Abstract:

This report presents an empirical comparison between the performance of multiple supervised learning methods: Boosted Decision Trees (BST-DT), K-Nearest Neighbours (KNN), and Logistic Regression (LR). The comparison is conducted on three binary classification problems using datasets from the UCI Machine Learning Repository, and the performance of each classifier is evaluated based on accuracy.

1. Introduction:

The main motivation and reference for this report is the paper “An Empirical Comparison of Supervised Learning Algorithms” by Caruana and Niculescu-Mizil, who examined and compared the performance of many supervised learning classifiers across a variety of datasets; and this report seeks to reinforce and confirm their results for a subset of the classifiers used in the original paper. This report will focus on the two-class classification problem, and accuracy will be the metric used to judge the performance of each classifier.

2. Method:

2.1. Learning Algorithms:

This section introduces the algorithms used, as well as the parameters used for each learning algorithm.

Logistic Regression (LR): I used the scikit-learn LogisticRegression classifier, with the Limited-memory BFGS optimization algorithm and L2 penalty. During cross-validation, the hyperparameter C was varied over the values of {0.01, 0.1, 1, 10}; and the maximum iterations varied over the values of {100, 200, 300}.

K-Nearest Neighbours (KNN): I used the scikit-learn KNeighborsClassifier, with uniform weight, the ‘auto’ algorithm used to compute the nearest neighbours, a leaf size of 30, a power parameter of 2, and the minkowski metric for distance computation. During cross-validation, the hyperparameter n_neighbours varied over the values of {5, 10, 15}.

Boosted Decision Trees (BST-DT): I used the scikit-learn HistGradientBoostingClassifier, which is a histogram-based gradient boosting classification tree classifier. I used the logistic loss function, a maximum of 31 leaf nodes. During cross-validation, the hyperparameter learning_rate varied over the values of {0.1, 0.2, 0.3}, and the maximum iterations varied over the values of {100, 200, 300}.

2.2. Performance Metrics:

The metric used to evaluate performance of each classifier will be the mean accuracy, which has a range between [0, 1].

2.3 Dataset and Problem Description:

I compare the 3 classifiers on 3 binary classification problems. The datasets of ADULT (Becker et al., 1996), BANK_MARKETING (Moro et al., 2012), and DRY_BEAN (Dry Bean Dataset, 2020) are from the UCI Machine Learning Repository (http://archive.ics.uci.edu/).

The ADULT dataset's categorical features have been encoded using one-hot encoding, and the target value of "<=50K" has been mapped to positive. A similar process was done to the BANK dataset, where the target value of "yes" was mapped to positive. DRY_BEAN has been converted into a binary classification problem by treating the largest class ("DERMASON") as positive and the other classes as negative. The other two datasets are originally binary classification problems. See Table 1 for characteristics of these problems. The last column, %Positive, represents the percentage of positive target values in each dataset after data processing.

Table 1: Problem description

Problem	#Features	Train Size	Test Size	%Positive
ADULT	14	5000	40211	76%
BANK	16	5000	43842	88%
DRY_BEAN	16	5000	8611	26%

2.4 Experimental Design:

For each dataset, 5000 instances are randomly selected for training, and the rest of the cases are set aside to be used for the final test set. Within the 5000 instances, different partitions are attempted to further split the data into a training set and a validation set, where the training set is used to train the models and the validation set is used to select the best hyperparameters.

The partitions used are 20/80, 50/50, and 80/20, where the first number represents the percentage of data allocated to the training set, and the latter number represents the percentage of data allocated to the validation set. Training with each partition is repeated three times, where the training set is used to train the models, and the validation set is used to select the best hyperparameters using cross-validation.

The training and validation accuracy are averaged over 3 trials to determine the best hyperparameters, which are then tested on the large remaining test set. The following sections show the classifiers' performance on each partition type averaged over each problem, as well as the classifiers' performance on each dataset averaged over each partition, with the raw data fully available in Appendix A.

