**Datasets:**

*Wine Quality(red):*

This dataset contains 1599 instances of red wine reviews and its physiochemical composition [1]. In particular, the red wines sampled were variants of the Portuguese “Vinho Verde” wine. The dataset is interesting because there are many facets of the wine to consider when reviewing it, including its grape variety, where it was grown, how long it had been aged, and so on. There are only 11 features in the dataset, including acidity, amount of chlorides, amount of sulphates, and alcohol content. While some, like density, is not typically something that a taster might consider when deciding whether a wine is good, some others typically have some bearing on the taste of the wine. Furthermore, they are all objective measures, and not swayed by subjective tastes. I believe that industry would be interested in a predictor of good or bad wine that is based on quantitative metrics that can be more readily manipulated.

For the wine dataset, not much data processing was required. The features were all normalized to zero mean and 1 standard deviation. The reviews were aggregated into two classes: good and bad. The class aggregation was based on the median of the reviews, which cut the data into roughly equal halves.

*Pet Adoption:*

This dataset contains information and outcomes about 25,000 dogs and cats at the Austin Animal Center [2]. There are 6 features in the dataset, including the name, age, breed, and color of the animal. This dataset is interesting because of the amount of data processing that is required, and also the potential of the predictor to help animal shelter workers focus their efforts on the animals that are less likely to be adopted or returned to their owner. While the wine dataset had features that were entirely continuous, the pet adoption dataset had mostly categorical data, which required different processing and may affect the applicability of the models.

For the categorical features, one-hot encoding was attempted. However, that left a large number of features (300-1000). Due to time and computational constraints, the features that led to an explosion in the number of features after one-hot encoding were disregarded. The single continuous feature was also normalized. The classes were also aggregated into 2 outcomes: good and bad. “Adopted” and “returned to owner” were treated as good outcomes, while “died”, “euthanized”, and “transferred” were treated as bad outcomes. While the distribution of the data between the two classes were not roughly equal, it was not too skewed either. There were about 10,000 instances of good outcomes, and 15,000 instances of bad outcomes.

**Hyperparameter Tuning:**

The best model was found through hyperparameter tuning of the models. The dataset was split first into two components, after a random shuffle to make sure that there is no ordering to the data. 10% of the dataset formed the testing set, and the remaining 90% the tuning set. The separation of the dataset into testing and tuning sets ensured that none of the testing data would be available to tune to model. This forces the testing data to then be an approximation for future data that the model has not seen.

From the tuning set, 5-fold cross-validation was performed. Cross-validation splits the data into 5 partitions, and trains 5 models with identical hyperparameters. At each training iteration, one of the partitions is taken as the validation set, and the remaining 4 as the training set. The model is trained on the training set and validated by predicting the validation set. At the next iteration, the 2nd partition is taken as a validation set, and the remaining 4 as the training set. At the end of all iterations, the scores of the 5 models are averaged. Because there can be variance in simply choosing one validation set and training set, cross-validation reduces the variance by training multiple models using the tuning set, and taking the average score.

Using the same tuning set, models with various hyperparameters are tested. The hyperparameters with the highest validation score is then taken to be the best model. In order to test the model, we use the chosen model to predict the testing set.

The score used for tuning varies with the context and distribution of the data. For a dataset that has extreme class imbalance, the accuracy metric is not recommended. For both the wine and pet adoption datasets, however, there is no extreme class imbalance, and I used the accuracy metric to score the models.

For the implementation of the models, I used the models that come with the scikit-learn package.

*K-Nearest Neighbors:*

K-nearest neighbor is an algorithm that calculates the similarity of the query instance to all the instances used to train the model. The labels of the k nearest instances are then used to predict the query instance. With a small k, the decision boundary follows the peculiarities of the training data more closely. This may lead to overfitting, and so a larger k is used to prevent the model from learning the noise of the training data.

The training and validation scores for this model over the 2 datasets can be seen in figure 1.

