CS 4641: Supervised Learning

Bradley Reardon

February 10, 2019

1 Introduction

The purpose of this assignment is to evaluate various supervised learning techniques in the context of two classification algorithms. We will focus on analyzing and adjusting five learning algorithms using two publically-available data sets, through cross-validation and hyperparameter adjustments. Then, we can use our adjusted models to compare supervised learning techniques in the context of each of our classification algorithms.

1.1 Collaboration disclosure

I collaborated on the programming portion of this assignment with Nikolai Vorobiev in order to produce a code-base capable of using scikit-learn to effectively evaluate the given data sets. Though the code we used to run analysis on the algorithms is the same, this analysis is wholly my own.

2 Classification algorithms

For the purposes of this report, data sets were obtained from the UCI Machine Learning Repository. Each data set downloaded was processed with a custom script in the corresponding folder, **process.py**, which randomly separated the data into an 80% training data and 20% test data split using the **train_test_split** function from scikit-learn. The split data was then serialized using Python's **pickle** module into **.dataset** files, to ensure that the training and test sets remained constant for evaluation.

For the purposes of this assignment, cross-validation scoring of the various algorithms for each classification problem will be shown as the average score of 5-fold cross validation.

2.1 Car data set

The Car Evaluation Database was created in June, 1997 by Marko Bohanec. It contains 1728 instances and six attributes. The purpose of this database is to attempt to classify a car as an unacceptable, acceptable, good, or very good choice based on factors including cost of ownership, comfort, and safety. Full details about the data set can be found at the source link below.

The problem at hand for this dataset is determining the acceptability for a car purchase based on the aforementioned attributes. Because this dataset notes that the instances in the data set completely cover the attribute space, this data set is interesting in particular due to its usefulness in comparing the optimization of different supervised learning algorithms. Note that this specific data set contains only categorical attributes. As scikit-learn does not support non-continuous variables, a one-hot encoder was used to re-encode categorical features into multiple dimensions.

Source: https://archive.ics.uci.edu/ml/datasets/car+evaluation

2.2 Breast Cancer Wisconsin data set

The Breast Cancer Wisconsin data set was donated to the UCI Machine Learning Repository in 1992, and contains data from one doctor's clinical cases, gathered from January 1989 to November 1991. In total, there are 699 instances signifying information about breast tumors such as clump thickness, uniformity in shape and size, and other screening factors. Data points are identified by two classes – benign or malignant. The features of the data points are encoded as 9 continuous attributes rating the screening factor from 1 to 10.

This data set contains unknowns in the form of question marks in the data. To deal with this, missing values were imputed, calculating missing data points based on the mean of other points in the specific column of the missing attribute.

The problem at hand for this dataset is determining whether a tumor is benign or malignant based on tumor screening characteristics identified in the data set.

Source: https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+%28Original%29

3 Decision trees with pruning

3.1 Parameter selection

To begin evaluating the decision tree algorithm, hyperparameter selection was essential to tuning the algorithms to each data set. Both the car and cancer datasets were evaluated with the GINI index criterion as well as the entropy criterion. The two criterion had no discernible differences in training results, so as a result, GINI was selected for the decision tree classifier on both sets, as it allows us to avoid a logarithmic calculation, saving compute time.

The next parameters that needed tuning were the pruning parameters included in the scikit-learn decision tree classifier. Namely, the max depth of the tree, as well as the minimum samples per leaf, were tuned to fit each data set.

For the car data set, parameter selection showed that a lower-depth tree had much lower accuracy, likely due to the fact that the dataset had instances covering the entire attribute space. As a result, a max depth of 9 nodes was selected for the car data set, with a minimum number of 5 samples occurring at each leaf node to minimize the amount of information loss.

