Journal of Machine Learning Research (Submitted) Submitted 11/17; Published N/A

Benchmarking Relief-Based Feature Selection Methods

Ryan J. Urbanowicz RYANURB@UPENN.EDU
Randal S. Olson OLSONRAN@UPENN.EDU
Peter Schmitt PSCHMITT@UPENN.EDU
Institute for Biomedical Informatics, University of Pennsylvania
Philadelphia, PA 19104, USA

Melissa Meeker MEMEEKER@URSINUS.EDU
Ursinus College
Collegeville, PA, 19426, USA

Jason H. Moore JHMOORE@UPENN.EDU
Institute for Biomedical Informatics, University of Pennsylvania
Philadelphia, PA 19104, USA

Editor: NA

Abstract

Modern data mining requires feature selection methods that can (1) be applied to large scale feature spaces, (2) function in noisy problems, (3) detect complex patterns of association (e.g. interactions), (4) be flexibly adapted to various problem domains and data types, and (5) are computationally tractable. To that end, this work examines a set of filter-style feature selection algorithms inspired by the 'Relief' algorithm, i.e. Relief-Based algorithms (RBAs). We implement and expand these RBAs in an open source framework called ReBATE (Relief-Based Algorithm Training Environment). We apply a comprehensive simulation study comparing existing RBAs, a proposed RBA called MultiSURF, and other established feature selection methods, over a variety of problem types. The results of this study (1) support the assertion that RBAs are particularly flexible, efficient, and powerful feature selection methods that differentiate relevant features having univariate, multivariate, epistatic, or heterogeneous associations, (2) confirm the efficacy of expansions for classification vs. regression, discrete vs. continuous features, missing data, multiple classes, or class imbalance, (3) identify previously unknown limitations of specific RBAs, and (4) suggest that while MultiSURF* performs best for explicitly identifying pure 2-way interactions, MultiSURF yields the most reliable feature selection performance across a wide range of problem types.

Keywords: Feature Selection, Feature Interaction, Relief, ReliefF, Filter, Epistasis, Heterogeneity, Classification, Regression, Missing Data, Imbalanced Data, MultiSURF

1. Introduction

Feature selection is often an essential task in data mining and modeling (i.e. induction), particularly in problems where the data is noisy, complex, and/or includes a very large feature space. Many feature selection strategies have been proposed over the years, generally falling into one of three categories: (1) filter methods, (2) wrapper methods, or (3) embedded methods (Saeys et al., 2007; Bolón-Canedo et al., 2013; Chandrashekar and Sahin, 2014; Tang et al., 2014; Jović et al., 2015; Mlambo et al., 2016; Urbanowicz et al., Submitted).

Feature selection methods have further been characterized based on whether selection relies on scores assigned to individual features or instead to a candidate subset of features (Yu and Liu, 2004; Bolón-Canedo et al., 2013).

The present study focuses on the family of *Relief-based* feature selection methods referred to here as Relief-Based Algorithms (RBAs) that can be characterized as *individual evaluation filter methods*. In work that pairs with this research paper, Urbanowicz et al. (Submitted) introduced and surveyed RBA methods, detailing why they are advantageous in contrast with other feature selection methods. To summarize here, RBAs retain the general benefits of filter-methods, i.e. they are relatively fast (with an asymptotic time complexity of $\mathcal{O}(instances^2 \cdot features)$), and the selected features are induction algorithm independent. More importantly, RBAs are the only filter-methods known that have the ability to capture feature dependencies in predicting endpoint/outcome, i.e. feature interactions (Bolón-Canedo et al., 2013). This unique ability has been attributed to Relief's use of 'nearest neighbor instances' in calculating feature weights (Kononenko et al., 1997; Kononenko and Šikonja, 2008).

The Relief algorithm concept has also been shown to be extendable to many different data type concerns including classification vs. regression, discrete vs. continuous features, missing data, multiple classes, and class imbalance. Unfortunately many RBAs and associated implementations have yet to be extended to data types beyond clean binary classification problems with discrete features. Furthermore, RBAs are advantageous because they output individual feature weights. These weights can be used both to flexibly set different criteria for defining a feature subset as well as be applied to feature weighting schemes, where for example feature weights probabilistically guide machine learning modeling downstream (Urbanowicz et al., 2012a). One other notable aspect of RBAs is that they do not eliminate feature redundancies (i.e. feature correlations). This could be viewed either as an advantage or disadvantage based on the problem at hand. For problems that require the removal of feature weights, many effective methods have been developed to remove feature redundancy as reviewed by Urbanowicz et al. (Submitted). Since this 'drawback' of not handling feature redundancy is undisputed but there are independent effective methods available to deal with redundancy when needed, we do not consider the topic further in this research.

1.1 Core Relief Algorithms

The present study specifically focuses on what we will refer to as 'core' RBAs, or algorithm variants designed to be run for a single iteration through the training data. In contrast, a handful RBA extensions have also been proposed to improve performance in very large feature spaces by applying a core Relief algorithm iteratively, e.g. I-RELIEFF (Sun and Li, 2006), TuRF (Moore and White, 2007), evaporative cooling ReliefF (McKinney et al., 2007), and iVLSReliefF (Eppstein and Haake, 2008), or applying it to a multitude of random feature subsets rather than the entire feature space to improve efficiency, i.e. VLSReliefF (Eppstein and Haake, 2008). The iterative extensions are much more computationally expensive and the success of the VLSReliefF and iVLSReliefF methods rely on additional run parameters that may require problem domain knowledge to be set optimally.

Stepping back, we propose that there are two larger Relief algorithm research questions. First, what is the most effective core Relief algorithm? This question is asked given the expectation that the performance of any core method alone will deteriorate as the feature space becomes very large (Moore and White, 2007; Eppstein and Haake, 2008). Second, what is the most effective iterative Relief expansion for improving performance in very large feature spaces? This work focuses exclusively on the first question. We expect that by first identifying and adopting the most reliable core algorithm this will maximize the performance of any iterative expansion, since they each rely on core algorithm functionality. However, it will be important to revisit strategies for very large feature spaces in future work. For a complete review of RBA research with respect to core, iterative, efficiency, and data type handeling methodologies we refer readers to Urbanowicz et al. (Submitted).

1.2 Bioinformatics

The focus of this work on RBAs is related to our interest in the application domain of bioinformatics. Most of the RBAs analyzed in this paper have been developed and applied within the purview of genetic association problems. Such problems are commonly characterized as (1) noisy, (2) having varied or sometimes mixed data types (e.g. discrete and continuous features), and (3) including very large feature spaces (Moore and White, 2007; Greene et al., 2009, 2010; Granizo-Mackenzie and Moore, 2013). Also, the detection of complex patterns of association between features and the endpoint is of particular interest in bioinformatics. In particular, this study considers two important complicating phenomena: epistasis, i.e. feature interactions (or gene-gene interactions in the context of bioinformatics) (Cordell, 2002; Moore and Williams, 2005) and heterogeneous associations with endpoint, i.e. genetic heterogeneity or phenocopy (in the context of bioinformatics problems) (Urbanowicz and Moore, 2010; Urbanowicz et al., 2013). Genetic heterogeneity occurs when the same or similar phenotypic endpoint can be the result of distinct independent relevant features (or set of relevant features) in different subsets of the sample population. Heterogeneous associations are recognized to confound most modeling techniques (Ritchie et al., 2003), with the exception of induction algorithms that can distinguish problem 'niches' such as rule-based machine learning approaches (Urbanowicz and Moore, 2015). While not all problem domains may be as complex as those in bioinformatics, we expect the findings of this work to be applicable to any data mining problems calling for feature selection.

1.3 Study Overview

In the present study, we (1) implement a variety of core RBAs as part of an accessible, open source Python software package called ReBATE, (2) introduce a new core RBA called MultiSURF, (3) extend all implemented algorithms to be able to accommodate varied data type issues, i.e. binary classification, multi-class classification, or regression, discrete, continuous or mixed feature types, missing data and class imbalance, (4) design, generate, and apply a comprehensive simulation study of 2280 datasets to validate the efficacy of our data type extensions, and to compare the efficacy of 13 feature selection methods, i.e. eight RBAs, three traditional 'filter' feature selection methods, and two 'wrapper' feature selection methods, (5) investigate the reasons for identified performance discrepancies among feature selection methods, (6) identify the best performing and most reliable feature selection algorithms evaluated, and (7) organize what we have learned from this investigation to guide future RBA application and development.

The remainder of this paper is organized as follows: Section 2 describes the methods, Section 3 presents the results, Section 4 offers discussion regarding why specific strengths and weaknesses were observed for respective methods. Lastly, Section 5 offers conclusions and directions for future work.

2. Methods

In this section, we begin by describing the ReBATE software including: (1) the previously defined RBAs that it implements, (2) our proposed MultiSURF algorithm, and (3) the adopted data type extension strategies. Next, we describe the evaluations including: (1) other filter and wrapper based feature selection methods compared, (2) design of the simulation study, and (3) a discussion of the evaluation metrics.

2.1 ReBATE

To facilitate the accessibility of various RBAs and promote their ongoing development and application, we have implemented Relief-Based Algorithm Training Environment (ReBATE). With ReBATE, we seek to balance data type flexibility, run time efficiency, and ease of development in a Python package framework. At the time of writing, ReBATE has been implemented with five core RBAs: i.e. ReliefF (Kononenko, 1994), SURF (Greene et al., 2009), SURF* (Greene et al., 2010), MultiSURF* (Granizo-Mackenzie and Moore, 2013), and our proposed MultiSURF algorithm. These core RBAs were chosen for this study because (1) they were explicitly developed for and previously evaluated on noisy and epistatic problems, (2) accessible Python implementations were available for each, and (3) they represented a competitive diversity of core RBAs. Additionally, we included the iterative TuRF algorithm (Moore and White, 2007) in ReBATE. However, as an iterative approach, TuRF is not evaluated further in this study.

These five core RBAs and TuRF were originally implemented in the open source Multifactor Dimensionality Reduction (MDR) (Ritchie et al., 2001) software package[1]. These Java implementations are computationally efficient, but can only handle complete data (i.e. no missing values) with discrete features and a binary endpoint. Python 2.7 versions of these algorithms were more recently implemented and made available within the open source Extended Supervised Tracking and Classifying System (ExSTraCS)[2] (Urbanowicz et al., 2014; Urbanowicz and Moore, 2015). These were less computationally efficient, but extended each algorithm to handle different data types including continuous features, multiclass endpoints, regression, and missing data. The ReBATE implementations of these RBAs have restructured these ExSTraCS implementations for efficiency and modularity, while preserving the ability to handle different data types. Notably, while these data type expansions had been previously proposed and implemented within the ExSTraCS framework, they had yet to be experimentally validated.

1. `http://sourceforge.net/projects/mdr`
2. `https://github.com/ryanurbs/ExSTraCS_2.0`

We have implemented ReBATE both as a stand alone software package[3] as well as a scikit-learn (Pedregosa et al., 2011) compatible format[4]. It was our goal to make data type flexible implementations of these algorithms available for real-world application, as well as encourage ongoing methodological development or expansion of the ReBATE algorithm repertoire.

ReBATE includes a data pre-processing step that automatically identifies essential data type characteristics. Specifically this includes (1) distinguishing discrete from numerical features, (2) distinguishing a discrete from numerical endpoint, (3) identifying the min-max value range for numerical features or endpoint, (4) for discrete classes, determining the number of unique classes (i.e. binary or multi-class) as well as the number of instances having each class label, and (5) identifying the presence of missing data, with a standard identifier, e.g. 'N/A'. This pre-processing automates the adaptation of each RBA to the relevant data types.

In the following subsections, we provide methodological summaries of these five core algorithms. They were described in contrast to the larger family of RBAs by Urbanowicz et al. (Submitted). Next, in the remaining subsections, we detail how these core RBAs have been universally extended to handle specific data-type challenges. Our focus in this study is to demonstrate that our adopted strategies are functional, but do not seek to claim that they are optimal. Currently there is minimal empirical evidence in the literature to support the conclusion that that any particular RBA data type handeling strategy performs optimally.

2.1.1 ReliefF

The original Relief algorithm (Kira and Rendell, 1992b,a) was quickly improved upon to yield the most widely known RBA to date, ReliefF (Kononenko, 1994). For clarity, we will begin with a complete algorithmic description of ReliefF. Other complete descriptions of Relief and ReliefF can be found in (Kononenko et al., 1996, 1997; Robnik-Šikonja and Kononenko, 2003). For simplicity ReliefF is described without the data type extensions that will introduced later. Algorithm 1 details ReliefF as it has been efficiently implemented in ReBATE. Specifically all RBAs in ReBATE have been structured into distinct stages, i.e. (Stage 1) pre-process the data, (Stage 2) pre-compute the pairwise instance distance array, and (Stage 3) neighbor determination and calculate feature weights.

The reasoning behind pre-computing the distance between all pairs of instances in Stage 2 is based on an assumption that was introduced in ReliefF and has largely persisted in most RBA implementations. Specifically we assume that all training instances will be utilized in scoring. This assumption is in contrast to the original Relief algorithm where the user could specify a subset of m random instances that would be used to update feature weights (Kira and Rendell, 1992b). However since it was found that the quality of weight estimates becomes more reliable as the parameter m approaches the total number of instances n, proposed the simplifying assumption that $m = n$ (Kononenko, 1994). In other words, every instance gets to be the 'target' for weight updates one time, i.e instances are selected without replacement. As such, all pairwise distances will be required to run each RBA in ReBATE.

3. `https://github.com/EpistasisLab/scikit-rebate`
4. `https://github.com/EpistasisLab/ReBATE`

Algorithm 1 Pseudo-code for ReliefF algorithm as implemented in ReBATE

```
Require: for each training instance a vector of feature values and the class value
 1: $n \leftarrow$ number of training instances
 2: $a \leftarrow$ number of attributes (i.e. features)
 3: Parameter: $k \leftarrow$ number of nearest hits '$H$' and misses '$M$'
 4:
 5: # STAGE 1
 6: pre-process dataset {$\approx a \cdot n$ time complexity}
 7: # STAGE 2
 8: pre-compute distance array {$\approx 0.5 \cdot a \cdot n^2$ time complexity}
 9: # STAGE 3
10: initialize all feature weights $W[A] := 0.0$
11: for $i$:=1 to $n$ do
12:    # IDENTIFY NEIGHBORS
13:    for j:=1 to $n$ do
14:       identify $k$ nearest hits and $k$ nearest misses (using distance array)
15:    end for
16:    # FEATURE WEIGHT UPDATE
17:    for all hits and misses do
18:       for A:= to $a$ do
19:          $W[A] := W[A] - \mathit{diff}(A, R_i, H)/(n \cdot k) + \mathit{diff}(A, R_i, M)/(n \cdot k)$
20:       end for
21:    end for
22: end for
23: return the vector $W$ of feature scores that estimate the quality of features
```

All RBAs calculate a proxy statistic for each feature that can be used to estimate feature 'relevance' to the target concept (i.e. predicting endpoint value). These feature statistics are referred to as feature weights ($W[A]$ = weight of feature 'A'), or more casually as feature 'scores' that can range from -1 (worst) to $+1$ (best).