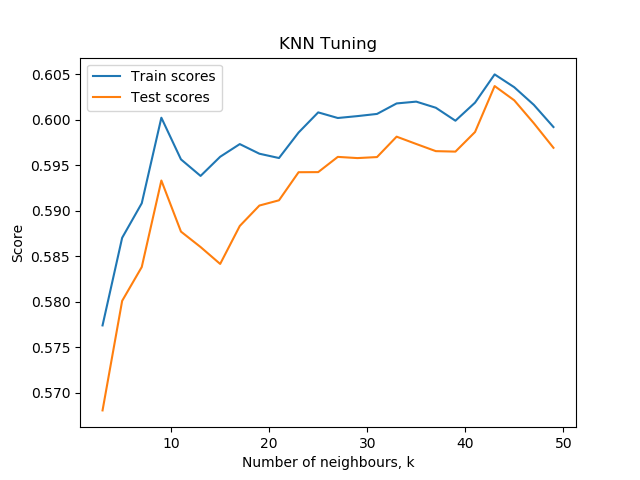
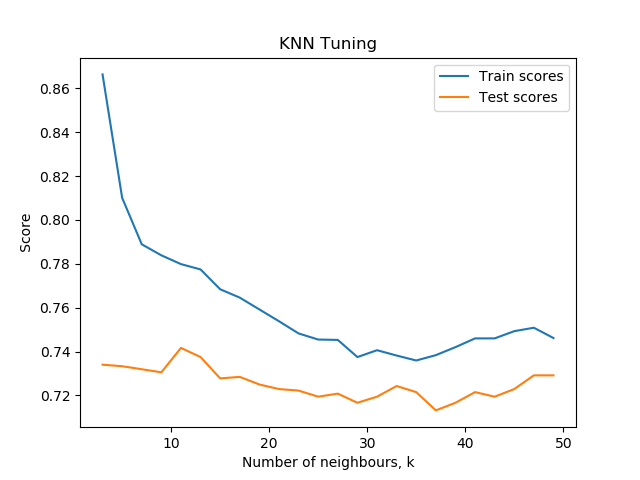


Figure 1. Training and validation accuracy for the wine (left) and adoption (right) datasets for the knn classifier

For the wine dataset, we can see that the model is overfitting at lower values of k, which is what we expect. We can also see that the accuracy is approximately the same over the various values of k, but there seems to be a slight improvement at k=11. While the exact value of k may change with a different run of cross-validation, this model on this dataset seems to perform better with smaller values of k.

For the adoption dataset, we see that there seems to be instability as k varies. The Euclidean distance was used as the similarity metric for both datasets, as they contain continuous data. However, it may be more appropriate to use a distance metric like the Hamming distance. That may account for the unusual scores, but the implementation of the model does not allow for the use of two different distance metrics, since Hamming distance does not work with continuous data. Nonetheless, we can see that the model on this dataset seems to perform better with larger values of k, and the training accuracy isn’t high at all. This may suggest that the points of both labels may be very mixed together, making it hard to have a clear decision boundary.

The accuracy on the held out testing set is shown in table 1.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Table 1: Training, validation, and testing accuracy for the k-nearest neighbor classifier** | | | | |
| *Dataset* | *k* | *Training Accuracy* | *Validation Accuracy* | *Testing Accuracy* |
| Wine | 11 | 0.780 | 0.741 | 0.692 |
| Pet Adoption | 43 | 0.605 | 0.604 | 0.627 |

*Decision Trees:*

Scikit-learn’s decision tree classifier implements an algorithm similar to the ID3 algorithm. At each node of the decision tree, a feature is selected and a question is set. The samples are then split into children nodes based on how they answer the question. The way the feature and the question is selected is based on the GINI impurity index, which is similar to the information gain metric based on entropy.

If there is no specified depth to the tree, it can be grown deep and come to ask many questions about the features. The more questions are asked, the better the tree fits to the data, but there is a risk of overfitting. To prevent overfitting, the trees can be pruned in various ways. For this classifier, I tuned the maximum depth hyperparameter, which limits the growth of the tree to a depth less than or equal to the hyperparameter value.

The training and validation scores for this model over the 2 datasets can be seen in figure 2.

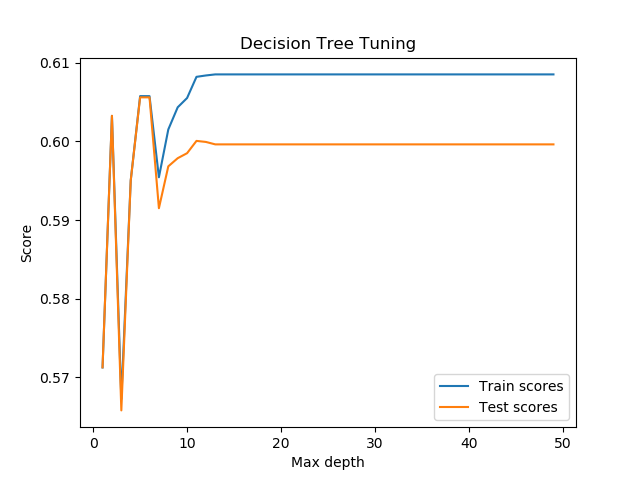
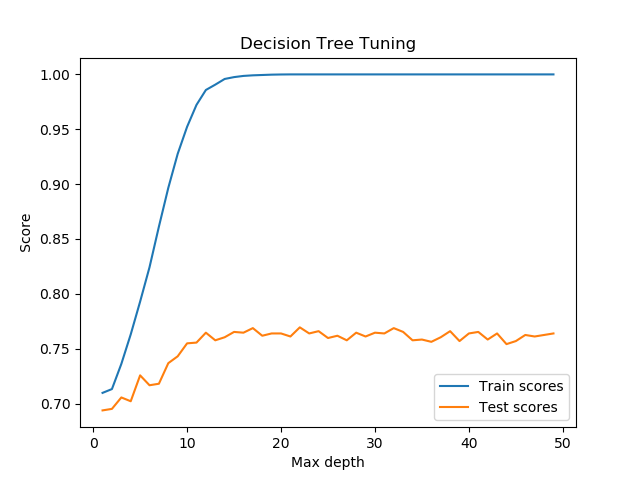


