Pilot Study: Assessing the impact of class imbalance on the performance of prediction models developed for dichotomous risk prediction.

Alex Carriero

January 10, 2022

Supervisors:

Maarten van Smeden, Utrecht Medical Center Utrecht
Kim Luijken, Utrecht Medical Center Utrecht
Ben van Calster, Catholic Univeristy of Leuven and Leiden University Medical Center

Program: Methodology and Statistics for Behavioral, BioMedical, and Social Sciences.

Host Institution: Julius Center for Health Science and Primary Care, UMC.

Candidate Journal: Statistics in Medicine.

Word Count: 2500

1. Introduction

a nice introduction.

2. Methods

In this paper, we implemented a simulation study to investigate the effect of class imbalance on the performance of various classification algorithms. We adhere to the ADEMP guidelines for the design and reporting of our simulation study[7].

Aim

The aim of this pilot study was to investigate the effect of class imbalance on the performance of six commonly used classification algorithms. In particular, we aimed to assess the out-of-sample predictive performance of prediction models trained with various classification algorithms, when the event fraction was varied across three levels and no imbalance corrections were implemented.

Data-Generating Mechanism

Data for each class was generated independently from two distinct multivariate normal distributions:

$$\text{Class 0: } \mathbf{X} \sim mvn(\boldsymbol{\mu_0}, \boldsymbol{\Sigma_0}) = mvn(\mathbf{0}, \boldsymbol{\Sigma_0})$$

$$\text{Class 1: } \mathbf{X} \sim mvn(\boldsymbol{\mu_1}, \boldsymbol{\Sigma_1}) = mvn(\boldsymbol{\Delta_\mu}, \boldsymbol{\Sigma_0} - \boldsymbol{\Delta_\Sigma})$$

The parameters (mean vector and covariance matrix) of these data generating distributions were distinct between the classes. In the formulae above, $\boldsymbol{\Delta_\mu}$ refers to the vector housing the difference in predictor means between the two classes. Similarly, $\boldsymbol{\Delta_\Sigma}$ refers to the matrix housing the difference in predictor variances/covariances between the classes.

In class 0, all predictor means were fixed to zero and all variances were fixed to 1. In class 1, all means were non-zero and are stored in the vector $\boldsymbol{\Delta_\mu}$. There was no variation in means among predictors within a class, thus, every element in the vector $\boldsymbol{\Delta_\mu}$ is equivalent; denoted by δ_μ. Similarly, there was no variation in predictor variances within a class, so every diagonal element in $\boldsymbol{\Delta_\Sigma}$ is equivalent; diagonal elements are denoted by δ_Σ.

Finally, 80% of the predictors were allowed to covary. All correlations among predictors in each class were set to 0.2. To ensure the correlation of predictors was not stronger in one class, the correlation matrices of the two classes were fixed to be equal. This was accomplished by computing the off-diagonal elements of $\boldsymbol{\Delta_\Sigma}$ such that the correlation matrices between the two classes were equivalent. Note, the covariance matrices were *not* equivalent between the classes. For example, in scenario where we have 8 predictors:

mean and covariance structure for class 0,

$$\boldsymbol{\mu_0} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \boldsymbol{\Sigma_0} = \begin{bmatrix} 1 & 0.2 & 0.2 & 0.2 & 0.2 & 0.2 & 0 & 0 \\ 0.2 & 1 & 0.2 & 0.2 & 0.2 & 0.2 & 0 & 0 \\ 0.2 & 0.2 & 1 & 0.2 & 0.2 & 0.2 & 0 & 0 \\ 0.2 & 0.2 & 0.2 & 1 & 0.2 & 0.2 & 0 & 0 \\ 0.2 & 0.2 & 0.2 & 0.2 & 1 & 0.2 & 0 & 0 \\ 0.2 & 0.2 & 0.2 & 0.2 & 0.2 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

