COGS 118A: Supervised Learning Algorithms Comparison

Kevin Pham (A15599562), Steve Kuk (A15521681), Brian Nguyen (A15654767)

Abstract

In replication of Rich Caruana and Alexandru Niculescu-Mizil's paper "An Empirical Comparison of Supervised Learning Algorithms", this paper explores and compares the results of different supervised learning methods: Logistic Regression, KNN, Decision Trees, and Random Forest. By running each algorithm on a training dataset over 15 trials while manipulating different hyperparameters, we were able to determine the most optimal classifier that would produce the highest score among our metrics of accuracy, precision, and F1.

Keywords: Supervised Learning Models, Logistic Regression, k-Nearest Neighbors, Decision Trees, Random Forest, Accuracy, Precision, F1- Score

1. Introduction

This project primarily focuses on the replication of Alexandru Niculescua-Mizil's paper, CNM06. The CNM06 project seeks to compare and contrast between the many various machine learning models and algorithms to evaluate the respective learning models' performance and differences over different datasets. In the CNM06 project (Caruana and Alexandru Niculescu-Mizil), they focused on the comparison between ten supervised learning models, SVMs, neural nets, logistic regression, naive bayes, memory-based learning, random forests, decision trees, bagged trees, boosted trees, and boosted stumps. Furthermore, they used eight different performance metrics divided into three categories: threshold metrics, ordering/rank metrics, and lastly probability metrics. The threshold metrics consisted of accuracy (ACC), F-score (FSC), and lift (LFT). The ordering/rank metrics include area under the ROC curve, average precision (APR), and precision/recall break even point (BEP). The probability metrics were squared error (RMS) and cross-entropy (MXE). In CNM06, this comparison between machine learning models and performance metrics were passed through 11 different datasets. In order to closely replicate CNM06, we chose to do a small-scale replication. We selected similar machine learning models in our project: Logistic Regression, KNN, Decision Trees, and Random Forest. Furthermore to compare the performance and differences between these models, we chose to use the threshold metrics: accuracy, precision, and F1 score. Then we ran them through 4 different datasets: Adult, AI, Magic, and Credit card (see 2.3 Datasets). The algorithms train through 15 different trials with each trial having a unique random 5,000 samples from its respective dataset. With 4 different datasets, 4 different algorithms, and 15 trials each we have a total of 240 trials across our replication. This small-scale CNM06 project would help determine whether the findings in CNM06 would still hold true when performed on a less extensive environment.

2. Methodology

2.1. Learning Algorithms

The four different algorithms we explored consist of: Logistic Regression, KNN, Decision Trees, and Random Forest. In this section we summarize the different parameters used in each of the five different algorithms we utilized. For any parameters not specified, the default parameter was used.

Logistic Regression (LR): We utilized 3 different sets of solvers, 3 sets of penalties, and 9 different C values. The 3 different sets of solvers include: (saga), (lbfgs), and (lbfgs, saga). The 3 different sets of penalties include: (l1, l2), (l2), and (none). Finally, the different C values include: 1.e-04, 1.e-03, 1.e-02, 1.e-01, 1.e+00, 1.e+01, 1.e+02, 1.e+03, and 1.e+04.

K-Nearest Neighbors (kNN): We utilized 25 different values of K ranging from five to 105 and tested four different measurements of distance: euclidean , manhattan, chebyshev, minkowski.

Random Forests (RF): We utilized 6 different max depths ranging from one to six and compared two different criterions: gini and entropy.

Decision Trees (DT): We utilized 6 different max depths ranging from one to six and compared two different criterions: gini and entropy.

2.2 Performance Metric

We used GridSearch in order to determine the most optimal classifier for three performance metrics: accuracy, precision, and f1 score.

