COGS 118A: Comparing Models for Classification

Harmeena Sandhu A15534745
Department of Cognitive Science,
COGS 118A Winter 2021, Fleischer

Abstract

The purpose of this paper is to replicate the work described in Caruana and Niculescu-Mizil's paper, written in 2006. That paper reports the results of the following algorithms: SVMs, ANN, Logistic Regression, KNN, Random Forests, Decision Trees, Bagged Trees, and Bagged Trees with Boosted Trees. These algorithms' performances are compared using the following threshold metrics: Accuracy, F-score, Lift. Grid searches are performed on each algorithm, which produces optimal performance with each dataset (as the most appropriate hyperparameters are established). The data gathered in *this* replication of the Caruana paper includes an assessment of accuracy, f1 score, and roc auc score, and what is reported is an average of those scores over 5 trials on each of the four datasets that were used, as well as averages across each metric for four datasets. With the original paper, it was found that decision trees alone are not the most efficient or effective in attaining accurate and precise results. Gradient boosted trees, as well as the random forests algorithm are much more effective in producing consistently better results with varying datasets.

Keywords: Support vector machine, k-Nearest Neighbors, Binary classification, Random forest, Logistic Regression, Machine learning, supervised machine learning models, Metrics

1. Introduction

CNM06, which refers to the Caruana-Niculescu-Mizil paper, compares myriad machine learning models in the realm of supervised learning. That paper was able to directly compare the performances of each algorithm, a feat that had not yet been tackled at the time of its publication. It also explored more sophisticated algorithms such as SVMs, random forests, and boosted trees (SVMs and Random Forests are covered in *this* paper). CNM06 also reported refined results of each algorithm, taking note of the differing threshold metrics, being F1 score and accuracy score, and ranking metrics such as AUC-ROC score, which deemed influential in assuring accuracy in evaluations of each algorithm. This is the case because rank metrics ensure that positive instances are ordered before negative instances, and because threshold metrics are determined by whether the correct predictions for the data are made.

The Caruana-Niculescu-Mizil paper experimented using eleven strong datasets with over 10,000 samples to ensure enough data to produce accurate results. With that, scores across algorithms and across datasets were averaged and reported. With those datasets and the SVM,

logistic regression, neural nets, kNN, decision trees, random forests, and naive bayes algorithms, the paper concluded that random forests was of the higher performing algorithms and logistic regression was of the lower performing algorithms. The public was informed that despite those algorithms falling where they did on the scale of performance, there are instances where the overall best can perform sub par in comparison to the overall worst given the dataset and type of problem that is meant to be solved.

This particular paper is modeled after CNM06, where a similar experimentation style is used across four algorithms, being logistic regression, k-nearest neighbors, support vector machines, and random forests. Four datasets were used, being Cover Type, Letter Recognition, Nursery, and California Housing. This experiment is done on a much smaller scale than the original, yet the results appear consistent, given the algorithms and metrics and datasets used. This helps in determining that the results of the 2006 paper, for the most part, are in line with the findings of this paper, despite refining the metrics and *slightly* skewing the hyperparameters used with each algorithm.

2. Methodology

2.1 Learning Algorithms

This paper explores the following algorithms: Logistic Regression, k-Nearest Neighbors, Support Vector Machines, and Random Forests. Each algorithm implements a gridsearch that selects the optimal hyperparameters that adjust each of the classifiers to produce the highest scoring results for that particular algorithm. Each gridsearch includes 5-fold cross validation, which ultimately determines how generalizable these selections are. Upon reporting the results, a paired t test was performed to determine how likely it is for the rankings of metrics to repeat themselves, or if there is any particular statistical significance with one metric over the other.

2.2 Classifiers of Each Algorithm

Logistic Regression utilized the following C - values : 10^-8, 10^-7, 10^-6, 10^-5, 10^-4, 10^-3, 10^-2, 10^-1, 1, 10^1, 10^2, 10^3, 10^4, in combination with the ‘saga’ solver and [‘saga’ ‘lfbgs’] solver array, as well as classifier penalties of ‘none’ or ‘l2’. A max_iter (max iterations) parameter was given to the logistic regression classifier as a way to eliminate a runtime error.

k-Nearest Neighbors utilized up to 25 neighbors (more specifically, a range from 1-26) , and uniform and distance weights that determine the importance of each neighbor as either equally weighted or weighted with regards to distance respectively. It also incorporates the Minkowski metric as either 1 or 2 (the standard euclidean), which determines distance of neighbors.