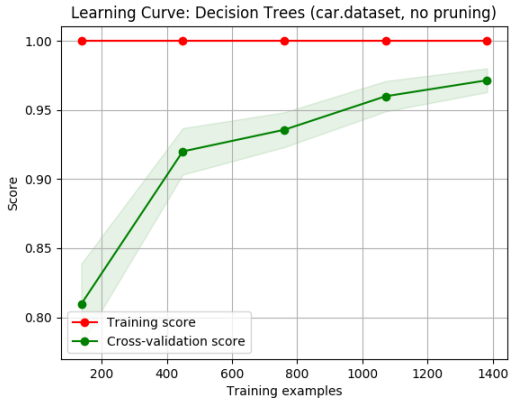
The cancer data set, on the other hand, responded much better to more aggressive pruning. The pruned decision tree classifier responded to the data with the best F1 score with a max depth of 5 nodes, and a minimum number of 10 samples occurring at each leaf.

3.2 Performance

Performance of the tuned classifier for each data set was measured with both wall clock time for fitting the model with the training data, as well as making a prediction with the test data. For the breast cancer data set, pruning resulted in both higher precision and accuracy, with a performance gain during prediction over a non-pruned decision tree (from 0.801ms to 0.479ms). However, the classifier took 0.57ms longer to train with pruning enabled (no pruning: 2.38ms, pruning: 2.95ms). For this data set, with the determined parameters was successful.

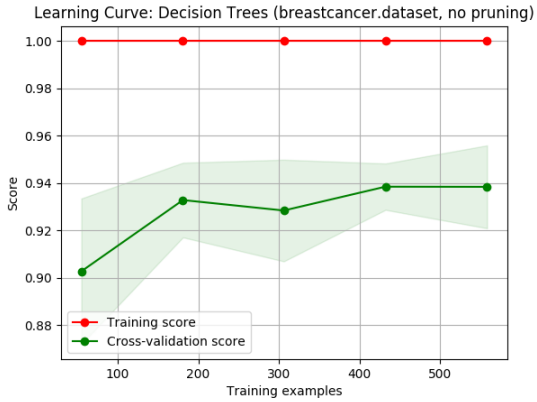
However, the car data set showed different results with pruning enabled. There was no significant change in either the precision or accuracy measures. The car data set's decision tree classifier took 11.51ms to fit without pruning, however with pruning this was reduced to 7.35ms with pruning. Prediction runtime for the pruned tree was only slightly longer than the unpruned tree. Due to the completeness of the data, pruning is likely not necessary for this dataset without further tuning.

Learning curves for the data sets, with and without pruning enabled, can be found in Figure 1.

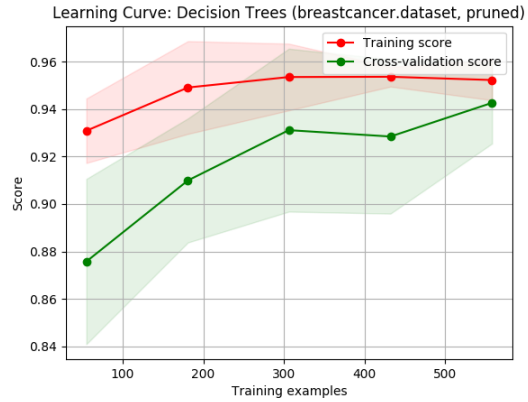


(a) Car data set, no pruning

(b) Car data set, with pruning



(c) Breast cancer data set, no pruning



(d) Breast cancer data set, with pruning

Figure 1: Learning curves for the car and breast cancer data sets using a decision tree classifier, with and without pruning.

