An Empirical Comparison of Supervised Learning Algorithms

Darren Yau DAYAU@UCSD.EDU
University of California, San Diego
La Jolla, CA 92093, USA

Editor: Me

Abstract

This paper attempts to replicate Caruana and Niculescu-Mizil's 2006 paper by comparing the performances of 3 supervised machine learning algorithms across 4 binary classification datasets. The algorithms include logistic regression, k-nearest neighbors, and random forests. The optimal model for each algorithm is selected by conducting grid searches through sets of possible hyperparameters. The optimal models are then assessed based on their Accuracy, F1-Score, and Area Under ROC Curve, averaged across 5 trials per algorithm, per dataset. The results of our experiment confirm the findings of Caruana and Niculescu-Mizil: logistic regression is among the poorest performing models, k-nearest neighbors is slightly better but still mediocre, and random forests perform consistently well across all performance metrics and datasets of various balances.

1. Introduction

Caruana and Niculescu-Mizil's 2006 paper (CNM06) brings outdated studies comparing learning algorithms into the modern era. The most well-known of these studies, STATLOG (King et al., 1995), was comprehensive when it was performed, but since then new algorithms with excellent performance have emerged. Modern learning methods such as bagging, boosting, SVMs, and random forests had not yet been extensively and empirically evaluated. CNM06 remedies this problem by evaluating more recent algorithms.

CNM06 also provides a more detailed evaluation of algorithms through domain-specific performance metrics. For example Precision/Recall measures are used in information retrieval; medicine prefers ROC area, etc. Different performance metrics measure different tradeoffs in the predictions made by a classifier, and it is possible for learning methods to perform well on one metric, but be suboptimal on other metrics. For this reason, CNM06 evaluates algorithms on a broad set of performance metrics.

CNM06 finds that random forests, bagged trees, and neural nets generally perform the best, averaged across eight performance metrics and eleven datasets. Similarly, the poorest performing models are naive bayes, logistic regression, and decision trees. Regardless, Caruana and Niculescu-Mizil remind us that there is no Free Lunch and even the best models perform poorly on some problems, and vice versa.

This paper attempts to replicate the findings of CNM06 on a smaller subset of algorithms (logistic regression, k-nearest neighbors, random forests), datasets (ADULT, COVTYPE, LETTER, DOTA), and performance metrics (ACC, FSC, AUROC). A smaller-scale repli-

cation of CNM06 would test the robustness of its original findings, as well as uncover any potential consequences associated with the specialization of any single result.

2. Methodology

2.1 Learning Algorithms

We utilize 3 different learning algorithms for this study. This section summarizes the set of possible parameters tested on each algorithm. The optimal parameters are then selected after a 5-fold cross validation grid search.

Logistic Regression (LOGREG): we train both unregularized and regularized models, varying the ridge (regularization) parameter by factors of 10 from 10^{-8} to 10^{4}. We use both l1 and l2 penalties on regularized models. We use newton-cg, lbfgs, liblinear, sag, and saga for the solvers, where appropriate.

KNN: we use 26 evenly-spaced values of K ranging from $K = 1$ to $K = 101$. In our opinion, it is ridiculous to investigate up to $K = |trainset|$ as in CNM06. A K that large just votes for the class with the most examples every time. We use KNN with Euclidean distance. We also use uniform and distance weighted KNN.

Random Forests (RF): every forest has 1024 trees. The size of the feature set considered at each split is 1,2,4,6,8,12,16 or 20.

We scale numerical attributes to 0 mean 1 std for all algorithms. In total, we train about 700 different models in each trial on each problem. We train about 14,000 unique models throughout this paper.

2.2 Performance Metrics

Algorithmic performance is measured across 3 metrics: Accuracy (ACC), F1-Score (FSC), and Area Under ROC Curve (AUROC). ACC is a threshold metric ranging from $[0, 1]$ and can be measured as follows:

$$ACC = \frac{TP + TN}{P + N} \tag{1}$$

FSC is a threshold metric ranging from $[0, 1]$ and is defined as follows:

$$FSC = \frac{2TP}{2TP + FP + FN} \tag{2}$$

AUROC is a rank metric and is calculated by finding the area under the precision-recall curve generated with varying thresholds.

2.3 Data Sets

We compare the algorithms on 4 binary classification problems. ADULT, COVTYPE, LETTER, and DOTA are all available online from the UCI Repository.

ADULT was readily available as a binary classification problem. Samples with $> 50k$ income are treated as positives, and those with $\leq 50k$ income as negatives. This yields a rel-

atively imbalanced dataset, with 24.1% positive labels and 75.9% negative labels. Nominal attributes are present, so one-hot encoding was employed.