Standard Support Vectors utilized only two C-values, being 1 and 10, as high C values lead to long training times. High C’s also weigh misclassification errors more heavily, and in

turn have harder margins, whereas low C's prioritize maximizing the margins and in turn result in more misclassifications. The kernel used was ['linear', 'poly', 'rbf', 'sigmoid'], which are different patterns used to separate the classifications of data, (with linear being a straight line and sigmoid having more of a hyperbolic tangent shape). The default degree used with the poly kernel is 3, but 5 was also added to provide more variation. Gamma values of ['scale', 'auto'] are paired with the rbf classifier kernel. Gamma values determine the reach of individual training values of each dataset.

Random Forests utilized 1024 decision trees. The maximum number of features per tree were split by the following: [1, 2, 4, 6, 8, 12, 16, 20]. That determines the subsets / splits that are to be considered before the tree splits. The max_depth parameter was also included as [4,5,6,7,8], which indicates how deep each tree in the forest will go, with the deeper the tree, the more splits, and the more information it has on the data it is classifying.

2.3 Performance Metrics

Each algorithm's performance given the 5-fold cross validation, grid search and predictions for training and testing sections of the data were scored using the following metrics.

Area Under The Curve - Receiver Operating Characteristics (AUC-ROC) is a performance measurement that displays a curve of probability against separability. The optimum AUC-ROC curve would be a right angle in the upper left corner of the graph, as that would maximize the area under the curve. It is calculated with the following formula:
True Positive Rate = True Positive / (True Positive + False Negative)
False Positive Rate = False Positive/(False Positive + True Negative)
Plot against each other and find the area under the curve.

Accuracy is a performance metric that is a number between 0 and 1. It summarizes the results of the performance of a particular model, which ultimately helps in evaluating the effectiveness of models. It is calculated with the following formula:
Accuracy = (True Positive + True Negative)*(1/Total population)

F1-Score is another metric that acts to measure the accuracy and effectiveness of a model, and it does so by looking at the recall and precision of the model. Like accuracy, it is also a number between 0 and 1. It is calculated with the following formula:
F1 = 2*((Precision * Recall) /(Precision * Recall)) = True Positives / (True Positives + (½) (False Positives + False Negatives)

2.4 Datasets

This paper's results are based on four datasets which can be acquired from the UCI Machine Learning Library, being the COVTYPE, Letter Recognition, Nursery, and California Housing

datasets. Refer to the attached notebook in the appendix for detailed visualizations of each dataset. The table below provides a summary of the four datasets.

Table 1: Dataset Information

Model		Attributes	Train Size	Test Size	Positive Ratio
Letter Recognition		20,000	5,000	15,000	0.497
California Housing		20,640	5,000	15,640	0.714
Cover Type		581,012	5,000	576,012	0.488
Nursery		12,960	5,000	7,960	0.566

From Table 1, we can see that the datasets all consist of over 10,00 samples, making the 5,000 training set size suitable. We also see that Cover Type has significantly more attributes than the other datasets, with Letter Recognition and California Housing both having around 20,000 and Nursery having the fewest samples. The data sets all appear to have an even distribution of positive and negative classifications, with the exception of California Housing, having a 70 - 30 ratio of positive to negative classifications. That may influence the results shown in Table 3, being the average metric scores across datasets, but a 70-30 split still has a substantial amount of variability.

3. Experiment & Results

This experiment was run by first splitting each dataset to where the training data consisted of 5000 random samples of data for each dataset, and thus making the remaining samples a part of the testing. Then, with each split made, the algorithm loops for a total of 5 trials per dataset, producing raw AUC-ROC, Accuracy and F1 scores to determine the performance of that algorithm for each dataset. Running the algorithm with the training data first allows it to predict the testing scores, which are a stronger indication of the effectiveness of each algorithm. Each algorithm was written separately, and each of the four datasets pases through each aloirthm with five trials each. The code for this is attached via the notebook in the appendix.

Tables 2, 3, 4 and 5 concisely describe the results of this experiment. The bolded numbers in each represent the best scoring model of each metric. Table 2 shows the highest scoring metric per model, and Table 3 shows the highest scoring dataset (averaged metrics) per model. The bolded value represents the highest number and numbers with an asterisk appended mean that the p-value was greater than 0.05, calculated in the code found in the appendix.

Table 2. Test Set Performance--Metrics

Model		AUC-ROC	Accuracy	F1	Means
Logistic Regression		0.818055	0.780082	0.790145	0.796094
k-Nearest Neighbors		0.878899	0.854184	0.882819*	0.871967
Support Vector Machines		0.787722	0.830476	0.834958*	0.81771867
Random Forests		**0.958586**	**0.901542**	**0.910325**	0.92348433

Given the results reported in Table 2, Random forests has the highest scores for all three metrics. These results were expected and are in line with the findings of the Caruana Paper, as Random Forests massively outshined Logistic Regression, as it does here, with k-Nearest Neighbors staying relatively in the middle of the pack across all three metrics.

