University of California, Berkeley
U.C. Berkeley Division of Biostatistics Working Paper Series

Year 2016 *Paper* 351

Propensity score prediction for electronic healthcare databases using Super Learner and High-dimensional Propensity Score Methods

Cheng Ju[*] Mary Combs[†] Samuel D. Lendle[‡]
Jessica M. Franklin[**] Richard Wyss[††]
Sebastian Schneeweiss[‡‡] Mark J. van der Laan[§]

[*] Division of Biostatistics, University of California, Berkeley, cju@berkeley.edu

[†] Division of Biostatistics, University of California, Berkeley, maryac330@berkeley.edu

[‡] Division of Biostatistics, University of California, Berkeley, lendle@stat.berkeley.edu

[**] Division of Pharmacoepidemiology and Pharmacoeconomics, Department of Medicine, Brigham and Women's Hospital, Harvard Medical School

[††] Division of Pharmacoepidemiology and Pharmacoeconomics, Department of Medicine, Brigham and Women's Hospital, Harvard Medical School

[‡‡] Division of Pharmacoepidemiology and Pharmacoeconomics, Department of Medicine, Brigham and Women's Hospital, Harvard Medical School

[§] Division of Biostatistics, University of California, Berkeley, laan@berkeley.edu

Propensity score prediction for electronic healthcare databases using Super Learner and High-dimensional Propensity Score Methods

Cheng Ju, Mary Combs, Samuel D. Lendle, Jessica M. Franklin, Richard Wyss, Sebastian Schneeweiss, and Mark J. van der Laan

Abstract

<blockquote>The optimal learner for prediction modeling varies depending on the underlying data-generating distribution. Super Learner (SL) is a generic ensemble learning algorithm that uses cross-validation to select among a “library” of candidate prediction models. The SL is not restricted to a single prediction model, but uses the strengths of a variety of learning algorithms to adapt to different databases. While the SL has been shown to perform well in a number of settings, it has not been thoroughly evaluated in large electronic healthcare databases that are common in pharmacoepidemiology and comparative effectiveness research. In this study, we applied and evaluated the performance of the SL in its ability to predict treatment assignment using three electronic healthcare databases. We considered a library of algorithms that consisted of both nonparametric and parametric models. We also considered a novel strategy for prediction modeling that combines the SL with the high-dimensional propensity score (hdPS) variable selection algorithm. Predictive performance was assessed using three metrics: the negative log-likelihood, area under the curve (AUC), and time complexity. Results showed that the best individual algorithm, in terms of predictive performance, varied across datasets. The SL was able to adapt to the given dataset and optimize predictive performance relative to any individual learner. Combining the SL with the hdPS was the most consistent prediction method and may be promising for PS estimation and prediction modeling in electronic healthcare databases.</blockquote>

1 Introduction

Traditional approaches to prediction modeling have primarily included parametric models like logistic regression (Brookhart, Schneeweiss, Rothman, Glynn, Avorn, and Stürmer, 2006). While useful in many settings, parametric models require strong assumptions that are not always satisfied in practice. Modern statistical and machine learning methods, including classification trees, boosting, and random forest , have been developed to overcome the limitations of parametric models by requiring assumptions that are less restrictive (Hastie, Tibshirani, Friedman, Hastie, Friedman, and Tibshirani, 2009). Several of these methods have been evaluated for modeling propensity scores and have been shown to perform well in many situations when parametric assumptions are not satisfied (Setoguchi, Schneeweiss, Brookhart, Glynn, and Cook, 2008, Lee, Lessler, and Stuart, 2010, Westreich, Lessler, and Funk, 2010, Wyss, Ellis, Brookhart, Girman, Funk, LoCasale, and Stürmer, 2014). No single prediction algorithm, however, is optimal in every setting and the best performing prediction model will vary across different settings and data structures.

Super Learner is a general loss-based learning method that has been proposed and analyzed theoretically in (van der Laan, , Polley, and Hubbard, 2007). It is an ensemble learning algorithm that creates a weighted combination of many candidate learners to build the optimal estimator in terms of minimizing a specified loss function. It has been demonstrated that the super learner performs asymptotically at least as well as the best choice among the library of candidate algorithms if the library does not contain a correctly specified parametric model; otherwise, it achieves the same rate of convergence as the correctly specified parametric model (van der Laan and Dudoit, 2003, Dudoit and van der Laan, 2005, van der Vaart, Dudoit, and van der Laan, 2006). Benkeser, Lendle, Ju, and van der Laan (2016) further proposed an online-version of Super Learner for streaming data or big data. While the SL has been shown to perform well in a number of settings (van der Laan et al., 2007, Gruber, Logan, Jarrín, Monge, and Hernán, 2015, Rose, 2016), it's performance has not been thoroughly investigated within large electronic healthcare datasets that are common in pharmacoepidemiology and medical research. Electronic healthcare datasets based on insurance claims data are different from traditional medical datasets. It is impossible to directly use all of the claims codes as input covariates for supervised learning algorithms, as the number of codes could be larger than the sample size.

In the this study, we compared several statistical and machine learning prediction algorithms for estimating propensity scores within three electronic healthcare datasets. We considered a library of algorithms that consisted of both nonparametric and parametric models. We also considered a novel strategy for prediction modeling that combines the SL with an automated variable selection algorithm for electronic healthcare databases known as the high-dimensional propensity score (hdPS) (discussed later). The predictive performance for each of the methods was assessed using the negative log-likelihood, AUC (i.e., c-statistic or area under the curve), and time complexity. While the goal of the PS is to control for confounding by balancing covariates across treatment groups, in this study we were interested in evaluating the accuracy in the predictive performance of various PS estimation methods rather than the estimation of treatment effects. This study extends previous work that has implemented the SL within electronic healthcare data by proposing and evaluating the novel strategy of combining the SL with the hdPS variable selection algorithm for PS estimation. This study also provides the most extensive evaluation of the SL within healthcare claims data by utilizing three separate healthcare datasets and considering a large set of supervised learning algorithms, including the direct implementation of hdPS generated variables within the supervised algorithms.