Accuracy can be calculated using the equation:

$$Accuracy = \frac{True\ Positive + True\ Negative}{True\ Positive + True\ Negative + False\ Positive + False\ Negative}$$

Precision can be calculated using the equation:

$$Precision = \frac{True\ Positive}{True\ Positive\ + False\ Positive +}$$

F1 Score can be calculated using the equation:

$$F - Score = 2 * \frac{Precision * Recall}{Precision + Recall}$$

NOTE: to calculate Recall in F score refer to:

$$Recall/Sensitivity = \frac{True\ Positive}{True\ Positive + False\ Negative}$$

2.3 Data Sets

The algorithms were trained and tested across four datasets found available from University of California, Irvine's Machine Learning respiratory. The datasets are listed as the following: AI4I2020, ADULT, MAGIC04, DEFAULT CREDIT CARD CLIENTS.

The ADULT dataset already contains binary target labels with some of the labels which are strings. In order to be able to consider these string labels within our modeling, we one-hot encoded every string label column. After our one-hot encoding, we ended up having a total of 98 columns that represent the original 15 features. The classification used within this dataset is a 1 for all those that make less than $50k a year and a 2 for those who make more than $50k a year. The reason for using 1's and 2's instead of negative positive class identifiers is due to the fact that precision does not take in negative values. ADULT returned 24% of the individuals included made more than $50k and the remaining 76% did not make more than $50k.

When working with the AI4I2020 dataset, the data included was already in integer-values therefore, there was no standardization required for this dataset. AI4I2020 did include columns which were additional index values to the ones given already within the dataframe so that attribute was dropped. Another attribute that was dropped was the identification number for each machine. The only string label was the machine type which we ended up performing a one-hot encoding in order to remove all string types within our set. After the required changes, the AI4I2020 dataset had a total 12 attributes. For our classification, a 1 indicated no machine failure and a 2 indicated machine failure. We decided to mark our binary-class with a positive integer value instead of positive and negative class because precision does not take in negative values. AI4I2020 returned a rate of 34% being a 2 which states that 34% of machines resulted in failure and 64% of the data returned a 1 stating that the machine did not fail.

The MAGIC04 dataset contains all integer-value data with the exception of the target label which needed to be replaced. We replaced everything within class 'g' with the number 1 and all appearances of class 'h' within the target column with a 2. The reason we chose 1's and 2's instead of positive and negative class was because of the precision metric. The data required no further manipulation and was trained and tested across its 11 features. MAGIC04 has 65% of its entries belonging within class 'h' and the remaining 35% were in class 'g'.

When we were working with the DEFAULT CREDIT CARD CLIENTS dataset, the data was already within integer-values and no standardization was required. However, we ended up removing one of the columns that had index values that we did not need within our training or testing. After our changes, the dataset had a total of 24 attributes. The target label was already in binary, however, in order to keep consistency with our other datasets we used 1 to mark those that did not have default payments for future months and 2 to mark those that did have default payments set up. DEFAULT CREDIT CARD CLIENTS

returned that 27% of users within the dataset do have payments for the next month paid by default and 73% of users within the dataset did not have payments for the next month paid by default.

Table 1: Description of Problems

PROBLEM	#ATTR	TRAIN SIZE	TEST SIZE	% POZ
ADULT	14	5000	43842	24 %
AI	14	5000	5000	34 %
MAGIC	11	5000	14020	65 %
CREDIT CARD	24	5000	25000	27 %

Table 2: Performance over metrics (Test set)

ALGORITHM	ACC	PREC	F1	MEAN
LOGIT REG	0.138!	0.126!	0.139!	0.134!
KNN	0.871*	0.876*	0.918*	0.888*
RANDOM FOREST	0.881*	0.882*	0.9232*	0.895*
DECISION TREES	0.882*	0.888*	0.9233*	0.898*

NOTE: "*" is used to denote a non-significant difference between the best algorithm and "!" is used to denote the worst algorithm. The best algorithm is highlighted in yellow.

