SUPERVISED LEARNING ANALYSIS

Shannon Ke
CS 4641
Supervised Learning Report

Classification Problems

The Nursery Dataset

The first data set, the nursery database, was constructed originally to assist in ranking and ultimately giving explanations for accepting or denying applicants from nursing school and found through the UCI machine learning repositories. The data was derived from a hierarchical decision model that was used in the 1980s when applications to certain nursing schools was particularly high. The dataset has eight attribute values and five possible classifications with 12960 instances of data. The combinations of types of attribute values makes this an interesting learning problem since personally, there was no readily obvious connection between sets of attribute values and target values. I was curious to see how each learning model would attempt to classify each instance into its corresponding class.

The Car Dataset

The second data set, the car database, was also found through the UCI machine learning repositories. The attribute values, of which there are six and include buying price, maintenance price, number of doors, etc. , determine the acceptability of the car in question. There are four values for acceptability: unacceptable, acceptable, good, and very good. This dataset is interesting as it takes a very practical problem that would be useful in many cases for normal people looking to buy a car or looking for car recommendations and turns it into a machine learning problem that could be implemented as an aid for many.

General Methodology

I pulled my datasets from the CSV files and encoded them using scikit's label encoder in order to transform string feature values into readable integers. Using scikit's train_test_split method, for graph analysis involving changing dataset sizes, I randomly split the data into training and test data. For graph analysis involving changing hyperparameters, I kept the data split (training data/test data) for the nursery dataset at (70/30) given that the dataset has a large number of samples, and the car dataset at (90/10) given that the dataset has a smaller number of samples.

I ran my datasets through all five algorithms twice, one with changing training dataset size and one with a chosen changing optimal hyperparameter. For decision trees, I chose max depth. For neural networks, I chose number of hidden layers. For boosting, I chose number of

boosting stages. For k-Nearest Neighbors, I chose number of neighbors. And finally, for support vector machines, I chose the complexity parameter *c*.

Decision Trees

I performed pre-pruning on the decision trees, varying amounts of pruning by varying max depth as my changing hyperparameter. Increasing max depth size should theoretically cause the performance to increase but then decrease due to overfitting. The following are graphs derived from incrementally increasing the max depth from 1 to 12.

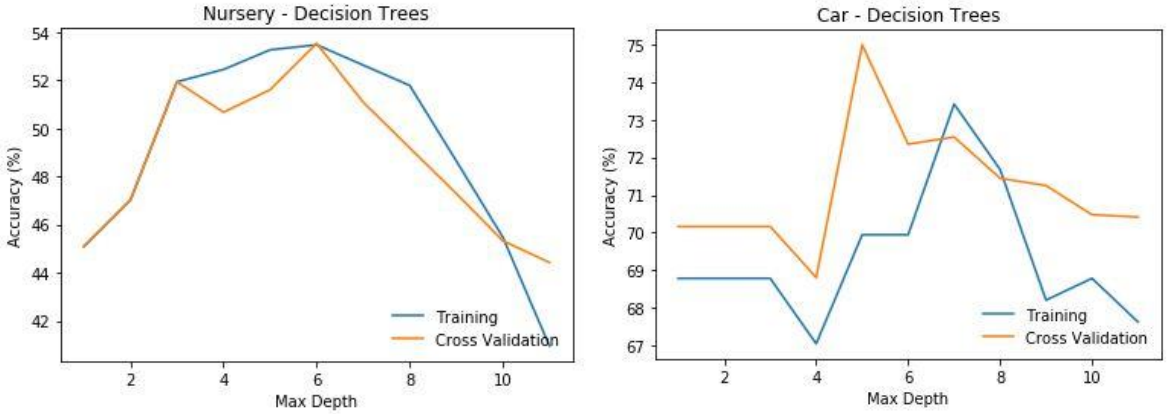

Figure 1. Accuracies for both the nursery and car dataset peaked at around 6 nodes deep before declining sharply due to overfitting. Cross validation scores decline much faster than training scores due to the earlier detection of overfitting. This also presents how with 5 different splits in the cross validation algorithm, the model is able to spike and converge faster than the training model.

Figure 1 describes the behavior of scikit's decision tree classifier on the two datasets. For the cross validation, I split the training sets 5 times and alternated over each, treating them as testing data sets. With better coverage, the algorithm is able to converge faster on a more accurate model than the normal decision tree training and testing sets.

