Adult Income and Letter Recognition - Supervised Learning Report

An objective look at classifier performance for predicting adult income and Letter Recognition

Dudon Wai
Georgia Institute of Technology
CS 7641: Machine Learning
Atlanta, GA
dwai3@gatech.edu

Abstract: This report presents an analysis on the performance of 5 classification algorithms tested on two UCI datasets, "Adult Income" and "Letter Recognition". A statistical analysis is presented on both problems, followed by individual analyses for the following algorithms: Decision Trees, Decision Trees with Adaptive Boosting, k Nearest Neighbors, Artificial Neural Networks, and Support Vector Machines.

Introduction

There are two types of machine learning problems: Classification and Regression. The purpose of this paper is to explore two classification problems and evaluate them using a variety of algorithms. The Adult Income and Letter Recognition datasets were selected from the UCI repository based on relevance to the fields of study and complexity to compare each algorithm. [1][2]

Dataset Information

Preprocessing

The quality datasets have not been modified. The Adult Income dataset was converted from a .csv to .arff, and the Letter Recognition dataset was converted from .data to .arff, both for processing in WEKA. In WEKA Explorer, both datasets have been separated into training, validation and test sets using RemovePercentage. The histograms **Figure 1** below show the existing distribution of the full datasets prior to splitting, which were used as a reference for additional segmented datasets.

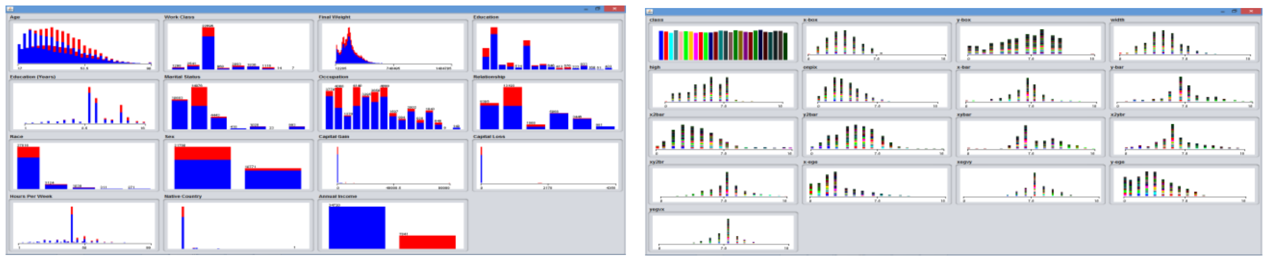


Figure 1: Histograms of Dataset 1 (left) and Dataset 2 (right)

Dataset 1: Adult Income

In the last decade, the global population living in poverty (defined as living with less than $2 per day, 1985 prices) has decreased dramatically from 80% in 1920, to 50% in 1970 to 10% in 2015 [3]. Similarly, the standard of living in nations globally is on the rise [4]. National policy plays a role in achieving this prosperity. In order for governments to determine changes in policy, it uses data to measure previous and current states. A primary resource for governments is census data, which collects socio-economic details of the population. Dataset 1 is a subset from the 1994 US Census, which is used to relate education, heritage and age (among others) against income, in this case, whether income is above or below $50,000 per year. Governments can use this data to determine the most impactful factors for increasing household income.

The dataset consists of 2 classes (<=$50k, >$50k), 14 socio-economic attributes, and over 30,000 instances which allows for sufficiently large subsets when splitting the overall dataset into training, validation and test sets. In terms of machine learning, Dataset 1 is interesting because the algorithms perform very similarly, all achieving near 85% accuracy. This may be due to an uneven distribution within the output class (24,720/7,481) as observed from the histogram. However, this is common for most datasets. In addition, the factors for household income may require more attributes than in the dataset. However this may introduce the curse of dimensionality.

Dataset 2: Letter Recognition

Computer vision is a fast-growing field within machine learning as algorithms, hardware and cloud computing are finally coming together to make technologies viable, such as virtual reality, augmented reality and autonomous vehicles. Within the field of computer vision, optical character recognition (OCR) plays an important role in the advance of technology. Many industries such as healthcare, finance, law and construction have used OCR to help with paperwork reduction, process improvement and task automation. A study of the accuracy of OCR is important to the computer vision industry, as it is more mature and can be used as a guide when developing for more difficult sub-domains like video tracking and object recognition.

The letter recognition dataset contains 26 classes (one for each letter in the alphabet), 16 attributes (position, length, statistical moments), and 20,000 instances of user-generated letters based on a variety of fonts. The size of the dataset allows for proper segmentation into training, validation and test sets. Dataset 2 is interesting with respect to machine learning because the performance of the algorithms can be radically different. This creates an opportunity to explore under what circumstances certain algorithms behave better than others.

Algorithm Implementation

All analyses performed in this summary were done using machine learning tools available for WEKA GUI. All algorithm results presented were found using 10-fold cross validation unless specified. To evaluate each of the 5 classification algorithms, the overall dataset was split into training, validation and test sets, as explained below and shown in **Figure 2**.