2 Data Sources and Study Cohorts

We used three data sets Schneeweiss, Rassen, Glynn, Avorn, Mogun, and Brookhart (2009), Ju, Gruber, Lendle, Franklin, Wyss, Schneeweiss, and van der Laan (2016) to assess the performance of the models: the Novel Oral Anticoagulant Prescribing (NOAC) data set, the Nonsteroidal anti-inflammatory drugs (NSAID) data set and the Vytorin data set. Each dataset consisted of two types of covariates: baseline covariates which were selected a priori using expert knowledge, and claims codes. Baseline covariates include demographic variables (e.g. age, sex, census region and race) and other predefined covariates that were selected a priori using expert knowledge. Claims codes included information on diagnostic, drug, and procedural insurance claims for individuals within the healthcare databases.

2.1 Novel Oral Anticoagulant (NOAC) Study

The NOAC data set was generated to track a cohort of new users of oral anticoagulants to study the comparative safety and effectiveness of warfarin versus dabigatran in preventing stroke. Data were collected by United Healthcare between October, 2009 and December, 2012. The dataset includes 18,447 observations, 60 baseline covariates and 23,531 claims code covariates. Each claims code within the dataset records the number of times that specific code occurred for each patient within a pre-specified baseline period prior to initiating treatment.. The claims code covariates fall into four categories, or ”data dimensions”: inpatient diagnoses, outpatient diagnoses, inpatient procedures and outpatient procedures. For example, if a patient has a value of 2 for the variable ”pxop_V5260”, then the patient received the outpatient procedure coded as V5260 twice between October, 2009 and December, 2012.

2.2 Nonsteroidal anti-inflammatory drugs (NSAID) Study

The NSAID dataset was constructed to compare new-users of a selective COX-2 inhibitor versus a nonselective NSAID in the risk of GI bleed. The observations were drawn from a population of patients aged 65 years and older enrolled in both Medicare and the Pennsylvania Pharmaceutical Assistance Contract for the Elderly (PACE) programs between 1995 and 2002. The dataset consists of 49,653 observations, with 22 baseline covariates and 9,470 claims code covariates (Schneeweiss et al. (2009)). The claims code covariates fell into eight data dimensions: prescription drugs, ambulatory diagnoses, hospital diagnoses, nursing home diagnoses, ambulatory procedures, hospital procedures, doctor diagnoses and doctor procedures.

2.3 Vytorin Study

This data set was generated to track a cohort of new users of Vytorin and high-intensity statin therapies. The data were collected to study the effects of these medications on the combined outcome, myocardial infarction, stroke and death. The dataset includes all United Healthcare patients between January 1, 2003 December 31, 2012, who were 65 years of age or older on the day of entry into the study cohort (Schneeweiss, Rassen, Glynn, Myers, Daniel, Singer, Solomon, Kim, Rothman, Liu et al. (2012)). The dataset includes 148,327 observations, 67 baseline covariates and 15,010 code covariates. The claims code covariates fell into five data dimensions: ambulatory diagnoses, ambulatory procedures, prescription drugs, hospital diagnoses and hospital procedures.

3 Methods

In this paper, we used R (version 3.2.2) for the data analysis. For each dataset, we randomly selected 80% of the data as the training set and the rest as the testing set. We centered and scaled each of the covariates as some algorithms are sensitive to the magnitude of the covariates. We conduct model fitting and selection only on the training set, and assessed the goodness of fit of all the models on only the testing set to ensure objective measures of prediction reliability.

3.1 The high-dimensional propensity score algorithm

The high-dimensional propensity score (hdPS) is an automated variable selection algorithm that is designed to identify confounding variables within electronic healthcare databases. Healthcare claims databases contain multiple data dimensions, where each dimension represents a different aspect of healthcare utilization (e.g., outpatient procedures, inpatient procedures, medication claims, etc.). When implementing the hdPS, the investigator first specifies how many variables to consider within each data dimension. Following the notation of (Schneeweiss et al. (2009)) we let n represent this number. For example, if $n = 200$ and there are 3 data dimensions, then the hdPS will consider 600 codes.

For each of these 600 codes, the hdPS then creates three binary variables labeled frequent, sporadic, and once based on the frequency of occurrence for each code during a covariate assessment period prior to the initiation of exposure. In this example, there are now a total of 1,800 binary variables. The hdPS then ranks each variable based on its potential for bias using the Bross formula (Bross (1966), Schneeweiss et al. (2009)). Based on this ordering, investigators then specify the number of variables to include in the hdPS model, which is represented by k. A detailed description of the hdPS is provided by Schneeweiss et al. (2009).

3.2 Machine Learning Algorithm Library

We evaluated the predictive performance of a variety of machine learning algorithms that are available within the caret package (version 6.0) (Kuhn (2008), Kuhn, Wing, Weston, Williams, Keefer, Engelhardt, Cooper, Mayer, Team, Benesty et al. (2014)) in the R programming environment. Due to computational constraints, we screened the available algorithms to only include those that were computationally less intensive. A list of the chosen algorithms is provided in the Web Appendix.

Because of the large size of the data, we used leave group out (LGO) cross-validation instead of V-fold cross-validation to select the tuning parameters for each individual algorithm. We randomly selected 90% of the training data for model training and 10% of the training data for model tuning and selection. For clarity, we refer to these subsets of the training data as the LGO training set and the LGO validation set, respectively. After the tuning parameters are selected, we fit the selected models on the whole training set, and assess the models on the testing set. The split could be different for different algorithms. See the appendix for more details of the base learners.

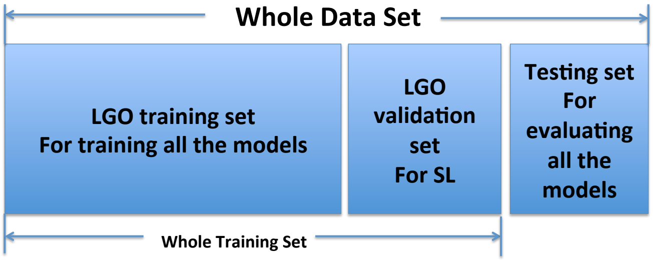


Figure 1: The split of dataset

3.3 Super Learner

Super Learner (SL) is a method for selecting an optimal prediction algorithm from a set of user-specified prediction models. The SL relies on the choice of a loss function (negative log-likelihood in the present study) and the choice of a library of candidate algorithms. The SL then compares the performance of the candidate algorithms using V-fold cross-validation: for each candidate algorithm, SL averages the estimated risks across the validation sets, resulting in the so-called cross-validated risk. Cross-validated risk estimates are then used to compute the best weighted linear convex combination of the candidate learners with the smallest estimated risk. This weighted combination is then applied to the full study data to produce a new set of predicted values and is referred to as the SL estimator (van der Laan et al. (2007), Polley and van der Laan (2010)).

