Datasets

I wanted to explore the differences between a medium-sized dataset and a large dataset when implementing the various machine learning algorithms. Both datasets were found in the UCI Machine Learning Repository [1].

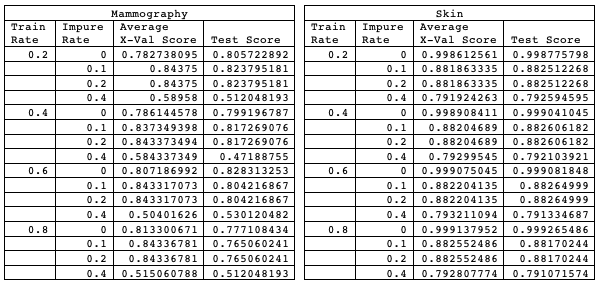
The first one chosen is based off of mammography data that is used to identify whether a tumour is malignant or benign [2]. This dataset has 6 attributes with a binary output and consisted of 961 samples. I chose this dataset because of my interest in healthcare and because I believe the application of machine learning has the potential to play a massive role in improving patient care. The second sample chosen was of identifying whether a particular RGB value corresponds to a skin colour [3]. This dataset only has 3 attributes and the output is also binary, however the number of samples is 245,057. My main reason for choosing this dataset was to contrast its large size to the mammography data. Its binary output also helps maintain consistency between the datasets.

Only the mammography data required pre-processing. It was reduced to 830 total samples due to missing values for some of the attributes. Some of the attributes, such as shape, were nominal and needed to be broken into multiple attributes to prevent some algorithms, like neural networks, from weighting one “better” than the other. I broke shape down into 4 mutually exclusive binary attributes: round, oval, lobular, and irregular. This way, “round” won’t be valued as a lower weight than “oval” if its index value was simply less in the mapping. I followed the same procedure for margin and density. Finally, in order to prevent the attributes from being far too specific, I transformed the age attribute into a decade attribute. This would place, for example, ages 60-69 into the 60 decade.

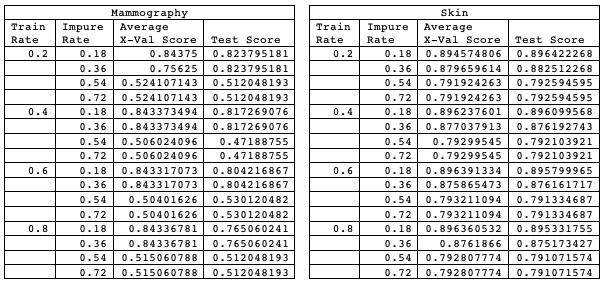
Experiments

The data was processed using the Scikit-learn library for Python 3 [4]. In each run, the training set made use of a 4-folded cross-validation procedure and the average was compared with the results of the testing set.

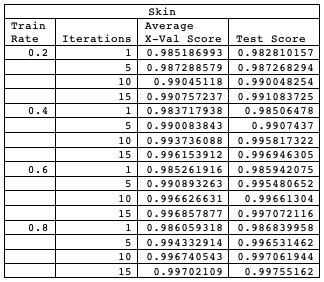
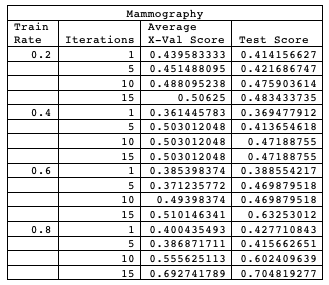
The decision tree algorithm used information gain to split the data because the difference in performance between it and Gini is marginal when the two are compared over a wide range of practical applications [5]. The abundance of documentation on information gain swayed me to choose it in the end. Pruning was applied by preventing further splits if the level of impurity that may result is less than some threshold. The results were as follows.



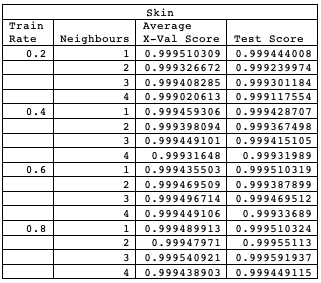
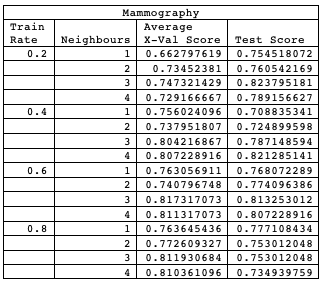
The boosted tree version of this made use of the AdaBoost algorithm to focus in on attributes that have more predictive power. By boosting the weights, I expected to be able to get away with a larger impurity allowance without sacrificing too much accuracy. The results below mostly confirmed my hypothesis.



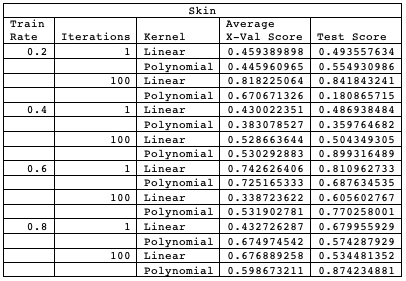
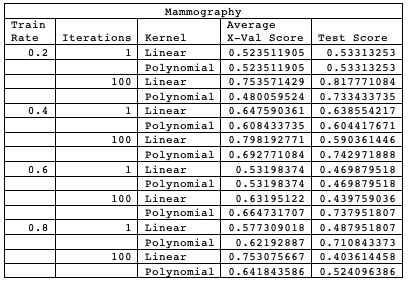
When choosing the implementation for the neural network experiments, I decided to use the logistic sigmoid activation function paired with a stochastic gradient descent solver. ReLU activation was also considered but it was ultimately not chosen because it has the possibility to cause a detriment to learning when paired with this solver in that gradients cannot flow backwards [6]. The results over several different iterations are below.