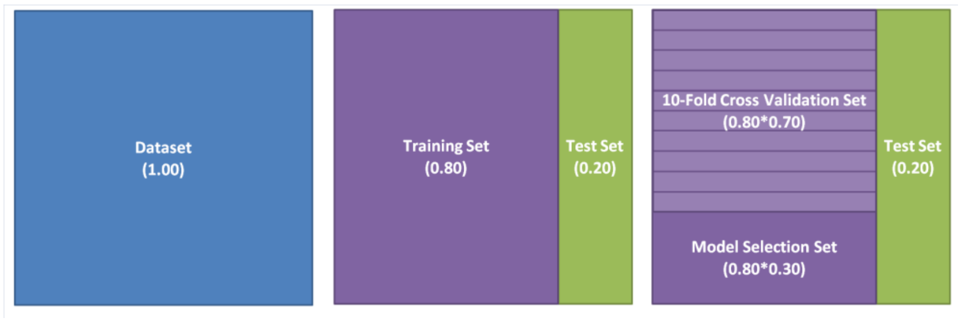


Figure 2: Training, Validation and Test Set Split for Dataset 1 and 2

To evaluate each of the 5 classification algorithms, the overall dataset was split the data 80/20 into training and test sets. The training set was used to tune each algorithm, and the test set is to remain untouched during all experiments until a model has been selected for each algorithm. Then the final performance will be evaluated on the test set.

From the 80% training set, it will be further split 70/30 into cross validation and model selection sets. The cross validation set will employ 10-fold cross validation to remove the bias of selecting a model that performs well on training data, but does not generalize well. The model selection set is a test set to evaluate model complexity and learning curves.

Since both datasets are classification problems, the performance of each algorithm will be based on accuracy (% correct instances) rather than the root mean squared error (RMSE). Before evaluating the various algorithms, note that to perform better than chance, the accuracy of each algorithms must be greater than 50% for Dataset 1 (1/2) and ~4% for Dataset 2 (1/26).

There are some biases to be aware of when observing the algorithms. With respect to model selection, the parameters are evaluated rules of thumb at first, which may influence the selection in further tests. Also, for example, computation time is independent of model accuracy, but this parameter played a part in how the series of experiments were constructed. In addition to these restrictive biases, there are preference biases towards simplicity, correctness, locality, smoothness and attribute equality.

Decision Tree

The WEKA J48 classifier ("J" indicates Java, and "48" indicates C4.8 – an extension of the C4.5 algorithm) was used to evaluate the decision tree. Pruning was employed to observe the effect of removing less relevant branches on the reduction of overfitting the training set. Pruning was conducted by manipulating the confidence factor, C, and the minimum number of outputs, M. A lower confidence factor helps induce pruning but may decrease accuracy, which was counter-balanced with 10-fold cross validation.

The decision tree algorithm is an eager learner, as it builds the model from the training data, which is then used for the test data. However, decision tree has relatively low computation time for training and testing compared to other eager learners (ANN, SVM).

Dataset 1: Adult Income

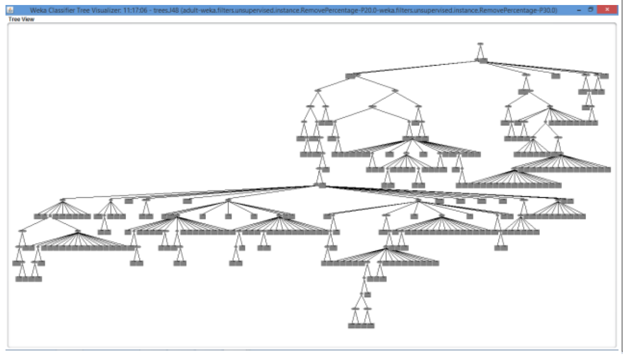
Table 1 below summarizes the model complexity experiments. Note the reduced size of tree when various degrees of pruning are applied. The results also demonstrate that decision trees are eager learners, since the build time is more than 10x the testing time. Lastly, the training and test accuracy fall within 2% of each other, indicating a small reliance on the input parameters. The chart shows that the parameters confidence, C = 0.25 and minimum outputs, M = 3 output the highest accuracy.

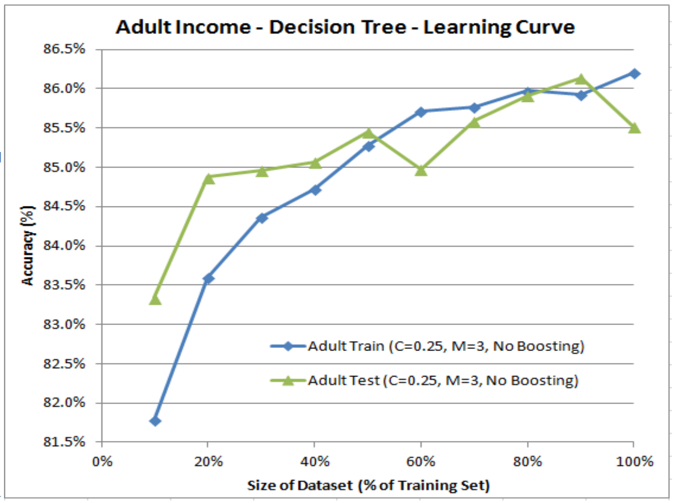
Table 1: Adult Income – Decision Tree – Model Complexity Experiments

Confidence, C	Minimum Outputs, M	Prune?	# Leaves	Size of Tree	Build Model Time	Test Model on Test Data	Train % Correct	Test % Correct
--	2	N	4485	5254	1.11	0.06	83.81%	83.20%
0.5	2	Y	1518	1868	0.95	0.08	85.06%	84.56%
0.25	2	Y	366	471	0.94	0.11	86.25%	85.20%
0.125	2	Y	170	227	0.94	0.09	86.14%	85.44%
0.25	3	Y	267	352	0.78	0.09	86.21%	85.52%
0.25	4	Y	218	286	0.7	0.06	86.17%	85.44%

Below, the confusion matrix and the decision tree Dataset 1 are shown. The decision tree shows the effect of pruning on the size of the tree. The root node of this tree is “capital gain”. The confusion matrix is a simple 2x2, the dimension of the matrix determined by the number of classes in the output.

```
    a    b   <-- classified as
 1280   69 |    a =  <=50K
  134  340 |    b =  >50K
```




Adult Income - Decision Tree - Learning Curve
86.5%
86.0%
85.5%
85.0%
84.5%
84.0%
83.5%
83.0%
82.5%
82.0%
81.5%
Accuracy (%)
0%
20%
40%
60%
80%
100%
Size of Dataset (% of Training Set)
Adult Train (C=0.25, M=3, No Boosting)
Adult Test (C=0.25, M=3, No Boosting)

Learning curve experiments were performed to determine the sensitivity of the algorithm's results to the size of the dataset. As the following graph shows, the training and test sets behave similarly for most dataset sizes. It is suspected that the test data sometimes performs better than the training data because the test set is larger for training sets up to 30-40%.