Table 3: Performance over datasets (Test set)

ALGORITHM	ADULT	AI	MAGIC	CREDIT CARD	MEAN
LOGIT REG	0.140!	0.0006!	0.204!	0.188!	0.133!
KNN	0.864*	0.980*	0.864*	0.845*	0.888*
RANDOM FOREST	0.866*	0.999*	0.839*	0.852*	0.889*
DECISION TREES	0.877*	0.9995*	0.879*	0.851*	0.902*

NOTE: "*" is used to denote a non-significant difference between the best algorithm and "!" is used to denote the worst algorithm. The best algorithm is highlighted in yellow.

3. Experiment & Results

3.1 Experiment

First, we select a dataset and machine learning model to use. The dataset is then cleaned for training. For the first trial, we randomly sample 5,000 data points from the dataset. For the later trials to come, each trial will have a unique 5,000 data points randomly sampled from the same dataset.

For each trial, we then train the model on the 5,000 randomly sampled data points using several different corresponding hyperparameters. We run a 5-fold cross validation with our selected threshold metrics (accuracy, precision, and f1 score) through GridSearchCV. GridSearch will then find the best and most optimized parameters and output the metric results. This training process is then repeated 14 more times for a total of 15 trials.

From the training process, we put the results of each trial on a table. We receive a table of the different threshold metrics on the algorithm across all 15 trials over the specified dataset (ie. a table of the accuracy, precision, and f1 scores over the 15 trained trials).

Using the training data, we then take the average of the results of each metric and use it to run it on the testing set.

This whole process is then repeated on every dataset over every algorithm we've selected.

3.2 Results

The results of this experiment is shown in Tables 2, 3, 4, 5, 6, and 7. "*" is used to denote a non-significant difference between the best algorithm and "!" is used to denote the worst algorithm. The best algorithm is highlighted in yellow. Looking at Table 2, we can see that the Decision Trees algorithm had the best performance throughout the different threshold metrics. Followed closely by Random Forest, then KNN, and lastly Logistic Regression (Table 2 shows the mean performance across all datasets, see Table 6 in Appendix for raw scores for each dataset).

Now looking at performance over datasets in the testing set(see Table 3), we can see the consistency of the results from performance over metrics (Table 2). The values in Table 3 is calculated by taking the average of all the metric scores for each algorithm over each dataset. We see that Random Forest does perform better in the Credit Card dataset, but overall, when we average the performance values between all of the datasets, Decision Trees still have the highest performance score. Followed up closely by Random Forest and then KNN. Again, we see that Logistic Regression is our worst performing algorithm and Decision Tree is the best, based on Table 2 and 3. We can further our analysis by looking at Tables 4,5,6, and 7.

In Table 4, similar to Table 3, it shows the performance over datasets in the training set. In this table, we can see that the Decision Trees algorithm performs better on the Adult dataset. In the AI dataset, Decision Trees and Random Forest perform very similarly. Finally in both the Magic and Credit card datasets, we see that the Random Forest algorithm is the best performing algorithm. When the average

performance is taken across the datasets, we find that the Random Forest and Decision Tree algorithms have similar performance averages. The average performances are then followed by KNN and lastly Logistic Regression.

We can also compare the raw scores between the training and testing sets across each dataset (see Table 5 and 6 in Appendix). When we compare the means between training set and testing set raw scores, we see that the average of the testing set is generally higher than the training set. Looking at the raw score tables, it shows that Logistic Regression is still the worst performing algorithm and that the Decision Tree algorithm generally has the best performance scores out of our algorithms.

Finally, we can compare the T-scores and P-values across algorithms on the training set (see Table 7), we are picking to compare Decision Trees with the rest of the algorithms. Setting our p-value statistic significance at ($p = 0.01$), most of our algorithm scores reject the null hypothesis and only Random Forest fails to reject the null hypothesis.