COVTYPE has been converted to a binary problem by treating the largest class as the positive and the rest as negative. This yields a balanced dataset, with 48.8% of the samples being positive and 51.2% being negative. Nominal attributes are present, so one-hot encoding was employed.

LETTER was booleanized by treating letters A-M as positives and the rest as negatives, yielding a well-balanced problem. 49.7% of the samples are positive and 50.3% are negative.

DOTA was readily available as a binary classification problem. Samples who won were treated as positives, and those who lost as negatives. By this definition, the dataset is 52.7% positive and 47.3% negative. Nominal attributes, surprisingly in the form of integers, are present, so one-hot encoding was employed.

See Table 1 for more detailed characteristics of these problems.

Table 1. Description of problems

PROBLEM	#ATTR	TRAIN SIZE	TEST SIZE	%POZ
ADULT	14	5000	25162	24.1%
COVTYPE	13	5000	576012	48.8%
LETTER	16	5000	15000	49.7%
DOTA	13	5000	87650	52.7%

3. Performances by Metric

Each algorithm and problem combination is subject to a 5-fold cross validation grid search to find the optimal hyperparameters. 5000 random samples are initially chosen as training data for this grid search, while the rest of the data compose the test set. Once the optimal hyperparameters are found from the training set, we train the algorithm once more on the test set, and measure its performance on a variety of metrics. This entire process constitutes 1 trial, and a total of 5 trials are conducted for each algorithm and problem combination. We use the averages of the metrics over all trials to determine the overall scores.

Table 2 shows the normalized score for each algorithm on each of the 3 metrics. Each entry is an average over the 4 problems. The last column, MEAN, is the mean normalized score over the 3 metrics, 4 problems, and 5 trials. In the table, the algorithm with the best performance on each metric is **boldfaced**. Other algorithms whose performance is not statistically distinguishable from the best algorithm at $p = 0.05$ using paired t-tests on the 5 trials are labeled with a *****. Entries in the table that are neither bold nor starred indicate performance that is significantly lower than the best models at $p = 0.05$.

Table 2. Normalized testing scores for each learning algorithm by metric

MODEL	ACC	FSC	AUROC	MEAN
LOGREG	0.727	0.692	0.706	0.708
KNN	0.779	0.746	0.754	0.760
RF	**0.796**	**0.767**	**0.777**	**0.780**

For all 3 performance metrics, RF appears as the clear winner among the 3 algorithms (Table 2). Its mean ACC is significantly higher than those of LOGREG and KNN across all 5 trials ($p = 0.0049, 0.0004$) (Table 2 Supplemental in Appendix). The same can be said for FSC ($p = 0.0015, 0.0010$) and AUROC ($p = 0.0032, 0.0001$). LOGREG appears to perform the worst across the board, while KNN's performance consistently falls between RF and LOGREG.

4. Performances by Problem

Table 3. Normalized testing scores for each learning algorithm by problem

MODEL	ADULT	COVTYPE	LETTER	DOTA	MEAN
LOGREG	0.757	0.751	0.726	**0.599**	0.708
KNN	0.737	0.783	**0.951**	0.569	0.760
RF	**0.768**	**0.824**	0.946	0.582	**0.780**

Table 3 shows the normalized score for each algorithm on each of the 4 problems. Each entry is an average over the 3 performance metrics. RF continues to dominate LOGREG and KNN on the ADULT ($p = 0.0001, 0.0001$) and COVTYPE ($p = 0.0001, 0.0001$) datasets (Table 3 Supplemental in Appendix). However, KNN actually outperforms RF on the LETTER dataset by a significant margin ($p = 0.004$). Surprisingly, even LOGREG takes home a win as it outperforms RF on the DOTA dataset by a significant margin ($p = 0.0001$). We are again reminded of the No Free Lunch Theorem, which suggests that there is no universally best learning algorithm. Even the best models (RF) perform poorly on some problems, and sometimes the worst performing models (LOGREG) perform well on other problems.

5. Conclusion

Our results seem to confirm the findings of Caruana and Niculescu-Mizil's 2006 paper. Random forests perform consistently well across all performance metrics and datasets, implying a certain robustness within the algorithm. Logistic regression has the worst general performance across all metrics, although it performs the best on the DOTA dataset. K-nearest neighbors also performs poorly in general, but its stellar performance on the LETTER dataset nets it a spot just above logistic regression, yet still well below random forests.

The DOTA dataset poses the hardest task for our learning algorithms, achieving only a maximum aggregate score of 0.599 (Table 3) with logistic regression. We suspect that this is an inherent quality of the dataset rather than a problem with our algorithms. Because the DOTA dataset is centered around a video game, it makes sense that the developers would balance certain team compositions to ensure equal play. This means that the better the developers balance the game, the harder it is for any algorithm to predict winners/losers based solely on team compositions. Anything above pure chance (0.5000) indicates that winners/losers can be predicted in the absence of individual player skills, which is definitely not what the developers intended.

