Supplement to "Training Replicable Predictors in Multiple Studies"

Prasad Patil[a,b] and Giovanni Parmigiani[a,b]

[a]Department of Biostatistics and Computational Biology, Dana-Farber Cancer Institute
[b]Department of Biostatistics, Harvard T.H. Chan School of Public Health

November 10, 2017

This document serves as a supplement to the main article "Training Replicable Predictors in Multiple Studies". Here, we present two supplementary analyses: illustrating the performance of cross-study learning approaches when the single-study learners are pre-trained; and training predictors for a classification task (debulking). We also present additional simulation results, including sensitivity analyses for the number of genes chosen per study and different assumptions on amounts of inter-study heterogeneity and error distributions.

1 Combining published predictions functions

Waldron et al.[4] conducted a systematic review and independent recoding of prognostic gene expression signatures for late-stage, high-grade, serous ovarian cancer using overall survival as the endpoint. Their work provides 14 prognostic models that passed their study inclusion criteria and could be independently implemented. These are used in our analysis leading to Figure 1.

We illustrate that it is possible to combine published prediction functions so that the result shows systematically better replicability. We train on a set of studies and compare three types of averages, one of which uses weights directly engineered to reward cross-study replicability.

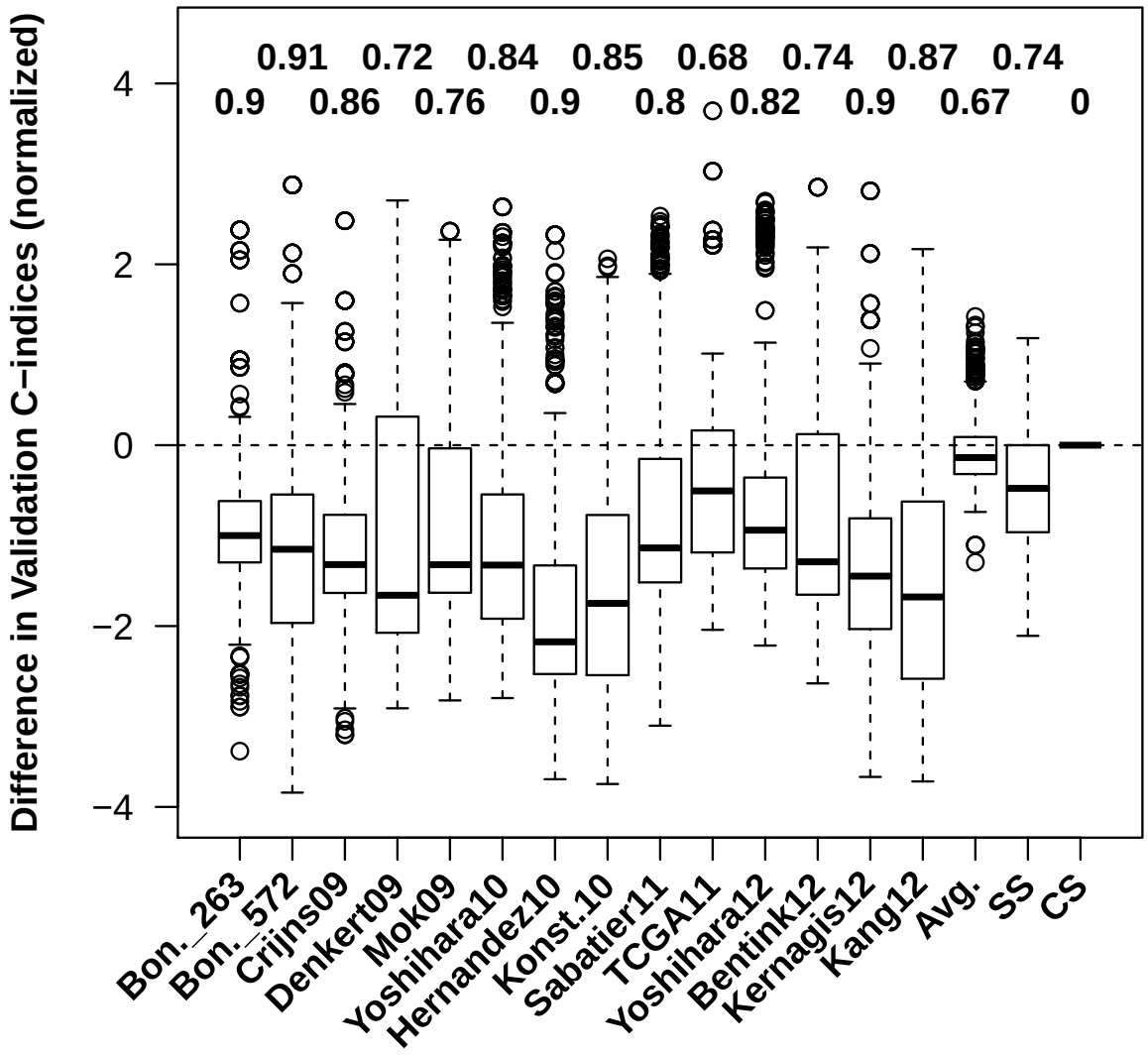

Figure 1: Distribution of the difference between the normalized C-index of each signature or other averaging approach and the performance of the CS-weighted ensemble in the validation studies. Boxplots reflect variation across simulations. Above each boxplot is the proportion of times the corresponding algorithm performs worse than the CS-weighted ensemble. Labels of published signatures follow[4].

