicated in bold. The percentages are underlined if a method is significantly worse than the best method on a 1% significance level using a McNemar mid-p test (Fagerland et al. 2013)

Table 3 Median and interquantile range of switching point

Dataset	Linear				Nonlinear			
	SEDC	LIME-C	SHAP-C	Random	SEDC	LIME-C	SHAP-C	Random
Flickr	**1(1 − 1)**	**1(1 − 1)**	**1(1 − 1)**	**1(1 − 1)**	**1(1 − 1)**	**1(1 − 1)**	**1(1 − 1)**	**1(1 − 2)**
Ecommerce	**1(1 − 1)**	**1(1 − 1)**	**1(1 − 1)**	**1(1 − 2)**	**1(1 − 1)**	**1(1 − 1)**	**1(1 − 1)**	**1(1 − 1)**
Airline	**1(1 − 2)**	**1(1 − 2)**	**1(1 − 2)**	2(1 − 3)	**1(1 − 1)**	**1(1 − 1)**	**1(1 − 1)**	2(1 − 3)
Twitter	**2(1 − 3)**	**2(1 − 3)**	**2(1 − 3)**	3(2 − 5)	**1(1 − 1)**	**1(1 − 1)**	**1(1 − 1)**	3(2 − 5.5)
Fraud	**1(1 − 1)**	**1(1 − 1)**	**1(1 − 1)**	**1(1 − 1)**	**1(1 − 1)**	**1(1 − 1)**	**1(1 − 1)**	**1(1 − 2)**
YahooMovies	**2(1 − 4)**	**2(1 − 4)**	**2(1 − 4)**	4(2 − 7)	**1(1 − 3)**	2(1 − 3)	2(1 − 3)	4(2 − 12)
TaFeng	**2(1 − 4)**	**2(1 − 4)**	**2(1 − 4)**	5(3 − 11)	**2(1 − 8)**	**2(1 − 3)**	**2(1 − 4)**	6(3 − 17)
KDD2015	**3(1 − 7)**	**3(1 − 7)**	**3(1 − 7)**	8(3 − 17)	**2(1 − 3)**	**2(1 − 3.8)**	**2(1 − 4)**	4.5(2 − 9)
20news	**2(1 − 4)**	**2(1 − 4)**	**2(1 − 4)**	11(4 − 24)	**1(1 − 3)**	**1(1 − 3)**	**1(1 − 3)**	8(3 − 19)
Movielens_100k	**2(1 − 4)**	**2(1 − 4)**	**2(1 − 4)**	5.5(3 − 10)	**2(1 − 4)**	**2(1 − 4)**	**2(1 − 4)**	5(2 − 9.75)
Facebook	**3(2 − 8)**	4(2 − 8.6)	4(2 − 8)	8(4 − 20)	4.5(1 − 13.25)	**4(2 − 9.2)**	4.4(2 − 10.4)	9.5(4 − 20)
Movielens_1m	**3(2 − 7)**	**3(2 − 7)**	**3(2 − 7)**	9(4 − 19)	**3(1 − 6)**	**3(2 − 8)**	**3(2 − 8)**	7(3 − 14.5)
LibimSeTi	**3(2 − 7)**	**3(2 − 8.2)**	**3(2 − 7.4)**	30(13.75 − 55)	**2.5(1 − 5)**	3(2 − 8.2)	3(2 − 7.2)	22(9.75 − 43.25)
# Wins	**13**	12	12	3	**12**	11	10	3