Due to the computational constraints, in this study, we used LGO validation instead of V-fold cross-validation. We first fit every candidate algorithm on the LGO training set, then compute the Super Learner weight on the LGO validation set. This is so-called sample split super learner algorithm. We used the Super Learner package in R (Version: 2.0-15) to evaluate the predictive performance of three Super Learner estimators:

SL1 Included only baseline variables with all 23 of the previously identified traditional machine learning algorithms in the SL library.

SL2 Identical to SL1, but with the addition of the hdPS algorithms with different tuning parameters in its SL library. Note that only the hdPS algorithms had access to the claims code variables in SL2.

SL3 Identical to SL1, but with the addition of claims code covariates selected by the hdPS screening method. Based on the performance of single hdPS algorithms, a fixed pair of hdPS tuning parameters is selected, and SL3 finds the optimal ensemble of all the algorithm candidates fitted on the same baseline and hdPS covariates.

Super Learner	Libray	Covariates
SL1	All machine learning algorithms	Only baseline covariates.
SL2	All machine learning algorithms and the hdPS algorithm	Baseline covariates; Only the hdPS algorithm can use code data.
SL3	All machine learning algorithms	Baseline covariates and hdPS covariates generated from code data by hdPS screening method.

Table 1: Details of the three Super Learners considered.

3.4 Performance Metrics

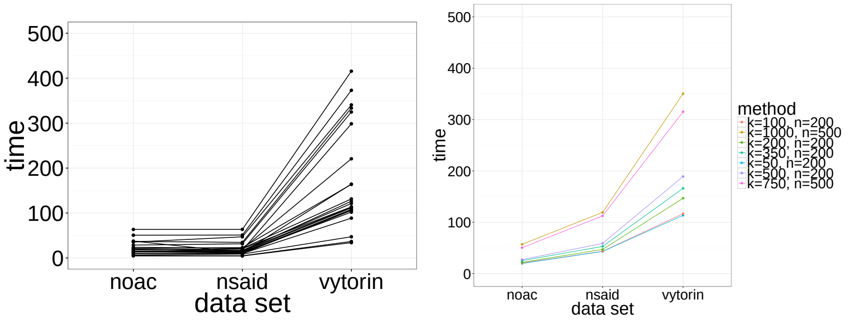
We used three criteria to evaluate the prediction algorithms: computing time, negative log-likelihood, and area under the curve (AUC). In statistics, a receiver operating characteristic (ROC), or ROC curve, is a plot that illustrates the performance of a binary classifier system as its discrimination threshold is varied. The curve is created by plotting the true positive rate against the false positive rate at various threshold settings. The AUC is then computed as the area under the ROC curve.

For both computation time and negative log-likelihood, smaller values indicate better performance, whereas for AUC the better classifier achieves greater values (Hanley and McNeil (1982)). Compared to the error rate, the AUC is a better assessment of performance for the unbalanced classification problem.

4 Results

4.1 Using the hdPS prediction algorithm with Super Learner

4.1.1 Computation Times



(a) Running time for the 23 individual machine learning algorithms with no Super Learner.

(b) Running time for the hdPS algorithms varying the parameter k from 50 to 750 for $n = 200$, and $n = 500$.

Figure 2: Running times for individual machine learning and hdPS algorithms without Super Learner. The y-axis is in log scale.

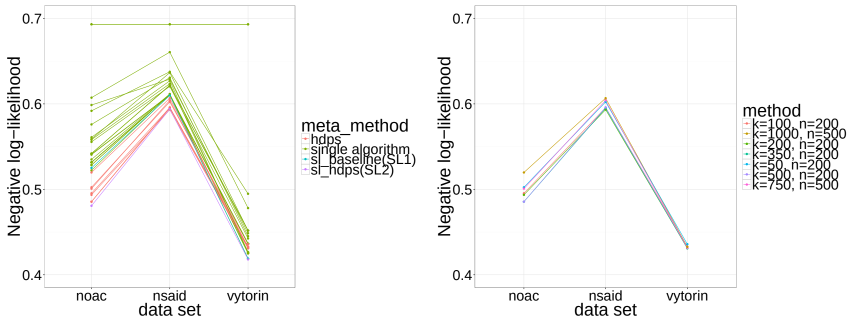
Figure 2 shows the running time for the 23 individual machine learning algorithms and the hdPS algorithm across all three datasets without the use of Super Learner. Running time is measured in seconds. Figure 2a shows the running time for the machine learning algorithms that only use baseline covariates. Figure 2b shows the running time for the hdPS algorithm at varying values of the tuning parameters k and n. Recall n represents the number of variables that the hdPS algorithm considers within each data dimension and k represents the total number of variables that are selected or included in the final hdPS model as discussed previously. The running time is sensitive to n, while less sensitive to k. This suggests most of the running time for hdPS is spent generating and screening covariates. The running time for the hdPS algorithm is generally around the median of all the running times of the machine learning algorithms with only baseline covariates. Here we only compared the running time for each pair of parameters for hdPS. It is worth noting that the variable creation and ranking only has to be done once for each value of n. Modifying values of k just means taking different numbers of variables from a list and refitting the logistic regression.

The running time of SL is not placed in the figures. Super Learner with baseline covariates takes just over twice as long as the sum of the running time for each individual algorithm in its library: SL splits data into training and validation sets, fits the base learners on the training set, finds weights based the on the validation set, and finally retrains the model on the whole set. In other words, Super Learner will fits every single algorithm twice, with additional processing time for computing the weights. Therefore, the running time will be about twice the sum of its constituent algorithms, which is what we see in this study (See Table 2).