These findings are not surprising, as Random Forests uses multiple, random decision trees to yield results, which deems appropriate for datasets of this caliber, as the hyperparameters given were meant to "understand" the data on a deeper level than the other algorithms, which look more at general patterns. Logistic Regression was the lowest in every metric with the exception of AUC-ROC, which indicates that it is slightly better at distinguishing classes than SVMs overall with this data. This could be because the C values used with my SVMs algorithm were relatively small, being 1 and 10, making the margins between the classes much softer than high C values would, therefore allowing more misclassifications to occur. This was done simply because the runtime of SVMs was exceeding 24 hours with more extreme C values than those chosen. K-Nearest Neighbors maintaining a steady value between 0.87-0.89 across each metric indicates that a few more neighbors may be a way to slightly boost the effectiveness of this algorithm, but too many may cause issues with overfitting. It was expected to perform in the middle of the pack simply because it's pattern recognition against the training set is effective, but the depth of the Random Forests algorithm in this case was simply better in handling and understanding these datasets.

The tables found in the appendix (5, 6, 7) report the results of a paired t test conducted using the raw data of each metric of each trial. Refer to Table 2 to see the bolded values, which represent the highest scoring algorithms per metric. The asterisked values represent instances where the p value (which is determined by conducting the t-test), it represents a statistically significant difference, whereas un asterisked values have a p value of less than 0.05, meaning there is less of a statistically significant difference. This analysis answers the question of "why is one value higher", and the results can be explained by the fact that auc roc and accuracy are more uniform metrics, whereas f1 is not as uniform because of the nature of binary classification.

Table 3. Average Across Metrics -- Datasets

Model		Letter	Housing	CoverType	Nursery	Means
Logistic Regression		0.752645	0.850333	0.619167	0.962231	0.796094
k-Nearest Neighbors		**0.969433**	0.727324	0.807656	0.983455	0.871967
Support Vector Machines		0.923429	0.682568	0.664878	**1.000000**	0.81771875
Random Forests		0.958488	**0.923169**	**0.844279**	0.967987	0.92348075

Table 3 provides more insight on the datasets used for the algorithms. It indicates that the Nursery dataset had the best response from every algorithm, with 100% effectiveness with the SVM algorithm. Upon closer examination of this dataset in the appendix, you can see that the predictability in general is not too complicated. There are a limited number of attributes with the Nursery dataset, meaning the algorithms have less data to process to make the classification decision. The classification decision is based on adding the values of the rows, and if that number is less than 8, it classifies as 1. SVMs managed to figure that out 100% of the time with this dataset as the few low C values stretch the margin, which in turn would cause such a high performance.

Covertype on the other hand had relatively low responses from Logistic Regression and SVMs. When referring to Table 1, Covertype has the least even distribution of positive and negative classifications, and it is also the largest dataset, implying that there is more variety in the data. With that, Logistic Regression and Support Vector Machines would probably require more parameters to be included in the hyperparameter search (which would also take longer to run). The low C values of SVM and the misclassification leniency affected Cover type negatively.

That C value discrepancy also negatively affected Housing, however, Logistic Regression performed better on Housing than k-nearest neighbors. This can be attributed to the range in attributes, which can be referenced in the data visualization section, as the pairplot shows. Overall, housing had the most variance in performance by each algorithm.

The Letter dataset was the second highest performing dataset, with Logistic Regression performing behind the others by about .2 , which is quite a lot. This can be attributed again to Logistic Regression being less robust. SVMs performed in the middle, and k-nearest neighbors and random forests performed highly together. Those high results can be attributed not only to the effectiveness of the algorithm, but rather the train and testability of the dataset, which can be referred to in the data visualizations section of the code linked in the appendix.

Table 4. Train Set Performance--Metrics

Model		AUC-ROC	Accuracy	F1	Means
Logistic Regression		0.818531	0.78206	0.792264	0.79761
k-Nearest Neighbors		**1.000000**	0.97249	0.965805	0.97943
Support Vector Machines		0.796270	0.86214	0.892102	0.85017
Random Forests		**1.000000**	**1.00000**	**0.999993**	**0.99999**

From Table 4, we are able to compare our test set performance against our train set performance. With that there are no surprises that the train set is much more accurate than the test set, as the train set is a smaller sample size that is what the model is using to predict the test set. Consistent with the test set, Random Forests is consistently the highest, with kNN remaining in the number 2 spot, and Logistic regression falling below SVMs. This is consistent with what is expected. The rankings of the algorithms remain the same as the test set, but the values overall are higher.