mean and covariance structure for class 1,

$$\boldsymbol{\mu_1} = \begin{bmatrix} \\ \delta_\mu \\ \delta_\mu \\ \delta_\mu \\ \delta_\mu \\ \delta_\mu \\ \delta_\mu \\ \delta_\mu \\ \delta_\mu \end{bmatrix}, \boldsymbol{\Sigma_1} = \begin{bmatrix} 1-\delta_\Sigma & z & z & z & z & z & 0 & 0 \\ z & 1-\delta_\Sigma & z & z & z & z & 0 & 0 \\ z & z & 1-\delta_\Sigma & z & z & z & 0 & 0 \\ z & z & z & 1-\delta_\Sigma & z & z & 0 & 0 \\ z & z & z & z & 1-\delta_\Sigma & z & 0 & 0 \\ z & z & z & z & z & 1-\delta_\Sigma & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1-\delta_\Sigma & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1-\delta_\Sigma \end{bmatrix}$$

.

Here, $z = \frac{(1-\delta_\Sigma)*0.2}{1}$, to ensure the correlation matrices of the two classes were equivalent.

In the study we investigated simlated data to reflect three unique scenarios. This was accomplished by varying the event fraction through the set {0.5, 0.2, 0.02}. For all scenarios, data for 8 predictors was generated and the sample size (N) was determined as the mimumum sample size required for the prediction model using the R package `pmpsampsize` [?] .

Under the assumption of normality for all predictors (in each class), the ΔC Statistic of the data can be expressed as a function of $\boldsymbol{\Delta_\mu}$, $\boldsymbol{\Sigma_0}$ and $\boldsymbol{\Sigma_1}$ [2]. For every scenario, the parameter values for the data generating distributions (δ_μ and δ_Σ) were selected to generate a ΔC Statistic = 0.85. Their values were computed analytically, based on the following formula from Demler et al. [2].

$$AUC = \Phi\left(\sqrt{\boldsymbol{\Delta_\mu}' \left(\boldsymbol{\Sigma_0} + \boldsymbol{\Sigma_1}\right)^{-1} \boldsymbol{\Delta_\mu}}\right) \tag{1}$$

In equation (1), Φ represents the cumulative density function (cdf) of the standard normal distribution; $\boldsymbol{\Delta_\mu}$, $\boldsymbol{\Sigma_0}$ and $\boldsymbol{\Sigma_1}$ maintain their previous definitions. To ensure a unique solution δ_Σ was fixed at 0.3 for each scenario, while equation (1) was solved to yield the appropriate value of δ_μ in each scenario.

Finally, given that data for each class were generated independently, we had direct control over how many observations were generated under each class. The number of observations from the positive class (n_1) was sampled from the binomial distribution with probability equal to the specified event fraction. The number of observations in the negative class (n_0) was then be computed as $N - n_1$, where N is the mimimum sample size specified for the prediction model.

Parameter values for the data generating distributions of the simulation scenarios are presented in table 1.

Table 1: Summary of the data generating parameters for each simulation scenario.

Event Fraction	No. Predictors	N	δ_μ	δ_Σ	ΔC
0.50	8	385	0.6043	0.3	0.85
0.20	8	247	0.6043	0.3	0.85
0.02	8	1797	0.6043	0.3	0.85

Estimands

The focus of this study was the out-of-sample predictive performance of clinical prediction models for dichotomous risk prediction.

Methods

Under each scenario, 200 data sets were generated. Each data set was comprised of training and validation data. The training and validation data were generated independently using identical data generating mechanisms; this was done to ensure a similar event fraction in the training and validation data. The validation data was generated to be 10x larger than the training set.

For each generated data set, six prediction models were developed, each using a different classification algorithm. All prediction models were trained using the training data. Out-of-sample performance was then assessed using the validation data.

Classification algorithms were selected based on a systematic review identifying common algorithms used to develop prediction models in a medical context [CITE CONSTANZA REVIEW]. These algorithms include: logistic regression, support vector machine, random forest and XGBoost. Additionally, based on literature summarizing common strategies to handle class imbalance [CITE PAPERS], we include two ensemble learning algorithms designed specifically to handle class imbalance: RUSBoost and and EasyEnsemble. Classification algorithms were implemented using their default hyperparameters. All classification algorithms and the R packages used for their implementation are summarized in Table 2.