Figure 2: Training and validation accuracy for the wine (left) and adoption (right) datasets for the decision tree classifier

For the wine dataset, the graph shows the training accuracy rising to 1, while the validation accuracy remains around 0.76 after the tree is more than 15 levels deep. We have only 11 features in this dataset, but because the features can be reused at each level while splitting at a different threshold, the tree can be grown deep enough to perfectly classify all the training data. Through the fluctuations in accuracy after level 15, we have our best validation accuracy at maximum depth=22. It also appears that it is likely that most of the features are useful as predictors, since best maximum depth is greater than the number of features.

For the pet adoption dataset, the graph has unusual fluctuations in accuracy with shallower trees, and then a steady accuracy with deeper trees. Because the algorithm doesn’t force the tree to grow to it’s maximum depth, one possible reason for the constant accuracy past ~15 levels is that the tree is no longer being grown. Furthermore, only one of the features in the dataset is continuous – for the other Boolean variables, they cannot be reused down the same path of the tree, which limits the maximum depth that it can be grown. For that reason as well, the best accuracy can be seen with a small maximum depth.

The accuracy on the held out testing set is shown in table 2.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Table 2: Training, validation, and testing accuracy for the decision tree classifier** | | | | |
| *Dataset* | *Maximum depth* | *Training Accuracy* | *Validation Accuracy* | *Testing Accuracy* |
| Wine | 22 | 1 | 0.769 | 0.723 |
| Pet Adoption | 5 | 0.610 | 0.610 | 0.595 |

*Support Vector Machines:*

Support vector machines (SVM) attempt to converge to a hyperplane, defined by their kernel function, that separates the data points. Here, I attempted linear and polynomial kernels. The hyperparameter to tune for SVMs are the regularization parameter, C. With a high C, the penalty for a misclassification is higher, and a smaller margin hyperplane would be preferred if the data is separable. As C grows larger, the model tends to overfit more.

The training and validation scores for the linear kernel model over the 2 datasets can be seen in figure 3, and the scores for the polynomial kernel model can be seen in figure 4.

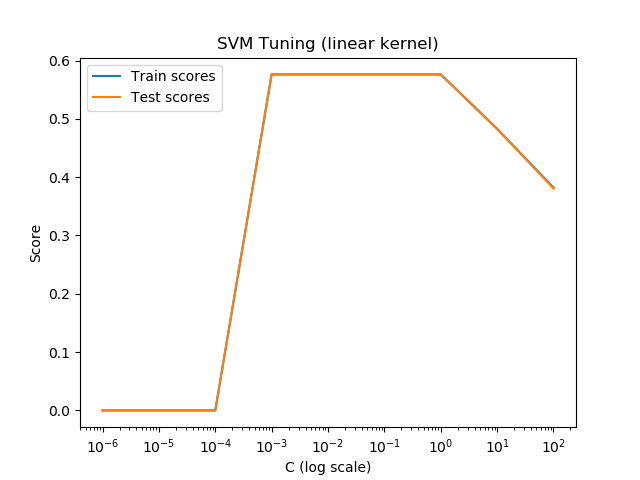
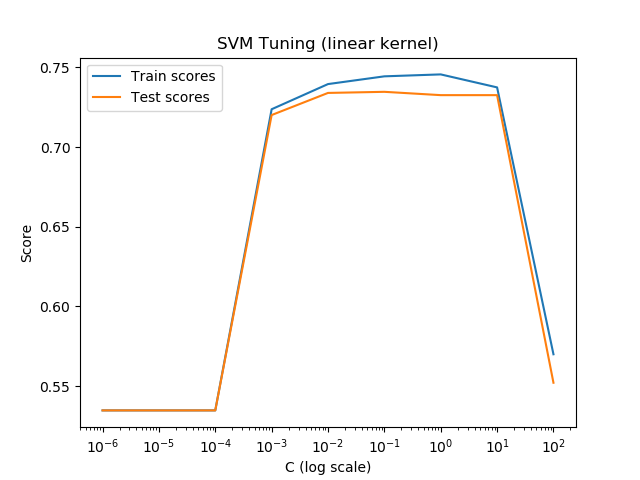


