# A Comparison of Supervised Learning Algorithms

Christina Yu
Geon Yoo
UC San Diego

## Abstract

This paper loosely replicates Caruana and Niculescu-Mizil's 2006 paper and compares the performance of 6 supervised machine learning algorithms across 8 different data sets. The algorithms that were used include support vector machines, multi-level perceptrons, decision trees, logistic regression, k-nearest neighbors, and random forest. By using GridSearchCV and inputting different hyper-parameters, we were able to find the optimal fit across each dataset. Using accuracy, receiver operating characteristic curve, and f1 score for our scoring, we find the most accurate results averaged across five trials per algorithm, per dataset. The most important aspect of our study is the use of different algorithms and creating criteria to compare learning methods. Our results show similar results to Caruana and Niculescu-Mizil's 2006 paper where decision trees were only good in some instances, when compared to other algorithms which performed consistently better.

## 1. Introduction

Adding to STATLOG(King et al., 1995), a famous comparison between supervised learning models, Caruana and Niculescu-Mizil, referred to as CNM06, adds onto the list of modern algorithms that were not present during the time of the STATLOG trails like support vector machines, boosted trees, and random forests.

In addition, CNM06 finds empirical comparison between ten supervised learning algorithms using eight performance criteria. They then evaluated their performance on eleven binary classifications as well as a variety of performance metrics like accuracy, F-score, lift, ROC area, and etc. For our study however, we compare 6 supervised learning algorithms using 3 performance criteria on 8 binary classification problems.

CNM06 reported that when calibrated boosted trees and random forests perform best while logistic regression, naive bayes, ANN, and decision trees perform the worst. When adapting their workfload they warn that since they are averages sometimes even the best algorithms performed worse than others and vice versa, stressing that it is the average.

The main purpose of this paper is to approach the comparison between supervised models similar to CNM06 and attempt to replicate the results in order to verify their study. We do so on our own selection of eight datasets, six different algorithms, and threshold measure of accuracy, ROC, and f1. There is some overlap

between CNM06's selection and ours, like Calhous, adult, letter. We then compare our results to CNM06 to see the differences and similarities of the datasets and their attributes and how well the classifiers perform on them.

## 2. Methodology

### 2.1 Learning Algorithms

For the algorithms selected, the hyperparameter selections closely resemble the ones chosen in CNM06, with some small differences.

**Random Forests (RF):** RF with 1024 trees was used. The size of the feature set was considered at each split, [1, 2, 4, 6, 8, 12, 16, 20]. The exception being for the datasets that had features less than 20, in which the max features selected corresponded to the amount of features the dataset had.
**Logistic Regression (LogReg):** LogReg was used with L1, L2, and no regularization, with ridge (regularization) parameter varying by factors of $10^{-8}$ to $10^{4}$.
**KNN:** KNN was used with odd values of K ranging from K = 1 to K = 101. KNN was also used with Euclidean distance and was distance weighted.
**Decision Trees(DT):** The parameters that were varied are the splitting criterion, pruning options, and smoothing, which aligns with CNM06's parameters.
**Perceptron (ANN):** We trained neural nets with gradient descent and varied the hidden_layer_sizes: [1,2,4,8,32,128] and momentum: [0,0.2,0.5,0.9]
**SVM:** We used the following kernels: [linear, rbf, and poly] with ridge (regularization) parameters varying by factors of 10 from $10^{-7}$ to $10^{3}$ for each kernel.

### 2.2 Performance Metrics

The performance metrics that were used in this comparison are accuracy, ROC, and F1. Accuracy was chosen as the main method for comparing the different algorithms used in this study. However, due to the imbalanced datasets used as a part of this study, accuracy has the potential to not capture the true performance of the algorithm. There is a possibility that there would be a high accuracy score, just solely due to the nature of the imbalanced datasets. For that reasoning, ROC and F1 are used to supplement the accuracy scores. ROC is a probability curve that illustrates the ability of a binary classifier algorithm as its threshold is varied. ROC has "baseline rates that are independent of the data" (Caruana, 2006) which provides a good baseline to measure. Lastly, F1 is used as a metric to combine both accuracy scores and ROC scores. F1 is a measure of a test's accuracy and is calculated from precision and recall. It is the "harmonic mean of the precision and recall" (Shung, 2020) and is in the range of [0,1] like accuracy. F1 is better to use than precision and recall for uneven class distribution, which is the bulk of our datasets.