As depicted in Algorithm 1, once ReliefF has pre-processed the data and pre-computed the distance between all instance pairs, Stage 3 cycles through n randomly ordered training instances (R_i), selected without replacement. Each cycle, R_i is the target instance and the feature score vector W is updated based on feature value differences observed between the target and neighboring instances. ReliefF relies on a 'number of neighbors' user parameter k that specifies the use of k nearest hits and k nearest misses in the scoring update for each target instance. Next, it selects k nearest neighbors with the same class called the *nearest hits* (H) and the other with the opposite class, called the *nearest misses* (M). The last step of the cycle updates the weight of a feature A in W if the feature value differs between the target instance R_i and any of the nearest hits H or nearest misses M. Features that have a different value between R_i and an M support the inference that they are informative of outcome, so the quality estimation $W[A]$ is increased. In contrast, features with different between R_i and H suggest evidence to the contrary, so the quality estimation $W[A]$ is decreased. The *diff* function in Algorithm 1 calculates the difference in value of feature A

between two instances I_1 and I_2 (where $I_1 = R_i$ and $I_2 =$ either H, or M when performing weight updates) (Robnik-Šikonja and Kononenko, 2001). For discrete (e.g. categorical or nominal) features, *diff* is defined as:

$$diff(A, I_1, I_2) = \begin{cases} 0 & \text{if } value(A, I_1) = value(A, I_2) \\ 1 & \text{if } otherwise \end{cases} \tag{1}$$

and for continuous (e.g. ordinal or numerical) features, *diff* is defined as:

$$diff(A, I_1, I_2) = \frac{|value(A, I_1) - value(A, I_2)|}{max(A) - min(A)} \tag{2}$$

This function ensures that weight updates fall between 0 and 1 for both discrete and continuous features. ReBATE adopts this strategy introduced by Relief (Kira and Rendell, 1992b) to extend all ReBATE algorithms for continuous features. Additionally, in updating $W[A]$, (see line 19 of Algorithm 1) dividing the output of *diff* by n and k guarantees that all final weights will be normalized within the interval $[-1, 1]$ adjusting for any level of class imbalance. This *diff* function is further applied in pre-computing the distance array, calculating Manhattan distances between instance pairs. For efficiency, ReBATE pre-normalizes any continuous variable (i.e. features or endpoint) so that it falls within a 0 to 1 value range.

Consider that while the above *diff* function performs well when features are either uniformly discrete or continuous, it has been noted that given a dataset with a mix of discrete and continuous features, this *diff* function can underestimate the quality of the continuous features (Kononenko and Šikonja, 2008). One proposed solution to this problem is a *ramp function* that naively assigns a full *diff* of 0 or 1 if continuous feature values are some user defined minimum or maximum value apart from one another (Hong, 1997; Robnik-Šikonja and Kononenko, 2003; Kononenko and Šikonja, 2008). However this naive approach adds two additional user-defined parameters requiring problem-specific optimization.

Figure 1 illustrates the major algorithmic differences between the original Relief algorithm (Kira and Rendell, 1992b), ReliefF, and the four other core RBAs implemented in ReBATE. Specifically, this figure focuses on respective strategies for neighbor selection. Note that while a k of 10 has been widely adopted as the default setting, a k of 3 was chosen for this conceptual illustration.

2.1.2 SURF

The SURF algorithm (Greene et al., 2009) inherits the majority of the ReliefF algorithm. In contrast with ReliefF, SURF eliminates the user parameter k, instead adopting a distance threshold T to determine which instances will be considered neighbors (see Figure 1). The radius defining the T hyper-circle in a-dimensional space around a given target instance is defined by the average distance between all instance pairs in the training data. This radius is therefore of a uniform size for each target instance.

2.1.3 SURF*

The SURF* algorithm (Greene et al., 2010) inherits the majority of the SURF algorithm. In contrast with SURF, SURF* introduced the concept of instances that were near vs. far

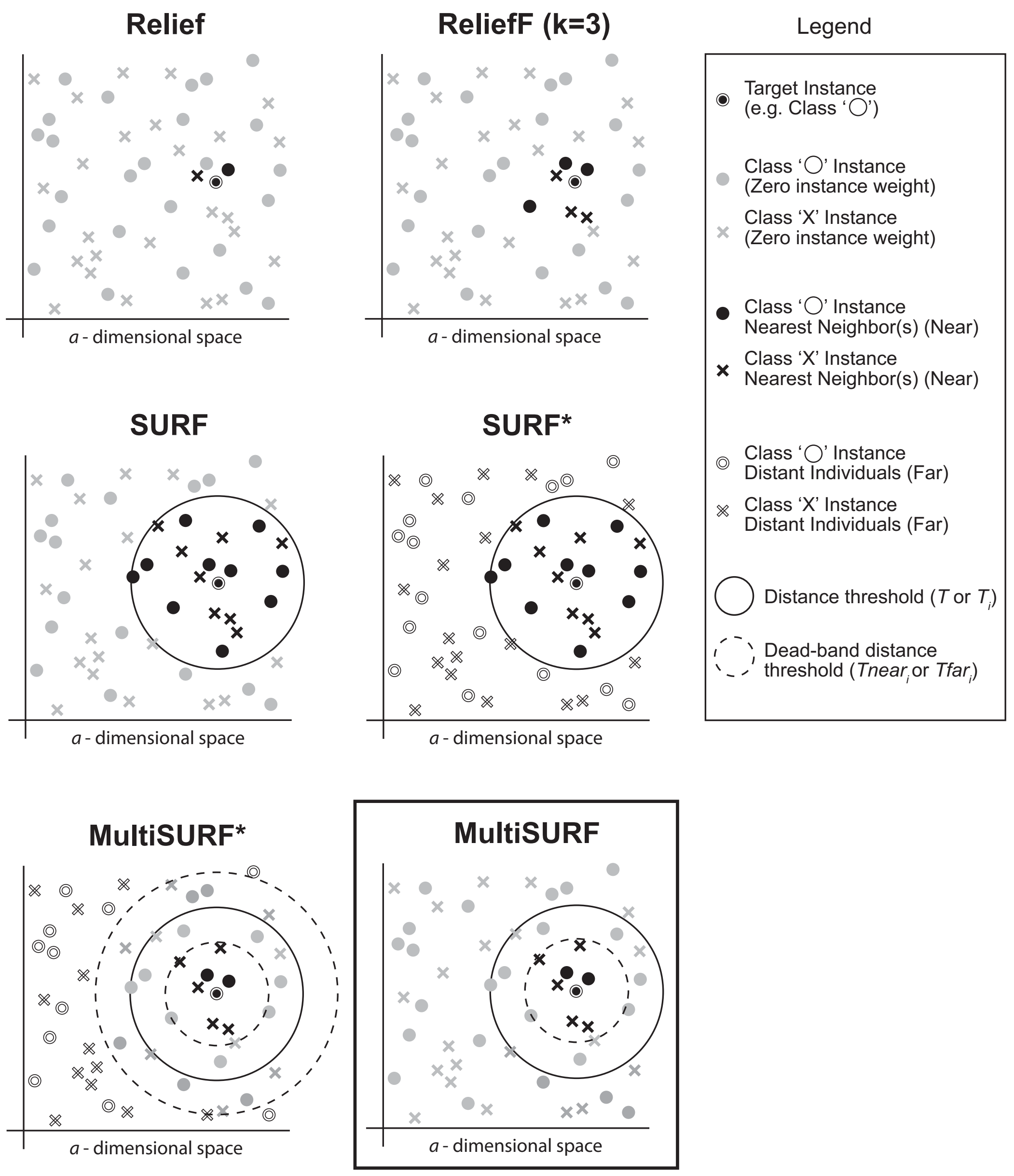
Relief
ReliefF (k=3)
Legend
SURF
SURF*
MultiSURF*
MultiSURF
a - dimensional space
Target Instance (e.g. Class '○')
Class '○' Instance (Zero instance weight)
Class 'X' Instance (Zero instance weight)
Class '○' Instance Nearest Neighbor(s) (Near)
Class 'X' Instance Nearest Neighbor(s) (Near)
Class '○' Instance Distant Individuals (Far)
Class 'X' Instance Distant Individuals (Far)
Distance threshold (T or T_i)
Dead-band distance threshold ($Tnear_i$ or $Tfar_i$)

Figure 1: Illustration of the neighbor selection differences between Relief, ReliefF, SURF, SURF*, MultiSURF*, and MultiSURF. Differences include the number of nearest neighbors or the method for selecting 'near' or 'far' instances for feature scoring. Note that for ReliefF, a k of 3 is chosen but a k of 10 is most common. These illustrations are conceptual and are not drawn to scale.

from the target instance (see Figure 1). Applying the same T from SURF, any instance within the threshold was considered near, and those outside were far. SURF* proceeds to weight 'far' instance differences in an opposite manner than 'near' instances. Specifically, feature value differences in hits differently receive a $+1$ while feature value differences in misses differently receive a -1. Notably, SURF* is the only RBA we implement that uses all instances in the training dataset to update feature weights each cycle. Note that the '*' in naming signifies the use of 'far' scoring in both SURF* and MultiSURF*.

2.1.4 MultiSURF*

The MultiSURF* algorithm (Granizo-Mackenzie and Moore, 2013) inherits the majority of the SURF* algorithm. In contrast with SURF*, MultiSURF* defines a threshold T_i as the mean pairwise distance between the target instance and all others, as opposed to the mean of all instance pairs in the data. This adapts the definition of near/far to a given part of the feature space. MultiSURF* also introduces a dead-band zone extending on either side of T_i, i.e. $Tnear_i$ or $Tfar_i$ (see Figure 1) to exclude instances near T_i from contributing to scoring, i.e. those that are ambiguously near or far. Accordingly, the width of the dead-band zone is the standard deviation σ of pairwise distances between the target instance and all others. Thus, for each target instance, the boundary circles, as illustrated for MultiSURF* in Figure 1, may have different radii. Lastly, the 'far' scoring logic was inverted in to reduce computations. Specifically in SURF*, *differences* in feature values in hits yielded a reduction in feature score, and an increase in misses. Since differences are expected to be more frequent in far individuals, MultiSURF* updates far instance feature weights with the *same* feature values, i.e. hits receive a $+1$, and misses receive a -1.

2.1.5 MultiSURF

The new RBA variant proposed in this study is called MultiSURF. As the minor name change suggests, MultiSURF is closely related to MultiSURF* (Granizo-Mackenzie and Moore, 2013). MultiSURF preserves all aspects of MultiSURF* but eliminates the 'far' scoring introduced in SURF* (see Figure 1). The dead-band boundary $Tnear_i$ in MultiSURF is equal to $T_i - \sigma_i/2$. Pseudo-code for MultiSURF is given by Algorithm 2, again excluding the data type expansions for simplicity. As before, the *diff* function of MultiSURF is given by Equations 1 and 2. Also note that because neighbors are defined by a threshold in MultiSURF, there can be a variable, or imbalanced number of hits and misses for each target instance. Lines 15 to 25 of Algorithm 2 identify nearest hits and misses and track counts of each (h and m, respectively). Line 29 normalizes weight updates based on n, h, and m. This accounts for imbalanced hit and miss counts for a given target instance. By using either an equal number of hits or misses (e.g. as in ReliefF), or using this type of h and m normalization, RBAs inherently adjusts for class imbalance in binary or multi-class problems (Robnik-Šikonja, 2003). As it is with all other core RBAs, the asymptotic time complexity of MultiSURF is $\mathcal{O}(n^2 \cdot a)$. The complete time complexity of MultiSURF is $c_0 a + c_1 0.5 n^2 a + c_2 n \log n + c_3 0.31 n^2 a + c_y n^2$ which is slightly faster than for MultiSURF*. Time complexity comparisons with of other RBAs are detailed in (Urbanowicz et al., Submitted).

2.1.6 Multi-class Endpoint

In the remaining subsections, we describe how RBAs in ReBATE were extended to handle respective data-type issues. To address multi-class endpoint problems, ReBATE methods

Algorithm 2 Pseudo-code for the proposed MultiSURF algorithm in ReBATE

```
Require: for each training instance a vector of feature values and the class value
 1: $n \leftarrow$ number of training instances
 2: $a \leftarrow$ number of attributes (i.e. features)
 3:
 4: # STAGE 1
 5: pre-process dataset $\{\approx a \cdot n$ time complexity$\}$
 6: # STAGE 2
 7: pre-compute distance array $\{\approx 0.5 \cdot a \cdot n^2$ time complexity$\}$
 8: for $i$:=1 to $n$ do
 9:   set $T_i$ to mean distances between instance $i$ and all others
10:   set $\sigma_i$ to standard deviation of those distances
11: end for
12: # STAGE 3
13: initialize all feature weights $W[A] := 0.0$
14: for $i$:=1 to $n$ do
15:   # IDENTIFY NEIGHBORS
16:   initialize hit and miss counters $h := 0.0$ and $m := 0.0$
17:   for j:=1 to $n$ do
18:     if distance between $i$ and $j$ is $< T_i$ - $\sigma_i/2$ (using distance array) then
19:       if $j$ is a hit then
20:         $h+ = 1$ {and identify instance as hit}
21:       else if $j$ is a miss then
22:         $m+ = 1$ {and identify instance as miss}
23:       end if
24:     end if
25:   end for
26:   # FEATURE WEIGHT UPDATE
27:   for all hits and misses do
28:     for A:= to $a$ do
29:       $W[A] := W[A] - diff(A, R_i, H)/(n \cdot h) + diff(A, R_i, M)/(n \cdot m)$
30:     end for
31:   end for
32: end for
33: return the vector $W$ of feature scores that estimate the quality of features
```

adopt the strategy proposed in ReliefF (Kononenko, 1994) that finds k nearest misses from every 'other' class, and averages the weight update based on the prior probability of each class. This expansion requires the calculation of prior class probabilities P(C) from the training data and, for a given target instance, identifying misses of every *other* class. In particular, ReliefF identifies k misses of every other class, while all other RBA implementations maintain counts of each type of near or far miss (to be used in normalizing the score update). Equation 3 defines how the weight update equation in ReliefF is adapted to multi-class endpoints. Equation 4 does the same for SURF, MultiSURF and for 'near' neighbors in SURF* and MultiSURF*. For 'far' instances in SURF* and MultiSURF*, the update would look like Equation 4, however the hit term would be added and the summed miss term would be subtracted.

$$\begin{aligned} W[A] := W[A] - \mathit{diff}(A, R_i, H)/(n * k) + \\ \sum_{C \neq class(target)} \Big[\frac{P(C)}{1 - P(class(target))} \mathit{diff}(A, R_i, M(C))\Big]/(n * k) \end{aligned} \tag{3}$$

$$\begin{aligned} W[A] := W[A] - \mathit{diff}(A, R_i, H)/(n * h) + \\ \sum_{C \neq class(target)} \Big[\frac{P(C)}{1 - P(class(target))} \mathit{diff}(A, R_i, M(C))\Big]/(n * m) \end{aligned} \tag{4}$$

2.1.7 Regression

To address regression, we propose an alternative, simpler approach than the current standard introduced in the Regressional ReliefF (RReliefF) algorithm (Kononenko et al., 1996; Robnik-Šikonja and Kononenko, 1997). The fundamental challenge of adapting Relief algorithms to continuous endpoints, is that we lose a clear definition for hit or miss, i.e. having the same or different class. RReliefF proposed a kind of "probability" that two instances belong to two "different" classes. This "probability" is modeled with the distance between feature and endpoint values of two learning instances as detailed by Robnik-Šikonja and Kononenko (1997). This includes an exponential weighting of instance contributions to W[A] based on distance between instances. Since current ReBATE methods do not apply distance based instance weights, and RReliefF requires an additional step computing prior and conditional probabilities, we propose a simpler regression scheme for our ReBATE methods.

Specifically, ReBATE calculates the standard deviation of the continuous endpoint (σ_E) and applies this as a simple threshold for determining whether two instances will be considered a "hit" or a "miss". This serves to contextually discretize the continuous endpoint into 'same class' or 'different class' from the perspective of the target instance. This proposed adaptation of RBAs to regression problems only requires pre-computing σ_E during Stage 1 of the algorithm, and changing the definition of a hit from $C_i = C_j$ to $|C_i - C_j| < \sigma_E$, and the definition of a miss from $C_i \neq C_j$ to $|C_i - C_j| \geq \sigma_E$.

