Analysis of Cancer Data with multiple supervised learning Algorithms

For this report I will be reporting on the tradeoffs between multiple supervised learning algorithms on a breast cancer and lung cancer dataset provided by the UCI Machine Learning Repository.

Breast Cancer Dataset:

I chose this data because breast cancer is a common occurrence, and a misdiagnosis could be fatal. If a false positive for a malignant tumor occurs, then it could lead to a person getting unnecessary chemotherapy, while if it's a false positive for a benign tumor, then they could live with breast cancer until it becomes a far bigger problem. It would be interesting to see how effective a machine learning algorithm could do on a set like this.

The breast cancer dataset comes from the Wisconsin Breast Cancer Database and contains 699 instances. The researchers noted 10 features, and noted whether the tumor detected was benign (not harmful) or malignant (harmful). The following is a breakdown of the 10 features used:

#	Attribute	Domain
1	Sample code number	Id number
2	Clump Thickness	1 - 10
3	Uniformity of Cell Size	1 - 10
4	Uniformity of Cell Shape	1 - 10
5	Marginal Adhesion	1 - 10
6	Single Epithelial Cell Siz	1 - 10
7	Bare Nuclei	1 - 10
8	Bland Chromatin	1 - 10
9	Normal Nucleoli	1 - 10
10	Mitoses	1 - 10
11	Class	2 for benign, 4 for malignant

Table 1.01: Breast Cancer Data Summary

Before starting with the data analysis, I cleaned the data to only focus on the features that would give a good prediction of the type of cancer. To do this, I removed the samples with missing information (containing '?' values). There were very few instances of this, only 16. This reduced our data set size to 683 instances. After this, I removed the "Sample Code Number" feature, as it is simply an id that has no relation to the actual cancer prediction. This reduces the number of features we are looking at to 9.

Finally, in order to ensure all of our features have equal weight, I normalized the data so that all attributes were between 0 and 1.

Roughly 35% of the dataset has a positive marker for cancer. This means that if we were to simply guess that the person has a benign tumor every time, then 65% of the time we would be right. We want to beat this.

Below, I decided to see if there were any correlations in the data as well that might explain which features would be the most helpful in explaining which variables have the highest "weight" in a prediction.

	Clump Thickness	Uniformity of Cell Size	Uniformity of Cell Shape	Marginal Adhesion	Single Epithelial Cell Size	Bare Nuclei	Bland Chromatin	Normal Nucleoli	Mitoses	Class
Clump Thickness	1.000000	0.642481	0.653470	0.487829	0.523596	0.593091	0.553742	0.534066	0.350957	0.714790
Uniformity of Cell Size	0.642481	1.000000	0.907228	0.706977	0.753544	0.691709	0.755559	0.719346	0.460755	0.820801
Uniformity of Cell Shape	0.653470	0.907228	1.000000	0.685948	0.722462	0.713878	0.735344	0.717963	0.441258	0.821891
Marginal Adhesion	0.487829	0.706977	0.685948	1.000000	0.594548	0.670648	0.668567	0.603121	0.418898	0.706294
Single Epithelial Cell Size	0.523596	0.753544	0.722462	0.594548	1.000000	0.585716	0.618128	0.628926	0.480583	0.690958
Bare Nuclei	0.593091	0.691709	0.713878	0.670648	0.585716	1.000000	0.680615	0.584280	0.339210	0.822696
Bland Chromatin	0.553742	0.755559	0.735344	0.668567	0.618128	0.680615	1.000000	0.665602	0.346011	0.758228
Normal Nucleoli	0.534066	0.719346	0.717963	0.603121	0.628926	0.584280	0.665602	1.000000	0.433757	0.718677
Mitoses	0.350957	0.460755	0.441258	0.418898	0.480583	0.339210	0.346011	0.433757	1.000000	0.423448
Class	0.714790	0.820801	0.821891	0.706294	0.690958	0.822696	0.758228	0.718677	0.423448	1.000000

Table 1.02: Breast Cancer data correlations by feature

Seeing this table, we can see that "Uniformity of Cell Size", "Uniformity of Cell Shape", and "Bare Nuclei" have the highest correlation with determining if the patient has or doesn't have a tumor. This means that we should see these features as one of the top splits in our decision tree.

I took out 20% of the data for testing, while using the rest for training and cross validation.

