Mini-Replication of CNM06: Classification Model Comparison

Anjali Ramesh

Department of Cognitive Science
University of California, San Diego, San Diego, CA 92093

Abstract

This paper is a semi-replication of Caruana and Niculescu-Mizil's 2006 paper, *"An Empirical Comparison of Supervised Learning Algorithms"*. The performance of three different supervised machine learning algorithms on binary classification problems across four data sets is compared. The algorithms compared are Support Vector Machines (SVM), Logistic Regression (LR), and K Nearest Neighbors (KNN). The algorithms are tuned using grid searches combined with 5 k-fold cross validation. The optimal fit of each algorithm is analyzed using three different metrics: accuracy, AUC score, and FSC score, which are averaged across five trials per algorithm for each data set.

Keywords: CNM06: Caruana and and Niculescu-Mizil (2006), SVM: support vector machine, LR: logistic regression, KNN: k-Nearest Neighbors, MAGIC: Magic Gamma Telescope dataset, LETTER: Letter dataset, A-M positive class classification, CALHOUS: California Housing dataset, COVER: Cover Type dataset

1. Introduction

This project is a replication of the method found in CNM06 to compare various machine learning algorithms. In this smaller scale replication, I compared four different binary classification datasets (3 found in CNM06, 1 outside source) across three models: SVM, LR and KNN, using three performance metrics taken from CNM06: AUC score, accuracy, and F1 score.

Although the CNM06 paper mentions that certain models, such as SVM, are not meant to be used for predicting probabilities, and uses two different calibration methods to help with this deficiency, neither Platt Scaling nor Isotonic Regression were used in this replication (Caruana). For this reason, and also because of other various discrepancies in methodology between CNM06 and this smaller scale replication, the results are not completely consistent with those of CNM06. Because of this, these results are very much preliminary and require much more testing and visualization in order to be truly considered valid.

After comparing many more models across a larger number of datasets, CNM06 concludes that boosted trees and random forest models perform the best on average, while logistic regression and naive bayes perform the worst. The results obtained from this replication were somewhat similar to those in CNM06, in that KNN performed better than both SVM and LR, although not by as wide a margin as in the original paper. Table 3 in CNM06 shows that KNN with no calibration performed slightly better than SVM with no calibration, which is a similar outcome to the one derived from this paper. However, LR in this replication was not the worst performing model as it was in CNM06.

Overall, although these results were obtained using fewer models, fewer

datasets, and evaluated using fewer metrics, the overall trend seen in CNM06 can be replicated to a certain extent and taken with a grain of salt to account for the various discrepancies and shortcoming of this smaller scale replication.

2. Methodology

2.1 Learning Algorithms

Three different learning algorithms were compared in this study: SVM, logistic regression, and KNN. The hyperparameters for each of these algorithms were taken from CNM06, but with slight modifications made. Other than the following parameters, the remaining parameters were set to default. These hyperparameters were then put inside of a search space which was used by the grid search algorithm with stratified k-folds to get the optimally tuned metrics.

Logistic Regression (LR) was trained using both unregularized and regularized models, varying the ridge (regularization) parameter by factors of 10 from 10^-8 to 10^4 (CNM06).

Support Vector Machine (SVM) was trained using the following kernels in SVC(): linear, polynomial degree 2 & 3, and radial with width {0.01,0.05,0.1,0.5,1,2}. The regularization parameter by factors of ten from 10−1 to 10^2 with each kernel.

k-Nearest Neighbors (KNN) was trained using 25 k values between 5 and 101, evenly spaced 4 values apart. The distance metric was set as Euclidean instead of the default Minkowski distance, since this was the one that consistently performed the best. However, as mentioned in CNM06, the grid search parameters contained both distance weighted KNN, and locally weighted averaging.

2.2 Performance Metrics

Each learning algorithm was evaluated using three metrics, AUC score (AUC), accuracy (ACC), and F1-score (FSC). ACC and F1-score are threshold metrics that range from 0 to 1, and "measure how well the positive cases are ordered before negative cases, and can be viewed as a summary of model performance across all possible thresholds" (Caruana). The AUC, or area under the ROC curve, also ranges in value from 0 to 1 and is a rank metric. This value measures the ability of a classifier to distinguish between classes, and the area under a probability curve that plots true positive rate versus false positive rate.

