Exploring Supervised Classification Algorithms

Andrew Lee aglee@ucsd.edu
Department of Cognitive Science
University of California, San Diego
La Jolla, CA 92093-5004, USA

Editor: Andrew Lee

Abstract

This paper describes a small scale replication of Caruana and Niculescu-Mizil's in which three supervised machine learning algorithms are implemented to classify four different binary classification problems. The different algorithms explored in this paper are Logistic Regression, Naive Bayes, and Random Forest. For each algorithm, the optimal hyper-parameters were found using a grid search (with 5-fold cross validation), and the efficacy of each set of algorithm was determined by three error metrics: accuracy, f1-score, and precision. The results were very similar to that of Caruana's paper: Random Forest performed the best on average, followed by logistic regression, and naive bayes performed the worst on average.

Keywords: Naive Bayes, Random Forest, Logistic Regression, Binary Classification

1. Introduction

Caruana and Nisculescu-Mizil in their paper commonly referred to as CNM06, executed an empirical comparison of supervised machine learning algorithms (2006). They used ten supervised algorithms, including support vector machines, neural nets, logistic regression, naive bayes, memory-based learning, random forests, decision trees, bagged trees, boosted trees, and boosted stumps (2006). CNM06 used 11 data sets. In this paper, 3 of these algorithms were examined (Naive Bayes, Logistic Regression, and Random Forest) using 3 of the same data sets and 1 outside data set.

The purpose of this paper is to create a small-scale replication of CNM06 in order to verify the veracity of CNM06's results. This paper confirms whether the results in CNM06 would hold using only a smaller subset of algorithms, data sets, and error metrics than used in CNM06. In addition, by adding an outside data set to the analysis, we are able to further substantiate whether the results from CNM06 were accurate across extended data sets.

2. Methods

2.1 Supervised Algorithms

The 3 learning algorithms used for this paper are random forests, naive bayes, and logistic regression. Each algorithm had different hyper-parameters that were explored when using grid search 5-fold cross validation. The hyper-parameters being searched were kept

as similar as possible to those in CNM06, but were modified to account for a resonable computation time on local machines.

Naive Bayes (NB)

Cross validation on the additive smoothing parameter (alpha) ranging from 10^{-8} to 10^{4} where the exponent increased in +1 integer steps was used to find the optimal version of the Bernoulli Bayes model (Dua et al., 2019).

Logistic Regression (LOG)

Cross validation on the logistic regression model was used to find the optimal hyper-parameters. The hyper-parameter search included constant value (C) ranging from 10^{-4} to 10^{4}, a penalty of either L1 or L2 loss, and 3 different solvers: sag, saga, and liblinear (Dua et al., 2019).

Random Forest (RF)

Cross validation on this ensemble method was used to find the optimal hyper-parameters. The hyper-parameter search included number of estimators (n_estimators) ranging from 2^{1} to 10^{7}, an impurity decrease (criterion) of gini or entropy, and a max feature size (max_feature) of $[1, 2, 4, 6, 8, 12, 16, 20]$ (Dua et al., 2019).

2.2 Performance Metrics

The three performance metrics used were accuracy, f1-score, and precision. Below are brief descriptions and an equation used to calculate each error metric. Note that in the equations below, TP/TN represents true positive/true negative and likewise, FP/FN represents false positive/false negative.

Accuracy (ACC)

Accuracy is a score ranging from [0,1] that is essentially the percent correct that the classifier labeled (2021).

$$\text{Accuracy} = \frac{TP+TN}{TP+TN+FP+FN}$$

Precision (PREC)

Precision is a score ranging from [0,1] that represents the closeness of the data. In other words, it shows how consistent the results were regardless of their accuracy (2021).

$$\text{Precision} = \frac{TP}{TP+FP}$$

F1-score (FSC)

The F1-score is a score ranging from [0,1] that represents the mean between the precision (defined above) and recall (TP/(TP + FN)) (2021).

$$\text{F1-score} = \frac{2*TP}{2*TP+FP+FN}$$

2.3 Data Sets

The 4 data sets used in this paper were all pulled from the UCI Machine Learning repository: letter recognition (LET), adult (ADT), covertype (COV), and king vs king and rook (KRK). Letter was the most balanced data set, containing an almost exact 50/50 split between positive and negative classes. The adult and covertype data sets had much lower positive classes than negative, and the King versus King and rook had almost all positive classes (which is logical if you are familiar with chess endgames).

LET

The letter recognition data set had 16 attributes, and since the classification was a letter, the original problem was not a binary classification problem (Dua et al., 2019). The data set was modified so that letters ranging from A-M were labeled as the positive class, and N-Z as the negative class (R. Caruana, 2006).

ADT

The adult data set was originally composed of 14 attributes, and the attributes nor the classification were binary for this data set (Dua et al., 2019). Some of the attributes were marked as not important were dropped, leaving a total of 11 attributes once cleaned. From there, each attribute was one-hot encoded, and the classification was marked as the positive class when income was $> 50k$ and negative when income was $<= 50k$.

