An Empirical Comparison of Linear SVM, k-NN and Random Forests Supervised Machine Learning Algorithms

Tomas Haugland Spangelo[a]

[a] *Final Project COGS 118A - University of California, San Diego*

Abstract

There exists a lot of theory on which supervised machine learning algorithms performs the best on certain types of classification problems. Preferences for the variety of different machine learning algorithms have also changed a lot during the history of machine learning, largely due to new applications, new discoveries and the introduction of big data. In this report I will do an empirical comparison of three different machine learning methods, namely support-vector machines (SVM), k-Nearest Neighbors (k-NN) and Random Forests, on three different datasets based on average training and testing performance.

1. Introduction

Supervised machine learning has a wide range of real world applications, for example spam detection, BioInformatics, object-recognition and speech recognition. The utility of such an application relies heavily on the choice of learning algorithm(s) (i.e. type of classifier), and it is therefore critical to choose the learning algorithm best suited for the problem. In this report, you will find an empirical comparison of three different types of supervised machine learning algorithms. That is, linear Support Vector Machines, k-Nearest Neighbors and Random Forests.

At the time Caruna and Niculescu- Mizil completed and published their empirical comparison of machine learning algorithms in 2006, it was one of a few comprehensive studies on the matter [1]. Although not as comprehensive, the goal for this report is to hopefully confirm and replicate some of the findings in their paper by doing a number of trials on two different partitions of three different datasets.

2. Methods

2.1. Learning Algorithms

In this experiment I have used three different supervised machine learning algorithms. Without going into too much detail, I will give a brief explanation of each algorithm, and how cross validation is used to tune the hyperparameters. I use scikit-learn's implementation of all of the three algorithms, including their grid search implementation for cross validation [2].

SVM: In this experiment I have used the linear SVM algorithm. The main idea of SVM is that it introduces the concept of margin (regularization) in the objective function together with a classification error. That is, the loss function that is to be minimized is as follows:

$$L(\vec{w}, b) = \frac{1}{2}||\vec{w}||^2 + C\sum_{i=1}^{n}(max(0, 1 - y_i(\vec{w}^T\vec{x_i} + b))^2), \quad (1)$$

where the first term is the regularization (margin) and the second term is the squared hinge loss, which is what is used in the experiment. The support vectors are described by the training samples that make the positive and negative planes (i.e. the margin). Hence the name support vector machine. The C in the loss function is the hyperparameter to be tuned, and the intuition behind it is that it determines the balance between classification error and margin. The best choice of C is attained for each partition and trial on each dataset using cross validation, with the different choices being {0.1, 1, 10, 100, 1000}.

k-NN: The k-nearest neighborhood algorithm is a non-parametric technique. That is, instead of learning parameters which are fixed with respect to the sample size (e.g. $\vec{w}$ and b in SVM), the number of parameters in non-parametric methods can grow with the sample size. Also, non-parametric methods are direct and easy to implement. The basic idea of the k-NN method is that it classifies $\vec{x}$ by looking at the classification of the k nearest samples, assigning the label to $\vec{x}$ that is is the most frequent among them. The hyperparameter to be tuned using cross validation is k, and it is chosen among the values {1, 2, 3, 4, 5, 6, 7, 8, 9} (i.e. should the classifier be looking at the 2 nearest neighbors? 3 nereast neighbors? etc.).

Random Forests: The Random Forests method is an ensemble based method. That is, the class label for $\vec{x}$ is predicted by aggregating a set of classifiers learned during training and their predictions. Specifically for Random Forests, this means using a combination of decision tree classifiers. Each individual decision tree is based on the values of a random sampling of the training samples. In addition, each decision tree chooses to split at each node

