**High Dimensional Selection with Interactions Algorithm on Feature Selection for Binary Outcome**

by

**Ziqian Zhuang** supervised by **Wei Xu**

**Rahi Jain**

Toronto, Ontario, Canada April 26, 2021

**Contents**

1. [Introduction](#_bookmark0) 4
2. [Methods](#_bookmark1) 7
   1. [HDSI-BO Algorithm](#_bookmark2) 7
   2. [Hyperparameter Optimization](#_bookmark4) 10
3. [Results](#_bookmark6) 13
   1. [Simulation Studies](#_bookmark7) 13
      1. [Simulation Settings](#_bookmark8) 13
      2. [Simulation Results](#_bookmark10) 14
   2. [Real Data Studies](#_bookmark13) 15
      1. [Real Dataset Summary](#_bookmark14) 15
      2. [Results](#_bookmark16) 17
4. [Conclusion and Discussion](#_bookmark19) 19

[Appendix](#_bookmark20) 24

**Acknowledgments References**

**Abstract**

Feature selection on high-dimensional data with interaction ef- fects is a challenging problem. Although most commonly used al- gorithms, for example, lasso-typed algorithms, can handle high di- mensional data, they do not consider interaction effects. To overcome this drawback, a high dimensional Selection with Interactions (HDSI) algorithm was proposed to incorporate interaction terms and existing techniques can be combined with HDSI to do high dimensional data feature selection. The purpose of our work is to extend the HDSI al- gorithm to binary outcome data (HDSI-BO) as the HDSI algorithm is only applied and assessed in continuous outcome data.

To modify the HDSI algorithm to fit for binary outcome data fea-

ture selection, proper performance measurements and feature selection criteria have been specially designed. Simulation and real data studies were carried out to assess the performance of the HDSI-BO algorithm. In the studies, the algorithm has been combined with standard logis- tic regression, lasso, ridge regression, elastic net, and adaptive lasso to verify its applicability. The feature selection and predictive perfor- mance of these HDSI-BO methods were compared with the standard methods.

Results of simulation and real data studies confirm the applicabil-

ity of the HDSI-BO algorithm. Furthermore, the HDSI-BO methods have a better ability to select true marginal features and interaction terms and consequently, have the predictive ability with higher levels of accuracy.

*keywords*: Feature selection; High dimensional data; Interaction terms; HDSI.

# Introduction

Feature selection is fundamental to high dimensional data problems, for im- proving the efficiency of data modelings and the predictive ability of models. The growth of data both in sample size (*n*) and feature dimension (*p*) has posed various challenges. There could be a large number of features in the original data that are not related to the target research questions. Remov- ing these features, commonly known as noisy features, by feature selection approaches before fitting final models can efficiently improve the precision of estimation on feature effects, measured by model coefficients, and the model prediction accuracy.

Many feature selection strategies have been proposed. The most primary way is to rely on the domain knowledge or experience of experts to shortlist significant variables(Heinze, Wallisch and Dunkler, 2018). But it’s very sub- jective and it will be a big challenge for experts if the feature dimension is high. Subset selection approaches based on standard linear regression only work when *p < n*. Penalized regression approaches such as lasso, ridge, etc. can achieve feature selection regardless of data dimensions through their own penalty principles or artificially predefined threshold values. However, they have their limitations. lasso has built-in feature selection but cannot select features more than *n* (Emmert-Streib and Dehmer, 2019). When there are highly correlated feature groups, lasso tends to randomly select only one fea- ture and shrink the coefficients of other correlated features to 0 (Zou and

Hastie, 2005). Ridge regression can solve the multicollinearity problem by shrinking some coefficient estimates close to 0 but not exactly 0, but doing so means ridge regression loses the ability to do feature selection. Elastic net break through the limitation of lasso and ridge regression. It supports group selection and can select more than *n* features (Zou and Hastie, 2005). Nonetheless, it inevitably selects some noise variables. As an extension of the above lasso-type approaches, adaptive lasso has nicer properties. However, its performance heavily depends on the accuracy of the OLS estimates of weights (Kang and Guo, 2009). As a result, adaptive lasso may suffer from multicollinearity problems, which lead to estimation bias.

Moreover, marginal features might not contain the complete information we need. Hence taking interaction terms into account to fit models sometimes becomes necessary. However, traditionally, up until the development of model fitting algorithms, most algorithms do not consider interaction terms directly despite interaction terms being added to models as necessary(Hahn, Ritchie and Moore, 2003).

To circumvent the problems mentioned above as well as improve the ac- curacy of feature selection, researchers proposed a new algorithm, named High Dimensional Selection with Interactions (HDSI) algorithm for feature selection (Jain and Xu, 2021). HDSI is a flexible framework that can be combined with multiple existing statistical techniques as well as incorporate interaction terms to do feature selection for high dimensional data. HDSI has been combined with several standard techniques, including simple linear

regression, lasso, and adaptive lasso, and applied to data with a continuous outcome feature. Simulation studies and real data studies have been carried out to compare the performance of HDSI and these standard techniques. In general, HDSI outperforms the commonly used standard algorithms in the ability to select true features and the prediction precision.

To expand the application of HDSI, we propose the HDSI-BO algorithm based on HDSI to handle binary outcome data. We design proper perfor- mance measurements and feature selection principles to fit for binary out- come data. Further, we combine HDSI-BO with multiple techniques such as logistic regression, lasso, ridge regression, elastic net and adaptive lasso to prove the wide applicability of the algorithm.

Section [section 2](#_bookmark1) introduces the algorithm of HDSI and the performance measurements and feature selection criteria used in HDSI for binary outcome data. In section [section 3,](#_bookmark6) we introduce the simulation studies and real data studies that compare the performance of HDSI with multiple standard meth- ods. and the results of the simulation studies are presented and interpreted. Finally, we conclude the research with a discussion in [section 4.](#_bookmark19)

# Methods

## HDSI-BO Algorithm

HDSI-BO algorithm, developed based on HDSI algorithm, inherits the idea of incorporating interaction terms and flexibility in combination with multiple algorithms. Furthermore, different performance measurements and feature selection criteria are specially designed for binary outcome data. The algo- rithm is summarized in Figure [1](#_bookmark3).

For original data with sample size *n* and feature dimension *p*, suppose *ω*-level interaction terms are considered, the algorithm of HDSI-BO is as follows:

* + 1. **Bootstrapping**: bootstrap *B* samples with size *n* from the original

dataset. For each bootstrap sample, randomly select *q*(*q < p*) features from the original feature space. Now the term set for one bootstrap sample has *q*

marginal features and Σ

*k*=2

*ω*.*q*Σ interaction term.

* + 1. **Modeling**: fit a model for bootstrap sample *i,i* = 1*, . . . ,B* to estimate

*k*

coefficients *βij,j* = 1*, . . . ,p,p* + 1*, . . . ,* Σ*ω* .*p*Σ. The coefficients of unselected

*k*=1

*k*

features for each bootstrap sample are considered missing.

* + 1. **Performance measuring**:
       1. compute (*a ,*1 *− a* ) quantile intervals for the coefficient estimates of

2

2

all marginal and interaction terms, where *a ∈* (0*,*1) is a hyperparameter.

* + - 1. compute minimum AUC for each marginal and interaction term based on all bootstrap models that include this term. For term *j,j* =