4 Boosting

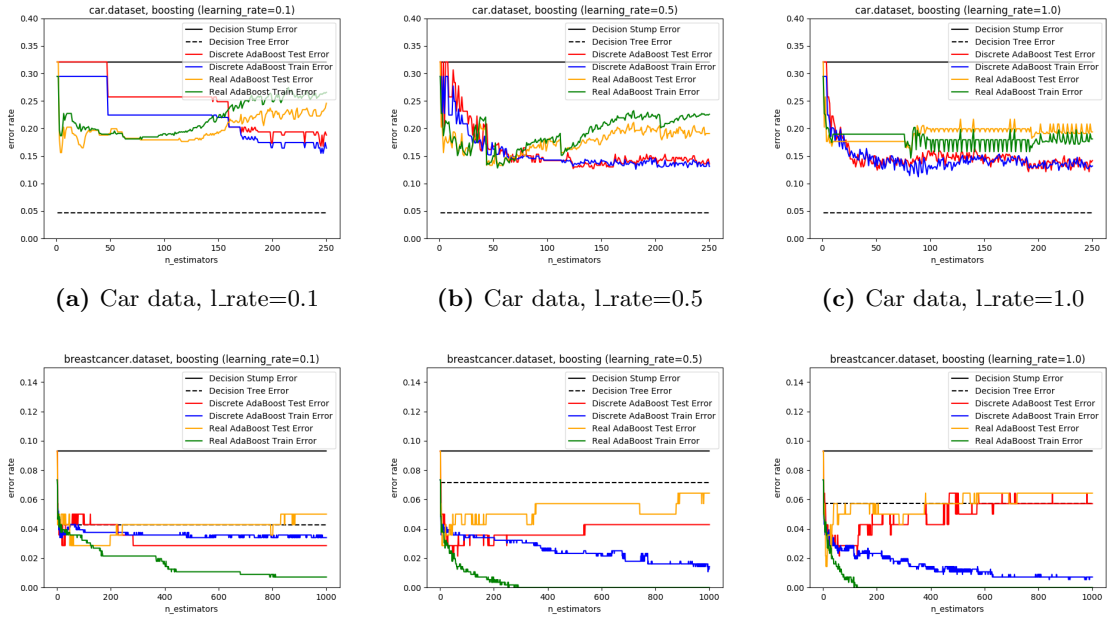
We will next evaluate the AdaBoost boosting algorithm included in scikit-learn, using the pruned decision trees from the previous section as base estimators for the algorithm.

4.1 Parameter selection

For the AdaBoost algorithm, there were three main parameters that needed to be determined. The graphs in Figure 2 were used to determine the proper learning rate, number of estimators, and algorithm for both data sets.

- **n_estimators:** the max number of classifiers (decision trees) to fit to the AdaBoost classifier
- **learning_rate:** shrinks the contribution of each classifier by this amount
- **algorithm:** scikit-learn supports two algorithms derived from AdaBoost – AdaBoost-SAMME and AdaBoost-SAMME.R. SAMME is usually used for discrete values, whereas SAMME.R is used for real/continuous values.

The graphs in Figure 2 show that the car data generally has lower error with the SAMME algorithm, whereas the breast cancer data leads to lower error when fit with the SAMME.R algorithm. For both sets, the error was generally minimized with a learning rate of 1.0, so that rate was selected for both classifiers.



(a) Car data, l_rate=0.1 **(b)** Car data, l_rate=0.5 **(c)** Car data, l_rate=1.0

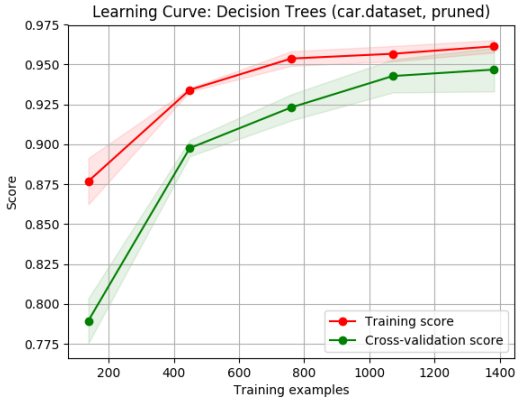
(d) Cancer data, l_rate=0.1 **(e)** Cancer data, l_rate=0.5 **(f)** Cancer data, l_rate=1.0

Figure 2: Selecting a learning rate and number of estimators for the AdaBoot algorithm, using the car and breast cancer sets.