To acquaint yourself with the raw values attained in this experiment, please refer to the secondary results portion of the notebook attached in the appendix of this paper.

4. Conclusion

In conclusion, the findings of this study are overall in line with those of the original Caruana Paper. Random Forests consistently performs at the top in terms of the tested metrics, with Logistic Regression and Support Vector Machines falling on the low end of performance with these specific algorithms. Not only is this the case across AUC-ROC, Accuracy and the F1 metrics, but also throughout the datasets (refer to Tables 2 and 3). In general, the datasets were relatively similar in terms of size, with COVTYPE being the only noticeably larger dataset (which affected its performance results) as well as Nursery being the most basic dataset of the four (which resulted in it performing the highest in every scenario). With that, we are able to further validate the findings of the original study, as our conclusions are very much in line with what was expected given CNM06.

5. Bonus

Three algorithms were required for this assignment, and this paper explores the results of an additional 4th algorithm. Adding a fourth algorithm provides more information to base the conclusion off of, and it gives more range to compare the results of the metrics of each algorithm, and their performance on each of the specified datasets.

Within the appendix, within the notebook, various efforts were made to visualize the datasets that were used in this paper. Each of the four datasets can be visualized with a boxplot grid of each attribute and the range of values each attribute is composed of to provide a univariate analysis for continuous features. Aside from that, a correlation matrix within a heatmap was made for each dataset in order to visualize the most correlated and significant features in determining the ultimate classification of a sample. Along with that, a chart with specified correlative values assigned to each variable was made to complement each heatmap. Pair Grids were also created to understand the linearly separable data by plotting each feature against one another.

For the Nursery Dataset in particular, a lot more work was dedicated to cleaning and turning what was a categorical dataset into a totally binary dataset. With reference to the notebook attached in the appendix, 9 total methods were made from scratch to enumerate the categorical variables that described the sample. From there, a new column 'qualify_rank' was created as a binary column that added the enumerated values, where another method was applied to that column to classify the values as binary.

The Nursery Dataset also includes a more personalized set of visualizations, including a hand created bar plot grid of each attribute against the qualify_rank column, where a trend can be seen where the higher enumerated value within the attribute, the closer a score of 1 is achieved in the binary column (refer to the notebook attached in the appendix).

6. Appendix

Here, the t-test tables that calculate the p values are found, and where the notebook containing the code can be accessed.

Table 5. Paired T-Test: P Values for AUC ROC

Model		T-statistic	P-value
RF vs LR		8.604045413971937	5.5917587933700904e-08
RF vs kNN		2.8710784659180013	0.00978018745266546
RF vs SVM		4.283004566048179	0.0004016336150604851

Table 6. Paired T-Test: P Values for Accuracy

Model		T-statistic	P-value
RF vs LR		5.309339665166026	4.002643632947779e-05
RF vs kNN		2.3990486587300883	0.02685814784288987

RF vs SVM		3.099492578154259	0.0059017919899592725

Table 7. Paired T-Test: P Values for F1

Model		T-statistic	P-value
RF vs LR		4.823222345091527	0.0001181338127309156 1
RF vs kNN		1.9433827395303251	0.06693398899011388
RF vs SVM		1.6134623808194055	0.12312988362538504

The notebook containing code is appended after references.

7. References

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.

“Using GridSearch and K-Fold Cross Val?: Data Science and Machine Learning.” *Kaggle*, www.kaggle.com/questions-and-answers/30560.

Malik, Usman. “Cross Validation and Grid Search for Model Selection in Python.” *Stack Abuse*, Stack Abuse, stackabuse.com/cross-validation-and-grid-search-for-model-selection-in-python/.

“Sklearn.model_selection.GridSearchCV¶.” *Scikit*, scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html#sklearn-model-selection-gridsearchcv.

“Sklearn.linear_model.LogisticRegression¶.” *Scikit*, scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html#sklearn.linear_model.LogisticRegression.predict_proba.

“Sklearn.svm.SVC¶.” *Scikit*, scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html#sklearn.svm.SVC.predict.

“Sklearn.neighbors.KNeighborsClassifier¶.” *Scikit*, scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html#sklearn.

neighbors.KNeighborsClassifier.predict_proba
scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html#sklearn.ensemble.RandomForestClassifier.predict_proba.

Ideas discussed/brainstormed in Office Hours and with Urmi, Anjali, Zeeshan, Ashna, Yukti, and Yana