Supervised Binary Classification Model Comparison

Jack Steussie

University of California San Diego Cognitive Science Department
jsteussi@ucsd.edu

Abstract

Machine learning has seen a lot of popularity and growth in the industry and university environments within the last decade. In this paper, I present a replication of Caruana and Niculescu-Mizil's 2006 paper and I empirically evaluate the performances of five supervised machine learning algorithms: KNN, neural nets, boosted decision trees, random forest, and logistic regression. The results of my analysis of these algorithms resemble Caruana's results, but differ slightly in which algorithm ended up giving the best results overall. I will be using three metrics to evaluate these algorithms so that we can see how they perform with respect to these different metrics, and the performances will be measured across four data sets as well. The main purpose of this study is to gain a better general understanding of the algorithms evaluated while taking into account their individual advantages and disadvantages.

1 Introduction

STATLOG (King et al., 1995) was the first well-known empirical comparison of supervised learning models, and Caruana and Niculescu-Mizil (2006), henceforth referred to as CNM06, followed up STATLOG in order to provide a more thorough and up-to-date analysis of the various algorithms available. CNM06 furthered the research done by STATLOG by evaluating more algorithms and evaluating said algorithms with respect to a larger variety of metrics. Caruana experiments using SVMs, neural nets, logistic regression, naive bayes, kNN, random forests, and decision trees, on 11 different data sets, and argues that on average boosted trees and random forests perform best out of all of the algorithms while logistic regression, decision trees, stumps, and naive bayes perform worst. However, they also state that certain algorithms still have poor performance on particular data sets and metrics. CNM06 is, in the end, a great example of how algorithms within the machine learning sphere should be compared and evaluated. Having said that, this paper will follow in the footsteps of CNM06 and attempt to replicate their results with five algorithms (random forests, KNNs, boosted decision trees, logistic regression, and artificial neural networks), four datasets that were not used in CNM06 (HTRU2, HIGGS, SUSY, and BIT), and three metrics (accuracy, F-score, and AUC-ROC). It is important to do the same type of analysis as CNM06, as this paper is doing, in slightly different ways while keeping the main goal of comparison because it brings insight into how we might be able to generalize the results found in CNM06.

2 Methodology

2.1 Learning Algorithms

For each learning algorithm being evaluated, I perform a grid search that covers as much of the parameter combinations for each algorithm within the limits of computation as possible. I also use

the sklearn implementation of each algorithm exclusively. It is also worth noting that I use validation sets for the hyper parameter search. All algorithms within each dataset use features that have been scaled to 0 mean 1 std.

KNN: I use 26 values of K at increments of four ranging from K = 1 to K = 105. I use KNN with Euclidean distance and Euclidean distance weighted by gain ratio.

ANNs: I train neural nets using stochastic gradient descent back propagation and vary the number of hidden layers {1, 2, 4, 8, 32, 128}. I use the rectified linear activation function because of how widely used it is in many situations. I also vary the maximum amount of training epochs allowed {2, 4, 8, 16, 32, 64, 128, 256, 512}. I also use constant and inverse scaling of the learning rate, varying the initial starting value for the learning rate by factors of ten from 10^{-3} to 10^{0}.

Logistic Regression: I train both unregularized and regularized models, varying the regularization parameter by factors of ten from 10^{-8} to 10^{4}. The regularization types used are L1 and L2 regularizations.

Boosted Decision Trees: We use the sklearn AdaBoost implementation of boosted trees. I chose to use the SAMME.R boosting algorithm as it seemed like the best fit for our datasets and it is known to converge faster and give lower test error than the SAMME solver. I also chose to vary the estimators and chose the values {2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048}. The final parameter I adjusted was the learning rate, which took the values {1e-3, 1e-2, 1e-1, 1e0, 1e1, 2e1, 5e1}.

Random Forest: The forests have 1024 trees, and the size of the feature set considered at each split is 1, 2, 4, 6, 8, 12, 16, or 20.

2.2 Performance Metrics

I use three metrics to evaluate model performance: accuracy (ACC), area under the ROC curve (ROC), and F-score (FSC). FSC and ACC are referred to as threshold metrics, while ROC is what is referred to as a rank metric. Threshold metrics are evaluated whether or not they are above a certain threshold of correct predictions. Ranking metrics show how the model performs in its ordering of positive cases before negative cases.

Table 1. Description of Problems (Pre-Data Sampling)

PROBLEM	# ATTR	TRAIN SIZE	VAL SIZE	TEST SIZE	%POZ
HTRU2	9	n/a	n/a	n/a	9.16%
HIGGS	28	n/a	n/a	n/a	52.99%
SUSY	18	n/a	n/a	n/a	45.76%
BIT	10	n/a	n/a	n/a	1.42%
Description of Problems (Post-Data Sampling)					
HTRU2	9	5000	5000	2000	50%
HIGGS	28	5000	5000	2000	50%
SUSY	18	5000	5000	2000	50%
BIT	10	5000	5000	2000	50%

2.3 Data Sets

The algorithms are trained and tested on four data sets available from the UCI Machine Learning repository: HTRU2, HIGGS, SUSY, and Bitcoin Heist Ransomware Address Data Set (BIT) (See

Table 1). When cleaning the data in each set, I down sampled the majority class randomly without replacement and up sampled the minority class without replacement to match the amount of samples in the majority class (after the down sampling). The HTRU2 is the only data set where I up sampled the minority class with replacement because of how few samples there actually were in the minority class. After down sampling and up sampling each data set, the data sets we used each ended up containing a total of 12000 samples with 6000 samples in each class. When looking at the original BIT data set, it is a *very* imbalanced dataset with only around 1.42% of the data set representing the positive class. Down sampling and up sampling the data changed this in the end, but it is definitely still worth noting. When we exclude the target column, this data set contains a total of nine attributes that are to be used as features. I removed the Bitcoin address, year, and day attributes as there is nothing that can be predicted from these. We are then left with a total of six attributes to work with. In the original HTRU2 data set, we see that there is eight attributes if we exclude the target label, and it is a rather unbalanced dataset with 9.16% of the data representing the positive target class. No data cleaning except the up and down sampling was done on this since all of the values are numerical in nature. Similarly, the SUSY data set had no cleaning done, but it has a total of 17 attributes excluding the target label. SUSY is a pretty balanced data set. The HIGGS data set contains a total of 27 attributes if we exclude the target label and had no cleaning done to it. It is also a very balanced dataset.