2.1.8 Missing Data

Missing feature values must be dealt with by RBAs at two points in the algorithm: (1) Calculation of distances between instance pairs and (2) updating the feature weights. Previously, a missing data strategy proposed in ReliefF (or more precisely in ReliefD) had been identified as 'best' with minimal empirical investigation (Kononenko, 1994). It was also designed explicitly for problems with discrete endpoints. Specifically, depending on whether one or both instances have a missing value for the given feature, the *diff* function returns the probability that the feature states are different given the class of each instance. This approach is implicitly a form of interpolation, making an 'educated' guess at what the missing value might be. Under the right circumstances, this can indeed improve performance, but if the guess is wrong, it could just as easily harm performance. Further, this approach is more computationally and conceptually challenging to extend to continuous endpoint data.

In ReBATE we propose what we call an 'agnostic' approach to missing data that is most similar to one considered in ReliefC (Kononenko, 1994). The idea behind an agnostic approach is that unknown, missing values should be ignored (i.e. treated neutrally) using normalization to bypass their inclusion rather than attempt to make a guess about their respective values. In contrast, the ReliefC method is only partially agnostic, in that uses the ReliefB method (Kononenko, 1994) to naively contribute a *diff* of $1 - \frac{1}{\#Unique\_Feature\_Values}$ when a missing value is encountered in calculating the distance between an instance pair (Kononenko, 1994). For example, this contribution would be 0.5 if the feature had two possible states, or 0.25 if it had four. However, when ReliefC updates feature weights, features with missing values contribute nothing and the distance score is normalized to reflect that it was calculated using $(a - \#Missing\_Features)$, where $\#Missing\_Features$ is the number of features where a missing value was observed for at least one of the two instances. Alternatively, ReBATE methods apply this agnostic treatment of missing data to both the calculation of instance pair distances as well as for feature weight updates. This approach easily integrates with all RBAs, and all other data-type extensions. To the best of our knowledge this study is the first to implement and test a fully agnostic missing data approach in RBAs.

2.2 Evaluation

In the present study, we compare 13 feature selection approaches over an archive of 2280 simulated datasets representing a variety of problem and data types. In addition to the 5 ReBATE algorithms already reviewed or described above (ReliefF, SURF, SURF*, MultiSURF*, and MultiSURF) we examine 3 alternate run settings for ReliefF as well as 5 established non-RBA feature selection methods.

2.2.1 ReliefF Runs

The different ReliefF runs will be labeled in the results as: ReliefF 10 NN (i.e. original ReliefF), ReliefF 100 NN, ReliefF 10% NN, and ReliefF 50% NN. The first two are ReliefF with a k of 10 or 100, respectively. The second two consider setting k in a dataset dependent manner, setting k based on a user defined percent of instances in the data. For example, if n were 1000 instances, ReliefF 10% NN would utilize 100 total instances, thus $k = 50$, i.e. 50 hits and 50 misses.

We explore these different settings of k in ReliefF to explore how the number of nearest neighbors impacts performance, as well as whether setting k based on a percentage of instances offers a potential alternative to threshold-based neighbor selection as used by SURF, SURF*, MultiSURF*, and MultiSURF.

2.2.2 Other Feature Selection Algorithms

We compare the ReBATE methods to a cross section of feature selection methods available in scikit-learn (Pedregosa et al., 2011). Specifically we compare to three established filter methods including the chi squared test (Vafaie and Imam, 1994; Zheng et al., 2004; Jin et al., 2006; Moore and White, 2007; Witten et al., 2016), ANOVA F-value (Guyon and Elisseeff, 2003; Forman, 2003; Jafari and Azuaje, 2006; Jović et al., 2015), and mutual information (i.e. information gain) (Hunt et al., 1966; Vergara and Estévez, 2014; Hoque et al., 2014). Like most filter methods, these methods are myopic, i.e. not expected to handle feature dependencies due to assumptions of variable independence. Notably, the chi squared test is limited to problems with a discrete endpoint, and the ANOVA F-value was selected for its applicability to multi-class endpoints.

Further, we compare two wrapper methods, each based on a random forest of decision trees, i.e. (1) ExtraTrees, and (2) RFE ExtraTrees (Geurts et al., 2006). Random forests have been recognized as a powerful ensemble machine learning approach (Liaw et al., 2002). For both algorithms, we utilized 500 estimators (i.e. number of trees in the forest), and left all other parameters to scikit-learn defaults. Feature importance scores output by either algorithm are used to rank features by potential relevance. RFE ExtraTrees is a random forest combined with a *recursive feature elimination* algorithm. We have run RFE ExtraTrees removing one feature each step (the most conservative setting). RFE ExtraTrees is an iterative approach, recalculating feature importance of the remaining features each iteration. We expect this algorithm to be, by far, the most computationally expensive of those evaluated. RFE ExtraTrees would be more fairly compared to iterative versions of RBAs including TuRF, but we include it here to emphasize the comparative power of core RBA methods.

2.2.3 Simulation Study

In previous comparisons of ReliefF, SURF, SURF*, and MultiSURF* these methods were evaluated on datasets with purely epistatic 2-way interactions (i.e. no main effects) with varying numbers of training instances (e.g. 200 to 3200) as well as different heritabilities (e.g. 0.01 to 0.4) (Greene et al., 2009, 2010; Granizo-Mackenzie and Moore, 2013). Heritability is a genetics term that indicates how much endpoint variation is due to the genetic features. In the present context, heritability can be viewed as the signal magnitude, where a heritability of 1 is a ‘clean’ dataset (i.e. with the correct model, endpoint values will always be correctly predicted based on feature values), and a heritability of 0 would be a completely noisy dataset with no meaningful endpoint associations. All features were simulated as single nucleotide polymorphisms (SNP) that could have have a discrete value of (0, 1, or 2) representing possible genotypes. In each dataset, two features were predictive (i.e. relevant) of a binary class while the remaining 998 features were randomly generated, based on genetic guidelines of expected genotype frequencies, yielding a total of 1000 features. Similarly, VLSRelief explored SNP simulations and 2-way epistasis varying heritability similar to the other studies, but fixing datasets to 1600 instances and simulating datasets with either 5000 or 100,000 total features (Eppstein and Haake, 2008). It should be noted that most of these studies sought to compare core RBAs to respective iterative TuRF expansions, which is why larger feature spaces were simulated.

Simulation studies such as these facilitate proper evaluation and comparison of methodologies because a simulation study can be designed by systematically varying key experimental conditions, and the ground truth of the dataset is known i.e. we know which features are relevant vs. irrelevant, we know the pattern of association between relevant features and endpoint, and we know how much signal is in the dataset (so we know what testing accuracy should be achievable in downstream modeling). This allows us to perform power analyses over simulated dataset replicates to directly evaluate the success rate of our methodologies.

For these reasons, we adopt the position that the best way to compare and evaluate machine learning methodologies is over a diverse panel of simulated datasets designed to ask generalizable questions about what a method can and cannot handle (e.g. 2-way epistatic interactions, missing data, high noise, a moderate sample size (800), etc.). Ultimately, it may not be very important that the same benchmark datasets are used across studies, but rather that simulation studies are designed to ask fundamental questions about generalizable methodological functionality.

Therefore in the present study we have designed and applied a simulation study that is inspired by, but goes well beyond, the other bioinformatic evaluations previously described. Broadly speaking, our simulation study is founded around a core set of pure 2-way interaction SNP datasets similar to those previously benchmarked, but we expand beyond these to include groups of SNP datasets with (1) a variety of simple main effects, (2) 3-way interactions, (3) genetic heterogeneity, (4) continuous-valued features, (5) a mix of discrete and continuous features, (6) multi-class endpoints, (7) continuous endpoints, (8) missing data, and (9) imbalanced data. Additionally, we include some clean toy benchmark datasets including the XOR problem (2-way to 5-way interactions) to explore higher order interactions, and the multiplexer problem (6-bit to 135 bit variations) to explore epistasis and heterogeneous associations simultaneously.

Table 1 breaks down the characteristics of each unique dataset group. Outside of the core datasets, most of the other dataset groups retain constraint settings known to be solvable among the core datasets. These include the inclusion of a 2-way interaction, 20 features, a heritability of 0.4, and 1600 training instances. This is done because it would be very computationally expensive to evaluate a full factorial set of dataset variations over all dataset constraints.

Any dataset group in Table 1 that simulates a 2-way pure epistatic interaction is marked by a '*'. The left-most column describes the generalized pattern of association or data type that might reflect a strength or weakness we wish to assess. The 'Configurations' column indicates the number of unique dataset configurations that were included in the respective group. For example, the first group of 'core datasets' includes 32 configurations, i.e. 2 model difficulties * 4 heritabilities * 4 instance counts. The 'Model Difficulty' refers the model architecture of the underlying simulated genetic model (Urbanowicz et al., 2012b). To capture this dimension of dataset complexity, we select models generated at the extremes of model difficulty labeled here as 'easy' (E) and 'hard' (H). The 'Config. Variation' column is a catchall for configuration variations in a given group. The 'Simulation Method' refers to the strategy used to generate the datasets. Datasets simulated with the GAMETES complex genetic model and dataset generation software (Urbanowicz et al., 2012c,b) are labeled with 'G'. Those generated by a custom script are labeled with 'C'. Those generated by GAMETES but later modified (e.g. discrete values transformed into a range of continuous values) are labeled with 'G+C'.

To clarify specific dataset groups, *2-feature additive effect* includes two features with main effects that are additively combined to determine endpoint. The ratio 50:50 indicates that both features equally influenced endpoint, while 75:25, indicates that one had a 3 times the influence (and thus the relevance) of the other. Regarding *4-feature additive*

Table 1: Simulation study datasets. 30 replicates of each dataset configuration were generated.

Simulated Data Group Description or Pattern of Association	**Configurations**	**Config. Variations**	**Predictive Features**	**Total Features**	**Model Difficulty**	**Heritability**	**Instances**	**Simulation Method**
2-way Pure Epistais (Core Datasets) *Others marked by '*'*	32	-	2	20	E, H	0.05, 0.1, 0.2, 0.4	200, 400, 800, 1600	G
1-Feature Main Effect	8	-	1	20	E, H	0.05, 0.1, 0.2, 0.4	1600	G
2-Feature Additive Effect	2	50:50, 75:25	2	20	E	0.4	1600	G
4-Feature Additive Effect	1	-	1	20	E	0.4	1600	G
4-Feat. Additive 2-way Epistasis	2	50:50, 75:25	2	20	E	0.4	1600	G
4-Feat. Heterogeneous 2-way Epistasis	2	50:50, 75:25	2	20	E	0.4	1600	G
3-way Pure Epistasis	1	-	3	20	E	0.2	1600	G
Number of Features*	4	-	2	100, 1000, 10000, 100000	E	0.4	1600	G
Continuous Features*	1	-	2	20	E	0.4	1600	G+C
Mix of Discrete and Continuous Features*	1	-	2	20	E	0.4	1600	G+C
Continuous Endpoint*	3	0.2, 0.5, 0.8	2	20	E	0.4	1600	G
Continuous Endpoint* (1-Threshold Model)	1	-	2	20	E	0.4	1600	G+C
Missing Data*	4	0.001, 0.01, 0.1, 0.5	2	20	E	0.4	1600	G+C
Imbalanced Data*	2	0.6, 0.9	2	20	E	0.4	1600	G
Multi-class Endpoint (Impure 2-way Epistasis)	2	3-class, 9-class	2	20	N/A	1	1600	C
XOR Model (Pure Epistasis)	4	2-way, 3-way, 4-way, 5-way	2 3 4 5	20	N/A	1	1600	C
Multiplexer (MUX) (Pure Epistasis and Heterogeneous Associations)	6	6-bit → 11-bit → 20-bit → 37-bit → 70-bit → 135-bit →	2 3 4 5 6 7	6 11 20 37 70 135	3-way 4-way 5-way 6-way 7-way 8-way	1	500 1000 2000 5000 10000 20000	C

effect, all four features contribute equally to endpoint. The group, *4-feature additive 2-way epistasis*, additively combines two distinct 2-way pure epistatic interactions, where each pair has the respective ratio of influence. The group, *4-feature heterogeneous 2-way epistasis*, generates a heterogeneous pattern of association between two independent 2-way pure epistatic interactions. This would be an example of simulated genetic heterogeneity since these are SNP datasets. The group, *continuous endpoints* is an example of a regression problem, often known in genetics as a quantitative trait endpoint. We apply the GAMETES software (Urbanowicz et al., 2012c) to generate quantitative trait values around each pairwise genotype combination with a standard deviation of either 0.2, 0.5, or 0.8. For these datasets, the effective heritability is degraded as the standard deviation setting increases. The group, *continuous endpoint* with a 1-threshold model, refers to an alternative approach to generating continuous endpoint datasets. This approach takes a SNP dataset generated by GAMETES software and converts any instance with a class of 0 to a random value between 0 and 50, and any with a class of 1 to a random value between 50 and 100. This creates a continuous endpoint scenario, where a meaningful quantitative threshold exists in the data (50 in this case). Notably, we use a similar approach to generate our continuous-valued features, and mixed discrete/continuous feature datasets. However for these, SNP values of 0, 1, or 2 are converted to random value between 0 and 50, 50 and 100, and 100 and 150, respectively. For *missing data*, different frequencies of 'N/A's' are added to respective core datasets. For *imbalanced data* the given class imbalance ratios are simulated. For *multi-class endpoint* we simulated SNP datasets with a model similar to the XOR model, however each 2-way genotype combination is assigned either one of 3 classes or one of 9-classes. In both situations datasets are generated with impure epistatic interactions (meaning that individual features each also have some main effect).

Lastly, we have included six multiplexer problem datasets. Multiplexer problems are detailed in (Urbanowicz and Browne, 2017). In summary, they are clean problems with binary feature values, and a binary endpoint that concurrently model a patterns of epistastis and heterogeneous associations. We set these datasets apart in the table, because each dataset has a unique set of characteristics. For instance, the 20-bit multiplexer has 20 total features, involves heterogeneous groups of 5-way pure epistastic interactions, and includes a sample size of 2000. Notably in all multiplexer problems, all the features are technically predictive (in at least some subset of the training data). However, for any multiplexer problem, specific features known as 'address bits' are predictive in every instance. We specify the number of address bits under the 'Predictive Features' column and the order of epistatic interaction in the 'Model Difficulty' column. Previously it was noted in (Urbanowicz and Moore, 2015) that properly prioritizing address bits over other features in feature weighting with MultiSURF* was key to solving the 135-bit multiplexer problem directly. Therefore, with respect to the multiplexer problems, we evaluate the ability of feature selection methods to rank address bits as predictive features above all others (regardless of the fact that all features are technically predictive).

In total we consider 76 unique dataset configurations, generating 30 randomly seeded replicate datsets for each configuration ($76 * 30 = 2280$ total datasets). We have made these datasets available for download[5]. There are certainly many other dataset variations that could be included in the future, but this proposed set represents the most diverse simulation study of RBAs to date, offering a broad snapshot of the basic strengths and weaknesses of the feature selection methods evaluated in this study.