Now, without further ado, let's actually run these algorithms.

Decision Tree:

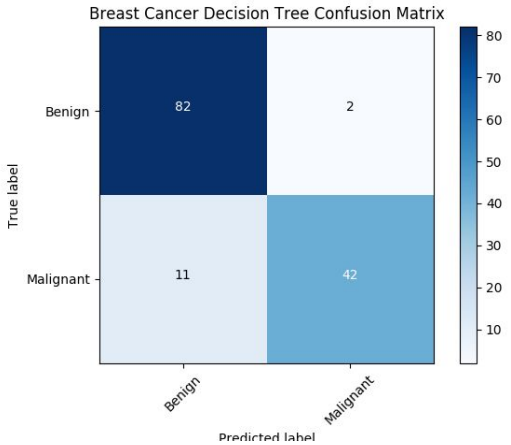
First we'll start with the Decision Tree Classifier with information gain metric, meaning we'll be splitting nodes based off how well a feature split can reduce entropy. I'll be using pre-pruning on this decision tree because it will help limit the size of the tree, making it a more general estimator. It also makes running through iterations of the decision tree faster. If I weren't to use pre pruning, it's likely that my tree could begin to overfit the training data, and hence possibly perform worse on the test data.

For the split I tried a few different options and found the following to help the Decision Tree the most.

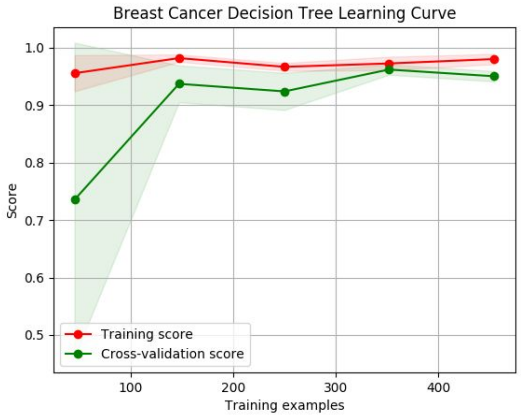
Min samples per split	Min samples per leaf	Accuracy
6	3	0.9051094890510949
1	0	0.9051094890510949
8	4	0.9197080291970803

Table 1.03: Decision Tree accuracies by min sample split and min leaf split

The following is the confusion matrix and learning curve for the decision tree.



Img 1.01: Decision Tree Confusion Matrix (Breast Cancer)

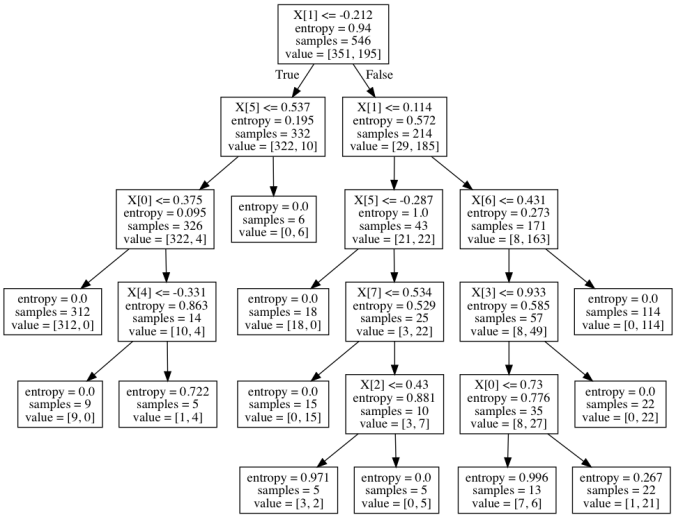


Img 1.02: Decision Tree Learning Curve (Breast Cancer)

From the confusion matrix, we can see that the decision tree is weak when it comes to predicting malignant tumors. It's eager to classify something as benign over malignant, which would be a possibly dangerous classification. This likely happens because our data split is 65% benign and if a leaf node does not meet the required number of splits, it will likely guess benign because it is a safe choice. This also means that the decision tree could have possibly split a node more to be more accurate but because of pre pruning it was limited.

I would split only if there were 8 data points per tree node, and at least 4 per split. This would help keep my tree from growing too big. If I were to let my tree grow to be bigger, it would be likely to overfit, so by limiting its size to a reasonable value, the decision tree should ideally be better at generalizing for future data.