Table 2: Summary of classification algorithms used in simulation study.

Classification Algorithm	Abbreviation	R Package
Logistic Regression	LR	base R [8]
Support Vector Machine	SVM	e1071 [6]
Random Forest	RF	randomForest [5]
XGBoost	XG	xgboost [1]
RUSBoost	RB	ebmc [3]
EasyEnsemmble	EE	iric [12]

Performance Measures

Out-of-sample model performance was assessed using measures of calibration, discrimination, overall performance. All performance metrics were computed based on validation data.

Calibration was assessed in term of calibration intercept and slope as defined in [10]. Additionally, model calibration was visualized using flexible calibration curves; fit using loess regression. In each simulation scenario, a flexible calibration curve was fit for all models in every iteration.

Discrimination was measured by area under the receiver operator curve (ΔC-statistic); computed using the R package `pROC` [9].

Overall performance was measured by brier score. Brier score was computed in accordance with its definition in Steyerberg et al. [10]; this metric reflects both model discrimination and calibration.

For the empirical measures of model performance (ΔC statistic, brier score, calibration intercept and slope), in each scenario, the mean over all iterations as well as the corresponding monte carlo standard error was calculated and reported.

Software

All analyses were conducting using R version 4.1.2 [8].

3. Results

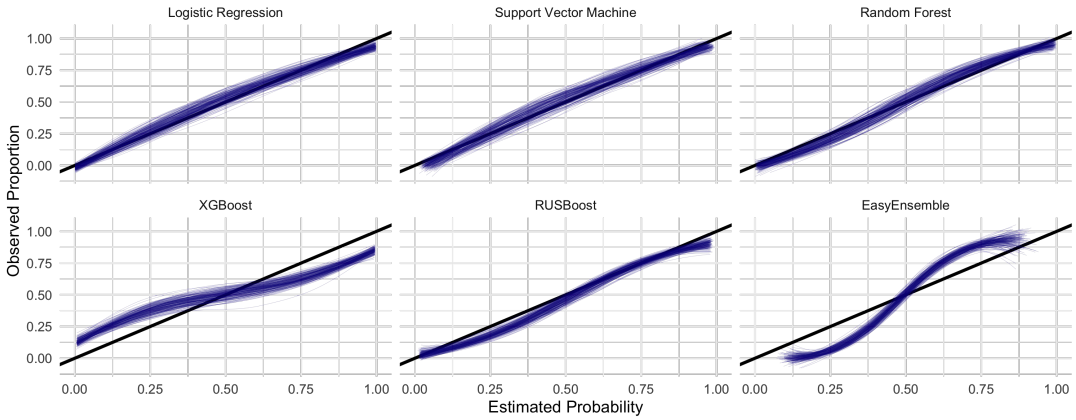
For the pilot study, empirical performance metrics are summarized in Table 3 and model performance visualizations are displayed in Figure 1.

In a well calibrated model, predicted probabilities correspond to observed proportions[11]. In terms of calibration intercept an slope, good calibration is represented by values of 0 and 1, respectively. For a flexible calibration curve, when predicted probabilities (x-axis) correspond well to the observed proportions (y-axis), the curve follows a diagonal line ($y = x$)[11]. For approximately balanced data (event fraction = 0.5), all algorithms, except XG and EE, were well calibrated. While, on average, both XG and EE had calibration intercepts near zero, their average calibration slopes dramatically deviated from 1. We see that for XGBoost, the risk predictions above 0.5 overestimated true risk, while the risk predictions below 0.5 underestimated true risk (Figure 1(a)). In other words, the XGBoost models resulted in risk estimates which were too extreme (calibration slope = XXX). The opposite pattern is true for EasyEnsemble; EE produced risk estimates which were too moderate (calibration slope = XXX). With respect to discrimination and overall performance, an ideal model will produce brier scores close to zero ΔC statistics close to one[10]. In the balanced data scenario, SVM and LR had similar discrimination and overall performance and both outperformed the other algorithm.