	Adult Dataset (20/80)		Car Dataset (20/80)		EEG Dataset (20/80)	
	Training Accuracy	Testing Accuracy	Training Accuracy	Testing Accuracy	Training Accuracy	Testing Accuracy
SVM	83.2% ± .0977%	82.8% ± .163%	96.5% ± .473%	94.9% ± .974%	55.1% ± .600%	55.6% ± .625%
k-NN	85.4% ± .264%	81.7% ± .141%	93.0% ± .853%	88.5% ± 1.45%	92.8% ± 5.08%	76.2% ± .308%
RF	85.1% ± .103%	82.8% ± .0966%	100% ± 0%	94.4% ± 1.24%	95.6% ± .403%	84.0% ± .439%

Table 1: Averaged accuracy over three trials after classification using linear SVM, k-Nearest Neighborhood (k-NN) and Random Forests (RF) on the datasets ADULT, Car Evaluation and EEG to Predict Eye State with 20/80 split. ± indicates standard deviation.

	Adult Dataset (80/20)		Car Dataset (80/20)		EEG Dataset (80/20)	
	Training Accuracy	Testing Accuracy	Training Accuracy	Testing Accuracy	Training Accuracy	Testing Accuracy
SVM	82.9% ± .0516%	82.9% ± .207%	96.0% ± .224%	95.1% ± .708%	58.1% ± .109%	58.7% ± .235%
k-NN	85.3% ± .0751%	82.3% ± .0276%	98.5% ± .207%	96.5% ± .409%	91.2% ± 1.19%	82.8% ± .680%
RF	83.7% ± .142%	83.0% ± .292%	100% ± 0%	97.3% ± .594%	91.2% ± .188%	86.1% ± .822%

Table 2: Averaged accuracy over three trials after classification using linear SVM, k-Nearest Neighborhood (k-NN) and Random Forests (RF) on the datasets ADULT, Car Evaluation and EEG to Predict Eye State with 80/20 split. ± indicates standard deviation.

(i.e. make a 'choice') based on a random subset of the features. A prediction for $\vec{x}$ is made based on the most frequent prediction from each of the individual decision trees. The hyperparameters to be tuned using cross validation is $n\_estimators$, which is the number of decision trees, and max_depth, which is the maximum depth of each decision tree. $n\_estimators$ is chosen among the values {25, 50, 100}, and max_depth is chosen among the values {1, 5, 10}.

2.2. Performance metrics

Averaged testing accuracy over the three trials on each of the two splits (20/80 and 80/20) on the three datasets is used to compare the performance of the three supervised machine learning algorithms. To get tables of rank order of the classifiers in comparison, which will be presented in the experiment section, I will first make two ranking tables based on the averaged testing accuracy over the three trials on the three datasets for the 20/80 split and the 80/20 individually, and then make a third overall ranking table based on the averaged testing accuracy for both of the splits. This will lead to insight in how the classifiers rank on the two splits and also the experiment as a whole.

2.3. Datasets

The three datasets used for this experiment are from the UCI Repository. The datasets represents three different binary classification problems, and are used to compare three supervised machine learning algorithms (i.e. SVM, k-NN and Random Forests). Please refer to Table 3 for a summary of the datasets.

ADULT [3]: This dataset is used to predict whether a person's income exceeds $50,000 based on census data. The dataset has 48842 samples and 14 features. After looking at some statistics for the data, I decided to drop the features *capital_gain* and *capitol_loss* because the value was zero for almost all samples. I also dropped the categorical feature *education*, because the feature $education - num$ is the numerical representation of *education*, effectively resulting in 11 features. In regard to the missing values (N/A), I decided to remove all of these samples, and this resulted in 45222 samples, which I consider to still be a relatively large number of samples. I used one-hot encoding for the categorical features, and performed normalization to ensure equal weight to the features. The resulting shape of the dataset is 45222x86, excluding the labels.

Problem	#ATTR	#SAMPLES	%POZ
ADULT	11/86	45222	25%
CAR	6/21	1728	30%
EEG	14	14980	45%

Table 3: Summary table for the datasets. #ATTR refers to the number of features A/B, where A is the number of features before one-hot encoding and B is the number of features after one-hot encoding. #SAMPLES refers to the number of samples used, and %POZ is the percentage of samples with positive ground trouth label.