Data Set	Algorithm	Processing Time (seconds)
NOAC	Sum of machine learning algorithms	481.13
	Sum of hdPS algorithms	222.87
	Super Learner 1	1035.43
	Super Learner 2	1636.48
NSAID	Sum of machine learning algorithms	476.09
	Sum of hdPS algorithms	477.32
	Super Learner 1	1101.84
	Super Learner 2	2075.05
VYTORIN	Sum of machine learning algorithms	3982.03
	Sum of hdPS algorithms	1398.01
	Super Learner 1	9165.93
	Super Learner 2	15743.89

Table 2: Running time of the machine learning algorithms, the hdPS algorithms, and Super Learners 1 and 2. Twice the sum of the running time of the machine learning algorithms is comparable to the running time of Super Learner 1 and twice the sum of the running times of both the machine learning algorithms and the hdPS algorithms is comparable to the running time of Super Learner 2.

4.1.2 Negative log-likelihood



(a) Negative log-likelihood for SL1, SL2, the hdPS algorithm, and the 23 machine learnng algorithms.

(b) Negative log-likelihood for the hdPS algorithm, varying the parameter k from 50 to 750 for $n = 200$, and $n = 500$.

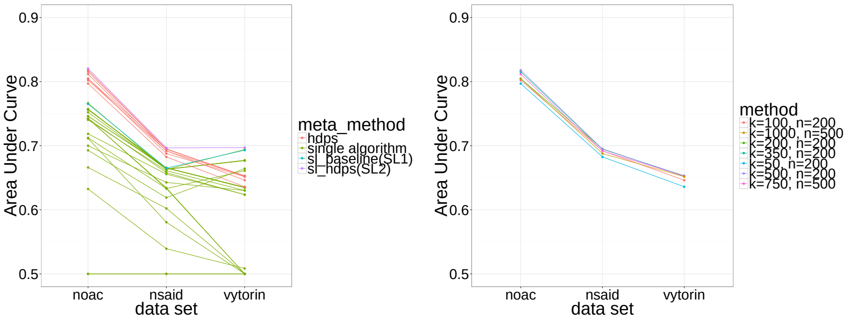
Figure 3: The negative log-likelihood for SL1, SL2, the hdPS algorithm, and the 23 machine learning algorithms.

Figure 3a shows the negative log-likelihood for Super Learners 1 and 2, and each of the 23 machine learning algorithms (with only baseline covariates) . Figure 3b shows the negative log-likelihood for hdPS algorithms with varying tuning parameters, n and k.

The performance of hdPS is not sensitive to either n or k. hdPS outperforms most individual algorithms in the library in most cases, as it takes advantage of the extra information from code data. However, in the Vytorin data set, there are still some machine learning algorithms which perform slightly better than hdPS with respect to the negative log-likelihood.

We can see the SL (without hdPS) outperforms all the other individual algorithms, empirically verifing the optimal property proved by previous literatures van der Laan et al. (2007), Polley and van der Laan (2010): the Super Learner can do at least as well as the best algorithm in the library. The figures show that including the hdPS algorithm improves the performance of Super Learner. With the help of hdPS, Super Learner achieves the best performance among all the algorithms (including hdPS itself). This suggests the time consumption is worthwhile for Super Learner.

4.1.3 AUC



(a) AUC of SL1, SL2, the hdPS algorithm, and the 23 machine learnng algorithms.

(b) AUC for the hdPS algorithm, varying the parameter k from 50 to 750 for $n = 200$, and $n = 500$.

Figure 4: The area under the ROC curve (AUC) for for Super Learners 1 and 2, the hdPS algorithm, and each of the 23 machine learning algorithms.

The SL uses loss-based cross-validation to select the optimal combination of individual algorithms. It is not surprising that it outperforms other algorithms with respect to the negative log-likelihood. As propensity score estimation can be considered a binary classification problem, we can use the Area Under the Curve (AUC) to compare performance across algorithms. Binary classification is typically determined by setting a threshold. As the threshold varies for a given classifier we can achieve different true positive rates (TPR) and false positive rates (FPR). A Receiver Operator Curve (ROC) space is defined by FPR and TPR as the x- and y-axes respectively, to depict the trade-off between true positives (benefits) and false positives (costs) at various classification thresholds. We then draw the ROC curve of TPR and FPR for each model and calculate the AUC. The upper bound for a perfect classifier is 1 while a naive random guess would achieve about 0.5.

In Figure 4a, we compare the performance of Super Learners 1 and 2, the hdPS algorithm, and each of the 23 machine learning algorithms. Although we optimized Super Learners with respect to the negative log-likelihood loss function, SL1 and SL2 have outstanding performance with respect to the AUC; Over the NOAC and NSAID data set, SL1 (with only baseline variables) achieves the best AUC compared to all machine learning algorithms in its library, with only a slightly weaker AUC performance than hdPS. In the VYTORIN data set, SL1 outperforms hdPS algorithms with respect to AUC, even though the hdPS algorithms use the additional claims data.

data	SL1	SL2	best hdPS (parameter k/n)
noac	0.7652	0.8203	0.8179 (500/200)
nsaid	0.6651	0.6967	0.6948 (500/200)
vytorin	0.6931	0.6970	0.6527 (750/500)

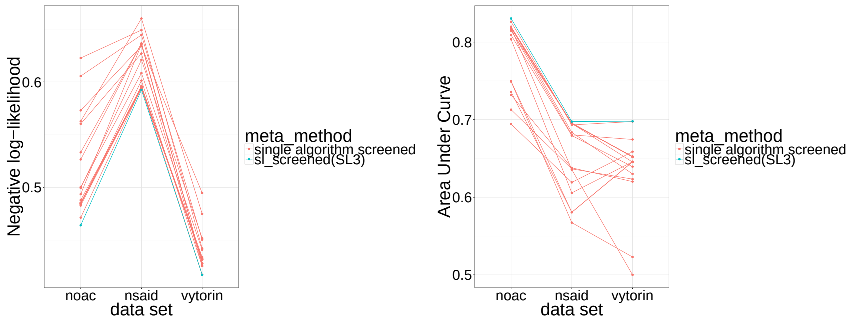
Table 3: Comparison of AUC for SL1, SL2 and best hdPS across three data sets. The best hdPS for noac is k = 500, n = 200, and for nsaid is k = 500, n = 200, for vytorin is k = 750, n = 500.

Super Learner 2 clearly combines the strength of the hdPS algorithm and all the machine learning algorithms in its library. Table 3 shows, in all three data sets, it achieves higher AUC over all the other algorithms, including hdPS and SL1.

