Comparison of Supervised Machine Learning Techniques to Predict Fetal Health at 32-Weeks using Cardiotocogram Data

Extending results to investigate utility of binary vs tertiary outcome

Peter Ehmann

Dr. Jason Klusowski

MSDS 534 – Statistical Learning

Rutgers, The State University of New Jersey

December 11, 2019

Link to GitHub repository: https://github.com/peter-ehmann/Cardiotocography

(1) results

(2) README.md

(3) stat534code_ehmann.Rmd

(4) stat534report_ehmann.pdf

(5) stat534slides_ehmann.pdf

Dataset available from UCI Repository: https://archive.ics.uci.edu/ml/datasets/Cardiotocography

INTRODUCTION

During pregnancy it can be difficult to noninvasively obtain information about a fetus' health status. Cardiotocography (CTG) is one frequently used measure used in the later stages of pregnancy to monitor fetal heart rate (FHR) over time to ensure the fetus is getting an adequate oxygen supply (Hon & Hess, 1957). CTGs are also useful in determining future complications during delivery and help prevent neurological disorders. An accurate analysis of the CTG by trained medical professionals is essential to ensure fetal health and a smooth delivery process. The basic elements of the CTG recording are baseline heart rate in addition to variability, accelerations, and decelerations in heart rate. A baseline of 120-160 beats per minute (bpm) is considered normal. Variability is determined by irregular fluctuations in amplitude and frequency in the baseline, and variability of fewer than 6 bpm is considered abnormal.

While its usefulness is rarely questioned, there can be considerable inter-rater variability caused by subjectivity in the doctors analyzing the test. Errors in the interpretation of a CTG can lead to unnecessary medical intervention (false positive) or permanent injury to the fetus (false negative). Both errors carry a financial burden to the hospital and healthcare system, with the latter error being much greater. The goal of the present analysis is to use supervised machine learning to predict a fetus' health status based on automatically generated variables derived from the CTG (Ayres-de-Campos *et al.*, 2000). However, with many machine learning approaches, it is unknown which will perform best on CTG data. The present analysis aims to utilize a wide variety of models to identify what produces the fewest false negative predictions.

Training computational models to predict healthy versus pathological states of a fetus using objective measures automatically derived from the CTG is indeed a potential method to reduce diagnoses errors, reduce cost to the patient and hospital, and provide doctors with extra

information. With increased data collection and automation, it may be possible to train a model that accurately predicts fetal health without analysis from the doctor. For example, doctors' time could be saved by only having to review CTGs classified as *Suspect* and acting according to protocol for *Normal* (no intervention) and *Pathologic* (immediate intervention) diagnoses.

Aims & Hypotheses

There were two primary aims of the present analysis. The first aim was to compare the classification accuracy of 10-class and 3-class classification models. It was hypothesized that the 3-class model would produce better prediction accuracy due to the increased proportion of training observations per class. The second aim was to evaluate the utility of the tertiary (NSP) classification vs a binary (NP) classification (i.e., testing the usefulness of the *Suspect* classification). In this analysis, 3 models were compared: (1) binary classification with *Suspect* observations coded as Normal, (2) binary classification with *Suspect* coded as *Pathologic,* and (3) tertiary classification (identical to the analysis in Aim 1). It was hypothesized that binary classification with *Suspect* observations coded as *Normal* would produce the best prediction accuracy because doctors are probably overly cautious and want to mitigate false negatives (i.e., not identify a problem when a problem exists) due to the financial and legal risks involved. A third aim was to also compare the predictive accuracy of various statistical and computational models including Logistic Regression, Naïve Bayes, Random Forest, Neural Network, and Boosting, in order to identify what model should be used to best guide clinical practice. It was hypothesized that the three machine learning methods would outperform Logistic Regression and Naïve Bayes but would require more computing time to learn and train to find optimal hyperparameters. Importantly, this analysis only focused on predictive accuracy and not model interpretability and variable importance.