Car Evaluation [4]: The car evaluation dataset is used to predict whether a car is acceptable or not based on price, tech and comfort measures (in total 6 features). The dataset has 1728 samples. Originally, the dataset had four ground truth labels, but I decided to only distinguish between unacceptable and acceptable in order to make it a binary classification problem, treating *acc*, *good* and *vgood* as acceptable. All of the features are categorical, and I used one-hot encoding. The resulting shape of the dataset is 1728x21, excluding the labels.

EEG to Predict Eye State [5]: This dataset is used to predict whether the eyes of an individual is closed or open based on 14 electroencephalogram (EEG) measures. The dataset has 14980 samples. All of the features are numerical, and I performed normalization to ensure equal weight to the features. The resulting shape of the dataset is 14980x14, excluding the labels.

3. Experiments

This section will discuss the results after fitting the three classifiers to the data using the best hyperparameter(s) found using cross validation, with the splits/partitions 20/80 and 80/20. Note that for each partition, the experiment is run three times. That is, for the 20/80 partition the experiment is run three times with the three different classifiers, and the same is done for the 80/20 partition. Consequently, I am looking at 3 trials/repeats x 3 classifiers x 3 datasets x 2 partitions, which is in total 54 runs.

3.1. 20/80 split

Rank order for 20/80 split			
	1st	2nd	3rd
RF	.67	.33	.00
SVM	.33	.33	.33
k-NN	.00	.33	.67

Table 4: Rank order for 20/80 split using averaged testing accuracy from Table 1.

Table 1 shows the classification results from the 20/80 split, and Table 4 shows the ranking order from this split. Reading directly from the ranking table, it is evident that the Random Forests algorithm performs the best in this part of the experiment. SVM is ranked at second, and k-NN is ranked at third.

On the *ADULT* dataset, which is by far the largest dataset, k-NN actually outperforms the two other methods when comparing the averaged training accuracy. However, k-NN is outperformed by both Random Forests and linear SVM when comparing averaged testing accuracy. This speaks to a larger generalization error for k-NN. It is also worth mentioning that when tuning the hyperparameter, k, for k-NN during cross validation we are using a wider array of possible values for k then for the hyperparameters for SVM and Random Forests. This may effect the results. Note that the averaged testing accuracy is the same for linear SVM and Random Forests on the *Adult* dataset. However, when calculating the ranking order for the 20/80 split, Random Forests is preferred due to lower standard deviation. Also, when using more significant figures, the averaged testing accuracy is slightly higher for Random Forests. k-NN is only slightly worse when using averaged testing accuracy as the performance metric.

On the *Car Evaluation* dataset there is once again a tight race between linear SVM and Random Forests. On this dataset, however, linear SVM is best. In contrast with the latter dataset, the difference between second and third place is grater. Clearly, I am on average worst off using k-NN on this classification problem with the 20/80 partition.

Lastly, for the 20/80 split, we have the *EEG to Predict Eye State* classification problem. Here, Random Forests performs the best (relatively) closely followed by k-NN. Linear SVM, however, performs extremely poorly on this classification problem. The averaged accuracy for both training and testing indicates that linear SVM is pretty close to simply guessing the labels, which would result in a accuracy of 50%. In addition, the averaged testing accuracy is higher than the averaged training accuracy by 0.5 percentage points, which only reinforces this. Linear SVM is clearly a poor choice of classifier for this classification problem.

3.2. 80/20 split

Rank order for 80/20 split			
	1st	2nd	3rd
RF	1.0	.00	.00
k-NN	.00	.67	.33
SVM	.00	.33	.67

Table 5: Rank order for 80/20 split using averaged testing accuracy from Table 2.

Table 2 shows the classification results from the 80/20 split, and Table 5 shows the rank order from this split. As with the 20/80 split, Random Forests also performs the best in this part of the experiment. Here, however, Random Forests performs the best on *all* of the classification problems. k-NN ranks in second, and SVM ranks in third.