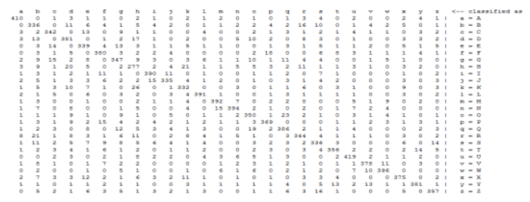
Dataset 2: Letter Recognition

The chart below summarizes the model complexity experiments. As compared with Dataset 1, the size of the tree is not reduced as much, likely due to the larger number of output classes. Again, the results show that decision trees are eager learners (although still compute is relatively show time frames). The training and test accuracy fall within 1% of each other, indicating a small reliance on the input parameters. Thus, the standard parameters confidence, C = 0.25 and minimum outputs, M = 2 were selected for the learning curve experiments. Note the high training accuracy for M=4, and the resulting **lower** test accuracy, indicating overfitting, despite using 10-fold cross validation.

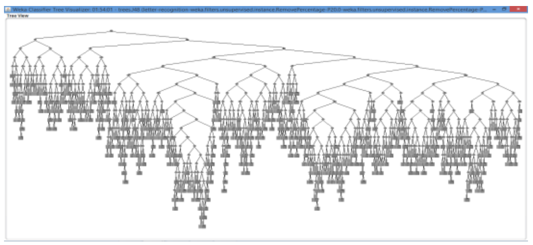
Table 1: Letter Recognition – Decision Tree – Model Complexity Experiments

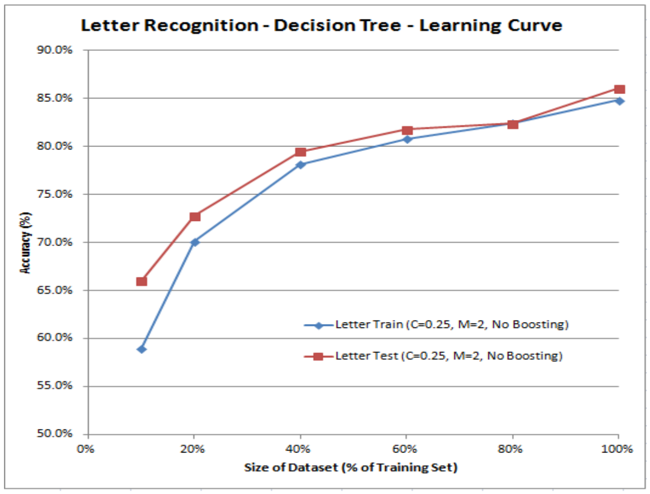
Confidence, C	Minimum Outputs, M	Prune?	# Leaves	Size of Tree	Build Model Time	Test Model on Test Data	Train % Correct	Test % Correct
--	2	N	951	1901	0.88	0.11	84.76%	85.90%
0.5	2	Y	895	1789	1.05	0.09	84.88%	86.06%
0.25	2	Y	868	1735	1.01	0.13	84.80%	86.04%
0.125	2	Y	816	1631	1.13	0.09	84.81%	86.04%
0.25	3	Y	710	1419	0.96	0.09	84.50%	85.69%
0.25	4	Y	600	1199	0.96	0.08	93.09%	84.33%