The threshold metrics can be calculated using the following equations:

$$\boldsymbol{accuracy} = \frac{\sum \boldsymbol{true\ positive} + \sum \boldsymbol{true\ negative}}{\sum \boldsymbol{total\ population}}$$

$$F_1 = 2 * \frac{precision * recall}{precision + recall}$$

2.3 Data Sets

Although many datasets were cleaned and explored, the datasets shown here are the ones that were ultimately used to perform these models on. All four datasets were taken from the UCI Machine Learning repository, which has free available datasets. These are MAGIC, LETTER, CALHOUS, and COVTYPE (see Table 1). Other cleaned datasets were either outputting extremely unsatisfactory results, such as LETTER PT. 1 from CNM06 or were

too large to run many C values on, such as ADULT.

MAGIC is the only dataset that was not used in CNM06, so this dataset was classified as closely as possible to a similar dataset used in the paper. Each instance was classified as either gamma (signal) or hadron (background). This was changed into a binary classification with gamma being the positive class and hadron being the negative class. All the attributes were continuous so no further cleaning was needed. This dataset had a slightly larger positive class, at 65%, than negative class.

LETTER consists of 16 attributes, all of which are numerical. Since each instance was not assigned a class, the letters A-M were classified as positive and the rest negative, as shown in CNM06. This dataset was rather well balanced, with both classes containing 50% of the data.

CALHOUS consists of 9 attributed. The dataset was not originally binary. The positive class contained instances where the median house value was greater than $130,000, and the rest were contained in the negative class. All the attributes in this dataset are numerical, so no further cleaning was needed.

COVTYPE first consisted of 13 attributes, and was not classified into a positive and negative class. As per CNM06, the largest class was classified as positive, and the rest was negative. This results in a slightly unbalanced positive and negative class. None of these attributes needed to be processed further.

Table 1: Problem Set Summary

Name	#ATTR	TRAIN SIZE	TEST SIZE	%POS
MAGIC	11	5000	14020	65%
LETTER	16	5000	14000	50%
CALHOUS	9	5000	14640	52%
COVTYPE	13/54	5000	25000	36%

3. Experiment & Results

3.1 Training and Testing Framework

Following the methods set in CNM06, for each algorithm and dataset combination, 5000 data samples were randomly chosen for the training set with replacement. 5-fold cross validation was performed on the training data to select the hyperparameters via a systematic gridsearch of the parameter space. These hyperparameters were tuned differently for each model SVM, LR and KNN. This process was repeated over five trials per algorithm/dataset combination. The model was then trained one more time on all 5000 training data samples and model performance was measured on the test set (all the data in the dataset other than the 5000 random data samples that were used as the training set), using the optimal hyperparameters that were found during the grid search and cross validation portion. The average AUC, ACC, and FSC metrics were computed and reported at the end of each trial for each trial and algorithm/dataset combination.

3.2 Performance

Tables 2, 3, 4, and 5 show the results and model performance from this experiment. In Table 2, each row is one model, and each column is one performance metric, with an additional column representing the mean value for each model. These values have been calculated by averaging each metric across all four datasets, and repeated for each model.

In Table 3, each row is still one model, while the columns represent each of the four datasets used in this experiment. These values have been calculated by average all three metrics together on a single dataset, and calculated for each model.

For each metric (AUC, ACC, or FSC) or dataset, the highest performing model is bolded. Scores that have asterisks were not significantly different from the best score in each column, and performed about the same, according to an independent sample t-test. An independent sample t-test was used because there was no random seed set as a parameter for the cross validation, meaning that each of the 5000 samples was taken randomly. These tests were performed between each value compared to the best performing metric in that column. The threshold for significance for the hypothesis test is set at p 0.05. As noted in both CNM06 and in the instructions, “performing this many independent t-tests, each at p = 0.05, is problematic. Some differences that are labeled significant in the table probably are not truly significant at p = 0.05” (Caruana), and vice versa as well.