On the *ADULT* classification problem, Random Forests performs the best, k-NN comes in second, and SVM in third. Although SVM seems to have the same averaged accuracy in training and testing, the training accuracy is actually slightly better, as it should be, using more significant figures. This speaks to high generalizability. This is also the case with Random Forests, as the averaged training accuracy is better than averaged testing accuracy by only .7 percentage points. Please refer to Figure 1, Figure 2 and Figure 3 to see heatmaps representing the results of hyperparameter tuning using cross validation in the last trial for each classifier on the 80/20 split (for the *ADULT* dataset).

Random Forests also performs the best on the *Car Evaluation* dataset, with an averaged training accuracy of 100%. There is a change when comparing to the 20/80 split, because k-NN now outperforms linear SVM. All of the classifiers perform really well on this classification problem with respect to the performance metric (close to 100% accuracy in testing), which is not surprising considering how small the *Car Evaluation* dataset is relative to the two others. Please refer to Figure 4, Figure 5 and Figure 6 to see

Figure 1: Heatmap showing cross validation accuracy for different values of the hyperparameter C for SVM during the last trial of the 80/20 split on the ADULT dataset. The best hyperparameter is C=10.

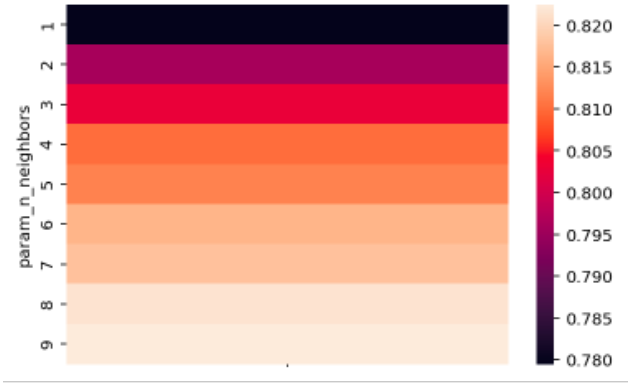

Figure 2: Heatmap showing cross validation accuracy for different values of the hyperparameter k for k-NN during the last trial of the 80/20 split on the ADULT dataset. The best hyperparameter is n_neighbors/k=9.

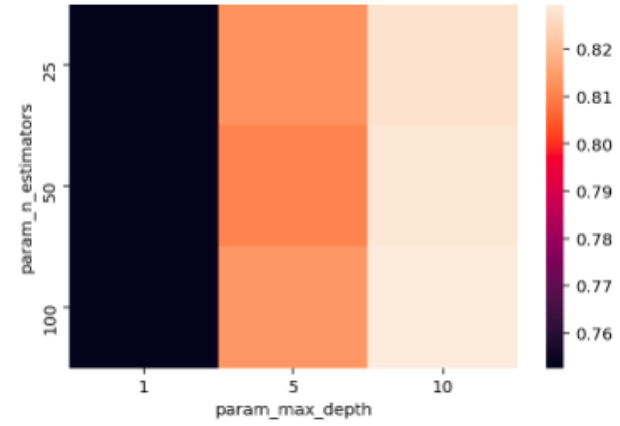

Figure 3: Heatmap showing cross validation accuracy for different combinations of the hyperparameters n_estimators and max_depth for Random Forests during the last trial of the 80/20 split on the ADULT dataset. The best hyperparameters are n_estimators=100 and max_depth=10.

heatmaps representing the results of hyperparameter tuning using cross validation in the last trial for each classifier on the 80/20 split (for the *Car Evaluation* dataset).

Figure 4: Heatmap showing cross validation accuracy for different values of the hyperparameter C for SVM during the last trial of the 80/20 split on the Car Evaluation dataset. The best hyperparameter is C=1.

