**A Comparison of Supervised Machine Learning Algorithms**

**Ahmad Sadeed asadeed@ucsd.edu**

Abstract

The goal of this paper is to compare six supervised machine learning algorithms on binary classification tasks. Four balanced and imbalanced datasets are used to evaluate logistic regression, decision trees, random forests, gradient boosting, neural networks, and naïve bayes algorithms using accuracy, average precision and receiver operating characteristic scores. Each classifier was tuned to wide ranges of values and hyperparameters during five trials and 5-fold cross validation using grid search. The results match experiments performed by Caruana and Niculescu-Mizil on a broader set of algorithms, problems, and performance metrics. Alternatives for null hypothesis statistical testing is proposed.

**1. Introduction**

As the No Free Lunch Theorem states, there is no one classifier that performs best on all datasets and on all metrics (Caruana and Niculescu-Mizil, 2006). Caruana and Niculescu-Mizil (referred to as CNM06 in this paper) experimented to test if there are classifier that generally do well on most of the problem sets or that tend to do poorly across most of the problems. Following CNM06, this paper attempts to replicate their experiment using logistic regression, decision trees, random forests, gradient boosting, neural networks, and naïve bayes classifiers on four problems. The metrics used include accuracy (ACC), average precision (APR) and receiver operating characteristic (ROC) score. Different hyperparameters were explored for each algorithm. This paper used null hypothesis statistical testing (NHST) with p = 0.05 to compare classifiers. In the discussion section, the shortcomings of NHST and alternative approaches are discussed.

**2. Methodology**

**2.1 Algorithms**

The following classifiers from Scikit-Learn library (Scikit-Learn, 2019) were used during this experiment:

**Logistic Regression (LOGIT):** The “liblinear” solver of Scikit-Learn implementation of logistic regression was used. This solver requires fewer iterations to converge and needs the “penalty” parameter to be set. The default “L2” penalty was used. The parameter for inverse of regularization strength or “C” was validated on values from to .

**Decision Tree (TREES):** The quality of split was measured using “gini” for impurity and “entropy” for information gain. The “max\_depth” (maximum depth of the tree) parameter was cross validated on values in [1, 4, 7, 10, 13, 16, 19, 22, 25]. The “min\_samples\_split” (minimum number of samples required to split an internal node) was tested on values in [1, 3, 5, 7, 9]. The Minimal Cost-Complexity Pruning parameter “ccp\_alpha” gave best result with the default value of 0.0 and was not used in second run of the process. The “min\_samples\_leaf” parameter (minimum number of samples required to be at a leaf node) which has the effect of smoothing the model was tested on values in [1,3,5,7].

**Random Forest (FOREST):** Number of trees in the forest or “n\_estimators” was tested on values in [2, 4, 8, 16, 32, 64, 128, 256, 512, 1024]. Number of features to consider or “max\_features” was evaluated on values in [1,2,4,6,8,12,16,20].

**Gradient Boosting (GBOOST):** Number of boosting steps (n\_estimators) was tested on values in [2,4,8,16,32,64,128,256,512,1024,2048]. In addition, the default “max\_depth” of 3 was replaced with values in [1, 3, 5, 7, 9] for further cross validation.

**Multi-Layer Perceptron (MLP\_ADAM):** The “hidden\_layer\_sizes” or the number of neurons in the only hidden layer was evaluated with values in [1,2,4,8,32,128,256]. Scikit-Learn uses Adam optimizer as default, which is a stochastic gradient-based optimizer and is usually faster and gives better results than “SGD” optimizer.

**Naive Bayes (NB):** The additive Laplacian smoothing parameter or “alpha” ranged in values from to .

The datasets were scaled to a mean of 0 and standard deviation of 1 for the LOGIT and MLP\_ADAM classifiers. The remaining classifiers were trained on non-scaled datasets similar to CNM06.