4.2 Using the hdPS screening method with Super Learner

In the previous sections, we compared machine learning algorithms limited to only baseline covariates with the hdPS algorithms across different parameters (negative log-likelihood and AUC). The results showed that including the hdPS algorithm in a Super Learner library increases performance significantly. In this section, we combined the strength of the claims code data via the hdPS screening method with the machine learning algorithms to improve the propensity score estimation.

We first used the hdPS screening method (with tuning parameters $n = 200, k = 500$) to generate and screen the hdPS covariates. Then we combined these hdPS covariates with the baseline covariates to generate augmented datasets for each of the three datasets under consideration. We build a Super Learner library which included each of the 23 individual machine learning algorithms, fitted on both baseline and hdPS covariates. Note that, as the original hdPS method uses logistic regression for prediction, it can be considered a special case of LASSO (with $\lambda = 0$).



(a) Negative log-likelihood

(b) AUC

Figure 5: Negative log-likelihood and AUC of SL1, SL2, and SL3, compared with each of the single machine learning algorithms (with and without using hdPS covariates). We could see among all the single algorithms and Super Learners, SL3 performs the best cross three datasets

For convenience, we differentiate Super Learners 1, 2 and 3 by their algorithm libraries: machine learning algorithms with only baseline covariates, augmenting this library with hdPS, and only the machine learning algorithms but with both baseline and hdPS screened covariates. (See Table 1).

Figures 5 compares the negative log-likelihood and AUC, respectively, of all three Super Learners and machine learning algorithms. It is clear that the performance of all algorithms increases significantly by including the hdPS screened code covariates. SL3 is slightly better than SL2, and the difference is very small.

Data set	Performance Metric	Super Learner 1	Super Learner2	Super Learner 3
NOAC	AUC	0.7652	0.8203	0.8304
NSAID		0.6651	0.6967	0.6975
VYTORIN		0.6931	0.6970	0.698
NOAC	Negative Log-likelihood	0.5251	0.4808	0.4641
NSAID		0.6099	0.5939	0.5924
VYTORIN		0.4191	0.4180	0.4171

Table 4: Performance as measured by AUC and negative log-likelihood for the three Super Learners with the following libraries: machine learning algorithms with only baseline covariates, augmenting this library with hdPS, and only the machine learning algorithms but with both baseline and hdPS screened covariates. (See Table 1).

In table 4, we can again see the trend that performance improves from Super Learner 1 to 2 and from 2 to 3. The differences in AUC and in negative log-likelihood between SL1 and 2 are large, while these differences between SL2 and 3 are small. This suggests two things: First, the prediction step in the hdPS algorithm (logistic regression) works well: it performs approximately as well as the best individual machine learning algorithm in the library for Super Learner 3. Second, the hdPS screened covariates make the propensity score estimation more flexible; using Super Learner we can easily develop different models/algorithms which incorporate the covariate screening method from hdPS.

4.3 Weights of Individual Algorithms in Super Learners 1 and 2

Data Set	Algorithms Selected for SL1	Weight
NOAC	SL.caret.bayesglm_All	0.30
	SL.caret.C5.0_All	0.11
	SL.caret.C5.0Tree_All	0.11
	SL.caret.gbm_All	0.39
	SL.caret.glm_All	0.01
	SL.caret.pda2_All	0.07
	SL.caret.plr_All	0.01
NSAID	SL.caret.C5.0_All	0.06
	SL.caret.C5.0Rules_All	0.01
	SL.caret.C5.0Tree_All	0.06
	SL.caret.ctree2_All	0.01
	SL.caret.gbm_All	0.52
	SL.caret.glm_All	0.35
VYTORIN	SL.caret.gbm_All	0.93
	SL.caret.multinom_All	0.07

Data Set	Algorithms Selected for SL2	Weight
NOAC	SL.caret.C5.0_screen.baseline	0.03
	SL.caret.C5.0Tree_screen.baseline	0.03
	SL.caret.earth_screen.baseline	0.05
	SL.caret.gcvEarth_screen.baseline	0.05
	SL.caret.pda2_screen.baseline	0.02
	SL.caret.rpart_screen.baseline	0.04
	SL.caret.rpartCost_screen.baseline	0.04
	SL.caret.sddaLDA_screen.baseline	0.03
	SL.caret.sddaQDA_screen.baseline	0.03
	SL.hdps.100_All	0.00
	SL.hdps.350_All	0.48
	SL.hdps.500_All	0.19
NSAID	SL.caret.gbm_screen.baseline	0.24
	SL.caret.sddaLDA_screen.baseline	0.03
	SL.caret.sddaQDA_screen.baseline	0.03
	SL.hdps.100_All	0.25
	SL.hdps.200_All	0.21
	SL.hdps.500_All	0.01
	SL.hdps.1000_All	0.23
VYTORIN	SL.caret.C5.0Rules_screen.baseline	0.01
	SL.caret.gbm_screen.baseline	0.71
	SL.hdps.350_All	0.07
	SL.hdps.750_All	0.04
	SL.hdps.1000_All	0.17

Table 5: Non-zero weights of individual algorithms in Super Learners 1 and 2 across all three data sets.

Super Learner produces an optimal ensemble learning algorithm, i.e. a weighted combination of the candidate learners in its library. Table 5 shows the weights for all the non-zero weighted algorithms included in the optimal, data set-specific ensemble learner generated by SL 1 and 2. We can see for all the three data sets, the gradient boosting algorithm gbm has the highest weight. It is also interesting to note that across the different data sets the hdPS algorithms have very different weights. Using NOAC and NSAID, the hdPS algorithm plays a dominating role: hdPS algorithms occupy more than 50% of the weight. However in VYTORIN, boosting still plays the most important role, with weight 0.71. This suggests that gradient boosting and hdPS plays a significant role in the prediction of propensity scores. In further studies of prediction/estimation of propensity scores on similar data sets, we may first only include the algorithms with high weights if computation time is limited.