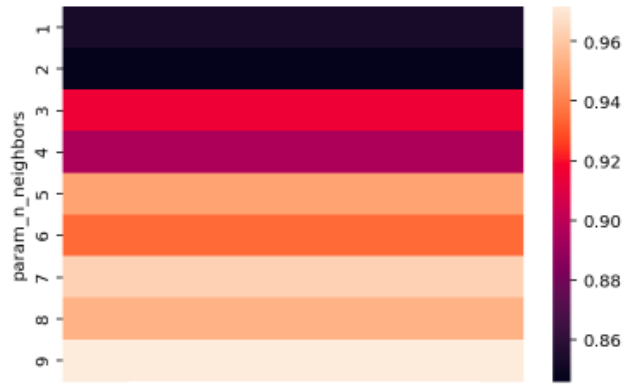

Figure 5: Heatmap showing cross validation accuracy for different values of the hyperparameter k for k-NN during the last trial of the 80/20 split on the Car Evaluation dataset. The best hyperparameter is n_neighbors/k=9.

As with the 20/80 split, SVM performs poorly on the *EEG to Predict Eye State* classification problem. Although slightly better both in averaged training and testing accuracy, it still performs close to a random guess. In addition, testing accuracy is higher than training accuracy. Both Random Forests and k-NN performs better in training compared to the 20/80 split. Once again, Random Forests performs the best. k-NN is in second, and SVM is by far in third. Please refer to Figure 7, Figure 8 and Figure 9 to see heatmaps representing the results of hyperparameter tuning using cross validation in the last trial for each classifier on the 80/20 split (for the *EEG to Predict Eye State* dataset).

3.3. Overall ranking and split comparison

Table 6 shows the overall rank ordering of the classifiers based on the overall averaged testing accuracy. It is evident

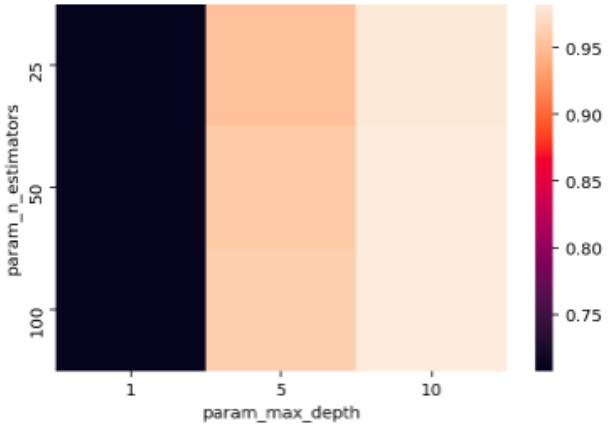

Figure 6: Heatmap showing cross validation accuracy for different combinations of the hyperparameters n_estimators and max_depth for Random Forests during the last trial of the 80/20 split on the Car Evaluation dataset. The best hyperparameters are n_estimators=50 and max_depth=10.

Figure 7: Heatmap showing cross validation accuracy for different values of the hyperparameter C for SVM during the last trial of the 80/20 split on the EEG to Prediction Eye State dataset. The best hyperparameter is C=100.

Figure 8: Heatmap showing cross validation accuracy for different values of the hyperparameter k for k-NN during the last trial of the 80/20 split on the EEG to Predict Eye State dataset. The best hyperparameter is n_neighbors/k=5.

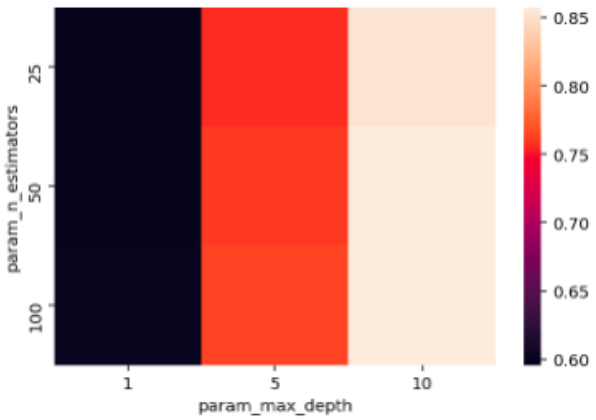

Figure 9: Heatmap showing cross validation accuracy for different combinations of the hyperparameters n_estimators and max_depth for Random Forests during the last trial of the 80/20 split on the EEG to Predict Eye State dataset. The best hyperparameters are n_estimators=100 and max_depth=10.

