TTML Project Final Report

Mu Li[1], Yijie Guo[2], Wenbo Jing[3], and Zichang Ye[4]

[1]ml4844@nyu.edu
[2]yg2418@nyu.edu
[3]wj2093@nyu.edu
[4]zy1545@nyu.edu

May 15, 2021

1 Introduction

Conditional Treatment Effect Estimation (CATE) is a crucial topic in causal inference and enjoys a wide application in social sciences. Meta-learners are methods to estimate such CATE functions. Kunzel et al [8] proposes X-learner and claims that it has favorable advantages over other meta-learners such as T-learners and S-learners in both empirical and simulated settings. In this project, we have four main tasks. Firstly, we aim to validate the performances of X-learners by reproducing some selected experiments in the papers. In addition, we would like to see whether the selection bias of the data would hurt the performances of the models, as claimed by the paper. Thirdly, we try to improve the existing method by discretizing and binarizing our data. Finally, we will try to replace the base learners in X-learners with more complicated functions and see whether that would improve the model's ability to learn the target function.

2 Result Reproduction

We reproduced three experiments described in the original paper: the data-inspired comparisons of metalearners using Go Out To Vote (GOTV) dataset, the simulation study 1 and 2 (referred as SI-1 and SI-2 hereafter) in the supporting materials.

2.1 Simulation with GOTV

2.1.1 Dataset

GOTV dataset comes from Gerber et al, 2008 [6]. This study shows that the voters who receive a letter indicating the voting history of their neighbors have a higher chance of voting in subsequent elections. The treatment group, the group that receives the letters, consists of 38,218 individuals, while the control group consists of 191,243 individuals.

2.1.2 Setup & Reproduction

The author first fits a T-learner to the entire data and uses the obtained CATE estimate as the ground truth for conditional treatment effect. Then, it trains the S-learner, X-learner, T-learner

with growing samples and compares their performances against a hold-out test set. In particular, we used the propensity scores predicted by a Random Forest Regressor for the weights needed in X-learner. In particular, they keep the uneven sample sizes in each group by fixing the $\mathbb{P}(W = 1) = 0.167$. Finally, they claim that X-learner outperforms both the T-learner and S-learner in this case of uneven sample size.

We noticed that the author does not specify the size of the test data in the paper. We chose to use the complement of the training data as a test set, maintaining that $\mathbb{P}(W = 1) = 0.167$. We tried to reproduce the results in both R and Python. In R, We utilized the `causalToolbox` package developed by the original author and wrote our code to perform the experiments. In Python, we used EconML and scikit-learn, which has a more efficient implementation of random forests to run the experiments with larger sample sizes.

2.1.3 Findings

Our findings are presented in Figures 1 and 2. We confirmed that X-learner performs better in terms of RMSE, variance, and bias in this setting of uneven sample sizes. However, there is two pieces of information from the study that we could not repeat.

First, the bias of the T-learner in our reproduction is consistently higher than X-learner and S-learner, while in the original paper, the bias of the T-learner is lower than the S-learner, similar to the level of X-learner. We double-checked this inconsistency by reproducing the experiment in Python. Even though we were not able to get the same results as we observed in R, the T-learner still has a higher bias than the S-learner, and the performances of the S-learner and X-learner becomes closer in the Python experiment.

Secondly, we were unable to reproduce the small magnitude of the RMSE that the paper achieved. The RMSE that the paper achieves is in the range of 0.01 to 0.03, while in our case, the RMSE is in the range of 0.1 to 0.3. Since the magnitude of the variance is about 0.02, most of the RMSE may come from the bias term. The original paper may have tuned the hyperparameters and reported the best results, while our base learner is a Random Forest Regressor with default hyperparameters.

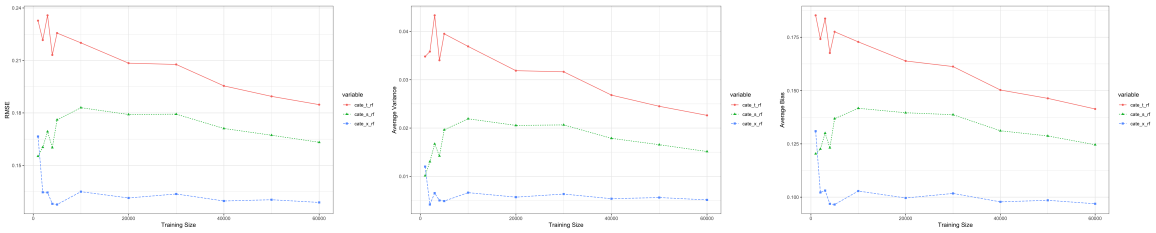

Figure 1: GOTV: The RMSE, variance, and bias of Metalearners (in R)

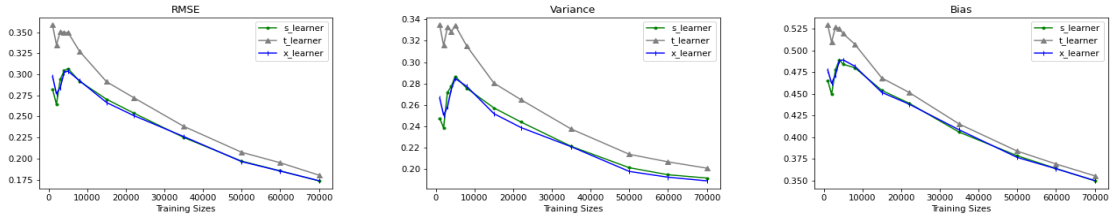

Figure 2: GOTV: The RMSE, variance, and bias of Metalearners (in Python)

2.2 Experiments with Simulated Data

In the supporting materials, the author lists out a few simulated setups to compare the performance of meta-learners using Random Forest (RF) and Bayesian Additive Regressions Trees (BART) as base learners.