The nursery set has relatively low accuracy overall, which is explained by the high number of possible labels for each sample. Since decision trees work better with a smaller label set, having five different ones made converging on a successful classification function more difficult. The nursery decision tree peaked in accuracy at a max depth of 6, hovering slightly under 54% accuracy, before falling due to overfitting in the leaf nodes. The decision tree still was able to create a somewhat successful function, since random guessing would only produce about a 20% accuracy rate, and the tree was able to achieve over 50%. Another learning algorithm would probably suit the dataset better.

The car set has higher accuracy than the nursery set, though still not obtaining exemplary results, most likely also due to the multidimensionality of the label state space. Cross validation peaked in accuracy at a depth of 5 nodes, with 75% being the accuracy value. Normal decision tree training was slower to peak and at a lower accuracy as well before both fall in percentage.

The calculation times for both the nursery dataset and the car dataset were similar with respect to the number of samples.

Figure 2. The accuracy with respect to how much data was included in the training dataset for both datasets. The car dataset had much higher accuracy overall than the nursery dataset. The size of the training dataset is the percentage of the total dataset used as training data.

The overall comparison of accuracy with respect to training dataset size both demonstrates how much better decision tree learning performed on the car dataset than the nursery dataset while also showing the general increase in accuracy of the decision tree overall. The training was done with max depths of both datasets set to 6, since as shown in Figure 1, the optimal depth for both decision trees was 6 nodes deep. In conclusion, while the decision tree performed decently on the car dataset, overall performance was most likely curbed by the state space size of the labels for both datasets. Perhaps other algorithms will perform more optimally.

Neural Networks

The datasets were run through scikit's neural network ten times, each subsequent time adding another hidden layer. Theoretically, the increase in model complexity should increase accuracy, which is generally true except for the addition of the final hidden layer. While overall accuracy increased with more added hidden layers, as demonstrated in Figure 3, there was plenty

of noise in the dataset for cars, causing the network to spike erratically with increasing numbers of hidden layers. The overall accuracy for the car data was once again higher than that of the nursery data.

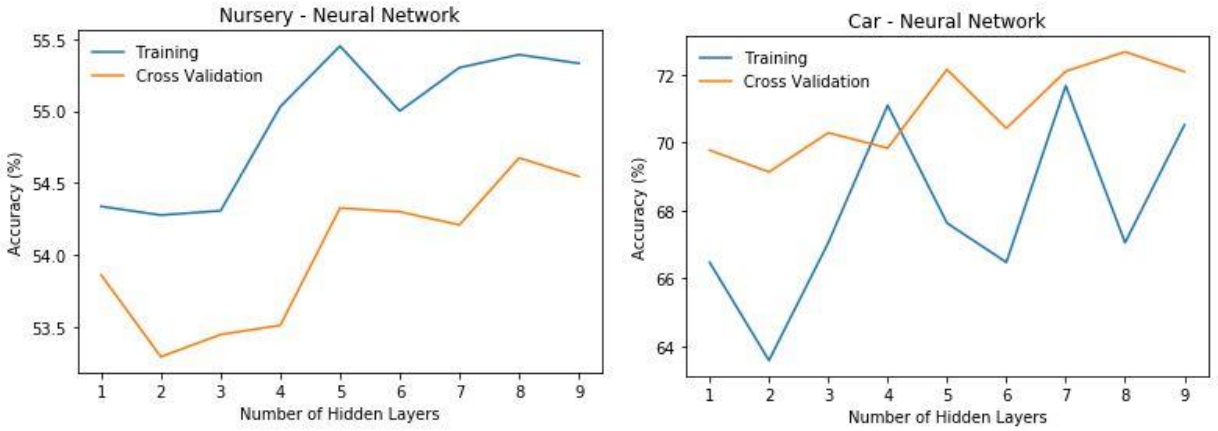

Figure 3. The number of hidden layers has been varied from 1 to 9. While the nursery data seems to have fairly uniform ascension with just a few peaks, the training data from the car neural network jumps all over the place. We can assume from this that the car data is high in noise, though over multiple runs of the neural network over the car data, the shape of the training line changed drastically. The nursery data had 30 nodes per hidden layer, and the car data had 60 nodes per hidden layer.