The confusion matrix and decision tree for Dataset 2 are shown below. The decision tree shows the effect of pruning on reducing the size of the tree. Pruning must balance avoiding overfit, while also maintaining sufficient complexity to properly model the dataset. The root node of this tree is "x-ege", or the mean edge count left to right. The confusion matrix is 26x26 based on the 26 letters in the alphabet.



```
   a   b   c   d   e   f   g   h   i   j   k   l   m   n   o   p   q   r   s   t   u   v   w   x   y   z   <-- classified as
 410   0   1   3   1   1   0   2   1   0   2   1   2   0   1   0   1   3   4   0   2   0   0   2   4   1 |   a = A
   0 336   0  11   6   4   1   5   4   2   0   1   1   2   2   4   2  16  10   0   1   4   2   5   0   1 |   b = B
   3   2 342   0  13   0   9   1   1   0   0   4   0   0   2   1   3   1   2   1   4   1   1   0   3   2 |   c = C
   3  13   0 381   0   1   2  17   1   0   2   0   0   5  10   2   0   8   3   0   1   0   0   3   3   2 |   d = D
   0   3  14   0 339   4  13   3   1   1   5   1   1   0   0   1   3   1   5   1   1   2   0   5   1   5 |   e = E
   0   3   1   5   0 380   3   2   2   4   0   0   0   0   2  18   0   0   6   8   3   1   1   1   4   1 |   f = F
   2   9  15   2   8   0 347   9   3   0   3   6   1   1  10   1  11   4   4   0   0   1   5   1   0   0 |   g = G
   3   9   1  20   5   0   2 277   2   4  21   1   1   5   5   3   2  11   1   1   3   1   0   3   2   0 |   h = H
   1   3   1   2   1  11   1   0 390  11   0   1   0   0   1   1   2   0   7   1   0   0   0   1   0   2 |   i = I
   2   5   1   3   3   6   2   2  15 335   4   1   2   0   1   0   3   1   4   2   0   0   0   3   0   3 |   j = J
   1   5   3  10   7   1   0  26   0   1 332   0   0   3   0   1   1   6   0   3   1   0   0   9   0   3 |   k = K
   2   1   5   0   6   0   3   2   0   3   4 391   1   0   0   1   3   1   1   1   1   0   0   3   0   2 |   l = L
   1   3   0   0   1   0   0   2   1   1   4   0 392   7   0   2   2   0   0   0   5   1   9   0   2   0 |   m = M
   1   7   0   8   0   0   1   5   0   0   4   0  15 394   2   1   0   2   0   1   7   2   4   0   0   0 |   n = N
   1   1   1   9   1   0   9   1   0   5   0   1   1   2 350   1  23   2   1   0   3   1   4   1   0   1 |   o = O
   1   3   1   9   2  15   4   2   4   2   1   2   1   1   3 369   0   0   0   1   1   2   3   1   3   1 |   p = P
   1   2   3   0   8   0  12   5   3   4   1   3   0   0  19   2 386   2   1   1   4   0   0   0   2   3 |   q = Q
   8  21   1   8   3   1   6  11   0   2   8   4   1   5   1   0   3 344   4   1   1   1   0   3   0   2 |   r = R
   1  11   2   5   7   9   8   5   6   4   1   4   0   0   3   2   3   2 336   3   0   0   0   6   0  14 |   s = S
   1   2   3   4   1   6   1   2   0   1   1   2   0   0   2   3   0   3   4 356   2   2   0   2  14   5 |   t = T
   0   0   2   3   0   2   1   8   2   2   0   4   3   6   5   1   3   0   0   2 419   2   1   1   2   0 |   u = U
   1   8   1   0   1   7   2   2   0   0   0   0   1   2   3   1   2   1   0   1   1 378  11   0   3   0 |   v = V
   0   2   0   0   1   0   5   1   0   0   1   0   6   1   6   0   2   1   2   0   7  10 396   0   0   0 |   w = W
   2   7   3   3  12   2   1   6   3   2  11   1   0   1   0   1   0   3   3   4   0   0   0 375   0   2 |   x = X
   1   1   0   1   1   2   1   1   0   0   3   1   1   1   1   1   4   0   5  13   2  13   1   1 381   1 |   y = Y
   0   5   2   1   6   3   5   1   3   2   1   3   0   0   1   1   6   3  16   1   0   0   0   5   0 357 |   z = Z
```


Weka Classifier Tree Visualizer: 01:54:01 - trees.J48 (letter-recognition-weka.filters.unsupervised.instance.RemovePercentage-P20.0-weka.filters.unsupervised.instance.RemovePercentage-P...
Tree View

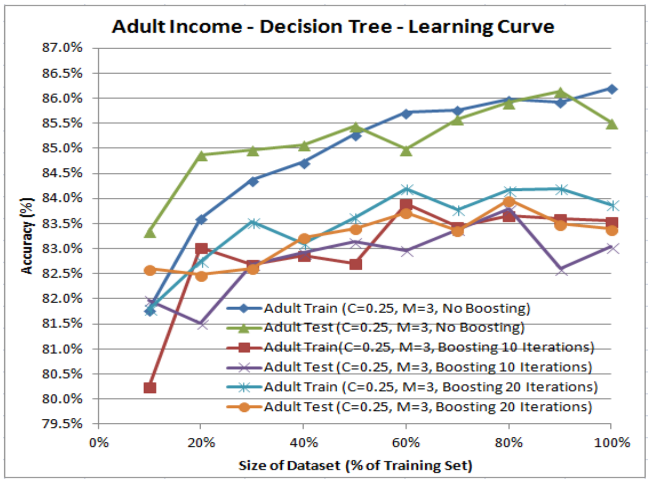


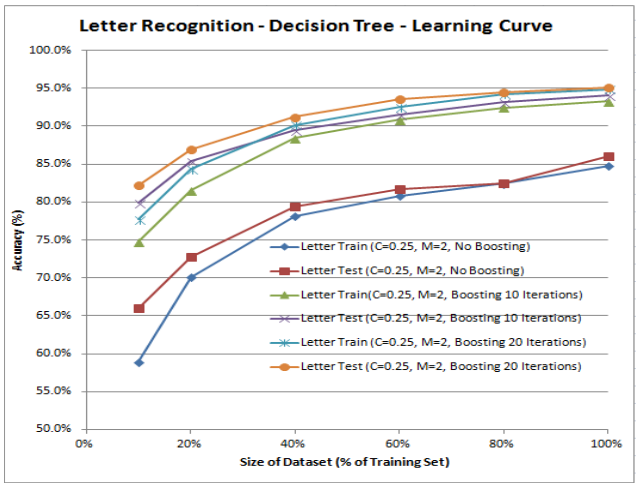
Learning curve experiments were performed to determine whether the algorithm was sensitive to the size of the dataset. The results are very predictable, as seen above, where the training and test accuracy follow closely, and increase with larger of the algorithm's results to the size of the dataset.

Decision Tree with Adaptive Boosting (AdaBoostM1)

The WEKA classifier AdaBoostM1was applied to the J48 decision tree algorithm to evaluate the effect of boosting. Boosting is an iterative process that applies information gain to learn over a subset of data. The process can be applied to other machine learning algorithms, and is observed with J48 in this paper.