2.2.1 Setup

According to the author, there are three steps of generating the simulated data.

1. Generate features data from a multi-variate normal distribution, namely $X_i \overset{iid}{\sim} N(0, \Sigma)$, where Σ is the random covariance matrix generated by vine[9] methods.

2. Create the potential outcomes for each data point:

$$Y_i(1) = \mu_1(X_i) + \epsilon_i(1)$$

$$Y_i(0) = \mu_0(X_i) + \epsilon_i(0)$$

   where $\epsilon_i(1), \epsilon_i(0) \overset{iid}{\sim} N(0, 1)$ and are independent of X_i. Each simulated setup will have a different $\mu_1(\cdot)$ and $\mu_0(\cdot)$.

3. Assign the treatment according to $W_i \sim Bern(e(X_i))$.

We attempted to reproduce SI-1 and SI-2. In SI-1, we are looking at a unbalanced case with a relatively simple target CATE function. In particular, the treatment effect is $8 \cdot \mathbb{1}(x_2 > 0.1)$.

$$e(x) = 0.01, d = 20$$

$$\mu_0(x) = x^T\beta + 5 \cdot \mathbb{1}(x_1 > 0.5), \beta \sim Unif([-5, 5]^d)$$

$$\mu_1(x) = \mu_0(x) + 8 \cdot \mathbb{1}(x_2 > 0.1)$$

In SI-2, we are looking at a balanced case with a complicated target CATE function in the sense that it is not at all smooth or follow regularity conditions.

$$e(x) = 0.5, d = 20$$

$$\mu_0(x) = x^T\beta_0, \beta_0 \sim Unif([1, 30]^d)$$

$$\mu_1(x) = x^T\beta_1, \beta_1 \sim Unif([1, 30]^d)$$

In the implementation, we used up to 12,000 samples to train and 500 samples to test for studies with BART as base learners, constrained by the RAM and time required to train and use a BART model. For studies with RF as base learners, we used up to 150,000 samples for training and 100,000 for testing.

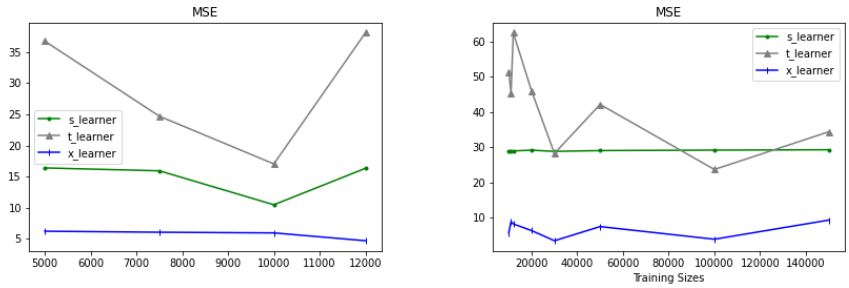

Figure 3: Simple Target (SI-1) using BART (Left) and Random Forest (Right) as base learners

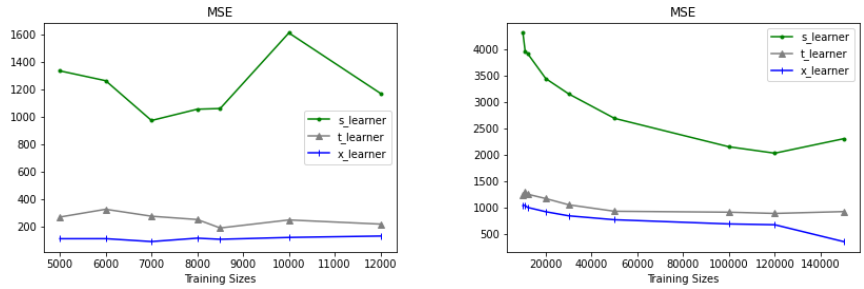

Figure 4: Complicated Target (SI-2) using BART (Left) and Random Forest (Right) as base learners

2.2.2 Results

For both simulated studies, we used the EconML library developed by Microsoft to build the meta-learners. For the base learners, we used scikit-learn to build the random forests, and bartpy to build the BART. Similar to the GOTV experiment, we used the propensity scores predicted by a Random Forest Regressor for the weights needed in X-learner. With the simulated data, we can confirm the overall claim by the paper that for a relatively simple target CATE function and a highly unbalanced setup, X-learner performs better than both S-learner and T-learner. Figure 3 shows the same results by using default Random Forest Regressor and BART Regressor as the base learner. In particular, in the Random Forest case, the S-learner and the T-learner have a similar trajectory confirmed in our reproduction. For BART trained on relatively smaller training samples than the samples we used for Random Forest, the takeaway still holds.