The learning curve shows that the decision tree was not able to reach full training accuracy, which means that it might have been possible to allow a bigger decision tree that would still reach full test accuracy. The following is what that decision tree ended up looking like.



Img 1.03: Decision Tree for Breast Cancer Data

From this tree we can see that we split at the root by the second attribute ("Uniformity of Cell Size") and then in subsequent layers we split by "Bare Nuclei" very quickly. This confirms that our correlations were indeed accurate--Uniformity of Cell Size, Bare Nuclei make a big difference in making the split decision.

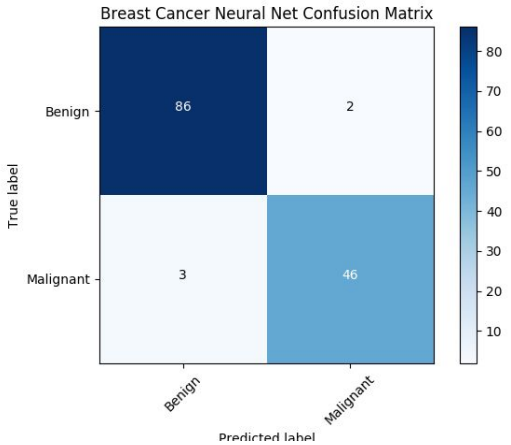
Neural Network:

The second machine learning algorithm I'll be using on this data set is an Artificial Neural Network. For this, I decided to use a neural net with one hidden layer. I tested the Neural Network with between 1 and 9 hidden layer nodes, and got the following as the best number of nodes to have in the hidden layer.

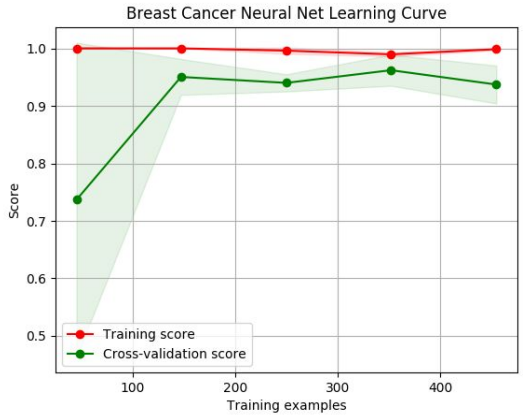
# of Hidden Layers	Accuracy
4	0.9927007299270073
8	0.9781021897810219
6	0.9635036496350365

Table 1.5: Top Neural Network accuracy for different number of hidden layers on breast cancer data

What this shows is that a neural network could do better with fewer nodes. This is likely because we have a few features that strongly influence the final classification. The following is the learning curve and confusion matrix for the 4 node hidden layer neural net.



Img 1.04: Neural Net Confusion Matrix (Breast Cancer)



Img 1.05: Neural Net Learning Curve (Breast Cancer)

From this we can see that 2 Benign classifications were misclassified as Malignant, and 3 Malignant classifications were misclassified as Benign. The neural net performed better than the decision tree, and this is likely because it is a good function approximator and could be better at finding the correlation with malignant labels than the decision tree.

The learning curve also shows that the neural net did better with more data, and consistently beat the decision tree in cross validation.

Boosting:

For boosting, I decided to use the Adaboost classifier. I decided to use a boosted version of the Decision Tree Classifier I used in the previous example so I would be able to accurately compare the effectiveness of the boosting.

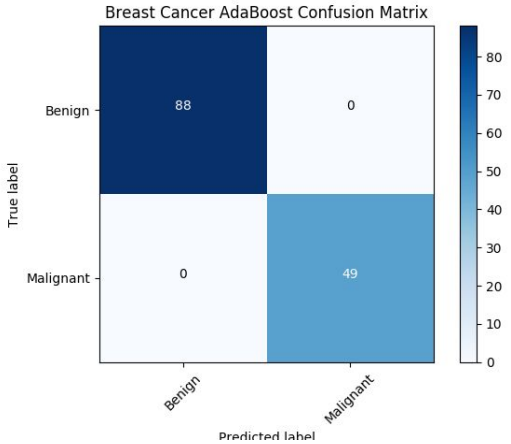
One parameter that I could change to increase the accuracy of the boosted tree was the learning rate. I decided to try 10 linearly spaced options between 0.01 and 1. I also ran the algorithm for up to 500 estimators, unless it achieved perfect accuracy, at which point the algorithm would stop.

