Comparing the performance of Supervised Machine Learning Algorithms on UCI repository datasets

Aidan Bell

Abstract

This paper replicates and extends "An Empirical Comparison of Supervised Learning Algorithms" (Caruana, Niculescu-Mizil 2006) by evaluating several supervised machine learning algorithms on three UCI repository datasets: the Adult dataset and two wine quality datasets. The study focuses on different training-validation splits (20:80, 50:50, 80:20) to assess their impact on model performance. Algorithms such as Linear SVM, Random Forest, Logistic Regression, ANN, KNN, and Decision Trees are optimized using GridsearchCV. Key findings reveal that Random Forests generally outperform other models across datasets, with nuanced differences in performance based on data type and training-validation splits.

1. Introduction

In this paper, I seek to replicate the results of *An Empirical Comparison of Supervised Learning Algorithms* (Caruana, Niculescu-Mizil 2006), while observing how models perform on 20:80, 50:50, and 80:20 train:validation partitions. In this modern world where deep learning is doing incredible things, it is important not to forget about classical machine learning algorithms that can perform really well on small datasets, and don't need huge amounts of computing power to train.

In this paper, three datasets are used. The "Adult" dataset has 14 US census variables and a binary income >50k, <=50k variable. There are 48,842 entries in the dataset. The task is to predict the binary income variable from other variables, like age, native-country, capital-gain, and marital-status (Becker, Kohavi,1996).

The white wine quality dataset (Cortez et. al 2009) has 4898 white wines in it. Each wine has a numerical value for readings such as pH, alcohol content, citric acid content, and a quality score of 1-9. I train predictors to predict if quality >=6.0 (this is the median quality value). Due to a very high number of wines rated "6", the # of wines >=6 is 3258, while the # <6 is 1640. There is no way to split quality into a boolean in a more even manner.

The red wine quality dataset (Cortez et. al 2009) has 1599 red wines in it. It has the same variable types as the white wine dataset, and I also turn quality into a boolean variable by splitting it on quality >=6. The number of "good" quality red wines is 855, "bad" is 744. The task for both of these datasets is to predict if the quality of wine is good or bad based on wine characteristics alone.

Datasets were processed using the python library pandas, and models were implemented using scikit-learn. Experimentation was done in a Jupyter Notebook, and the source code can be found here: https://github.com/abell-ucsd/Cogs118a-project

2. Methodology

2.1. Learning Algorithms

All learning algorithms implemented used scikit-learn and GridsearchCV to find optimal hyperparameters.Once found, the optimal hyperparameters were then used to calculate average accuracies across three trials for each partition size.

Random Forest: I used the Random Forest Classifier with 1024 estimators and a max_features = [1, 2, 4, 8, 16]. This was the only hyperparameter varied in the reference paper (Caruana, Niculescu-Mizil, 2006), so we only vary this hyperparameter for consistency.

Logistic Regression: I also used the logistic regression classifier. The regularizer parameter C was varied by factors of ten from 10^{-7} to 10^{3} similar to the reference paper (Caruana, Niculescu-Mizil, 2006).

ANN (Artificial Neural Network): I implemented a basic one layer MLP, and varied the size of the hidden layer to [1,4,32,128]. Training time was capped at 5 epochs because of processing power limitations. increasing epochs to 10 showed little to no training improvement, so 10 epochs is likely sufficient.

KNN (K-Nearest Neighbors): I used KNN with euclidean distance. I tested 13 values of K ranging from 1-1000, similar to the reference paper but optimized for runtime (Caruana, Niculescu-Mizil, 2006).

Decision Trees: I used the Decision Tree Classifier, and varied gini and entropy criterion. I also varied max depth at [None, 5 10, 15]. All other values were defaults in scikit-learn.

Linear SVM (Support Vector Machine): I used the Support Vector Classifier with a linear kernel, and used GridSearchCV to vary the regularization parameter C by factors of ten from 10^{-7} to 10^{3}. Unlike other models,GridSearchCV had to be re-run for each trial after randomizing data. the regularization hyperparameter must be able to change when re-shuffling data. Finding a value for C and then reshuffling data and training again would sometimes result in accuracy drastically dropping from ~.78 to ~.3. Allowing the hyerparameter to change with reshuffling helped this to some extent, but this wide variation in accuracy for SVM persisted.

Logistic Regression: I varied the regularization parameter C by factors of ten from 10^{-7} to 10^{3}. The solver used in scikit-learn was “liblinear”, and class weight was “balanced”.