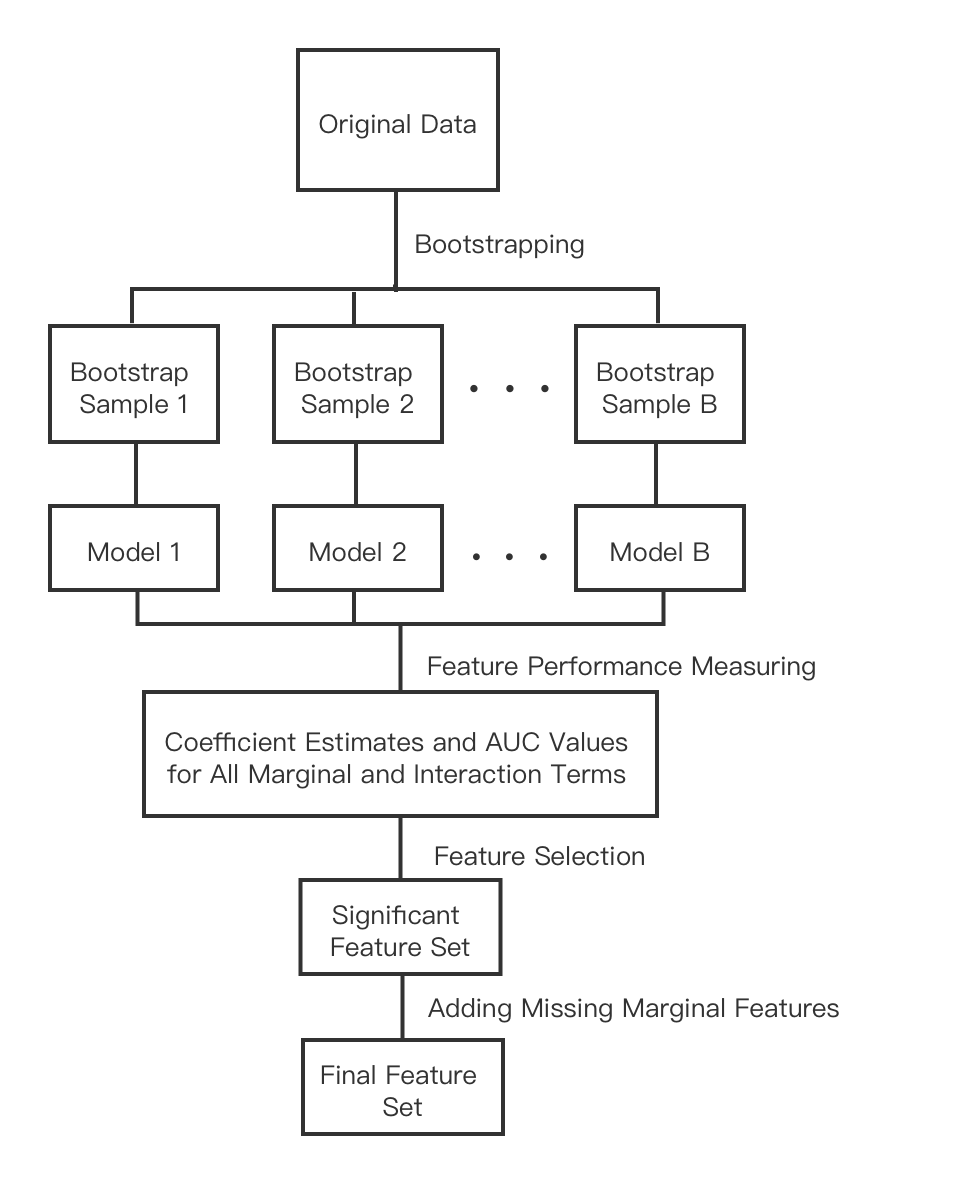


Figure 1: Algorithm of HDSI-BO.

1*, . . . ,p,p* + 1*, . . . ,* Σ

*k*=1

*ω*.*p*Σ,

minAUC*j* = minAUC*ij | i ∈ {*1*, . . . , B}*

*k*

Note that the coefficients and AUC values of unselected features for each bootstrap sample are considered missing. The missing values are dropped during the performance measuring step.

* + 1. **Feature selection**: select marginal or interaction term *j* if
       1. the quantile interval does not include 0.
       2. minAUC*j > µ*minAUC + *bσ*minAUC, where

Σ

*p*

*µ*minAUC = minAUC*j/p*

*j*=1

.Σ

*σ*minAUC =

*p j*=1

(minAUC*j − µ*minAUC)2 *p −* 1

*b ∈* (*−∞,∞*) is a hyperparameter.

* + 1. **Adding missing marginal features**: Since it’s unnatural if a marginal feature is absent in a model while its interaction is present, we add these missing marginal features to obtain a final complete feature set.

Note that any modeling technique, with or without built-in feature se- lection, can be adopted to model bootstrap samples. This flexibility enables HDSI-BO to combine with multiple techniques such as simple logistic regres- sion, lasso, etc.

## Hyperparameter Optimization

In the HDSI-BO algorithm, there are four hyperparameters whose values need to be predetermined:

* + 1. number of bootstrap samples *B*;
    2. number of features for each bootstrap sample *q*;
    3. hyperparameter *a* used to determine the bound of quantile intervals;
    4. hyperparameter *b* used to determine the threshold value for minAUC.

Since the performance of a marginal or interaction term is measured based on its coefficient estimates and values of AUC from different bootstrap models that include this term, it is important to ensure that every term is sampled sufficient times.

HDSI-BO supposes the mean of coefficients are zero and then compares the estimated mean values from bootstrap models with zero. In this case, the minimum number of times a term should be selected *L* can be determined based on Lehr’s equation (Lehr, 2010),

*L* = 8*/*∆2

where ∆ is the effect size. According to Cohen’s rule of thumb (Cohen, 2013), ∆ = 0*.*2 indicates a small, 0.5 a moderate, and 0.8 a big effect. Smaller effect sizes lead to higher values of *L* and thus lead to more accurate mean estimates of coefficients.

Suppose *ω*-level interaction terms are considered, then the probability of

a term to be included in a sample set with *q* features is

*ω ω*

*ρ* = Σ .*q*Σ*/* Σ .*p*Σ

*k*=2

*k*=2

*k*

*k*

The *B* bootstrap samples can be viewed as *B* trials. Then the times of a term to be selected in *B* trials *X ∼* Binomial(*B,ρ*) and the probability of one term to be selected at least *L* times can be calculated as

*L−*1

Σ

Pr(*X ≥ L*) = 1 *−* Pr(*X* = *m*) = *f* (*B,q,L*)

*m*=0

The probability should be higher enough to ensure almost every term can be selected at least *L* times. Therefore, the value of *B* is dependent on *q* and the effect size if a threshold value of this probability is given.

The optimal values of hyperparameters that can achieve the best accuracy of feature selection and best predictive performance of final models vary with datasets. Therefore, a suitable hyperparameter selection method is needed to determine the optimal values of hyperparameters for given data before the HDSI-BO algorithm can be applied.

Given that there are multiple hyperparameters to be optimized and the distribution of a model’s predictive performance based on these hyperparam- eters is completely unknown, we choose to use the genetic algorithm for the optimization of (*q,a,b*) in this study (Thede and Scott, 2004). The genetic algorithm mimics the operation of evolution. It generates new combina- tions of values through mutation and cross-over and searches for the optimal value combinations of hyperparameters that can optimize the performance

of HDSI-BO. To measure the performance of HDSI-BO quantitatively, we fit a logistic regression model based on the selected feature set obtained from HDSI-BO in the training dataset and use the model’s AUC in the test dataset as a measure. For the simulation and real data studies below, we use five- fold genetic algorithms with cross-validation. [Figure 2](#_bookmark5) depicts the complete process of applying HDSI-BO in this study.