Although the decision tree algorithm is normally a lazy learner, applying adaptive boosting to decision tree increasing the learning process by at least an order of magnitude. However, this is a welcome trade-off as the accuracy of the algorithm in Dataset 2 improved dramatically from 85% to 95%. In **Figure 6** below, the learning curves for decision trees with adaptive boosting are presented, for both datasets.





Dataset 1: Adult Income

The chart expands on the learning curve experiment from the previous section for decision trees. Similar curves were produced for boosting with 10 and 20 iterations and plotted together with the non-boosted results. It is observed for this dataset that boosting did not improve the accuracy of the decision tree model, and instead, decreased the accuracy. Two possibilities may explain this result: the model overfits the training data; adaptive boosting reduces error but not necessarily improves accuracy, which makes it more consistently effective in regression problems.

Dataset 2: Letter Recognition

In contrast, the letter recognition dataset was very responsive to boosting. Both 10 and 20 iterations increased the full training set accuracy from 85% to 95%, and there still may be some minor improvements at higher iterations (50+). This indicates that the decision tree algorithm generalizes the data very well on its own, and boosting effectively tunes the weights within the classifier.

K Nearest Neighbors

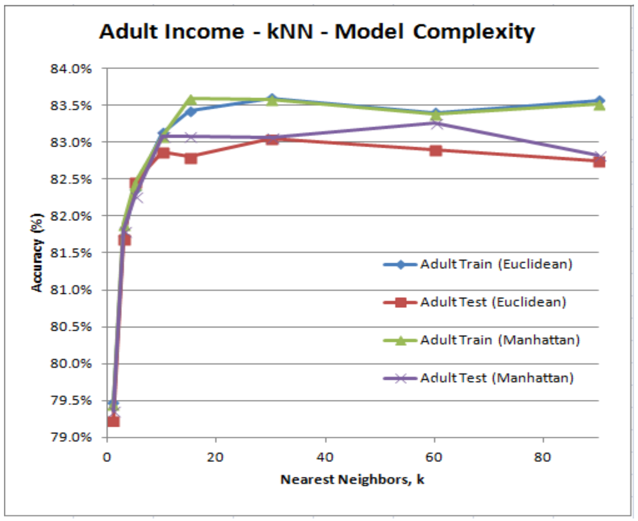
The WEKA classifier IBk (for Instance Based learning, with k nearest neighbors) was used to evaluate both datasets. The algorithm is a lazy learner because it works by identifying similar instances (neighbors) while it processes the dataset. The number of nearest neighbors, k, that are evaluated per instance were tested between 1 and 90. Improved performance at k=1 indicates a high repetition of instances with the same attributes and output class. Improved performance at high values of k indicates a more complex model where there may be many dominating attributes.

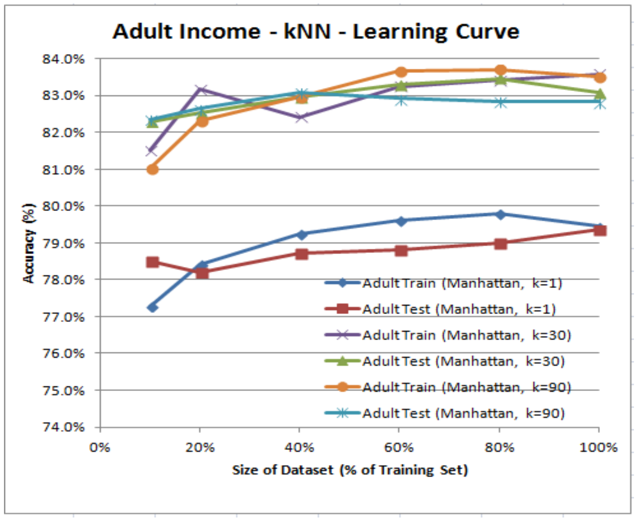
Two distance functions were explored for both datasets, as they were discussed in lecture: Euclidean (the norm) and Manhattan. The expectation is that the squared distance in the Euclidean distance function more aggressively weighs the closest neighbors than in the Manhattan function – this may not always perform better.

Dataset 1: Adult Income

The number of nearest neighbors (k) and the distance function were manipulated to evaluate kNN on Dataset 1. In the figure below, the model complexity chart indicates a 2-3% increase in accuracy from k=1 to k>15. Greater than k=15, the value of k has less of a significant impact. There is also a very small difference in results using the Euclidean and Manhattan distance functions, however best performing model employed k=60 and the Manhattan distance function. To explore further, the Manhattan distance function was selected and evaluated on more values of k.

The learning curve chart explored how the model learns across various sizes of datasets. It was suspected that models with lower values of k may reach peak performance at different dataset sizes. However, the experiments show that peak performance occurs at 80-100%.