In total I ran the algorithm with 90 options and tested using 20% of the data, and these were the top 3 learning rates.

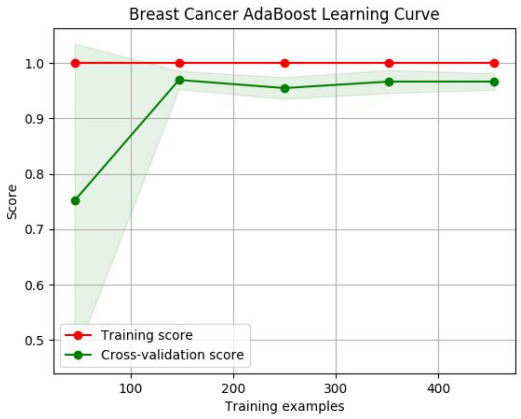
Learning Rate	Accuracy
0.56	1.0
0.23	0.9927007299270073
0.67	0.9927007299270073

Table 1.06: Accuracies with varying learning rates on the Adaboost Classifier.

The following is the Confusion Matrix for the classifier, which scored perfectly on the data.



Img 1.06: Adaboost Confusion Matrix (Breast Cancer)



Img 1.07: Adaboost Learning Curve (Breast Cancer)

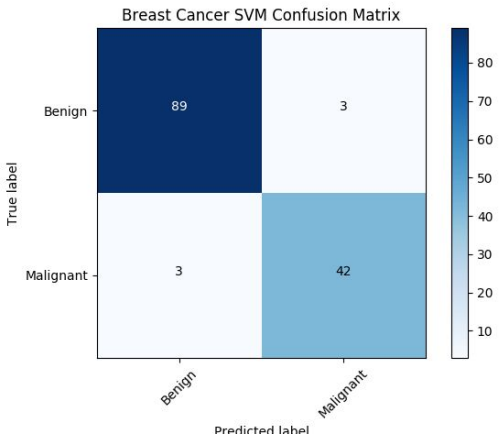
As we can see, the classifier reached perfect accuracy on the training examples, and stagnated around 95% accuracy after. This makes sense because the adaboost algorithm would train on many decision trees and when predicting, would average the results of the decision trees it used to come up with a prediction. It's able to score better than the neural net or individual decision tree because it can continue use the classifications of many algorithms to best improve its own classification, hence why it's a good ensemble learner.

SVM:

For this data set I used a support vector classifier with the RBF kernel. This is so that if there are any non linearities in the data, the SVM should be able to perform better than the other algorithms. If not, then it will likely have similar accuracies when compared to algorithms that are good with linearly separable data. RBF essentially takes the data to a higher dimension, finds the best separating hyperplane, and then reduces the dimensionality back. This allows the SVM to better classify non linear data as well.

I ran this algorithm until it met a tolerance of 1e-3 and did not set a limit of how many iterations the algorithm should stop at. I ran this 10 times, and got a best accuracy of 0.9854014598540146.

The following is the confusion matrix for that model.



Img 1.08: SVM Confusion Matrix (Breast Cancer)

From this, we can observe that the accuracy of our SVM is better than that of our decision tree. Since we are using the RBF kernel, it is likely that it performs better than the decision tree, and similarly to the neural net because there may be some non linearity in the data.

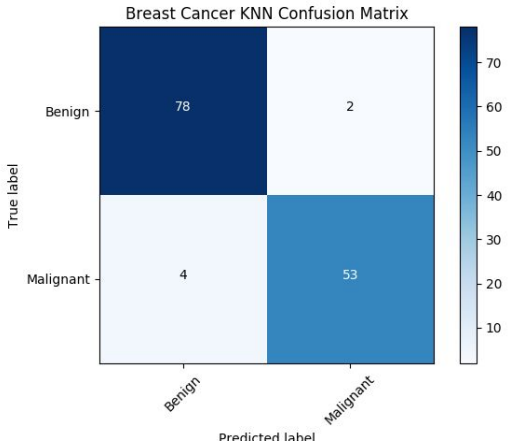
KNN:

For the last algorithm, I will be using K nearest neighbors. The most significant parameter that can be changes in KNN is the number of neighbors needed to determine a point's classification. For this I wanted to test out values between 1 and 10. I chose these because I wanted to see how many neighbors makes the most effective algorithm.