COV

The covertype data set contained 13 attributes, and all these labels were binary, so no further cleaning was needed on the features (Dua et al., 2019). The covertype classification distinction was replicated from CNM06, labeling the largest class (7) as the positive class and the rest of the classes (1-6) as the negative class (R. Caruana, 2006).

KRK

The King and rook versus king contained 6 attributes, which were the file and rank of each piece (king, king, and rook). The file and rank represent the column and row of the chessboard, which is a standard notation used to represent the coordinates for chess pieces. The ranks were already in integer form, but the files were letters, so the file attributes were one-hot encoded to make binary labels (Dua et al., 2019). The classification used was the negative class if the result was a draw and positive if the result was not a draw.

Table 1: Table 1. Data Set Summary

Name	#ATTR	TRAIN SIZE	TEST SIZE	%POS
LET	16	5000	15000	50.0%
ADT	11/73	5000	960152	25.0%
COV	13	5000	576012	3.5%
KvKR	6	5000	219440	90.0%

3. Experiment & Results

Each data set was classified using each algorithm, and there was a total of 5 trial each, resulting in a total of 60 trials ($dataset(4) \cdot algo(3) \cdot trials(5)$). In each trial, 5000 random data samples were collected from the data set as the training set, and the rest of the data was labeled as the test set. An optimal hyper-parameter search was performed using a grid search with the default 5-fold cross validation. Performance metrics include accuracy, precision, and f1-score which are displayed in tables 2-5. In these tables, the highest scores in each column are highlighted, and the (*) represents scores that are not significantly different from the best-performing (bold) score in that column. These annotations are done through two-sampled t-tests (with threshold set at p = .05) between the best performing algorithm and the rest of the algorithms per column.

Table 2: Main Matter - Mean Test Set Performance: Accuracy

Model	LET	ADT	COV	KRK
NB	.5581	.9931*	.9739*	**.9875**
LOG	.7257	.9933*	.9762*	.9874*
RF	**.9453**	**.9936**	**.9812**	.9849*

Table 3: Main Matter - Mean Test Set Performance: Precision

Model	LET	ADT	COV	KRK
NB	.6148	.8017*	.8286	.6335
LOG	.7263	**.8468**	.8443	.6332
RF	**.9453**	.8165*	**.9148**	**.7338**

Table 4: Main Matter - Mean Test Set Performance: F1-score

Model	LET	ADT	COV	KRK
NB	.4923	.7623*	.7824*	.6492*
LOG	.7256	.7299*	.8074*	.6490*
RF	**.9453**	**.8048**	**.8374**	**.7109**

From Tables 2-4, we see that the best performing algorithm by far spanning not only each error metric but almost all data sets is random forest. The random forest algorithm outperforms naive bayes and logistic regression even when the difference between the results is

fairly close such as the accuracy in adult, covertype, and king/rook data sets. One interesting outcome was that the error metrics were fairly close to each other (within $p = .05$ significance) across all data sets except the letter data set. This was verified using a two-tailed t-test where the threshold was set at $p = .05$ (Appendix, Table 2-4 supplementals). This may be perhaps, due to the fact that the letter data set was the most balanced data set, so it portrayed the most accurate performance of the various algorithms. Or, it could be due to some other factor that could be explored in a future follow up experiment.

Tables 6-8 located in the appendix show the training set error metrics for each algorithm. For all models, the training set had a slightly better performance than the test set on average. However, it was not significant, and in some cases, the test set actually performed better than the training set.

Table 5: Main Matter - Mean Test Set Performance per Algorithm

Model	ACC	PREC	FSC
NB	.8782	.7196	.6716
LOG	.9207*	.7625	.7280
RF	**.9763**	**.8526**	**.8246**

Table 5 clearly illustrates that on average, random forest performs the best across all error metrics and data sets. The accuracy of logistic regression is not significantly different from random forest, but the precision and f1-score of random forest surmount the other algorithms.

4. Conclusion & Discussion

The result of this small-scale replication contributes to the accuracy of the results in CNM06. In other words, the algorithms used in this paper along with their results for each data seem to align with the results of CNM06. In terms of accuracy, precision, and f1-score, random forest was the best performing on average (Table 5). The second best performing algorithm was logistic regression, followed by naive bayes in last place. The accuracy of each algorithm was not significantly different between the adult, covertype, and king/rook data sets. However, there were some interesting nuances discovered in the letter data columns of the main matter tables (Table 2-4). The more balanced the data set was (Table 1), the higher the discrepancy between the error metrics of each algorithm was. This may lend support to evidence that some algorithms are more efficient at classifying unbalanced data sets, such as naive bayes which has a very low run time, while algorithms such as random forest are better for balanced data sets and overall average performance but will take longer to train.