Table 2: Main Matter - test set scores for each learning algorithm by metric

Model	AUC	ACC	FSC	MEAN
SVM	0.670*	0.686*	0.617*	0.658
LR	0.793*	0.736*	0.759*	0.763
KNN	**0.842**	**0.807**	**0.849**	**0.832**

Table 3: Main matter - test set scores for each learning algorithm by dataset

Model	MAGIC	LETTER	CALHOUS	COVTYPE	MEAN
SVM	0.719	0.900*	0.681	0.329	0.657*
LR	0.822	0.756	**0.853**	0.620	0.763*
KNN	0.**837**	**0.963**	0.723	**0.809**	**0.833**

As seen in Table 2, KNN performed the best when evaluated using each of the three metrics AUC, ACC, and FSC. It performed uniformly better than LR and SVM across all three metrics, and by a similar margin each time. The second best-performing model was LR, followed by SVM. This was different from the results obtained in CNM06, where LR was the worst performing model out of the three used in this replication. This may be due to the number of C values that was used as a hyperparameter in both LR and SVM, which was considerably different from those used in the original paper. Although the ridge (regularization) parameter was varied by factors of 10 from 10^−8 to 10^4 for LR, SVM used considerably less C values, which could have definitely had an effect on how it performed across datasets. Also, according to the p-values in *Table 2 Supplemental* (see Appendix), each of these scores is starred, meaning that the difference between the LR and SVM metrics as compared to the best performing value for each column is not significantly different. Therefore, we cannot conclude that KNN didn't perform better purely by chance.

Table 3 shows the test set performances by algorithm/dataset combo averaged across all three metrics. KNN is still the best performing model for the majority of the datasets, except in the case of CALHOUS. For this dataset, LR seemed to outperform both SVM and KNN. However, the average performance of LR was still not as high as the highest performance that KNN showed in different datasets. This could be due to an error in the code or the composition of the CALHOUS dataset.

Something else noteworthy in this table is that although KNN did perform the best out of the three models on the LETTER dataset, the SVM error average, 0.900, was considerably higher than other SVM averages, and was quite close to the KNN value, 0.963. This SVM value also subsequently had a p-value higher than 0.05, although not by much, as seen in *Table 3 Supplemental* (see Appendix). This means that this value was the only one out of the other values to be starred, or to not be statistically significant. The LETTER dataset overall had the highest SVM and KNN values as compared to the other datasets, which might have been a result of the near perfect 50/50 positive and negative class split when being classified.

COVTYPE has an abnormally low average value for SVM compared to the other datasets, especially considering how close the difference between SVM and KNN values was for the LETTER dataset. This shows that it is most likely an issue with the COVTYPE dataset itself, or how it was

classified. The positive and negative classification is more unbalanced than the other datasets, which could be a factor for this result. Also, while the same value reported in CNM06 is fairly low, it is not as low as 0.3. This dataset performed on average worse than MAGIC, LETTER, and CALHOUS, but to a different degree than in CNM06.

Table 4 (see Appendix) shows the mean training performance for the three trials of each algorithm per dataset. Since the model is using the training set to predict training data, some values have 100% accuracy, AUC, or F1. These values are bolded in the table. KNN was the model that achieved the most '1's, or the best accuracy, AUC, and F1 compared to other models. It achieved 100% in both accuracy and AUC, and a value very close to that for F1. This further shows that KNN was the best performing model for these chosen datasets.

4. Discussion

Overall, this mini replication somewhat recreates the results created by Caruana and Niculescu-Mizil, but the results are not as consistent or valid because of differences in hyperparameters and datasets. Out of these three models, KNN seems to have performed the best across three out of four datasets, and had the best metric averages across all three metrics. SVM shows the worst performance for the same three out of four datasets, and across all three metrics. LR performed the best on the LETTER dataset, but not on any other dataset or across any other metrics.

When looking at test scores for each dataset, LETTER on average had the highest overall average score across all three models, most likely due to its almost perfect classification split. On the other hand, COVTYPE had the worst results, maybe due to its uneven distribution of positive and negative classifications. Although these results did not completely verify the results of CNM06, they did so to a point where CNM06 can definitely be further validated. The difference in hyperparameters, calibration methods, performance metrics used, and datasets used would most likely have contributed to the poorer results in this replication, but the results obtained here are still generally similar to those in CNM06.