3 Experiments

3.1 Performances by Metric

Table 2. Scores by Metric Averaged Over 4 Problems

MODEL	ACC	ROC	FSC	MEAN
RF	**0.795**	**0.852**	**0.790**	**0.812**
KNN	0.744*	0.794*	0.750	0.762
BST-DT	0.785	0.843*	0.780*	0.802
LR	0.725*	0.766*	0.745*	0.745
ANN	0.750*	0.803*	0.756*	0.770

For each test problem, I randomly select 5000 samples for training, 5000 for validation and 2000 for testing each trial (of which there are five). I use k-fold cross validation on the 5000 validation samples in order to find the best hyper parameters. We use a validation set in order to make sure that no information from the training data is getting leaked to the classifier, which could potentially provide artificially better performances within each metric.

Table 2 shows the score for each algorithm averaged over the set of three metrics we are measuring. Similar to what was done in CNM06, we find the best set of hyper parameters for each algorithm with the validation set, then report that model's score on the test set. Each entry in the table averages the scores of its corresponding algorithm across the five trials and four test problems. Higher scores indicate better performance in a metric. The last column in Table 2 is the mean of all the metrics with respect to a given algorithm. A **boldfaced** entry indicates that the algorithm it corresponds to is the best score out of all the other algorithms for a given metric. A * next to any entry in the table indicates that the algorithm that entry corresponds with is statistically indistinguishable from the score of the best algorithm with respect to the metric that entry corresponds to. In order to determine whether an algorithm was statistically distinguishable from the best or not, we took related paired t-tests at a value of $p = 0.05$ (refer to appendix to see the p-values of each test done).

Looking at the mean scores of each algorithm over the four problem sets with respect to each metric (see Table 2), random forests has a better score than the rest of the algorithms across the board, which corresponds to the findings of CNM06, but statistically it is only better than a couple of other algorithms with respect to a couple of the metrics (which does not correspond with CNM06). Random forests turns out to be statistically better than boosted decision trees with respect to accuracy and it is also statistically better than KNN with respect to F-score. The latter is correspondent with CNM06 while the former is not.

3.2 Performances by Problem

If we refer to the mean scores of each algorithm across all metrics with respect to each problem set (see Table 3), we can see that random forest numerically out performs all other algorithms across each problem except for SUSY, where ANN performs best. For all problems, the best classifier was statistically better than all of the other algorithms. In fact, the p-values from the t-tests conducted were all very low (all of them are below 10^{-3}. Numerically, logistic regression performed the worst over all datasets except for SUSY, where KNN performed the worst.

Table 3. Scores by Problem Averaged Over 3 Metrics

MODEL	HTRU2	HIGGS	SUSY	BIT
RF	**0.984**	**0.730**	0.803	**0.731**
KNN	0.977	0.641	0.765	0.666
BST-DT	0.975	0.716	0.805	0.713
LR	0.956	0.607	0.799	0.618
ANN	0.961	0.669	**0.810**	0.639

4 Discussion

Looking at my experimental results overall, they generally point to the same conclusion that was reached in CNM06: random forests generally performs better than the other algorithms presented. This comes with the caveat, however, that in my study, even though random forests does perform *numerically* better than most algorithms with respect to each problem over the three metrics, it does not perform *statistically* better than the rest (more often than not). When we look at the results with respect to each metric over the four problems, we do see that it performs statistically better than the rest more often than not. In the case of when the ANN performed better than random forests in the SUSY problem, this could be indicative of why neural networks have seen a growth in popularity over the past years.When we compare the performance of KNN and random forests in the test set to their respective training performances (not shown but they had near perfect training performance) we can see that this corresponds with random forests' and KNNs' tendency to over fit. Boosted decision trees performed worse on average than random forests, which coincides with the findings in CNM06. Almost all of the results in this study match those of CNM06, giving the study more merit than it did before I conducted my own version.

5 Bonus

In addition to the three required algorithm, this study explores an additional two algorithms.

References

(1) Caruana , Rich, and Alexandru Niculescu-Mizil. "An Empirical Comparison of Supervised Learning Algorithms." In Proceedings of the 23rd International Conference on Machine Learning, 2006.

(2) Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011.

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

A P-Values

A.1 Table 2 P-Values

P-Values Table 2			
MODEL	ACC(RF)	ROC(RF)	FSC(RF)
RF	1.000	1.000	1.000
KNN	0.085	0.104	0.034
BST-DT	0.046	0.254	0.144
LR	0.102	0.160	0.086
ANN	0.117	0.189	0.148

A.2 Table 3 P-Values

P-Values Table 3				
MODEL	HTRU2(RF)	HIGGS(RF)	SUSY(ANN)	BIT(RF)
RF	1.000	1.000	4.405e-4	1.000
KNN	3.264e-7	3.346e-8	3.211e-9	1.963e-9
BST-DT	7.970e-6	1.440e-3	1.191e-4	2.749e-3
LR	1.512e-8	2.822e-7	8.739e-5	5.412e-6
ANN	2.529e-7	6.249e-6	1.000	5.224e-7