Specifically, we examined a set of fourteen published prognostic algorithms for ovarian cancer, and fifteen sets of patient data. Both the prognostic algorithms and the data sets are identified through a comprehensive literature review (see methods). For comparability, we work with the risk ranks provided by each algorithm. We repeatedly ($R = 250$) split the fifteen studies into groups of twelve training and three validation studies. At each iteration, we form three combined scores: the average rank, the average rank weighted by the original training study sample size, and the average rank weighted with replicability weights (see methods in

main manuscript), giving higher weights to prediction functions that exhibit good cross-study performance within the training studies. We then evaluate the performance of each original algorithm, as well as the three combined ones, in the held-out validation sets. Because Y is a time-to-event outcome and we wish to work with ranks, our evaluation criterion is the time-to-event C-index [3]. When a validation dataset was originally used to train one of the published algorithms, that algorithm was excluded from that iteration. Because all signatures cannot be validated on all datasets, we normalize the C-indices within each iteration. This makes performance of individual signatures more comparable: suppose, for example, that the TCGA dataset is difficult to make predictions on. Then TCGA's unnormalized performance may look better than that of other signatures which have the difficult task of making predictions on the TCGA dataset.

The results are summarized in Figure 1. The ensembles generally outperform the original signatures. The CS-weighted ensemble specifically also outperforms the plain average (Avg) and the sample size weighted average (SS). The relative performances of each signature mirror those detailed in Waldron et. al. [4]. Sample size weighting is penalized by some predictors trained on large training sets which happen to exhibit poor cross-study performance. For example Kang12 was trained on 511 samples but imposes strong *a priori* restrictions on available predictors. This conclusion is likely to be specific to this case study and may not generalize. Outliers in this plot are caused by repeat observations of certain signature-validation dataset pairs.

This illustration suggests that we can achieve systematic gains in replicability by embedding cross-study performance in training steps. A major limiting factor here is working with pre-existing algorithms as building blocks.

Cross-study learning in ovarian cancer gene expression data

We return to the ovarian cancer gene expression datasets with the goal of developing a CSL to predict a clinical variable known as debulking status, reflecting the likelihood of remaining tumor tissue after initial surgery [2]. We consider the eleven ovarian cancer datasets which had mostly complete debulking records for each patient. Similar to the simulation study, the eleven studies were randomly reassigned as eight training and three validation studies $R = 200$ times so that a sufficient number of possible reassignments are realized. The gene features in each dataset were reduced to the intersection across all datasets: 7,655 genes. For ease of computation, we implemented a simple feature selection step whereby only the top twenty genes as ranked by the absolute T-test statistic were used as candidates. We compare four training options: merged, simple weights, sample size weights, and cross-study weights. We use CART and Random Forest as learners. Predictions are made as class probabilities and averaged accordingly.

	Merged	Simple Average	SS-weighted
SSL: CART	89.5	59.5	77.5
SSL: Random Forest	68.0	57.0	46.0

Table 1: Predicting debulking status: percentage of iterations with worse performance than CS weighting.

Table 1 summarizes the results. We only use the learners we considered in the simulation setting which

also learn classification rules: CART and Random Forest. CSLs perform better than the merged learner in the vast majority of cases. The median difference in performance between each learner and the CS-weighted learner is generally small when the SSL is Random Forest (median 0 to 1% or less) and slightly larger when the SSL is CART (median 0.5 to 4%), so the advantage gained by CS-weighting is very small across the board. We can relate this real-data scenario to the simulations in Figure 3 of the main text when the coefficient perturbation windows were larger. Under substantial inter-study heterogeneity, ensembling is more beneficial than merging, and using CS weighting offers an additional, though modest, gain in performance over plain averaging. We note that for Random Forest, weighting by sample size slightly outperforms CS weighting. This is due to the fact that the Random Forest SSLs perform very similarly across studies. As a result there is more variability in the study sample sizes than the cross-study weights, and SSLs trained on larger studies in this setting tend to perform better. Although CS weighting outperforms simple averaging, the differences are small for the same reason in this example.

2 Additional Simulation Results

2.1 Full result tables for Figure 3

We mention that the results for regression-based weighting excluding an intercept and for normalization of coefficients were not included in Figure 3. Here we present the full result tables for each panel including columns for the excluded weighting schemes. Each numerical entry is the average RMSE in the validation datasets averaged across 100 simulation iterations. The columns for each table refer to: merging; unweighted average; sample size weighted average; cross-study weighted average; stacked regression without ntercept; stacked regression without intercept and normalized coefficients; stacked regression with intercept; study-specific regression weight average without intercept; study-specific regression weight average without intercept and normalized coefficients; study-specific regression weight average with intercept. Each row corresponds to a different coefficient perturbation window.

Window	Merged	Avg.	n-weighted	CS	Stack_noint	Stack_norm	Stack_int	SS_noint	SS_norm	SS_int
0.25	1.14	1.20	1.18	1.18	1.13	1.17	1.13	1.14	1.19	1.14
1	1.31	1.27	1.33	1.21	1.25	1.30	1.25	1.21	1.28	1.21
2	2.91	2.17	2.84	1.53	2.65	2.68	2.65	2.20	2.26	2.20
10	5.15	3.95	4.97	2.31	4.80	4.65	4.80	4.02	3.96	4.02

Table 2: Lasso

Window	Merged	Avg.	n-weighted	CS	Stack_noint	Stack_norm	Stack_int	SS_noint	SS_norm	SS_int
0.25	8.13	7.40	7.47	7.38	6.45	7.51	6.44	6.46	7.78	6.45
1	7.79	7.08	7.19	7.03	6.08	7.18	6.07	6.09	7.47	6.09
2	8.70	7.31	7.56	7.42	6.81	7.43	6.80	6.64	7.68	6.64
10	8.96	7.21	7.73	7.14	7.38	7.29	7.38	6.98	7.39	6.98