4. Conclusion (Discussion)

In CNM06, overall, the algorithm that had the most excellent performance through all eight of their metrics was the calibrated Boosted Trees. Random Forests ranked a close second place and then uncalibrated Bagged Trees, calibrated SVMs, and then uncalibrated Neural Nets. The algorithms that performed the worst were Naive Bayes, Logistic Regression, Decision Trees, and Boosted Stumps. In general, the results of our less extensive replication of CNM06 only somewhat replicate the findings in CNM06. From our experiment, our results showed that the best performing algorithms were Decision Trees and Random Forests. Our worst performing algorithms were Logistic Regression and KNN. Based on our results and CNM06's findings, we can only confirm that Random Forests is one of the better performing algorithms and that the Logistic Regression algorithm is one of the poor-performing algorithms.

In "Supervised Machine Learning Algorithms: Classification and Comparison" by Osisanwo, Akinsolo, Awodele, Hinimikaiye, Olakanmi, and Akinjobi (IJCTT, see References), they did a similar experiment comparing supervised machine learning algorithms in both small and large sample sizes. The algorithms they compared include: SVM, Random Forest, Naive Bayes, Decision Tree, Neural Networks, and more. They found that Decision Tree and Decision Table algorithms had the least precision score with larger datasets, SVM and Random Forest algorithms showed high accuracy and precision with smaller datasets. The paper explained how in general, the performance of a machine learning model depends on the given conditions such as the size of the dataset, parameters, etc. Though, comparing our experiment results with IJCTT, we see that not much of our findings align with the findings in IJCTT.

Finally in "An Empirical Comparison of Supervised Ensemble Learning Approaches" by Mohamed Bibimoune, Haytham Elghazel, and Alex Aussem (ICML06, see References), they compared different ensemble models in Machine Learning with and without calibration such as: Boosting, Bagging, Random Forests, Decision Trees, and more similar variants. They found that there are many factors that influence the performance of these algorithms (such as dataset sizes), but concluded that calibrating algorithms is a very effective way of improving performance of boosted-based algorithms. Furthermore, from the experiments, they concluded that the Rotation Forest family of algorithms (such as Random Forest), outperformed many of the other ensemble methods by a large margin. The worst performing models were Decision Trees, Bagged Trees, and AdaBoost Stump. Comparing our and ICML06's results, we can confirm that Random Trees were one of the best performing algorithms. Additionally, ICML06's

results also closely align with the findings of CNM06, confirming that Random Trees serves as one of the best performing algorithms and that Decision Trees were one of the worst performing.

In our trial to recreate the research done within CNM06, we found throughout our datasets and models that the best performing algorithms were Random Forests and Decision Trees. As a matter of fact, our findings went against the findings of CNM06. If we refer to Table 7, we can observe that our p-values and t-scores have more abnormal results than what we would wish to expect. While we did expect for our p-values to be small numbers that would reject the null, our values turned out to be a lot smaller with only two values actually failing to reject the null hypothesis. Contrary to this, our t-scores were actually very large in number ranging from -4 to 48. We believe that this vast range and abnormally small p-value is due to using "1's' and "2's" to label our classes instead of using "1" and "-1" to signify a positive and negative class. In addition to our labeling, we believe that we could have made errors within our implementation of the algorithms across our datasets. Through the duration of our project, we discovered that supervised machine learning applications can be very useful in predicting and observing trends within datasets. Although these systems may not always be accurate, we believe that machine learning has a large role within our day-to-day lives as well as opening new opportunities for the future. To further our implementations of this project, we could use additional datasets, algorithms, and trials to provide a more thorough analysis of our findings and have more reliable conclusions.