For SI-2, we could reproduce results very similar to the original work as Figure 4 shows. We noticed that S-learner does not perform well for a very complicated linear function, while X-learner is better. There are two inconsistencies we noticed. One is that the T-learner performs very well with both BART and RF as base learners, while in the supporting materials, T-learner is not as good as S-learner with BART. We think our solution makes sense, as in the setting of SI-2, the response functions for treatment and control groups are very different, so T-learners that estimate a function for each group separately should perform better than S-learners. Another mismatch is that the author claims that the selection of the base learner matters, while we have found similar patterns of performances of three meta learners with either BART or RF.[1]

[1]We are intrigued by why X-learners are still performing the best in the case where the treatment is more even in the SI-2. Two scenarios where the X-learners are better, according to the author, are their ability to adapt to uneven treatment/control ratio and structural properties of the functions. But neither of these two are satisfied in this setting, as there is virtually no information that one in the treatment group can learn about the control group.

3 Selection Bias

[1] made an argument that X-Learner fails to consider the effect of selection bias, but did not conduct any experiment to prove the argument. In this section, we will conduct experiments to check the validity of this argument. We randomly generate 100 seeds to randomly split the data without any assumption and condition. We use half the data as a training set and half the data as a testing set. The calculation of ground truth follows the same procedure described in section 2.1.2. The result is shown in Figure 5.

Invariant to the base algorithm, the X-Learner performances are very stable across 100 different data split. We will then use the results in Figure 5 as references to test the results in man-made datasets that contain a clear selection bias problem. The result is shown in the Table 1

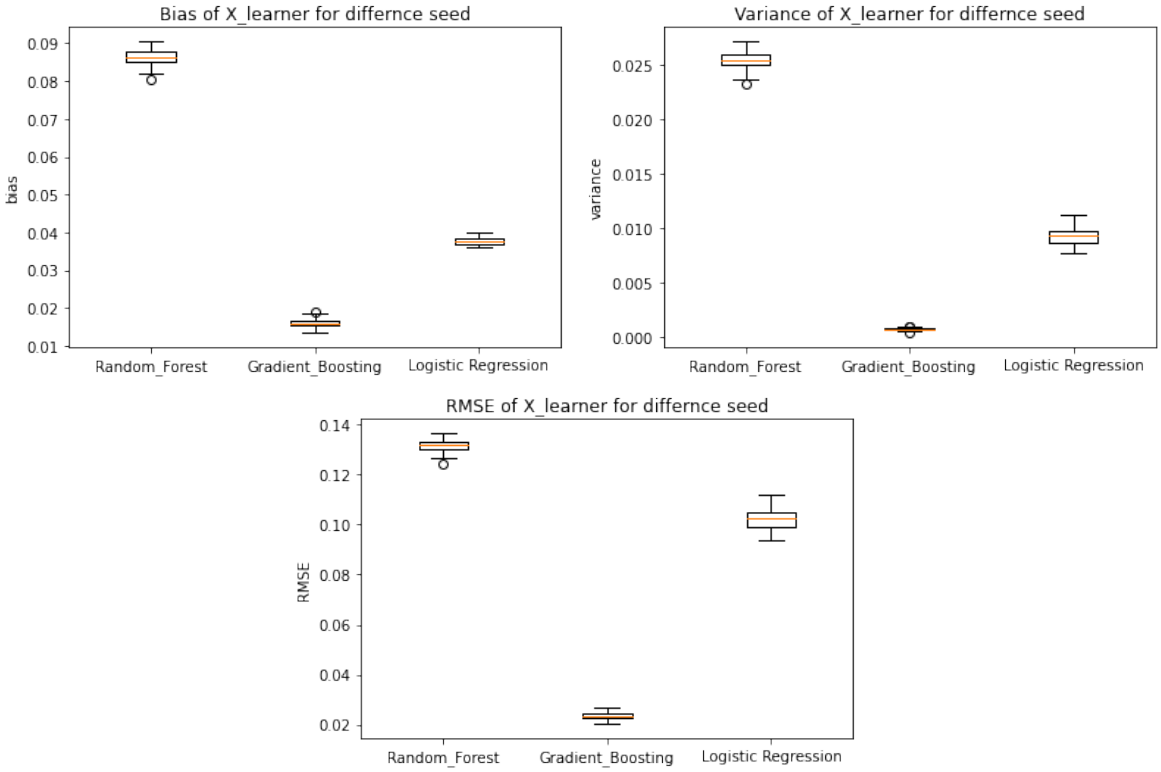

Figure 5: Bias, Variance and RMSE of X-Learner for different base algorithm for 100 data split

	RF(M)	RF(E)	GB(M)	GB(E)	LR(M)	LR(E)
Bias	0.1045	0.1378	0.0174	0.0268	0.0157	0.0264
Variance	0.0295	0.0392	0.0007	0.0024	0.0030	0.0006
RMSE	0.1584	0.1953	0.0250	0.0474	0.1181	0.1612

Table 1: Bias, Variance and RMSE of X-Learner for Mild Selection Bias(M) dataset and Extreme Selection Bias(E) dataset

In the Mild Selection Bias dataset, the dataset is split into two groups: the people who have been voted twice in 2002 and people who at least miss one vote in 2002. 80% of the former group goes into the training set, and 30% of the latter group goes into the training set. The rest goes

to the testing set. In the Extreme Selection Bias dataset, 99% of the people who voted four times in 2000 and 2002 goes into the training set, while only 1% of the people who never voted in 2000 and 2002 goes into the training set. For the remaining people, 80% of the people who are younger than 50 years old goes into the training set, and 20% of the people who are older than 50 years old goes into the training set. The rest goes into the testing set. Both datasets roughly have the same number of samples in the training set and testing set to match the setting in the first experiment.