3. Performance by Partition

Table 2 shows the test accuracy for each classifier. For each problem, the best hyperparameters are found using the validation sets set aside by cross-validation, and the accuracies represent the mean accuracies over the three problems where the model is fit on the large remaining test set. In the table, the partition with the best testing accuracy within each partition for each classifier is **boldfaced**.

Table 2: Mean accuracies separated by partition

Classifier	Train/Test Split	Mean Train Accuracy	Mean Validation Accuracy	Mean Test Accuracy
LR	20/80	0.8766	0.8790	0.8729
	50/50	0.8851	0.8819	0.8808
	80/20	0.8841	0.8847	**0.8812**
KNN	20/80	0.9160	0.9028	0.9003
	50/50	0.9135	0.8990	0.9033
	80/20	0.9370	0.9058	**0.9062**
BST-DT	20/80	0.9998	0.8999	**0.9050**
	50/50	0.9989	0.8953	0.9040
	80/20	0.9803	0.9012	0.9040

One trend that can be observed is that the model performance does increase when more data is being used for training, where the mean test accuracy is the highest for the 80/20 split for two of the classifiers, with the BST-DT classifier being the odd one out. One possible explanation for this is that the classifier has hit an accuracy ceiling, where the accuracy does not increase further even when it is supplied with more training data, and this is supported by the relatively small difference between partitions, as well as the fact that the BST-DT test accuracy does not increase between the 50/50 and the 80/20 split.

4. Performance by Problem

Table 3 shows the mean test accuracy for each classifier averaged over the different types of partitions for each test problem. At a glance, it seems that DRY_BEAN is a relatively easier problem as compared to the other two datasets, as all classifiers were able to achieve a higher score compared to their accuracy in the other problems. In the table, the best testing accuracy within each problem is **boldfaced**.

Table 3: Mean test accuracies separated by problem

Classifier	ADULT	BANK	DRY_BEAN
LR	0.7975	0.8930	0.9444
KNN	**0.9042**	**0.9048**	0.9008
BST-DT	0.8580	0.9003	**0.9547**

A surprising finding when comparing the mean test accuracy over partitions is the performance of BST-DT not being the highest among the classifiers, although this may be due to two reasons: BST-DT being more susceptible to overfitting, (more clearly shown in Appendix A), and therefore to changes in the types of partitions, making the averaged test accuracy not reflective of its performance; and other classifiers responding to certain problems better.
The performance of the LR classifier also surprised me with how high it was, but it is also likely that the algorithm used in the implementation by scikit-learn is different from the one in the original paper by Caruana and Niculescu-Mizil, leading to much better performance than demonstrated in their paper.

5. Discussion and Conclusion

The results from this experiment were relatively surprising, as the performance of the Logistic Regression and K-Nearest Neighbours were much higher than I expected, when compared to the results obtained by Caruana and Niculescu-Mizil in their original paper.
I suspect this is mainly due to a difference in the implementation of the algorithms, as the scikit-learn library is much more recent compared to the 2006 paper.
Boosted Decision Trees were concluded to be the best classifier in the original paper, but it did not have as much of a lead in this experiment. A possible explanation for this is overfitting, as the classifier consistently hits a training accuracy very close to 1 even with very few iterations of training. (Limiting the maximum number of iterations below 100 does not solve this problem either.)

A phenomenon that does mirror the findings in the paper is the variability of classifier performance across datasets. The BST-DT classifier in particular performs badly on the ADULT dataset, which replicates the same finding in the original paper.
In conclusion, the BST-DT classifier achieved the highest single accuracy out of all classifiers in a particular partition, but in general, its performance varies quite a lot depending on the dataset. KNN performed quite well, and is the most consistent across different datasets. LR is the worst performing out of the three, but can still score relatively high depending on the dataset.

6. References

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

Caruana, R., & Niculescu-Mizil, A. (2006). An empirical comparison of supervised learning algorithms. Proceedings of the 23rd International Conference on Machine Learning - ICML ’06. https://doi.org/10.1145/1143844.1143865

Dry Bean Dataset. (2020). UCI Machine Learning Repository. https://doi.org/10.24432/C50S4B.