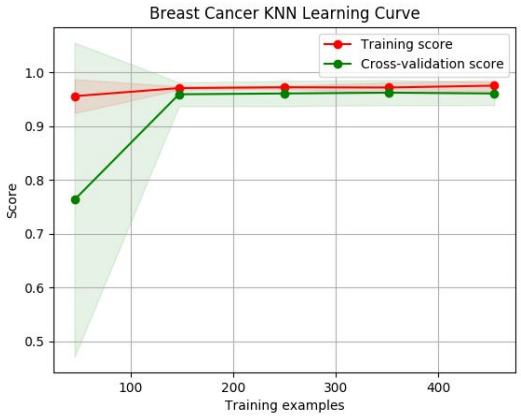
Neighbors	Accuracy
1	0.9416058394160584
2	0.9416058394160584
3	0.9635036496350365
4	0.9708029197080292
5	0.9781021897810219
6	0.9854014598540146
7	0.9562043795620438
8	0.9781021897810219
9	0.9781021897810219
10	0.9635036496350365

Table 1.7: KNN accuracies with varying number of neighbors

Hence from this I decided to make the number of neighbors 6, as it had the highest test accuracy. The following is the confusion matrix for the model with 6 neighbors as its metric.



Img 1.09: KNN Confusion Matrix (Breast Cancer)



Img 1.10: KNN Learning Curve (Breast Cancer)

The training and cross validation accuracy are very similar. This makes sense because points that are mislabeled are very likely to be equidistant between the two classifications. There are not very many of them and assuming an even distribution of them, then they should have roughly the same accuracy on test and cross validation. This is also an indication that the algorithm didn't overfit on the training data.

Compared to the other algorithms, this is the one that likely least overfit to the data and would generalize well, however the accuracy could be improved if we selected a different number of neighbors and had more data to support it.

Data Set 1 Conclusion:

For this particular dataset I achieved the highest accuracy with the Adaboost classifier (100% accuracy), because as an ensemble method, it was able to take advantage of the different classifications of its decision trees. The second best is the neural network (99.27% accuracy), because it was able to take advantage of all the data, and had enough hidden nodes to express the data, though this is tricky because it is very easy to overfit or have a ton of iterations/increased wall clock time because of how many variations of neural network architecture you could have. The third and fourth best are the SVM and KNN (98.54% accuracy). These both take advantage of the non linearities in the data and are able to generalize well but likely have problems with edge case scenarios that the neural net and adaboost classifier were able to edge them out on. The worst overall was the decision tree (92% accuracy) and this is likely because of how many variety of splits I could choose, but with a chance of overfitting.

Dataset 2:

For the second data set I will be using a lung cancer data set that is very ill conditioned. I chose this dataset specifically because I wanted to see how the algorithms would perform with very few training instances, and a large number of features.

The data set has 32 instances and 57 features, however I will be removing 5 of those points because of missing data. This means that we'll have a total of 27 data points and 57 features. I will be referring to the features by the number feature they are.

The output classification of the cancer type is an integer 1 and 3 with roughly equal distributions representing the class of cancer it is. The following summarizes this.

Classification	Number

Class 1	8
Class 2	10
Class 3	9

Table 2.01: Number of each classification for lung cancer dataset

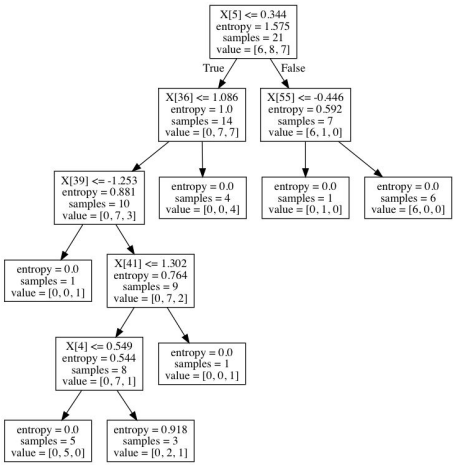
Decision Tree:

I will be using pre pruning for this Decision Tree. To figure out the best split, I ran through leaf splits from 1 through 9, and noted the top three results on 20% of the data reserved for a test set.