Appendix

Table 6: Mean Training Set Performance: Accuracy

Model	LET	ADT	COV	KRK
NB	.5593	.9933	.9758	.9873
LOG	.7282	.9942	.9782	.9875
RF	1.0000	.9999	.9999	.9901

Table 7: Mean Training Set Performance: Precision

Model	LET	ADT	COV	KRK
NB	.6145	.7820	.8164	.63208
LOG	.7283	.8869	.8717	.6321
RF	1.0000	.9981	.9999	.8812

Table 8: Mean Training Set Performance: f1-score

Model	LET	ADT	COV	KRK
NB	.4959	.7466	.8076	.6484
LOG	.7280	.7485	.8225	.6484
RF	1.0000	.9964	.9994	.8331

Table 9: Raw training Set Scores (5 Trial Each): Accuracy

Model	LET	ADT	COV	KRK
NB	.5610	.9932	.9774	.9854
	.5544	.9926	.9752	.9882
	.5592	.9914	.9728	.9876
	.5612	.9966	.9744	.9878
	.5606	.9928	.9790	.9876
LOG	.7396	.9942	.9762	.9876
	.7268	.9960	.9780	.9876
	.7316	.9948	.9760	.9870
	.7262	.9938	.9818	.9888
	.7166	.9918	.9794	.9866
RF	1.0000	1.0000	.9998	.9902
	1.0000	1.0000	1.0000	.9904
	1.0000	.9998	1.0000	.9898
	1.0000	.9998	1.0000	.9902
	1.0000	.9998	.9998	.9902

Table 10: Raw training set scores (5 Trial Each): Precision

Model	LET	ADT	COV	KRK
NB	.6297	.8106	.8137	.6272
	.6037	.5844	.8343	.6339
	.6130	.8040	.7856	.6337
	.6085	.8680	.7903	.6343
	.6178	.8428	.8579	.6313
LOG	.7396	.9392	.8639	.6343
	.7271	.9257	.8631	.6325
	.7318	.8810	.8713	.6298
	.7263	.8431	.8850	.6345
	.7168	.8454	.8751	.6295
RF	1.0000	1.0000	.9998	.8756
	1.0000	1.0000	1.0000	.8682
	1.0000	.9967	1.0000	.8938
	1.0000	.9971	1.0000	.8774
	1.0000	.9969	.9998	.8911

Table 11: Raw training set scores (5 Trial Each): F1-score

Model	LET	ADT	COV	KRK
NB	.4909	.7675	.8254	.6457
	.4892	.6197	.8045	.6494
	.5044	.7381	.7804	.6493
	.4997	.8370	.8051	.6496
	.4952	.7709	.8224	.6480
LOG	.7395	.7517	.7961	.6496
	.7267	.8086	.8344	.6486
	.7315	.7631	.8242	.6471
	.7261	.7119	.8322	.6497
	.7164	.7074	.8258	.6470
RF	1.0000	1.0000	.9986	.8266
	1.0000	1.0000	1.0000	.8406
	1.0000	.9932	1.0000	.8661
	1.0000	.9941	1.0000	.8291
	1.0000	.9947	.9985	.8029

Table 2 Supplemental - p-values (p) and test statistic (t)

Model	LET_p	LET_t	ADT_p	ADT_t	COV_p	COV_t	KRK_p	KRK_t
NB	.7580	-0.3527	.9997	-0.0003	.9962	.0052	1.0000	0.0000
LOG	.8707	-0.1842	.9998	-0.0002	.9974	.0036	.9999	-7.1609
RF	1.0000	0.0000	1.0000	0.0000	1.0000	0.0000	.9987	.0018

Table 3 Supplemental - p-values (p) and test statistic (t)

Model	LET_p	LETt_t	ADT_p	ADTt_t	COV_p	COVt_t	KRK_p	KRKt_t
NB	.7970	-0.2931	.9726	-0.0386	.9506	-0.0698	0.9270	-0.1034
LOG	.8712	-0.1837	1.0000	0.0000	.9599	-0.0566	.9506	-0.0698
RF	1.0000	0.0000	.9818	-0.0257	1.0000	0.0000	1.0000	0.0000

Table 4 Supplemental - p-values (p) and test statistic (t)

Model	LET_p	LET_t	ADT_p	ADT_t	COV_p	COV_t	KRK_p	KRK_t
NB	.7121	-0.4250	.9729	-0.0383	.9661	-0.0480	.9547	-0.0641
LOG	.8707	-0.1843	.9513	-0.0689	.9817	-0.0258	.9545	-0.0643
RF	1.0000	0.0000	1.0000	0.0000	1.0000	0.0000	1.0000	0.0000

Table 5 Supplemental - p-values (p) and test statistic (t)

Model	ACC_p	ACC_t	PREC_p	PREC_t	FSC_p	FSC_t
NB	.9472	.0747	.9160	-0.1192	.8988	-0.1438
LOG	.9707	-0.0414	.9444	-0.0787	.9380	-0.0878
RF	1.0000	0.0000	1.0000	0.0000	1.0000	0.0000

References

Confusion matrix. (2021, February 27). Retrieved March 01, 2021, from https://en.wikipedia.org/wiki/Confusion_matrix

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.

Instructions for formatting jmlr articles. (n.d.). Retrieved March 01, 2021, from https://www.jmlr.org/format/format.html

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, D., Brucher, M., Perrot, M., Duchesnay, E. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research, 12, 2825–2830.*

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