Figure 3: Training and validation accuracy for the wine (left) and adoption (right) datasets for the SVM classifier with polynomial kernel

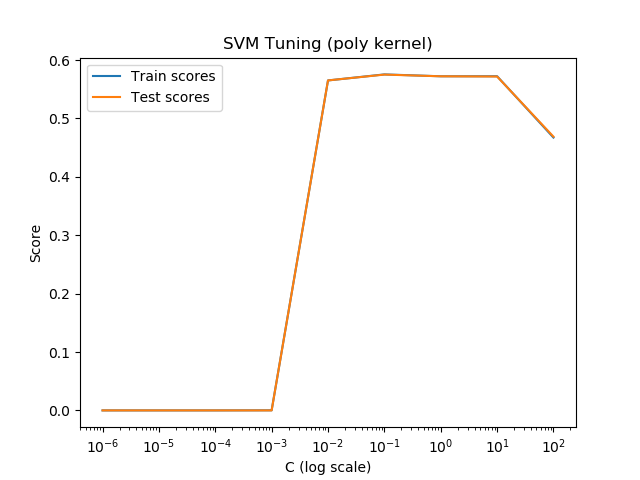
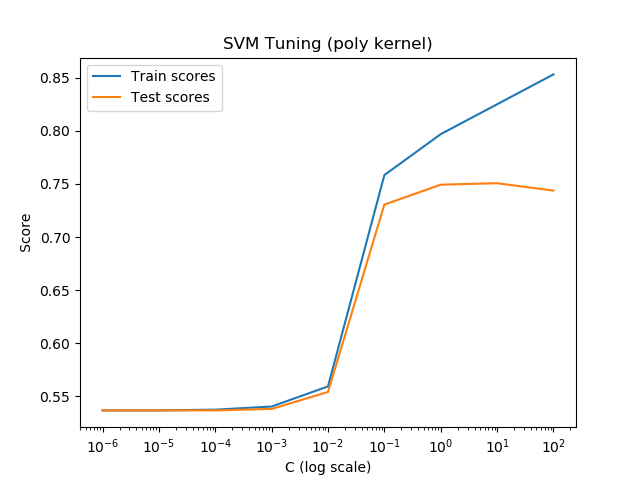


Figure 4: Training and validation accuracy for the wine (left) and adoption (right) datasets for the SVM classifier with polynomial kernel

For both datasets, there is very little difference between the training and validation accuracy across the board. In 3 of the 4 cases, the training accuracy also went down as C grew. This was surprising to see, but it may indicate that the model is not flexible enough to overfit on the data for a higher training accuracy. Unlike the decision tree classifier, which has the potential to add levels until it perfectly classifies the training data, the SVM is constrained by the shape of the kernel function.

The accuracy on the held out testing set is shown in table 3.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Table 3: Training, validation, and testing accuracy for the SVM classifier** | | | | |
| *Dataset* | *C* | *Training Accuracy* | *Validation Accuracy* | *Testing Accuracy* |
| Wine (linear) | 1 | 0.744 | 0.732 | 0.761 |
| Wine (poly) | 10 | 0.825 | 0.751 | 0.755 |
| Pet Adoption (linear) | 0.001 | 0.576 | 0.576 | 0.552 |
| Pet Adoption (poly) | 0.1 | 0.579 | 0.579 | 0.563 |

*Boosted Decision Trees:*

Boosting is an ensemble classifier of weak learners that primarily reduces bias in its weak learners. The algorithm used for boosting was Adaboost, which works by iteratively training weak learners with more weight given to the samples that the previous learners found difficult to classify correctly. Hyperparameters that could be tuned include the hyperparameters of the learners themselves, and the number of learners in the boosting ensemble. In the scikit-learn implementation of boosting, the learners are decision trees with a maximum depth of 1, making them simple, highly biased models. Since the ensemble reduces bias, more iterations and number of learners improves the accuracy.

The training and validation scores for the boosted decision tree classifier over the 2 datasets can be seen in figure 5.