We can infer from the shapes of the graph that the car data set may have included more noise, but another possibility is due to the smaller size of the dataset. The network is more highly susceptible to overfitting as the network propagates with higher complexity over a small number of samples. Cross validation helps take care of the noise and overfitting by slicing the data and testing multiple combinations of slices five different times and taking the mean over all of those values. The smooth curve in the car graph is a great example of how cross validation is able to compensate for noise and overfitting.

While the nursery dataset should not have been as susceptible to overfitting due to the size of the set, near the end, there still appears to be some evidence of overfitting after 8 hidden layers. The general trend of the curve is to increase in accuracy though, which is accurate to what a neural network should be converging to. The graph plateaus slightly near the end, but if more hidden layers had been added, the cross validation and training networks might have converged together in the process of finding the optimal classification function.

Neural networks are said to be able to map any function, which is not shown in my data. The nursery accuracy continues to hover around 54%, though performing slightly better than my decision tree. However, I also did not increase the number of nodes per hidden layer. In addition, if I had increased the number of hidden layers to something closer to the hundreds or thousands, perhaps the network might have eventually learned an accurate classification function. I can only conclude that the datasets contain too much noise or the number of possible values is too high to successfully converge on a high accuracy function with only nine hidden layers.

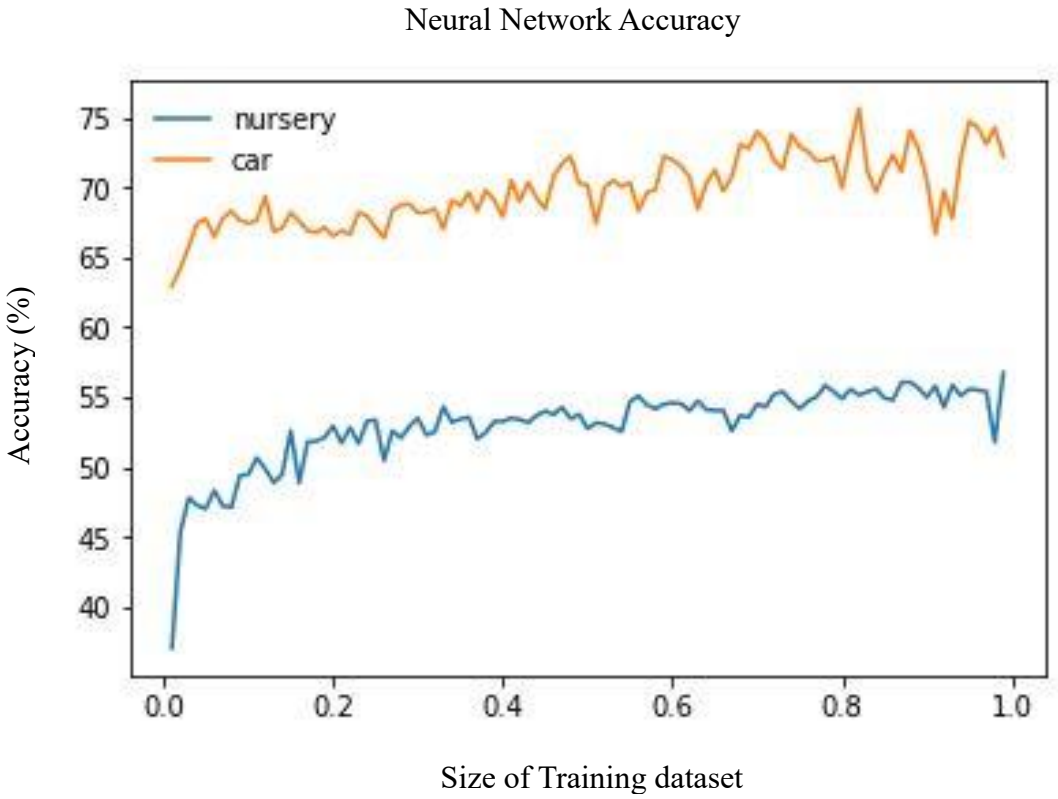

Figure 4. The plotted overall accuracies of the neural network at 8 hidden layers with increasing training dataset size. As in the decision tree, the car has much higher overall accuracy than the nursery dataset. There is a noticeable upwards trend, which is to be expected as the model is able to more accurately fit the testing data with more training data. However, this also makes the model weaker to overfitting and noise.

Boosting

For boosting, I used scikit's Gradient Boosting Classifier. Since boosting responds well to robust pruning, I actually achieved the highest accuracy on the nursery dataset with only a max depth of 2 nodes, whereas I achieved the highest accuracy on the car dataset with a max depth of 6.