3. Experiment

3.1. Process

Each of the learning algorithms described above was trained on each of the 3 selected datasets. Before any training, datasets were shuffled, and 20% of each dataset was set aside for testing. Then, three partitions of train:validation data were made on the remaining data, 20:80, 50:50, and 80:20. For each partition, the model was trained on the training data for that partition with GridSearchCV to find optimal hyperparameters. Accuracy and hyperparameters are saved. Then, training data is shuffled and the same partition size is made again, and the model is trained again with the found hyperparameter. This step is repeated again, and for the given partition, the average Train, validation, and test accuracies are reported across the 3 trials.

Then, this whole process is repeated for each partition size.

3.2. Results

Something strange & unique happened when repeating trials on linear SVM. Most trials had consistently decent accuracy, but ~⅓ of the time accuracy would be much lower, ~40% lower in the adult dataset, and ~6% lower in the red and white wine dataset. I am unsure why this is happening. all other models had very stable accuracies when reshuffling the dataset, only varying 0-2%.

Table 1: Average Accuracy of models on Adult dataset (Becker, Kohavi,1996)

Classifier	Train/Test Split	Training Acc	Validation Acc	Testing Acc	Hyperparameters
Logistic Regression	20:80	0.801254	0.802489	0.802334	{'C': [100]}
Logistic Regression	50:50	0.798423	0.800225	0.799365	{'C': [100]}
Logistic Regression	80:20	0.792149	0.797441	0.793735	{'C': [100]}
Linear SVC	20:80	0.776171	0.773217	0.773706	{'C': [0.1]}
Linear SVC	50:50	0.784381	0.778966	0.782032	{'C': [1e-06]}
Linear SVC	80:20	0.614029	0.609853	0.615382	{'C': [0.01]}
Random Forest	20:80	1.000000	0.854890	0.856587	{'max_features': [16]}
Random Forest	50:50	1.000000	0.855249	0.852288	{'max_features': [16]}
Random Forest	80:20	1.000000	0.857326	0.854745	{'max_features': [16]}
Decision Tree	20:80	0.859483	0.851403	0.854847	{'criterion': ['gini'], 'max_depth': [5]}
Decision Tree	50:50	0.852477	0.847213	0.849729	{'criterion': ['gini'], 'max_depth': [5]}
Decision Tree	80:20	0.853637	0.849648	0.851776	{'criterion': ['gini'], 'max_depth': [5]}
ANN	20:80	0.763757	0.759877	0.760979	{'hidden_layer_sizes': [(32,)]}
ANN	50:50	0.765254	0.758356	0.761900	{'hidden_layer_sizes': [(32,)]}
ANN	80:20	0.763453	0.754148	0.762070	{'hidden_layer_sizes': [(32,)]}
K Neighbors	20:80	0.776171	0.772066	0.773569	{'n_neighbors': [84]}
K Neighbors	50:50	0.788339	0.782106	0.785546	{'n_neighbors': [84]}
K Neighbors	80:20	0.794133	0.781574	0.794657	{'n_neighbors': [84]}

Barring a slight increase in performance for K Nearest Neighbors (and variance in SVM, see above), partition size did not seem to affect accuracy on this massive dataset. Decision trees and Random Forests were the top performers here, followed by Logistic Regression and K Neighbors on the 80:20 partition.

Table 2: Average Accuracy of models on Red Wine dataset (Cortez et. al 2009)

	Classifier	Train/Test Split	Training Acc	Validation Acc	Testing Acc	Hyperparameters
0	Logistic Regression	20:80	0.721569	0.748047	0.743750	{'C': [1]}
1	Logistic Regression	50:50	0.754304	0.737500	0.746875	{'C': [1]}
2	Logistic Regression	80:20	0.748778	0.722656	0.743750	{'C': [1]}
3	Linear SVC	20:80	0.671895	0.620117	0.637500	{'C': [100]}
4	Linear SVC	50:50	0.735003	0.718229	0.729167	{'C': [0.01]}
5	Linear SVC	80:20	0.745520	0.709635	0.731250	{'C': [0.1]}
6	Random Forest	20:80	1.000000	0.725586	0.743750	{'max_features': [2]}
7	Random Forest	50:50	1.000000	0.762500	0.787500	{'max_features': [2]}
8	Random Forest	80:20	1.000000	0.812500	0.806250	{'max_features': [2]}
9	Decision Tree	20:80	0.866667	0.637695	0.687500	{'criterion': ['gini'], 'max_depth': [5]}
10	Decision Tree	50:50	0.809077	0.712500	0.743750	{'criterion': ['gini'], 'max_depth': [5]}
11	Decision Tree	80:20	0.794721	0.664062	0.737500	{'criterion': ['gini'], 'max_depth': [5]}
12	ANN	20:80	0.495425	0.505534	0.509375	{'hidden_layer_sizes': [(4,)]}
13	ANN	50:50	0.516432	0.542187	0.529167	{'hidden_layer_sizes': [(4,)]}
14	ANN	80:20	0.573151	0.549479	0.560417	{'hidden_layer_sizes': [(4,)]}
15	K Neighbors	20:80	0.576471	0.525391	0.531250	{'n_neighbors': [84]}
16	K Neighbors	50:50	0.629108	0.637500	0.603125	{'n_neighbors': [84]}
17	K Neighbors	80:20	0.659824	0.617188	0.628125	{'n_neighbors': [84]}