For the KNN algorithm, I chose a uniform weight distribution because the spread of values was large in both datasets and there was a large potential for over-fitting otherwise (especially on the skin data). For example, just because a specific RGB in the brown range isn’t a skin sample shouldn’t mean that a close RGB should be classified as a non-skin sample as well, especially since we know that there are a wide variety of brown tinges that do correspond to skin colours. The results when varying the number of neighbours are below.



For SVM learning, the kernels chosen were linear and polynomial of degree-2. Since linear is simply a degree-1 polynomial, I wanted to compare it with the next logical progression and analyze the results. For each training rate, the different kernels were applied with a single iteration and 100 iterations to record the effect of repetition on the datasets. The results are as follows.



Analysis

To ease readability in this section, the average cross-validation scores are joined by dashed lines while the test scores by solid lines.

The decision trees for the mammography data had the least error on the testing data when the training rate was 60% for impurities 0 and 0.4. This strongly suggests that beyond this rate tended to overfit the data for these pruning levels. There is further evidence of this fact by the trend of the error rate decreasing on the testing data as the training portion rises. In contrast, the best training rate for impurities 0.1 and 0.2 was only 20%. Beyond this point, the error rate on the testing data hardly increased. So while the best single run was on impurity level 0 with an error rate of 18.2%, both impurity levels 0.1 and 0.2 only needed a 20% training portion to accomplish a very close error rate of 18.6%. This leads me to believe that very little score is sacrificed by pruning and the gains in performance can far outweigh the deficit.

As opposed to the mammography output, the error and testing scores were very similar in all test runs for the skin data. The un-pruned tree was the significantly better structure in this case with a <0.1% error rate on both the training data and the testing data as it gets a higher training portion. Impurities 0.1 and 0.2 are tied in scores once again across all training rates. Their lowest error rate is at 11.8% however it’s very close for all training portions. Very little is sacrificed if only 20% of the data is used to train in this case, for all impurity levels. This may be attributed to the fact that there are only 3 features and that skin colour is very well-defined typically. Colours that are green, yellow, and red can be easily identified as not representing usual skin appearance and the decision tree algorithm does a fantastic job of identifying this with a small training portion.

The most noteworthy change in the results when using AdaBoost on the mammography data was that even with a high impurity level like 0.36, the algorithm was able to have an error rate of 18.6% on the testing set at a 20% training rate. This is the same as the error rates at impurity levels 0.1 and 0.2 before. The boosting seemed to have allowed additional impurities to creep in to the splits without forgoing any accuracy at all in this case. However, as the boosting is stressed even more with impurity levels 0.54 and 0.72, the excessive pruning hugely detriments score, even dipping below 50% at its worst.

Boosting on the skin data was ineffective and, in the case of no pruning, yielded much worse results if anything. This may be due to the fact there’s no one important attribute out of red, green, or blue, and so there would be nothing to gain out of weighting these. Weighting them would be counter-intuitive in this regard, which would result in inaccurate labels.

The neural network results on the mammography dataset are unsurprising in that the error rate on the testing set decreases as more iterations are added, with the exception of 5 iterations near the end. What did surprise me was how poorly it did in general. The lowest error rate was 29.5% over both training error and testing error. There was also a sharp rise in training error for 5 iterations between 0.4 and 0.6 training rate. Since the testing error dropped in this interval, there may have been some misrepresenting of the data occurring because that would explain the underfit.

In contrast, the neural network scored extremely well on the skin dataset. The error rates were as low as 0.3% on the training data and 0.2% on the testing data. There was also no indication of overfitting, the scores simply improved as more training was provided.

KNN on mammography data was the most successful with 20% of the samples used for training with 3 neighbours. The largest drop in error rate was 9.2% for K = 1 between the 20% and 40% training portion. This was the training error drop and leads to the conclusion that the extra 20% training was much more important earlier on. Especially when considering that training error dropped noticeably for K = 3 and K = 4 in the same interval.

KNN on the skin data was, simply put, the most successful of all the experiments. The error rate in the worst case was <0.1% and while the best score was for the K = 3 test data, the difference was marginal. The success of this experiment could be attributed to the size and spread of the dataset such that there was always a close neighbour nearby.

The SVM approach yielded a great deal of variability in the results on the mammography dataset. The polynomial kernel at 100 iterations was the most consistent, hovering at around a 25% testing error rate until overfitting began after increasing the training portion past 60%. However, the lowest error rates were for the linear kernels at 100 iterations, even though overfitting had a massive effect on the testing error rate as the training portion increased.

This approach on the skin data had even more variability, the most seen in all the experiments. The most immediately noticeable item of interest is the testing error dropping close to 70% when increasing the training portion from 20% to 40% for the 100 iteration polynomial run of the algorithm. This experiment demonstrated very well how important the training portion can be and how quickly scores can very based off of this parameter. At its worst, the testing error was close to 80%, that’s impressively bad considering that random chance at labelling correctly is 50%.

Discussion