Class Imbalance

- Minority class much smaller than majority
- Class of interest ("positive class"), e.g., is only 5% of data
- Problem: 95% accurate guessing all negative; not helpful algorithm

Sampling Techniques

Over

- Randomly replicate minority class samples until training data is balanced

Under

- Randomly throw out majority class samples until training data is balanced

SMOTE

- Create synthetic minority cases, based on k-nearest neighbors

Ensembling Algorithms

Bagging

- In this study, using random forest
- Bootstrap the training data, select a subset of predictors, train decision tree on this data
- Do this a given number of times
- Prediction is majority vote of all of these decision trees

Boosting

- In this study, using AdaBoost and XGBoost
- Both train trees *serially*
- Learn from the mistakes of past trees by updating weights based on mistakes or updating based on the residuals using gradient descent

Data Generation

- Two multivariate normal predictors (A and B) are generated. A and B are correlated at $r = .65$. These two variables contributed to the log-odds by $4A + 4B + 2AB$

- Another variable, $J \sim U(-1, 1)$, was generated. This variable further added to the log-odds by $J^3 + 2 \times \exp(-6 \times (J - 0.3)^2)$

- Two more variables, $K \sim U(0, 1)$ and $L \sim U(0, 1)$, were generated and contributed to the log-odds by $2 \times \sin(K \times L)$

- For each data set, a number X was selected, where $X \sim N(50, 7)$. Another number, Y, was selected, where $Y \sim N(.15, .033)$. $Z = X - (X \times Y)$ variables were generated from a $N(0, 1)$ distribution. Each of these Z variables further added to the log-odds in a simple additive fashion, where coefficients were (a) of alternating signs and (b) evenly spaced from 2.50 to 0.25

- $\frac{Y}{2}$ variables were generated from a $N(0, 1)$ distribution and did not contribute to the log-odds

- The log-odds for each case were converted to probabilities. For each data set, a positive (i.e., minority) class proportion, M, was sampled from $N(.03, .007)$. Probabilities were sorted from lowest to highest. The difference between the probability for the $1 - M$th highest probability and M was calculated, and this constant was added to the probability for each case

- Lastly, the number of cases in each data set were randomly drawn from a distribution $N(40000, 5000)$. 500 data sets were generated, and sixteen combinations of sampling techniques and algorithms were fit to each of these data sets

Performance Assessment

Precision

- $\frac{TP}{TP+FP}$

Recall

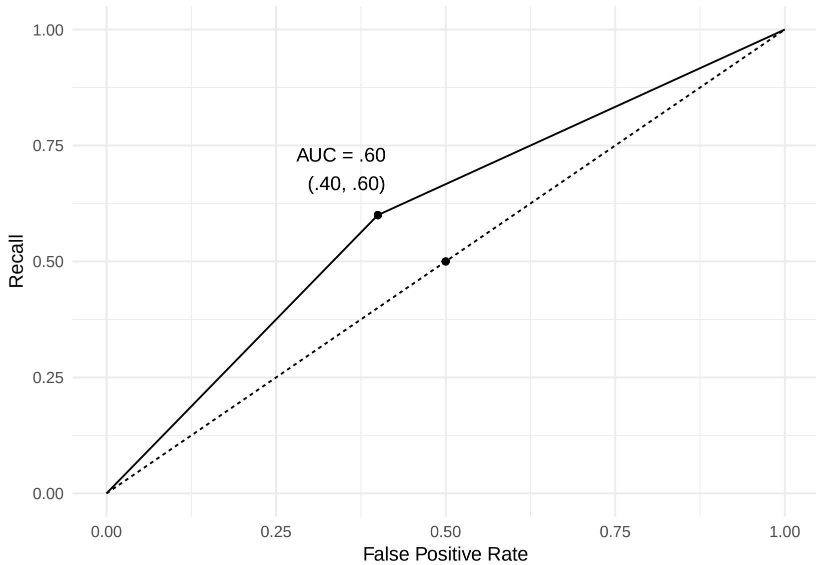
- $\frac{TP}{TP+FN}$

F1

- $F_1 = 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}}$

AUC(ROC)