5 Discussion

Data Set	Method	Negative Log Likelihood	AUC	Negative Log Likelihood (Train)	AUC (Train)	Processing Time (Seconds)
NOAH	k=50, n=200	0.50	0.80	0.51	0.79	19.77
	k=100, n=200	0.50	0.80	0.50	0.80	20.69
	k=200, n=200	0.49	0.80	0.49	0.81	22.02
	k=350, n=200	0.49	0.82	0.47	0.83	25.38
	k=500, n=200	0.49	0.82	0.46	0.84	27.35
	k=750, n=500	0.50	0.81	0.45	0.85	50.58
	k=1000, n=500	0.52	0.80	0.43	0.86	57.08
	sl_baseline	0.53	0.77	0.53	0.77	1035.43
	sl_hdps	0.48	0.82	0.47	0.83	1636.48
NSAID	k=50, n=200	0.60	0.68	0.61	0.67	43.15
	k=100, n=200	0.60	0.69	0.60	0.69	43.48
	k=200, n=200	0.59	0.70	0.60	0.69	47.08
	k=350, n=200	0.60	0.69	0.59	0.70	52.99
	k=500, n=200	0.60	0.69	0.59	0.71	58.90
	k=750, n=500	0.60	0.69	0.58	0.71	112.44
	k=1000, n=500	0.61	0.69	0.58	0.72	119.28
	sl_baseline	0.61	0.67	0.61	0.66	1101.84
	sl_hdps	0.59	0.70	0.59	0.71	2075.05
VYTORIN	k=50, n=200	0.44	0.64	0.43	0.64	113.45
	k=100, n=200	0.43	0.65	0.43	0.65	116.73
	k=200, n=200	0.43	0.65	0.43	0.66	146.81
	k=350, n=200	0.43	0.65	0.42	0.67	166.18
	k=500, n=200	0.43	0.65	0.42	0.67	189.18
	k=750, n=500	0.43	0.65	0.42	0.68	315.22
	k=1000, n=500	0.43	0.65	0.42	0.68	350.45
	sl_baseline	0.42	0.69	0.42	0.70	9165.93
	sl_hdps	0.42	0.70	0.41	0.71	15743.89

Table 6: Perfomance for hdPS algorithms and Super Learners

5.1 Tuning Parameters for hdPS Screening Method

The screening process of hdPS needs to be cross-validated in the same step as its predictive algorithm. For this study, the computation is too expensive for this procedure, so there is an additional risk of overfitting due to the selection of hdPS covariates. A solution would be to generate various hdPS covariate sets under different hdPS hyper parameters and fit the machine learning algorithms on each covariate set. Then, SL3 would find the optimal ensemble among all the hdPS covariate set/learning algorithm combinations.

5.2 Performance of hdPS

Although hdPS is a simple logistic algorithm, it wisely takes advantage of extra information from claims data. It is, therefore, reasonable that the hdPS outperforms most algorithms in most cases. Processing time for the hdPS is sensitive to *n* while less sensitive of *k* (see 2). The performance is, however, not sensitive to either *n* or *k* (see 6). Therefore, Super Learners which include hdPS may save processing time by including only a limited selection of hdPS algorithms without sacrificing performance.

5.2.1 Risk of overfitting for hdPS

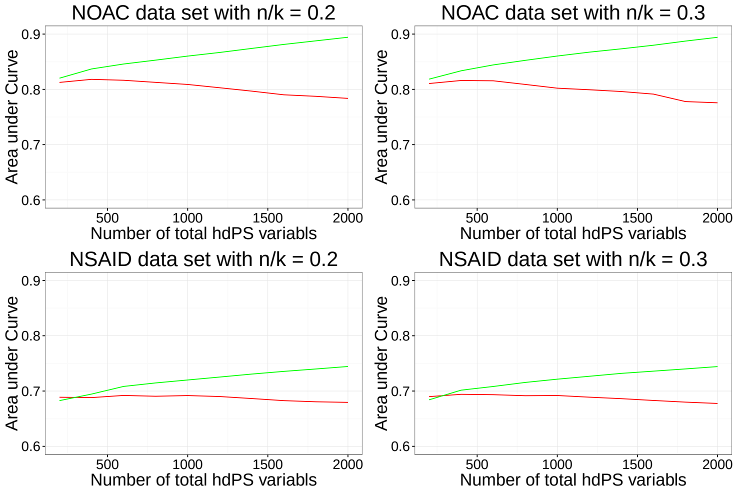


Figure 6: AUC for hdPS algorithms with different number of variables, *k*.

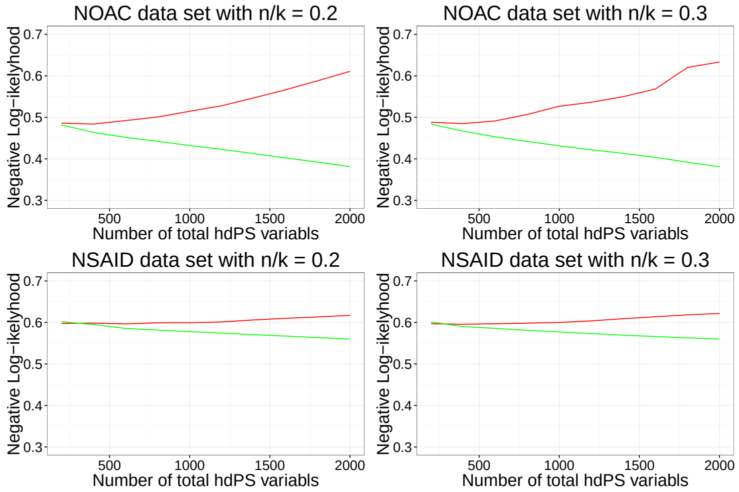


Figure 7: Negative loglikelihood for hdPS algorithms with different number of variables, k.

The hdPS algorithm utilizes many more features than traditional methods, which may raise the risk of overfitting. The performance table (table 6) shows the negative loglikelihood for both training set and testing set. We can see the difference of performance of hdPS in training set and test set are very small, and not sensitive to k and n. As long as we control k and n in a resonable range, we may not need to worry about the curse of dimensionality.

To study the risk of overfitting for hdPS across each data set, we fix the propotion of number of variable per dimension (n) and number of total hdPS variables (k), then increase k to see the performance of hdPS algorithms. The green lines represent performance over the training sets and red lines represent peformance over the test sets.