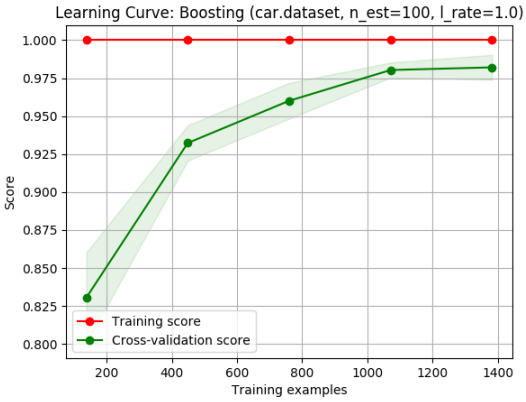
The number of estimators was also determined from the error graphs in Figure 2 on the previous page. The lowest number of estimators where error for the chosen algorithm was minimized was chosen as the number of estimators for each data set. For the car data set, this was approximately 100 estimators, while the cancer data set performed best with around 200 estimators.

4.2 Performance

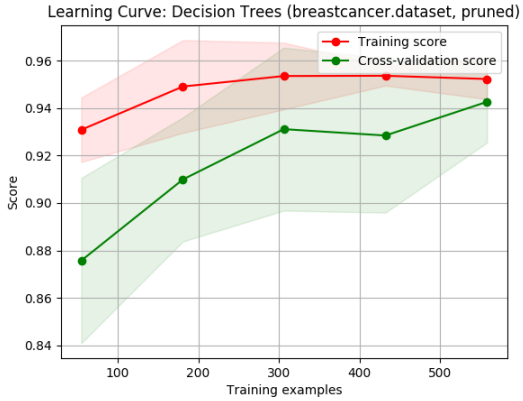
Figure 3 shows the learning curves for the car and data set pruned decision tree classifiers before and after applying boosting. Boosting provided a tangible benefit in accuracy and precision for both data sets after proper parameter selection. With the chosen parameters, the car data set and the cancer data set both showed better cross-validation scores with boosting than over the standard pruned decision tree classifiers in the previous section.



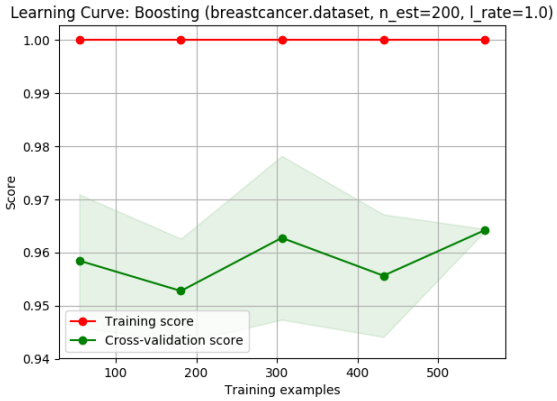
(a) Car data, pruning, no boosting



(b) Car data, pruning, with boosting



(c) Breast cancer data, pruning, no boosting



(d) Breast cancer data, pruning, with boosting

Figure 3: Learning curves for the car and cancer data sets, with and without boosting on a pruned decision tree.

However, this increase in accuracy and precision for both data sets does come at the cost of extra wall clock time during both the fit and predict phases of testing. Due to the nature of the AdaBoost algorithm, fit and prediction times increase with some linearity to the number of estimators used to

fit the classifier. This increase in time represented approximate 70-fold and 61-fold increases in fit times for the cancer and car data sets respectively. Similar slow downs resulted in prediction using the boosting algorithm over a standard pruned decision tree.

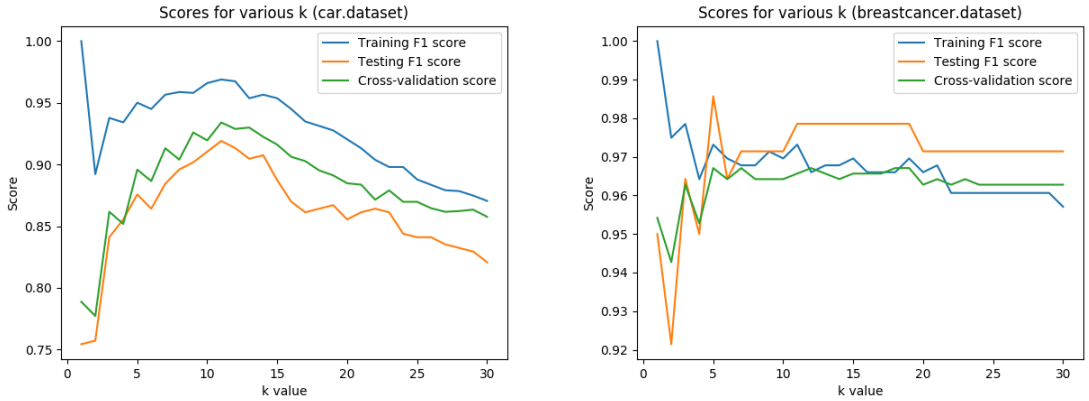
Therefore, boosting does increase the accuracy and precision for both models, however at the cost of slowing down fit and classification times across the board. This tradeoff will be discussed further in a later section.