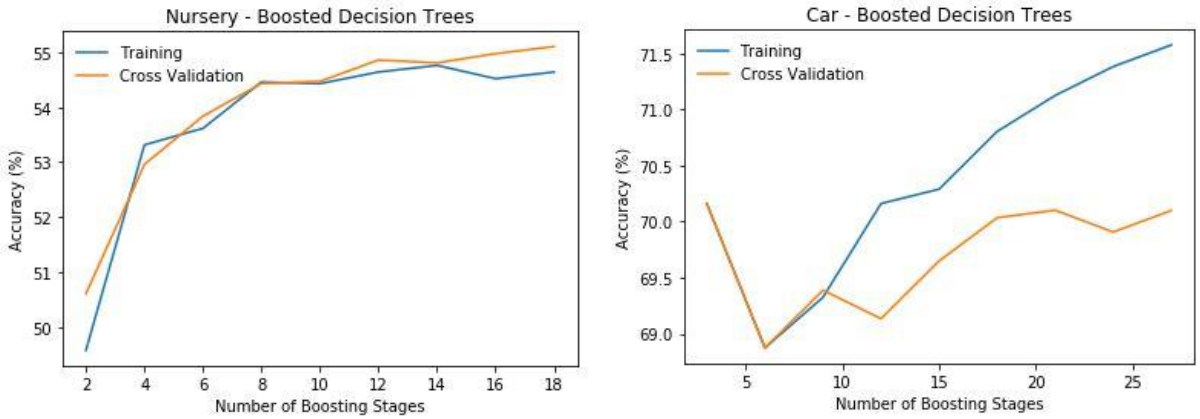

Figure 5. The nursery dataset for boosted decision trees was performed with max depth of 2 nodes, and the car dataset was performed with a max depth of 1 node. With the high amount of training data in the nursery dataset and low max depth, overfitting was prevented by gradient boosting, thus the training and cross validation curves are very similar. The car dataset is smaller, so even with a small max depth of one node, some overfitting still occurs. However, the cross validation curve still behaves similarly to the training curve.

The hyperparameter I toggled was the number of boosting stages which is represented by n_estimators in the code. Since gradient boosting is robust to overfitting, increasing this number typically results in increased performance, which is observed in both the nursery dataset and car dataset for the training curves. As such, the curves for cross validation and training should closely match each other since the training data will not be as skewed by overfitting. This is demonstrated by both the nursery data to a large extent and the cars data to a lesser extent (after all, the dataset is still smaller, and boosting is not so perfect as to be able to ignore all overfitting).