On this small dataset (<2000 entries), partition size had a large impact on model performance for all models other than Logistic Regression. Logistic regression and Random first had similar test accuracy for the 20:80 partition, but with more training data Random forest surpassed Logistic regression performance, which did not increase as training data increased. Random forest with max features = 2 outperformed all other models by over 5% accuracy in the 80:20 partition.

Table 3: Average Accuracy of models on White Wine dataset (Cortez et. al 2009)

	Classifier	Train/Test Split	Training Acc	Validation Acc	Testing Acc	Hyperparameters
0	Logistic Regression	20:80	0.693487	0.711962	0.704082	{'C': [100]}
1	Logistic Regression	50:50	0.718224	0.710056	0.697959	{'C': [100]}
2	Logistic Regression	80:20	0.719847	0.711735	0.702041	{'C': [100]}
3	Linear SVC	20:80	0.687952	0.674960	0.668027	{'C': [0.01]}
4	Linear SVC	50:50	0.685554	0.693211	0.692177	{'C': [100]}
5	Linear SVC	80:20	0.688151	0.687075	0.686395	{'C': [0.01]}
6	Random Forest	20:80	1.000000	0.780861	0.774490	{'max_features': [1]}
7	Random Forest	50:50	1.000000	0.808065	0.803061	{'max_features': [1]}
8	Random Forest	80:20	1.000000	0.829082	0.827551	{'max_features': [1]}
9	Decision Tree	20:80	0.943806	0.697927	0.656122	{'criterion': ['entropy'], 'max_depth': [10]}
10	Decision Tree	50:50	0.910158	0.727412	0.737755	{'criterion': ['entropy'], 'max_depth': [10]}
11	Decision Tree	80:20	0.884174	0.760204	0.750000	{'criterion': ['entropy'], 'max_depth': [10]}
12	ANN	20:80	0.661132	0.670707	0.689116	{'hidden_layer_sizes': [(32,)]}
13	ANN	50:50	0.625489	0.617152	0.609524	{'hidden_layer_sizes': [(32,)]}
14	ANN	80:20	0.669007	0.668367	0.679592	{'hidden_layer_sizes': [(32,)]}
15	K Neighbors	20:80	0.674330	0.660287	0.664286	{'n_neighbors': [84]}
16	K Neighbors	50:50	0.669729	0.675855	0.680612	{'n_neighbors': [84]}
17	K Neighbors	80:20	0.681876	0.656888	0.689796	{'n_neighbors': [84]}

On this larger dataset (~5k entries) we see Random forest again come out on top, this time by ~10% higher accuracy than all other models. All other models seemed to be stuck at around 70% accuracy.

4. Conclusion

My results of random forests dominating over SVC, Logistic Regression, Decision Tree, ANNs, and KNN algorithms are consistent with the findings of the reference paper *An Empirical Comparison of Supervised Learning Algorithms* by Rich Caruana and Alexandru Niculescu-Mizil. Just like in the reference paper,I also found accuracy of RF to be ~10% higher than the other algorithms implemented. It is very interesting that in the income prediction task, Binary trees performed almost identically to RF, but in wine quality tasks RF significantly outperformed BT. This paper successfully reproduced the findings of *An Empirical Comparison of Supervised Learning Algorithms* on a small scale.

5. Bonus Points

Bonus points are requested for comparing results on 6 different algorithms, rather than just the required 3.

References

Cortez, P. et al. “Modeling wine preferences by data mining from physicochemical properties.” *Decis. Support Syst.* 47 (2009): 547-553.

Becker, Barry and Kohavi, Ronny. (1996). Adult. UCI Machine Learning Repository. https://doi.org/10.24432/C5XW20.

Caruana, Rich, and Alexandru Niculescu-Mizil. "An empirical comparison of supervised learning

algorithms." *Proceedings of the 23rd international conference on Machine learning*. 2006.