Exploring the Efficacy of Machine Learning Algorithms Across Datasets and Training Sizes

Eamon "Boogie" Mikulec

Abstract

We test the efficacy of five machine learning algorithms (support vector machines, logistic regression, random forests, K-nearest neighbors, artificial neural networks) across diverse binary classification tasks using five UCI datasets. Employing a search with K-fold cross-validation, we optimized hyperparameters and analyzed model performance across three training/testing split ratios (20/80, 50/50, 80/20). Random forests and artificial neural networks emerged as the top performing models, with mean testing accuracies of 96.6%, while support vector machines and logistic regression lagged behind with mean testing accuracies of 91.8% and 92.2%, respectively. With varying training/testing splits, we are able to analyze how sensitivity models are to training set size, finding that random forests are the most sensitive to different training/testing splits, whereas logistic regression is the least sensitive. Finally, we analyze the optimal hyperparameters for each model and split combination, and find that optimal hyperparameters are generally consistent across training/testing splits, but vary across datasets.

1 Introduction

This is an investigation of the efficacy of various machine learning algorithms on a diverse selection of binary classification tasks drawn from the UCI Machine Learning Repository. Five algorithms representing distinct paradigms are employed: support vector machines (SVM), logistic regression (LOGREG), random forests (RF), K-nearest keighbors (KNN), and artificial neural networks (ANN). The performance of these algorithms is assessed across five datasets encompassing a range of complexities and characteristics: Iris (Fisher [1988]), Breast Cancer Wisconsin (BCW) (Wolberg and Street [1995]), Mushroom (mus [1987]), Letter Recognition (Slate [1991]), and Spambase (Hopkins and Suermondt [1999]).

Models are trained and assessed across three different training/testing ratios (20/80, 50/50, 80/20) to evaluate model performance across a range of training set sizes. For each split, we preform hyperparameter tuning, performing a search with 5-fold cross-validation to identify the parameter configurations maximizing validation accuracy within the training set. Models are assessed on three metrics: mean training accuracy and mean validation accuracy across the 5 folds, and test accuracy. This multifaceted approach sheds light on key aspects of algorithm performance: optimal hyperparameter settings, sensitivity to training/testing ratios, and suitability for specific datasets.

2 Method

2.1 Training/Testing Splits

To assess model performance across varying training set sizes and data variability, we used three different training/testing split ratios: 20/80, 50/50, and 80/20. This allows us to observe how models cope with data scarcity and abundance, proving insight into the suitability of each model for different datasets. For each split, we randomly partitioned the dataset into a training set and a test set, with the training set containing the specified percentage of the data.

2.2 Hyperparameter Tuning

To perform hyperparameter tuning, we combine K-fold cross-validation (K=5) with an exhaustive search of the hyperparameters. We search through a pre-defind set, performing K-fold cross-validation for configuration. We then select the hyperparameter configuration that maximizes the mean validation accuracy across the K folds.

2.3 Evaluation Metrics

We assess model performance across three metrics: mean training accuracy, mean validation accuracy, and test accuracy. Mean training accuracy is the average accuracy across the K folds of the training set. Mean validation accuracy is the average accuracy across the K folds of the validation set. Test accuracy is the accuracy of the model on the test set. These metrics allow us to compare the performance of each model, as well as evaluate how the hyperparameter selection generalizes to the training data.

2.4 Models

We employ five machine learning algorithms, each representing a different paradigm: SVM, LOGREG, RF, KNN, and ANN. All models are implemented using the scikit-learn library (Pedregosa et al. [2011]), and many of the pre-defind hyperparameters are adapted from Caruana and Niculescu-Mizil [2006].

Support Vector Machine (SVM): Our SVM leverages a linear kernel with square hinge loss. To find the optimal hyperparameters, we explore a range of C values: {0.005, 0.1, 1, 2, 5, 10, 100}. Additionally, the stopping criteria for the optimization process is based on a tolerance of 10^{-5}.

Logistic Regression (LOGREG): Our logistic regression model uses a solver based on Fan et al. [2008] if the size of the test set is less than 5000, and a solver based on Defazio et al. [2014] otherwise. We range the inverse of the regularization strength across {0.001, 0.01, 0.1, 1, 10, 100}. We also first pre-process the data using a robust scaler that ensures robustness to outliers, specifically scikit-learn's `RobustScaler`.

Random Forest (RF): We used a random forest with maximum number of features equal to the square root of the number of features in the dataset and without a maximum depth. We vary the number of estimators across the values {10, 100, 1000}.

K-Nearest Neighbors (KNN): We used a KNN classifier with Euclidean distance. We range the number of neighbors across 5 samples taken uniformly between 1 and the size of the training set.