For the result, we observe that only the bias of random forest increase in the Extreme Selection Bias dataset. Gradient Boosting and Logistic Regression are immune selection bias problems in the X-Learner setting. Overall, the above experiments prove that X-Learner is very robust and can perform well under selection bias circumstances.

4 Discretization and Binarization

In the real world, data generally tends to be noisy, missing, and inconsistent due to their huge size and possible multiple, heterogeneous sources, leading to low-quality data mining results. Therefore, analysts sometimes would preprocess data, such as data transformation, data cleaning, data integration, data reduction, etc[3]. In this section, we would experiment with whether the data transformation, the discretization and the binarization process, could reduce the bias and improve the confidence interval coverage for S-Learner, T-Learner, and X-Learner using random forests (RF) and bayesian additive regression trees (BART).

4.0.1 Discretization

We used a histogram to categorize continuous values into discrete integers, which depended on the number of bins in the histogram. If more bins were used in the histogram, the original floating-point values would be represented more accurately in the resulting discrete dataset [5].

4.0.2 Binarization

We used a unique technique, binary trees, to convert continuous values into binary digits. These digits indicated the intervals where the continuous values fell into. Each branching was decided by either the mean or median of the subsets within that interval. In addition, the height of the binary trees would decide the accuracy and the information loss of the resulting binary dataset. Therefore, we could transform the floating-point dataset into a hierarchical structure, which could discover semantical and meaningful hidden patterns[5].

4.0.3 Bias Reduction

We reproduced the bias and RMSE experiments on the GOTV data. All S-Learner, T-Learner, and X-Learner used RFs and BARTs on the original floating-point data, the discrete integer data and the binary bit data. Due to the computation power, limited memory size and training time, we estimated the biases and RMSEs from 1000 training size and 50 testing size up to 20000 training size and 1000 testing size. The bias and RMSE of S-Learner, T-Learner and X-Learner using the original, discrete and binary data are shown in the Figure 6, Figure 7 and Figure 8.
From the figure, we could see that the discrete data and the binary data increased the RMSE due to the increment of the biases. Each feature of these datasets covered a range of values instead of one accurate floating-point value. The discretization and binarization could only reduce the noises. This noise reduction would be beneficial for models that were sensitive towards noises, like linear

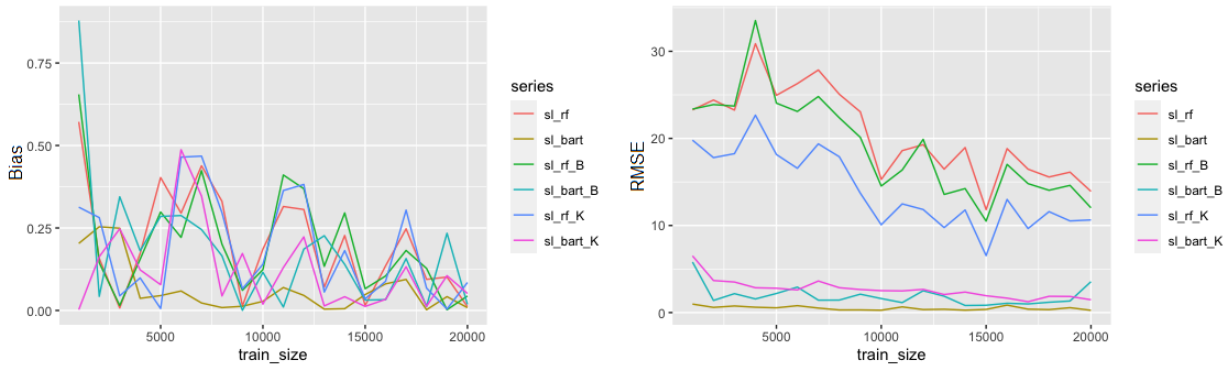

Figure 6: Bias and RMSE of S-Learner with original R, discrete K data and binary B data

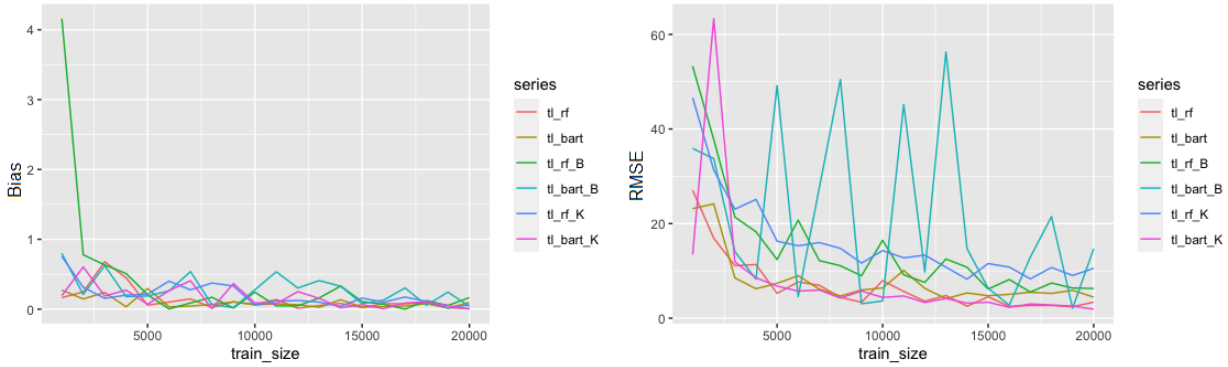

Figure 7: Bias and RMSE of T-Learner with original R, discrete K data and binary B data