For stochastic *LIME-C/SHAP-C*, this is the average median/range over 5 runs. The switching point is measured over the subset of instances where *all* methods have found a switching point. The best (median) switching points are indicated in bold. The values are underlined if a method is significantly worse than the best method on a 1% significance level using a McNemar mid-p test (Fagerland et al. 2013)

algorithms. For linear models, in general, the low switching points of *SEDC* are not a surprising result: it is optimal for linear models, i.e., it will always find the minimum-sized subset of features (Martens and Provost 2014). Comparing the results of the novel algorithms *LIME-C* and *SHAP-C*, which are approximation methods, against *SEDC*, for linear models they usually find the smallest-sized explanations as well. For the nonlinear models, however, no method dominates. *LIME-C* and *SHAP-C* perform worse than *SEDC* on the *YahooMovies* and *LibimSeTi* data sets. *SEDC* performs worse in terms of median switching points than *LIME-C* and *SHAP-C* on the *Facebook* data. The *Facebook* data present an interesting case. The mean switching points of *SEDC*, *LIME-C* and *SHAP-C* for *Facebook/nonlin* are respectively 8.34, 3.59 and 4.13, indicating that there are more outlier values for *SEDC*. The reason here is similar to the discussion of the iteration limit above, but there is an additional factor: we stop the search after the first explanation is found. This may be penalizing *SEDC* in terms of explanation length, but giving it an advantage in terms of computational efficiency. Finally, when comparing the methods with the random benchmark, we conclude that all approaches are significantly better at pinpointing the most important features, except for the *Ecommerce*, *Flickr* and *Fraud* data, where random performs as well because of the few active features per instance.

5 Results: computational efficiency

Table 4 summarizes the computation times. We observe that the median computation times of *SEDC* are very small, compared to *LIME-C* and *SHAP-C*: for all our data and models, the median computation time for *SEDC* is less than 1 second. The interquantile ranges and the mean computation times also inform us about the efficiency of *SEDC*. More specifically, for all data, there are many outlier values for computation times. This is because *SEDC*'s efficiency (mostly) depends on the number of features in the explanation. We observe that, for the data with very low switching points (e.g., *YahooMovies*), *SEDC* is very efficient over the entire set of test instances: there are not many extreme values. For instances that need more features to be removed before a predicted class change is obtained,[14] *SEDC* is slower (*Movielens_1m, Facebook*, *LibimSeTi*). This becomes an issue for classification problems where instances are "harder" to explain with counterfactuals, i.e., more features need to be removed to change the predicted class. For data with small instances (e.g., *Ecommerce*) or classification problems where data instances are "easier" to explain by a counterfactual, *SEDC* is always the most efficient method. Note that, despite the fact that *LibimSeTi* has, on average, a smaller switching point than *Facebook*, it still takes much longer to generate counterfactual explanations for the *LibimSeTi* data. This is because the number of active features is, on average, very large for *LibimSeTi*, which also plays an important role in determining the computation time.

Overall, *LIME–C* and *SHAP–C* have a stable efficiency and the computation time does not depend on the switching point. (In contrast to *SEDC*, for which the compu-

[14] These instances are "harder" to explain by counterfactuals as they, for example, have many active features that contribute to the model prediction (positive evidence).