Accuracy formula:

$$ACC = (True\ Positive + True\ Negative) / (TP + TN + FP + FN)$$

ROC formula: Area under the curve

F1 formula: $F1 = 2 * (Precision * Recall) / (Precision + Recall)$

### 2.3 Calibration Methods

Calibration was not used in this experiment due to the computational load and time constraints of the project. Since some classifiers are not designed to predict probability, calibration was used in the original CNM06 study, however, for this study calibration was not used due to the above mentioned reasons, which theoretically would produce lower scores for certain classifiers, such as Decision Trees.

### 2.4 Datasets

The algorithms were compared on 8 different binary classification problems. Adult, Letter, and Skin datasets were all from the UCI repository (Blake & Merz, 1998). Adult contains nominal attributes and was one-hot encoded for classification. The classification target for this dataset was salary, either >50k or <= 50k.Letter was separated into 2 dataset problems by converting to boolean in 2 ways. For letter.p1, the letter O is treated as positive and the remaining 25 letters are treated as negative, which yields a very imbalanced problem. For letter.p2, letters A-M are treated as positive, and letters N-Z are treated as negative, which yields a balanced dataset. For the Skin dataset, whether the classification was skin or not was changed from values of 1 and 2 to values of 0 and 1, so that the classifiers did not give unnecessary weight to one or the other. Dry bean, calhous, seoul bike, and Traffic_V were also all from the UCI repository (Blake & Merz, 1998). All four datasets included nominal data. For seoul bike, the classification of the seasons were changed from a value 0 to 3 depending on the season they were. For calhous, the proximity of the ocean was classified from a value between 0 to 10 depending on the closeness of the ocean. One classification was excluded because it only appeared four times and keeping it would result in our k-fold classification to not work. Traffic_V was also classified to where the weather type from a value 0 to 5 depending on where they were classified to. Lastly dry bean's class was classified to a nominal value to run our test on.

| Dataset | #ATTR | Train size | Test size | %poz |
|---|---|---|---|---|
| Adult | 14/104 | 5000 | 27561 | 25% |
| Letter.p1 | 16 | 5000 | 15000 | 3% |
| Letter.p2 | 16 | 5000 | 15000 | 53% |

| | | | | |
|---|---|---|---|---|
| Skin | 4 | 5000 | 245057 | 26% |
| Dry Bean | 16 | 10889 | 2723 | 80% |
| Calhous | 9 | 16512 | 4129 | 80% |
| Seoul Bike | 10 | 7008 | 1753 | 80% |
| Traffic_V | 2 | 38560 | 9641 | 80% |

*Table 1: description of datasets*

### 3. Performances by Metric

For each dataset, we randomly select 5000 data points for training and use the rest of the points as a test set. 5-fold cross validation is used on the 5000 training points for 5 trials used with the varying hyperparameters. Stratified cross validation is used on the datasets that have very imbalanced data, in order to get a more unbiased result. Some of the datasets required more than 5000 data points, so we shifted over to using a 80-20 split to train and test our data.

In table 2, we have the score for each algorithm of each of the metrics that we have chosen. For each dataset, we find the best parameter settings for each algorithm chosen and then report the classifier's score on the final test set. Since none of the models were calibrated, we do not have a calibration column like in CNM06. The algorithm with the best performance on each metric is in bold font and a * is used to denote a non-significant difference between the best algorithm and the others.

From the final test set metrics, we see that RF has the highest metrics score all around. However, KNN and SVM were compared to RF and were found to have a non-significant difference from RF with KNN having a p-value of 0.0002 and SVM having lower than a 0.05 p-value as well. However, oddly enough the F1 score for KNN was considerably different in p-value than RF, while LogReg's p-value was close for F1. ANN and LogReg clearly underperformed compared to the others consistently. It is expected that RF would perform generally better, as RF is the "combined results of many decision trees" (Wickramanayake, 2020). RF, however, is expected to be better than DT because of the random nature of the model. By randomly selecting features, some trees in the forest can isolate the more important features and increase the overall accuracy of the results. KNN, on the other hand, most likely has better results than LogReg on average because it is a robust classifier with regard to the search space, and the classes do not have to be linearly separable. The cost exchange is a slow run time. SVM is also a better classifier, because it tries to find the best distance between the line and the support vectors that separates the classes, which reduces the risks of errors on the data.

