Comparing Supervised Algorithms

Preston Wong pww001@ucsd.edu

Abstract

With many supervised learning algorithms available, it has always peaked interest in which methods will be the most efficient and accurate. Using the UCI repository and the scikit-learn libraries, the following supervised learning algorithms are studied and compared against three different datasets: decision trees, random forest, and neural networks. These algorithms are tested with performance metrics such as accuracy, to find a comprehensive difference among each dataset and algorithm.

1. Introduction

There have been multiple studies on comparisons of supervised learning algorithms, though the most comprehensive paper that was best to be studied and reproduced for testing results was An Empirical Comparison of Supervised Learning Algorithms (Caruna & Niculescu-Mizil, 2006). Through their comparisons of nine different learning algorithms and eleven datasets proves to be very useful for academia to discover and compare the current learning algorithms available.

As more data is being collected, these supervised learning algorithms can be important for tasks being predictions that can help classification problems, in which this paper will be also studying. It is significant for the comparison of the algorithms that can help more accurately classify and handle larger amounts of data if needed. This paper will study an empirical comparison on three supervised learning algorithms evaluated on their accuracies. The supervised learning algorithms that will be used are decision trees, random forests, and neural networks. Also examined are the tuned hyperparameters for each supervised learning algorithm, in which the best hyperparameter will be used to obtain the best accuracy for each.

Following similar methodology and testing with the Caruna and Niculescu-Mizil paper, the results produced in this paper were similar within a 5-10% testing difference. This can be accounted for in the methods section where the implementations of the algorithms varied from the paper which will be discussed. Out of the three supervised learning algorithms tested here, the results were not surprising at all, that follows the paper where random forests obtained the highest accuracy while neural networks the worst out of the three.

2. Methodology

2.1 Datasets and Preprocessing

To compare and study the different algorithms, the supervised algorithms are applied to three binary classification problems. The three datasets are from the UCI repository which include COV, ADULT, and LETTER datasets (Blake & Merz, 1998). Each dataset does not contain classified binary values to begin with, so some data preprocessing will be involved for each particular dataset.

For the COV dataset, which examines forest cover types, the largest class will be chosen to be the positive class while the rest will be negative (Caruna & Niculescu-Mizil, 2006). In this case, the largest class used was class 2, and in further detail from the description page, the Lodgepole Pine class will be positive which contains 283,301 records to be classified as a positive +1, while the rest of the records will be negative containing 297,771 records. To convert this, whenever the class value was 2, it was filled with a positive value of 1, while all others were negative as 0.

The LETTER dataset examines 26 capital letters of the English alphabet that is represented with 16 integer attributes for each letter from A-Z (Caruna & Niculescu-Mizil, 2006). To represent the classification as a balanced problem, letters A-M are classified positive while the rest, N-Z are negative. To convert each letter to be classified as binary values, the uppercase letters are first converted to ASCII values, and for each ASCII value that represented A-M, it was filled with a positive value of 1, while all others were negative as 0.

ADULT dataset predicts whether an adult will exceed an annual income of $50K that contain nominal attributes (Caruna & Niculescu-Mizil, 2006). It is simple to convert this to a binary classification problem, where if it is >50K then it will be classified as a positive value of 1, while all others, or <=50K were classified as a negative value of 0. Changing the nominal values of the ADULT dataset was used by one-hot encoding the values of each attribute while also dropping any rows with NULL or unknown values. This scaled the original 14 attributes to become 100 total attributes after one-hot encoding.

DATASET	# ATTRIBUTES	TOTAL SIZE
COV_TYPE	54	581011
LETTER	16	20000
ADULT	100	32561

Table 1. Dataset description.

2.2 Learning Algorithms

Using as similar hyperparameters as possible to the Caruna & Niculescu-Mizil paper, the three supervised learning algorithms are through the implementations of the scikit-learn libraries. From the scikit-learn libraries also used is a cross-validation implementation to find the best hyperparameter for each algorithm.

Random Forest: Using the ensemble module from scikit-learn, we use the RandomForestClassifier to reproduce the methods. With this implementation, the default number of trees in the forest is set to 10, but changed to 1024 reproduce similar results. Then the hyperparameters used to determine the size of each split are: {2, 4, 6, 8, 12, 16, 20}.

Decision Trees: The decision tree classifier is similar to the homework problem performed in class also using the scikit-learn library. For the values to determine the maximum depth of the tree, the hyperparameters used are integers 1 through 20 with a step size of 1.

Neural Networks: The neural network implementation from scikit-learn used is the multi-layer perception classifier using a stochastic gradient descent optimizer, with hyperparameters of the hidden layers and momentum for gradient descent updates. The hyperparameters used for the hidden layers are: {1, 2, 4, 8, 32, 128}, and for the momentum update: {0, 0.1, 0.2, 0.5, 0.8, 0.9}.