5 k-nearest neighbors

Next, the k-nearest neighbors algorithm was tuned and evaluated for performance on the two data sets.

5.1 Parameter selection

The main parameter adjustment for the k-nearest neighbors algorithm was setting a sensible value for k such that cross-validation score is maximized. Figure 4 shows the performance of the algorithm for each data set, with different values of k.



(a) Car data set

(b) Breast cancer data set

Figure 4: Testing k-nearest neighbors with various values of k.

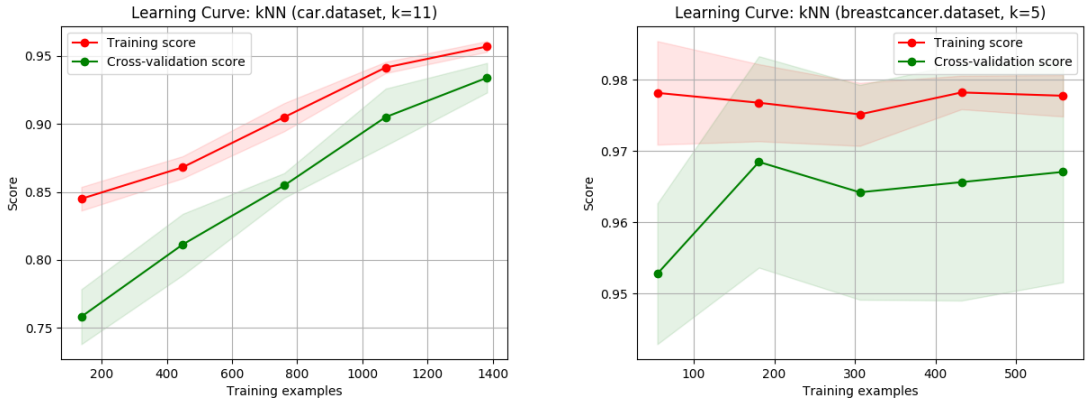
Cross-validation scores were best for the car data set with a value of $k = 11$, whereas the cancer data performed best with $k = 5$. Therefore, these values were selected as the optimal parameters of the data sets.

5.2 Performance

The k-nearest neighbors algorithm performed better for the breast cancer data set than the car data set with optimized values of k. This is likely due to the fact that the breast cancer data contains only continuous attributes, while the car data set contains only nominal data. The testing F1 scores for the cancer and car data sets using this algorithm were 98.57% and 91.91% respectively.

In terms of wall clock performance, k-nearest neighbors was resonably fast to fit the classifier to both sets, doing so in 4.97ms for the car data set and 2.17ms for the cancer data set. However, prediction time for the car data set was significantly slower than other algorithms at 23.84ms, whereas the cancer data set also had a relatively slow prediction runtime of 5.33ms.

Figure 5 shows the learning curves for the two data sets using k-nearest neighbors with optimized k.



(a) Car data set, $k = 11$

(b) Breast cancer data set, $k = 5$

Figure 5: Learning curves for k-nearest neighbors with optimized k-values.

6 Support vector machines

Support vector machines (SVMs) are the next algorithm that will be evaluated. The car and cancer data sets were both evaluated using the linear, RBF, and polynomial kernels included in scikit-learn. For the sake of brevity, the three algorithms were evaluated with otherwise default kernel parameters.