| Model | ACC | ROC | F1 | Mean |
|---|---|---|---|---|
| RF | **0.940** | **0.965** | **0.941** | **0.949** |
| LOGREG | 0.850 | 0.824 | 0.847* | 0.840 |
| KNN | 0.922* | 0.952* | 0.922 | 0.932 |
| DT | **0.932** | 0.652 | **0.921** | 0.835 |
| ANN | 0.832 | 0.812 | 0.845 | 0.830 |
| SVM | 0.911* | **0.932** | 0.921* | **0.921** |

*Table 2: Test scores for each learning algorithm by metric*

**4. Performances by Problem**

Table 3 is a comparison of classifiers based on the dataset. Each entry in the table is the average over the performance metrics and the 5 trials. The best classifier is denoted with bold text, and the algorithms that are not significantly different from the best classifier are denoted with a *.

RF and KNN consistently got very high train scores for all the datasets. This could be due to overfitting to the training data. However, the testing scores were also quite high, so it appears that in general RF and KNN did well. Though since the testing scores for RF and KNN are still lower, there may be some overfitting in the training scores. It could also be due to the imbalanced nature of the datasets that there is overfitting occurring, pushing the metric values higher. LogReg, on the other hand, had varying training and testing scores. It appears LogReg had a much more difficult time classifying the data, both in training and testing scores. The scores for LogReg, DT, ANN, and SVM are a great deal more varying in scores than RF and KNN, though DT's highest score reaches just below RF and KNN's scores. A most likely reasoning for the Letter.p2 dataset having a lower scoring than the others, is that I did not use stratified K-Fold on that dataset like I did for the others. The dataset was very balanced on average, but since I did not stratify the K-fold, it most likely did not sample equally from positive and negative classes. The reason I did not use stratified K-fold for letter.p2 is because of time constraints, it was noticed that stratified K-fold was not used for this dataset, but there was no time to rerun the classifier.

| Model | Adult | Ltr.p1 | Ltr.p2 | Skin | Mean |
|---|---|---|---|---|---|
| RF | **1.** | **1.** | **1.** | **1.** | **1.** |
| LOGREG | 0.838 | 0.958 | 0.712 | 0.892 | 0.850 |

| | | | | | |
|---|---|---|---|---|---|
| KNN | 1.* | 1.* | 1.* | 1.* | 1.* |

*Table 3.1: Train scores of each learning algorithm by dataset*

| Model | Dry Bean | Seoul Bike | Traffic_V | Calhous | Mean |
|---|---|---|---|---|---|
| DT | 0.91 | 0.79 | 0.28 | **0.97** | **0.738** |
| ANN | **0.94** | **0.81** | 0.24 | 0.92 | 0.728 |
| SVM | 0.927* | 0.729* | **0.305** | 0.802 | 0.691 |

*Table 3.2: Train scores of each learning algorithm by dataset*

## 5. Conclusions

The results of this study replicate CNM06 in some aspects with some differences in other aspects, though overall supports CNM06's findings. Like in CNM06, Random Forest performed consistently the best across all datasets, which proves the robustness of the classifier compared to other algorithms. SVM also performed consistently well on test data sets like in CNM06, however, on training datasets it was much more volatile. This could point to the classifier "guessing" due to the dataset being heavily imbalanced and getting the classes correct. On the other hand, KNN performed extremely well in our findings compared to CNM06. CNM06's KNN classifier had much more volatile values, while our values remained consistently high, only slightly below Random Forest, and determined to have a non-significant difference in comparison using a t-test. Also unlike CNM06, our SVM and Decision Trees were slightly more volatile. Though, on some of the datasets the scores reached similar values to CNM06. This may be due to the datasets themselves being imbalanced, not suited to the classifiers, or improper data cleaning. This is evidenced by the Traffic_V dataset having overall very poor scoring on all the algorithms tested.

Looking at the scores for all of the datasets, Traffic_V had by far the lowest scores, while the rest of the datasets had generally high scores in the 80s or 90s in terms of percentage accuracy. Despite the imbalanced datasets, most of the algorithms did generally well in classifying, matching CNM06's results for the most part. And also despite the odd results for some datasets, overall our classifiers support CMN06. Like CNM06, Random Forest did the best, while the ones that did the poorest was Logistic Regression and ANN. Though due to the volatility of some of the classifiers, like with SVM and Decision Trees on training data, it is still not quite known how well they match up with CNM06's results.