Table 4 Median and interquantile range of computation time in seconds

Dataset	Linear			Nonlinear		
	SEDC	LIME-C	SHAP-C	SEDC	LIME-C	SHAP-C
Flickr	**0.00(0.00−0.00)**	0.19(0.19−0.23)	0.01(0.00−0.01)	**0.00(0.00−0.02)**	0.24(0.23−0.25)	0.01(0.01−0.08)
Ecommerce	**0.00(0.00−0.01)**	0.22(0.19−0.24)	**0.00(0.00−0.01)**	**0.00(0.00−0.02)**	0.27(0.26−0.28)	0.01(0.01−0.01)
Airline	**0.02(0.01−0.04)**	0.79(0.62−0.91)	0.08(0.02−0.59)	**0.02(0.02−0.03)**	1.18(0.96−1.33)	0.10(0.02−0.79)
Twitter	**0.03(0.01−0.05)**	1.21(1.09−1.32)	0.37(0.09−1.09)	**0.01(0.01−0.01)**	0.89(0.82−0.95)	0.13(0.03−0.43)
Fraud	**0.00(0.00−0.02)**	0.24(0.22−0.28)	0.02(0.01−0.17)	**0.02(0.02−0.02)**	0.65(0.60−0.72)	0.05(0.02−0.82)
YahooMovies	**0.07(0.02−0.19)**	0.27(0.26−0.31)	0.80(0.77−0.83)	**0.09(0.04−0.30)**	0.63(0.62−0.67)	1.11(1.08−1.16)
TaFeng	**0.05(0.02−0.19)**	0.53(0.43−0.63)	1.99(1.68−2.27)	**0.03(0.02−0.39)**	0.55(0.47−0.68)	1.45(1.26−1.63)
KDD2015	**0.09(0.02−0.74)**	0.36(0.32−0.43)	0.87(0.82−0.92)	**0.14(0.03−0.53)**	0.57(0.52−0.64)	1.07(1.02−1.13)
20news	**0.19(0.05−1.43)**	2.95(1.88−3.96)	3.36(2.49−4.16)	**0.10(0.03−0.76)**	1.94(1.34−2.69)	2.39(1.88−2.95)
Movielens_100k	**0.07(0.02−0.32)**	0.35(0.31−0.57)	0.87(0.83−1.04)	**0.14(0.07−0.70)**	0.42(0.34−0.66)	0.93(0.88−1.13)
Facebook	**0.11(0.02−1.19)**	0.35(0.28−0.51)	0.90(0.84−1.03)	**0.17(0.02−2.03)**	0.39(0.32−0.55)	0.95(0.88−1.07)
Movielens_1m	**0.34(0.06−2.92)**	0.56(0.35−0.99)	1.06(0.30−1.39)	**0.35(0.06−1.59)**	0.72(0.51−1.24)	2.49(2.21−3.33)
LibimSeTi	**0.37(0.13−3.12)**	0.70(0.59−0.97)	1.17(1.09−1.38)	**0.84(0.19−3.48)**	0.71(0.58−0.97)	1.18(1.09−1.39)
# Wins	13	0	1	13	0	0

For stochastic *LIME-C/SHAP-C*, this is the average median/range over 5 runs. The best (median) computation times are indicated in bold. The values are underlined if a method is significantly worse than the best method on a 1% significance level using a McNemar mid-p test (Fagerland et al. 2013)

tation time is sensitive to the number of features in the explanation.) The efficiency of *LIME-C* and *SHAP-C* depends mostly on the number of active features of an instance. The results indicate that *SHAP-C*'s efficiency seems more prone to the number of active features of the instance (median and interquantile range values are relatively larger starting from *YahooMovies*).

Lastly, the algorithms are generally slower for textual data than for behavioral data. We conjecture this is because of the time to evaluate the SVM scoring function f, which may be higher compared to the ℓ_2-LR and MLP scoring functions. As an illustration, take the *Facebook* data (behavioral) and *20news* data (textual). Even though the *Facebook* data has more active features per instance and more features in the model (1,22,924 compared to 41,356), the median time to compute a counterfactual for all three algorithms is higher for the *20news* data than for the *Facebook* data.

6 Conclusion and future work

From this study applying model-agnostic, instance-level explanation methods to the finding of counterfactual explanations for high-dimensional behavioral and textual data, we can draw several conclusions. First, the (straightforward) extensions *LIME-C* and *SHAP-C* as expected find reasonable, if not always optimal, counterfactual explanations. Furthermore, extending these algorithms to find counterfactual explanations addresses an open problem with the application of these methods to high-dimensional data, namely, which features should be reported in the explanation. The answer for *LIME-C* and *SHAP-C* is: those that allow the creation of an Evidence Counterfactual. *SHAP-C* does have problems with highly unbalanced data sets. Despite this, *SHAP-C* may still be preferable when the user is particularly interested in the theoretical interpretation of the importance weights (Lundberg and Lee 2017). *LIME-C* showed a stable effectiveness for all data and models, and even for very large data instances that require many features to be removed for a predicted class change, *LIME-C* computes counterfactuals relatively fast. Moreover, the results indicate that the efficiency of *LIME-C* is less sensitive to the number of active features compared to *SHAP-C*.