Appendix

Table 4: Performance over datasets (Training set)

ALGORITHM	ADULT	AI	MAGIC	CREDIT CARD	MEAN
LOGIT REG	0.140	0.0009	0.204	0.184	0.133
KNN	0.848	0.980	0.862	0.846	0.884
RANDOM FOREST	0.871	0.999	0.873	0.862	0.901
DECISION TREES	0.8795	0.999	0.872	0.851	0.901

Table 5: Raw scores (Training set)

ALGORITHM	METRIC	ADULT	AI	MAGIC	CREDIT CARD	MEAN
LOGIT REG	ACC	0.1499	0.0009	0.20	0.189	0.13495

“	**PREC**	**0.120**	**0.0009**	**0.198**	**0.176**	**0.123725**
“	**F1**	**0.1499**	**0.0009**	**0.209**	**0.189**	**0.1372**
KNN	**ACC**	**0.824**	**0.975**	**0.852**	**0.815**	**0.8665**
“	**PREC**	**0.829**	**0.977**	**0.840**	**0.834**	**0.87**
“	**F1**	**0.893**	**0.987**	**0.894**	**0.889**	**0.91575**
RANDOM FOREST	**ACC**	**0.851**	**0.999**	**0.867**	**0.836**	**0.88625**
“	**PREC**	**0.908**	**0.999**	**0.848**	**0.847**	**0.9005**
“	**F1**	**0.854**	**0.9995**	**0.905**	**0.902**	**0.915125**
DECISION TREES	**ACC**	**0.851**	**0.999**	**0.862**	**0.822**	**0.8835**
“	**PREC**	**0.908**	**0.9995**	**0.901**	**0.838**	**0.911625**
“	**F1**	**0.854**	**0.999**	**0.853**	**0.894**	**0.9**

Table 6: Raw scores (Testing set)

ALGORITHM	**METRIC**	**ADULT**	**AI**	**MAGIC**	**CREDIT CARD**	**MEAN**
LOGIT REG	**ACC**	**0.1503**	**0.0006**	**0.2112**	**0.1904**	**0.1381**
“	**PREC**	**0.1205**	**0.000621**	**0.2002**	**0.1823**	**0.1259**
“	**F1**	**0.1503**	**0.0006**	**0.2112**	**0.1904**	**0.1381**
KNN	**ACC**	**0.8395**	**0.976**	**0.8539**	**0.8147**	**0.871**
“	**PREC**	**0.8512**	**0.9768**	**0.8434**	**0.8323**	**0.8759**
“	**F1**	**0.9002**	**0.9876**	**0.8935**	**0.8887**	**0.9175**
RANDOM FOREST	**ACC**	**0.8435**	**0.999**	**0.8566**	**0.8250**	**0.881**
“	**PREC**	**0.8511**	**0.9993**	**0.8392**	**0.8365**	**0.8815**
“	**F1**	**0.9035**	**0.9996**	**0.8955**	**0.8940**	**0.9231**

DECISION TREES	**ACC**	**0.8547**	**0.9994**	**0.8516**	**0.8211**	**0.8817**
“	**PREC**	**0.8666**	**0.9993**	**0.8377**	**0.8412**	**0.8862**
“	**F1**	**0.9089**	**0.9996**	**0.8927**	**0.8918**	**0.9232**

Table 7: T-scores & P-values across Algorithms (Training set): Decision Tree compared with rest

ALGORITHM	T-SCORE	P-VALUE
LOGIT REG ACCURACY	**39.165**	**6.281e-44**
LOGIT REG F1	**48.798**	**2.142e-49**
LOGIT REG PRECISION	**42.175**	**9.228e-46**
KNN ACCURACY	**11.932**	**2.284e-17**
KNN F1	**13.263**	**2.363e-19**
KNN PRECISION	**8.902**	**1.662e-12**
RANDOM FOREST ACCURACY	**-2.593**	**0.012**
RANDOM FOREST F1	**-4.166**	**0.0001**
RANDOM FOREST PRECISION	**1.836**	**0.071**

References

M. Bibimoune, H. Elghazel, A. Aussem. (2013). “An Empirical Comparison of Supervised Ensemble Learning Approaches.”

Osisanwo, F. Y., Akinsola, J. E. T., Awodele, O., Hinmikaiye, J. O., Olakanmi, O., & Akinjobi, J. (2017). Supervised machine learning algorithms: classification and comparison. *International Journal of Computer Trends and Technology (IJCTT)*, *48*(3), 128-138.

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