6.1 Kernel selection

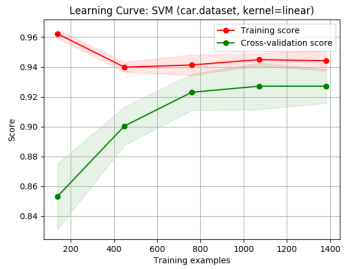
Kernel selection is important for ensuring that the support vector machine is able to find the shape of the feature boundary within the attribute space. To evaluate the merits of the linear, RBF, and polynomial kernels, a learning curve was generated for each data set with each of the three kernels.

It is immediately obvious that the polynomial kernel does not sufficiently classify the data for the car data set. This, along with the RBF kernel's reduced performance on the car data set with respect to the linear kernel, suggests that the car data set is linearly separable. Thus, the ideal kernel for the car data set is chosen to be the linear kernel.

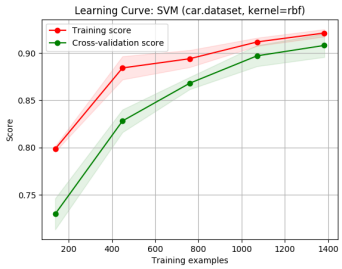
However, for the cancer data set, all three kernels report decent scores. The linear kernel again performs best, though the RBF kernel is close behind. The polynomial kernel also has above-90% cross-validation score. These results suggest that, with sufficient kernel parameter tuning, a SVM trained on the cancer data set may have better accuracy than the linear kernel than with the RBF

or polynomial kernels. However, for the purpose of the assignment, we select the linear kernel for use with the cancer data as well.

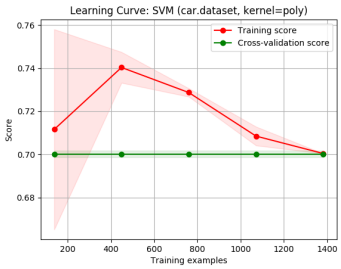
The learning curves used for this evaluation can be found in Figure 6.



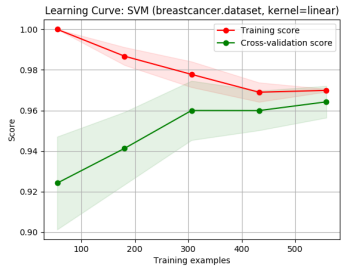
(a) Car data, linear kernel



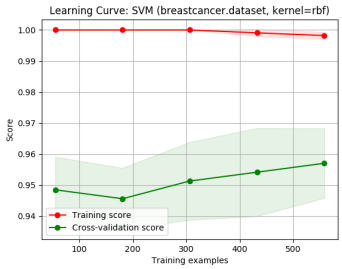
(b) Car data, RBF kernel



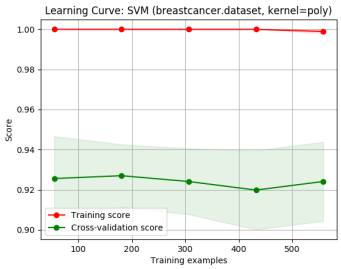
(c) Car data, poly kernel



(d) Breast cancer, linear kernel



(e) Breast cancer, RBF kernel



(f) Breast cancer, poly kernel

Figure 6: Learning curves for the car and breast cancer data sets using an SVM classifier, with linear, RBF, and poly kernels.

6.2 Performance

The support vector machine for the cancer data set performed well with the linear kernel. Wall clock time to fit the classifier was only 6.25ms, with a tested prediction runtime of 0.57ms. The F1 score reported for the test data was 96.43%, indicating good accurary for the model.

On the other hand, the SVM trained with the car data set and the linear kernel had good, but not excellent results in terms of both accurary and wall clock time. The test F1 score was 93.06%, however the model took 40.81ms to fit, with a less-than-impressive prediction runtime of 6.76ms. In terms of accuracy, the car data set performed well with a linear kernel, however the wall clock time for using SVM for this data set leaves this algorithm as a less-than-desirable choice for real-world use.