Table 3: CART

Window	Merged	Avg.	n-weighted	CS	Stack_noint	Stack_norm	Stack_int	SS_noint	SS_norm	SS_int
0.25	1.71	3.40	2.98	3.04	2.48	2.52	2.48	3.17	3.03	3.16
1	1.75	3.14	2.81	2.82	2.41	2.44	2.41	3.00	2.89	3.00
2	4.66	4.15	4.30	3.65	3.81	3.84	3.81	4.08	4.01	4.08
10	8.29	5.48	6.16	4.00	5.21	5.11	5.21	5.47	4.92	5.47

Table 4: Neural Network

Window	Merged	Avg.	n-weighted	CS	Stack_noint	Stack_norm	Stack_int	SS_noint	SS_norm	SS_int
0.25	6.42	5.48	5.65	5.45	4.42	5.41	4.42	4.42	5.44	4.43
1	6.37	5.31	5.47	5.28	4.03	5.28	4.03	4.04	5.29	4.04
2	6.50	5.29	5.72	5.23	4.07	5.32	4.07	4.00	5.35	4.00
10	7.59	6.20	7.27	5.54	5.93	6.00	5.93	5.72	5.93	5.72

Table 5: Mas-o-Menos

Window	Merged	Avg.	n-weighted	CS	Stack_noint	Stack_norm	Stack_int	SS_noint	SS_norm	SS_int
0.25	5.25	6.88	6.43	6.52	3.87	6.60	3.87	5.78	6.95	5.78
1	5.06	6.63	6.22	6.26	3.76	6.38	3.75	5.55	6.74	5.55
2	5.48	6.89	6.56	6.53	4.78	6.76	4.78	5.84	7.23	5.84
10	6.05	7.00	6.78	6.71	6.93	7.08	6.92	6.04	7.47	6.04

Table 6: Random Forest

Window	Merged	Avg.	n-Weighted	CS	Stack_noint	Stack_norm	Stack_int	SS_noint	SS_norm	SS_int
0.25	3.34	3.79	3.56	3.52	2.20	3.56	2.20	2.27	3.63	2.27
1	3.58	4.01	3.76	3.70	2.30	3.78	2.30	2.39	3.89	2.39
2	4.18	3.90	4.03	3.69	3.13	3.99	3.13	2.82	3.95	2.82
10	5.71	4.66	5.37	4.13	5.73	5.18	5.73	4.45	4.64	4.45

Table 7: Boosting

Window	Merged	Avg.	n-weighted	CS	Stack_noint	Stack_norm	Stack_int	SS_noint	SS_norm	SS_int
0.25	1.14	3.29	3.27	2.14	1.13	1.70	1.13	2.20	2.58	2.20
1	1.33	3.46	3.43	2.31	1.27	2.05	1.27	2.35	2.72	2.35
2	4.01	3.69	3.90	3.23	3.00	3.74	3.00	3.00	3.05	3.00
10	7.89	4.51	5.03	3.93	5.34	5.22	5.34	4.53	4.55	4.53

Table 8: L-C-N-M

Generally, including an intercept improves both types of regression-based weights very slightly or has no effect. Normalizing the regression-based weights has an unpredictable effect and varies across learners as well. We note that adding the constraint of regression coefficients summing to one to the existing non-negativity constraint already imposed in our coefficient estimation is a worthwhile future direction to explore.

2.2 Sensitivity Analysis

In the simulation result in the main text, we use a randomly selected subset of 20 genes or fewer within each dataset to create the data-generating model. We show in Figures 2 and 3 the same figure for a subset of 40 and 100 genes, respectively. The results differ somewhat but the overall message is consistent with the figure in the main text. Namely, the behavior of ensembling strategies varies by learner and we continue to see a trend of merging and stacked OLS weighting performing well at low levels of heterogeneity within certain learners. The 20 gene subset we use for the main figure is merely for convenience of computation of this illustrative example and does not represent a necessary gene selection strategy.