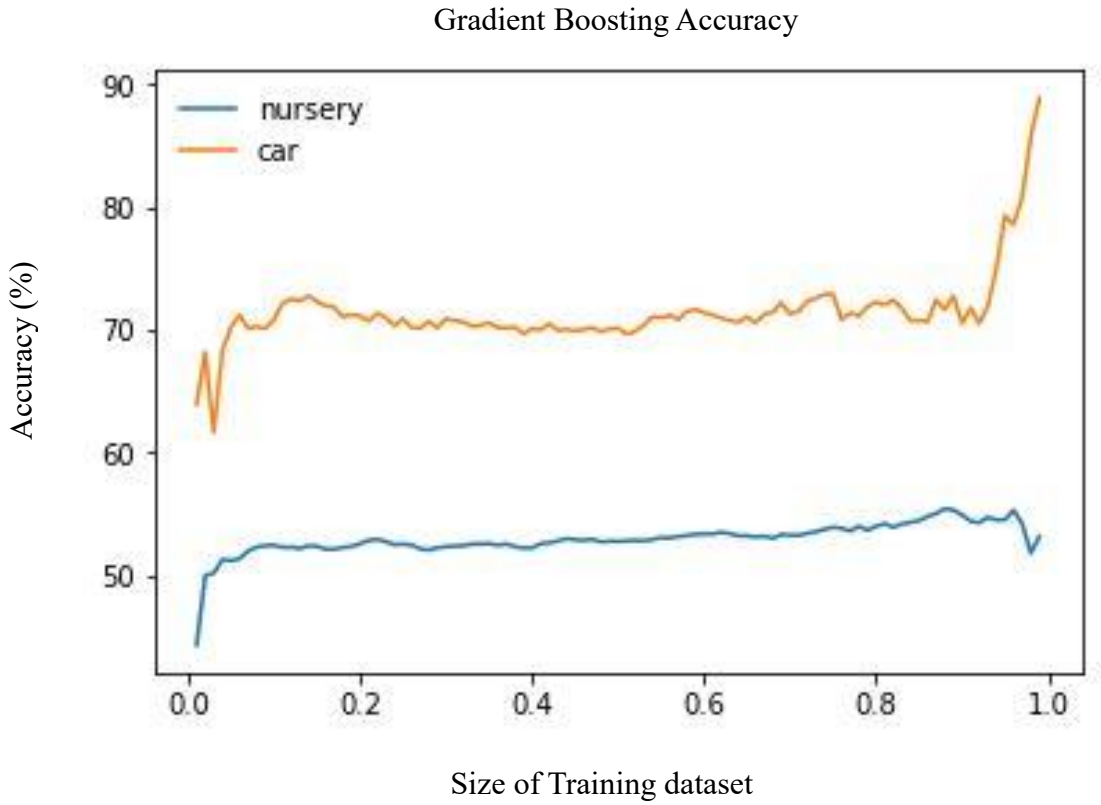

Figure 6. Once again, a demonstration for the discrepancy in accuracy between the nursery and car datasets. However, a more noticeable upwards trend is present, given gradient boosting's tendency to aggressively prune and ignore overfitting. This being said, the more training data is added, overfitting will not become a problem, so the trend is generally more upward-bound. This seems to be a particularly good training algorithm for the cars dataset, actually pushing the accuracy into the 90s.

k-Nearest Neighbors

The number of neighbors was changed for each iteration, going from 1 to 10 neighbors. Generally, we would expect an increase in performance with more neighbors, which is demonstrated fairly well with the corresponding graphs. This makes sense since the smaller k is, the more we restrict or blind the algorithm, making the boundary curves around the data more flexible with low bias. Clearly, this will create a function that is too highly specific towards outliers. Additionally, k-nearest neighbors suffers with the high number of features that both datasets contain, as the algorithm performs more smoothly on datasets with lower dimensionality.

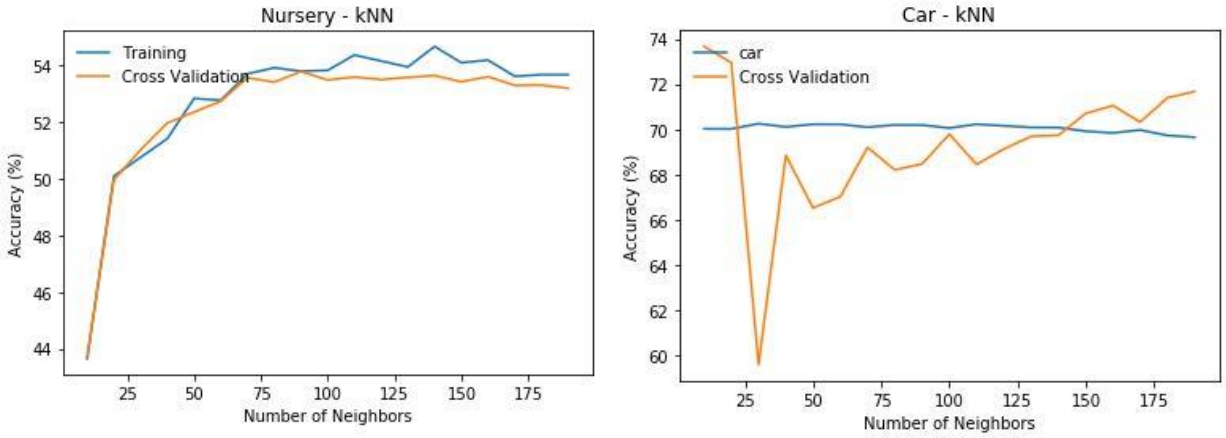

Figure 7. The plotted graphs for the datasets display an overall increase in accuracy, with the cross validation curve being close to the training curve for the nursery data and a bit more hectic for the car data.

The nursery curve did not quite behave the way I was expecting it to, as an increase in the number of neighbors actually reduces complexity and causes higher curve smoothing. The training curve in the car dataset behaved more accordingly, although the cross validation curve may be attempting to compensate for overfitting around the 25 neighbor mark. Since a low k value actually results in higher overfitting, we would expect the training curve to have a higher accuracy value, and lower accuracy values with higher k values. The high variance in the cross validation curve for the cars dataset could be explained by many outliers that may not have been detected in the training curve.