METHODS

Dataset

2126 fetal CTG readings were automatically processed and diagnostic features (21) were measured automatically. The CTGs were classified by three expert obstetricians and a consensus classification label was assigned to each of them. Classification was both with respect to a morphologic pattern (CLASS; 1-10) and to a fetal state (NSP; Normal, Suspect, and Pathologic). There were twenty-one predictors which are summarized on the following page.

LB	FHR baseline (beats per minute)	*AC*	# of accelerations per second
FM	# of fetal movements per second	*UC*	# of uterine contractions per second
DL	# of light decelerations per second	*DS*	# of severe decelerations per second
DP	# of prolonged decelerations per second	*MSTV*	mean value of short-term var.
ASTV	% time with abnormal short-term var.	*MLTV*	mean value of long-term var.
ALTV	% time with abnormal long-term var.	*Width*	width of FHR histogram
Min	minimum of FHR histogram	*Max*	maximum of FHR histogram
Nmax	# of histogram peaks	*Nzeros*	# of histogram zeros
Mode	histogram mode	*Mean*	histogram mean
Med	histogram median	*Var*	histogram variance
Tend	histogram tendency		

Procedures

All analyses were conducted using R version 3.6.1 in RStudio. The full dataset (.xlsx) was imported from the UCI Machine Learning Repository (Dua & Graff, 2019) and cleaned by removing observations with missing data and removing unnecessary columns. Observations were randomly split into sets of 1700 and 426 (80:20) for training and testing. The training set was

further split randomly into sets of 1275 and 425 (75:25) for training and development. These splits were used for (1) training model weights, (2) tuning model hyperparameters, and (3) evaluating model accuracy. Models were specified in the format y ~ x1 + … + x21 and all predictor variables were entered. For each of the Random Forest, Neural Network, and Boosting models, two hyperparameters were tuned to find the minimum development error using grid search. The optimal hyperparameters were used in generating each model that then predicted classifications on the test set with respect to the actual observations to evaluate model performance. In total, four experiments were conducted in order to answer the three aims described in the Introduction. Aim 1 was answered by comparing experiments 1 and 2. Aim 2 was answered by comparing experiments 2, 3 and 4. Aim 3 was answered by comparing all experiments holistically. The experiments are briefly described below.

Experiment 1 – Classification of CLASS (10 classes).

Experiment 2 – Classification of NSP (3 classes).

Experiment 3 – Binary classification of NP (2 classes) with *Suspect* coded as *Normal*.

Experiment 4 – Binary classification of NP (2 classes) with *Suspect* coded as *Pathologic*.

Model Evaluation

Models were evaluated using accuracy (1-error), precision, recall, and F-score. The formulas and definitions of each performance index is shown below. For multi-class (3+) problems, macro-averaging was implemented.

Accuracy	(TP + FP) / (TP + TN + FP + FN)	overall effectiveness of a classifier
Precision	TP / (TP + FP)	use when concerned about FP
Recall	TP / (TP + FN)	use when concerned about FN
F1-score	2 × (P + R) / (P × R)	harmonic mean of precision & recall

Machine Learning Methods

Logistic Regression is an extension of linear regression that is used when the outcomes are categorical. It was implemented in the present study using the glmnet R package (Friedman, Hastie, & Tibshirani, 2010). Logistic regression is a linear prediction method with a logit link function that calculates the log of the odds ratio (i.e., the probability of an observation being classified). Elastic net regularization was used as a variable selection method to reduce the potential for overfitting. In this case, both the L_1 and L_2 penalties are controlled. In addition, the training and validation sets were combined and 5-fold cross-validation was performed to optimally tune the α and β penalties. The value of λ used in the model that provides the minimum mean cross-validated error was applied to the test set. The notation for logistic regression with elastic net regularization is shown below.

$$\min_{(\beta_0,\beta)\in\mathbb{R}^{p+1}} -\left[\frac{1}{N}\sum_{i=1}^{N} y_i \cdot (\beta_0 + x_i^T\beta) - \log(1 + e^{(\beta_0+x_i^T\beta)})\right] + \lambda\left[(1-\alpha)||\beta||_2^2/2 + \alpha||\beta||_1\right]$$

Naïve Bayes is a probabilistic learning algorithm based on Bayes Theorem implemented using the e1071 R package. No hyperparameters were tuned in this model and the full training set was used to train the classifier (i.e., no development set was used) For each observation, it calculates the probability it belongs to a class, $P(c \mid \{x_1, \ldots, x_n\})$. Bayes Theorem along with each of the terms is shown below.