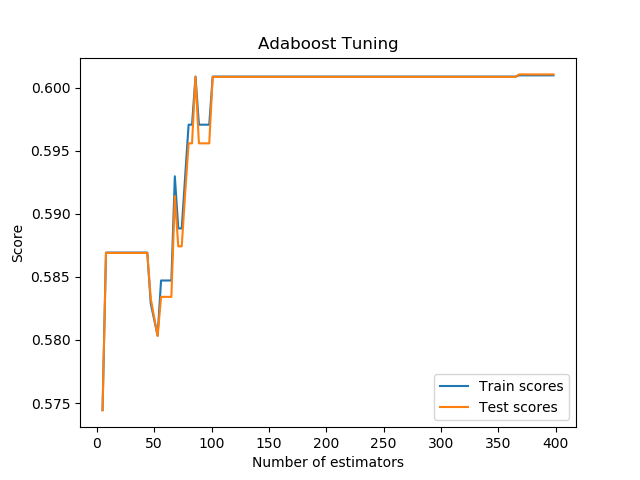
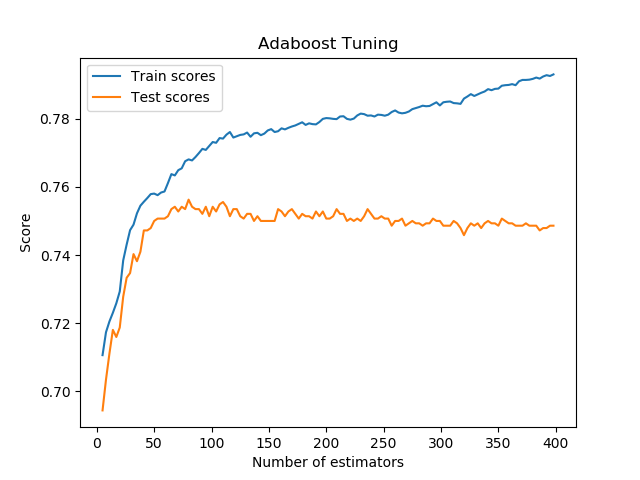


Figure 5: Training and validation accuracy for the wine (left) and adoption (right) datasets for the boosted decision tree classifier

For the wine dataset, we can see that the training accuracy is rising with each iteration. However, the validation accuracy has reached a plateau at approximately 80 learners. We can see a similar trend with the pet adoption dataset, but the changes in accuracy with each iteration is much sharper. We can also see the same plateau at around 100 learners, but then see a slight uptick in accuracy at around 350 estimators. This difference in smoothness may be due to the number of features in the dataset. The pet adoption dataset has far fewer features (4), compared to the wine dataset (11). Because each decision tree has to decide what feature to split on, and the pet adoption dataset only has a handful of choices, it may split on the same feature again and again for a few iterations before finally picking a different one. In addition, the features of the pet adoption dataset are mostly Boolean, so even if the same feature is picked, there is no threshold to set to split the data differently.

The accuracy on the held out testing set is shown in table 4.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Table 4: Training, validation, and testing accuracy for the boosting classifier** | | | | |
| *Dataset* | *Maximum depth* | *Training Accuracy* | *Validation Accuracy* | *Testing Accuracy* |
| Wine | 80 | 0.768 | 0.756 | 0.793 |
| Pet Adoption | 368 | 0.601 | 0.601 | 0.597 |

*Neural Networks:*

The neural network implemented by scikit-learn is a multi-layer perceptron, with rectified linear units as the activation function for each hidden layer. There are many different hyperparameters to tune for neural networks, including the number of hidden units in each hidden layer, and the number of layers. To tune the model, I fixed a number of hidden units, and varied the number of hidden layers. Neural networks can be allowed to express any arbitrary function, but is limited by the complexity of the network architecture. With more hidden units and more hidden layers, the neural network can represent more functions, and therefore become more prone to overfitting.

The training and validation scores for the boosted decision tree classifier over the 2 datasets can be seen in figure 6.

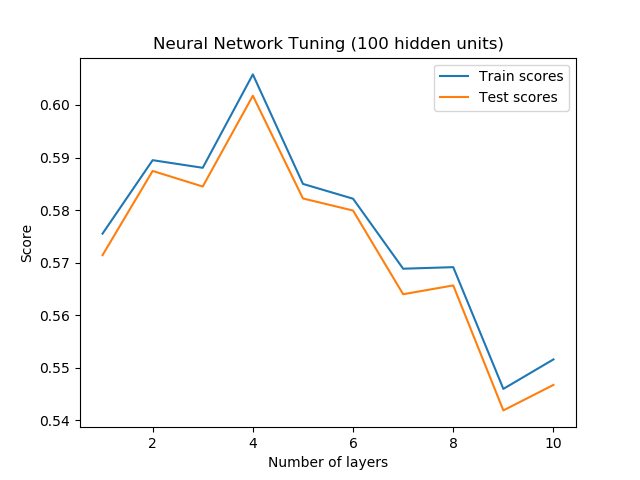
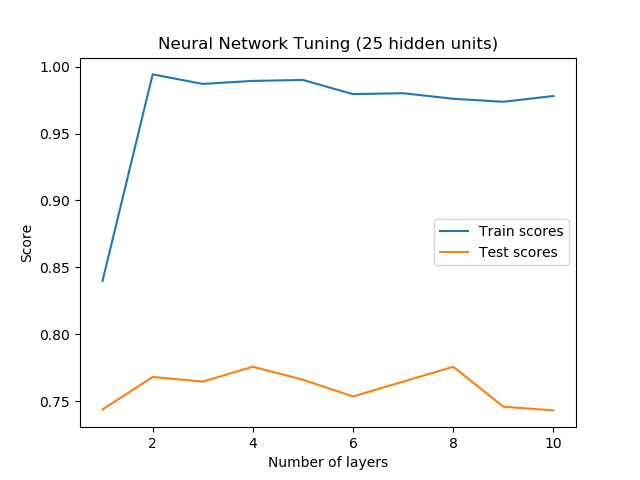


