# An Empirical Comparison For Three Popular Supervised Learning Algorithms

**Zhongyu Chen**
University of California San Diego
San Diego, CA 92092
zhc180@ucsd.edu

## Abstract

Machine learning has entered the deep learning era for a few years and various neural networks seem to become the first resort for most problem. However, traditional supervised learning algorithm still stand a place in nowadays and can achieve unexpectedly good results in many cases. In this paper, we present the classification results for 3 models: KNN, Random Forest and Adaptive Boosting in 3 different data sets.

## 1 Introduction

With new machine learning algorithms emerging every year, it can be challenging to cherry pick the model that works best for a particular problem. Some models are great for small data sets only and some others are tailored for fast training process with some compromise in accuracy. Although previous study has revealed a comprehensive performance comparison across many classic classifiers, it might be too obscure to understand without substantial knowledge in the field. Hence, a simpler comparison of three supervised learning algorithms become necessary and could serve as a guidance to the students that just set foot on machine learning's realm.

This paper presents results of an empirical comparison of three supervised learning algorithms in terms of accuracy. We evaluate the performance of k nearest neighbors, random forests and boosting on three classification problems (two binary classifications and one multi-class classification). Before the comparison, we have done a thorough hyperparameter tuning to make sure we push each classifier to its limit at a particular problem, and ran training of each models on each data set three times to avoid surprise result.

After all the test and experiment, the result coincide with those from previous studies. To preview: Random forests generally works well in all three problems, and it ranks the first in two of them; K nearest neighbor has surprisingly good accuracy and performs almost as good as boosting algorithm. Boosting, on the other hand, does not give consistent classify performance even though it works almost equally well in two of the problem. Since we also tested the data on three different splits: 20/80, 50/50 and 80/20. The result regarding the relation between size of training data and accuracy also matches the expectation: the more portion of the data is used for training, the better the accuracy is.

## 2 Methodology

### 2.1 Learning Algorithms

**K Nearest Neighbors(KNN):** we compare performance of 20 values of K ranging from K = 1 to K = 180. We tried out uniformly weighted data and inverse distance weighted data. We also set the parallel computing option to use all the CPU core to gain results faster.

**Random Forests (RF):** we tried 100 values of number of estimators ranging from 100 to 1500, and for each decision tree in random forests, we tested 11 values of maximum tree depth between 10 to 110 and different results for the number of features to consider when looking for the best split: 'auto', 'sqrt' and 'log2'.

**AdaBoost Classifier(AB):** we only tune the number of estimators in this ensemble model. The value we have tested are 20, 50 and 100.

### 2.2 Performance Metrics

We use solely the accuracy(ACC) as the metric of our comparison. We used this because it provide a concise yet informative description of a models performance. During cross-validation, we reported both the training accuracy and the validation accuracy to see if any potential over fit has taken place.

### 2.3 Data Sets

We compare the algorithms on 3 classification problems: ADULT, BANK_NOTES, and CAR, all from the UCI Repository(Blake & Merz, 1998). Among these three, ADULT and BANK_NOTES are binary classification problems, and CAR is multiclass classification problem with four class type: 'unacc', 'acc', 'good' and 'v-good'. Originally, labels in ADULT are strings '<=50K' and '>50K', but we converted them to 0 and 1 respectively. There are features consist of string in both ADULT and CAR data sets so we use LabelEncoder to encode these features so that all are converted to integers. We picked LabelEncoder over OneHotEncoder for different reasons in two data sets: CAR's features are ordinal, which means they imply a relation; ADULT's features, though not ordinal, significantly increase in dimension if we use one hot encoding, which make the training process too slow to finish in a short amount of time. Features in BANK_NOTES are floats, and since they are in good distribution, we did not normalize the features, neither did we encode the features in any way.

## 3 Experiment

Given a data set, we partitioned the data set by three partition way: 80% training, 20% testing; 50% training, 50% testing; 20% training, 80% testing. For each kind of partition, we ran three trials to prevent accidental results. In each trial, we shuffle the data and then do the split to make sure that each time we are getting different training and testing data even though we use the same partition rate. Then we ran all three learning algorithms with cross-validation on this data. We designed the code in the way that it prints out the training and validation accuracy after choosing the best hyper parameter. After that, we also average over three trials to get the testing accuracy of our models. We would like to run more trials, but this is a very expensive set of experiments. Fortunately, even with only three trials, we are able to discern interesting differences between methods.