## 6. References

King, R., Feng, C., & Shutherland, A. (1995). Statlog: comparison of classification algorithms on large real world problems. Applied Artificial Intelligence, 9.

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.

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

Shung, K. (2020, April 10). Accuracy, precision, recall or f1? Retrieved from https://towardsdatascience.com/accuracy-precision-recall-or-f1-331fb37c5cb9

Wickramanayake, B. (2020, April 17). Random forest vs logistic regression. Retrieved from https://medium.com/@bemali_61284/random-forest-vs-logistic-regression-16c0c8e2484c.

## 7. Appendix

“A” represents the Adult dataset, “L1” represents Letter.p1, “L2” represents Letter.p2, “SK” represents the Skin dataset. “DB” represents the Dry Bean dataset. “CAL” represents the CalHous dataset. “SB” represents the Seoul bike dataset. “TV” represents the traffic_v dataset.

| Model | 1A | 2A | 3A | 4A | 5A | 1L1 | 2L1 | 3L1 | 4L1 | 5L1 |
|---|---|---|---|---|---|---|---|---|---|---|
| RF | 0.849 | 0.852 | 0.871 | 0.838 | 0.844 | 0.990 | 0.990 | 0.992 | 0.990 | 0.985 |
| LOG | 0.819 | 0.833 | 0.844 | 0.816 | 0.827 | 0.958 | 0.958 | 0.958 | 0.958 | 0.957 |
| KNN | 0.820 | 0.820 | 0.835 | 0.829 | 0.823 | 0.984 | 0.982 | 0.958 | 0.981 | 0.983 |

*Table 4.1: Raw Testing Performance by Trial*

| Model | 1L2 | 2L2 | 3L2 | 4L2 | 5L2 | 1SK | 2SK | 3SK | 4SK | 5SK |
|---|---|---|---|---|---|---|---|---|---|---|
| RF | 0.940 | 0.936 | 0.935 | 0.932 | 0.943 | 0.995 | 0.997 | 0.997 | 0.997 | 0.995 |
| LOG | 0.72 | 0.696 | 0.723 | 0.711 | 0.700 | 0.891 | 0.889 | 0.891 | 0.902 | 0.888 |
| KNN | 0.885 | 0.884 | 0.887 | 0.889 | 0.893 | 0.990 | 0.990 | 0.988 | 0.993 | 0.990 |

*Table 4.2: Raw Testing Performance by Trial*

| Model | 1DB | 2DB | 3DB | 4DB | 5DB | 1CAL | 2CAL | 3CAL | 4CAL | 5CAL |
|---|---|---|---|---|---|---|---|---|---|---|
| SVM | 0.925 | 0.929 | 0.934 | 0.928 | 0.920 | 0.780 | 0.806 | 0.789 | 0.834 | 0.782 |
| MLP | 0.94 | 0.93 | 0.95 | 0.93 | 0.93 | 0.81 | 0.80 | 0.81 | 0.81 | 0.81 |
| DT | 0.91 | 0.90 | 0.91 | 0.90 | 0.91 | 0.80 | 0.79 | 0.77 | 0.81 | 0.78 |

*Table 4.3: Raw Testing Performance by Trial*

| Model | 1SB | 2SB | 3SB | 4SB | 5SB | 1TV | 2TV | 3TV | 4TV | 5TV |
|---|---|---|---|---|---|---|---|---|---|---|
| SVM | 0.727 | 0.73 | 0.725 | 0.723 | 0.731 | 0.327 | 0.331 | 0.330 | 0.260 | 0.278 |
| MLP | 0.93 | 0.92 | 0.92 | 0.92 | 0.91 | 0.25 | 0.27 | 0.24 | 0.21 | 0.23 |
| DT | 0.97 | 0.96 | 0.98 | 0.96 | 0.98 | 0.28 | 0.28 | 0.25 | 0.27 | 0.27 |