Figure 6: Training and validation accuracy for the wine (left) and adoption (right) datasets for the neural network classifier

For the wine dataset, we can see that there is a large gap between the training and the testing error. The training error is close to 1 after 2 layers, while the testing error hovers around 0.77. This is a moderately complicated network, since there are only 11 features in the dataset and 25 hidden units in each layer. However, it is complex enough to better learn the noise in the training data, while not do better at generalizing over the validation data.

For the pet adoption dataset, we see a more unusual trend. The training and testing score gap is smaller, while the number of hidden units have risen to 100. We also do not see the typical overfitting curve, where the training score remains high as the parameter increases. We have seen this trend of a small training and testing score gap in the other models as well, leading me to believe that this may be due to the dataset itself – possibly the dominance of Boolean data in the features. It is difficult to determine the cause of such behavior, however, because the complexity of the neural network calculations tend to make it a black box.

The accuracy on the held out testing set is shown in table 5.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Table 5: Training, validation, and testing accuracy for the neural network classifier** | | | | | |
| *Dataset* | *Hidden Units* | *Hidden Layers* | *Training Accuracy* | *Validation Accuracy* | *Testing Accuracy* |
| Wine | 25 | 4 | 0.989 | 0.776 | 0.767 |
| Pet Adoption | 100 | 4 | 0.606 | 0.602 | 0.613 |

**Learning Rate:**

*KNN, SVM, and Decision Trees*

The learning rate of an algorithm illustrates how much data the algorithm needs in order to perform well. For the 3 non-iterative algorithms (KNN, SVM, and decision trees), the entire dataset was split into a training and testing set of 90% and 10%, respectively. Of the training set, different fractions were used to train a model with the best hyperparameters found in the previous section. The results are shown in figure 7.

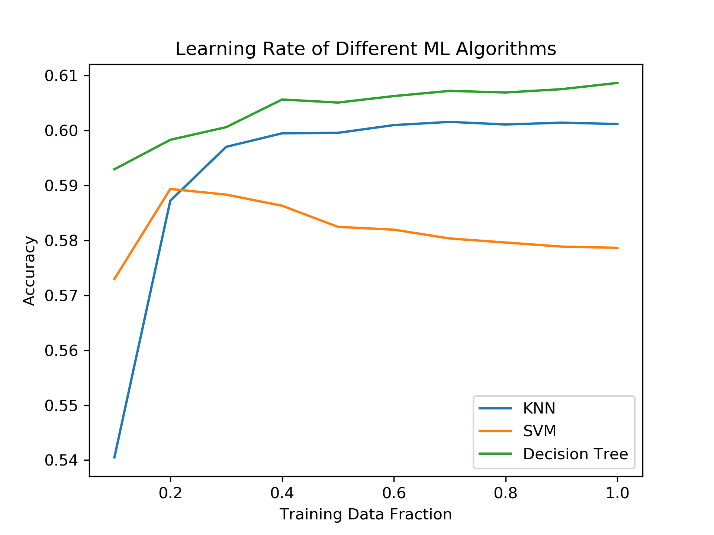
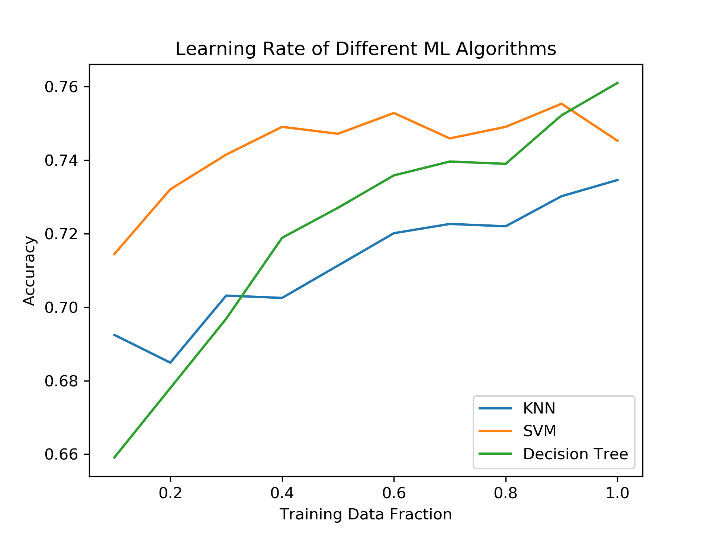


Figure 7: Accuracy as training set size increases for the wine (left) and adoption (right) datasets for the KNN, SVM, and decision tree classifiers

Most of the models showed a general improvement in accuracy as training size increased, except for the SVM on the pet adoption dataset. The accuracy levels off at higher training fractions, however, and the peak at 0.2 training fraction may have been the result of variance caused by randomization.