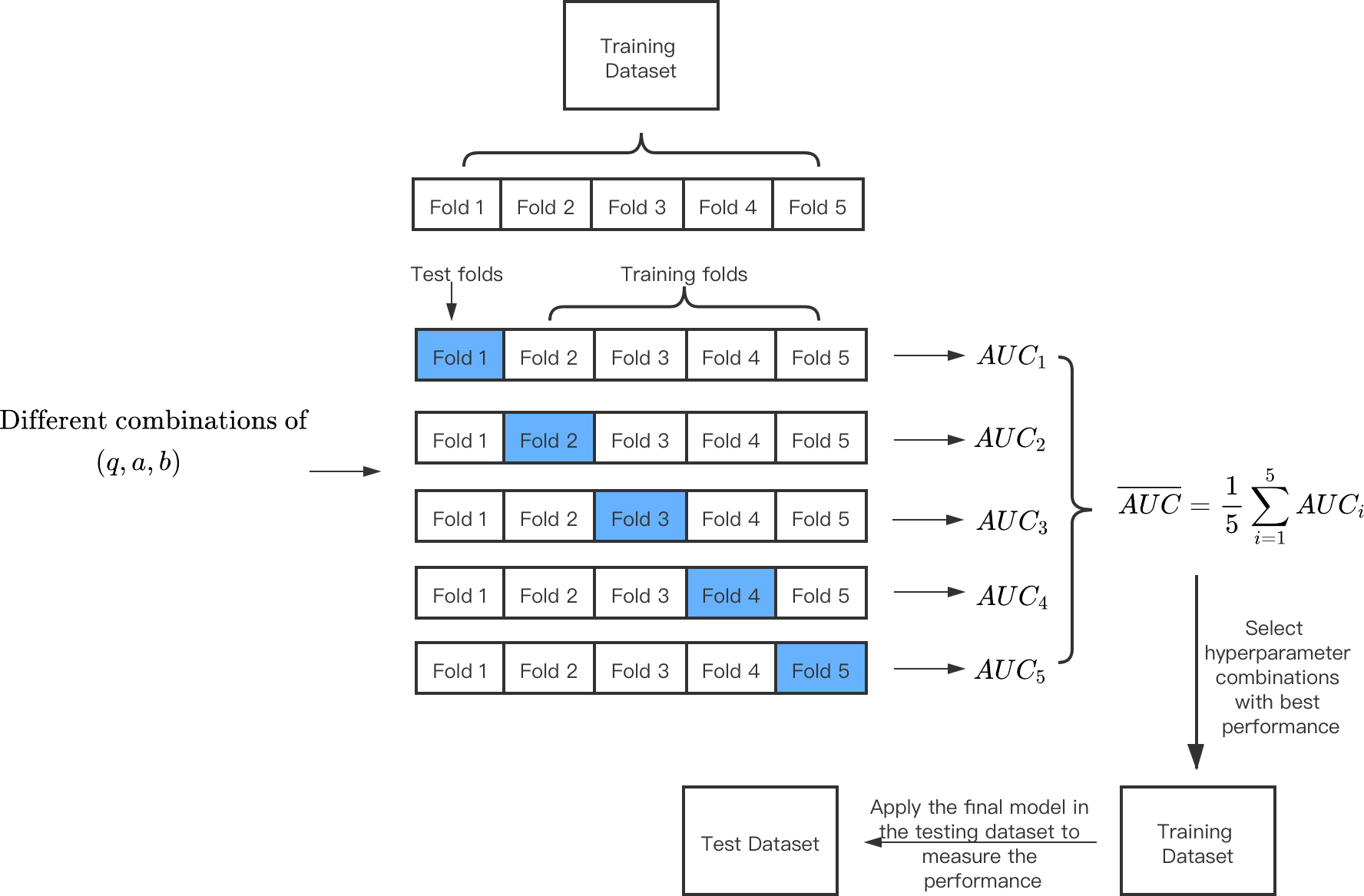


Figure 2: The process of hyperparameter optimization with five-fold genetic algorithm with cross-validation, applying HDSI-BO with selected optimal values of hyperparameters, and finally validating performance.

# Results

## Simulation Studies

### Simulation Settings

To verify the applicability of HDSI-BO in combining with different methods, we combine HDSI-BO with simple logistic regression, lasso, ridge regression, elastic net, adaptive lasso, and carry out simulation studies with two scenar- ios to compare the predictive performance of HDSI-BO with these methods.

For each scenario, we generate a dataset with a total size of 1000 from a regression model

logit*y* = *β*0 + *β*1*x*1 + *· · ·* + *βpxp* + *β*12*x*1*x*2 + *· · ·* + *s*

where *x*1*, · · · ,xp ∼ N* (0*,*1),*s ∼ N* (0*,*0*.*01).

The sizes of the training set and the testing set are both 300. Strati- fied sampling is used to ensure that two possible outcomes have equivalent numbers of cases in both sets. For the sake of simplicity, we only take two- way interaction terms into account in the simulation studies although the algorithm itself can incorporate higher-level interactions. The settings of coefficients and feature dimensions are summarized in [Table 1.](#_bookmark9)

Table 1: The settings of coefficients and feature dimensions of two simulation scenarios. *βj* is the coefficient of *Xj* and *βij* is the two-way interaction term of *Xi* and *Xj*. Coefficients not shown in the table are all zero, which indicates the corresponding features are noisy features.

Scenario Feature Dimension *p* Non-zero Coefficients

1 25 (*β*0*,β*1*,β*2*,β*3*,β*4*,β*12*,β*34) = (1*.*8*,*0*.*5*,*0*.*4*, −* 0*.*4*,*0*.*45*,*0*.*6*, −* 0*.*6)

2 25 (*β*0*,β*1*,β*2*,β*3*,β*4*,β*12*,β*34) = (2*.*0*,*0*.*7*,*0*.*6*, −* 0*.*6*,*0*.*65*,*0*.*8*, −* 0*.*9)

A covariance matrix is designed to add multicollinearity among true and noisy features,

 *x x x x . . x x · · ·* 

 1 0*.*3 0*.*3 0*.*6 0*.*6 0 0 *.* 0 

1 1 1 2 1 7

 *x*2*x*1 *x*2*x*2 *. . x*2*x*7 *· · ·*





.

0*.*3 1 0*.*3 0*.*2 0*.*1 0 0 *.* 0

 0*.*3 0*.*3 1 0*.*2 0*.*1 0 0 *.* 0

 

 

. .

 . .



*x*6*x*1 *x*6*x*2 *· · x*6*x*7 *· · ·* = 0*.*6 0*.*1 0*.*1 0*.*1 1 0 0 *.* 0



 *· · · · ·*

. . .

. . .

. . *· ·* 

0*.*6 0*.*2 0*.*2 1 0*.*1 0 0 *.* 0

 

 0 0 0 0 0 1 0*.*1 *.* 0





*x*7*x*1 *x*7*x*2 *x*7*x*7





*· · · · · · · ·*

*xpx*1 *· · · · · · ·*

0 0 0 0 0 0*.*1 1 *.* 0

*. . . . . . . .* 0 0 0 0 0 0 0 0 *.* 1

For both sceanrios, the effect size used for HDSI-BO is large. We adopt genetic algorithm with five-fold cross-validation to optimize hyperparameters (*q,a,b*) in the training sets and the hyperparameter combination with best mean predictive performance is selected. To measure the predictive perfor- mance of different techniques, we use simple logistic regression to fit final models after obtaining sets of selected features and then use AUC of the final models in the testing sets as a measure.

All statistical analysis was performed using software R 4.0.3 in Linux.

Sample R codes are provided in the Appendix [A](#_bookmark20)

### Simulation Results

The feature selection performance of different methods is presented in [Ta-](#_bookmark11) [ble 2.](#_bookmark11) In both scenarios, the five statistical techniques can select all true marginal and interaction terms after being combined with HDSI-BO, which outperforms these standard techniques alone. Note that the standard logistic

regression model cannot include any interaction terms since it cannot work when *p > n* while HDSI-BO enables it to work under the same condition. Be- sides, although some methods combined with HDSI-BO tend to select more noisy terms, but it does not hurt the predictive performance of final models. [Table 3](#_bookmark12) shows that methods combined with HDSI-BO all have significantly higher AUC compared with standard methods.

Table 2: Feature selection performance of logistic regression (Reg), lasso, ridge regression (Ridge), elastic net, adaptive lasso alone and after being combined with HDSI-BO.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Scenario | | 1 | |  | | | | 2 | |
| Number of Selected Features | | Mar | ginal | Interaction | | Marginal | | Interaction | |
| True(4) | Noisy(21) | True(4) | Noisy(298) | True(2) | Noisy(21) | True(2) | Noisy(298) |
|  | Reg | 4 | 21 | 0 | 0 | 4 | 21 | 0 | 0 |
|  | Lasso | 2 | 0 | 0 | 1 | 1 | 0 | 1 | 0 |
| Standard | Ridge | 4 | 21 | 2 | 296 | 4 | 21 | 2 | 298 |
|  | Elastic Net | 2 | 0 | 1 | 12 | 3 | 0 | 1 | 31 |
|  | Adaptive Lasso | 3 | 3 | 1 | 72 | 3 | 0 | 1 | 16 |
|  | Reg | **4** | 2 | **2** | 2 | **4** | 0 | **2** | 0 |
|  | Lasso | **4** | 7 | **2** | 8 | **4** | 6 | **2** | 7 |
| HDSI-BO | Ridge | **4** | 4 | **2** | 5 | **4** | 2 | **2** | 4 |
|  | Elastic Net | **4** | 8 | **2** | 8 | **4** | 4 | **2** | 5 |
|  | Adaptive Lasso | **4** | 14 | **2** | 18 | **4** | 13 | **2** | 21 |

Table 3: AUC performance of different methods in simulation studies

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Scenario | Reg | Lasso | Ridge | Standard  Elastic Net | Adaptive Lasso | Reg | Lasso | Ridge | HDSI-BO  Elastic Net | Adaptive Lasso |
| 1 | 0.686 | 0.683 | 0.669 | 0.688 | 0.657 | 0.758 | 0.755 | 0.775 | 0.765 | 0.756 |
| 2 | 0.732 | 0.724 | 0.677 | 0.705 | 0.696 | 0.834 | 0.816 | 0.825 | 0.834 | 0.834 |

## Real Data Studies

### Real Dataset Summary

We combine the HDSI-BO methods with multiple model fitting techniques and compare their performance with these standard techniques on three real- world datasets. For simplicity, we only take continuous features that do not

have many missing values into account.