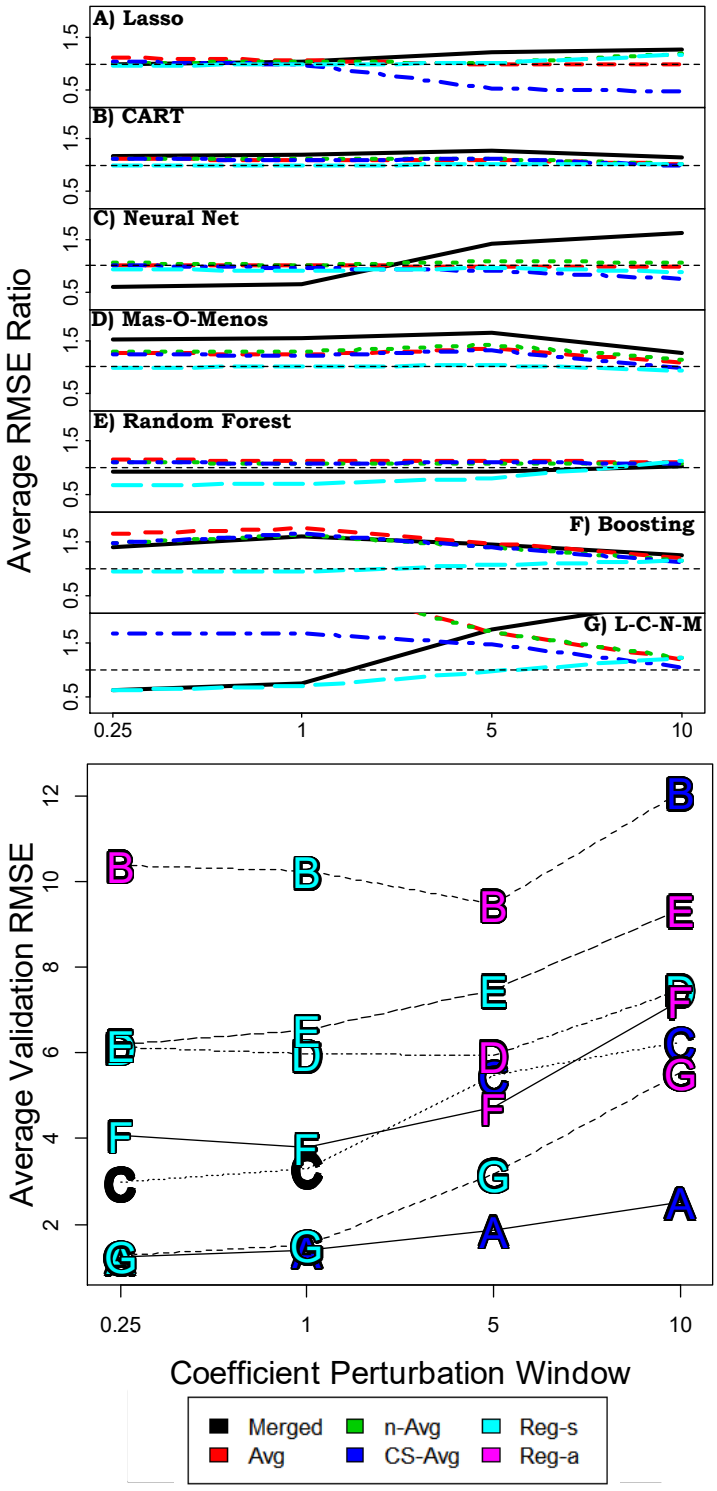

Figure 2: Linear relationship, normal errors, low-perturbation validation, 40 gene model.

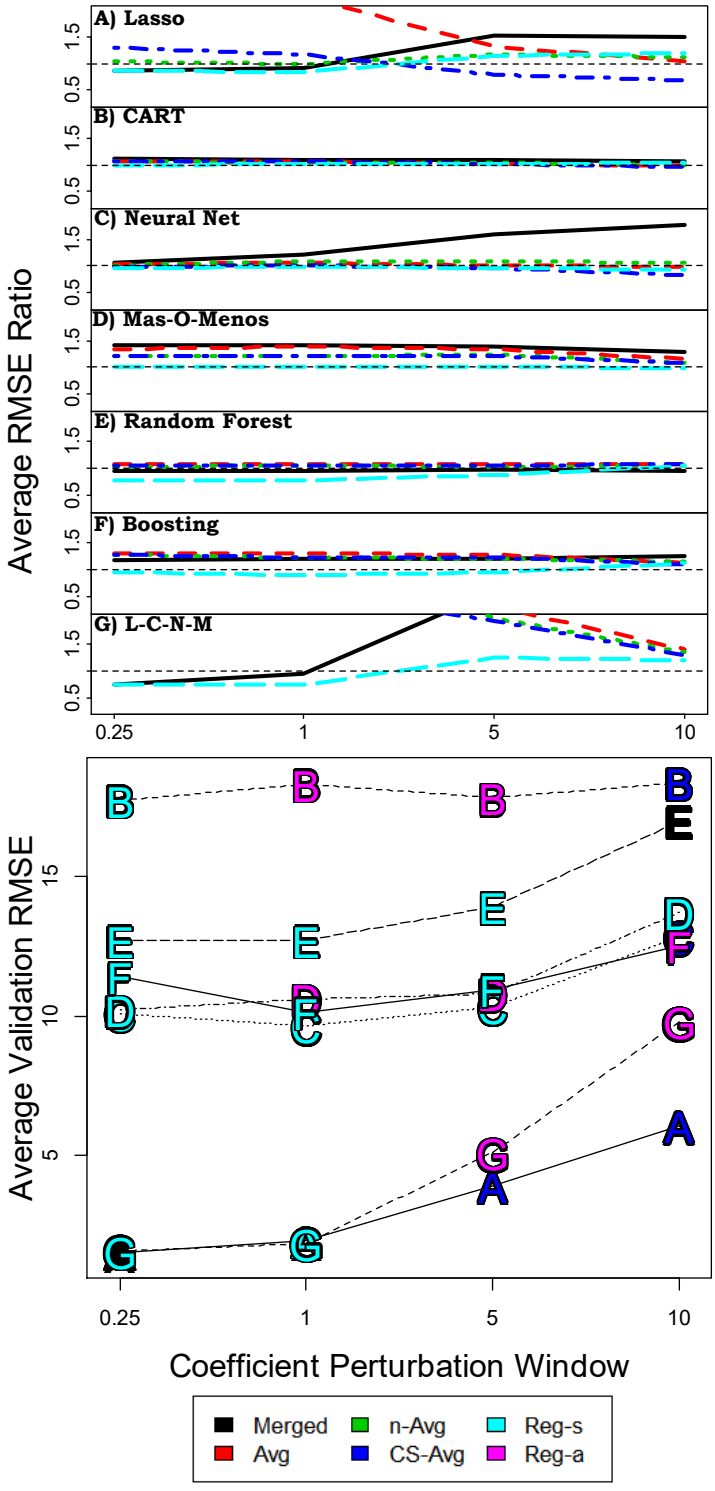

Figure 3: Linear relationship, normal errors, low-perturbation validation, 100 gene model.

