Classification Problems

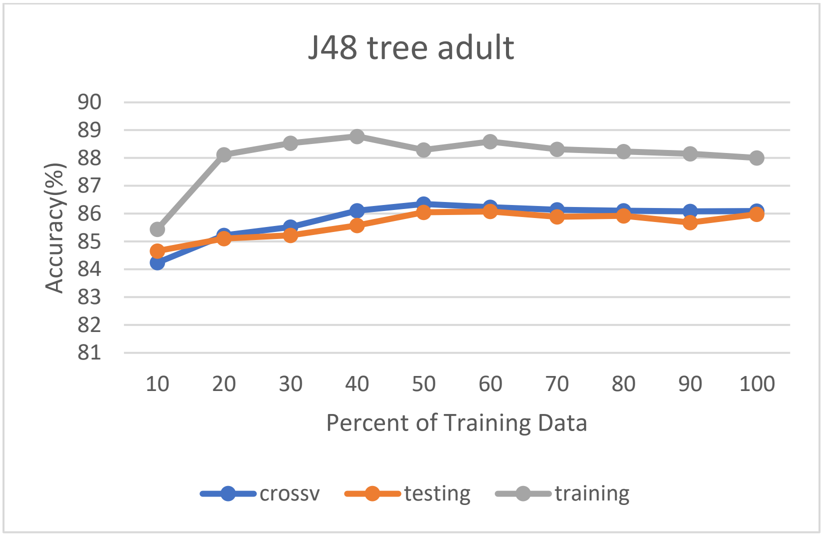
The first problem deals with a person's income in relation to several attributes. The dataset "adult" consists of attributes age, work class, final weight*, education, marital-status, occupation, relationship status, race, sex, capital-gain, capital-loss, hours-per-week, and native country. The prediction task is trying to determine based on these attributes if the person makes either >50k or <=50k. The dataset is large with 32,561 instances, and 7% had missing values which were replaced with the mode (nominal) or mean (numeric) of the missing attribute (using a filter). The dataset is also quite imbalanced with approximately 76% in the <=50k class and 24% in the >50K class.

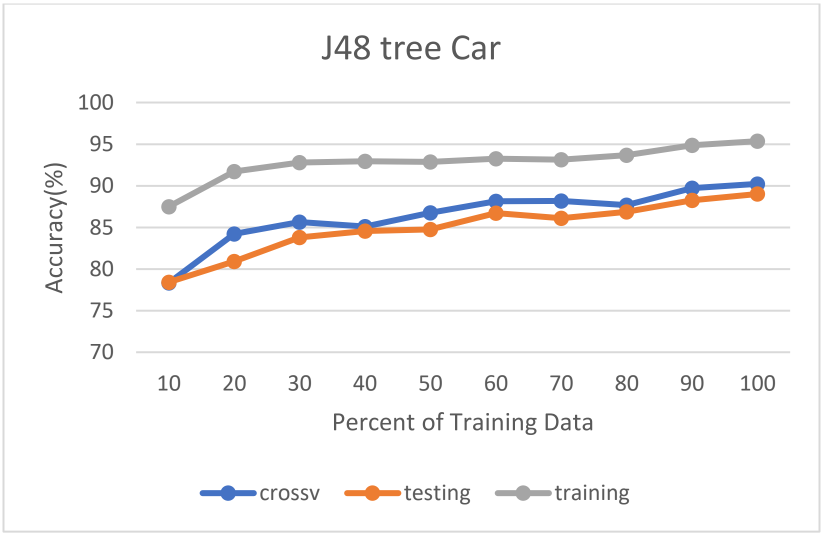
This problem is interesting in a practical sense as income is a measure of success in our society, and factors contributing to success should be study closely, as statistical advantages and disadvantages in certain groups can be evidence of societal issues. Analyzing data like this can provide insight into income inequality, and specifically the effect of race, nationality, and sex on income. From a machine learning perspective, I think this data is interesting as it is a large data set, consisting of 14 attributes with 8 nominal and 6 numeric.