For every dataset, we incorporate all two-way interaction terms. A large effect size is used for estimating *B*. We adopt the same performance mea- surement as the simulation studies. In addition, 30 trials are repeated for each method in one dataset to generate a range of the number of selected features and 95% confidence interval for AUC.

Dataset I is from National Social Life, health and Aging Project (NSHAP) dataset for Wave 3 (2015-2016) (available at [https://www.icpsr.umich.](https://www.icpsr.umich.edu/icpsrweb/NACDA/studies/36873) [edu/icpsrweb/NACDA/studies/36873](https://www.icpsr.umich.edu/icpsrweb/NACDA/studies/36873)). The original dataset contains data

on 1470 features related to health, social life, and well-being for 4377 older American residents. The original outcome feature is continuous, we convert it into binary by categorizing values above-median as 1 and below-median as 0. Dataset I used for the study has sample size *n* = 1633 and marginal feature dimension *p* = 20.

Dataset II is from Community Health Status Indicators (CHSI) dataset (available at [https://healthdata.gov/dataset/community-health-sta](https://healthdata.gov/dataset/community-health-status-indicators-chsi-combat-obesity-heart-disease-and-cancer) [tus-indicators-chsi-combat-obesity-heart-disease-and-cancer](https://healthdata.gov/dataset/community-health-status-indicators-chsi-combat-obesity-heart-disease-and-cancer)) which

contains USA county-level data on various demographics and health parame- ters. The original dataset contains data on 578 features for 3141 US counties. We convert the self-rated health status score variable into binary by catego- rizing values above-median as 1 and below-median as 0 for the study. The final dataset we use has a sample size and feature size of 1470 and 29, re- spectively.

Dataset III is the Breast Cancer Coimbra dataset (available at [https:](https://archive.ics.uci.edu/ml/datasets/Breast%2BCancer%2BCoimbra)

[//archive.ics.uci.edu/ml/datasets/Breast+Cancer+Coimbra](https://archive.ics.uci.edu/ml/datasets/Breast%2BCancer%2BCoimbra)). It con- tains 9 quantitative predictors, which are anthropometric data and param- eters gathered in routine blood analysis and a binary dependent variable, indicating the presence or absence of breast cancer. The sample size is 116.

[Table 4](#_bookmark15) presents the summary of these real datasets. Each dataset is divided into training and test sets.

Table 4: Summary of real datasets

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | | Total | Train | Test |  |
| I | Height | 1633 | 1306 | 327 | 20 |
| II | Health status | 1470 | 1176 | 294 | 29 |
| III | Breast cancer | 116 | 93 | 23 | 9 |

Dataset Outcome feature Sample size(n) Marginal features(p)

### Results

[Table 5](#_bookmark17) and [Table 6](#_bookmark18) summarize the results of real data studies. In most cases, the HDSI-BO methods select less marginal and interaction terms while the mean values of AUC of the HDSI-BO methods are close or higher than those of the standard methods, which indicates that the HDSI-BO methods have comparable or better predictive ability compared with existing standard methods.

Table 5: Feature selection performance of different methods. Ranges are generated based on 30 trials.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Method | | Dataset | | | | | |
| I | |  | II |  | III |
| Marginal(20) | Interaction(190) | Marginal(29) | Interaction(406) | Marginal(9) | Interaction(36) |
|  | Reg | 20(20-20) | 19(190-190) | 29(29-29) | 406(406-406) | 9(9-9) | 36(36-36) |
|  | Lasso | 19(16-20) | 28(11-64) | 11(4-16) | 9(1-30) | 4(1-6) | 6(1-13) |
| Standard | Ridge | 20(20-20) | 190(190-190) | 29(29-29) | 406(406-406) | 9(9-9) | 36(36-36) |
|  | Elastic Net | 20(16-20) | 65(8-152) | 18(13-20) | 42(0-106) | 6(3-9) | 12(3-32) |
|  | Adaptive Lasso | 18(15-20) | 24(9-83) | 14(5-20) | 11(0-43) | 4(2-6) | 4(2-8) |
|  | Reg | 14(8-16) | 12(7-16) | 15(10-21) | 15(8-26) | 5(2-7) | 3(1-5) |
|  | Lasso | 19(15-20) | 24(16-30) | 16(8-23) | 17(8-26) | 7(3-9) | 7(2-9) |
| HDSI-BO | Ridge | 14(10-16) | 18(14-23) | 22(18-26) | 28(17-36) | 4(2-6) | 3(1-4) |
|  | Elastic Net | 16(12-19) | 21(13-28) | 17(14-24) | 16(10-21) | 5(2-9) | 4(1-9) |
|  | Adaptive Lasso | 18(13-20) | 28(19-39) | 8(3-14) | 6(2-13) | 6(4-9) | 4(2-8) |

Table 6: AUC performance of different methods. The values presented in the table are the average and 95% confidence intervals of AUC over 30 trials.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Method | | AUC (95% CI)  I II | | III |
|  | Reg | 0.748(0.680-0.817) | 0.940(0.919-0.961) | 0.648(0.481-0.814) |
|  | Lasso | 0.785(0.746-0.824) | 0.925(0.903-0.946) | 0.784(0.537-1) |
| Standard | Ridge | 0.747(0.706-0.788) | 0.911(0.891-0.945) | 0.777(0.585-0.969) |
|  | Elastic Net | 0.783(0.739-0.828) | 0.930(0.907-0.953) | 0.761(0.531-0.990) |
|  | Adaptive Lasso | 0.786(0.740-0.833) | 0.927(0.905-0.949) | 0.793(0.588-0.998) |
|  | Reg | 0.799(0.749-0.834) | 0.925(0.893-0.956) | 0.810(0.625(0.995) |
|  | Lasso | 0.802(0.756-0.847) | 0.930(0.905-0.955) | 0.792(0.609-0.974) |
| HDSI-BO | Ridge | 0.793(0.745-0.841) | 0.930(0.904-0.956) | 0.793(0.610-0.976) |
|  | Elastic Net | 0.797(0.750-0.844) | 0.931(0.909-0.953) | 0.782(0.591-0.972) |
|  | Adaptive Lasso | 0.800(0.758-0.842) | 0.919(0.887-0.951) | 0.787(0.576-0.997) |

# Conclusion and Discussion

In this paper, we introduced a new high-dimensional data feature selection algorithm for binary outcome data (HDSI-BO). It takes interaction terms into account and can incorporate with existing modeling techniques.

Based on the results of the simulation studies, the HDSI-BO methods can select more true features than the standard methods. Although the HDSI-BO methods selected more noisy features than standard methods in some cases, the values of AUC for the HDSI-BO methods are still higher than those of the standard methods in most cases. It indicates that these noisy features do not significantly damage the predictive ability of the models. Therefore, it is acceptable that HDSI-BO methods select some noisy features. The real data studies showed similar results in terms of the predictive ability of the HDSI-BO methods. In summary, the HDSI-BO algorithm, combined with the statistical techniques used in the simulation and real data studies, performs better in selecting true features and consequently, in prediction accuracy compared with the standard techniques.

In addition to the potential of HDSI-BO in doing feature selection and prediction, the current studies also show limitations. First, the existing fea- ture selection criteria can be modified and improved to reduce the selection of noisy features. Second, AUC, as the single metric of performance, cannot comprehensively measure the feature selection performance of the HDSI-BO methods. Finally, the HDSI-BO algorithm has not been combined with non-

linear methods such as random forest, support vector machine. Such exten- sion is important to further verify the applicability. Future work can focus on overcoming the limitations mentioned above and extending the applications of HDSI-BO.

# Acknowledgments

I would like to express my sincere gratitude to my practicum supervisor Dr. Wei Xu and co-supervisor Dr. Rahi Jain for the continuous support during my practicum. Their guidance helped me in all the time of research and writing of this report.

I am also grateful to Professor Tony Panzarella and all professors who contributed towards the practicum course. Because of them, I was able to work on this project and make it good and enjoyable experience.