2.3 Additional simulation settings

In the main text, we report the result of a simulation where (1) the validation studies have the same coefficient perturbation setting as the low-perturbation studies in the training set; (2) Y is generated as a linear function of a subset of the predictors with normal errors. In addition to this basic setting, we allowed the validation studies to resemble the high-perturbation studies in the training sets. We also varied the relationship between Y and $\mathbf{X}$ in three ways:

- Linear relationship with N(0,1) errors: $Y = \beta X + \epsilon$, where the length of β is chosen from $unif[2, 20]$ and $\epsilon \sim N(0, 1)$.
- Linear relationship with "slash" errors: Same as (1), except per Friedman[1], $\epsilon = s * \frac{u}{v}$, $u \sim N(0, 1)$, $v \sim unif[0, 1]$, resulting in a heavier-tailed distribution. We set $s = 0.05$ to maintain manageable bounds on extreme errors.
- Linear relationship with constant interaction terms. For this scenario, the length of β was chosen from $unif[3, 20]$, and $Y = \beta X + 2.8X_1X_2 - 1.6X_1X_3 + \epsilon$, $\epsilon \sim N(0, 1)$. This introduces some non-linearity to the determination of Y, and although the β are perturbed we did not perturb the interaction coefficients to maintain comparable error rates across all methods.

These variations yield six total simulation settings, the first of which is summarized in Figure 3 of the main text. We provide analogous figures for the five remaining simulation settings here. Because the more volatile and relational settings can produce extreme cases in certain iterations of the simulation, we report the median of the average RMSE in the validation datasets across 100 iterations. In the original Figure 3, we report the mean.

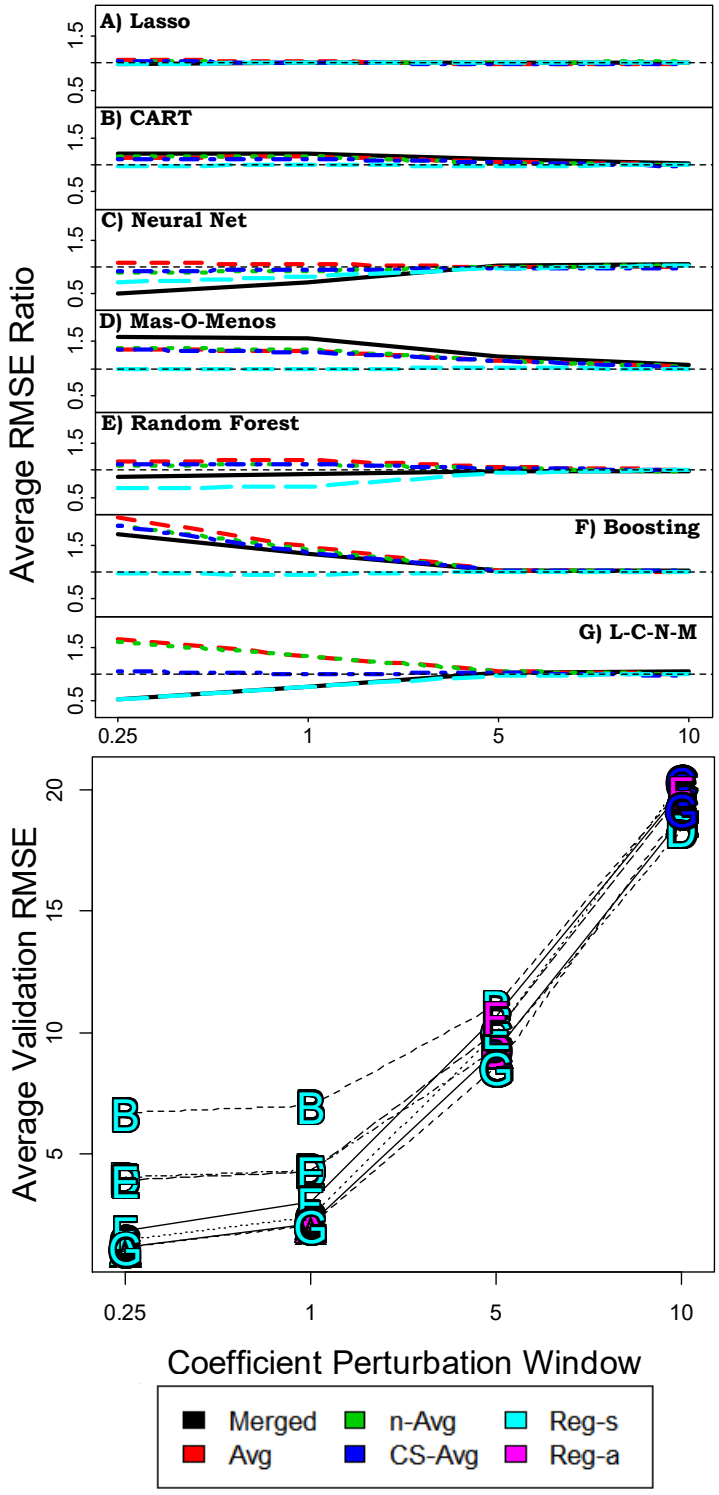

Figure 4: Linear relationship, normal errors, high-perturbation validation.