5. `https://github.com/EpistasisLab/rebate-benchmark`

2.2.4 Analysis

The strategy for evaluating a feature selection approach can depend on whether it outputs a ranked feature list (i.e. individual evaluation filter approaches), or a specific feature subset (all methods can do this). Assuming a ranked feature list and a dataset where the ground truth is known ahead of time, it is most common to examine where all relevant features rank in the ordered feature score list (i.e. are they at the very top or among some top percentile?) (Kira and Rendell, 1992b; Flórez-López, 2002; Moore and White, 2007; Eppstein and Haake, 2008; Greene et al., 2009, 2010; Stokes and Visweswaran, 2012; Granizo-Mackenzie and Moore, 2013). Ideally, relevant features will all have higher scores than irrelevant features as a best case scenario but it is most important that relevant features at least make it above the selected feature subset cutoff. Other metrics including *separability* and *usability* have also been proposed (Robnik-Šikonja and Kononenko, 2003).

Alternatively, if the feature selection approach outputs a feature subset we can evaluate success by (1) examining the number of relevant and irrelevant features that comprise a selected feature subset (assuming ground truth is known) (Belanche and González, 2011; Bolón-Canedo et al., 2013), or (2) determining the testing accuracy of some induction algorithm model trained on that feature subset (if ground truth is not known) (Sun and Li, 2006; Bolón-Canedo et al., 2013). The downside to the second approach is that it is difficult to separate the performance of the feature selection approach from the modeling of the induction algorithm. If we are dealing with an individual evaluation approach that has employed a selection cutoff to define a feature subset, then the downside to the first approach is that we are evaluating the feature weighting as well as the cutoff criteria (which can also be difficult to separate).

For individual evaluation filter approaches like RBAs, the best way to evaluate performance on simulation study data where the ground truth is known, is to identify where the relevant feature rank in the ordered feature score list over a number of dataset replicates. This offers the clearest analysis for interpretation and is consistent with previous evaluations of the selected ReBATE methods. Specifically we apply a power analysis, examining where the lowest scoring of the relevant features ranks in the ordered feature list. Power, i.e. success rate, is then calculated as the proportion of successes out of the 30 replicate datasets. In this study, we calculate and report the power of each algorithm to identify all predictive features within each percentile of the ranked feature list.

As mentioned in this study, we compare 13 feature selection approaches across 76 unique dataset configurations. We have generated heatmaps displaying feature selection power at each percentile of a respective feature list. As an example, see Figure 2. On the y-axis we have our 13 algorithms, along with the results of a negative control, labeled as 'Random Shuffle'. This negative control represents shuffling the feature list randomly 30 different times and calculating power. In other words, this is the power expected by randomly ranking features. The 13 feature selection algorithms are ordered and delineated into groups. From the top down, the first group includes the myopic filter-based feature selection methods (i.e. chi square, ANOVA F-value, and mutual information). The second group includes the random forest wrapper methods (i.e. ExtraTrees and RFE ExtraTrees). The third group includes the ReliefF algorithm with different settings of the k parameter. The fourth group includes the set of recent RBAs that have eliminated the k parameter, and the last algorithm

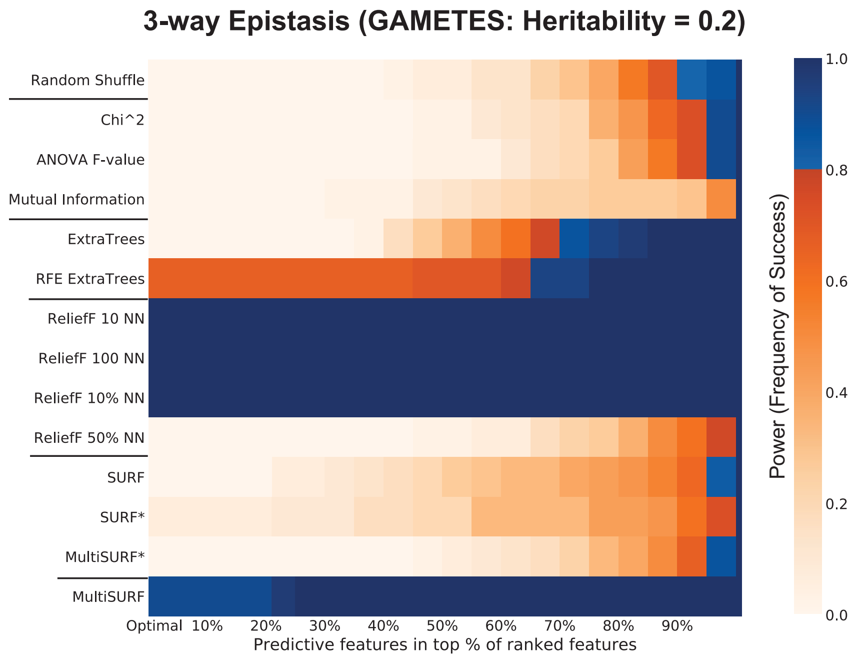

Figure 2: This heatmap illustrates the power of different feature selection algorithms to rank all predictive features in the the top scoring ‘x’ percent of features in the dataset. Results for the noisy 3-way epistatic interaction.

is MultiSURF, proposed in this study. We will use lines delineating these algorithm groups in some figures where the text of the algorithm names would be too small to read, but the order of these algorithms is preserved for all remaining power analysis figures.

The x-axis of Figure 2 gives the percentile of the ranked feature scores (e.g. 30% represents the top 30% of all feature scores). The zero percentile is marked as ‘optimal’. At optimal, all relevant features are scored above all irrelevant features. To the right of the figure is the key depicting increasing power with increasing color darkness. To facilitate interpretation, any measure of power at or above 0.8 (i.e. 80% power) is given as a shade of blue, rather than a shade of orange. While somewhat of an arbitrary selection, 80% power or above is often used as a significance cutoff for success rate. It is useful here to more quickly identify the lowest feature percentile within which all relevant features are ranked with significant rate of success (i.e. the threshold between orange and blue). For example, if we look at the power of RFE ExtraTrees in Figure 2, we can see that this algorithm successfully ranks all relevant features somewhere in the top 67% of features for at least 80% of the replicate datasets. An algorithm that performs perfectly will have dark blue band (100% power) over the entire percentile range (e.g. see ReliefF 10 NN in Figure 2). This indicates that the algorithm succeeded in scoring all relevant features above each irrelevant feature in every replicate dataset. Keep in mind that when it comes to selecting a feature subset from a ranked feature list, the minimum basis for success is whether all predictive features are included within that set, not whether all relevant features are ranked above every irrelevant feature. For example if we had decided to keep the top 25% of features in subset selection, then we would want to pick an algorithm that had reliable power at the 25th percentile. Lastly, notice that by the 100th percentile, all algorithms will report 100% power. This is to be expected since we know that all relevant features will be found *somewhere* in the entire feature set. A Jupyter notebook including our analysis code and power analysis figure generation has been made available[6].

3. Results

The results of this study are organized by major data configuration themes.

3.1 2-way Epistasis

Figure 3 presents the results for the core set of 2-way pure epistastic interaction datasets. This figure assembles power plots over a range of heritabilities (left y-axis), number of instances (x-axis), and the two model architecture difficulties, E and H (right y-axis). Each of the 32 subplots represents a power analysis for one data configuration. They are arranged roughly so that the 'easiest' configurations are towards the upper right corner (e.g. heritability = 0.4, $n = 1600$, and architecture = E), and the most 'difficult' configurations are towards the lower left corner (e.g. heritability = 0.05, $n = 200$, and architecture = H).

The most obvious observation from this analysis is that the three myopic filter algorithms (i.e. chi square, ANOVA F-test, and mutual information) consistently fail to successfully rank relevant features with 2-way interactions, i.e. their performance is on par with the random shuffle negative control. Also notice that overall power of all other algorithms deteriorates as we go from the upper right plot to the lower left plot. This is consistent with the the expectation that power will degrade with decreasing heritability, training set size, and a 'harder' model architecture. Note how our simulation study design includes data configurations where our RBAs of interest completely succeed (upper right) as well as completely fail (lower left). These trends are consistent with previous evaluations of ReliefF (10 NN), SURF, SURF*, and MultiSURF* (Greene et al., 2009, 2010; Granizo-Mackenzie and Moore, 2013).

Focusing on the performance of ReliefF with different settings of k one observation stands out. Specifically ReliefF 100 NN, fails in all configurations where $n = 200$. For a k of 100 in a balanced dataset (such as this), ReliefF is using all other instances as neighbors. As we reviewed by Robnik-Šikonja et al. (2003), this effectively removes the requirement that instances used in scoring be 'near' and turns ReliefF (using all neighbors) into a myopic algorithm, unable to handle 2-way interactions. This is verified empirically by these results. In contrast, examination of ReliefF 10 NN results (e.g. heritability = 0.05, $n = 800$ or $n = 1600$, difficulty = E) in contrast to other RBAs suggests that increasing noise (i.e. lower heritability) is better handled by a somewhat larger k. This is consistent with previous observations (Kononenko, 1994; Greene et al., 2009). It should be noted the concept of a 'low' or 'high' k should always be considered with respect to n. For example, while $k = 100$ was detrimental in a sample size of 200, this setting performed quite well when n was 400 to 1600.

6. `https://github.com/EpistasisLab/rebate-benchmark`

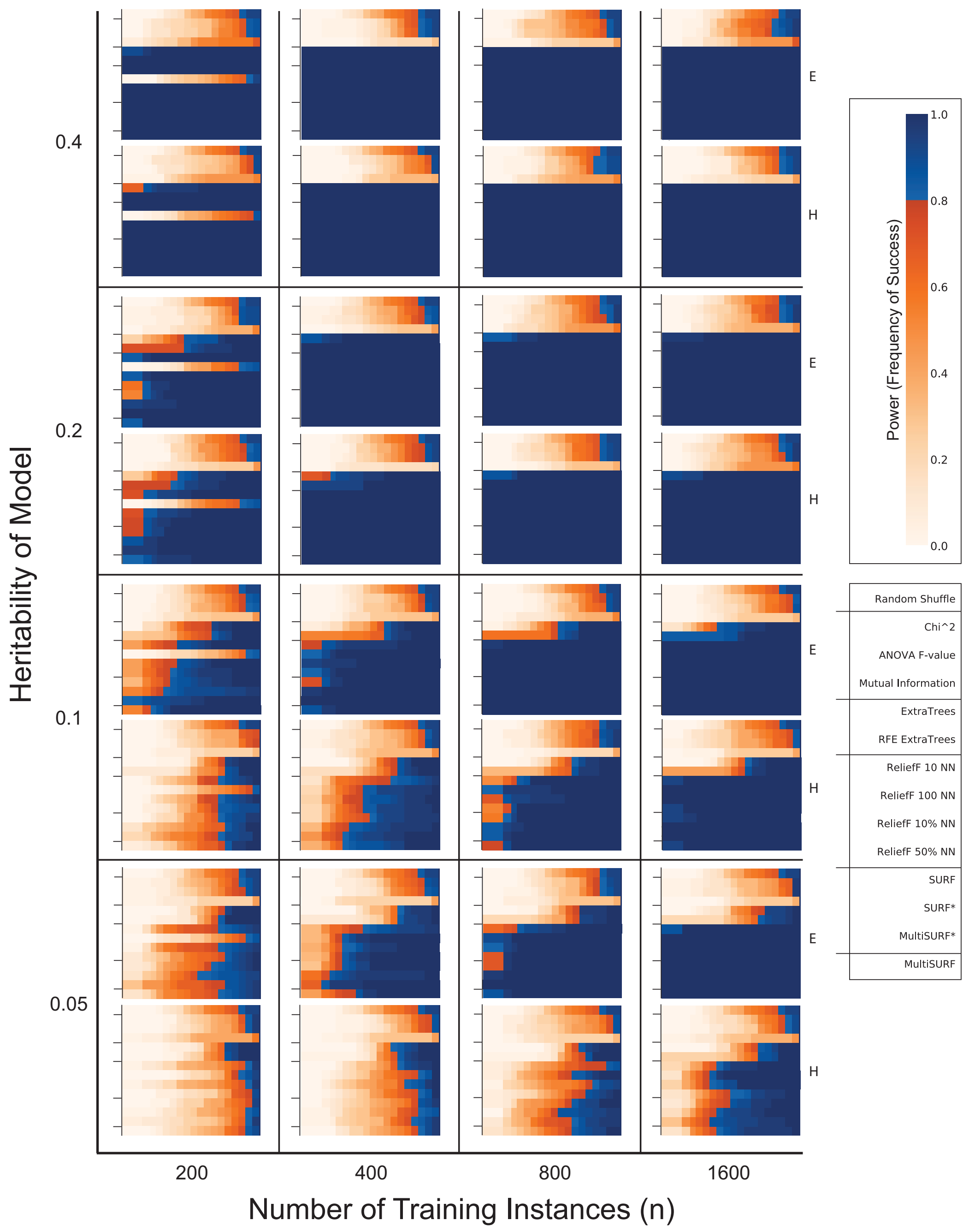

Figure 3: Results for all core 2-way epistatic interaction datasets. Keys relevant to all plots are given on the far right. Tick marks delineating algorithm groups are provided for each sub-plot.

Regarding the the ExtraTrees wrapper algorithms, we note that these random forest approaches were able to detect pure 2-way epistatic interactions (at least in datasets with

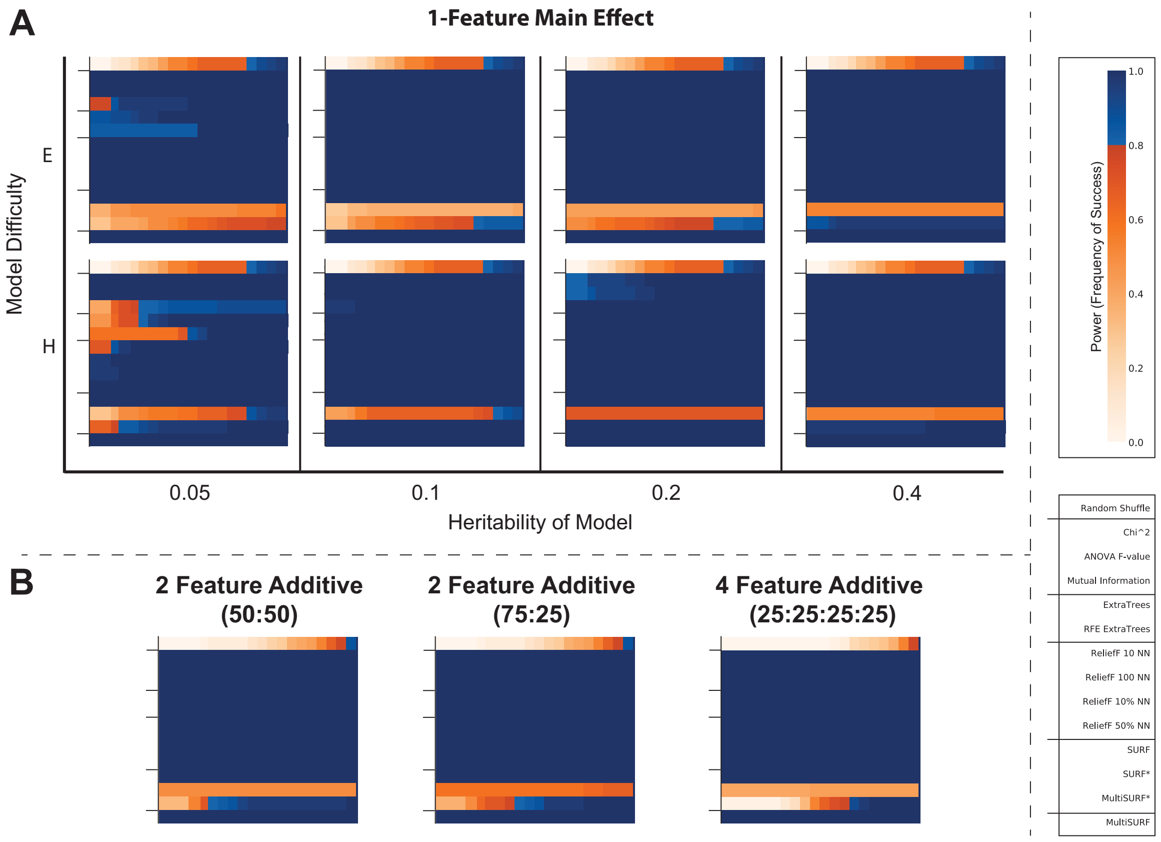

Figure 4: Results for detecting single feature main effects (A) and additive main effects (B). Keys relevant to all plots are given on the far right. Tick marks delineating algorithm groups are provided for each sub-plot.