From figure 6, we see that increasing the number of variables in hdPS, results in an increase in AUC in the training sets. This is deterministically a result of increasing model complexity. To mitigate this effect, we looked at the AUC over the test sets to determine if model complexity reduces performance. For both $n/k = 0.2$ and $n/k = 0.4$, AUC in the testing sets is fairly stable for $k < 500$, but for larger values of k begins to decrease. We find then, that hdPS is only sensitive to overfitting for $k > 500$.

Similarly, in figure 7, the negative log-likelihood decreases as k gets larger. The negative log-likelihood in the testing sets begins to increase only for $k > 500$, similary to what we found for AUC. Thus, we conclude the negative log-likelihood is also not sensitive to k for $k < 500$.

Due to the large sample sizes of our datasets, the binary nature of the claims code covariates, and the sparsity of hdPS variables, these hdPS algorithms are at less of a risk of overfitting. However, the high dimensionality data may lead to -some computation issues.

5.2.2 Penalized hdPS

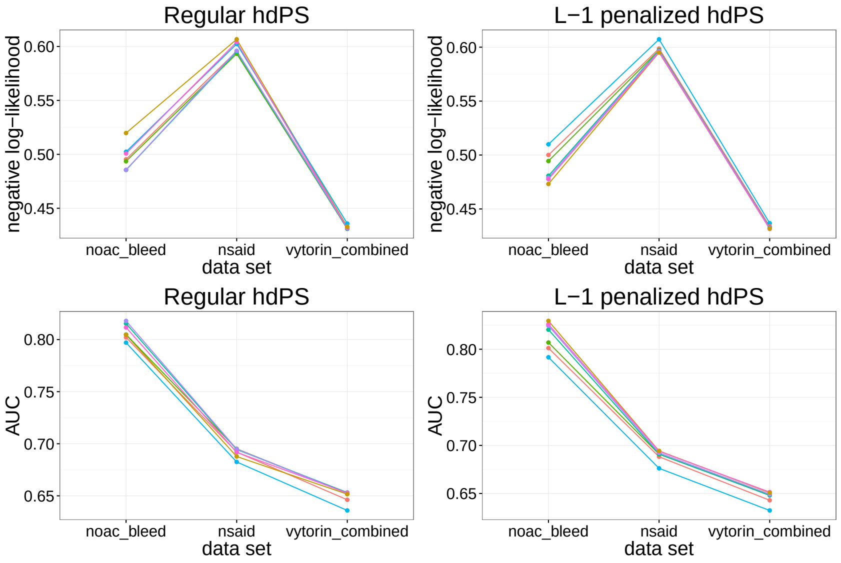


Figure 8: Unregularized hdPS Compared with Regularized hdPS

The hdPS algorithm uses multivariate logistic regression for its estimation. We compared the performance of this algorithm against that of regularized regression by implementing the estimation step using the cv.glmnet method in glmnet Friedman, Hastie, and Tibshirani (2009) package in R, which uses cross-validation to find the best tunin parameter λ.

To study if regularization can decrease the risk of overfitting for hdPS, we used $L-1$ regularization (LASSO) for the logistic regression step in hdPS. For every regular hdPS we used cross-validation to find out the best tunning parameter based on discrete Super Learner.

Figure 8 shows the negative log-likelihood and AUC over the test sets for unregularized hdPS (left) and regularized hdPS (right). We can see that using regularization can increase performance slightly. In this study, the sample size is relatively large. The regularization does not help a lot. However, when dealing with smaller data set, it is highly suggested to use regularized regression for the last step of hdPS algorithmm, or first generate hdPS covariates and then use Super Learner (as the idea of SL3).

5.3 Predictive Performance for SL

SL is a weighted linear combination of candidate learner algorithms that has been demonstrated to perform asymptotically at least as well as the best choice among the library of candidate algorithms, whether or not the library contains a correctly specified parametric statistical model. The results in previous sections show the performance of SL in a finite data set.

From the previous sections, we found that among algorithms which only utilize baseline variables, SL always outperforms the best candidate in its library. Also with respect to the prediction of propensity scores:

- Super Learner reliably outperforms candidate algorithms in AUC, even though the model selection step in SL minimizes a different performance criterion: the cross-validated negative log-likelihood.
- The hdPS screening method offers a simple way to utilize the information from claims data which increases estimation performance significantly. It is therefore reasonable to take advantage of hdPS covariates in Super Learner.

5.4 Data-adaptive property of SL

Besides the outstanding estimation performance, there are several other reasons to use Super Learner for the estimation of propensity scores: First, esimating the propensity score using a parametric model requires accepting strong assumptions concerning the functional form of the relationship between treatment allocation and the covariates, while propensity score model misspecification may result in significant bias in the treatment effect estimate (Rubin (2004), Brookhart et al. (2006)). Second, the relative performance of different algorithms relies heavily on the underlying data generate distribution. Therefore, to avoid model misspecification, we must try many models. This paper clearly demonstrates the strength of this approach: some algorithms perform well over several data sets, but not always. Including many different types of algorithms in the SL library accommodates this inevitability. Cross-validation helps us avoid the risk of overfitting, and so we may include as many algorithms as we can, if the computation consumption permits.

To summarize:

- From figure 5, the Gradient Boosting and hdPS have the dominating weight in all three data sets. Hence they are two most powerful individual algorithms for prediction of propensity scores in these three data sets. We may include them first if computation resource is limited.
- The optimal learner for prediction will highly depend on the underlying data-generating distribution: one model may succed in one case, while may fail in another data set. Therefore it is recommendable to use SL including as many competing algorithms as possible.

6 Conclusion

This article demonstrates the practical performance of Super Learner in finite sample sets for a specific prediction problem (propensity score prediction). The outstanding performances for both negative log-likelihood and AUC demonstrate the reliability of the performance of Super Learner. Based on the cross-validation procedure, SL can adaptively combine a number of different estimators with non-negative weights while avoiding overfitting.

One of the advantages of Super Learner is how easily it can adopt the strengths of field-specific algorithms. In this study that focused on electronic healthcare databases, we successfully implemented SL utilizing the hdPS algorithm to predict propensity scores.

In conclusion, this study has two contributions:

- The primary contribution of this paper is that this is the first paper to consider and introduce the novel strategy of combining the SL with the hdPS.
- The other contribution is that this is the most thorough evaluation of the SL within healthcare claims data. While there are published papers that have looked at the SL in claims data before, they have not done so to this extent and detail.