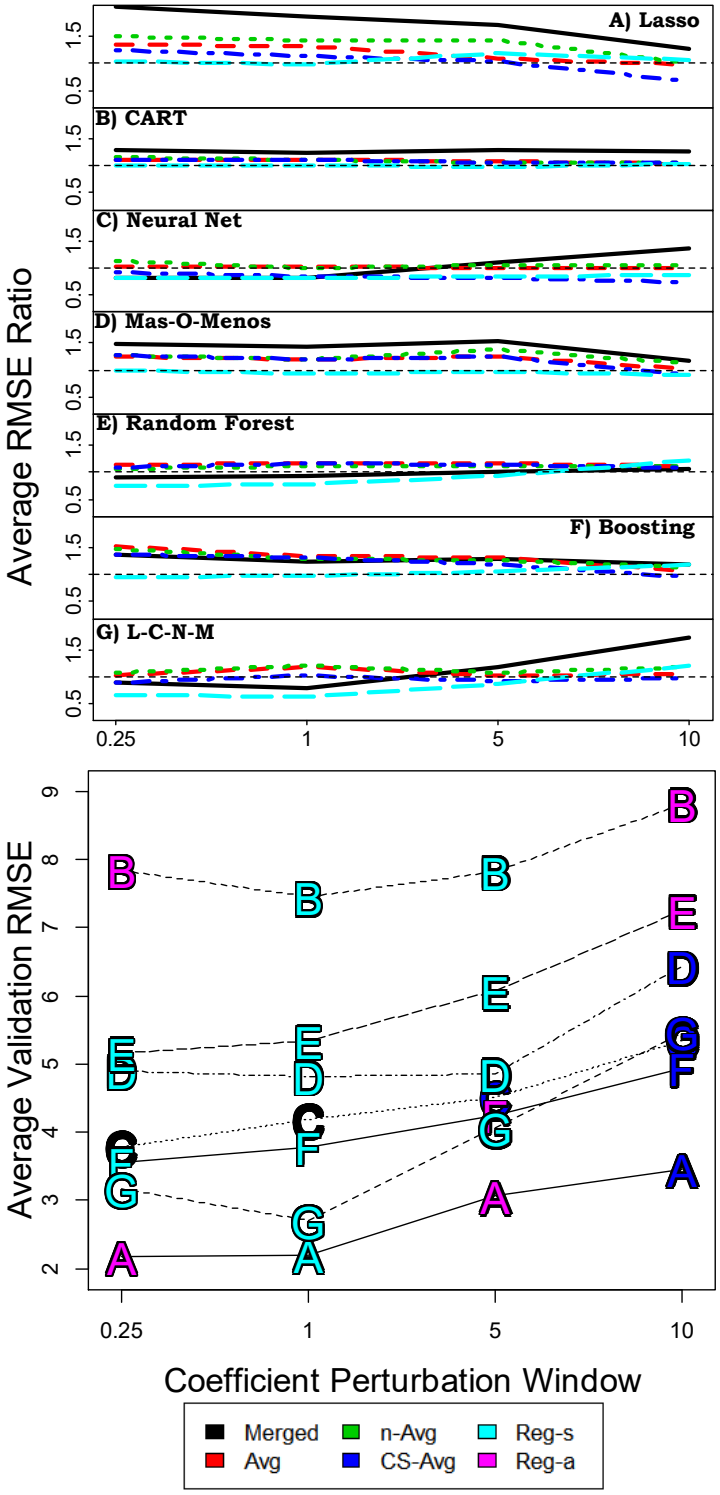

Figure 5: Linear relationship, "slash" errors, low-perturbation validation.