Bayes Theorem	$P(c \mid x) = P(x \mid c) \times P(c) / P(x)$
Posterior Probability	$P(c \mid x)$
Likelihood	$P(x \mid c)$
Class Prior Probability	$P(c)$
Predictor Prior Probability	$P(x)$

Random Forests are a computational model based on averaging many decision trees created with a subset of the total number of input variables (p). The *n.trees* parameter was set to 5000 for all random forest models, but this parameter was not tuned to find the optimal number of decision trees to use in the model. More computational time would be needed to tune this parameter along with the two other parameters of interest. *Mtry* and *nodesize* were tuned using values of 2-9 and 1-5, respectively. Based on current guidelines, for classification problems *mtry* should be set to floor(sqrt(p)) = 4. *Mtry* indicates the number of predictor variables that a decision tree considers at each node split (i.e., a fraction of the full set of predictors), and allows for this type of model to retain low bias while lowering the variance compared to other similar models such as bagging. *Nodesize* was tuned in order to prevent overfitting of the training data as it denotes the minimum number of observations allowed to fall into a terminal node of a decision tree. A lower *nodesize* (default = 1) allows for a very complex model but could overfit the training data. A higher *nodesize* may underfit the data and result in poor model performance. As a result, 8 (*mtry*) × 5 (*nodesize*) = 40 models were compared.

Neural Networks are computing structures modelled after the human brain. In the present analysis a simple feedforward neural network architecture was built with one hidden layer. At each layer, the input is multiplied by weights and a bias term is added. At the next layer, an activation function (e.g., tanh or relu) is applied, and that is fed into the next layer. At the output layer, the activation function is either softmax (3+ classes) or sigmoid (2 classes) which computes a probability. The model then calculates a loss function by comparing predictions to actual output observations and backpropagation updates the learned weights. The hyperparameters tuned were *nnodes* (60, 80, 100, 120, 140 nodes) and *decay* (0, 0.1. 0.01. 0.001). In total, 5 (*nnodes*) × 4 (*decay*) = 20 models were compared.

Boosting is a computational method in which a series of weak classifiers are trained sequentially but slightly improved based on the mistakes of previous classifiers. It was implemented using the gbm R package. Similar to random forest, *n.trees* was set to 5000. However, with boosting, *n.trees* being set too high will not result in overfitting; therefore, a value too high was not an issue. The two parameters tuned were *shrinkage* and *n.minobsinnode*. *Shrinkage* indicates the rate of change or influence each step has on the next step. A high *shrinkage* rate will learn quickly, but may miss the optimum, while a low *shrinkage* will learn too slowly and may never reach the optimum. *nminobsinnode* has the same influence as *nodesize* in random forests with relation to overfitting. In total, 4 (*shrinkage*) × 5 (*n.minobsinnode*) = 20 models were compared.

RESULTS

Development error rates and comparison of models for optimal hyperparameters are shown in Tables 2-5. A comparison of development and test error rates (i.e., identifying potential overfitting) is shown in Table 6. Test error rates for each of the experiments' optimal models are displayed in Figure 2 and recall rates are displayed in Figure 3.

Dataset Statistics

The number of observations within each CLASS (10) and NSP (3) classification are shown in Tables 1a and 1b. The dataset is imbalanced such that there are many more healthy/normal observations than critical/pathologic observations, most likely identical to the actual proportion of each occurrence in practice. According to Figure 1, it appears that there is a relationship between CLASS and NSP classifications such that lower CLASS categories indicate *Normal* NSP observations and higher CLASS categories indication *Pathologic* NSP observations.

Experiment 1

The 10-class classification (CLASS) experiment produced models with the worst predictive performance, as was expected. The optimal tuning parameters are shown below.