As the hyperparameters reported, we found that for most case the number of estimators for our ensemble learning algorithm is high when we achieve the best cross-validation results, In our case, 467 for the random forest and 100 for the AdaBoost. Moreover, all the runs of the random forest seem to favor a relatively small max tree depth(either 10 or 20 in our case), thereby keeping each weak learner less complicated. The most important hyperparameters in knn, k, works the best in many cases at the value of 19, among the several options we give. Also, 18 out of 27 runs of knn shows that data with their distance weighted work better.

Table 1 shows the accuracy for each algorithm on all 3 problems with 3 ways of partition. We use superscript 1 to denote 80/20 split; 2 to denote 50/50 split; 3 to denote 20/80 split. In the table, each row corresponds to one learning algorithm. This same naming rule applies to Table 2 and 3, where we display the validation and testing accuracy.

Looking at table 3, we can see during testing, the random forest gives the most stable classification performance, ranking the first place 4 out of 9 times. knn is having very good results on BANK data set, but not as well in the other two datasets. It ranked top in 3 out of 9 times. The adaptive boosting algorithm, however, only ranked the first twice. Comparing the same problem with different splitting, we can see a clear tendency that the larger the training set is, the better the testing accuracy is. For example, knn gives an accuracy of 0.919, 0.843 and 0.772 on CAR dataset with training data of 80%, 50%, and 20% respectively. This is also true for the other two models when working on this dataset.

Table 1: Training accuracy over different problems

| MODEL | $ADULT^1$ | $CAR^1$ | $BANK^1$ | $ADULT^2$ | $CAR^2$ | $BANK^2$ | $ADULT^3$ | $CAR^3$ | $BANK^3$ |
|---|---|---|---|---|---|---|---|---|---|
| KNN | 1.000 | 1.000 | 1.000 | 1.000 | 1.000 | 1.000 | 1.000 | 1.000 | 1.000 |
| RF | 0.868 | 1.000 | 1.000 | 0.907 | 1.000 | 1.000 | 0.892 | 1.000 | 1.000 |
| AB | 0.866 | 0.781 | 1.000 | 0.866 | 0.806 | 1.000 | 0.868 | 0.803 | 1.000 |

Table 2: Validation accuracy over different problems

| MODEL | $ADULT^1$ | $CAR^1$ | $BANK^1$ | $ADULT^2$ | $CAR^2$ | $BANK^2$ | $ADULT^3$ | $CAR^3$ | $BANK^3$ |
|---|---|---|---|---|---|---|---|---|---|
| KNN | 0.839 | 0.919 | 1.000 | 0.800 | 0.843 | 0.999 | 0.775 | 0.772 | 0.999 |
| RF | 0.865 | 0.969 | 0.996 | 0.862 | 0.964 | 0.990 | 0.860 | 0.897 | 0.985 |
| AB | 0.866 | 0.806 | 0.996 | 0.866 | 0.809 | 0.987 | 0.861 | 0.770 | 0.990 |

Table 3: Testing accuracy over different problems

| MODEL | $ADULT^1$ | $CAR^1$ | $BANK^1$ | $ADULT^2$ | $CAR^2$ | $BANK^2$ | $ADULT^3$ | $CAR^3$ | $BANK^3$ |
|---|---|---|---|---|---|---|---|---|---|
| KNN | 0.800 | 0.919 | 1.000 | 0.789 | 0.843 | 0.999 | 0.843 | 0.772 | 0.999 |
| RF | 0.865 | 0.969 | 0.996 | 0.862 | 0.964 | 0.990 | 0.964 | 0.897 | 0.984 |
| AB | 0.866 | 0.806 | 0.996 | 0.867 | 0.809 | 0.987 | 0.809 | 0.770 | 0.990 |

Comparing the accuracy in table 3 with those in table 1 and 2, knn always has 100% accuracy at training data but the accuracy drops significantly when it comes to validation or testing accuracy. In addition, for all three models, the training accuracy is generally higher than the validation or testing accuracy in the same problem setup.

## 4 Conclusion

We purposely choose three datasets that can test different facet of the classifiers, with the ADULT, we try to see which one works the best in a relatively large dataset; with the CAR, we try to see how they work in a medium-size dataset where there are multiple classes; with the BANK, we test out the performance of these classifiers on dataset consist of floats data. We see different results on three different datasets and though random forest performs well most of the time, there is some variability. Without throughout calibration, boosting does not work well and sometimes even performs worse than knn, a model that has the worst average testing accuracy among these three but occasionally stands out.

We also want to point out that our results do not match exactly the results from previous work, partially because of the limitation of dataset size and the number of hyper parameters that we are tuning.

## References

[1] Caruana, Rich, and Alexandru Niculescu-Mizil. "An empirical comparison of supervised learning algorithms." Proceedings of the 23rd international conference on Machine learning. ACM, 2006.