Split Number	Accuracy
3	0.8333333333333334
7	0.6666666666666667
5	0.6666666666666667

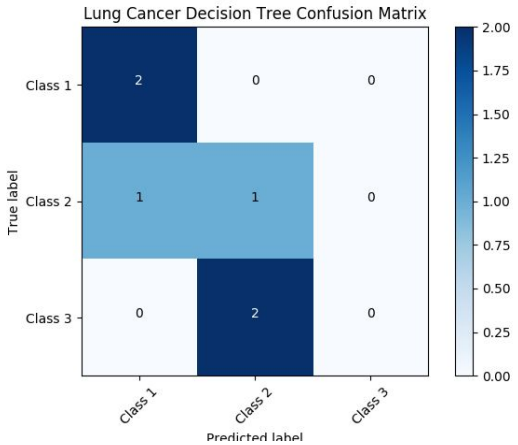
Table 2.02: Accuracy given different splits for decision tree

The estimator was best when we limited it to needing 3 samples to split a leaf node, and at least 1 sample per node. The decision tree is visualized below.

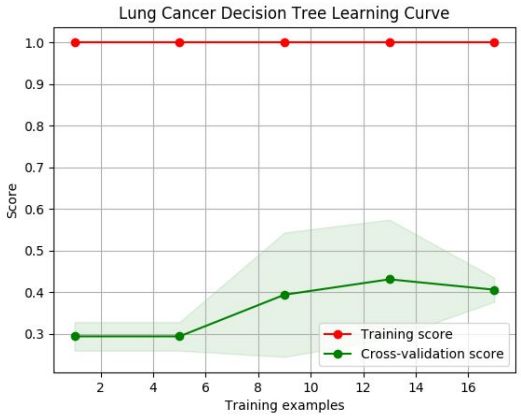


Img 2.01: Decision Tree for lung cancer data

The best features that had the highest influence on the classification were feature 5, 36, 55 because they reduced the entropy the most between layers. The following is the confusion matrix and the learning curve.



Img 2.02: Decision Tree Confusion Matrix (Lung Cancer)



Img 2.03: Decision Tree Learning Curve (Lung Cancer)

What we can see is that the decision tree struggled to classify the points correctly but was able to receive a modest peak of about 45% accuracy. This is better than the expected 33% random chance accuracy for 3 features.

The decision tree performed poorly in the confusion matrix as it missed 3 classifications in total, 1 from class 1, and 2 from class 2. The problem was ill posed, and so the decision tree likely couldn't split edge cases.

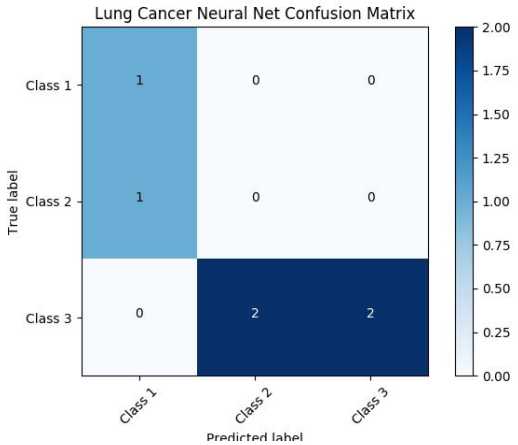
Neural Networks:

For the neural network, I decided to try out 1 hidden layer, and vary the number of nodes from 1 to 19 nodes (up to the number of features divided by 3). The following were the top 3 neural net architectures.

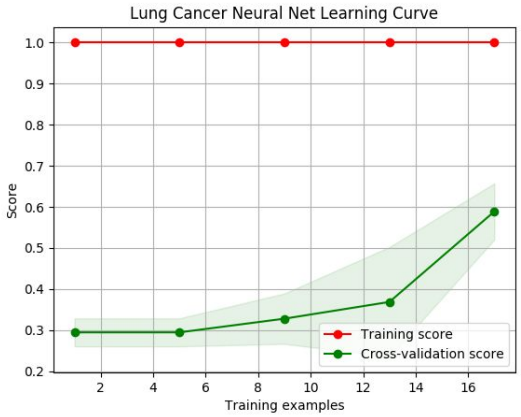
Nodes	Accuracy
4	0.8333333333333334
9	0.8333333333333334
13	0.8333333333333334

Table 2.03: Top 3 accuracies for neural networks with different number of hidden nodes

All these architectures achieved the same accuracy, and in order to make the neural network best with generalization, considering the number of features, I will be taking the neural network architecture with 13 nodes. This is because a neural network with more nodes will be able to better generalize to the number of features. With 57 features, it would be best to use a network that can express as many of those features as accurately as possible. The following is the confusion matrix and learning curve.