20 features). Comparing them to RBAs we observe random forest performance to be more negatively impacted by increased noise and decreased n.

Focusing on SURF, SURF*, and MultiSURF*, the results in Figure 3 subtly but consistently support the previous findings that over a spectrum of 2-way pure epistatic interactions, SURF $<$ SURF* $<$ MultiSURF* with respect to power (Greene et al., 2009, 2010; Granizo-Mackenzie and Moore, 2013). Lastly, examination of our proposed MultiSURF variant, suggests that it's performance is generally competitive with other RBAs (on par with SURF (Greene et al., 2009)), but is slightly outperformed by SURF* and MultiSURF* (both methods that adopt far scoring) on 2-way epistasis problems with increased noise and decreased n. For a detailed discussion regarding why far scoring improves the detection of 2-way interactions, see Section 4.

3.2 Main Effects

Main effects (i.e. the effect of a single independent variable on a dependent variable) are generally understood to be easier and less computationally expensive to detect than

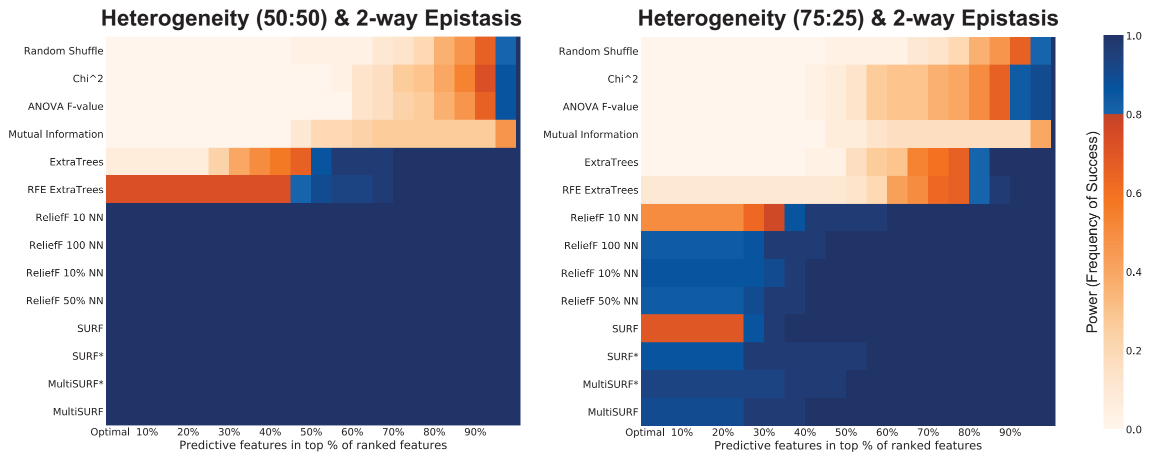

Figure 5: Results for detecting two independent heterogeneous 2-way epistatic interactions.

interactions. It is likely for this reason that the RBAs in ReBATE had never been tested on simulations of single feature main effects or multiple feature additive main effects. Figure 4A presents main effect results over increasing heritabilities (x-axis) and model difficulty (y-axis). First, notice that all three myopic filter algorithms (i.e. chi squared test, ANOVA F-value, and mutual information) generally succeed at identifying the respective main effects. Of the three, mutual information was least successful, particularly when heritability was low and the model was 'hard'. A similar performance loss was observed in the random forest wrappers.

The most dramatic and unexpected finding in this analysis was the performance loss of SURF* and MultiSURF*, the only two algorithms utilizing 'far' scoring. Interestingly, this loss was even more significant in easy main effect models. This is in contrast with ReliefF, SURF, and our proposed MultiSURF algorithm that were completely successful here. Examination of additive multi-feature main effects in Figure 4B each with a heritability of 0.4 and 'E' model, reveal similar performance losses for SURF* and MultiSURF*. Notably, for all main effect datasets, MultiSURF* has less of a performance loss than SURF*. For a detailed discussion regarding why far scoring hinders or eliminates the ability to detect main effects, see Section 4.

3.3 Genetic Heterogeneity

Figure 5 gives the power analysis results for data that models heterogeneous patterns of association between independent 2-way epistatic interactions. In other words, in one subset of training instances one 2-way interaction is relevant, and in the other a different pair of interacting features are relevant. The given ratio indicates the proportion of instances within which each interaction is relevant. Heterogeneous patterns of association have been commonly recognized in biomedical problems and are known to confound traditional machine learning approaches (Ritchie et al., 2003; Thornton-Wells et al., 2006; Urbanowicz and Moore, 2010; Urbanowicz et al., 2013). Accounting for such patterns in feature selection is thus an important target. Figure 5 suggests that all tested RBAs can handle heterogeneity concurrently modeled with epistatic interactions, while all other methods fail, or fail to perform nearly as well, in the case of the random forest wrappers. With a more extreme ratio of 75:25, MultiSURF*, and MultiSURF appear to perform best with SURF* and ReliefF (with larger k settings) close behind. Overall, RBAs in general appear uniquely suited to detecting patterns of both heterogeneous association. To the best of our knowledge this is the first formal evaluation of RBAs on heterogeneous patterns of association.

3.4 3-way Epistasis

Referring back to Figure 2 we provide the first evaluation of RBA performance on epistatic interactions with a dimmensionality higher than 2 (i.e. 3-way interactions). Due to mathematical constraints, the GAMETES epistasis dataset simulation software was unable to generate a 3-way SNP interaction dataset with a heritability of 0.4, so instead we simulated 3-way interaction SNP datasets with heritability=0.2. As expected, the myopic methods fail to perform well. This is also true for the ExtraTrees wrappers. Interestingly, the only RBAs that succeeded on this problem were those that utilized the smallest number of neighbors in scoring (i.e. ReliefF with 10 or 100 NN, ReliefF with 10% NN, and our proposed MultiSURF algorithm). This suggests that detecting higher order interactions is best achieved when the number of neighbors is low with respect to n. In this analysis the dataset included 1600 instances. ReliefF performed well with 10, 100, or 80 $(0.1 * 1600/2)$ nearest hits and misses, but completely failed with 400 $(0.5 * 1600/2)$ nearest hits and misses. We expect that MultiSURF $<$ MultiSURF* $<$ SURF $<$ SURF* with respect to the number of instances involved in each scoring cycle, where SURF* utilizes all instances in scoring and MultiSURF should utilize the least given that scoring only includes the inside of the dead-band zone.

There is no evidence to suggest that 'far' scoring itself is helpful or harmful in this analysis. This is one prominent scenario where our proposed MultiSURF algorithm succeeds where other modern RBAs fail. While ReliefF performs slightly better, it's success is dependent on setting k properly. It may be useful to track the number of instances involved in scoring in these algorithm in future investigations of RBAs detecting higher order interactions.

3.5 Number of Features

All of the results presented so far were run on datasets with a relatively small number of features to save computational time while asking basic questions about algorithm abilities. Figure 6 presents algorithm power over an increasing number of irrelevant features in datasets with 2-way epistatic interactions. We examined feature space sizes up to the maximum number of features investigated in a previous simulation study of an RBA (i.e 100,000) (Eppstein and Haake, 2008). Keep in mind that we don't expect any of these core methods to perform particularly well in very large feature spaces without ultimately combining them with some iterative RBA approach. As usual, the myopic approaches fail to detect 2-way interactions. The next methods to fail in a feature space of increasing size are the random forest wrappers (at 1000 features). At 10,000 features, ReliefF with $k = 10$ begins to fail suggesting that a small number of neighbors performs less well in noisy

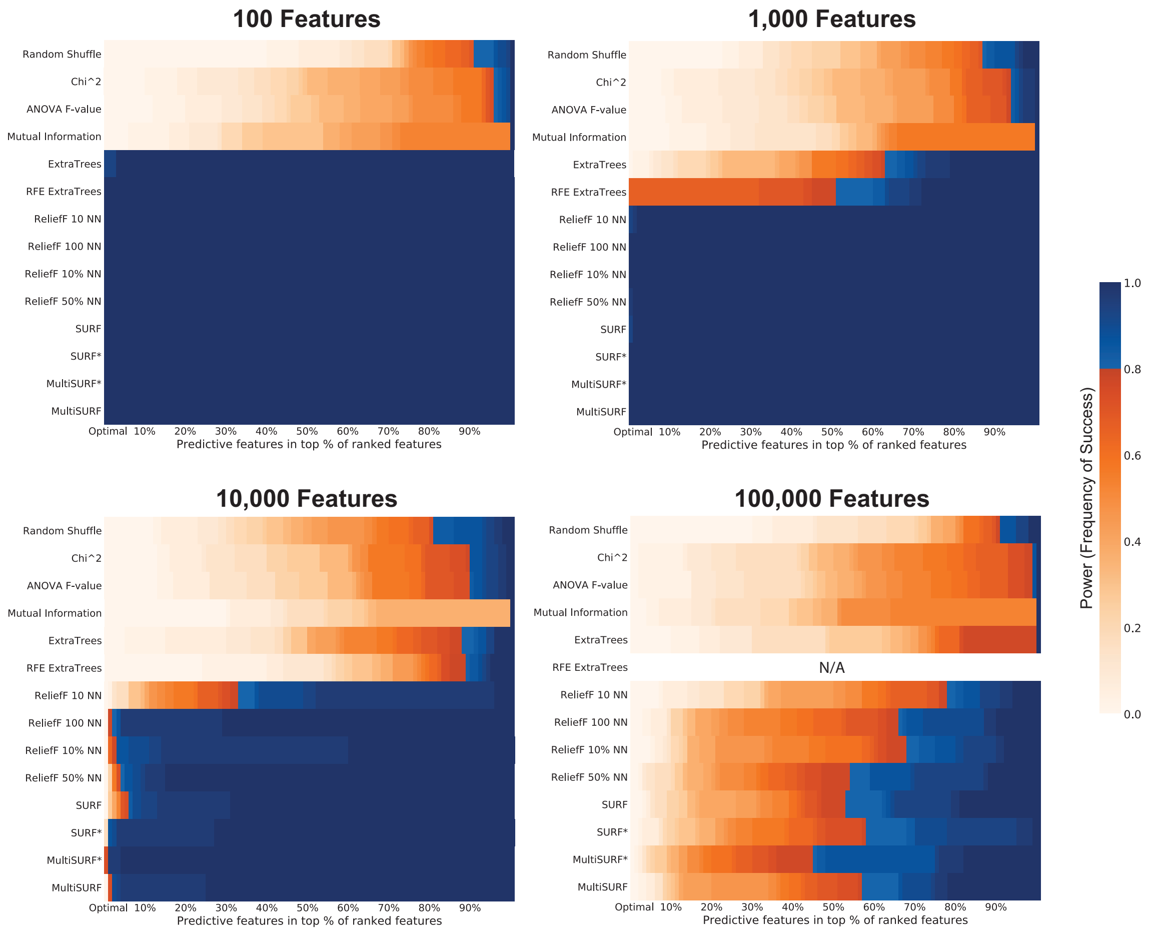

Figure 6: Results for detecting 2-way epistatic interactions with an increasing number of irrelevant features in the datasets.

problems, particularly as the feature space grows. At 10,000 features, MultiSURF* appears to perform slightly better than the rest, consistent with our previous findings in Figure 3. Lastly, as expected at 100,000 features none of the methods perform particularly well on their own, but still certainly better than a random shuffle. Results for RFE ExtraTrees are missing because this iterative random forest approach did not finish running within a reasonable amount of time (i.e. over two days). Based on the results of RFE ExtraTrees in smaller feature sets, it is reasonable to assume it would have performed poorly at 100,000 features as well. Notably, most RBAs (with the exception of ReliefF 10 NN, 100 NN, and 10% NN) demonstrate high power (i.e. $> 80\%$) to rank the predictive features above the 60th percentile, and MultiSURF* is the only method with significant power above the 50th percentile. Also of note, if we look for the percentile at which algorithms acheive full power, our experimental MultiSURF achieves the best percentile (i.e. approximately the 80th percentile). In combination with an iterative approach, for example TuRF, the lowest ranking features would be removed each iteration, improving the estimation of remaining feature

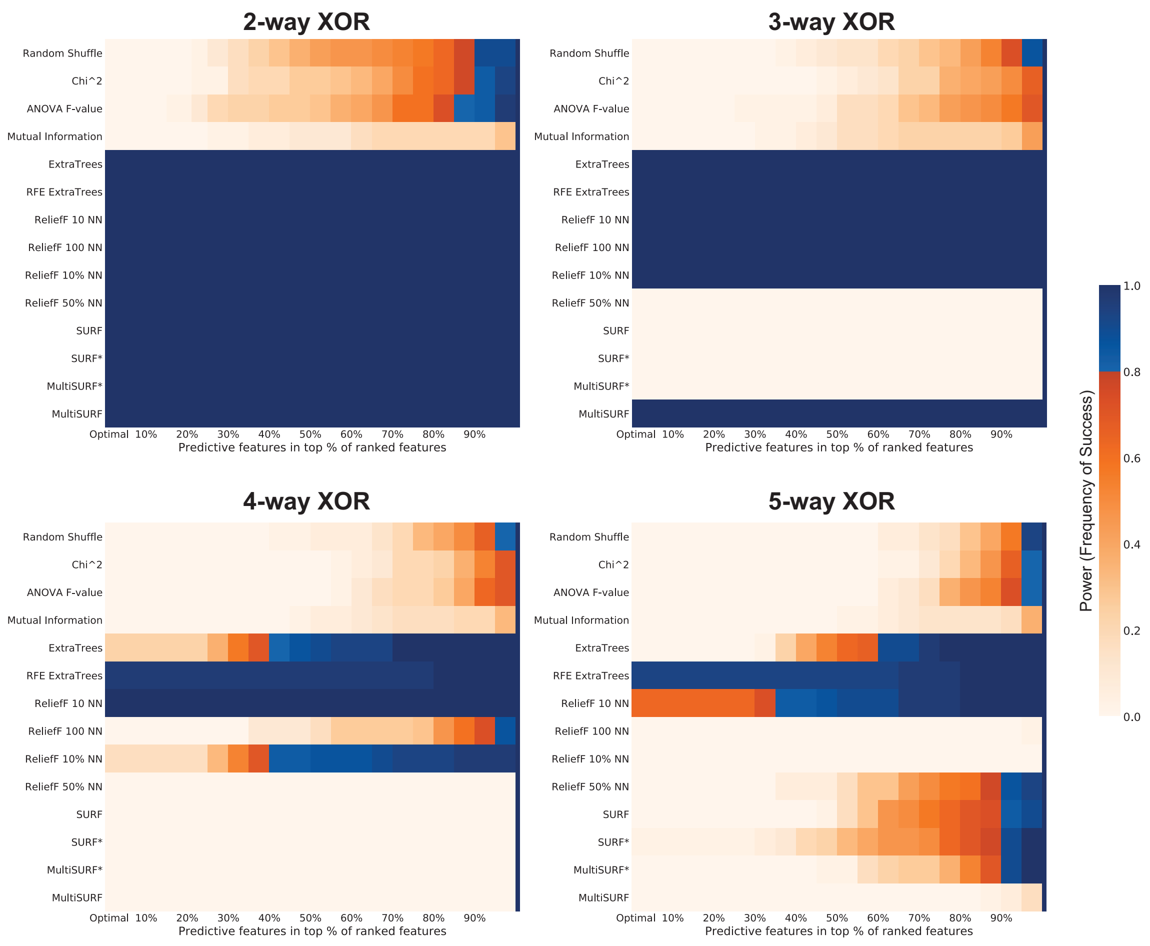

Figure 7: Results for detecting 2-way, 3-way, 4-way, and 5-way epistatic interactions based on ‘clean’ XOR models.

scores in subsequent iterations. Therefore it is useful here to consider what proportion of underlying features could be removed in the first (and subsequent) iterations without losing any relevant features.