Artificial Neural Network (ANN): We used a Multi-Layer Perceptron with ReLU activation. For smaller training sets (less than 5000), we use LBFGS, otherwise, we use an solver based on Kingma and Ba [2017]. We vary the learning rate across the values {0.0001, 0.001, 0.01}. Also, the data is pre-processed using the same robust scaler as LOGREG.

2.5 Datasets

We use a five datasets from the UCI Machine Learning Repository, spanning a range of domains and sizes. The Iris dataset (150 data points, 4 features) serves as a baseline, which has been converted to a binary classification problem by two of the three classes into one. Breast Cancer Wisconsin (BCW) (569 data points, 30 features) provides a medical diagnosis task, proving a relatively small dataset but still larger than Iris. Moving towards larger datasets, Mushroom (8124 data points, 22 features) features many categorical attributes, which we pre-processed with hot one encoding. Letter Recognition consists of 20,000 handwritten letters which are converted into 16 numerical attributes. Based on Caruana and Niculescu-Mizil [2006], we transformed this dataset into two separate tasks: classifying the letters A-M (LETTER.1) and only identifying the letter O (LETTER.2), giving us an imbalanced dataset. Finally, Spambase consists of 4601 email messages labeled as spam or not spam, with 57 features describing word counts and other characteristics. This spectrum of datasets, ranging from small and interpretable to large and complex, allows us to comprehensively evaluate the chosen machine learning models.

Table 1: Model performance across datasets and training/testing splits

	Iris			BCW			Mushroom			Spambase			Letter.1			Letter.2			Mean		
Model (Split)	**MA**	**VA**	**TA**	**MA**	**VA**	**TA**	**MA**	**VA**	**TA**	**MA**	**VA**	**TA**	**MA**	**VA**	**TA**	**MA**	**VA**	**TA**	**MA**	**VA**	**TA**
SVM (80/20)	.972	**.973**	**1.0**	.988	.965	.965	**1.0**	**.969**	**1.0**	.924	.905	.916	.962	.962	.965	.725	.724	.728	.928	.916	.929
SVM (50/50)	.972	**.973**	.920	.988	.965	.961	**1.0**	**.969**	**1.0**	.924	.905	.910	.962	.962	.963	.725	.724	.727	.928	.916	.913
SVM (20/80)	.972	**.973**	.942	.988	.965	.950	**1.0**	**.969**	**1.0**	.924	.905	.898	.962	.962	.963	.725	.724	.723	.928	.916	.912
LOGREG (80/20)	.977	.960	**1.0**	.989	**.979**	.965	**1.0**	.959	**1.0**	.931	.912	.910	.962	.962	.965	.726	.727	.729	.931	.917	.928
LOGREG (50/50)	.977	.960	.960	.989	**.979**	.968	**1.0**	.946	**1.0**	.931	.912	.917	.962	.962	.963	.726	.727	.727	.931	.914	.923
LOGREG (20/80)	.977	.960	.917	.989	**.979**	.971	**1.0**	.946	**1.0**	.931	.912	.915	.962	.962	.963	.726	.726	.724	.931	.914	.915
RF (80/20)	**1.0**	.967	**1.0**	.997	.963	.965	**1.0**	.955	**1.0**	**1.0**	**.931**	**.949**	**1.0**	.993	**.997**	**1.0**	**.975**	**.980**	**1.0**	**.964**	**.982**
RF (50/50)	**1.0**	.967	.947	**1.0**	.961	.940	**1.0**	.936	**1.0**	**1.0**	.929	.948	**1.0**	.993	.993	**1.0**	**.976**	.965	**1.0**	.960	.965
RF (20/80)	.998	.967	.917	**1.0**	.963	.932	**1.0**	.940	**1.0**	**1.0**	**.931**	.937	**1.0**	.993	.989	**1.0**	**.975**	.934	**1.0**	.961	.952
KNN (80/20)	**1.0**	.933	.900	**1.0**	.953	.930	**1.0**	.944	**1.0**	**1.0**	.867	.909	**1.0**	**.994**	**.996**	**1.0**	.970	.973	**1.0**	.944	.951
KNN (50/50)	**1.0**	.933	.880	**1.0**	.953	.937	**1.0**	.944	**1.0**	**1.0**	.867	.890	**1.0**	**.994**	.993	**1.0**	.970	.964	**1.0**	.944	.944
KNN (20/80)	**1.0**	.933	.858	**1.0**	.953	.945	**1.0**	.944	**1.0**	**1.0**	.867	.867	**1.0**	**.994**	.990	**1.0**	.970	.937	**1.0**	.944	.933
ANN (80/20)	**1.0**	.960	**1.0**	**1.0**	.963	**.974**	**1.0**	.868	**1.0**	.988	.910	.934	.996	.993	**.997**	.980	.963	.969	.994	.943	.979
ANN (50/50)	**1.0**	.953	.973	**1.0**	.967	.965	**1.0**	.944	**1.0**	.989	.907	.924	.996	.994	.991	.983	.964	.959	.995	.955	.969
ANN (20/80)	**1.0**	.953	.925	**1.0**	.965	.963	**1.0**	.946	**1.0**	**1.0**	.907	.913	**1.0**	**.995**	.989	**1.0**	.969	.928	**1.0**	.956	.953