Img 2.04: Neural Network Confusion Matrix (Lung Cancer)



Img 2.05: Neural Network Learning Curve (Lung Cancer)

As expected with the neural net, it performs far better when there is more data. This cross validation score significantly trumps that of the decision tree's. The number of hidden nodes also likely helps express the data better because it has more degrees of freedom to work with.

It however, did get the same accuracy as the decision tree because it was poor at classifying class 1 and class 2. The neural network performs best with more data, and again likely couldn't generalize based off of the little amount of data. It was likely that edge cases or non linearities prevented the neural network from being able to generalize well on such little data.

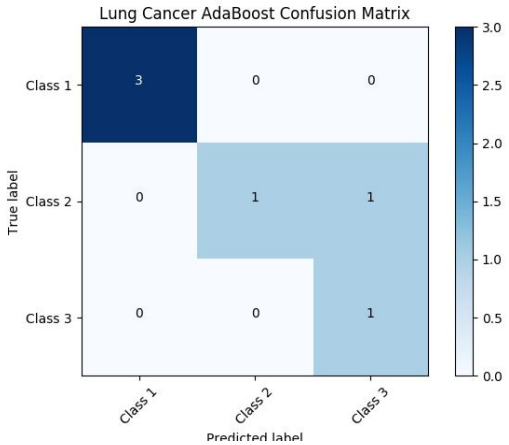
Adaboost:

For this, I have used the same decision tree estimator for this dataset, and let the number of iterations go till 500, or until it achieves a perfect score on the train data. I also again experimented with the learning rates of the boosted decision tree. Here are the results:

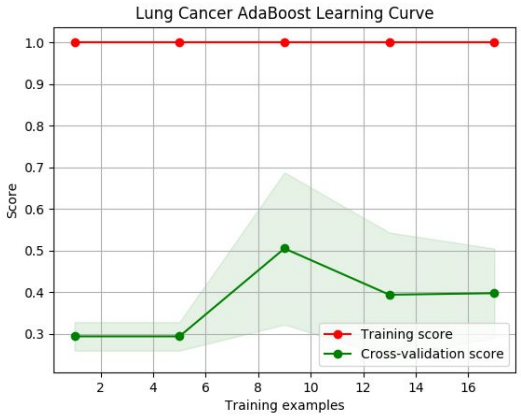
Learning Rate	Accuracy
0.45	0.8333333333333334
0.34	0.8333333333333334
0.01	0.8333333333333334

Table 2.04: Top 3 accuracies for Adaboost with varying learning rates

In order to have my algorithm achieve good accuracy in a reasonable amount of time, I'll be using the 0.34 learning rate. The following are the confusion matrix and learning curve.



Img 2.06: Adaboost Confusion Matrix (Lung Cancer)

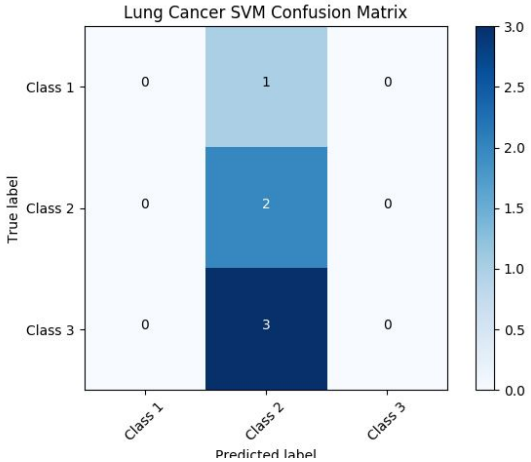


Img 2.07: Adaboost Learning Curve (Lung Cancer)

The Adaboost algorithm performed well on the confusion matrix, only getting 1 wrong, but had problems during training, likely because it was limited to the small sample of data. The classifier only had one error with class 3, but this could also be a fluke considering how poorly it performed in the learning curve.

SVM:

For the SVM, I used the best algorithm selected by the SVC function. The result was a very poor accuracy of 0.3333333333333333.