The steepness of the learning curve also varied between the two datasets. For the KNN and decision tree models on the wine dataset, the accuracy doesn’t seem to have hit a plateau yet. It may be that more data would improve the accuracy of the models further. This may be because of the curve of dimensionality, since the number of instances (1350) is moderately small compared to the number of features (11). For the KNN and SVM models on the pet adoption dataset, both seem to hit a plateau at around 30% of the training data. Likewise, the reason could be that the number of instances (24,000) is large compared to the number of features (4). In addition, the absolute number of instances at which the pet adoption models plateau is 7300, a number still far greater than the entire wine training dataset.

*Boosted Decision Trees and Neural Networks*

The learning rate for iterative models like the boosted decision tree and neural networks can be measured by calculating the testing accuracy at each iteration. For the boosting classifier, one iteration refers to the training of a learner over the weighted samples. For the neural network, one iteration is typically taken to mean one epoch, which is the training of the network over the whole training set. A neural network is then trained iteratively over multiple epochs. The results for the boosting classifier are found in figure 8, and the results for the neural network in figure 9.

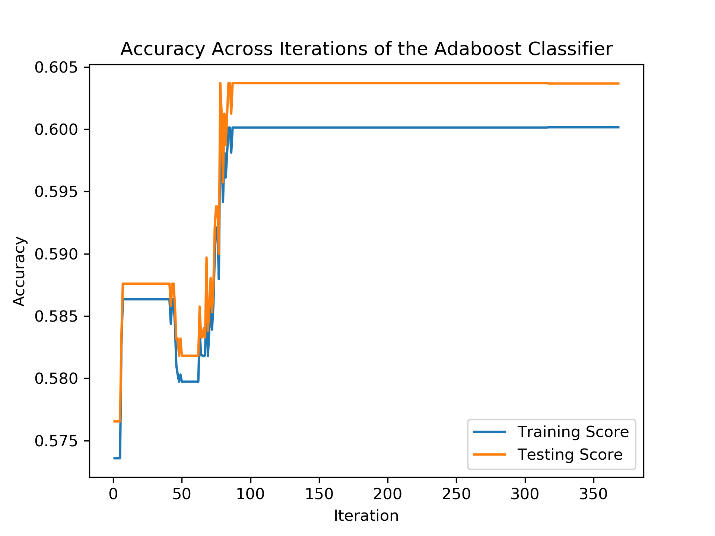
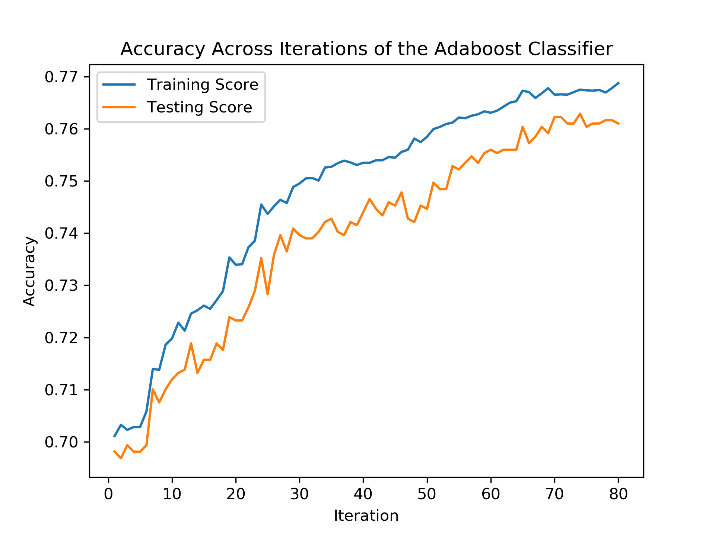


Figure 8: Accuracy across iterations of the boosted decision tree classifier for the wine (left) and adoption (right) datasets

We can see that both training and testing error increases with each iteration for the boosting classifier. Just like in the earlier analysis for the results of tuning, the sharp jumps in accuracy for the pet adoption dataset may be due to the small number of features and the domination of Boolean features. Since features cannot be selected in too many different ways, and the Boolean features cannot be split in in different ways, each iteration may add the same decision tree stump over multiple iterations, causing the accuracy to remain the same until a different stump is chosen.

We can also see that the number of iterations required to reach a good accuracy is approximately the same for both datasets (80-100). While the testing accuracy is higher than the training accuracy for the pet adoption dataset, that is most likely due to the random split between training and testing sets.