5. Bonus

In addition to the datasets shown and evaluated here, the ADULT dataset from CNM06 was also previously cleaned and included in the analysis, but proved to be too large to get efficient results from. However, this code is still included in the attached Jupyter Notebook. Also, LETTER was originally split into LETTER PT 1. And LETTER PT. 2, was cleaned accordingly, and was included in the model evaluations, but the performance metrics were extremely low, and affected the averages of the other datasets without providing any valuable information about the three models used.

Acknowledgements

I would like to acknowledge Professor Jason Fleischer and the entire COGS 118A teaching staff for teaching and imparting the knowledge required to understand, start, and complete this project and providing the resources necessary to do so in an efficient manner. Also, a big thank you to my fellow classmates who also struggled the same amount that I did for this project right after struggling on a midterm.

References

1. 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.
2. 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.
3. Scikit-learn: Machine Learning in Python, Pedregosa *et al.*, JMLR 12, pp. 2825-2830, 2011.

Appendix

Table 2 Supplemental - p-values

Model	AUC	ACC	FSC	MEAN
SVM	0.439	0.416	0.342	0.0039
LR	0.576	0.342	0.29	0.0317
KNN	**1**	**1**	**1**	**1**

Table 3 Supplemental - p-values

Model	MAGIC	LETTER	CALHOUS	COVTYPE	MEAN
SVM	1.41068e-06	0.056	3.9199e-07	2.9172e-08	0.245
LR	1.3332e-06	3.0014e-10	**1**	1.1277e-08	0.365
KNN	**1**	**1**	3.5393e-08	**1**	**1**

Table 4: Training Performances by Data Set

Model	AUC	ACC	FSC
SVM	0.5	**1**	**1**
LR	0.796	0.738	0.761
KNN	**1**	**1**	0.994

Table 5: Raw test values

Model/Metric	D	Trial 1	Trial 2	Trial 3	Trial 4	Trial 5
SVM AUC	1	0.739	0.721	0.715	0.693	0.737
SVM ACC		0.648	0.649	0.649	0.652	0.654
SVM FSC		0.785	0.787	0.787	0.788	0.790
SVM AUC	2	0.988	0.987	0.988	0.987	0.988
SVM ACC		0.894	0.852	0.868	0.847	0.872

SVM FSC		0.882	0.828	0.848	0.820	0.854
SVM AUC	3	0.494	0.496	0.496	0.495	0.497
SVM ACC		0.716	0.715	0.713	0.713	0.713
SVM FSC		0.834	0.834	0.832	0.833	0.832
SVM AUC	4	0.443	0.452	0.497	0.497	0.497
SVM ACC		0.512	0.512	0.512	0.512	0.512
SVM FSC		0.0	0.0	7.124e-06	2.84e-05	1.422e-05
LR AUC	1	0.832	0.831	0.823	0.836	0.828
LR ACC		0.787	0.787	0.791	0.790	0.790
LR FSC		0.846	0.846	0.849	0.848	0.849
LR AUC	2	0.812	0.812	0.813	0.813	0.809
LR ACC		0.726	0.727	0.725	0.732	0.72
LR FSC		0.729	0.730	0.728	0.738	0.719
LR AUC	3	0.868	0.866	0.866	0.887	0.869
LR ACC		0.813	0.808	0.809	0.820	0.813
LR FSC		0.876	0.872	0.872	0.885	0.875
LR AUC	4	0.659	0.659	0.666	0.659	0.658
LR ACC		0.612	0.609	0.618	0.613	0.617
LR FSC		0.590	0.586	0.587	0.576	0.594
KNN AUC	1	0.853	0.852	0.850	0.849	0.853
KNN ACC		0.799	0.801	0.801	0.801	0.799
KNN FSC		0.856	0.859	0.860	0.860	0.857
KNN AUC	2	0.989	0.989	0.990	0.992	0.990
KNN ACC		0.946	0.946	0.951	0.949	0.951
KNN FSC		0.946	0.946	0.952	0.949	0.951
KNN AUC	3	0.669	0.659	0.659	0.666	0.662

KNN ACC		0.688	0.693	0.705	0.696	0.684
KNN FSC		0.799	0.805	0.814	0.813	0.809
KNN AUC	4	0.864	0.862	0.861	0.682	0.861
KNN ACC		0.782	0.782	0.783	0.782	0.782
KNN FSC		0.781	0.783	0.786	0.781	0.783