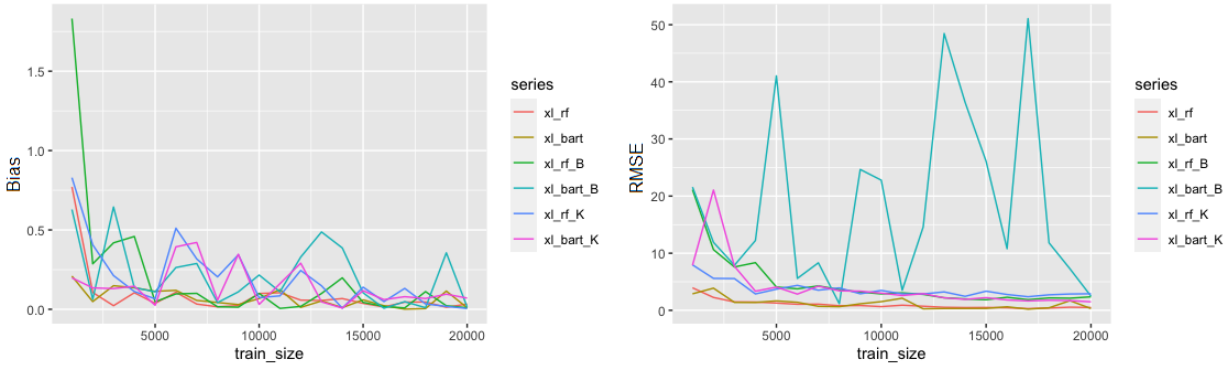

Figure 8: Bias and RMSE of X-Learner with original R, discrete K data and binary B data

models. Nonetheless, the non-linear tree-based algorithms, like RF and BART, would not take this advantage from the discrete data and the binary data. These two models were robust towards noises. [7].

4.0.4 Confidence Interval (CI) Coverage

As the author mentioned in their paper, the estimation of the bootstrap confidence interval required massive computation power and huge memory consumption. Due to the limited resources, we could

not fully replicate the original confidence interval simulation with the same setting. Our experiments used 10,000 training examples and 500 testing examples running 100 bootstrap sampling to estimate confidence interval. The CI coverage experiment results are shown in Figure 9. Because previously

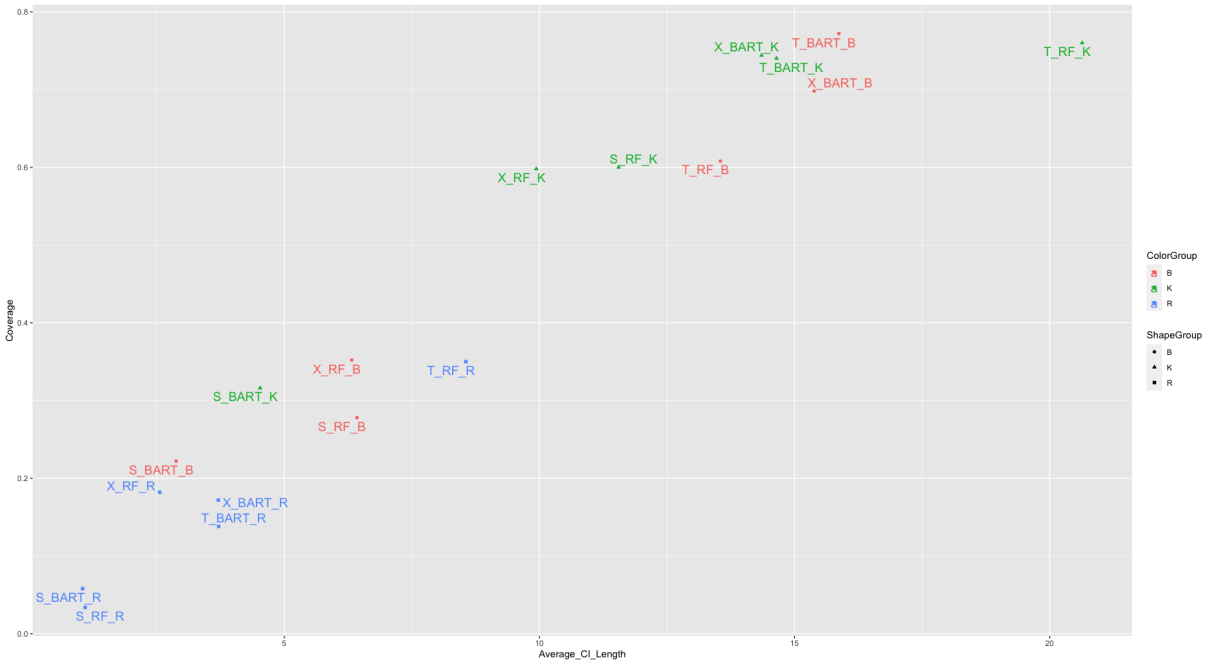

Figure 9: Confidence Interval Coverage with the original R, discrete K and binary B data

we had learned the discretization and binarization would not reduce the biases, we could expect that the discrete and binary dataset would not exceed the original floating-point dataset in achieving the ideal 95% CI coverage. The results proved our suspicion that none of these learners with corresponding datasets achieved the target. Also, the linearity of the confidence interval coverage and the average confidence interval length matched the results from the experiment of the original paper.