3. Experiments

To test each supervised learning algorithm on each full dataset would be very expensive and time consuming in some cases, so to avoid that we use 5000 random values to be trained and tested for binary classification. For those 5000 values used, we do splits of the data by 80% training /20% validation, 50% training/50% validation, 20% training/80% validation for each dataset and algorithm used. After data preprocessing and tuning the hyperparameters with 5-fold cross-validation, the best hyperparameter will be selected and used for the specified learning algorithm classifier through scikit-learn's implementations. From the classifiers, we can identify three different accuracies: training, validation, testing, on each split for each classifier for each dataset.

The best hyperparameters of each data split for each classifier will be used then the average accuracy of all three splits is also calculated (Table 2,3,4). To calculate the most consistent accuracy, each test case is ran a total of 4 times then averaged, which the table will represent the final averaged values of the accuracies.

	COV_TYPE	ADULT	LETTERS A-M
DECISION TREE			
test1 (80/20)	0.8980	0.8510	0.8780
train1 (80/20)	0.9220	0.8936	0.8732
val1 (80/20)	0.8628	0.8209	0.8115
max_depth:	20.0000	3.0000	20.0000
test2 (50/50)	0.8800	0.8512	0.8508
train2 (50/50)	0.9279	0.9041	0.8841
val2 (50/50)	0.8587	0.8250	0.8039
max_depth:	13.0000	5.0000	18.0000
test3 (20/80)	0.8630	0.8178	0.8318
train3 (20/80)	0.9379	0.9113	0.8845
val3 (20/80)	0.8456	0.8115	0.7607
max_depth:	7.0000	6.0000	18.0000
avg_test	0.8800	0.8400	0.8535
avg_train	0.9296	0.9030	0.8806
avg_val	0.8557	0.8191	0.7920

Table 2. Decision tree classifier results.

RANDOM FOREST	COV_TYPE	ADULT	LETTERS A-M
test1 (80/20)	0.9180	0.8430	0.9070
train1 (80/20)	0.9759	0.9429	0.9805
val1 (80/20)	0.9028	0.8522	0.8993
min_samples_split:	12.0000	12.0000	6.0000
test2 (50/50)	0.9028	0.8528	0.9052
train2 (50/50)	0.9737	0.9445	0.9761
val2 (50/50)	0.8919	0.8477	0.8777
min_samples_split:	8.0000	12.0000	4.0000
test3 (20/80)	0.8748	0.8530	0.8468
train3 (20/80)	0.9664	0.9439	0.9680
val3 (20/80)	0.8779	0.8169	0.8316
min_samples_split:	2.0000	16.0000	4.0000
avg_test	0.8985	0.8496	0.8863
avg_train	0.9720	0.9438	0.9749
avg_val	0.8908	0.8389	0.8695

Table 3. Random forest classifier results.

Neural Network	COV_TYPE	ADULT	LETTERS A-M
test1 (80/20)	0.7100	0.7700	0.8440
train1 (80/20)	0.7659	0.6274	0.7327
val1 (80/20)	0.7637	0.6272	0.7254
hidden_layer_sizes	(128,)	(32,)	(128,)
momentum	0.5000	0.2000	0.1000
test2 (50/50)	0.8324	0.7868	0.8492
train2 (50/50)	0.7467	0.6298	0.7019
val2 (50/50)	0.7445	0.6295	0.6956
hidden_layer_sizes	(128,)	(128,)	(128,)
momentum	0.5000	0.8000	0.2000
test3 (20/80)	0.8060	0.7888	0.8080
train3 (20/80)	0.6995	0.5614	0.6865
val3 (20/80)	0.6987	0.5616	0.6608
hidden_layer_sizes	(1,)	(128,)	(128,)
momentum	0.0000	0.0000	0.2000
avg_test	0.7828	0.7819	0.8337
avg_train	0.7373	0.6062	0.7070
avg_val	0.7357	0.6061	0.6939

Table 4. Neural Network (Multi-layer perceptron) classifier results

After putting together, the results from all test cases and comparing the three supervised algorithms and classifiers, we are able to see similarities in results to Caruna & Niculescu-Mizil paper. The accuracy is close to similar but will not be the exact same due to different implementations of each classifier and varying hyperparameters and dataset sizes. Although, the big differences in the results being the decision tree classifier performing about 20% better than the results in the paper (Table 2). The most accurate being the random forest classifier and the worst being the neural network which aligns with the results from the paper (Table 2, 4). When examining the different data splits, with less training data the testing accuracy decreases for the decision tree and random forest classifier, while the testing accuracy stays around the same or increases a bit when there is less training data.

4. Conclusion

When trying to reproduce the results of the Caruna & Niculescu-Mizil paper the results of the classifiers chosen in this paper are similar to the results also. Random forest provides the best accuracy and performance when testing, and is fairly easy to implement. Decision tree classifier was also performing very well, almost similar to the random forest classifier when testing. Both runtimes for each classifier ran very well and did not take too long of amounts to process large datasets. For the MLP neural network classifier, performance was close to Caruna & Niculescu-Mizil results but the runtime did not perform as well as expected and was sometimes delayed because of the stochastic gradient descent optimizer did not converge as maximum iterations were reached.

5. References

Blake, C., & Merz, C. (1998). UCI repository of machine learning databases.

Caruana, R., & Niculescu-Mizil, A. (2006). *An Empirical Comparison of Supervised Learning Algorithms*