References

Benkeser, D., S. D. Lendle, C. Ju, and M. J. van der Laan (2016): “Online cross-validation-based ensemble learning,” .

Brookhart, M. A., S. Schneeweiss, K. J. Rothman, R. J. Glynn, J. Avorn, and T. Stürmer (2006): “Variable selection for propensity score models,” *American journal of epidemiology*, 163, 1149–1156.

Bross, I. D. (1966): “Spurious effects from an extraneous variable,” *Journal of chronic diseases*, 19, 637–647.

Dudoit, S. and M. J. van der Laan (2005): “Asymptotics of cross-validated risk estimation in estimator selection and performance assessment,” *Statistical methodology*, 2, 131–154.

Friedman, J., T. Hastie, and R. Tibshirani (2009): “glmnet: Lasso and elastic-net regularized generalized linear models,” *R package version*, 1.

Gruber, S., R. W. Logan, I. Jarrín, S. Monge, and M. A. Hernán (2015): “Ensemble learning of inverse probability weights for marginal structural modeling in large observational datasets,” *Statistics in medicine*, 34, 106–117.

Hanley, J. A. and B. J. McNeil (1982): “The meaning and use of the area under a receiver operating characteristic (roc) curve.” *Radiology*, 143, 29–36.

Hastie, T., R. Tibshirani, J. Friedman, T. Hastie, J. Friedman, and R. Tibshirani (2009): *The elements of statistical learning*, volume 2, Springer.

Ju, C., S. Gruber, S. D. Lendle, J. M. Franklin, R. Wyss, S. Schneeweiss, and M. J. van der Laan (2016): “Scalable collaborative targeted learning for large scale and high-dimensional data,” , U.C. Berkeley Division of Biostatistics Working Paper Series. Working Paper 352. `http://biostats.bepress.com/ucbbiostat/paper352`.

Kuhn, M. (2008): “Building predictive models in r using the caret package,” *Journal of Statistical Software*, 28, 1–26.

Kuhn, M., J. Wing, S. Weston, A. Williams, C. Keefer, A. Engelhardt, T. Cooper, Z. Mayer, R. C. Team, M. Benesty, et al. (2014): “caret: classification and regression training. r package version 6.0-24,” .

Lee, B. K., J. Lessler, and E. A. Stuart (2010): “Improving propensity score weighting using machine learning,” *Statistics in medicine*, 29, 337–346.

Polley, E. C. and M. J. van der Laan (2010): “Super learner in prediction,” , U.C. Berkeley Division of Biostatistics Working Paper Series. Working Paper 266. `http://biostats.bepress.com/ucbbiostat/paper266`.

Rose, S. (2016): “A machine learning framework for plan payment risk adjustment,” *Health Services Research*, 51, 2358–2374.

Rubin, D. B. (2004): “On principles for modeling propensity scores in medical research,” *Pharmacoepidemiology and drug safety*, 13, 855–857.

Schneeweiss, S., J. A. Rassen, R. J. Glynn, J. Avorn, H. Mogun, and M. A. Brookhart (2009): “High-dimensional propensity score adjustment in studies of treatment effects using health care claims data,” *Epidemiology*, 20, 512–522.

Schneeweiss, S., J. A. Rassen, R. J. Glynn, J. Myers, G. W. Daniel, J. Singer, D. H. Solomon, S. Kim, K. J. Rothman, J. Liu, et al. (2012): “Supplementing claims data with outpatient laboratory test results to improve confounding adjustment in effectiveness studies of lipid-lowering treatments,” *BMC medical research methodology*, 12.

Setoguchi, S., S. Schneeweiss, M. A. Brookhart, R. J. Glynn, and E. F. Cook (2008):

"Evaluating uses of data mining techniques in propensity score estimation: a simulation study," *Pharmacoepidemiology and drug safety*, 17, 546–555.

van der Laan, M. J., , E. C. Polley, and A. E. Hubbard (2007): "Super learner," *Statistical Applications in Genetics and Molecular Biology*, 6, Article 25.

van der Laan, M. J. and S. Dudoit (2003): "Unified cross-validation methodology for selection among estimators and a general cross-validated adaptive epsilon-net estimator: Finite sample oracle inequalities and examples," , U.C. Berkeley Division of Biostatistics Working Paper Series. Working Paper 130. `http://works.bepress.com/sandrine_dudoit/34/`.

van der Vaart, A. W., S. Dudoit, and M. J. van der Laan (2006): "Oracle inequalities for multi-fold cross validation," *Statistics & Decisions*, 24, 351–371.

Westreich, D., J. Lessler, and M. J. Funk (2010): "Propensity score estimation: neural networks, support vector machines, decision trees (cart), and meta-classifiers as alternatives to logistic regression," *Journal of clinical epidemiology*, 63, 826–833.

Wyss, R., A. R. Ellis, M. A. Brookhart, C. J. Girman, M. J. Funk, R. LoCasale, and T. Stürmer (2014): "The role of prediction modeling in propensity score estimation: an evaluation of logistic regression, bcart, and the covariate-balancing propensity score," *American journal of epidemiology*, 180, 645–655.

Appendix

Model name	abbreviation
Bayesian Generalized Linear Model	bayesglm
C5.0	C5.0
Single C5.0 Ruleset	C5.0Rules
Single C5.0 Tree	C5.0Tree
Conditional Inference Tree	ctree2
Multivariate Adaptive Regression Spline	earth
Boosted Generalized Linear Model	glmboost
Penalized Discriminant Analysis	pda
Shrinkage Discriminant Analysis	sda
Flexible Discriminant Analysis	fda
Lasso and Elastic-Net Regularized Generalized Linear Models	glmnet
Penalized Discriminant Analysis	pda2
Stepwise Diagonal Linear Discriminant Analysis	sddaLDA
Stochastic Gradient Boosting	gbm
Multivariate Adaptive Regression Splines	gcvEarth
Boosted Logistic Regression	LogitBoost
Penalized Multinomial Regression	multinom,
Penalized Logistic Regression	plr,
CART	rpart
Stepwise Diagonal Quadratic Discriminant Analysis	sddaQDA
Generalized Linear Model	glm
Nearest Shrunken Centroids	pam
Cost-Sensitive CART	rpartCost