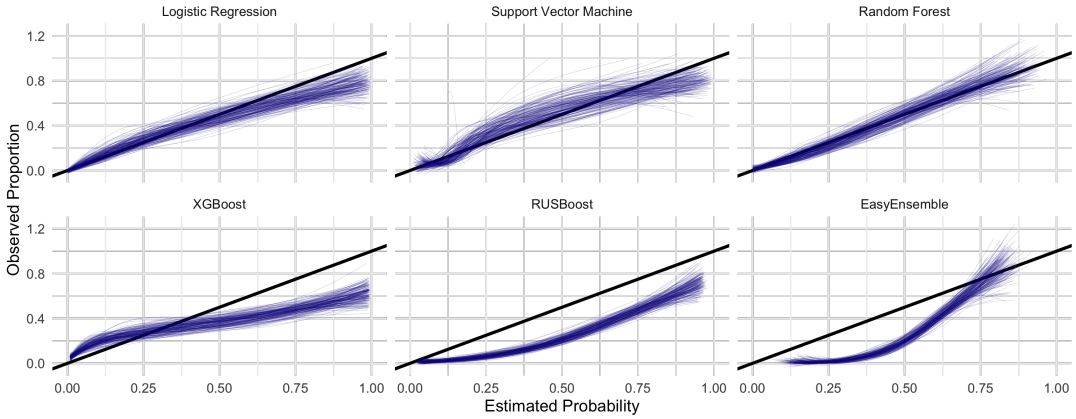
With the event fraction at 0.2, for half of the algorithms, model calibration was moderate. While the linear classifiers and random forest maintained adequate calibration in this scenario, XGBoost, RUSBoost and EasyEnsemble all, on average, producde risk predictions that dramatically over estimated true risk (Figure 1(b)). With respect to discrimination and overall performance, in this scenario, LR was the best performing algorithm.

At an event fraction of 0.02, all algorithms exhibited miscalibration. From Figure 1 (c), we see that for the linear classifiers and random forest, the calibration curves were sporadic; there is large variation in the calibration curves produced for each iteration of the simulation. Meanwhile, for XGBoost, RUSBoost, and EasyEnsemble, the calibration curves did not vary much across the iterations, rather, they exhibited a specific pattern of miscalibration: all risk predictions over estimated true risk. With respect to overall performance and discrimination, in this scenario, LR was again, the best performing algorithm.

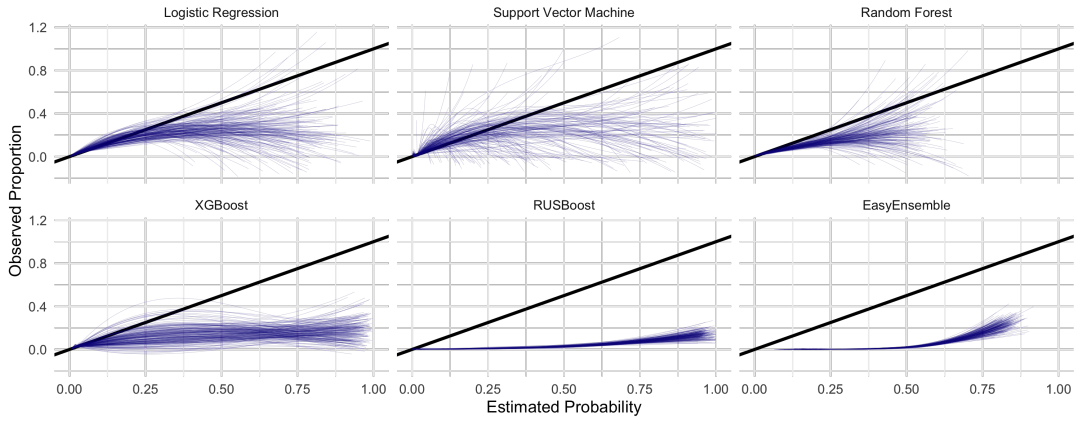
Overall, as the imbalance between the classes was magnified, model calibration deteriorated across all algorithms. Meanwhile, discrimination maintained relatively constant. Interestingly, as the imbalance between the classes was magnified, overall performance appeared to improve, especially for the linear classifiers. This apparent improvement in overall performance is misleading and is a result of a poor choice in performance metric.