**References**

1. Jacob Cohen. *Statistical power analysis for the behavioral sciences*. Aca- demic press, 2013.
2. Frank Emmert-Streib and Matthias Dehmer. High-dimensional lasso- based computational regression models: regularization, shrinkage, and selection. *Machine Learning and Knowledge Extraction*, 1(1):359–383, 2019.
3. Lance W Hahn, Marylyn D Ritchie, and Jason H Moore. Multifactor dimensionality reduction software for detecting gene–gene and gene– environment interactions. *Bioinformatics*, 19(3):376–382, 2003.
4. Georg Heinze, Christine Wallisch, and Daniela Dunkler. Variable selection–a review and recommendations for the practicing statistician. *Biometrical journal*, 60(3):431–449, 2018.
5. Rahi Jain and Wei Xu. HDSI: High dimensional selection with in- teractions algorithm on feature selection and testing. *PloS one*, 16(2):e0246159, 2021.
6. Jian Kang and Jian Guo. Self-adaptive lasso and its bayesian estimation.

Technical report, Working Paper, 2009.

[7]R. Lehr. Sixteen s-squared over d-squared: a relation for crude sample size estimates. *Statistics in Medicine*, 11(8):1099–1102, 2010.

1. Petro Liashchynskyi and Pavlo Liashchynskyi. Grid search, random search, genetic algorithm: A big comparison for nas. *arXiv preprint arXiv:1912.06059*, 2019.
2. Scott M Thede. An introduction to genetic algorithms. *Journal of Computing Sciences in Colleges*, 20(1):115–123, 2004.
3. Hui Zou and Trevor Hastie. Regularization and variable selection via the elastic net. *Journal of the royal statistical society: series B (statistical methodology)*, 67(2):301–320, 2005.

**Appendix A**

Below is the code for running genetic algorithm to optimize hyperparameters.

1 #Setworkingd i r e c t o r y

3 #Loadpackagesandf u n c t i o n s l i b r a r y( memoise )

5 l i b r a r y( glmnet ) l i b r a r y(ROCR)

7 l i b r a r y( Rmisc ) l i b r a r y(GA)

9 l i b r a r y( mosaic )

l i b r a r y( p a r a l l e l )

11 l i b r a r y( d o P a r a l l e l ) l i b r a r y( c a r e t )

13 l i b r a r y( msgps ) l i b r a r y(pROC)

15 l i b r a r y( p l y r )

#Readparameterv a l u e sfromthebashs c r i p t

17 args=commandArgs(T) s c e n a r i o=args[ 1 ]

19 f o l d e r=args[ 2 ] method=args[ 3 ]

21 k=as.numeric(args[ 4 ] )

sapply(l i s t. f i l e s (”HDSI −code”, pattern=” ∗ .R$”) ,f u n c t i o n( x ) sou rce( paste 0 (”HDSI −code/”, x ) ) )

23 #readt r a i n i n gd a t a s e tandt e s td a t a s e t i f( f o l d e r==”r e a l d a t a”) *{*

25 sample*<*−readRDS (f i l e= paste 0 (”output/r e a l d a t a”, s c e n a r i o ,”/ sample1.RData”) )

t r a i n d f*<*−sample $ t r a i n

27 e l s e

*} {*

t r a i n d f*<*−readRDS (f i l e= paste 0 (”output/s c e n a r i o”, s c e n a r i o ,”/ t r a i n d f 1.RData”) )

29

#Generateformula&d es i gnmatrix

*}*

31 i n t term=2

x*<*−c(r ep(”. ∗”, i n t term −1) ,”.”)

33 x*<*−Reduce (’paste 0’, x )

f=as.formula(paste(”y˜”, x ) )

35 Matrix=model.matrix( f , t r a i n d f ) [ , − 1 ]#model.matrix:c r e a t ea d es i g nmatrix

outvar=”y”

37

39

##################################

##GeneticAlgorithm## ##################################

GA auc*<*−ga ( type=”r e a l −valued”, f i t n e s s = f i t n e s s , t r a i n=t r a i n d f , e f f e c t s i z e=”l a r g e”, r u l e=”auc”, method=method ,l ower=c

( 2 , 0 . 8 , 0 . 5 ) ,upper=c( k , 1 , 2 ) , pop Size =20 , run =10 , maxiter =200 , p a r a l l e l=T, p c r o s s o v e r = 0 . 8 , pmutation = 0 . 4 , n f o l d =5)

summary(GA auc )

capture . output (summary=summary(GA auc ) ,f i l e=paste 0 (”output/”,

f o l d e r , s c e n a r i o ,”/r e s u l t s/GA ”, method ,” auc.txt”) ,append=T)

41

43

Below is the code for running the HDSI-BO algorithm

1 HDSI binary*<*−f u n c t i o n(df, i n t=i n t e r a c t i o n numb , outvar=”y”, method= c(”glm”,”forward”,”l a s s o”,”r i d g e”,”e l a s t i c”,”Alasso”) ,

*{*

e f f e c t s i z e=”l a r g e”, seed =1 ,q=NA, a , b , s e l e c t i o n=c(”auc”,”b i c”) ) #readt r a i n i n gd a t a s e tandt e s t d a t a s e t

3 t r a i n d f*<*−d f $ t r a i n t e s t d f*<*−d f $ t e s t

5 #Generateformula&d es i gnmatrix

f*<*−gen formula( i n t=int , outvar=outvar )

7 Matrix=model.matrix( f , t r a i n d f ) [ , − 1 ]#model.matrix:c r e a t ea d es i g nmatrix

#Bootstraps

9 boots*<*−mbootsample ( k=q,i n t e r a c t i o n numb=i nt , e f f e c t s i z e=

e f f e c t s i z e , i n p u t d f=t r a i n d f , outvar=outvar , seed m u l t i p l i e r= seed )

11 #FitmodelswithHDSImethods

m e t h o d l i s t=l i s t(glm= HDSI r e g r e s s i o n , l a s s o = HDSI l a s s o , forward = HDSI forward , r i d g e=HDSI r idge , e l a s t i c=HDSI

e l a s t i c , Alasso=HDSI Alasso )

13 #op *<*−pbapply: :pboptions(nout=9000)#themaximumnumbero f t imesthep r o g r e s sbari supdated

*{*

r e s u l t=l a p p l y( 1 :l e n g t h( boots ) ,f u n c t i o n( x )

15 rows=boots [ [ x ] ] [ [ 2 ] ]#samples

columns=boots [ [ x ] ] [ [ 1 ] ]#f e a t u r e s

17 df=t r a i n d f [ rows , columns ] y*<*−t r a i n d f [ rows , outvar ]

19 df*<*−c bind(df, y ) #Runthemodel

21 r et ur n( m e t h o d l i s t [ [ method ] ] ( t r a i n d f =df, outvar = outvar , f

= f , boot=x ) )

*}* )

23

#pbapply: :pboptions(op)

#Summarizethef i n a lr e s u l t s

r e s*<*−r bind. f i l l ( fun 1 ( r e s u l t , 1 ) ,a l l=T) auc*<*−r bind. f i l l ( fun 1 ( r e s u l t , 2 ) ,a l l=T) b i c*<*−r bind. f i l l ( fun 1 ( r e s u l t , 3 ) ,a l l=T) #Features e l e c t i o n

f e a t u r e*<*−f e a t u r e s e l e c t i o n ( res , auc , bic , a=a , b=b , s e l e c t i o n= s e l e c t i o n )

s p l i t var=u n l i s t(s t r s p l i t( f e a t u r e ,”:”) )

f e a t u r e .new*<*−u nique(c(s p l i t var, f e a t u r e ) ) i n t e r a c t i o n*<*−sum( g r e p l (”:”, f e a t u r e .new) )

marginal*<*−l e n g t h( f e a t u r e .new) −i n t e r a c t i o n

i f(l e n g t h( f e a t u r e .new)==0)re t ur n(l i s t(summary=data.frame( marginal =0 ,i n t e r a c t i o n=0 , auc=0) ) )

#Fitnewmodels

f .new=as.formula(paste( outvar ,”˜”,paste( f e a t u r e .new, c o l l a p s e

=”+”) ) )

glm.new *<*− glm( f .new,f a m i l y=binomial(l i n k= l o g i t ) ,data= t r a i n d f , maxit = 200 )

auc . f i n*<*−auc ( t e s t d f [ , outvar ] ,as.v e c t o r(p r e d i c t(glm.new, t e s t d f , type=”r esp on se”) ) )

f i n*<*− l i s t(summary=data.frame( marginal=marginal ,i n t e r a c t i o n= i n t e r a c t i o n, auc=auc . f i n ) , f e a t u r e s=f e a t u r e .new)

r et ur n( f i n )

*}*

25

27

29

31

33

35

37

39

41

43

Below are functions used for the genetic algorithm and the HDSI-BO algorithm.