Random Forest	*Neural Network*	*Boosting*
mtry = 6, nodesize = 2	size = 140, decay = 0.1	shrinkage = 0.1, n.minobsinnode = 5

The best predictive performance (0.89) was achieved with a random forest and the best recall (0.84) was achieved with boosting.

Experiment 2

The 3-class classification (NSP) experiment produced models with the second-best predictive performance. The optimal tuning parameters are shown below.

Random Forest	*Neural Network*	*Boosting*
mtry = 3, nodesize = 2	size = 120, decay = 0.1	shrinkage = 0.1, n.minobsinnode = 1

The best predictive performance (0.95) and the best recall (0.90) were achieved with the same random forest model.

Experiment 3

The binary classification (*Suspect* coded as *Normal*) experiment produced models with the best predictive performance. The optimal tuning parameters are shown below.

Random Forest	*Neural Network*	*Boosting*
mtry = 3, nodesize = 5	size = 80, decay = 0.1	shrinkage = 0.001, n.minobsinnode = 2

The best predictive performance (0.99) and the best recall (0.95) were achieved with the same random forest model.

Experiment 4

The binary classification (*Suspect* coded as *Pathologic*) experiment produced models with the third-best predictive performance. The optimal tuning parameters are shown below.

Random Forest	*Neural Network*	*Boosting*
mtry = 7, nodesize = 1	size = 80, decay = 0	shrinkage = 0.1, n.minobsinnode = 1

The best predictive performance (0.96) was achieved with a random forest model and the best recall (0.85) was achieved with a boosting model.

DISCUSSION

In the present study, two baseline models (Logistic Regression & Naïve Bayes) and three machine learning models (Random Forest, Neural Network, & Boosting) were compared in various supervised classification scenarios using CTG data made available by the UCI Machine Learning Repository. The results corroborate findings of Sahin & Subasi (2015) showing that an optimized random forest model performed the best compared to other machine learning techniques. Boosting provided excellent predictive accuracy as well, and outperformed random forest models in some cases. These aforementioned findings were extended by showing that random forests outperformed other models in a variety of other classification cases (e.g., 10-class, tertiary, and binary). Most importantly, the results suggest that doctors are indeed overly cautious with classifying *Suspect* issues when objective data suggest that their profiles match that of *Normal* CTG readings.

Limitations

Although extremely high accuracy (0.99) and recall (0.95) rates were achieved for some of the machine learning techniques, there are indeed some limitations of the present analysis that can be addressed in future work. The size of the dataset and unbalanced nature of the observations may have hindered model performance of some techniques more than others. It was impractical to balance the observations of each class in this case because it would have lowered to sample size significantly. Nevertheless, low test error rates were obtained for some models despite this shortcoming. Another limitation of the present analyses was the focus on predictive accuracy and not variable importance measures. By exploring variable importance, doctors could

be informed of the few important features of the CTG to manually observe in order to improve diagnoses. Furthermore, a variable selection step could be added to the present analysis in which nonimportant variables are removed from the model and a simpler model is trained. This would significantly reduce computational time and model complexity. Another shortcoming of the present analysis was the limited grid search on 2 hyperparameters on each machine learning technique. More hyperparameters could be tuned and on more potential values. When reviewing Tables 2-5, if the optimal (lowest) error rate was observed at the right column or bottom row of the grid, it is possible that the actual optimal value was not searched for. However, this method was computationally exhaustive, especially for four experiments with three different techniques each and training each model 20-40 times. A final oversight was not conducting a fifth experiment by investigating binary predictive accuracy by removing the Suspect NSP observations from the dataset. This would set as a separate benchmark to compare Experiments 3 and 4 to as opposed to the tertiary classification of Experiment 2.

Conclusion

In summary, the results showed that the 10-class variable was not very useful in terms of predictive accuracy. The 3-class (NSP) classification achieved acceptable accuracy (0.95) and recall (0.90) but was greatly improved by a binary classification model in which *Suspect* observations were recoded as *Normal*. In general, random forest models produced the best predictive accuracy and highest recall. With more data, these scores should increase. In practice, it would be possible to implement a model trained on more data in order to save doctors' time. The random forest model could output a NSP suggestion based on automatic analysis of the CTG. *Normal* classifications could be disregarded, *Pathologic* classifications could be acted upon immediately, and Suspect classifications could be the only CTGs manually inspected.