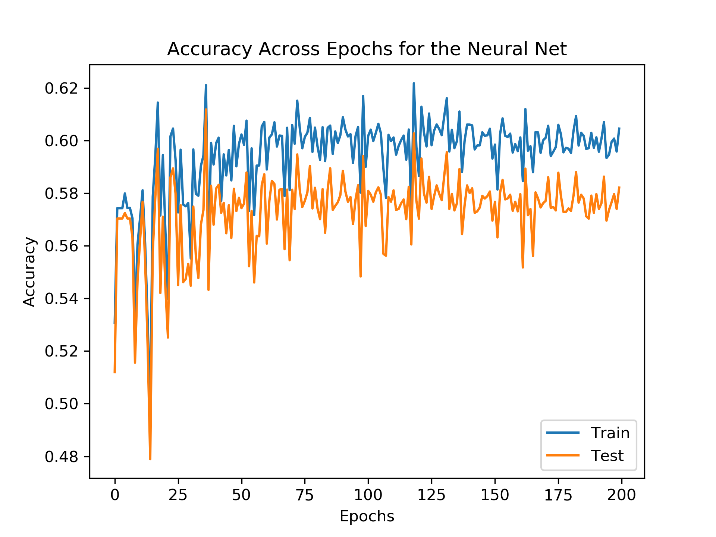
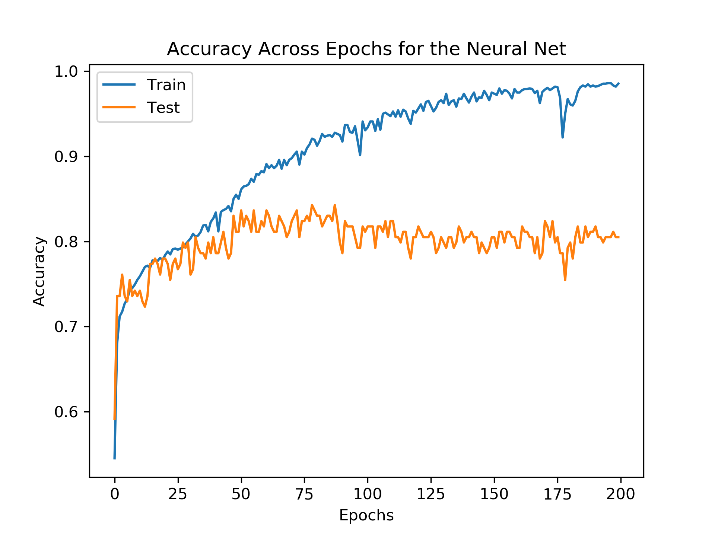


Figure 9: Accuracy across epochs during the training of the neural network for the wine (left) and adoption (right) datasets

For the neural network training, we can see that the neural network over the wine dataset tends to overfit as the number of epochs increases. The training accuracy rises close to 1, while the testing accuracy fluctuates around 0.77. We don’t see that in the pet adoption dataset, which would be consistent with the tuning curve in the previous section. The training and testing accuracy follows each other closely, but we can also see that the magnitude of the fluctuation tends to decrease as the number of epochs increases.

The cause of the fluctuation could be due to the schedule of the learning rate. For these models, the learning rate was a small constant number. In some schedules, the learning rate would start high and slowly decrease close to zero, allowing the model to start the search in a wider area for a better local minimum, and then hone in on the minimum in later iterations. In my implementation, due to the time and computational constraints, I decided on a small learning rate to minimize instability.

**Conclusion:**

The testing accuracy should be used when comparing the different models, since that is the dataset that most accurately represents future data that the model has not yet seen. The testing accuracies are shown in table 6.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Table 6: Testing accuracy for the best models of each type** | | | | | |
| *Dataset* | *KNN* | *Decision Tree* | *SVM* | *Boosting* | *Neural Network* |
| Wine | 0.692 | 0.723 | 0.761 | 0.793 | 0.767 |
| Pet Adoption | 0.627 | 0.595 | 0.563 | 0.597 | 0.613 |

For the wine dataset, the boosted decision trees performed the best, while the KNN classifier performed better for the pet adoption dataset. This difference may be due to the kind and number of features for each dataset. With continuous features, the wine dataset has more options when it comes to decision tree splitting, and the reduction in bias of the simple decision tree stumps may have led to a better model.

On the other hand, the other models may have more difficult time with handling Boolean data. The KNN classifier also employs a similarity metric that works better in lower dimensions than higher dimensions. Since the number of dimensions in the pet adoption dataset is small, the measure of similarity may have worked better.

**Sources:**

[1] Wine quality dataset source: https://archive.ics.uci.edu/ml/datasets/wine

[2] Pet adoption dataset source: https://www.kaggle.com/c/shelter-animal-outcomes