*Table 4.4: Raw Testing Performance by Trial*

| Model | 1A | 2A | 3A | 4A | 5A | 1L1 | 2L1 | 3L1 | 4L1 | 5L1 |
|---|---|---|---|---|---|---|---|---|---|---|
| RF | 1. | 1. | 1. | 1. | 1. | 1. | 1. | 1. | 1. | 1. |
| LOG | 0.840 | 0.839 | 0.833 | 0.838 | 0.839 | 0.958 | 0.957 | 0.958 | 0.958 | 0.958 |
| KNN | 1. | 1. | 1. | 1. | 1. | 1. | 1. | 1. | 1. | 1. |

*Table 5.1: Raw Training Performance by Trial*

| Model | 1L2 | 2L2 | 3L2 | 4L2 | 5L2 | 1SK | 2SK | 3SK | 4SK | 5SK |
|---|---|---|---|---|---|---|---|---|---|---|
| RF | 1. | 1. | 1. | 1. | 1. | 1. | 1. | 1. | 1. | 1. |
| LOG | 0.710 | 0.716 | 0.708 | 0.710 | 0.715 | 0.894 | 0.890 | 0.892 | 0.892 | 0.894 |
| KNN | 1. | 1. | 1. | 1. | 1. | 1. | 1. | 1. | 1. | 1. |

*Table 5.2: Raw Training Performance by Trial*

| Model | 1DB | 2DB | 3DB | 4DB | 5DB | 1CAL | 2CAL | 3CAL | 4CAL | 5CAL |
|---|---|---|---|---|---|---|---|---|---|---|
| SVM | 0.925 | 0.929 | 0.934 | 0.928 | 0.920 | 0.780 | 0.806 | 0.789 | 0.834 | 0.782 |
| MLP | 0.942 | 0.932 | 0.952 | 0.933 | 0.932 | 0.812 | 0.802 | 0.813 | 0.811 | 0.812 |
| DT | 0.913 | 0.904 | 0.912 | 0.901 | 0.913 | 0.803 | 0.791 | 0.772 | 0.813 | 0.782 |

*Table 5.3: Raw Training Performance by Trial*

| Model | 1SB | 2SB | 3SB | 4SB | 5SB | 1TV | 2TV | 3TV | 4TV | 5TV |
|---|---|---|---|---|---|---|---|---|---|---|
| SVM | 0.727 | 0.733 | 0.725 | 0.723 | 0.731 | 0.327 | 0.331 | 0.330 | 0.260 | 0.278 |
| MLP | 0.931 | 0.922 | 0.923 | 0.922 | 0.913 | 0.253 | 0.279 | 0.247 | 0.214 | 0.236 |
| DT | 0.972 | 0.961 | 0.982 | 0.961 | 0.982 | 0.281 | 0.286 | 0.257 | 0.276 | 0.274 |

*Table 5.4: Raw Training Performance by Trial*

| | P-value - LogReg | P-value - KNN |
|---|---|---|
| RF - ACC | 0.208 | 0.0002 |
| RF - ROC | 0.210 | 0.015 |
| RF - F1 | 0.023 | 3.442 |

*Table 6.1: p-values for table 2 (RF vs the rest)*

| | P-value - ANN | P-value - SVM |
|---|---|---|
| DT - ACC | 3.214 | 0.025 |
| DT - ROC | 5.264 | 0.221 |

| | | |
|---|---|---|
| DT - F1 | 3.61 | 0.006 |

*Table 6.2: p-values for table 2 (RF vs the rest)*

| Dataset/Model | RF |
|---|---|
| ADULT (LOGREG) | 1.42 |
| ADULT (KNN) | ND |
| LETTER.P1 (LOGREG) | 1.92 |
| LETTER.P1 (KNN) | ND |
| LETTER.P2 (LOGREG) | 6.092 |
| LETTER.P2 (KNN) | ND |
| SKIN (LOGREG) | 2.226 |
| SKIN (KNN) | ND |

*Table 7.1: p-values for table 3 (RF vs the rest)*
*(ND stands for no difference, as the values were both 1.)*

| Dataset/Model | DT |
|---|---|
| DB(ANN) | 2.42 |
| DB(SVM) | 3.232 |
| SB(ANN) | 1.42 |
| SB(SVM) | 5.723 |
| CAL(ANN) | 4.123 |
| CAL(SVM) | 2.12 |
| TV(ANN) | 2.34 |
| TV(SVM) | 6.213 |

*Table 7.2: p-values for table 3 (RF vs the rest)*