4.0.5 Bias and Confidence Interval Experiment Summary

The discretization and binarization did not reduce the bias and improve RMSE score, which was probably because the learners used the non-linear models, the RF and BART, that the feature transformation preprocess did not have huge impact on these tree-based models. Thus, the learners using the original floating-point data had already achieved their state-of-the-art performance. On the other hand, the bootstrap sampling still suffered the bias from the CATE estimators to produce the 95% CI coverage. Therefore, the discretization and binarization would not adjust the bias, improve the learners using RF and BART and provide the target CI coverage.

5 Meta-learners with different base algorithms

In [8], the base algorithms meta-learners based on are chosen to be honest Random Forest (RF) [2] and Bayesian Additive Regression Tree (BART) [4], which are both tree-based algorithms combining multiple tree models. It's well known that RF uses randomization to build a set of single tree models and aggregates them by linear averaging. Similarly, BART fits an additive model by summing up

multiple singles trees, each constrained by a specific prior and fitted by an iterative Bayesian backfitting MCMC algorithm. Generally speaking, RF and BART both belong to ensemble tree-based methods, which achieves complex model structure by synthesizing and strengthening the results from weaker learners. In this section, we wish to study the effect of these two ensemble tree-based methods by doing experiments using meta-learners with simpler non-ensemble base algorithms.

Besides BART and RF, two more learners are considered in our experiments, Support Vector Machine (SVM) and Neural Networks (NN). We believe they are both popular and can represent typical types of machine learning algorithms. Technically, SVM only stands for a classifier. However, in regression tasks there is a corresponding learner called Support Vector Regression (SVR). They are both packed into the "svm" function in the R package `e1071`. We won't distinguish them in the following part. SVM could represent the basic machine learning algorithms with a linear structure, fitting an additive model (a linear combination of support vectors) by optimizing the parameters over a constrained set. Moreover, one could introduce kernel methods into SVM to capture non-linear structures. In practice, we use the most popular radial basis kernel $K(x, x') = e^{-(x-x')^2}$. Neural Networks are also widely used in the deep learning world. Unfortunately, we are not able to study a complex deep neural network due to computing complexity. Instead, we use a two-layer network, each layer with the same dimension as the feature space and a sigmoid activation function. Other hyper-parameters are set to be the default ones in the R package `neuralnet`. We compare BART, RF, SVM, and NN through the following two experiments, including the aforementioned GOTV data and another simulation study.

5.1 Real data: GOTV

The first experiment is about estimating the treatment effect of the GOTV data. As suggested by [8], we fit a Random Forest-based T-learner on the whole GOTV dataset and view its result as the ground truth of the treatment effect. This approximation is reasonable considering the large size of the GOTV dataset and relatively low-dimensional feature space (i.e. $N \gg p$). Then from the whole dataset, we randomly sample a relatively small subset with size n, train meta-learners with different base learners on it, and estimate the generalizing ability by computing Prediction Mean Squared Error (PMSE) on a test set of fixed size 2000. We repeat 50 times for each n and show the boxplot in Figure 10.