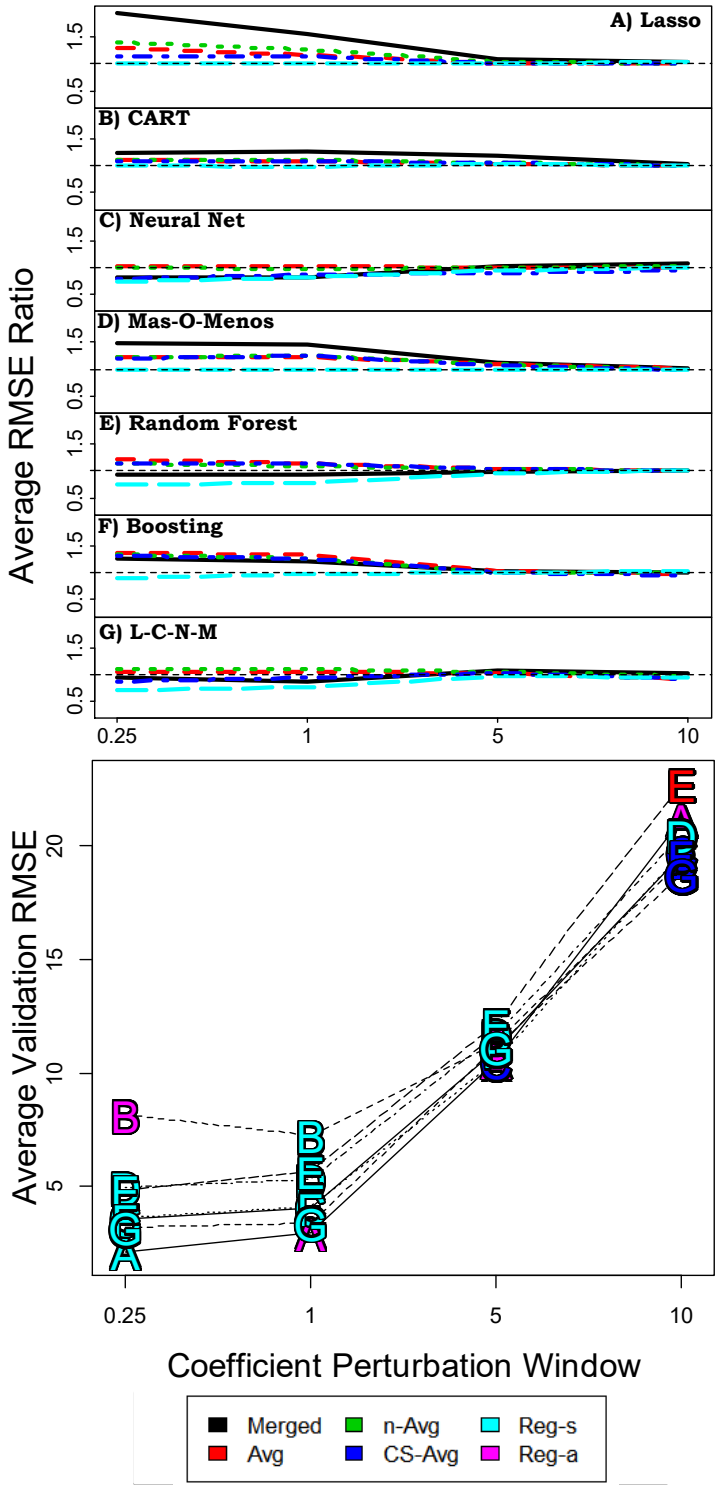

Figure 6: Linear relationship, "slash" errors, high-perturbation validation.

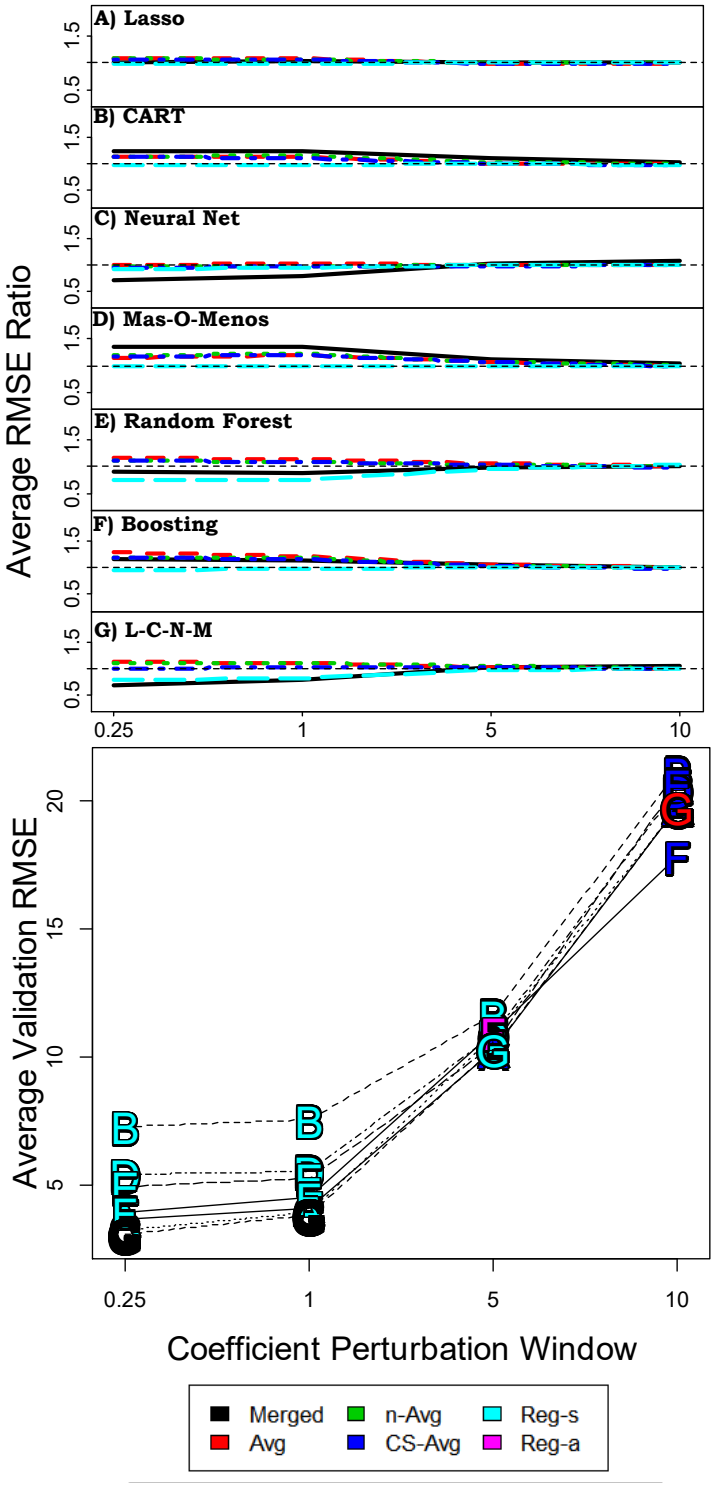

Figure 7: Non-linear relationship, normal errors, low-perturbation validation.