Moro,S., Rita,P., and Cortez,P.. (2012). Bank Marketing. UCI Machine Learning Repository. https://doi.org/10.24432/C5K306.

Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011.

Appendix A: Raw Data

ADULT:

Classifier	Train/Test Split	Training Accuracy	Validation Accuracy	Testing Accuracy	Hyperparameters
LR	[0.2, 0.8]	0.8130	0.8122	0.7973	{'C': 10, 'max_iter': 200}
	[0.5, 0.5]	0.8080	0.8095	0.7960	{'C': 1, 'max_iter': 200}
	[0.8, 0.2]	0.8108	0.8203	0.7992	{'C': 10, 'max_iter': 200}
KNN	[0.2, 0.8]	0.9143	0.9032	0.9001	{'n_neighbors': 15}
	[0.5, 0.5]	0.9127	0.8960	0.9066	{'n_neighbors': 10}
	[0.8, 0.2]	0.9354	0.9010	0.9058	{'n_neighbors': 5}
BST-DT	[0.2, 0.8]	0.9993	0.8452	0.8589	{'learning_rate': 0.1, 'max_iter': 100}
	[0.5, 0.5]	0.9969	0.8369	0.8565	{'learning_rate': 0.2, 'max_iter': 100}
	[0.8, 0.2]	0.9411	0.8520	0.8587	{'learning_rate': 0.1, 'max_iter': 100}

BANK:

Classifier	Train/Test Split	Training Accuracy	Validation Accuracy	Testing Accuracy	Hyperparameters
LR	[0.2, 0.8]	0.8783	0.8856	0.8925	{'C': 10, 'max_iter': 200}
	[0.5, 0.5]	0.9000	0.8929	0.8947	{'C': 1, 'max_iter': 300}
	[0.8, 0.2]	0.8868	0.8867	0.8918	{'C': 1, 'max_iter': 300}
KNN	[0.2, 0.8]	0.9143	0.8983	0.9036	{'n_neighbors': 15}
	[0.5, 0.5]	0.9137	0.9017	0.9029	{'n_neighbors': 10}
	[0.8, 0.2]	0.9387	0.9163	0.9078	{'n_neighbors': 5}
BST-DT	[0.2, 0.8]	1.0000	0.9018	0.9015	{'learning_rate': 0.1, 'max_iter': 100}
	[0.5, 0.5]	0.9999	0.8955	0.9005	{'learning_rate': 0.1, 'max_iter': 100}
	[0.8, 0.2]	0.9998	0.8997	0.8988	{'learning_rate': 0.2, 'max_iter': 100}

DRY_BEAN:

Classifier	Train/Test Split	Training Accuracy	Validation Accuracy	Testing Accuracy	Hyperparameters
LR	[0.2, 0.8]	0.9383	0.9393	0.9288	{'C': 1, 'max_iter': 200}
	[0.5, 0.5]	0.9472	0.9432	0.9518	{'C': 1, 'max_iter': 200}
	[0.8, 0.2]	0.9546	0.9470	0.9526	{'C': 1, 'max_iter': 200}
KNN	[0.2, 0.8]	0.9193	0.9071	0.8971	{'n_neighbors': 15}
	[0.5, 0.5]	0.914	0.8993	0.9002	{'n_neighbors': 15}
	[0.8, 0.2]	0.9368	0.9000	0.9051	{'n_neighbors': 5}
BST-DT	[0.2, 0.8]	1.0000	0.9529	0.9547	{'learning_rate': 0.1, 'max_iter': 100}
	[0.5, 0.5]	1.0000	0.9535	0.9552	{'learning_rate': 0.1, 'max_iter': 200}
	[0.8, 0.2]	1.0000	0.9520	0.9544	{'learning_rate': 0.2, 'max_iter': 200}

Appendix B: Training Code

https://github.com/theTY2002/COGS-118A-Final-Project