MA = mean training accuracy, VA = mean validation accuracy, TA = test accuracy. Metric bolded if tied with or is the best on a dataset.

3 Experiment

3.1 Model Performance

We first analyze the performance of each model across the five datasets and three training/testing splits. Table 1 shows the mean training accuracy (MA), mean validation accuracy (VA), and test accuracy (TA) for each model and training/testing split. As these models are state-of-the-art, we expect them to perform well across all datasets. Indeed, all models achieve high accuracy, with the worst performing model, SVM with 20/80 split ratio, achieving 91.2% mean accuracy across all datasets.

The best performing model across all datasets is RF (80/20) with 98.2% testing accuracy. This aligns with the findings of Caruana and Niculescu-Mizil [2006] which found that RF outperformed other models on a range of datasets. However, averaging across all datasets and training/testing splits, the best performing model is a tie between RF and ANN, both with a mean of 96.6% testing accuracy. This is surprising, as Caruana and Niculescu-Mizil [2006] found that RF significantly outperformed ANN. This reflects the rapid progress in artificial neural networks in recent years, as the ANN solver used for large datasets (Kingma and Ba [2017]) was based on a paper published in 2017, while Caruana and Niculescu-Mizil [2006] was published in 2006.

Behind RF and ANN, the next best performing model is KNN, with a mean testing accuracy of 94.3%. This is followed by LOGREG (92.2%) and SVM (91.8%). KNN outperforming LOGREG and SVM aligns with the findings of Caruana and Niculescu-Mizil [2006]. However in their study, they found that LOGREG underperformed compared to the other models, while we found that it outperformed SVM. Similar to ANN, this may suggest an improvement in the performance of logistic regression in recent years, as the solvers used here were based on papers (Defazio et al. [2014], Fan et al. [2008]) published after the publication of Caruana and Niculescu-Mizil [2006].

3.2 Training Size Sensitivity

We analyze the sensitivity of each model to different training/testing splits to investigate how the amount of training data affects model performance. For each dataset and model, we calculate the standard deviation of the test accuracy across the three training/testing splits. Then, we average these standard deviations for each model across all datasets. This gives us a measure of the sensitivity of each model to different training/testing ratios and hopefully thus training size. Table 2 shows the results of this analysis.

We find that RF is the most sensitive to different training/testing splits, with a mean standard deviation of 0.016, followed by ANN (0.013). This is reflected in the results of Table 1, where RF regularly performs better with more training data. Specifically, RF achieves 98.2% testing accuracy with

Table 2: Standard deviation of test accuracy across training/testing splits.

Model	Iris	BCW	Mushroom	Spambase	Letter.1	Letter.2	Mean
SVM	0.041	0.008	0.000	0.010	0.001	0.003	0.010
LOGREG	0.042	0.003	0.000	0.004	0.001	0.002	0.009
RF	0.042	0.017	0.000	0.006	0.004	0.023	0.016
KNN	0.021	0.008	0.000	0.021	0.003	0.019	0.012
ANN	0.038	0.006	0.001	0.010	0.004	0.021	0.013

the 80/20 split compared to 95.2% with the 20/80 split. However, the model with the lowest mean standard deviation is LOGREG (0.009), followed by SVM (0.010). This suggests that these models are more robust to different training/testing splits, and are less sensitive to the amount of training data available.

3.3 Optimal Hyperparameters

We may gain insight into the optimal hyperparameters for each model by analyzing the hyperparameters selected by the grid search. Table 3 (next page) shows the hyperparameter with the highest mean validation accuracy across the five folds for each model and training/testing split combination. As each model varies only one hyperparameter (with the exception of ANN and LOGREG, varying their solver based on training set size, which we will ignore), we list the optimal value for that hyperparameter: inverse regularization strength for SVM and LOGREG, the number of estimators for RF, the number of neighbors for KNN, and the size of the hidden layer for ANN.