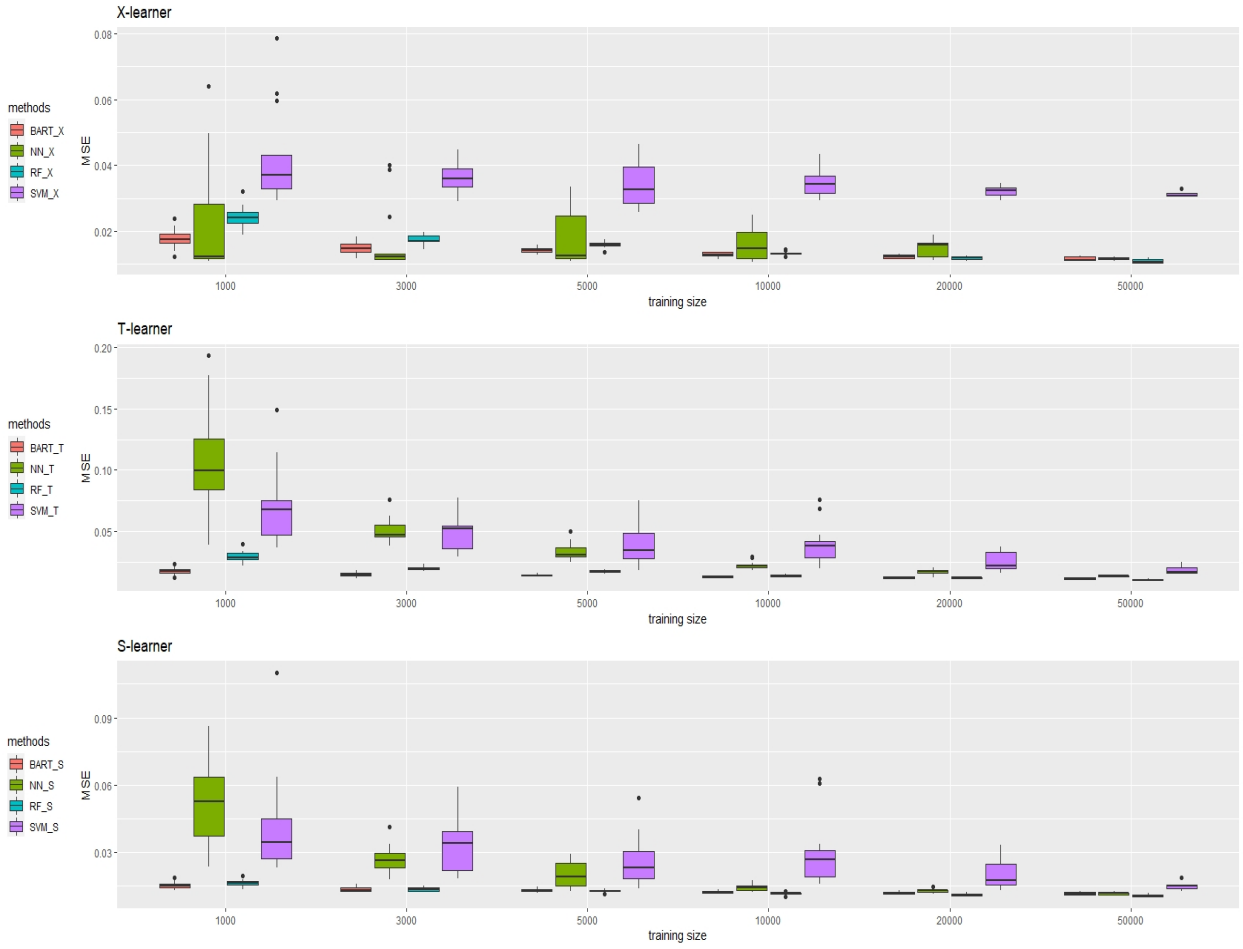

Figure 10: Boxplot of the PMSEs of X-learner, T-learner and S-learner with different base algorithms (BART, RF, NN and SVM) fit on sampled GOTV dataset. The x-axis stands for the training size n.

It is obvious from Figure 10 that for all the three meta-learners, the BART and RF based ones show stronger stability against NN and SVM, which mainly because that ensemble pattern reduces the variance by linear averaging. When the sample size is large enough, the PMSEs of all the learners converge to a small value, except for SVM-based X-learner. However, when the sample size is small, BART and RF have an advantage in both stability and accuracy. In some cases, such as $n = 1000$, although the mean value of the PMSE of Neural Network based X-learner is lower than that of BART and RF based X-learner, we still prefer the latter because the Neural Network based X-learner suffers from high variance.

5.2 Simulated data: linear functions with non-linear treatment effect

The previous experiment on the GOTV dataset uses a binary response: vote or not. To test a more general case, we simulate a regression setting analogous to SI2 mentioned before with some small modification:

$$Y(0) = X\boldsymbol{\beta} + \varepsilon,$$
$$Y(1) = X\boldsymbol{\beta} + I(X_1 > 0) + \varepsilon,$$

where $X = (X_1, X_2, X_3, X_4)$, $X_i \sim N(0,1)$, $\boldsymbol{\beta} = (\beta_1, \beta_2, \beta_3, \beta_4), \beta_i \sim Unif(-1,1)$ and $\varepsilon \sim N(0, 0.1)$. $I(\cdot)$ denotes the identical function and the true treatment effect in this setting is $I(X_1 > 0)$. Note that in this setting, the implicit structure is a linear combination of continuous variables, while the treatment effect is binary. This could test both the classification and regression ability of the learners. We set the treatment probability to be 0.2, and do the same procedure as in the previous section. The results are shown in Figure 11. The neural network still suffers from high variance, but a main difference is that SVM performs much better than before, because the underlying structure becomes simpler for it to learn. Another notable thing is for S-learner, BART is shown to be the worst base learner in our setting, no matter how large the training size is. The possible reason is that when the treatment variable W is included into the feature space (which is how S-learner works), BART is hard to capture the difference between $W = 1$ and $W = 0$ when $\mathbb{P}(W = 1)$ is small.