#f i t n e s sf u n c t i o n

f i t n e s s =f u n c t i o n( x , n f o l d =3 , t r a i n=t r a i n d f ,i n t e r a c t i o n numb=2 ,

outvar=”y”, method=”glm”, e f f e c t s i z e=”l a r g e”, seed =1 , r u l e=c(”auc

q*<*−c e i l i n g( x [ 1 ] )

d a t a s i z e*<*−nrow( t r a i n )

index*<*−CVgroup (K=n fold , d a t a s i z e=d a t a s i z e , seed=seed ) #c l*<*−m ake Cluster( 39 ,type=”FORK” )

r e s*<*−s apply( 1 : nfold ,K HDSI ,data=t r a i n ,index=index,i n t e r a c t i o n numb=i n t e r a c t i o n numb , method=method , e f f e c t s i z e= e f f e c t s i z e , seed=seed ,q=q, a=x [ 2 ] , b=x [ 3 ] , s e l e c t i o n=rule , outvar=outvar )

#s t o p C l u s t e r(c l)

”,”b i c”) ) *{*

1

3

5

7

9

11 *}*

mean auc*<*−mean( r e s ) r et ur n(mean auc )

13 #S p l i tKgroups

CVgroup *<*− f u n c t i o n(K, d a t a s i z e , seed ) *{*

15

17

19

21

23 *}*

c v l i s t *<*− l i s t( ) s e t. seed ( seed )

n *<*− rep( 1 : K,c e i l i n g( d a t a s i z e/K) ) [ 1 : d a t a s i z e ] temp *<*− sample( n , d a t a s i z e )

x *<*− 1 :K

dataseq *<*− 1 : d a t a s i z e

c v l i s t *<*− l a p p l y( x ,f u n c t i o n( x ) dataseq [ temp==x ] ) r et ur n( c v l i s t )

25 #K−f o l dCross −v a l i d a t i o n

K HDSI =f u n c t i o n( x ,data=t r a i n ,index=index,i n t e r a c t i o n numb=

i n t e r a c t i o n numb , method=method , e f f e c t s i z e= e f f e c t s i z e , seed= seed ,q=q, a=a , b=b , s e l e c t i o n , outvar=outvar ) *{*

27

29

31 *}*

t r a i n d f*<*−d ata[ −index[ [ x ] ] , ] t e s t d f*<*−d ata[index[ [ x ] ] , ]

df*<*− l i s t( t r a i n=t r a i n d f , t e s t=t e s t d f )

r et ur n( HDSI binary (df=df, i n t=i n t e r a c t i o n numb , outvar=outvar , method=method , e f f e c t s i z e= e f f e c t s i z e , seed=seed ,q=q, a=a , b=b , s e l e c t i o n=s e l e c t i o n ) $summary$ auc )

#C a l c u l a t eAUC

33 CalAUC =f u n c t i o n(model, t e s t d f ,r e a l)

*{*

pred=p r e d i c t(model, newdata = t e s t d f , type=’r es pon se’)

35 r o c r . pred=p r e d i c t i o n ( pred ,r e a l)

r o c r . p e r f=performance ( r o c r . pred ,’auc’)

37 as.numeric( r o c r . perf@ y . v a l u e s )

*}*

39

#Features e l e c t i o n

41 f e a t u r e s e l e c t i o n=f u n c t i o n(coef, auc , bic , a , b , s e l e c t i o n=c(”auc”,” b i c”) )

*{*

*{*

ind*<*−a pply(coef, 2 ,f u n c t i o n( x )

43 x*<*−x [!i s.na( x ) ] mean x*<*−mean( x )

45 sd x*<*−sd( x )

#x lower*<*−mean x−1.96 ∗sd x

47 #x upper*<*−mean x+1.96 ∗sd x q low*<*−q u a n t i l e( x ,(1 −a )/2 )

49

51

53

55

57

59

61

63

65

67

69

71

73

75

77

79

81

83

85

87

89

91

93 *}*

q upper*<*−q u a n t i l e( x,1 −(1−a )/2 )

a*<*− i f e l s e( (q low *<*0&q upper *>*0) i s.na(sd x ) , 0 , 1 ) ) c o e f name*<*−names( ind ) [ ind!=0 ]

*| }*

#aucthreshold,b=

#max.auc *<*−a pply(auc, 2 ,f u n c t i o n(x) #x *<*−x[!i s.na(x)]

*{*

#i f(l e n g t h(x)==0)r e tu rn(NA) #e l s e

*}*

#r et ur n(max(x,na.rm=T)) ) #max.auc *<*−max.auc[ −1 ]

#max.auc *<*−max.auc[!i s.na(max.auc)] #mean auc*<*−mean(max.auc)

#sd auc*<*−sd(max.auc)

#auc upper*<*−mean auc+1.64 ∗sd auc

#y*<*−names(max.auc)[max.auc *>*=q u a n t i l e(max.auc, 0 . 9 )] i f( s e l e c t i o n==”auc”)

*{*

*{*

min. auc *<*−a pply( auc , 2 ,f u n c t i o n( x ) x*<*−x [!i s.na( x ) ]

i f(l e n g t h( x )==0)r e tu rn(NA) e l s e

*}*

r et ur n(min( x ,na.rm=T) ) ) min. auc *<*−min. auc [ −1 ]

min. auc *<*−min. auc [!i s.na(min. auc ) ] mean.min auc*<*−mean(min. auc )

sd.min auc*<*−sd(min. auc )

auc lower*<*−mean.min auc+b∗sd.min auc

auc name*<*−names(min. auc ) [min. auc *>*auc lower] comb*<*−i n t e r s e c t(c o e f name , auc name)

r et ur n( comb) e l s e

*{*

*} {*

max. b i c *<*−a pply( bic , 2 ,f u n c t i o n( x ) x*<*−x [!i s.na( x ) ]

i f(l e n g t h( x )==0)re tu rn(NA) e l s e

*}*

r et ur n(max( x ,na.rm=T) ) ) max. b i c *<*−max. b i c [ −1 ]

max. b i c *<*−max. b i c [!i s.na(max. b i c ) ] mean.max. b i c *<*−mean(max. b i c )

sd.max. b i c *<*−sd(max. b i c )

b i c upper*<*−mean.max. b i c+b ∗sd.max. b i c

b i c name*<*−names(max. b i c ) [max. bic *<*b i c upper] comb*<*−i n t e r s e c t(c o e f name , b i c name)

r et ur n( comb)

*}*

#Generateformula

gen formula=f u n c t i o n( int , outvar ) *{*

x*<*−c(r ep(”. ∗”, int −1) ,”.”) x*<*−Reduce (’paste 0’, x )

f*<*−a s.formula(paste( outvar ,”˜”, x ) )

r et ur n( f )

*}*

#Creater e s u l tl i s t

make r e s u l t =f u n c t i o n( type=c(”c o e f”,”auc”) , method=method , n=

i f(!r e q u i r e( p l y r ) )i n s t a l l.packages(”p l y r”) l i b r a r y( p l y r )

r e s*<*−l a p p l y( 1 : n ,f u n c t i o n( x ) readRDS (f i l e= paste 0 (”output/ s c e n a r i o”, s ,”/”, method , x ,”.RData”) ) [ [ type ] ] )

r e s*<*−r bind. f i l l ( res ,a l l=T)

r et ur n( r e s )

l e n g t h( boots ) , s=s c e n a r i o ) *{*

*}*

#Maker e s u l tl i s t − whennoRDatai ssaved

fun 1 *<*− f u n c t i o n( l s t , n ) *{*

sapply( l s t , ‘ [ ‘ , n )

*}*

#BIC

tLL *<*−− deviance(model)

k *<*− model$ df

b i c c a l*<*−f u n c t i o n(model) *{*

n *<*− model$ nobs

#AICc *<*−− tLL+2∗k+2∗k∗ (k+1)/(n −k−1) #AICc

BIC*<*−l o g( n ) ∗k − tLL r et ur n( BIC)