REFERENCES

1. Ayres-de-Campos D, Bernardes J, Garrido A, Marques-de-Sa J, & Pereira-Leite L (2000). SisPorto 2.0: a program for automated analysis of cardiotocograms. *Journal of Maternal-Fetal Medicine*, 9(5), 311-318.

2. Dua, D & Graff, C. (2019). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.

3. Friedman J, Hastie T, & Tibshirani R (2010). Regularization Paths for Generalized Linear Models via Coordinate Descent. *Journal of Statistical Software,* 33(1), 1-22.

4. Hon, EH & Hess, OW (1957). Instrumentation of fetal electrocardiography. *Science,* 125(3247), 553-554.

5. Sahin, H & Subasi, A (2015). Classification of the cardiotocogram data for anticipation of fetal risks using machine learning techniques. *Applied Soft Computing*, *33*, 231-238.

Table 1a. Descriptive statistics for 3-class (NSP) outcome variable.

NSP	count	percent
1	1655	77.8
2	295	13.9
3	176	8.3

Table 1b. Descriptive statistics for 10-class (CLASS) outcome variable.

CLASS	count	percent
1	384	18.1
2	579	27.2
3	53	2.5
4	81	3.8
5	72	3.4
6	332	15.6
7	252	11.9
8	107	5.0
9	69	3.2
10	197	9.3

Table 2a. Random Forest – Experiment 1 development misclassification rates.

		mtry							
		2	*3*	*4*	*5*	***6***	*7*	*8*	*9*
nodesize	*1*	.129	.115	.106	.104	.099	.101	.104	.104
	2	.127	.113	.106	.106	***.099***	.104	.108	.106
	3	.134	.111	.106	.106	.101	.106	.106	.106
	4	.127	.113	.106	.104	.104	.106	.104	.108
	5	.129	.111	.104	.106	.106	.106	.106	.106

Table 2b. Neural Network – Experiment 1 development misclassification rates.

		size				
		60	*80*	*100*	*120*	***140***
decay	*0*	.296	.421	.355	.336	.367
	0.001	.266	.362	.296	.282	.263
	0.01	.285	.300	.289	.261	.273
	0.1	.344	.275	.264	.320	***.256***

Table 2c. Boosting – Experiment 1 development misclassification rates.

		shrinkage			
		0	*0.001*	*0.01*	***0.1***
n.minobsinnode	*1*	.831	.160	.125	.111
	2	.831	.155	.113	.122
	3	.831	.155	.113	.120
	4	.831	.155	.111	.118
	5	.831	.151	.108	***.106***

Table 3a. Random Forest – Experiment 2 development misclassification rates.

		mtry							
		2	***3***	*4*	*5*	*6*	*7*	*8*	*9*
nodesize	*1*	.068	.064	.064	.064	.064	.061	.059	.061
	2	.071	***.061***	.066	.066	.064	.061	.059	.064
	3	.071	.064	.066	.066	.066	.064	.061	.061
	4	.071	.064	.066	.066	.066	.064	.64	.061
	5	.073	.066	.066	.066	.066	.064	.064	.061

Table 3b. Neural Network – Experiment 2 development misclassification rates.

		size				
		60	*80*	*100*	***120***	*140*
decay	*0*	.101	.118	.094	.094	.125
	0.001	.127	.118	.120	.113	.101
	0.01	.146	.122	.125	.108	.111
	0.1	.125	.132	.113	***.092***	.104

Table 3c. Boosting – Experiment 2 development misclassification rates.

		shrinkage			
		0	*0.001*	*0.01*	***0.1***
n.minobsinnode	***1***	.216	.080	.054	***.044***
	2	.216	.080	.061	.049
	3	.216	.082	.059	.054
	4	.216	.080	.071	.056
	5	.216	.082	.071	.054