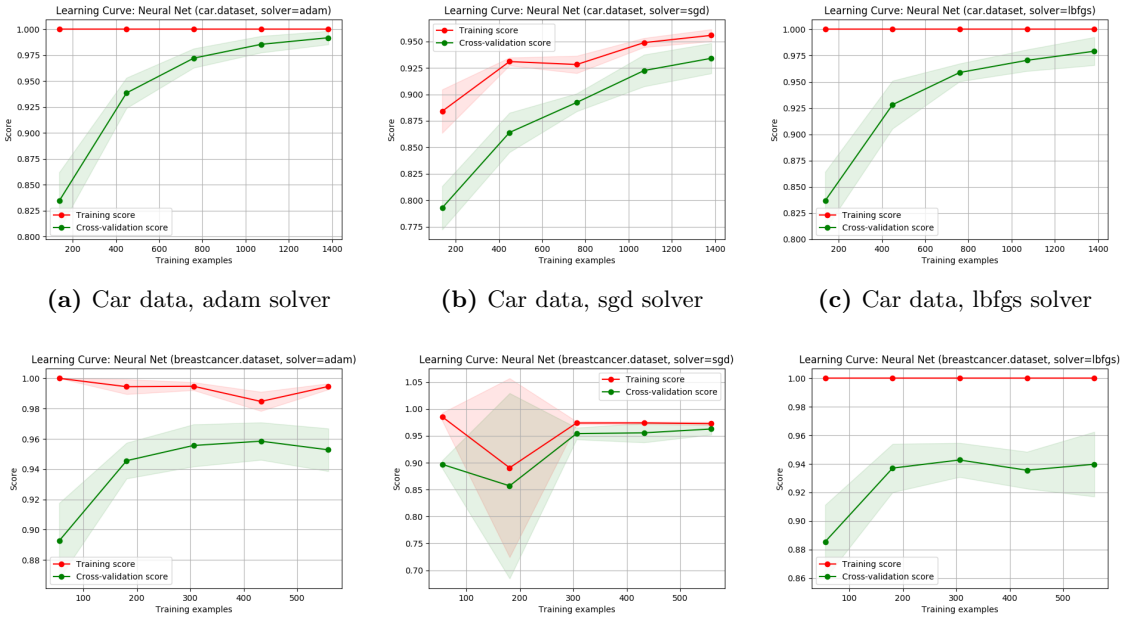
7 Neural networks

Finally, we evaluate scikit-learn's built in multi-layer perceptron neural network algorithm for classifying our two data sets.

7.1 Solver selection

Because neural network generation is computationally intensive, parameter selection for the neural network algorithm was restricted to selecting the best solver for weight optimization. The three options that scikit-learn provides are the 'adam' solver, the 'sgd' (or stochastic gradient descent) solver, and the 'lbfgs' solver. Neural networks were allowed 1,000 total iterations each to ensure the highest possibility of convergence.

The learning curves used for this evaluation can be found in Figure 7.



(a) Car data, adam solver **(b)** Car data, sgd solver **(c)** Car data, lbfgs solver

(d) Breast cancer, adam solver **(e)** Breast cancer, sgd solver **(f)** Breast cancer, lbfgs solver

Figure 7: Learning curves for the car and breast cancer data sets using a MLP neural network, with adam, sgd, and lbfgs solvers.

Cursory analysis of the learning curves for the different solvers shows that, for the car data set, the 'adam' and 'lbfgs' solvers perform very well with default parameters. On a single run, 'lbfgs' performed better than the 'adam' solver, however, due to the lower variance in cross-validation score of the 'adam' solver, we'll further discuss this implementation using only the 'adam' solver.

The breast cancer data set, however, showed the best accurary and lowest variance cross-validation score when using the 'sgd' solver. Thus, further analysis will occur using the 'sgd' solver on the breast cancer data set.

Note that stochastic gradient descent is heavily tunable via hyperparameters, however in the interest of time, the defaults were used in all cases to get a good overview of algorithmic performance out of the box.

7.2 Performance

TODO

8 Analysis

TODO

9 Discussion

TODO

10 Conclusion

TODO