(a) Flexible calibration curves with event fraction: 0.5.



(b) Flexible calibration curves with event fraction: 0.2



(c) Flexible calibration curves with event fraction: 0.02

Figure 1: Visual representation of model calibration for each simulation scenario in the Pilot Study.

Table 3: Mean (monte carlo error) of performance metrics across 200 iterations in each simulation scenario.

	Δ C Statistic	Brier Score	Calibration Int.	Calibration Slope
Event Fraction: 0.5				
Logistic Regression	0.845 (0.007)	0.161 (0.004)	-0.001 (0.14)	0.928 (0.098)
Support Vector Machine	0.849 (0.008)	0.158 (0.004)	-0.002 (0.131)	1.022 (0.12)
Random Forest	0.842 (0.008)	0.163 (0.004)	-0.016 (0.124)	1.168 (0.087)
XGBoost	0.79 (0.012)	0.205 (0.007)	-0.046 (0.217)	0.464 (0.024)
RUSBoost	0.813 (0.01)	0.178 (0.005)	-0.183 (0.081)	1.063 (0.072)
EasyEnsemble	0.826 (0.01)	0.187 (0.004)	0.001 (0.045)	2.279 (0.176)
Event Fraction: 0.2				
Logistic Regression	0.836 (0.013)	0.122 (0.005)	-0.035 (0.216)	0.86 (0.137)
Support Vector Machine	0.81 (0.026)	0.123 (0.007)	-0.025 (0.197)	1.049 (1.057)
Random Forest	0.808 (0.018)	0.125 (0.005)	-0.079 (0.179)	1.079 (0.12)
XGBoost	0.755 (0.022)	0.153 (0.008)	0.006 (0.304)	0.445 (0.04)
RUSBoost	0.796 (0.018)	0.184 (0.011)	-1.335 (0.107)	1.078 (0.115)
EasyEnsemble	0.814 (0.018)	0.196 (0.009)	-1.302 (0.064)	2.331 (0.256)
Event Fraction: 0.02				
Logistic Regression	0.839 (0.011)	0.019 (0.001)	0.014 (0.199)	0.904 (0.124)
Support Vector Machine	0.698 (0.034)	0.019 (0.001)	-0.013 (0.193)	1.255 (2.652)
Random Forest	0.756 (0.022)	0.019 (0.001)	-0.115 (0.189)	0.655 (0.083)
XGBoost	0.707 (0.028)	0.022 (0.001)	-0.331 (0.186)	0.508 (0.047)
RUSBoost	0.778 (0.027)	0.158 (0.019)	-3.696 (0.157)	0.869 (0.152)
EasyEnsemble	0.81 (0.019)	0.2 (0.014)	-3.738 (0.08)	2.295 (0.304)

4. Discussion

In this paper, the problem of class imbalance was introduced. The results from the Pilot Study, investigating the performance of the classification algorithms, without imbalance correction, were presented. From these results, the problem of class imbalance is clearly demonstrated. As the event fraction drops from 0.2 to 0.02, all classification algorithms exhibited miscalibration. Further, these results confirm that assessing only discrimination and overall performance is insufficient. As imbalance between the classes is magnified, these metrics appear relatively unaffected.

Based on the results of the pilot study, brier score appeared to be an uninformative measure of overall performance when class imbalance is extreme. With event fraction of 0.02, a trivial majority classifier (a model that predicts everyone will belong to the majority class) would yield a brier score of 0.02. Therefore, in the full study, we will utilize another metric of overall performance, such as the index of prediction accuracy (IPA)[4] or a re-scaled brier score[10] (note to self: are these the same thing?) which are known to be more informative in the presence of class imbalance.