Table 4a. Random Forest – Experiment 3 development misclassification rates.

		mtry							
		2	***3***	*4*	*5*	*6*	*7*	*8*	*9*
nodesize	*1*	.014	.012	.012	.012	.012	.012	.012	.012
	2	.014	.012	.012	.012	.012	.012	.012	.012
	3	.014	.012	.012	.012	.012	.012	.012	.012
	4	.014	.012	.012	.012	.012	.012	.012	.012
	5	.014	***.012***	.012	.012	.012	.012	.012	.012

Table 4b. Neural Network – Experiment 3 development misclassification rates.

		size				
		60	***80***	*100*	*120*	*140*
decay	*0*	.024	.021	.014	.014	.014
	0.001	.024	.019	.014	.019	.014
	0.01	.019	.024	.026	.016	.019
	0.1	.024	***.014***	.014	.016	.014

Table 4c. Boosting – Experiment 3 development misclassification rates.

		shrinkage			
		0	*0.001*	***0.01***	*0.1*
n.minobsinnode	*1*	.082	.024	.014	.014
	2	.082	.024	***.012***	.014
	3	.082	.024	.014	.014
	4	.082	.024	.014	.016
	5	.082	.024	.014	.016

Table 5a. Random Forest – Experiment 4 development misclassification rates.

		mtry							
		2	*3*	*4*	*5*	*6*	*7*	*8*	*9*
nodesize	***1***	.052	.054	.049	.049	.049	***.045***	.047	.047
	2	.054	.049	.052	.052	.049	.047	.047	.047
	3	.052	.052	.052	.052	.049	.049	.049	.047
	4	.052	.052	.052	.049	.049	.049	.049	.049
	5	.052	.054	.052	.052	.049	.049	.049	.049

Table 5b. Neural Network – Experiment 4 development misclassification rates.

		size				
		60	***80***	*100*	*120*	*140*
decay	***0***	.118	***.082***	.096	.108	.127
	0.001	.100	.094	.101	.108	.106
	0.01	.104	.100	.108	.092	.101
	0.1	.108	.110	.094	.106	.094

Table 5c. Boosting – Experiment 4 development misclassification rates.

		shrinkage			
		0	*0.001*	*0.01*	***0.1***
n.minobsinnode	***1***	.216	.082	.047	***.033***
	2	.216	.085	.047	.035
	3	.216	.082	.049	.038
	4	.216	.082	.052	.040
	5	.216	.085	.049	.035

Table 6. Comparison of development and test misclassification rates.

	Random Forest			**Neural Network**			**Boosting**		
	dev	*test*	*%*	*dev*	*test*	*%*	*dev*	*test*	*%*
Exp. 1	.099	.113	+ 14	.256	.268	+ 5	.106	.127	+ 20
Exp. 2	.061	.052	- 15	.092	.117	+ 27	.044	.063	+ 43
Exp. 3	.012	.009	- 25	.014	.028	+ 100	.012	.016	+ 33
Exp. 4	.045	.047	+ 4	.082	.101	+ 23	.033	.059	+ 79