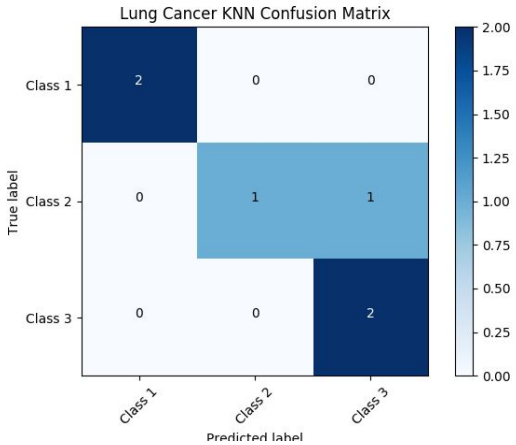


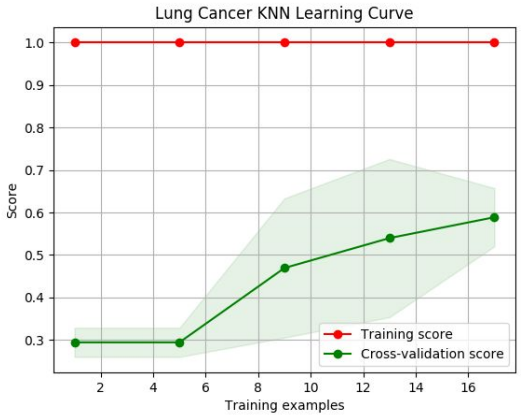
Img 2.08: SVM Confusion Matrix (Lung Cancer)

This is very poor likely because it is difficult for a separator to be passed through a data set with this many features, and hence it found a bad separating hyperplane. The algorithm selected was likely linear and only predicted class 2. This makes it a very bad algorithm for higher order data sets with little data, despite SVMs being generally good when there's higher order data.

KNN:

For this algorithm, there was little opportunity to play with the number of neighbors because the test set would have to have enough neighbors to successfully classify a point. Hence for this, I'll be training the KNN algorithm with only 1 neighbor.





Img 2.09: KNN Confusion Matrix (Lung Cancer) *Img 2.10: KNN Learning Curve (Lung Cancer)*

The single nearest neighbor algorithm did surprisingly well on cross validation and on the confusion matrix (83.34% accuracy), only misclassifying 1 point in class 3. KNN would likely be a strong contender along with neural networks and decision trees for this sort of classification. The nearest neighbor algorithm was also extremely fast compared to the other algorithms because of only needing to find a single nearest neighbor rather than many nearest neighbors. This performed on par with all other algorithms except the SVM.

Data Set 2 Conclusion:

Overall, a lot of the algorithms performed the same, however, I would say that the Adaboost algorithm and KNN performed the best because of the variety of classes they were able to predict and having a strong cross validation learning curve. This is because the ensemble algorithm for adaboost is able to take advantage of the strengths of multiple algorithms. KNN performed well because it simply has to find the nearest neighbor, which was a good enough heuristic for this dataset, and likely would have been able to do well even if there were significantly more features. The neural network performed well, but struggled on the confusion matrix, and has a high tendency to overfit. The decision tree was good, but also struggled with cross validation, likely because of how many features there were. Since it is a binary split on each node, it must select the best ones to reduce entropy, which wasn't very easy to do. Finally the SVM performed the poorest overall. This is because the automatic algorithm chosen was likely linear and attempted to classify everything as a single class, essentially performing as good as random guessing. The SVM would perform better with more data, however it struggled because there were too few instances for how many features there were.

Overall Conclusions:

The algorithms all performed well, but there were clear advantages to having one over another. For large datasets with lots of data, KNN, neural nets, and adaboost tended to perform well because they are good at drawing out the edge cases of data well and generalizing over a lot of points. For small datasets, Adaboost and KNN tended to do the best because they used advantage of ensemble methods (adaboost) or that data points tended to be classified together if they were close together in a hyperplane. The decision tree was an overall not bad option, but needed to focus on proper pruning. The SVM had problems with very little data, but would be able to generalize well if it were given enough data, and is very good with non linearly separable data.

Citations:

UCI Machine Learning Repository: https://archive.ics.uci.edu/ml/index.php

Scikit Learn: http://scikit-learn.org/