The algorithms RUSBoost and EasyEnsemble are designed specifically to handle class imbalance. Interestingly, these algorithms exhibited the highest degree of miscalibration in the presence of class imbalance. Further, in the pilot study, they did not out perform logistic regression with respect to classification and discrimination in any scenario. In fact, these algorithms had worse overall performance than a trivial majority classifier at the most extreme event fraction.

A concluding paragraph where I discuss the upcoming work in the full study. - outline of full simulation - mention hyper parameter tuning - new performance metrics: re-scaled brier score, clinical utility

References

[1] Tianqi Chen, Tong He, Michael Benesty, Vadim Khotilovich, Yuan Tang, Hyunsu Cho, Kailong Chen, Rory Mitchell, Ignacio Cano, Tianyi Zhou, Mu Li, Junyuan Xie, Min Lin, Yifeng Geng, Yutian Li, and Jiaming Yuan. *xgboost: Extreme Gradient Boosting*, 2022. URL https://CRAN.R-project.org/package=xgboost. R package version 1.6.0.1.

[2] Olga V. Demler, Michael J. Pencina, and Ralph B. D'Agostino Sr. Equivalence of improvement in area under roc curve and linear discriminant analysis coefficient under assumption of normality. *Statistics in Medicine*, 30(12):1410–1418, 2011.

[3] Hsiang Hao and Chen. *ebmc: Ensemble-Based Methods for Class Imbalance Problem*, 2022. URL https://CRAN.R-project.org/package=ebmc. R package version 1.0.1.

[4] Michael W. Kattan and Thomas A. Gerds. The index of prediction accuracy: an intuitive measure useful for evaluating risk prediction models. *Diagnostic and Prognostic Research*, 2(1):7, 2018. doi: 10.1186/s41512-018-0029-2. URL https://doi.org/10.1186/s41512-018-0029-2.

[5] Andy Liaw and Matthew Wiener. Classification and regression by randomforest. *R News*, 2(3):18–22, 2002. URL https://CRAN.R-project.org/doc/Rnews/.

[6] David Meyer, Evgenia Dimitriadou, Kurt Hornik, Andreas Weingessel, and Friedrich Leisch. *e1071: Misc Functions of the Department of Statistics, Probability Theory Group (Formerly: E1071), TU Wien*, 2022. URL https://CRAN.R-project.org/package=e1071. R package version 1.7-12.

[7] Tim P. Morris, Ian R. White, and Michael J. Crowther. Using simulation studies to evaluate statistical methods. *Statistics in Medicine*, 38(11):2074–2102, 2019.

[8] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2021. URL https://www.R-project.org/.

[9] Xavier Robin, Natacha Turck, Alexandre Hainard, Natalia Tiberti, Frédérique Lisacek, Jean-Charles Sanchez, and Markus Müller. proc: an open-source package for r and s+ to analyze and compare roc curves. *BMC Bioinformatics*, 12:77, 2011.

[10] Ewout W Steyerberg, Andrew J Vickers, Nancy R Cook, Thomas Gerds, Mithat Gonen, Nancy Obuchowski, Michael J Pencina, and Michael W Kattan. Assessing the performance of prediction models: a framework for traditional and novel measures. *Epidemiology (Cambridge, Mass.)*, 21(1):128–138, 01 2010.

[11] Ben Van Calster, David J. McLernon, Maarten van Smeden, Laure Wynants, Ewout W. Steyerberg, Patrick Bossuyt, Gary S. Collins, Petra Macaskill, David J. McLernon, Karel G. M. Moons, Ewout W. Steyerberg, Andrew J. Vickers, On behalf of Topic Group 'Evaluating diagnostic tests, and prediction models'of the STRATOS initiative. Calibration: the achilles heel of predictive analytics. *BMC Medicine*, 17(1):230, 2019.

[12] Bing Zhu, Zihan Gao, Junkai Zhao, and Seppe K.L.M. vanden Broucke. Iric: An r library for binary imbalanced classification. *SoftwareX*, 10:100341, 2019.