SEDC, which was designed to find counterfactual explanations, is generally fast and effective, but not always. In the main results, *SEDC* was clearly the fastest. It is provably optimal for linear models, and also empirically found smaller counterfactuals (on average) for the nonlinear models on two data sets. However, for certain instances on certain data sets, *SEDC*'s run time was comparably quite large. Furthermore, the search stopping criteria were met before *SEDC* found explanations in a non-negligible number of cases. As a best-first search algorithm, there is an effectiveness vs. efficiency tradeoff that we did not explore comprehensively in this paper.

This work indicates that there is a good deal of room for more research on this topic. For example, instead of *LIME-C* and *SHAP-C*, other hybrid algorithms could be created. For example, LIME (or SHAP) could be run first to fix a search order for a search algorithm like *SEDC*. In those cases where *LIME-C* (*SHAP-C*) produces a great explanation, this new hybrid would find it fast. But the algorithm could keep searching, and would be biased towards trying the best features found by LIME (SHAP) before the others, which likely would lead to finding even better explanations faster. Furthermore,

we see optimized search algorithms performing quite well for computationally hard problems (Schreiber et al. 2018); we conjecture that similar algorithms could be applied in the context of classification from big, sparse data to find optimal explanations fast.

Acknowledgements Funding was provided by Research Foundation – Flanders (Grant No. 11G4319N).

References

Arras L, Horn F, Montavon G, Müller K-R, Samek W (2017) "What is relevant in a text document?": an interpretable machine learning approach. PLoS One 12(8):1–23

Attenberg J, Weinberger K, Smola Q, Dasgupta A, Zinkevich M (2009) Collaborative email-spam filtering with the hashing-trick. In: Proceedings of the 6th conference on email and anti-spam, pp 1–5

Breiman L (2001) Random forests. Mach Learn 45:5–32

Brozovsky L, Petricek V (2007) Recommender system for online dating service. In: Proceedings of conference Znalosti, VSB, Ostrava, Czech Republic

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 World wide web conference, pp 1–10. https://doi.org/10.1145/1526709.1526806

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

Chhatwal R, Gronvall P, Huber N, Keeling R, Zhang J, Zhao H (2018) Explainable text classification in legal document review: a case study of explainable predictive coding. In: 2018 IEEE international conference on big data, pp 1905–1911. https://doi.org/10.1109/BigData.2018.8622073

Craven MW (1996) Extracting comprehensible models from trained neural networks. The University of Wisconsin, Madison

De Cnudde S, Martens D, Evgeniou T, Provost F (2019a) A benchmarking study of classification techniques for behavioral data. Int J Data Sci Anal 9(17):1–43

De Cnudde S, Moeyersoms J, Stankova M, Tobback E, Javaly V, Martens D (2019b) What does your Facebook profile reveal about your creditworthiness? Using alternative data for microfinance. J Oper Res Soc 70(3):353–363

De Cnudde S, Ramon Y, Martens D, Provost F (2019c) Deep learning for big. Sparse Behav Data Big Data 7(4):286–307

Doshi-Velez F, Kim B (2017) Towards a rigorous science of interpretable machine learning, pp 1–13. ArXiv preprint arXiv:1702.08608

Fagerland M, Lydersen S, Laake P (2013) McNemar test for binary matched-pairs data: mid-p and asymptotic better than exact conditional. BMC Med Res Meth 13:1–8

Fernandez, C, Provost, F, Han, X (2019) Explaining data-driven decisions made by AI systems: the counterfactual approach, pp 1–33. arXiv:2001.07417

Freitas A (2014) Comprehensible classification models: a position paper. SIGKDD Explor Newsl 15(1):1–10

Goodman B, Flaxman S (2016) EU regulations on algorithmic decision-making and a "right to explanatio". AI Maga 38:50–57

Gregor S, Benbasat I (1999) Explanations from intelligent systems: theoretical foundations and implications for practice. MIS Q 23(4):497–530

Harper FM, Konstan JA (2015) The movielens datasets: history and context. ACM Trans Interact Intell Syst. https://doi.org/10.1145/2827872