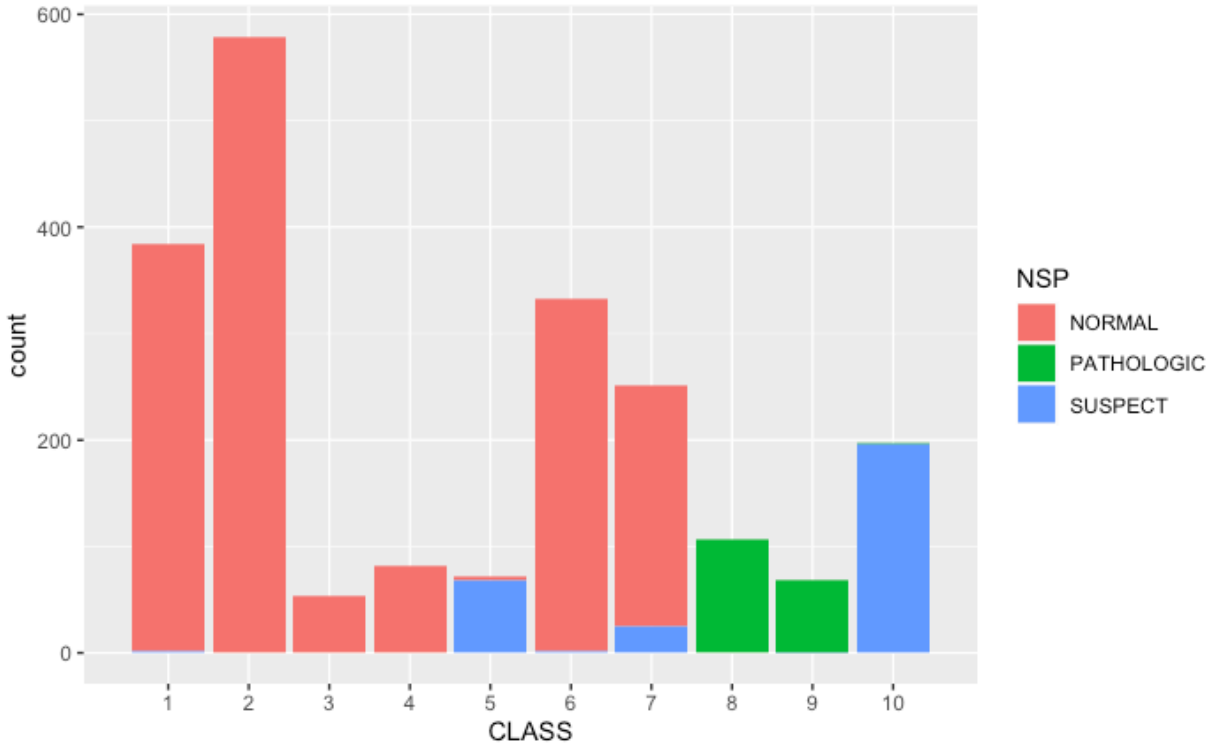

Figure 1. Relationship between outcome variables CLASS and NSP.