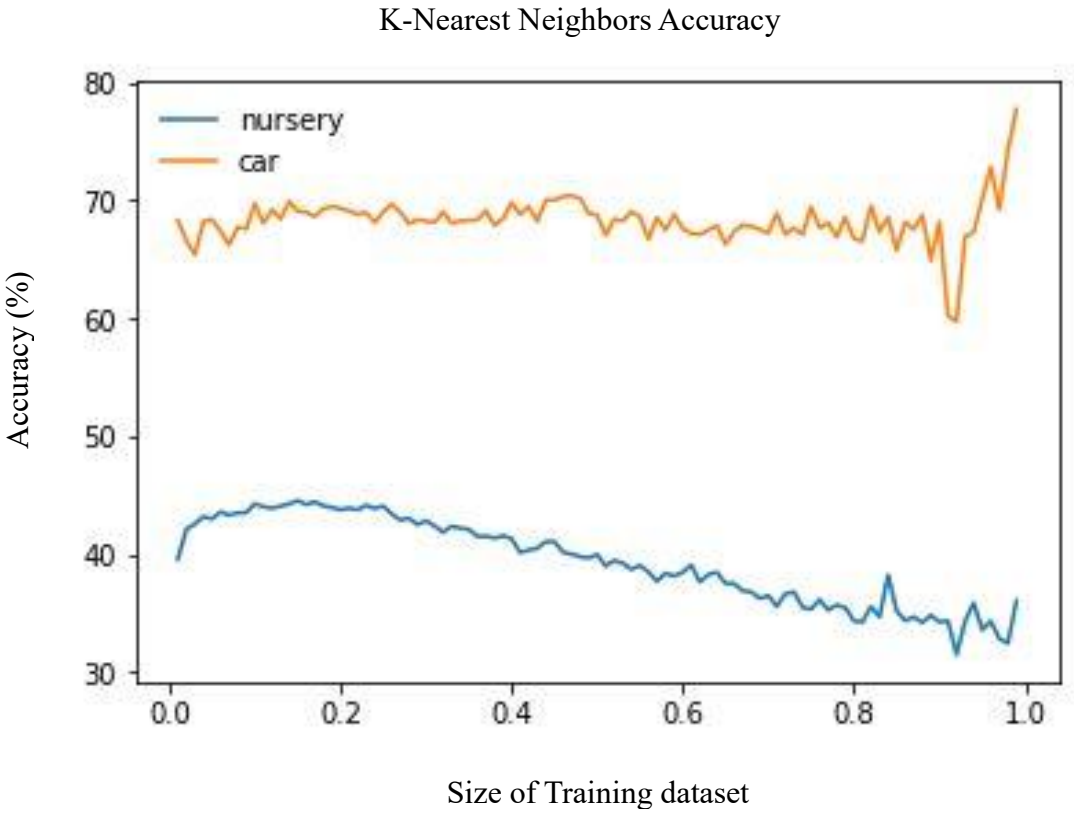

Figure 8. Once again demonstrating the different in accuracy between the nursery and cars datasets. It seems thus far, none of the supervised learning algorithms have been able to adequately describe the nursery dataset, and none of outperformed the others in classifying the car dataset. This graph was built while holding k at a constant value of 5, which is relatively small. Thus, with more data added to the training set, the more overfitting will occur as kNN attempts to accurately describe the data with high complexity.

Support Vector Machines

For support vector machines, I varied the complexity parameter *c*, increasing it from 1 to 200. In general, the default value for *c* is 1.0, but by changing it up and increasing the value, the SVM attempted to more accurately classify all of the training points correctly. The smaller the value of *c,* the more misclassified examples should appear. Given that a linear kernel is used, high accuracy is not expected for the nursery and car datasets since they might not necessarily be linearly separable and contain noise.

With the default complexity parameter of 1, support vector machines typically have high variance and wider margins in the hyperplane. This means that bias will be low, and there is a higher chance to misclassify data samples, especially with a highly dimensional dataset. As such, the expectancy of support vector machines to perform optimally on the car and nursery dataset is low, since both datasets have relatively high dimensionality and are most likely not linearly

separable. A support vector machine involving a kernel of a higher dimension could probably achieve a classification with higher accuracy than the linear kernel used here.