- $\text{AUC(ROC)} = \frac{1 + \text{recall} - \text{false positive rate}}{2}$



Results

Making Enough Positive Predictions

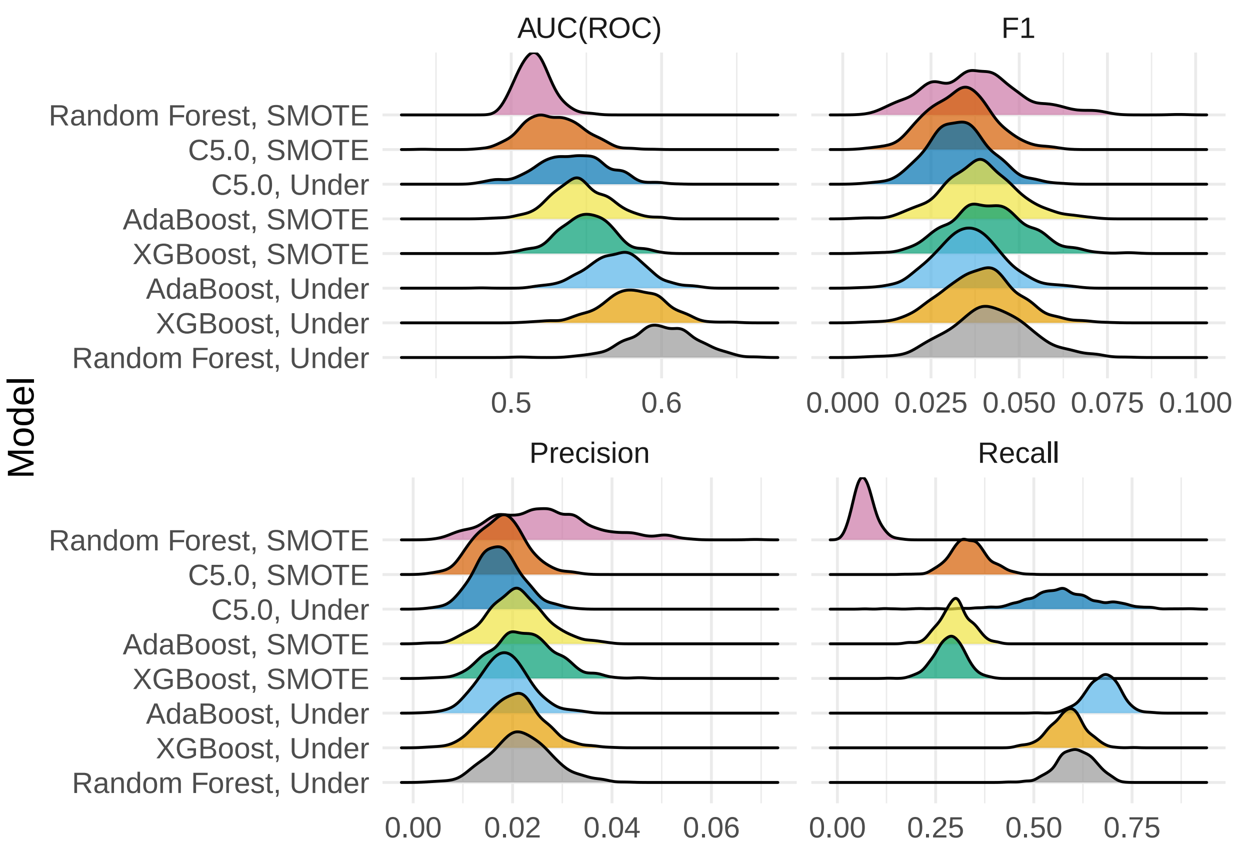
Model	Proportion P = 0
C5.0, None	1.000
Random Forest, Over	1.000
Random Forest, None	0.976
XGBoost, None	0.964
AdaBoost, Over	0.510
AdaBoost, None	0.230
AdaBoost, SMOTE	0.000
AdaBoost, Under	0.000
C5.0, Over	0.000
C5.0, SMOTE	0.000
C5.0, Under	0.000
Random Forest, SMOTE	0.000
Random Forest, Under	0.000
XGBoost, Over	0.000
XGBoost, SMOTE	0.000
XGBoost, Under	0.000

Model	Proportion F1 is N/A
C5.0, None	1.000
Random Forest, None	1.000
Random Forest, Over	1.000
XGBoost, None	0.998
AdaBoost, Over	0.972
AdaBoost, None	0.922
C5.0, Over	0.046
XGBoost, Over	0.004

Model	Proportion F1 is N/A
AdaBoost, SMOTE	0.000
AdaBoost, Under	0.000
C5.0, SMOTE	0.000
C5.0, Under	0.000
Random Forest, SMOTE	0.000
Random Forest, Under	0.000
XGBoost, SMOTE	0.000
XGBoost, Under	0.000

Comparing Mean Performance

Model	Precision	Recall	F1	AUC(ROC)
Random Forest, Under	0.022	0.606	0.042	0.600
XGBoost, Under	0.020	0.585	0.039	0.581
AdaBoost, Under	0.018	0.676	0.036	0.570
XGBoost, SMOTE	0.022	0.286	0.041	0.550
AdaBoost, SMOTE	0.021	0.301	0.039	0.545
C5.0, Under	0.017	0.570	0.033	0.540
C5.0, SMOTE	0.018	0.335	0.033	0.528
Random Forest, SMOTE	0.026	0.069	0.037	0.516



Outcome	Pairwise Comparison	Difference	2.5%	97.5%
AUC(ROC)	Random Forest - XGBoost	0.019	0.017	0.022
	Random Forest - AdaBoost	0.030	0.028	0.033
	XGBoost - AdaBoost	0.011	0.009	0.014
F1	Random Forest - XGBoost	0.003	0.002	0.004
	Random Forest - AdaBoost	0.006	0.005	0.007
	XGBoost - AdaBoost	0.003	0.002	0.004
Recall	Random Forest - XGBoost	0.022	0.015	0.029
	Random Forest - AdaBoost	-0.069	-0.076	-0.063
	XGBoost - AdaBoost	-0.091	-0.098	-0.084
Precision	Random Forest - XGBoost	0.002	0.001	0.002
	Random Forest - AdaBoost	0.003	0.003	0.004
	XGBoost - AdaBoost	0.002	0.001	0.003

Performance With Data Characteristics