**2.2 Performance Metrics**

Accuracy score (ACC), average precision score (APR), and receiver operating characteristic score (ROC AUC) were used to compare performances of classifiers. ACC belongs to the category of threshold metrics. APR and ROC belong to ordering/rank metrics. In addition to ACC, it is crucial that we evaluate classifiers using other metrics to account for shortcomings of ACC as it is possible to achieve high accuracy rate in cases with class imbalance by always choosing the majority class. Average precision (Scikit-Learn) provides the weighted mean of precisions at each threshold. and are the precision and recall at nth threshold and the increase in recall from previous threshold ( is used as the weight:

Similar to ACC, APR is between 0 and 1 where a higher score is preferred. Receiver operating characteristic (ROC) score computes a single value from the ROC curve and shows the performance of a binary classifier as its discrimination threshold is changed (Wikipedia, 2019). ROC curve incorporates a model’s true positive rate (TPR), also known as sensitivity or recall and false positive rate (FPR), also known false alarm and [1 – specificity]. ROC AUC score can assist in identifying optimal and suboptimal classifiers independent from class distribution.

**2.3 Data Sets**

As shown in Table 1, the six classifiers were evaluated on four data sets from UCI repository: LETTER, ADULT, and COVTYPE. LETTER was converted to create two different datasets. LETTER1 (which is LETTER.P2 in CNM06) was generated by converting letter A-M as positive and the rest as negative class. LETTER2 (LETTER.P1 in CNM06) was generated by treating letter “O” as positive and the remaining letters as negative. LETTER1 is well balanced with 50% positive samples. LETTER2 is an imbalanced set with only 4% positive samples. ADULT was turned into a classification problem by mapping INCOME column to 0 for INCOME <= 50K and 1 for INCOME > 50K, leading to 25% positive samples. For COVTYPE, the largest class for the COV column was mapped to 1 and the rest to 0. COVTYPE had a sample size of over 550K and a smaller subset of dataset was used in this experiment. Categorical variables in ADULT were one-hot-encoded leading to 104 attributes.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *Table 1*: Description of Datasets | | | | |
| DATASET | #ATTR | TRAIN SIZE | TEST SIZE | %POZ |
| ADULT | 14/104 | 5000 | 40222 | 25% |
| LETTER1 | 16 | 5000 | 15000 | 50% |
| LETTER2 | 16 | 5000 | 15000 | 4% |
| COVTYPE | 54 | 5000 | 24051 | 49% |

**3. Experiments and Results**

Each classifier was evaluated on each dataset for five trials. In each of the five trials, random 5000 samples were selected from the data for the training set and the remaining large portion of the data was assigned to the test set. Within each trial, the 5000 samples were split into five folds to run cross validation using grid search method to find the best hyperparameter settings. Through using the “multiple metric evaluation” setup within Scikit-Learn GridSearchCV, separate best hyperparameter settings for ACC, APR, and ROC AUC were stored and then utilized to train the models on the 5000 samples as a whole. Table 4 in the appendix section contains the mean training set performance on each metric averaged over four datasets.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *Table 2*: Test Set Performance (over four datasets) | | | | |
| MODEL | ACC | APR | ROC | MEAN |
| LOGIT | 0.822754 | 0.623291 | 0.850184 | 0.76540967 |
| TREES | 0.867867\* | 0.779514 | 0.893362 | 0.84691433 |
| FOREST | 0.900355\* | 0.897413\* | 0.946725\* | 0.914831\* |
| GBOOST | **0.907358** | **0.906995** | **0.950503** | **0.92161867\*** |
| MLP\_ADAM | 0.893842\* | 0.881304\* | 0.938188\* | 0.90444467\* |
| NB | 0.749619 | 0.502576 | 0.703253 | 0.651816 |

The best hyperparameter setting for each metric was then used to test classifiers on the test set. Table 2 contains the test set performance of classifiers averaged over the four datasets. Table 3 includes the test set performance on each dataset averaged over three metrics. In tables 2, 3 and 4, classifiers with best performance in each column is **boldfaced**. Classifiers that are not statistically distinguishable from the best classifier in that column at p = 0.05 are \*’ed using ttest\_ind from SciPy’s stats module on five trials (SciPy.org).