Overall, we consider this paper a successful small-scale replication of CNM06. We look forward to evaluating new learning algorithms that may be developed in the future.

Acknowledgments

We would like to acknowledge support for this project from the the COGS 118A instructional staff, Dr. Jason Fleischer, and helpful classmates on Piazza. And Stack Overflow.

Appendix

Table 2 Supplemental. P-values by metric

MODEL	ACC	FSC	AUROC
LOGREG	0.0049	0.0015	0.0032
KNN	**0.0004**	**0.0010**	**0.0001**
RF	1.0000	1.0000	1.0000

Table 3 Supplemental. P-values by problem

MODEL	ADULT	COVTYPE	LETTER	DOTA
LOGREG	0.0001	**0.0001**	**0.0001**	1.0000
KNN	**0.0001**	0.0001	1.0000	**0.0001**
RF	1.0000	1.0000	0.0004	0.0001

Table 4. Normalized training scores for each learning algorithm by dataset

MODEL	ACC	FSC	AUROC	MEAN
LOGREG	0.737	0.701	0.715	0.718
KNN	0.879	0.874	0.975	0.909
RF	1.000	1.000	1.000	1.000

Table 4 shows the normalized training score for each algorithm on each of the 3 metrics. The testing score for each algorithm and metric combination (Table 2) is lower than its respective training score (Table 4). This illustrates the negative effects of generalizing a trained model onto new data. Random forests suffer the most from generalization errors, with all performance metrics dropping from 1.000 to around 0.780 (Table 2). This suggests that random forests are actually overfitting the training data and introducing more variance into the model than necessary. Regardless, random forests still outperform logistic regression and k-nearest neighbors, implying that it is indeed a more robust model, even with overfitting.

Table 5. Raw testing scores for all trials

MODEL	PROBLEM	ACC	FSC	AUROC
LOGREG	ADULT	0.846	0.660	0.764
		0.847	0.664	0.760
		0.846	0.664	0.761
		0.845	0.665	0.764
		0.847	0.660	0.763
	COVTYPE	0.752	0.750	0.752
		0.749	0.749	0.750
		0.750	0.746	0.750
		0.752	0.750	0.752
		0.755	0.754	0.755
	LETTER	0.723	0.723	0.723
		0.730	0.733	0.730
		0.724	0.725	0.725
		0.723	0.722	0.723
		0.728	0.732	0.728
	DOTA	0.587	0.624	0.583
		0.585	0.627	0.580
		0.587	0.629	0.583
		0.584	0.620	0.580
		0.584	0.647	0.582
KNN	ADULT	0.834	0.637	0.742
		0.832	0.641	0.755
		0.832	0.633	0.750
		0.832	0.625	0.746
		0.833	0.619	0.743
	COVTYPE	0.784	0.782	0.785
		0.782	0.783	0.782
		0.783	0.784	0.784
		0.784	0.785	0.785
		0.780	0.781	0.781
	LETTER	0.956	0.955	0.948
		0.953	0.953	0.943
		0.951	0.951	0.946
		0.954	0.954	0.949
		0.953	0.952	0.947
	DOTA	0.553	0.623	0.544
		0.545	0.607	0.538
		0.546	0.613	0.538
		0.550	0.619	0.542
		0.548	0.630	0.538

Table 5. (continued)

MODEL	PROBLEM	ACC	FSC	AUROC
RF	ADULT	0.848	0.677	0.778
		0.848	0.681	0.783
		0.847	0.674	0.774
		0.852	0.685	0.782
		0.851	0.673	0.772
	COVTYPE	0.819	0.815	0.823
		0.826	0.825	0.828
		0.816	0.815	0.825
		0.824	0.826	0.824
		0.830	0.829	0.830
	LETTER	0.949	0.948	0.949
		0.945	0.944	0.944
		0.948	0.948	0.947
		0.947	0.947	0.946
		0.944	0.944	0.945
	DOTA	0.570	0.626	0.563
		0.560	0.615	0.556
		0.567	0.630	0.558
		0.566	0.628	0.559
		0.565	0.619	0.559

References

[1] R. Caruana and A. Niculescu-Mizil. (2006). "An empirical comparison of supervised learning algorithms." *In Proceedings of the 23rd international conference on Machine learning,161-168.*

[2] Sklearn.linear_model.logisticregression. Retrieved March 18, 2021, from https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html

[3] Sklearn.neighbors.kneighborsclassifier. Retrieved March 18, 2021, from https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html

[4] Sklearn.ensemble.randomforestclassifier. Retrieved March 18, 2021, from https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

[5] Sklearn.model_selection.gridsearchcv. Retrieved March 18, 2021, from https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html