*}*

#samplingdataf o rs i m u l a t i o ns c e n a r i o s

gendata*<*−f u n c t i o n( ta ) *{*

index 0*<*−which( ta $ y==0) index 1*<*−which( ta $ y==1)

index 0*<*−sample(i ndex 0 , 150 ,r e p l a c e=F)

index 1*<*−sample(i ndex 1 , 150 ,r e p l a c e=F) index*<*−c(i ndex 0 ,index 1 )

r et ur n( ta [index, ] )

95

97

99

101

103

105

107

109

111

113

115

117

119

121

123

125

127

129

131

133

135

137

#Regressionf o rHDSI −BO

*}*

139 HDSI r e g r e s s i o n=f u n c t i o n( t r a i n d f =df, outvar = outvar , f = f , boot )

*{*

x=t r a i n d f [ ,names( t r a i n d f )!=outvar ]

141

143

145

147

149

151

*}*

y=t r a i n d f [ , outvar ]

glm *<*− glm( f ,f a m i l y=binomial(l i n k= l o g i t ) ,data= t r a i n d f , maxit = 200 )

c o e f.glm *<*−a s.data.frame(t(glm $ c o e f f i c i e n t s) )

auc . ord*<*−auc ( y ,p r e d i c t(glm, t r a i n d f , type=”r es pon se”) ) aucdata .glm *<*−a s.data.frame(t(glm $ c o e f f i c i e n t s) )

aucdata .glm[ ] *<*−auc . ord b i c*<*−BIC (glm)

b i c .glm *<*−a s.data.frame(t(glm $ c o e f f i c i e n t s) ) b i c .glm[ ] *<*−b i c

r e s*<*− l i s t(c o e f=c o e f.glm, auc=aucdata .glm, b i c=b i c .glm)

153 #LASSOf o rHDSI −BO

HDSI l a s s o=f u n c t i o n( t r a i n d f=df, outvar=outvar , f=f , boot ) *{*

155 Matrix=s t a t s : :model.matrix( f , t r a i n d f ) [ , − 1 ]

c v f i t=cv . glmnet ( Matrix , t r a i n d f [ , outvar ] ,f a m i l y=”binomial”, alpha =1 , type . measure=”c l a s s”)

157 lambda . 1se= c v f i t $ lambda . 1se lambda .min= c v f i t $ lambda .min

159 model=glmnet ( Matrix , t r a i n d f [ , outvar ] , lambda = lambda . 1se, alpha =1 , s t a n d a r d i z e=F ,f a m i l y=”binomial”)

c o e f=as.data.frame(t(as.matrix(c o e f(model, s=lambda . 1se) ) ) )

161 i f(sum(c o e f!=0 ) *<*=1) #t e s ti fthemodelwithlambda. 1sei stoo s imple(onlynoparameters)

*{*

model= glmnet : : glmnet ( Matrix , t r a i n d f [ , outvar ] , lambda = lambda .min, alpha =1 , s t a n d a r d i z e=F ,f a m i l y=”binomial”)

163 c o e f=as.data.frame(t(as.matrix(c o e f(model, s=lambda .min) ) ) )

*}*

165 l a s s o . pred=p r e d i c t(model, newx=Matrix , type=”res pon se”) pred*<*− p r e d i c t i o n ( l a s s o . pred , t r a i n d f [ , outvar ] )

167 auc . p e r f*<*−p erformance ( pred , measure=”auc”) auc . ord*<*−u n l i s t( auc . perf@ y . v a l u e s )

169 aucdata . l a s s o*<*−c o e f

aucdata . l a s s o [ aucdata . l a s s o!=0 ] *<*−u n l i s t( auc . ord )

171 aucdata . l a s s o [ aucdata . l a s s o ==0]*<*−NA c o e f[c o e f==0] *<*−NA

173 b i c*<*−b i c c a l (model) b i c . l a s s o*<*−c o e f

175 b i c . l a s s o [ b i c . l a s s o!=0 ] *<*−b i c

b i c . l a s s o [ b i c . l a s s o ==0]*<*−NA

177 r e s*<*− l i s t(c o e f=coef, auc=aucdata . l a s s o , b i c=b i c . l a s s o ) #r e s*<*− l i s t(c o e f=coef,auc=aucdata.l a s s o)

179 #saveRDS(res,f i l e=paste 0(”output/s c e n a r i o” ,s c e n a r i o, ”/l a s s o ” ,boot, ” .RData”))

*}*

181 #E l a s t i cnetf o rHDSI −BO

HDSI e l a s t i c=f u n c t i o n( t r a i n d f=df, outvar=outvar , f=f , boot ) *{*

183

185

187

189

191

193

195

197

199

*}*

Matrix=s t a t s : :model.matrix( f , t r a i n d f ) [ , − 1 ]

e l a s t i c*<*−t r a i n ( f ,data=t r a i n d f , method=”glmnet”, t r C o n t r o l= t r a i n C o n t r o l (”cv”, number=5) , tune Length =10)

alpha*<*− e l a s t i c $ bestTune [ 1 ] lambda*<*− e l a s t i c $ bestTune [ 2 ]

model=glmnet ( Matrix , t r a i n d f [ , outvar ] , lambda = lambda , alpha= alpha , s t a n d a r d i z e=F ,f a m i l y=”binomial”)

c o e f=as.data.frame(t(as.matrix(c o e f(model, s=lambda ) ) ) ) c o e f[c o e f==0] *<*−NA

e l a s t i c . pred=p r e d i c t(model, newx=Matrix , type=”re spo ns e”) pred*<*− p r e d i c t i o n ( e l a s t i c . pred , t r a i n d f [ , outvar ] )

auc . p e r f*<*−p erformance ( pred , measure=”auc”) auc . ord*<*−u n l i s t( auc . perf@ y . v a l u e s )

aucdata . e l a s t i c*<*−c o e f

aucdata . e l a s t i c [!i s.na( aucdata . e l a s t i c ) ] *<*−u n l i s t( auc . ord ) b i c*<*−b i c c a l (model)

b i c . e l a s t i c*<*−c o e f

b i c . e l a s t i c [!i s.na(c o e f) ] *<*−b i c

r e s*<*− l i s t(c o e f=coef, auc=aucdata . e l a s t i c , b i c=b i c . e l a s t i c )

201 #Ridger e g r e s s i o nf o rHDSI −BO

HDSI r i d g e=f u n c t i o n( t r a i n d f=df, outvar=outvar , f=f , boot ) *{*

203 Matrix=s t a t s : :model.matrix( f , t r a i n d f ) [ , − 1 ]

c v f i t=cv . glmnet ( Matrix , t r a i n d f [ , outvar ] ,f a m i l y=”binomial”, alpha =0 , type . measure=”c l a s s”)

205 lambda . 1se= c v f i t $ lambda . 1se lambda .min= c v f i t $ lambda .min

207 model=glmnet ( Matrix , t r a i n d f [ , outvar ] , lambda = lambda . 1se, alpha =0 , s t a n d a r d i z e=F ,f a m i l y=”binomial”)

c o e f=as.data.frame(t(as.matrix(c o e f(model, s=lambda . 1se) ) ) )

209 i f(sum(c o e f!=0 ) *<*=1) #t e s ti fthemodelwithlambda. 1sei stoo s imple(onlynoparameters)

*{*

model= glmnet : : glmnet ( Matrix , t r a i n d f [ , outvar ] , lambda = lambda .min, alpha =0 , s t a n d a r d i z e=F ,f a m i l y=”binomial”)

211 c o e f=as.data.frame(t(as.matrix(c o e f(model, s=lambda .min) ) ) )

*}*

213 l a s s o . pred=p r e d i c t(model, newx=Matrix , type=”res pon se”)

pred*<*− p r e d i c t i o n ( l a s s o . pred , t r a i n d f [ , outvar ] )

215 auc . p e r f*<*−p erformance ( pred , measure=”auc”) auc . ord*<*−u n l i s t( auc . perf@ y . v a l u e s )

217 aucdata . l a s s o*<*−c o e f

aucdata . l a s s o [ aucdata . l a s s o!=0 ] *<*−u n l i s t( auc . ord )

219 aucdata . l a s s o [ aucdata . l a s s o ==0]*<*−NA c o e f[c o e f==0] *<*−NA

221 b i c*<*−b i c c a l (model) b i c . l a s s o*<*−c o e f

223 b i c . l a s s o [!i s.na( b i c . l a s s o ) ] *<*−b i c

r e s*<*− l i s t(c o e f=coef, auc=aucdata . l a s s o , b i c=b i c . l a s s o )

225 #r e s*<*− l i s t(c o e f=coef,auc=aucdata.l a s s o)