3.6 XOR Benchmarks

Here we examine clean benchmark datasets that further test epistatic patterns. Figure 7 presents power analyses for increasingly higher order versions of the XOR problem simulated here as SNP data (Urbanowicz et al., 2012c). These XOR benchmark datasets have no noise and include pure epistatic interactions from 2-way up to 5-way interactions. While all non-myopic methods solve the 2-way XOR with little trouble, we observe a similar pattern of algorithm success for the 3-way XOR as we did with the noisy GAMETES generated 3-way dataset in Figure 2. Specifically, RBAs that utilized fewer neighbors were successful, including ReliefF with 10 or 100 NN, ReliefF with 10% NN and our proposed MultiSURF algorithm. Differently, the ExtraTrees algorithms were equally successful in detecting the 3-way XOR as those RBAs. Another interesting observation for the 3-way XOR is that for RBAs that used a larger proportion of neighbors in scoring, relevant features were consistently ranked with the lowest overall scores. This was also true for some of the 4-way and 5-way analyses. This is interesting because if the methods were not detecting any difference between relevant and irrelevant features, we would expect to see results similar to the random shuffle negative control. Instead, in specific high-order interaction problems, relevant features are consistently being assigned the most negative score updates. There may be opportunity for future RBA improvement leveraging this observation. Currently this is a problem for those respective RBAs since users will assume the at features scoring at the bottom of the feature ranking should be eliminated from consideration. It is only because we know the ground truth of these simulated datsets that we can observe this pattern here.

Overall, the only method able to solve all XOR problems is the RFE ExtraTrees, however based on our previous findings we expect this success to quickly disappear if the feature space was larger than 20 features. The next best algorithm is ReliefF with 10 neighbors, followed by ExtraTrees and ReliefF with 10% (i.e 80 neighbors). Notably, the the 4-way and 5-way XOR problems represent one situation in which our proposed MultiSURF algorithm fails to perform. It is clear that higher order interactions can be problematic for RBAs unless they apply few neighbors in scoring.

3.7 Multiplexer Benchmarks

Figure 8 presents power analyses for increasingly higher order versions of the multiplexer benchmark datasets including the 6-bit through the 135-bit versions. This power analysis should be interpreted somewhat differently from all others since in these datasets, technically all features are relevant (in at least some subset of training instances), and instead we evaluate the power to properly rank the address bits of the multiplexer problem (i.e. the features that are relevant for every instance in the dataset. We had previously observed in (Urbanowicz and Moore, 2015) that MultiSURF*'s ability to rank address bits over all others in feature weighting was one of the keys to solving the 135-bit multiplexer benchmark directly for the first time in the litterature.

Like the XOR problem, the multiplexer problems have no main effects. As a result, the myopic methods again fail to perform on these datasets. The only method to perform perfectly over all datasets was Relief 10% NN. MultiSURF* perfromed next best followed by SURF* and our MultiSURF method trailing close behind. These results are interesting but difficult to clearly interpret given that all features are technically predictive (particularly with respect to the 6-bit problem which is the 'easiest' to solve). However, this explains why feature weighting using MultiSURF* in ExSTraCS was facilitated solving the 135-bit multiplexer. These results also emphasize the ability of RBAs in general to detect relevant features in the presence of both feature interactions and heterogeneous patterns of association.

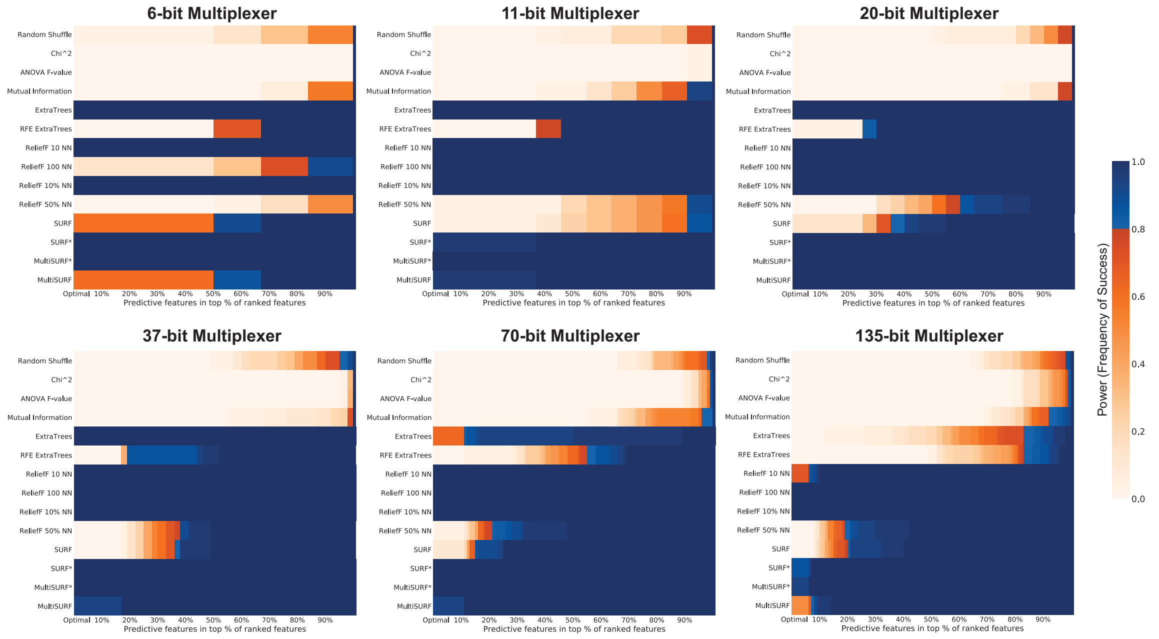

Figure 8: Results for detecting the address bits of different scalings of the Mulitplexer benchmark problem. Each problem is 'clean', epistatic, and heterogeneous. Note that all features in these datasets are predictive in at least one subset of the training instances, and power reflects the ability to rank the subset of features that are important in all training instances (address bits), from those that are important only in a given subset (register bits).

3.8 Data Types

We conclude our results section with power analyses for the different data type scenarios considered. This will demonstrate the functionality of our ReBATE algorithm data type extensions. Only power analyses yielding interesting differences are explicitly presented.

Figure 9 gives power results over four numerical endpoints configurations, each simulated with an underlying 2-way interaction. First, note the the Chi Square test is not applicable to data with continuous endpoints. As usual, the other myopic methods did not perform well with the exception of mutual information (on continuous endpoint data generated with a standard deviation of 0.2). It is possible that the strategy we employed to generate continuous endpoint SNP data from epistatic models introduced some small main effects at a low standard deviation that mutual information was able to pick up. Interestingly, the random forest methods performed slightly worse than the RBAs. Overall, the success of all RBAs implemented in ReBATE in these analyses suggests that our simpler and computationally less expensive proposed RBA regression approach offers a functional alternative to the one proposed in RReliefF (Kononenko et al., 1996; Robnik-Šikonja and Kononenko, 1997). A comparison between the two is outside the scope of the current investigation but should be the focus of future study.

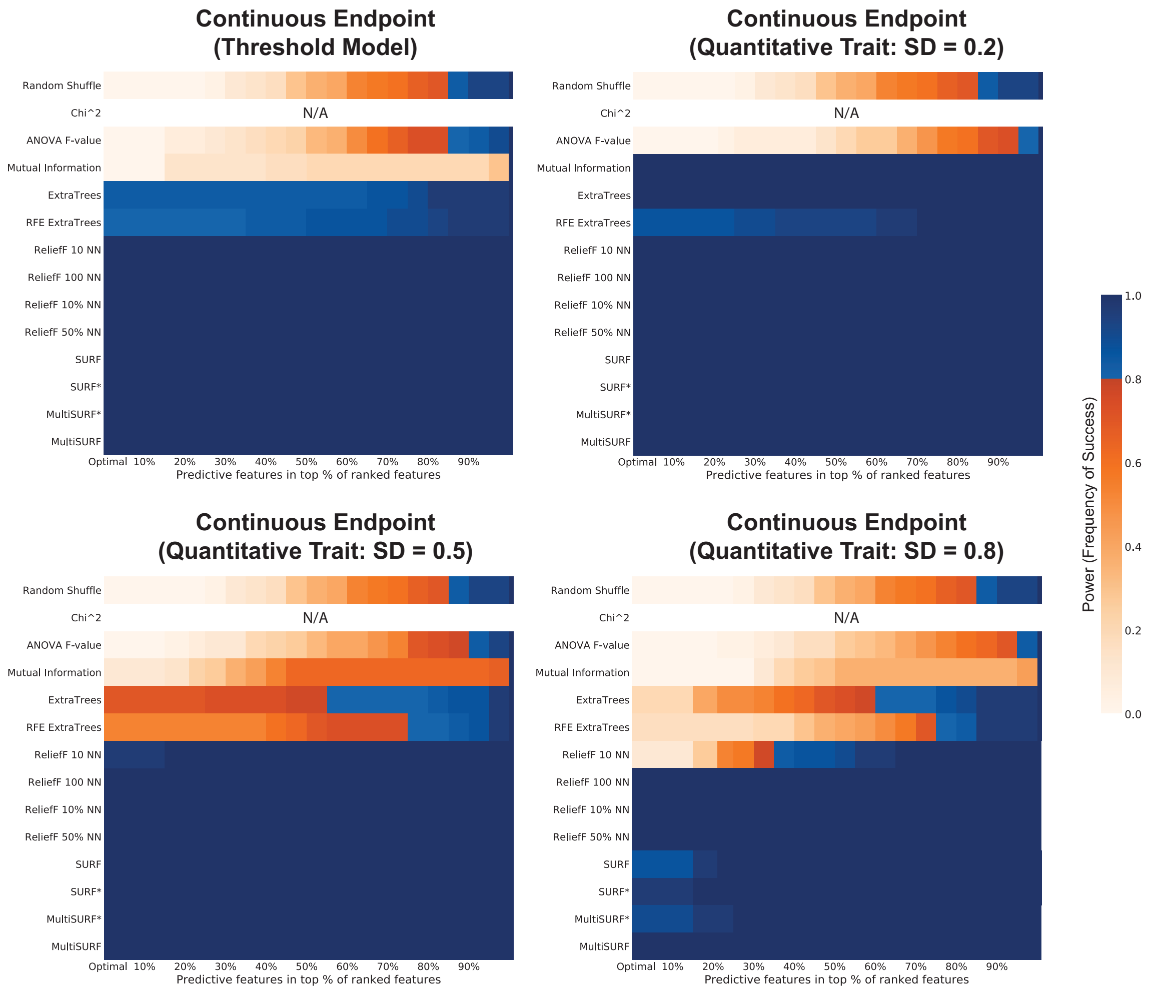

Figure 9: Results for accommodating continuous (i.e. numerical) endpoints in datasets.

Most of our power analyses for the other data type configurations yielded the same successful results for all RBAs, thus results are not presented here. Specifically, both multi-class data configurations were solved by all feature selection methods, demonstrating the basic efficacy of the multi-class expansion for Relief introduced in (Kononenko, 1994) and adopted in ReBATE. This also indicated that our simulated multi-class datasets (with impure epistasis) had strong enough main effects to be successfully ranked by all myopic methods.

For datasets with only continuous features and an underlying 2-way interaction, all but the myopic methods ranked features ideally, reinforcing the functionality of the proposed *diff* function for continuous features originally introduced in (Kira and Rendell, 1992b). The same was true for data with an imbalance of 0.6 (i.e. 60% class 0, 40% class 1), and for missing data with data randomly missing at a frequency of either 0.001, 0.01, or 0.1. Notably in the analyses of missing data, all 5 feature selection methods run with scikit-learn (i.e. chi square test, ANOVA F-test, mutual information, ExtraTrees, and RFE ExtraTrees)

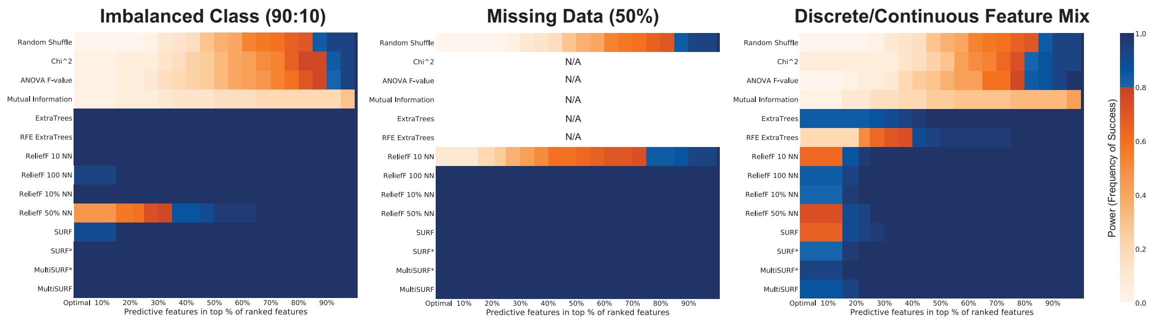

Figure 10: Results for accommodating different 'data type' issues. Specifically this figure examines extreme examples of class imbalance, missing data, and the combination of discrete and continuous features within the same dataset.

could not be completed since scikit-learn is not set up to handle missing data. Preprocessing such as removal of instances with missing values or imputation would instead be required.

Figure 10 presents all remaining results in which feature selection performance differences were observed. First off, for each, myopic approaches again failed to detect the underlying 2-way interactions. For a class imbalance of 0.9 (i.e. 90% class 0, 10% class 1), we observe that ReliefF with a large number of neighbors (i.e. 50%) fails to perform, ReliefF with 100 NN and SURF demonstrate slight deficits, but all other RBAs perform optimally. For missing data with a frequency of 0.5, we observe all ReliefF methods perform optimally with the exception of ReliefF 10 NN (the fewest neighbors in scoring). This suggests that having more neighbors in scoring makes these methods more resiliant to missing data, and also demonstrates that our proposed agnostic missing data strategy implemented in ReBATE is successful. A comparison between different missing data handling strategies is beyond the scope of this investigation, but should be examined in future study. Lastly, Figure 10 gives the results for the data configuration with a mix of discrete and continuous features. For each of the 30 dataset replicates a random half of the features is re-encoded with a continuous value, and the rest are left as discrete values. Similar to previous observations regarding mixed feature types with Relief (Kononenko and Šikonja, 2008), none of the RBAs or feature selection methods in general handle this dataset configuration optimally. MultiSURF* works best followed by MultiSURF, SURF*, and ReliefF with 100 NN or 10% NN. Overall this suggests that mixed feature types are still an issue for RBAs, and worth further study and development in ReBATE. Future work should consider the proposed ramp function (Hong, 1997; Robnik-Šikonja and Kononenko, 2003; Kononenko and Šikonja, 2008), as well as other approaches that do not require setting user parameters to address this issue.

4. Discussion

In this section we discuss why specific strengths and weaknesses were observed for respective methods under different data configurations.