Hsu C-N, Chung H-H, Huang H-S (2004) Mining skewed and sparse transaction data for personalized shopping recommendation. Mach Learn 57(1):35–59

Joachims T (1998) Text categorization with suport vector machines: learning with many relevant features. In: Proceedings of ECML-98, 10th European conference on machine learning, vol 1398, pp 137–142

Junqué de Fortuny E, Martens D, Provost F (2013) Predictive modeling with big data: is bigger really better? Big Data 1(4):215–226

Kosinski M, Stillwell D, Graepel T (2013) Private traits and attributes are predictable from digital records of human behavior. Natl Acad Sci 110(15):5802–5805
Lang K (1995) Newsweeder: learning to filter netnews. In: Proceedings of the twelfth international conference on machine learning, pp 331–339
Lipton ZC (2018) The mythos of interpretability. ACM Queue 16(3):1–28
Lundberg SM, Lee S-I (2017) A unified approach to interpreting model predictions. In: Advances in neural information processing systems 30 (NIPS 2017). Curran Associates Inc., pp 4765–4774
Martens D, Baesens B, van Gestel T, Vanthienen J (2007) Comprehensible credit scoring models using rule extraction from support vector machines. EJOR 183(3):1466–1476
Martens D, Provost F (2014) Explaining data-driven document classifications. MIS Q 38(1):73–99
Martens D, Provost F, Clark J, Junque de Fortuny E (2016) Mining massive fine-grained behavior data to improve predictive analytics. MIS Q 40:869–888
Matz SC, Kosinski M, Nave G, Stillwell DJ (2017) Psychological targeting as an effective approach to digital mass persuasion. PNAS 114(48):12714–12719
Moeyersoms, J, d'Alessandro, B, Provost, F, Martens, D (2016) Explaining classification models built on high-dimensional sparse data. In: Workshop on human interpretability, machine learning: WHI 2016, 23 June 2016, New York, USA/Kim, Been [edit.], pp 36–40
Nguyen D (2018) Comparing automatic and human evaluation of local explanations for text classification. In: 16th conference of the NACA for CL, pp 1–10
Provost F (2014) Understanding decisions driven by big data: from analytics management to privacy-friendly cloaking devices. Keynote Lecture, Strata Europe. https://learning.oreilly.com/library/view/strata-hadoop/9781491917381/video203329.html. Accessed 17 June 2019
Provost F, Fawcett T (2013) Data science for business: what you need to know about data mining and data-analytic thinking, 1st edn. O'Reilly Media Inc., USA
Provost F, Martens D, Murray A (2015) Finding similar mobile consumers with a privacy-friendly geosocial design. Inf Sys Res 26(2):243–265
Ras G, van Gerven M, Haselager P (2018) Explanation methods in deep learning: users, values, concerns and challenges. In: Explainable and interpretable models in computer vision and machine learning. Springer, pp 19–36
Ribeiro MT, Singh S, Guestrin, C (2016) Why should I trust you? Explaining the predictions of any classifier. In: Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining, pp 1135–1144
Samek W, Binder A, Montavon G, Bach S, Müller K-R (2015) Evaluating the visualization of what a deep neural network has learned, pp 1–13. Arxiv preprint arXiv:1509.06321
Schreiber EL, Korf RE, Moffitt MD (2018) Optimal multi-way number partitioning. J ACM 65(4):1–61
Shmueli G (2017) Analyzing behavioral big data: methodological, practical, ethical, and moral issues. Qual Eng 29(1):57–74
Sokol K, Flach PA (2019) Counterfactual explanations of machine learning predictions: opportunities and challenges for AI safety. In: CEUR workshop proceedings, pp 1–5
Strumbelj E, Kononenko I (2010) An efficient explanation of individual classifications using game theory. J Mach Learn Res 11:1–18
Wachter S, Mittelstadt BD, Russell C (2018) Counterfactual explanations without opening the black box: automated decisions and the GDPR. Harv J Law Technol 31(2):1–47