We find that the optimal hyperparameters are generally consistent across training/testing splits, with a few exceptions. For example, the optimal regularization strength LOGREG on Letter.2 is 10000 for the 50/50 split, but only 1 for the other splits. This suggests that the optimal hyperparameters for a model is mostly independent of the training/testing split, and thus the amount of training data.

On the other hand, the sensitivity of optimal hyperparameters to the dataset is more dependent on the model. For example, the optimal number of neighbors for KNN is 1 for all datasets, whereas the optimal regularization strength for SVM varies across datasets. Although, there does seem to be some trends. For example, the ANN seems to prefer larger hidden layers for more complex datasets, with Iris being the only dataset where 8 hidden units was optimal. This suggests that the optimal hyperparameters for a model are dependent on the dataset, and thus the complexity of the task.

4 Conclusion

We have investigated the performance of five machine learning algorithms across five datasets and three training/testing splits, with hyperparameter tuning. This has given us insight to model performance across a range of datasets, training set sizes, as well as identify trends in hyperparameter optimality and sensitivity to training/testing splits.

Sensitivity to training data varied slightly across models. RF emerged as the most susceptible to different training/testing splits, with performance fluctuating by 3% between the 20/80 and 80/20 splits. In contrast, logistic regression displayed remarkable stability, suggesting its effectiveness remains relatively constant regardless of data volume.

Diving deeper, we analyzed hyperparameter optimality across splits and datasets. Interestingly, optimal settings displayed consistency within splits but diverged across datasets. This suggests model-specific hyperparameters are primarily influenced by task complexity (dataset) rather than data volume, which aligns with our intuition.

We find that all models achieve high accuracy across all datasets, with the best performing model being a tie between RF and ANN. This reflects the rapid progress in artificial neural networks in recent years, as Caruana and Niculescu-Mizil [2006] found that RF significantly outperformed ANN. As a whole, there seemed to be three tiers of performance, with RF and ANN at the top, KNN in the middle, and SVM and LOGREG at the bottom.

Table 3: Optimal hyperparameters for each model and split combination.

Dataset (Split)	SVM	LOGREG	RF	KNN	ANN
Spambase (80/20)	0.005	1	1000	1	8
Spambase (50/50)	0.005	1	1000	1	8
Spambase (20/80)	0.005	1	100	1	256
Iris (80/20)	5	10000	1000	1	256
Iris (50/50)	5	10000	1000	1	128
Iris (20/80)	5	10000	100	1	128
BCW (80/20)	100	1	10	1	128
BCW (50/50)	100	1	1000	1	128
BCW (20/80)	100	1	100	1	256
Mushroom (80/20)	100	10000	10	1	256
Mushroom (50/50)	100	100	10	1	256
Mushroom (20/80)	100	100	10	1	128
Letter.1 (80/20)	0.005	0.0001	1000	1	256
Letter.1 (50/50)	0.005	0.0001	1000	1	256
Letter.1 (20/80)	0.005	0.01	1000	1	128
Letter.2 (80/20)	10	1	1000	1	256
Letter.2 (50/50)	10	10000	1000	1	256
Letter.2 (20/80)	10	1	1000	1	256

Hyperparameter shown: inverse regularization strength for SVM and LOGREG, the number of estimators for RF, the number of neighbors for KNN, and the size of the hidden layer for ANN.

References

R. A. Fisher. Iris. UCI Machine Learning Repository, 1988. DOI: https://doi.org/10.24432/C56C76.

Mangasarian Olvi Street Nick Wolberg, William and W. Street. Breast Cancer Wisconsin (Diagnostic). UCI Machine Learning Repository, 1995. DOI: https://doi.org/10.24432/C5DW2B.

Mushroom. UCI Machine Learning Repository, 1987. DOI: https://doi.org/10.24432/C5959T.

David Slate. Letter Recognition. UCI Machine Learning Repository, 1991. DOI: https://doi.org/10.24432/C5ZP40.

Reeber Erik Forman George Hopkins, Mark and Jaap Suermondt. Spambase. UCI Machine Learning Repository, 1999. DOI: https://doi.org/10.24432/C53G6X.

F. Pedregosa, 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.

Rich Caruana and Alexandru Niculescu-Mizil. An empirical comparison of supervised learning algorithms. page 161–168, 2006. doi: 10.1145/1143844.1143865. URL `https://doi.org/10.1145/1143844.1143865`.

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. Liblinear: A library for large linear classification. *Journal of Machine Learning Research*, 9(61):1871–1874, 2008. URL `http://jmlr.org/papers/v9/fan08a.html`.

Aaron Defazio, Francis Bach, and Simon Lacoste-Julien. Saga: A fast incremental gradient method with support for non-strongly convex composite objectives, 2014.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.