Similar to CNM06, gradient boosting classifier outperformed other algorithms on each metric. Results from random forest and multi-layer perceptron using Adam optimizer were not statistically distinguishable from GBOOST. Surprisingly, decision tree classifier had a higher accuracy than what is achieved in CNM06. This may be due to tuning the hyperparameters for max depth, min\_samples\_split and min\_samples\_leaf of the DT. In addition, DT has good performance for LETTER1 and LETTER2, which are easier dataset to classify compared to other datasets. This is also evident in figure 1. DT has higher ROC and precision-recall (PR) score for LETTER1 compared to ADULT dataset. ROC and PR curves for all classifier/dataset combination is included in the appendix section.

|  |  |
| --- | --- |
| Figure1: DT ROC and PR Curve | |
|  |  |

Naïve bayes and logistic regression are among the poor performing classifiers. This distinction becomes more evident after comparing APR scores for each classifier. Even though NB and LOGIT appear to have “good” ACC, their APR is much worse. This proves the point of using a multi metric evaluation setup. Except MLP on LETTER2, gradient boosting classifier has best performance for all datasets. Neural networks outperform other algorithms on the imbalanced LETTER2 dataset. As in CNM06, NB and LOGIT do poorly on all datasets, except LOGIT on ADULT dataset, which is also an imbalanced dataset with 25% positive class. On the training set, shown in table 4, random forest does better than GBOOST on all metrics. Boosting is more susceptible to be affected by the initial iterations of boosting (CNM06). Besides, random forest is more prone to overfitting.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *Table 3:* Test Set Performance (over all metrics) | | | | |
| MODEL | ADULT | LETTER1 | LETTER2 | COVTYPE | MEAN |
| LOGIT | 0.832672\* | 0.784137 | 0.650625 | 0.794205 | 0.76540975 |
| TREES | 0.817271 | 0.908687 | 0.879192 | 0.782507 | 0.84691425 |
| FOREST | 0.839766\* | 0.975356\* | 0.976052\* | 0.868150\* | 0.914831\* |
| GBOOST | **0.862398** | **0.979594** | 0.976011\* | **0.868472** | **0.92161875** |
| MLP\_ADAM | 0.816646 | 0.975062\* | **0.982937** | 0.843133\* | 0.9044445\* |
| NB | 0.777775 | 0.573208 | 0.543166 | 0.713114 | 0.65181575 |

**4. Discussion and Conclusion**

Gradient boosting outperformed all classifiers on the three metrics and was the best on all datasets except LETTER2, in which neural network classifier did best. On ADULT dataset, LOGIT and DT performed better than neural network classifier. This solidifies the point of No Free Lunch Theorem. It was also noteworthy that decision tree classifier performed much better than what was reported on CNM06. Thus, cross validation on different hyperparameter settings is key to finding the true difference among classifiers performance. It is also important to evaluate models on different metrics to get a better measurement of each model’s features.

Comparison of the classifiers depended on use of null hypothesis statistical testing (NHST). This paper used p-value of 0.05 or 95% confidence interval. However, this **does not** mean that one classifier outperforms another classifier with 95% probability (Benavoli et al., 2017). According to Benavoli et al, it is “the probability of getting the observed (or a larger) difference between classifiers if the null hypothesis of equivalence was true.” To find if one classifier is better than another or to get the probability of a classifier being better (posterior probabilities), a better approach is to use Bayesian correlated t-test, Bayesian signed rank test or a Bayesian hierarchical model (2017). Similar to realizing the shortcomings of using a single metric to evaluate an algorithm, this paper realizes the shortcomings of using statistical tests that do not appropriately answer the proposed hypothesis.

**5. Bonus Points**

An additional three classifiers were used to get a better comparison. Averaged over trials, each algorithm and dataset combination was evaluated by plotting ROC and PR curves. The plots are provided in the appendix section.