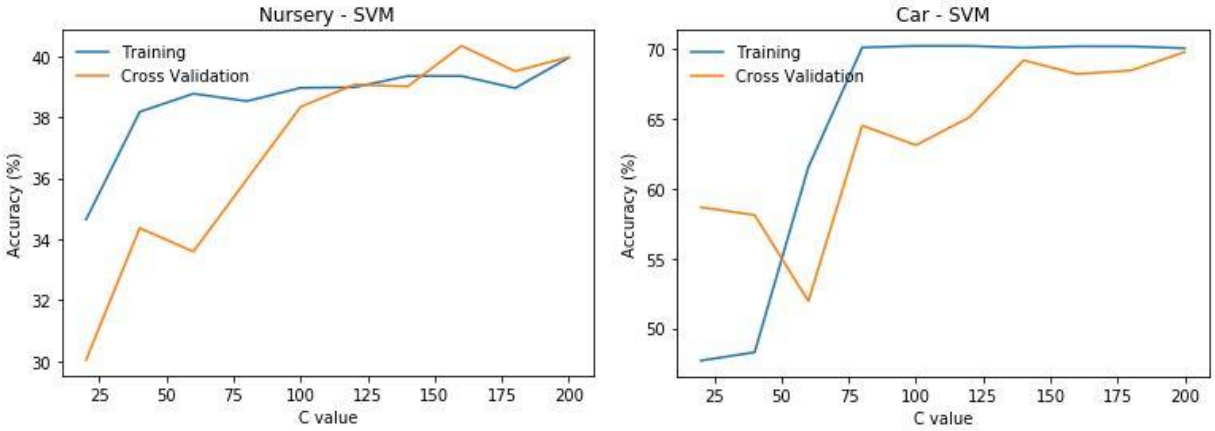

Figure 9. As expected, a lower *c* value instilled lower accuracy rates, since the hyperplane is created with more generous margins and is more likely to misclassify data. As the complexity parameter is increased, the smaller the hyperplane margins will become in the SVM's attempt to classify more points accurately. The car dataset most definitely experienced overfitting as a result of this, causing the training curve to jump above the cross validation curve. Some overfitting also occurred in the nursery dataset, though not to the extent of the car dataset.

Since the *c* parameter focuses mainly on finding a hyperplane that correctly separates as many samples of data as possible versus finding a hyperplane with the largest minimum margin, the margins between data points will shrink as a more perfect hyperplane fit will be found in order to include more outliers. As explored in the previous learning algorithms, the data is not exactly neatly separated and is highly dimensional, both causing problems for the SVM.

Figure 10. There is not really a noticeable climb in accuracy with added data samples since adding data samples does increase possible noise. However, this graph was constructed with the default *c* parameter of 1, meaning that the SVM was not looking to correctly classify every example and was more focused on finding a hyperplane with the highest possible minimum margin. Thus, even with added noise, the hyperplane created would not have moved much from its original placement.

Conclusion

With the completion of five different supervised learning algorithms, surprisingly none of them particularly fit the datasets well. I can only conclude that this is due to high variance and dimensionality within the data or perhaps incompatibility of scikit's algorithms with the chosen datasets. However, if I were to hypothesize on which algorithms should have performed optimally on the datasets, I would have expected the neural network to find the most accurate classification of the data. With a drastic increase in number of neurons and hidden layers, the network most likely could have found a suitable function for the complex data inputs. Decision trees, k-nearest neighbors (with a linear kernel), and support vector machines are more suitable for low dimensional datasets, such as ones with binary classifications. Thus, it would have been difficult even given ample training time for those algorithms to find an optimal solution that avoids overfitting.

An overall summary of the datasets given the learning algorithms shows that the nursery dataset performed the best with neural networks and boosted trees. This behavior is to be expected, given a neural network's ability to fit highly complex datasets and boosted trees ability to aggressively prune to avoid too much overfitting to highly varied data. The car dataset performed about equally across all the algorithms but performed the best with decision trees. The car dataset was originally generated to operate well with decision trees, as the values are all non-continuous and not as varied as the nursery dataset. I do believe a better decision tree classifier could have been implemented since the clarity of the car dataset should have yielded highly accurate classifications from the learning algorithms. Perhaps some more hyperparameters could have been played around with and combined to find the optimal combination that would create highly accurate classifications.