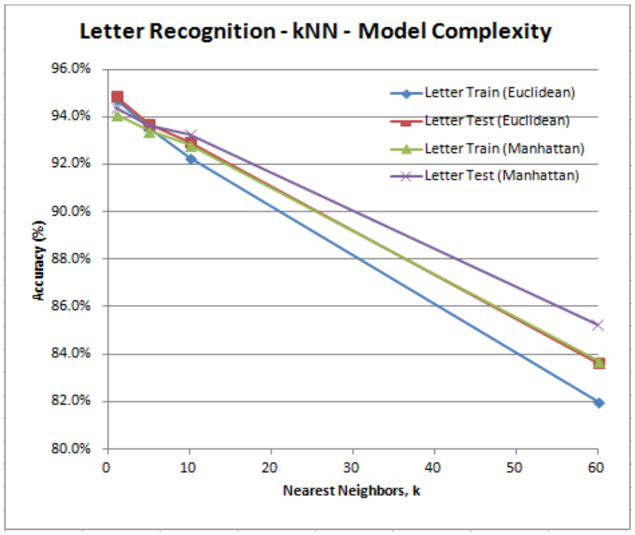


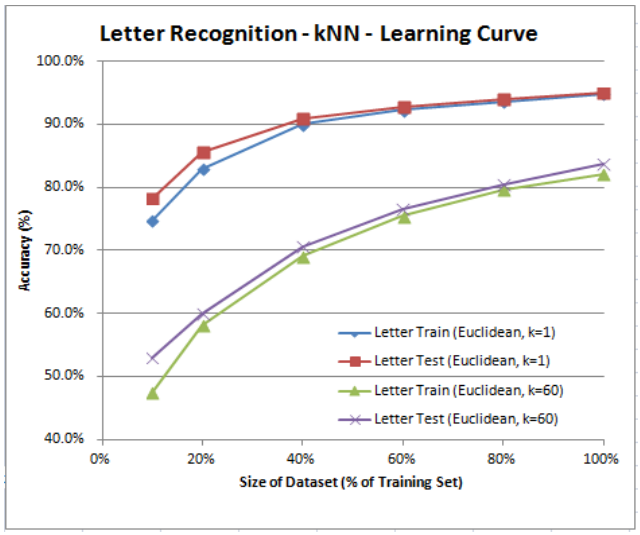


Dataset 2: Letter Recognition

In the letter recognition dataset, the observations vary significantly from the adult income dataset. The model complexity plot demonstrates a sharp decline in performance for increasing k, therefore k=1 was selected. Again, the performance between distance functions was quite comparable, however Euclidean performed best at k=1 and was selected for further study on the learning curve.

In the learning curve experiment, the primary focus was to evaluate the k=1 model. However, the k=60 model was included to observe whether the poor performance is a result of dataset size, such that, more data may help the k=60 model perform comparably or better than k=1. While the k=60 model improves faster than the k=1 model at 100% training size, also note that k=1 already performs very well at 95%, and further improvements are expected to be incremental.





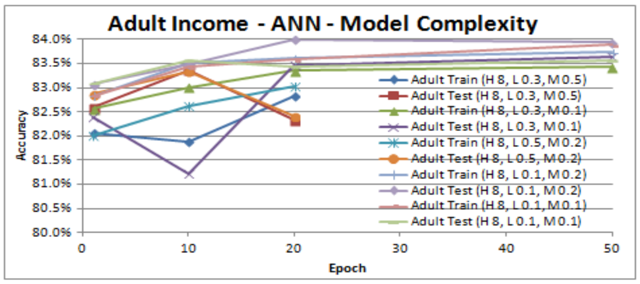
Artificial Neural Network

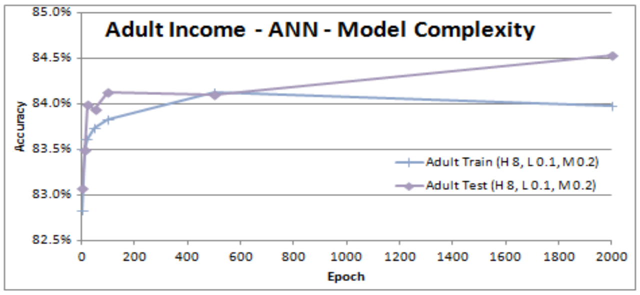
The WEKA classifier MultilayerPerception was used to evaluate the behavior of neural networks. In this model, 4 parameters were manipulated: hidden layers (both the number of layers, and number of units per layer), the learning rate, momentum and iterations (also known as epochs). There is a wide array of neural network configurations, and each can be studied deeply. For the model complexity experiments, a starting point for selecting hidden layers was to use the average of the number of attributes and outputs classes. This performed very well for both datasets ([14+2]/2 = 8 units, [26+16]/2 = 21 units). Variations to the standard configuration all showed poorer performance, such as removing the hidden layer, adding a hidden layer, or varying number of units per layer. Learning rate and momentum were adjusted to develop a model that trains effectively without overfitting to the training dataset.

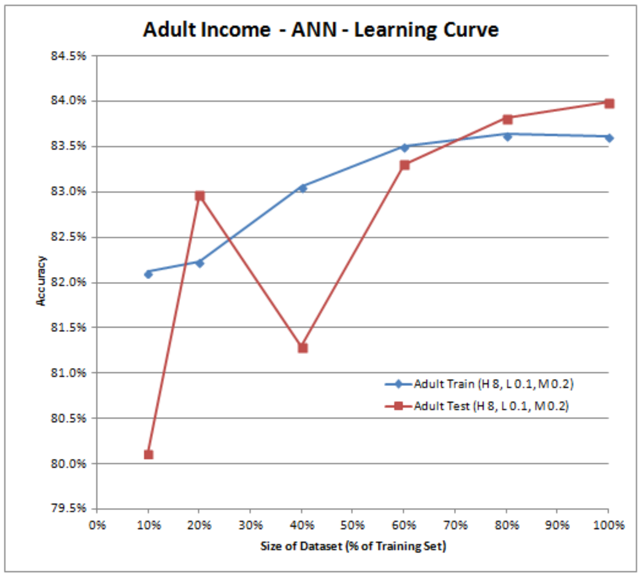
The artificial neural network algorithm is an eager learner because it uses back-propagation to determine appropriate weights between perceptrons. In the interest of computation time, the model complexity experiments were performed with a maximum of I=50 iterations. While this approach may appear to introduce a restrictive bias to models that perform better with more iterations, it was observed in both datasets that performance improvement between I=50 and I=2000 is less than 2%.