#saveRDS(res,f i l e=paste 0(”output/s c e n a r i o” ,s c e n a r i o, ”/l a s s o ” ,boot, ” .RData”))

227 *}*

#Adaptivel a s s of o rHDSI −BO

229 HDSI Alasso=f u n c t i o n( t r a i n d f=df, outvar=outvar , f=f , boot ) Matrix=s t a t s : :model.matrix( f , t r a i n d f ) [ , − 1 ]

*{*

231 Alasso*<*−cv . glmnet ( Matrix , t r a i n d f [ , outvar ] ,f a m i l y=”binomial”, alpha =0 , type . measure=”c l a s s”)

lambda .min *<*−A lasso $ lambda .min

233 best r i d g e c o e f*<*−a s.numeric(c o e f( Alasso , s=lambda .min) ) [ −1 ]

a l a s s o cv*<*−cv . glmnet ( Matrix , t r a i n d f [ , outvar ] ,f a m i l y=”binomial”

, alpha =1 , penalty .f a c t o r=1/abs( best r i d g e c o e f) , type . measure

=”c l a s s”)

235 a l a s s o*<*−glmnet ( Matrix , t r a i n d f [ , outvar ] ,f a m i l y=”binomial”, alpha

=1 , penalty .f a c t o r=1/abs( best r i d g e c o e f) , lambda=a l a s s o cv $ lambda . 1se)

Coef . a l a s s o=as.data.frame(t(as.matrix(c o e f( a l a s s o cv , s=a l a s s o cv $ lambda . 1se) ) ) )

*{*

237 i f(sum( Coef . a l a s s o!=0 ) *<*=1) #t e s ti fthemodelwithlambda. 1se i stoosimple(onlynoparameters)

a l a s s o*<*−glmnet ( Matrix , t r a i n d f [ , outvar ] ,f a m i l y=”binomial”, alpha =1 , penalty .f a c t o r=1/abs( best r i d g e c o e f) , lambda= a l a s s o cv $ lambda .min)

239 Coef . a l a s s o=as.data.frame(t(as.matrix(c o e f( a l a s s o cv , s= a l a s s o cv $ lambda .min) ) ) )

241 Coef . a l a s s o [ Coef . a l a s s o ==0]*<*−NA

*}*

a l a s s o . pred=p r e d i c t( a l a s s o , newx=Matrix , type=”re sp ons e”)

243 auc*<*−auc ( t r a i n d f [ , outvar ] ,as.v e c t o r( a l a s s o . pred ) ) #rownames(Coef.a l a s s o)[Coef.a l a s s o!=0 ]

245 aucdata . a l a s s o*<*−Coef . a l a s s o

aucdata . a l a s s o [!i s.na( aucdata . a l a s s o ) ] *<*−auc

247 b i c*<*−b i c c a l ( a l a s s o )

249

251 *}*

b i c . a l a s s o*<*−Coef . a l a s s o

b i c . a l a s s o [!i s.na( b i c . a l a s s o ) ] *<*−b i c

r e s*<*− l i s t(c o e f=Coef . a l a s s o , auc=aucdata . a l a s s o , b i c=b i c . a l a s s o )

#Estimatethenumbero fboostrapsamplesB

253 B e s t=f u n c t i o n( p , k=NA, rows=NA,i n t e r a c t i o n numb=2 , e f f e c t s i z e=c (”l a r g e”,”medium”,”small”) )

*{*

#p r i n t(c(p,k))

255

257

259

261

263

265

267

269

271

273

275

277

279

denominator=choose( p ,i n t e r a c t i o n numb)

numerator=choose( k ,i n t e r a c t i o n numb)#where,2i stheorder o fi n t e r a c t i o nego f2orderi n t e r a c t i o ni sX1 X2

#p r i n t(c(numerator,denominator))

i n i t prob=numerator/denominator

#Minimumb o o t s t r a p swhichw i l lg i v eminimum13o c c u r r e n c e s( c o n s i d e rl a r g ee f f e c t)o fani n t e r a c t i o nv a r i a b l ewith99% c o n f i d e n c e

*{ }*

i f( e f f e c t s i z e==”l a r g e”) p r e f e r occurence = 13

e l s ei f( e f f e c t s i z e ==”medium”) p r e f e r occurence = 32 e l s ei f( e f f e c t s i z e ==”s mall”) p r e f e r occurence = 200 e l s e p r e f e r occurence = e f f e c t s i z e

*{ }*

*{ }*

*{ }*

#p r i n t(c(p r e f e r occurence,i n i t prob))

min boot=c e i l i n g( p r e f e r occurence/i n i t prob )#where32i s minimumnumbero foccurenced e s i r e df o rani n t e r a c t i o n

v a r i a b l eduringboots t rapping. #p r i n t(min boot)

max boot= 8 ∗min boot #p r i n t(max boot)

f opt=f u n c t i o n( x , y= i n i t prob ,max x=max boot , p r e f e r=p r e f e r occurence )

*{*

Val=qbinom( 0 . 0 5 ,f l o o r( x ) , y )#P(X *>*Val) *>*=95% value =( p r e f e r −Val ) ˆ2+(x/max x )

r et ur n( value )

*}*

bootvalue =optimize( f opt , i n t e r v a l =c(min boot ,max boot ) ) #p r i n t(c e i l i n g(bootvalue $minimum))

r et ur n(c e i l i n g( bootvalue $minimum) )

*}*

#Bootstrapping

281 bootsample=f u n c t i o n( k=NA,i n t e r a c t i o n numb=2 , e f f e c t s i z e=”l a r g e”, inputdf , outvar , seed m u l t i p l i e r =1)

*{*

p*<*−dim( i n p u t d f ) [ 2 ] − l e n g t h( outvar )

283 rows*<*−dim( i n p u t d f ) [ 1 ] #Findki fnotprovidedbyuser

285 i f(i s.na( k )==T)

*{*

optimal k=f u n c t i o n( x , r=rows )

*{*

287 i n t=choose( x ,i n t e r a c t i o n numb) value=abs( ( i n t+x ) −r )/r

289 r et ur n( value )

*}*

291 k=s t a t s : :optimize( optimal k , i n t e r v a l =c(i n t e r a c t i o n numb , p )

) $minimum k=f l o o r( k )

293 *}*

295 #p r i n t(p)

#Findthenumbero fsamplethatneedtobec r e a t e d

297 boots=B e s t ( p=p , k=k , rows=rows ,i n t e r a c t i o n numb=i n t e r a c t i o n numb , e f f e c t s i z e= e f f e c t s i z e )

#cat(c(”Bootstraps: ” ,boots))

299

##CreateVariablel i s tf o reachbootstrap

301 r e s=bootsample binary ( boots=boots , p=p , rows=rows , k=k , seed

m u l t i p l i e r = seed m u l t i p l i e r , i n p u t d f=inputdf , outvar=outvar )

303

*}*

305

r et ur n( r e s )

bootsample binary=f u n c t i o n( boots=boots , p , rows , k , seed m u l t i p l i e r =1 , inputdf , outvar )

*{*

307 i f(!r e q u i r e( pbapply ) )i n s t a l l.packages(”pbapply”) l i b r a r y( pbapply )

309 ##CreateVariablel i s tf o reachbootstrap samples =1: rows

311 op *<*− pbapply : : pboptions ( type =”timer”)#addsp r o g r e s sbarto v e c t o r i z e dRf u n c t i o n s

*{*

r e s=pblapply ( 1 : boots ,f u n c t i o n( x )

313 s e t. seed ( x ∗ seed m u l t i p l i e r ) ###C reatev a r i a b l ecombinations

315 f e a t u r e s *<*− sample( p , k ,r e p l a c e= FALSE)#r e p l a c e dsamples withf e a t u r e s/3

f e a t u r e s*<*− s o r t( f e a t u r e s )

317 ##CreateSamplel i s tf o reachbootstrap CAT 1 *<*− which( i n p u t d f [ , outvar ]==1)

319 CAT 0 *<*− which( i n p u t d f [ , outvar ]==0)

S 1*<*−sample(CAT 1 ,l e n g t h(CAT 1 ) ,r e p l a c e=T)

321 S 0*<*−sample(CAT 0 ,l e n g t h(CAT 0 ) ,r e p l a c e=T) samples *<*− c( S 1 , S 0 )

323 l i s t( f e a t u r e s , samples )

325

pbapply : : pboptions ( op )

r et ur n( r e s )

*}* )

*}*

mbootsample = memoise : : memoise ( bootsample )

327