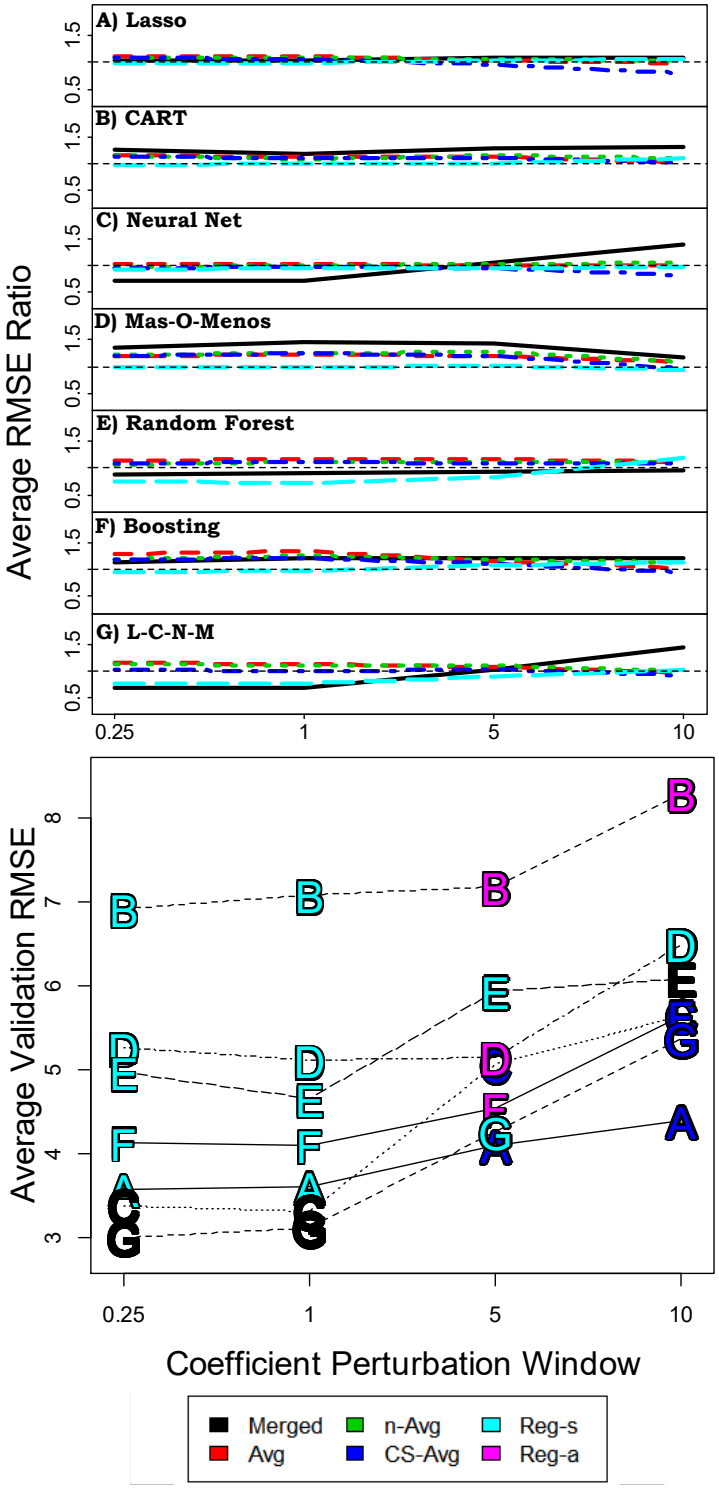

Figure 8: Non-linear relationship, normal errors, high-perturbation validation.

We see similar patterns of behavior across the two validation perturbation scenarios. Generally, it is much more difficult to predict well when the validation datasets are highly variant. Similar to Figure 3 in the main text, we observe a trend of merging/stacking performing well at low perturbation levels and different rates of transition to ensemble weights performing better as the perturbation window is increased. Figure 7, where some non-linearity is introduced in the generation of Y, we see that Neural Network and the unified "L-C-N-M" learner outperform LASSO for low levels of perturbation in the training data. Lasso improves as the perturbation window is increased. It is possible that LASSO's shrinking of coefficients provides some additional benefit in the high-perturbation scenarios, even when there is non-linearity in the generation of Y. A more complex non-linear generation scheme for Y may tease this out further.

3 Interplay between within-study and cross-study ensembling

As a brief extension of the ideas presented in the main manuscript, we examined a modification of our simulation setting. Instead of training one SSL on each study, we train 100 SSLs (CART without pruning). These are trained on bootstrap samples of each dataset and random subsets of predictors are used for each SSL. We then average the prediction from each SSL across all datasets. We compared the performance of this "ensemble of ensembles" to Random Forest - merged, simple average of SSLs (one SSL per study), and cross-study weighted average of SSLs (one SSL per study). We did this for the largest perturbation window for five of the ten training datasets. We compared average RMSE in low-perturbation validation sets over $R = 100$ simulation iterations. Table 9 shows that the "ensemble of ensembles" approach described here does not compete well with Random Forest. We posit that incorporating more cross-study information to understand which weak learners could be excluded from the final ensemble may improve its performance.

Window	Merged	Avg.	CS	EoE
10	6.04	6.88	6.60	7.32

Table 9: Comparing "Ensemble of ensembles" approach to a subset of previously-described weighting schemes. Each value is the average RMSE in the validation studies averaged over 100 iterations.

References

[1] Jerome H Friedman. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232, 2001.

[2] Markus Riester, Wei Wei, Levi Waldron, Aedin C Culhane, Lorenzo Trippa, Esther Oliva, Sung-Hoon Kim, Franziska Michor, Curtis Huttenhower, Giovanni Parmigiani, and Michael J Birrer. Risk prediction for late-stage ovarian cancer by meta-analysis of 1525 patient samples. *JNCI Journal of the National Cancer Institute*, 106(5):dju048–dju048, May 2014.

[3] H Uno, T Cai, M J Pencina, and R B D'Agostino. On the C-statistics for evaluating overall adequacy of risk prediction procedures with censored survival data. *Statistics in ...*, 2011.

[4] Levi Waldron, Benjamin Haibe-Kains, Aedin C Culhane, Markus Riester, Jie Ding, Xin Victoria Wang, Mahnaz Ahmadifar, Svitlana Tyekucheva, Christoph Bernau, Thomas Risch, Benjamin Frederick Ganzfried, Curtis Huttenhower, Michael Birrer, and Giovanni Parmigiani. Comparative meta-analysis of prognostic gene signatures for late-stage ovarian cancer. *JNCI Journal of the National Cancer Institute*, 106(5):dju049–dju049, May 2014.