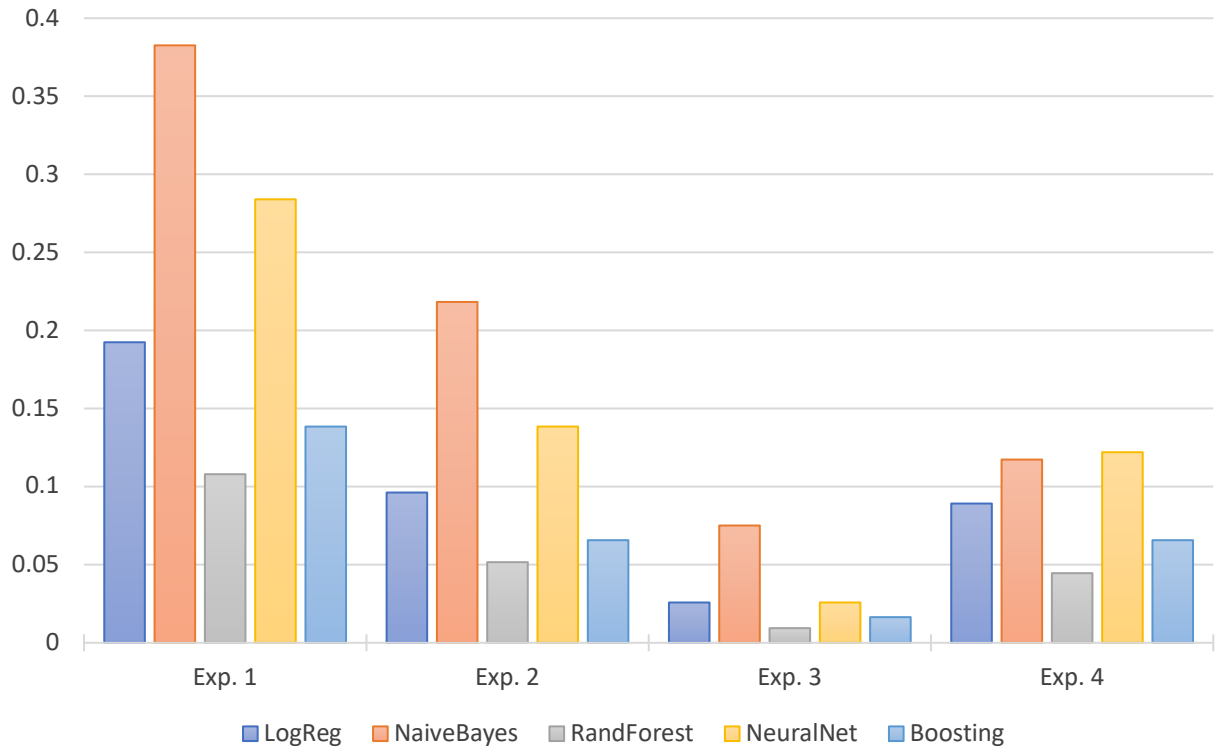

Figure 2. Test misclassification rate for each experiment's set of optimal models.

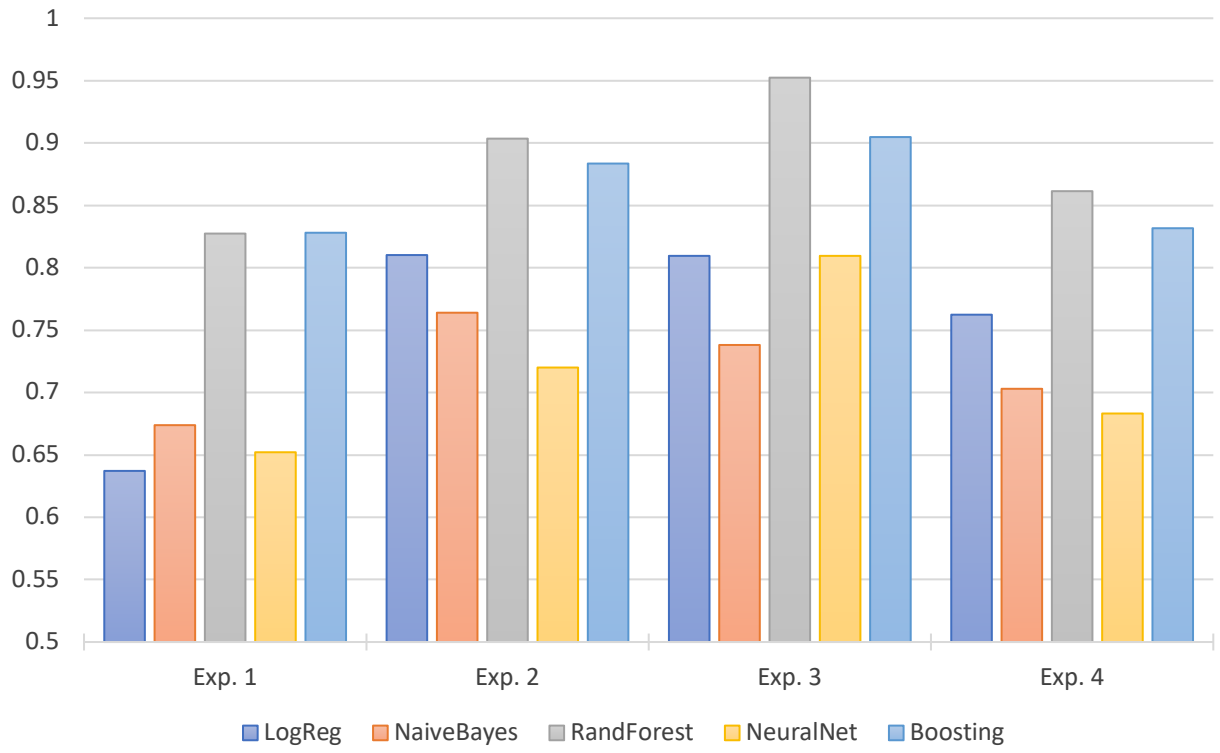

Figure 3. Test recall for each experiment's set of optimal models.