Dataset 1: Adult Income

For this dataset, single hidden layers of 0, 8 and 14 were explored, and the results were accuracies around 83%, all within a range of 1% especially for iterations I>50. The best performing model H = 8 was selected, and learning rates and momentum values were manipulated to determine that L=0.1 and M=0.2 produced the most accurate model. Surprisingly from I=500 to I=2000, the model showed improved on test accuracy but decreased on training accuracy. This is like due to a small variation between the training and test sets. Variation can also be seen in the learning curve results, as performance dips for the test set at 40%. This may have been reduced by randomizing the data before segmentation.



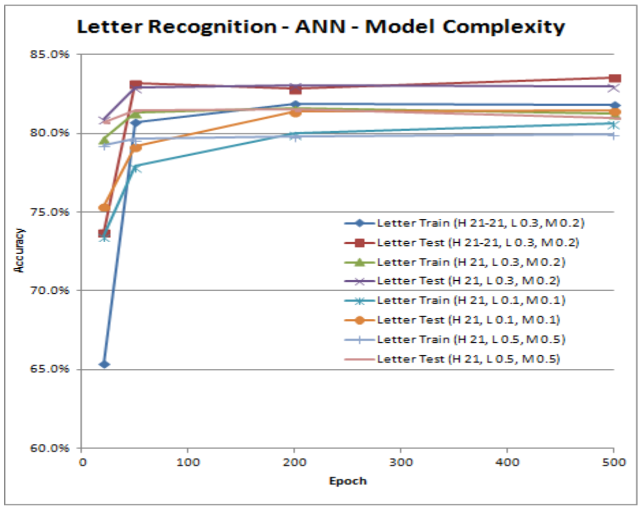


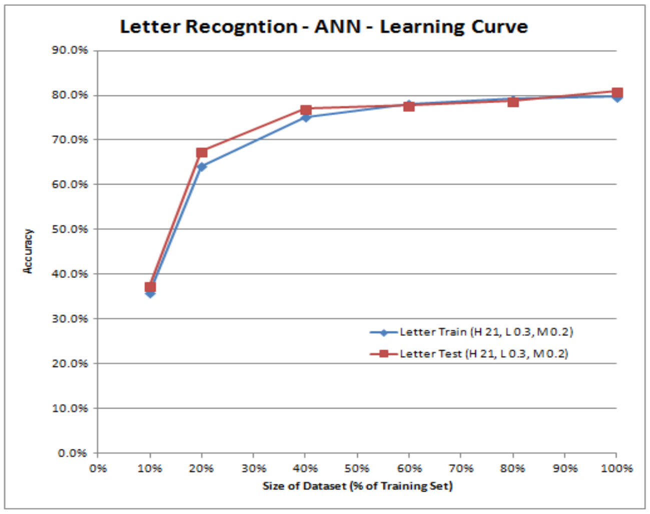


Dataset 2: Letter Recognition

For this dataset, several configurations were evaluated and the most relevant models had no hidden layer, and one hidden layer of 21. The value 21 was determined as an average of 26 output classes and 16 attributes. Two hidden layers of 21 were explored and performed closely with one hidden layer of one. As a consideration for further study, the performance of two hidden layers of 21 can be explored at iterations greater than I=50, which the model complexity experiments were performed with. A hidden layer of 21 was selected, and the evaluation of learning rate and momentum showed that L=0.1 and M=0.1 was not as optimal as L=0.3 and M=0.2. Additionally, L=0.5 and M=0.5 also showed poorer performance, indicating the parameters were too aggressive and had overfit the training data.

The learning curve shows that the performance of the algorithm improves significantly at 40% when there is sufficient data. The positive slope at 100% indicates that a larger training set may further improve the model up to 85% accuracy.



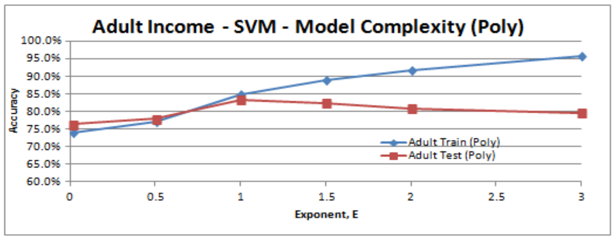


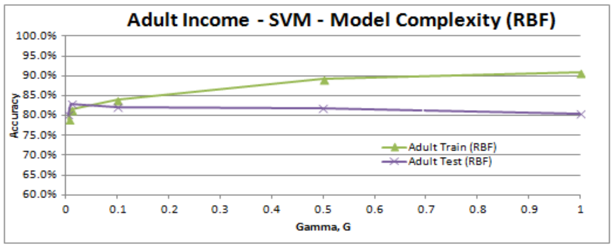
Support Vector Machines

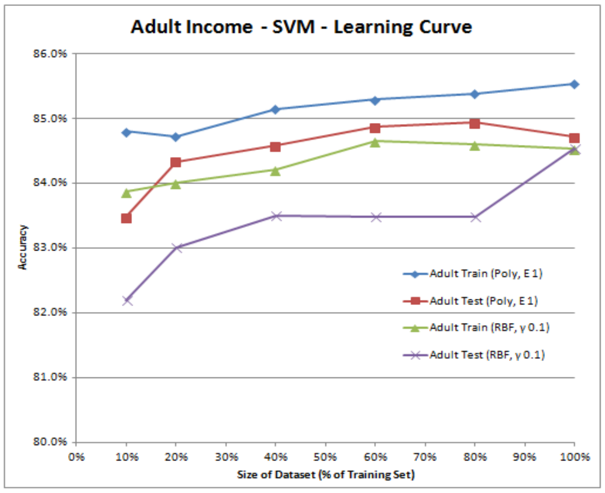
The WEKA classifier SMO (for sequential minimal optimization) was the final algorithm used to evaluate the two datasets. Support vector machines is an eager learning approach that works by creating a hyperplane to separate groups of classes, and optimizes the distance between points and this hyperplane. The shape of the hyperplane is controlled by a linear kernel, which may be replaced by other functions using the Kernel Trick. In this paper, two kernel functions were explored: Polynomial and RBF (radial basis function). The polynomial kernel function uses an exponential parameter E, while the RBF kernel function uses a gamma factor, G.