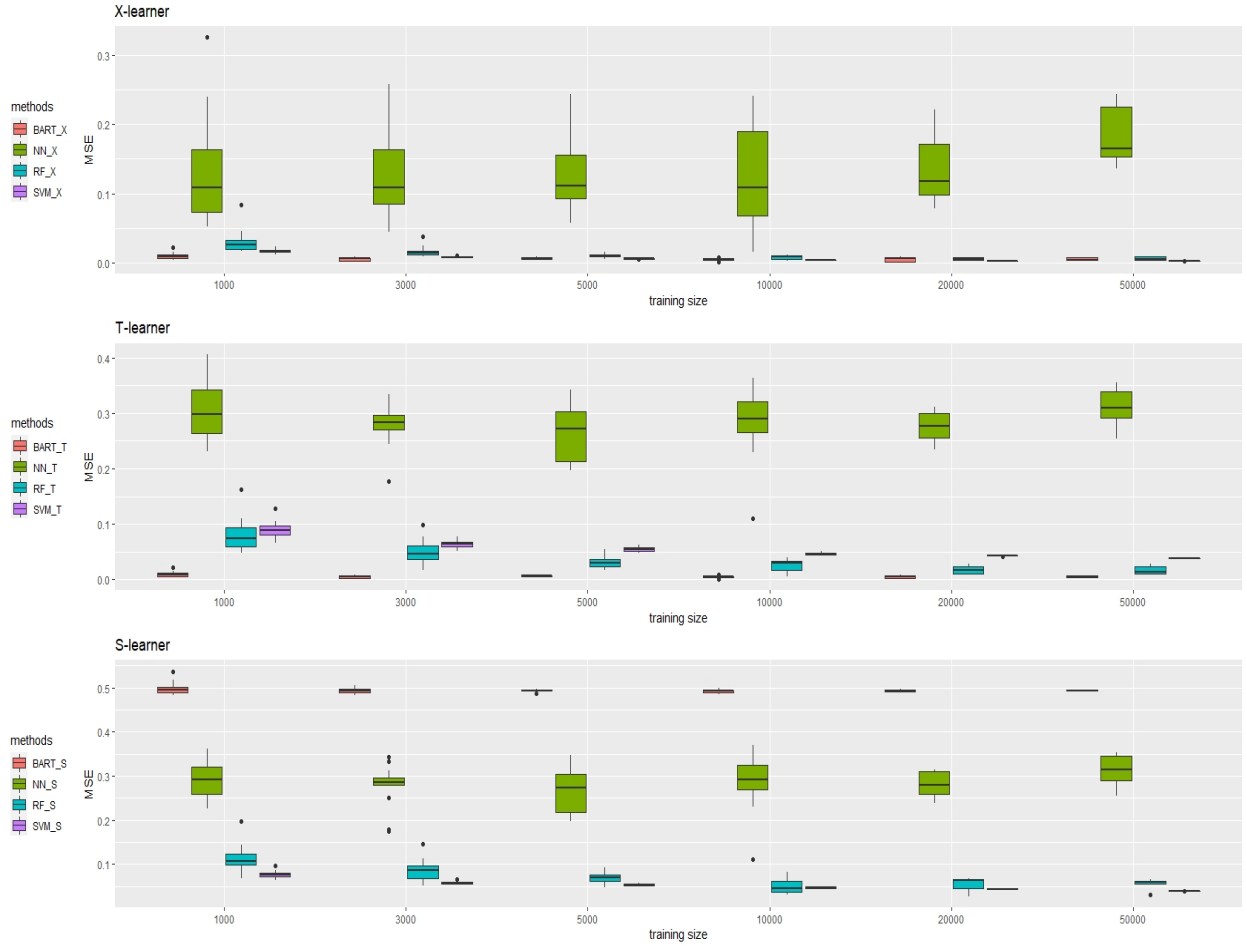

Figure 11: Boxplot of the PMSEs of X-learner, T-leanrer and S-learner with different base algorithms (BART, RF, NN and SVM) fit on simulated dataset described in section 5.2. The x-axis stands for the training size n.

6 Conclusion

In a nutshell, we confirmed most of the arguments claimed by the author and the good performance of X-learners in both synthetic and empirical data. However, we found inconsistencies regarding the comparative behaviors of the T-learners and S-learners in GOTV and SI-2. Part of the discrepancies might come from the different choices of hyperparameters, yet a deeper dive and more experiment with the meta-learners are needed to make a solid explanation. Also, we had shown that X-Learner performs really well against selection bias, invariant to the base algorithm. Next, neither the discretization nor binarization process reduced the bias and the RMSE for learners using the tree-based RF and BART to improve their performance using the original floating-point data. Additionally, further experiments showed that the transformed data could not boost confidence interval coverage for all learners due to the unadjusted biases. Finally, our experiments using different base learners verified the benefit of the ensemble algorithms (BART and RF) in enhancing the stability and accuracy, compared to SVM and Neural Networks, especially when the sample size was not large enough.

References

[1] Ahmed Alaa and Mihaela Schaar. Limits of estimating heterogeneous treatment effects: Guidelines for practical algorithm design. pages 129–138, 2018.

[2] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[3] Soumen Chakrabarti, Earl Cox, Eibe Frank, Ralf Hartmut Gting, Jiawei Han, Xia Jiang, Micheline Kamber, Sam S. Lightstone, Thomas P. Nadeau, Richard E Neapolitan, Dorian Pyle, Mamdouh Refaat, Markus Schneider, Toby J. Teorey, and Ian H. Witten. *Data Mining: Know It All*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2008.

[4] Hugh A Chipman, Edward I George, Robert E McCulloch, et al. Bart: Bayesian additive regression trees. *The Annals of Applied Statistics*, 4(1):266–298, 2010.

[5] James Dougherty, Ron Kohavi, and Mehran Sahami. Supervised and unsupervised discretization of continuous features. In *Machine learning proceedings 1995*, pages 194–202. Elsevier, 1995.

[6] Alan S Gerber, Donald P Green, and Christopher W Larimer. Social pressure and voter turnout: Evidence from a large-scale field experiment. *American political Science review*, pages 33–48, 2008.

[7] Muhammad Nur Fikri Hishamuddin, Mohd Fadzil Hassan, and Ainul Akmar Mokhtar. Improving classification accuracy of random forest algorithm using unsupervised discretization with fuzzy partition and fuzzy set intervals. In *Proceedings of the 2020 9th International Conference on Software and Computer Applications*, ICSCA 2020, page 99–104, New York, NY, USA, 2020. Association for Computing Machinery.

[8] Sören R Künzel, Jasjeet S Sekhon, Peter J Bickel, and Bin Yu. Metalearners for estimating heterogeneous treatment effects using machine learning. *Proceedings of the national academy of sciences*, 116(10):4156–4165, 2019.

[9] Daniel Lewandowski, Dorota Kurowicka, and Harry Joe. Generating random correlation matrices based on vines and extended onion method. *Journal of Multivariate Analysis*, 100(9):1989–2001, 2009.