Overall rank order			
	1st	2nd	3rd
RF	1.0	.00	.00
SVM	.00	.67	.33
k-NN	.00	.33	.67

Table 6: Overall rank order using the averaged testing accuracy over both splits from Table 1 and Table 2.

from this table that the best supervised machine learning algorithm in this experiment is Random Forests. In second comes linear SVM, and in last place k-NN. Even though k-NN ended up in last place, this algorithm scored consistently relatively good with respect to averaged training accuracy. Random Forests also performed good in each split on each of the three datasets. SVM performed good on the *ADULT* and *Car Evaluation* classification problems, but performed poorly on the *EEG to Predict Eye State* classification problem. Choosing another dataset could have given linear SVM a better chance of beating the Random Forests, but since Random Forests performed the best on the two other datasets, it still would have won.

When comparing the averaged testing accuracy for the 20/80 and the 80/20 split using Table 1 and Table 2, we can clearly see that for all the classifiers on all of the datasets, the number is higher for the 80/20 split. The averaged testing accuracy for linear SVM is increased by .1 percentage points on the *ADULT* dataset, by .2 percentage points on the *Car Evaluation* dataset, and by 3.1 percentage points on the *EEG to Eye Prediction* dataset. The averaged testing accuracy for k-NN is increased by .6 percentage points on the *ADULT* dataset, by 8 percentage points on the *Car Evaluation* dataset, and by 6.6 percentage points on the *EEG to Eye Prediction* dataset. The averaged testing accuracy for Random Forests is increased by .2 percentage points on the *ADULT* dataset, by 2.9 percentage points on

the *Car Evaluation* dataset, and by 2.1 percentage points on the *EEG to Eye Prediction* dataset. This shows that that the test accuracy increases with more training data and less test data.

4. Conclusions

One of the objects of this experiment was to confirm and replicate some of the findings of Caruna and Niculescu-Mizil [1], which I have successfully managed to do, in regard to ranking order of the algorithms. Off course, this experiment did not lead to the exact same results, as different measures were taken in the data preparation and preprocessing phase, resulting in different numbers of features and samples. Furthermore, the *ADULT* dataset was the only common dataset used, and different implementations of the algorithms were used. Limitations in computational power also resulted in this experiment having a smaller range of values used in tuning the hyperparameters, which is unfortunate.

Random Forests was ranked number one in the 20/80 split, the 80/20 split, and overall. It is fair to conclude that Random Forests was the best algorithm in this experiment. SVM was ranked second in the 20/80 split, third in the 80/20 split, and second overall. k-NN was ranked third in the 20/80 split, second in the 80/20 split, and third overall. The overall ranking order is consistent with that of Caruna and Niculescu-Mizil.

It was also shown in the experiment that without exceptions the averaged test accuracy for each classifier on each dataset was higher in the 80/20 split than the 20/80 splits. This leads to the conclusion that test accuracy increases with more training data and less test data.

References

[1] Caruana, R. and A. Niculescu-Mizil: *An empirical comparison of supervised learning algorithms*. 2006.

[2] Pedregosa, F., G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay: *Scikit-learn: Machine learning in Python*. Journal of Machine Learning Research, 12:2825–2830, 2011.

[3] Kohavi, R. and B. Becker: *UCI machine learning repository*, 1996. https://archive.ics.uci.edu/ml/datasets/Adult.

[4] Bohanec, M. and B. Zupan: *UCI machine learning repository*, 1997. https://archive.ics.uci.edu/ml/datasets/Car+Evaluation.

[5] Roesler, O.: *UCI machine learning repository*, 2013. https://archive.ics.uci.edu/ml/datasets/EEG+Eye+State.