4.1 2-way Epistastis Performance Gains

From the results, one may wonder why is it that ‘far’ scoring in SURF* and MultiSURF* appears to enhance power to detect 2-way interactions? To help answer this question, Table 2, offers a simple example 2-way epistasis dataset that we will use to walk through Relief scoring. In this example, A_1 and A_2 are relevant features with a pure interaction between them (i.e. no individual main effects). When they have opposing values, the class is one otherwise the class is zero. A_3 is an irrelevant feature.

Table 3 breaks down how scoring would proceed over 8 cycles with each instance getting to be the respective target. To simplify this example, we only select one near or far instance for each RBA, focusing instead on the impact of scoring scheme. Notice that we will keep track of nearest hits and misses (and the corresponding features with a different value), as well as far instances used in scoring by SURF* and MultiSURF* (i.e. the farthest hit or miss, each with both the same and different feature value(s) identified between instances). For each target, we see what instance is the nearest or farthest hit and miss, as well as which feature has a different value between the instances (given in parentheses). Further, we see what feature has the same value between farthest instances (in grey). All of these will be relevant to scoring in at least one of the ReBATE algorithms. If there is a tie for nearest neighbor, both instances are listed with their respective different valued feature. For example, when R_1 is the target, it’s nearest hit is R_2. The only feature with a different value between these two instances is A_3. The nearest miss for R_1 is a tie between R_5 and R_7 that have feature value differences at A_1 and A_2, respectively. The farthest miss for R_1 is a tie between R_6 and R_8 that have feature value differences at A_1 and A_3, and A_2 and A_3, respectively. Lastly, we keep track of the feature values that are the same between the farthest hits and misses (grey shaded cells). For the farthest hit, no feature values are the same since we already noted they were all different. For the farthest miss, we again have R_6 and R_8 that have feature value equalities at A_2 and A_1, respectively.

Table 2: Example dataset with interaction between A_1 and A_2. A_3 is irrelevant. Adapted from Kononenko et al. (1997).

Instances	A_1	A_2	A_3	C
R_1	1	0	1	1
R_2	1	0	0	1
R_3	0	1	1	1
R_4	0	1	0	1
R_5	0	0	1	0
R_6	0	0	0	0
R_7	1	1	1	0
R_8	1	1	0	0

Table 4 summarizes the resulting score contributions from Table 3. As before, when there is a tie between instances for nearest or farthest neighbor, we give each feature difference half credit since only one can contribute at a time. For example, since there are no same feature values for any of the farthest hits in Table 3 (shaded in grey), all three features get a zero score contribution under ‘Farthest Hit (same value)’ in Table 4.

Table 3: Breakdown of near/far, hit/miss, and features with different or same values for each target instance. Illustrates where feature scores come from for the 2-way epistasis dataset in Table 2.

	Far (Same Value)		Near (Different Value)		Far (Different Value)	
Target	Hit	Miss	Hit	Miss	Hit	Miss
R_1	None	$R_6(A_2)$,$R_8(A_1)$	$R_2(A_3)$	$R_5(A_1)$,$R_7(A_2)$	$R_4(A_1, A_2, A_3)$	$R_6(A_1, A_3)$,$R_8(A_2, A_3)$
R_2	None	$R_5(A_2)$,$R_7(A_1)$	$R_1(A_3)$	$R_6(A_1)$,$R_8(A_2)$	$R_3(A_1, A_2, A_3)$	$R_5(A_1, A_3)$,$R_7(A_2, A_3)$
R_3	None	$R_6(A_1)$,$R_8(A_2)$	$R_4(A_3)$	$R_5(A_2)$,$R_7(A_1)$	$R_2(A_1, A_2, A_3)$	$R_6(A_2, A_3)$,$R_8(A_1, A_3)$
R_4	None	$R_5(A_1)$,$R_7(A_2)$	$R_3(A_3)$	$R_6(A_2)$,$R_8(A_1)$	$R_1(A_1, A_2, A_3)$	$R_5(A_2, A_3)$,$R_7(A_1, A_3)$
R_5	None	$R_2(A_2)$,$R_4(A_1)$	$R_6(A_3)$	$R_1(A_1)$,$R_3(A_2)$	$R_8(A_1, A_2, A_3)$	$R_1(A_1, A_3)$,$R_4(A_2, A_3)$
R_6	None	$R_1(A_2)$,$R_3(A_1)$	$R_5(A_3)$	$R_2(A_1)$,$R_4(A_2)$	$R_7(A_1, A_2, A_3)$	$R_1(A_1, A_3)$,$R_3(A_2, A_3)$
R_7	None	$R_2(A_1)$,$R_4(A_2)$	$R_8(A_3)$	$R_1(A_2)$,$R_3(A_1)$	$R_6(A_1, A_2, A_3)$	$R_2(A_2, A_3)$,$R_4(A_1, A_3)$
R_8	None	$R_1(A_1)$,$R_3(A_2)$	$R_7(A_3)$	$R_2(A_2)$,$R_4(A_1)$	$R_5(A_1, A_2, A_3)$	$R_1(A_2, A_3)$,$R_3(A_1, A_3)$

Table 4: Summary of score contributions for 2-way epistasis dataset in Table 2 yielding ReliefF, SURF, SURF*, MultiSURF*, and MultiSURF scores. Illustrates why 'far' scoring improves 2-way interaction detection.

	Features			RBA Scoring Schemes				
	A_1	A_2	A_3	ReliefF	SURF	SURF*	MultiSURF*	MultiSURF
Farthest Hit (same value)	0	0	0				- 1	
Farthest Miss (same value)	4	4	0				+1	
Nearest Hit (different value)	0	0	8	- 1	- 1	- 1	- 1	- 1
Nearest Miss (different value)	4	4	0	+1	+1	+1	+1	+1
Farthest Hit (different value)	8	8	8			+1		
Farthest Miss (different value)	4	4	8			- 1		

ReliefF Score Total	4	4	-8
SURF Score Total	4	4	-8
SURF* Score Total	8	8	-8
MultiSURF* Score Total	8	8	-8
MultiSURF Score Total	4	4	-8

As Table 4 depicts, all five scoring approaches assign positive scores to A_1 and A_2, and a negative score of -8 to A_3. Thus, all of these methods can differentiating relevant from irrelevant features in the presence of a pure 2-way interaction. However 'far' scoring in SURF* and MultiSURF* make the score difference between a relevant and irrelevant feature score even greater. In the context of 2-way interactions, 'far' scoring serves to reinforce scores of interacting features by offering a larger 'scoring sample size', as originally intended in (Greene et al., 2010). However as this example breakdown also indicates, the scoring approach does not explain performance differences between SURF* and MultiSURF* in detecting 2-way interactions. That is likely the result of MultiSURF*'s target-centric threshold calculation and/or its application of a dead-band zone.

4.2 Main Effects Performance Losses

To understand why 'far' scoring in SURF* and MultiSURF* leads to main effect performance loss, we lay out a conceptual scenario in Table 5. This scenario assumes an example problem with binary features and endpoint. On the far left we have differently labeled instances including a hypothetical target instance (R_i), as well as every other possible type of instance from the perspective of the target instance R_i. Other instance types are differentiated based on properties such as 'Distance', (near or far), 'Class' (hits or misses), and if the relevant feature (i.e. with a main effect) has the same or different value as R_i (instance types with the same feature value are shaded in grey). For simplicity we will assume that all irrelevant features are expected to end up with a feature score of approximately zero as derived in (Kira and Rendell, 1992a).

The key to understanding this conceptual scenario is understanding where the expected frequencies of each instance type come from. To do that, first it's important to define a few expectations of datasets that include a main effect: (1) irrelevant feature values will be randomly distributed, (2) instances that have a relevant feature with the same value between them will tend to be closer on average (assuming the first expectation), and (3) if at least one feature in the dataset is relevant, we expect instances with the same class to be closer on average (since the one relevant feature will typically have the same feature value between instances with the same class value). Based on these expectations we can describe 'likely' combinations of distance with class, and distance with feature value. These include the following; (1) near and same class, (2) near and same feature value, (3) far and different class, and (4) far and different feature value. All other combinations are less likely based on our expectations. Finally we estimate the frequency of an instance 'type' based on the number of 'likely' combinations it includes. Specifically, if it has two likely combinations it is 'high' frequency, if it has two unlikely combinations it is 'low' frequency, and if it has one of each it has 'medium' frequency.

With this in mind, Table 5 lays out the scoring scheme for each algorithm in the right hand columns. First we look at ReliefF, which incidentally has the same scoring scheme as SURF and MultiSURF in the context of this table. Based on the scoring scheme, the only instance types that will contribute to a feature score update are neighbors (i.e. near) with a different feature value than the target (Hit_3 and $Miss_3$), where hits receive -1 and misses receive $+1$ every score update. Therefore over all training instance updates, what total score is expected? For simplicity lets assign our estimated frequencies a numerical value (i.e. high = 3, medium = 2, and low = 1). Thus ReliefF scoring should yield a total of $+1$ for a main effect feature (i.e. medium - low). Since we expect irrelevant features to have a score of approximately 0, we expect ReliefF, as well as SURF and MultiSURF, to be able to distinguish relevant main effect features from irrelevant ones. However, SURF* adds far scoring such that far instances with different feature states also contribute to feature score updates (i.e. Hit_4 and $Miss_4$). If we add up the frequencies we get an estimated feature score of zero (i.e. medium + medium - high - low). Again assuming that irrelevant features will have a score of approximately 0, it makes sense that SURF* is having difficulty

Table 5: Conceptual scenario illustrating why ‘far’ scoring deteriorates main effect performance.

	Instance ‘Type’ Properties					RBA Scoring Schemes				
Instance Type	**Distance**	**Relevant Feature**	**Irrelevant Features**	**Class**	**Frequency**	**ReliefF**	**SURF**	**SURF***	**MultiSURF***	**MultiSURF**
Target (R_i)	-	0	. . .	0	-	-	-	-	-	-
Hit_1	Near	0	. . .	0	High					
$Miss_1$	Near	0	. . .	1	Medium					
Hit_2	Far	0	. . .	0	Medium				-1	
$Miss_2$	Far	0	. . .	1	Low				$+1$	
Hit_3	Near	1	. . .	0	Low	-1	-1	-1	-1	-1
$Miss_3$	Near	1	. . .	1	Medium	$+1$	$+1$	$+1$	$+1$	$+1$
Hit_4	Far	1	. . .	0	Medium			$+1$		
$Miss_4$	Far	1	. . .	1	High			-1		
Total Score						$+1$	$+1$	0	0	$+1$

separating relevant from irrelevant features. The same issue occurs in MultiSURF* that similarly adopts far scoring. However, here ‘far’ instances with the *same* feature values contribute to scoring to save computational time (i.e. Hit_2 and $Miss_2$). If we add up these frequencies we again get an estimated feature score of zero (i.e. medium + low - medium - low). This conceptual illustration explains our findings and we expect this trend to hold for continuous features and endpoints. Notably, it is possible that this far scoring performance loss may be less apparent when the number of irrelevant features becomes very large. When this happens, relevant features have a much smaller influence on the distance between instances. Regardless, we still always expect ReliefF, SURF, and MultiSURF to perform as good or better than SURF* or MultiSURF* with respect to detecting main effects.

Next, we will examine a specific example of a simple main effect datasets given in Table 6. This dataset has the same number of features and instances as given in Table 2. However in this dataset, A_1 has a strong main effect relevant to class, while A_2 and A_3 are irrelevant. Table 7 breaks down nearest and farthest hits and misses in the same way as Table 3. Table 8 presents the summary of the algorithm score contributions in the same way as Table 4. As demonstrated by the score totals in this example problem, the failure of ‘far’ scoring schemes in SURF* and MultiSURF* is the result of farthest miss (different value) or farthest hit (same value) negative score contributions, respectively. While not explicitly tested in this study we predict that the SWRF* (Stokes and Visweswaran, 2012) algorithm (a related RBA), would similarly suffer from main effect performance loss due to it’s adoption of ‘far’ scoring, but this should be explicitly tested in future work.

Table 6: Simple example dataset with a relevant main effect in A_1. Features A_2 and A_3 are irrelevant.

Instances	A_1	A_2	A_3	C
R_1	1	0	1	1
R_2	1	1	0	1
R_3	1	0	1	1
R_4	1	1	0	1
R_5	0	0	1	0
R_6	0	1	0	0
R_7	0	0	1	0
R_8	0	1	0	0

Table 7: Breakdown of near/far, hit/miss, and features with different or same values for each target instance. Illustrates where feature scores come from for the main effect dataset in Table 6.

	Far (Same Value)		Near (Different Value)		Far (Different Value)	
Target	Hit	Miss	Hit	Miss	Hit	Miss
R_1	$R_2(A_1)$,$R_4(A_1)$	None	None	$R_5(A_1)$,$R_7(A_1)$	$R_2(A_2, A_3)$,$R_4(A_2, A_3)$	$R_8(A_1, A_2, A_3)$,$R_6(A_1, A_2, A_3)$
R_2	$R_1(A_1)$,$R_3(A_1)$	None	None	$R_6(A_1)$,$R_8(A_1)$	$R_1(A_2, A_3)$,$R_3(A_2, A_3)$	$R_5(A_1, A_2, A_3)$,$R_7(A_1, A_2, A_3)$
R_3	$R_2(A_1)$,$R_4(A_1)$	None	None	$R_5(A_1)$,$R_7(A_1)$	$R_2(A_2, A_3)$,$R_4(A_2, A_3)$	$R_8(A_1, A_2, A_3)$,$R_6(A_1, A_2, A_3)$
R_4	$R_1(A_1)$,$R_3(A_1)$	None	None	$R_6(A_1)$,$R_8(A_1)$	$R_1(A_2, A_3)$,$R_3(A_2, A_3)$	$R_5(A_1, A_2, A_3)$,$R_7(A_1, A_2, A_3)$
R_5	$R_6(A_1)$,$R_8(A_1)$	None	None	$R_1(A_1)$,$R_3(A_1)$	$R_6(A_2, A_3)$,$R_8(A_2, A_3)$	$R_2(A_1, A_2, A_3)$,$R_4(A_1, A_2, A_3)$
R_6	$R_1(A_1)$,$R_3(A_1)$	None	None	$R_2(A_1)$,$R_4(A_1)$	$R_5(A_2, A_3)$,$R_7(A_2, A_3)$	$R_1(A_1, A_2, A_3)$,$R_3(A_1, A_2, A_3)$
R_7	$R_6(A_1)$,$R_8(A_1)$	None	None	$R_1(A_1)$,$R_3(A_1)$	$R_6(A_2, A_3)$,$R_8(A_2, A_3)$	$R_2(A_1, A_2, A_3)$,$R_4(A_1, A_2, A_3)$
R_8	$R_5(A_1)$,$R_7(A_1)$	None	None	$R_2(A_1)$,$R_4(A_1)$	$R_5(A_2, A_3)$,$R_7(A_2, A_3)$	$R_1(A_1, A_2, A_3)$,$R_3(A_1, A_2, A_3)$

5. Conclusions and Future Study

This work has made a number of contributions with respect to feature selection and RBAs. Specifically we have (1) introduced ReBATE as an open source, user-friendly, and data-type flexible software package for applying a variety of RBAs, (2) designed and applied the most expansive simulation study of RBAs to date, (3) implemented and evaluated strategies to extend RBAs previously only designed for clean data with discrete features and endpoints to a variety of different dataset types, (4) compared the performance of select RBAs to other established feature selection algorithms, (5) identified known or suspected reasons for differences in algorithm performance to guide ongoing RBA development, and (6) introduced and evaluated the MultiSURF algorithm, identifying it to have significant, reliable power in the greatest diversity of dataset configurations.