**6. Resources**

Benavoli, A., Corani, G., Demšar, J., & Zaffalon, M. (2017). Time for a change: A tutorial for comparing multiple classifiers through bayesian analysis. *Journal of Machine Learning Research*, *18*(77), 1–36. https://jmlr.org/papers/v18/16-305.html

*Metrics and scoring: quantifying the quality of predictions — scikit-learn 0.22.1 documentation*. (n.d.). Scikit-Learn.org. https://scikit-learn.org/stable/modules/model\_evaluation.html

R. Caruana, A. Niculescu-Mizeil, *An Empirical Comparison of Supervised Learning Algorithms*, 2008

Scikit-Learn. (2019). *1. supervised learning - scikit-learn 0.21.3 documentation*. Scikit-Learn.org. https://scikit-learn.org/stable/supervised\_learning.html#supervised-learning

*Scipy.stats.ttest\_ind — SciPy v1.4.1 Reference Guide*. (n.d.). Docs.scipy.org. https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.ttest\_ind.html

*Sklearn.metrics.average\_precision\_score — scikit-learn 0.24.1 documentation*. (n.d.). Scikit-Learn.org. Retrieved March 18, 2021, from https://scikit-learn.org/stable/modules/generated/sklearn.metrics.average\_precision\_score.html

*Sklearn.model\_selection.GridSearchCV — scikit-learn 0.22 documentation*. (2019). Scikit-Learn.org. https://scikit-learn.org/stable/modules/generated/sklearn.model\_selection.GridSearchCV.html

*UCI Machine Learning Repository*. (2018). Uci.edu. https://archive.ics.uci.edu/ml/index.php

Wikipedia. (2019, March 20). *Receiver operating characteristic*. Wikipedia; Wikimedia Foundation. https://en.wikipedia.org/wiki/Receiver\_operating\_characteristic

**7. Appendix**

|  |  |  |  |
| --- | --- | --- | --- |
| P-values for *Table 2* | | | |
| MODEL | ACC | APR | ROC |
| LOGIT | 0.002565 | 1.68E-04 | 9.81E-10 |
| TREES | 0.110256 | 4.99E-06 | 4.54E-04 |
| FOREST | 0.754209 | 6.98E-01 | 7.95E-01 |
| GBOOST | 1.0 | 1.0 | 1.0 |
| MLP\_ADAM | 0.574147 | 3.74E-01 | 4.47E-01 |
| NB | 0.000155 | 1.48E-07 | 5.97E-11 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| P-values for *Table 3* | | | | |
| MODEL | ADULT | LETTER1 | LETTER2 | COVTYPE |
| LOGIT | 0.13882 | 5.41E-16 | 0.002483 | 8.57E-07 |
| TREES | 0.044035 | 5.18E-11 | 0.003152 | 3.38E-08 |
| FOREST | 0.222889 | 0.5732501 | 0.37994 | 0.9801786 |
| GBOOST | 1.0 | 1.0 | 0.389911 | 1.0 |
| MLP\_ADAM | 0.043496 | 0.5360766 | 1 | 0.05236602 |
| NB | 0.001434 | 1.25E-33 | 0.000155 | 2.44E-13 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *Table 4:* Training Set Performance (over four datasets) | | | |  |
| MODEL | ACC | APR | ROC | MEAN |
| LOGIT | 0.82566 | 0.633612 | 0.856742 | 0.77200467 |
| TREES | 0.91844 | 0.896812 | 0.945291 | 0.920181 |
| FOREST | **0.99996** | **1.000000** | **1.000000** | **0.99998667** |
| GBOOST | 0.97645 | 0.979642 | 0.989608 | 0.9819 |
| MLP\_ADAM | 0.93330 | 0.934849 | 0.970080 | 0.94607633 |
| NB | 0.75123 | 0.508511 | 0.708125 | 0.65595533 |

|  |  |  |  |
| --- | --- | --- | --- |
| P-values for *Table 4* | | | |
| MODEL | ACC | APR | ROC |
| LOGIT | 4.30E-10 | 3.07E-06 | 2.98E-19 |
| TREES | 2.83E-05 | 6.06E-07 | 1.99E-06 |
| FOREST | 1.0 | 1.0 | 1.0 |
| GBOOST | 1.68E-02 | 1.77E-02 | 2.15E-02 |
| MLP\_ADAM | 6.83E-05 | 3.03E-03 | 5.91E-04 |
| NB | 9.88E-09 | 9.69E-10 | 1.27E-13 |