Dataset 1: Adult Income

In the model for the adult income dataset, the polynomial kernel performed best, peaking at E=1 after which overfit became apparent as training accuracy increased and test accuracy decreased. For the RBF kernel, peak performance occurred at low values of G, and the same overfit pattern from the polynomial kernel was seen for higher values of G.

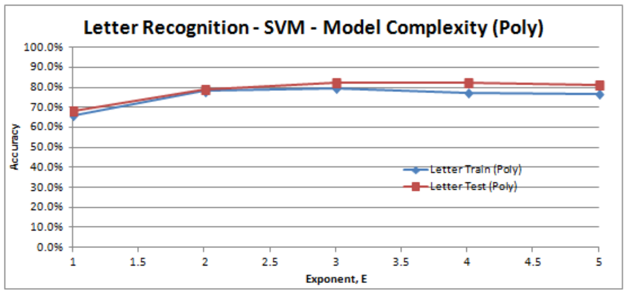


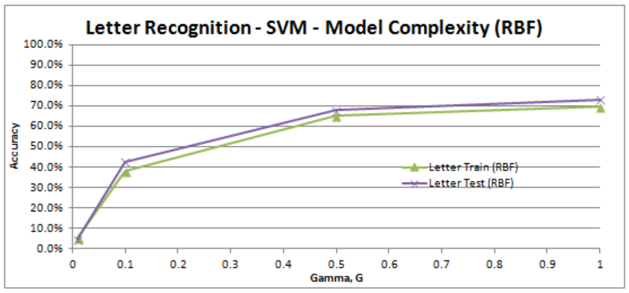


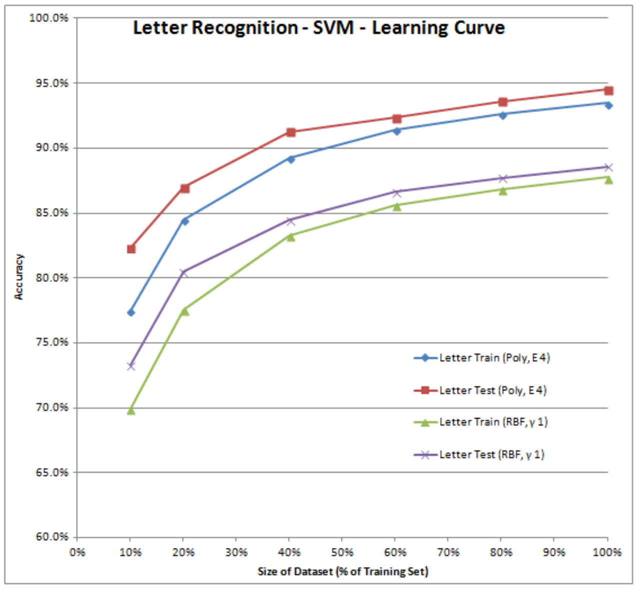


Dataset 2: Letter Recognition

The model complexity experiments were performed using 40% of the training set, for faster computation. As with Dataset 1, the polynomial kernel fit the data better for modeling this data, with E=4. Unlike Dataset 1, overfit is not observed for either kernel during model complexity experiments; however this may occur for larger training sets. Furthermore, the positive slope at 100% in the learning curve suggests more data will improve the performance of the model, despite already achieving 95% accuracy.







Conclusion

The model complexity and learning curve experiments were conducted on 80% of each of the overall datasets. The remaining 20% was set aside specifically for evaluating the models together. As observed, Dataset 1 is best represented by the decision tree algorithm, and Dataset 2 is best generalized by support vector machines. The proportion of false negatives to positives, the effect of more iterations and further tuning of the model parameters may be interesting for more analysis.

	Dataset 1		**Dataset 2**	
Classifier	Parameters	Accuracy	Parameters	Accuracy
Decision Tree	C=0.25, M=3	85.41%	C=0.25, M=2	86.05%
Adaboost	C=0.25, M=3, I=1	85.41%	C=0.25, M=2, I=500	94.08%
kNN	Manhattan, k=60	83.14%	Euclidean, k=1	94.13%
ANN	H=8, L=0.1, M=0.2, Epoch=500	83.38%	H=21/21, L=0.3, M=0.2, Epoch=500	82.75%
SVM	PolyKernel, E=1	84.55%	PolyKernel, E=4	94.48%

Bibliography

[1] Dataset 1: Lichman, M. (2013). UCI Machine Learning Repository [https://archive.ics.uci.edu/ml/datasets/Adult]. Irvine, CA: University of California, School of Information and Computer Science.

[2] Dataset 1: Lichman, M. (2013). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml/datasets/Letter+Recognition]. Irvine, CA: University of California, School of Information and Computer Science.

[3] Visual History of the World. Retrieved September 20, 2016. Retrieved from https://ourworldindata.org/slides/world-poverty/

[4] Human Development Index (HDI). Retrieved on September 20, 2016. Retrieved from https://ourworldindata.org/human-development-index/