The results of this study support the following conclusions; (1) existing popular feature selection methods (i.e. chi square, ANOVA, mutual information, ExtraTrees, and RFE ExtraTrees) fail to perform, or perform as competitively on a variety of generalizable problem types (2) RBAs are proficient at detecting 2-way epistatic interactions, but MultiSURF* in particular performs best in this regard, (3) 'far' scoring in RBAs (i.e. SURF* and MultiSURF*) improves the detection of 2-way epistatic interactions (4) 'far' scoring deteriorates or even eliminates the ability of SURF* and MultiSURF* to detect simple main effects (5) RBAs function in the presence of patterns of heterogeneous association, (6) the number of neighbors used in RBA scoring is a critical and problem dependent factor with respect to algorithm success, (7) only RBAs that use fewer neighbors in scoring can detect 3-way

Table 8: Summary of score contributions for main effect dataset in Table 6 yielding ReliefF, SURF, SURF*, MultiSURF*, and MultiSURF scores. Illustrates why 'far' scoring hinders main effect detection.

	Features			RBA Scoring Schemes				
	A_1	A_2	A_3	ReliefF	SURF	SURF*	MultiSURF*	MultiSURF
Farthest Hit (same value)	8	0	0				- 1	
Farthest Miss (same value)	0	0	0				+1	
Nearest Hit (different value)	0	0	0	- 1	- 1	- 1	- 1	- 1
Nearest Miss (different value)	8	0	0	+1	+1	+1	+1	+1
Farthest Hit (different value)	0	8	8			+1		
Farthest Miss (different value)	8	8	8			- 1		

ReliefF Score Total	8	0	0
SURF Score Total	8	0	0
SURF* Score Total	0	0	0
MultiSURF* Score Total	0	0	0
MultiSURF Score Total	8	0	0

interactions (i.e. ReliefF 10 NN, and MultiSURF), (8) all implemented data-type expansions in ReBATE were successful, however performance losses are still observed in datasets with a mix of discrete and continuous features, (9) MultiSURF and ReliefF are the only examined methods that can detect all of the following; main effects, heterogeneity, and 2 or 3-way interactions, (10) the main drawback of ReliefF is that the user has to specify a k parameter which our results indicate can dramatically impact success depending on the noise (e.g. heritability), number of training instances, size of the feature space, the heterogeneity ratio, the dimensionality of the interaction, and/or the amount of missing data, (11) the main drawback of MultiSURF is that it failed to detect 4-way and 5-way interactions, and (12) MultiSURF is the most generally flexible and successful method as well as being easier to apply successfully in contrast with ReliefF.

Furthermore, while the asymptotic time complexity of core RBAs is $\mathcal{O}(n^2 \cdot a)$, the complete time complexity of MultiSURF is slightly less than that for MultiSURF* which is the best RBA for detecting 2-way epistasis. MultiSURF also appears to scale competitively in feature spaces of increasing size.

Beyond MultiSURF, the results of this study strongly suggest that the RBA concept can be improved further. As such we have a number of suggested targets for future research; (1) explicitly compare alternate or novel strategies for handling missing data, regression, and mixed feature types, (2) explore alternative strategies to improve the performance of RBAs in detecting higher dimensional interactions (e.g. 4-way interactions and beyond), (3) given what we have learned in this study, evaluate the impact of instance pair distance weighted scoring similar to strategies proposed in SWRF* and some other core RBAs

(Robnik-Šikonja and Kononenko, 1997; Draper et al., 2003; Sun and Li, 2006; Sun, 2007; Stokes and Visweswaran, 2012), and (4) integrate MultiSURF and other promising RBAs with iterative RBAs to determine the best combination(s) for scaling up to very large feature spaces. In future work we will also seek to expand the diversity of our simulation studies even further and we recommend that other feature selection investigations adopt similar approaches to evaluating/comparing methods as well.

Acknowledgments

Special thanks to Drs. William La Cava and Brian Cole for their constructive feedback. We also thank the Penn Medicine Academic Computing Services for the use of their computing resources. This work was supported by National Institutes of Health grants AI116794, DK112217, ES013508, EY022300, HL134015, LM009012, LM010098, LM011360, TR001263, and the Warren Center for Network and Data Science.

References

Lluıs A Belanche and Félix Fernando González. Review and evaluation of feature selection algorithms in synthetic problems. *arXiv preprint arXiv:1101.2320*, 2011.

Verónica Bolón-Canedo, Noelia Sánchez-Maroño, and Amparo Alonso-Betanzos. A review of feature selection methods on synthetic data. *Knowledge and information systems*, 34 (3):483–519, 2013.

Girish Chandrashekar and Ferat Sahin. A survey on feature selection methods. *Computers & Electrical Engineering*, 40(1):16–28, 2014.

Heather J Cordell. Epistasis: what it means, what it doesn't mean, and statistical methods to detect it in humans. *Human molecular genetics*, 11(20):2463–2468, 2002.

Bruce Draper, Carol Kaito, and José Bins. Iterative relief. In *Computer Vision and Pattern Recognition Workshop, 2003. CVPRW'03. Conference on*, volume 6, pages 62–62. IEEE, 2003.

Margaret J Eppstein and Paul Haake. Very large scale relieff for genome-wide association analysis. In *Computational Intelligence in Bioinformatics and Computational Biology, 2008. CIBCB'08. IEEE Symposium on*, pages 112–119. IEEE, 2008.

Raquel Flórez-López. Reviewing relief and its extensions: a new approach for estimating attributes considering high-correlated features. In *Data Mining, 2002. ICDM 2003. Proceedings. 2002 IEEE International Conference on*, pages 605–608. IEEE, 2002.

George Forman. An extensive empirical study of feature selection metrics for text classification. *Journal of machine learning research*, 3(Mar):1289–1305, 2003.

Pierre Geurts, Damien Ernst, and Louis Wehenkel. Extremely randomized trees. *Machine learning*, 63(1):3–42, 2006.

Delaney Granizo-Mackenzie and Jason H Moore. Multiple threshold spatially uniform relieff for the genetic analysis of complex human diseases. In *European Conference on Evolutionary Computation, Machine Learning and Data Mining in Bioinformatics*, pages 1–10. Springer, 2013.

Casey S Greene, Nadia M Penrod, Jeff Kiralis, and Jason H Moore. Spatially uniform relieff (surf) for computationally-efficient filtering of gene-gene interactions. *BioData mining*, 2 (1):5, 2009.

Casey S Greene, Daniel S Himmelstein, Jeff Kiralis, and Jason H Moore. The informative extremes: using both nearest and farthest individuals can improve relief algorithms in the domain of human genetics. In *European Conference on Evolutionary Computation, Machine Learning and Data Mining in Bioinformatics*, pages 182–193. Springer, 2010.

Isabelle Guyon and André Elisseeff. An introduction to variable and feature selection. *Journal of machine learning research*, 3(Mar):1157–1182, 2003.

Se June Hong. Use of contextual information for feature ranking and discretization. *IEEE transactions on knowledge and data engineering*, 9(5):718–730, 1997.

Nazrul Hoque, DK Bhattacharyya, and Jugal K Kalita. Mifs-nd: a mutual information-based feature selection method. *Expert Systems with Applications*, 41(14):6371–6385, 2014.

Earl B Hunt, Janet Marin, and Philip J Stone. *Experiments in induction.* Academic Press, 1966.

Peyman Jafari and Francisco Azuaje. An assessment of recently published gene expression data analyses: reporting experimental design and statistical factors. *BMC Medical Informatics and Decision Making*, 6(1):27, 2006.

Xin Jin, Anbang Xu, Rongfang Bie, and Ping Guo. Machine learning techniques and chi-square feature selection for cancer classification using sage gene expression profiles. In *International Workshop on Data Mining for Biomedical Applications*, pages 106–115. Springer, 2006.

Alan Jović, Karla Brkić, and Nikola Bogunović. A review of feature selection methods with applications. In *Information and Communication Technology, Electronics and Microelectronics (MIPRO), 2015 38th International Convention on*, pages 1200–1205. IEEE, 2015.

Kenji Kira and Larry A Rendell. The feature selection problem: Traditional methods and a new algorithm. In *AAAI*, volume 2, pages 129–134, 1992a.

Kenji Kira and Larry A Rendell. A practical approach to feature selection. In *Proceedings of the ninth international workshop on Machine learning*, pages 249–256, 1992b.

Igor Kononenko. Estimating attributes: analysis and extensions of relief. In *European conference on machine learning*, pages 171–182. Springer, 1994.

Igor Kononenko and Marko Robnik Šikonja. Non-myopic feature quality evaluation with (r) relieff. *Computational Methods of Feature Selection*, pages 169–191, 2008.

Igor Kononenko, Marko Robnik-Sikonja, and Uros Pompe. Relieff for estimation and discretization of attributes in classification, regression, and ilp problems. *Artificial intelligence: methodology, systems, applications*, pages 31–40, 1996.

Igor Kononenko, Edvard Šimec, and Marko Robnik-Šikonja. Overcoming the myopia of inductive learning algorithms with relieff. *Applied Intelligence*, 7(1):39–55, 1997.

Andy Liaw, Matthew Wiener, et al. Classification and regression by randomforest. *R news*, 2(3):18–22, 2002.

Brett A McKinney, David M Reif, Bill C White, JE Crowe, and Jason H Moore. Evaporative cooling feature selection for genotypic data involving interactions. *Bioinformatics*, 23(16): 2113–2120, 2007.

Nyararai Mlambo, Wilson K Cheruiyot, and Michael W Kimwele. A survey and comparative study of filter and wrapper feature selection techniques. *International Journal of Engineering and Science (IJES)*, 5(8):57–67, 2016.

Jason H Moore and Bill C White. Tuning relieff for genome-wide genetic analysis. In *European Conference on Evolutionary Computation, Machine Learning and Data Mining in Bioinformatics*, pages 166–175. Springer, 2007.

Jason H Moore and Scott M Williams. Traversing the conceptual divide between biological and statistical epistasis: systems biology and a more modern synthesis. *Bioessays*, 27(6): 637–646, 2005.

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

Marylyn D Ritchie, Lance W Hahn, Nady Roodi, L Renee Bailey, William D Dupont, Fritz F Parl, and Jason H Moore. Multifactor-dimensionality reduction reveals high-order interactions among estrogen-metabolism genes in sporadic breast cancer. *The American Journal of Human Genetics*, 69(1):138–147, 2001.

Marylyn D Ritchie, Lance W Hahn, and Jason H Moore. Power of multifactor dimensionality reduction for detecting gene-gene interactions in the presence of genotyping error, missing data, phenocopy, and genetic heterogeneity. *Genetic epidemiology*, 24(2):150–157, 2003.

Marko Robnik-Šikonja. Experiments with cost-sensitive feature evaluation. In *European Conference on Machine Learning*, pages 325–336. Springer, 2003.

Marko Robnik-Šikonja and Igor Kononenko. An adaptation of relief for attribute estimation in regression. In *Machine Learning: Proceedings of the Fourteenth International Conference (ICML97)*, pages 296–304, 1997.

Marko Robnik-Šikonja and Igor Kononenko. Comprehensible interpretation of reliefs estimates. In *Machine Learning: Proceedings of the Eighteenth International Conference on Machine Learning (ICML2001), Williamstown, MA, USA. San Francisco: Morgan Kaufmann*, pages 433–40, 2001.

Marko Robnik-Šikonja and Igor Kononenko. Theoretical and empirical analysis of relieff and rrelieff. *Machine learning*, 53(1-2):23–69, 2003.

Marko Robnik-Šikonja, David Cukjati, and Igor Kononenko. Comprehensible evaluation of prognostic factors and prediction of wound healing. *Artificial intelligence in medicine*, 29 (1):25–38, 2003.

Yvan Saeys, Iñaki Inza, and Pedro Larrañaga. A review of feature selection techniques in bioinformatics. *bioinformatics*, 23(19):2507–2517, 2007.

Matthew E Stokes and Shyam Visweswaran. Application of a spatially-weighted relief algorithm for ranking genetic predictors of disease. *BioData mining*, 5(1):20, 2012.

Yijun Sun. Iterative relief for feature weighting: algorithms, theories, and applications. *IEEE transactions on pattern analysis and machine intelligence*, 29(6), 2007.

Yijun Sun and Jian Li. Iterative relief for feature weighting. In *Proceedings of the 23rd international conference on Machine learning*, pages 913–920. ACM, 2006.

Jiliang Tang, Salem Alelyani, and Huan Liu. Feature selection for classification: A review. *Data Classification: Algorithms and Applications*, page 37, 2014.

Tricia A Thornton-Wells, Jason H Moore, and Jonathan L Haines. Dissecting trait heterogeneity: a comparison of three clustering methods applied to genotypic data. *BMC bioinformatics*, 7(1):204, 2006.

Ryan J Urbanowicz and Will N Browne. *Introduction to Learning Classifier Systems.* Springer, 2017.

Ryan J Urbanowicz and Jason H Moore. The application of michigan-style learning classifiersystems to address genetic heterogeneity and epistasisin association studies. In *Proceedings of the 12th annual conference on Genetic and evolutionary computation*, pages 195–202. ACM, 2010.

Ryan J Urbanowicz and Jason H Moore. Exstracs 2.0: description and evaluation of a scalable learning classifier system. *Evolutionary intelligence*, 8(2-3):89–116, 2015.

Ryan J Urbanowicz, Delaney Granizo-Mackenzie, and Jason H Moore. Using expert knowledge to guide covering and mutation in a michigan style learning classifier system to detect epistasis and heterogeneity. In *International Conference on Parallel Problem Solving from Nature*, pages 266–275. Springer, 2012a.

Ryan J Urbanowicz, Jeff Kiralis, Jonathan M Fisher, and Jason H Moore. Predicting the difficulty of pure, strict, epistatic models: metrics for simulated model selection. *BioData mining*, 5(1):15, 2012b.

Ryan J Urbanowicz, Jeff Kiralis, Nicholas A Sinnott-Armstrong, Tamra Heberling, Jonathan M Fisher, and Jason H Moore. Gametes: a fast, direct algorithm for generating pure, strict, epistatic models with random architectures. *BioData mining*, 5(1): 16, 2012c.

Ryan J Urbanowicz, Angeline S Andrew, Margaret Rita Karagas, and Jason H Moore. Role of genetic heterogeneity and epistasis in bladder cancer susceptibility and outcome: a learning classifier system approach. *Journal of the American Medical Informatics Association*, 20(4):603–612, 2013.

Ryan J Urbanowicz, Gediminas Bertasius, and Jason H Moore. An extended michigan-style learning classifier system for flexible supervised learning, classification, and data mining. In *International Conference on Parallel Problem Solving from Nature*, pages 211–221. Springer, 2014.

Ryan J Urbanowicz, Melissa Meeker, Randal S Olson, William LaCava, and Jason H Moore. Relief-based feature selection: introduction and review. *Journal of Machine Learning Research*, Submitted.

Haleh Vafaie and Ibrahim F Imam. Feature selection methods: genetic algorithms vs. greedy-like search. In *Proceedings of the International Conference on Fuzzy and Intelligent Control Systems*, volume 51, 1994.

Jorge R Vergara and Pablo A Estévez. A review of feature selection methods based on mutual information. *Neural Computing and Applications*, 24(1):175–186, 2014.

Ian H Witten, Eibe Frank, Mark A Hall, and Christopher J Pal. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, 2016.

Lei Yu and Huan Liu. Efficient feature selection via analysis of relevance and redundancy. *Journal of machine learning research*, 5(Oct):1205–1224, 2004.

Zhaohui Zheng, Xiaoyun Wu, and Rohini Srihari. Feature selection for text categorization on imbalanced data. *ACM Sigkdd Explorations Newsletter*, 6(1):80–89, 2004.