The second problem deals with evaluating cars as either unacceptable, acceptable, good, or very good. The dataset "car" consists of discrete attributes: buying price, maintenance price, doors, persons, lug_boot (trunk), and safety. This data set consists of 1,728 instances, and had no missing values. For this assignment, I split the dataset 70%/30% into a training set and a testing set, respectively. The data set is highly imbalanced with a split of 70%, 22%, 4%, and 4% between classes.

This problem is interesting mainly from a machine learning prespective. This dataset is much smaller than adult, which makes algorithm such as SVM and Neural Networks much faster, allowing all of the data to be used. The imbalance is also much higher with only 4% of the training data in the good and very good classes.

Decision Trees – trees.J48



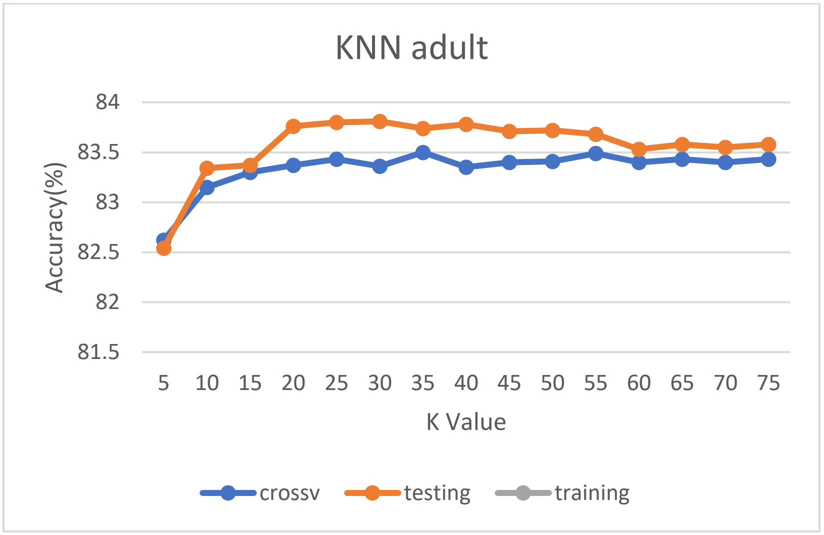


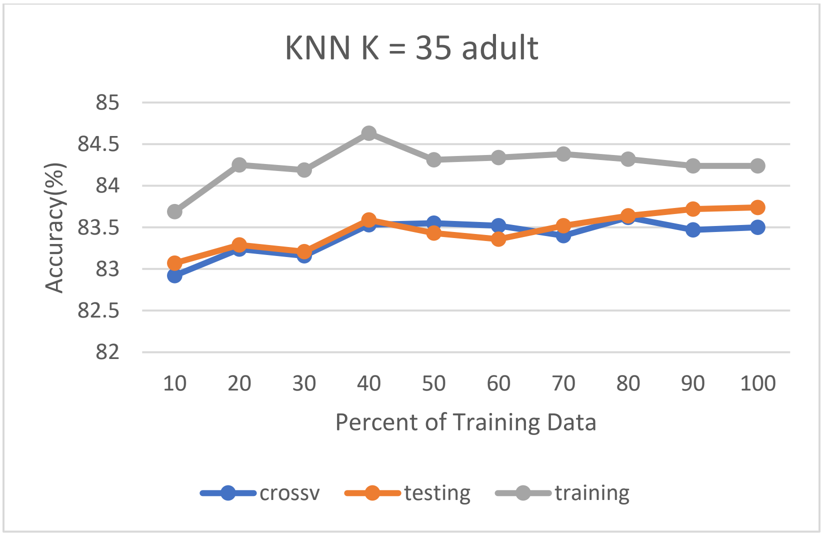
This algorithm was run with pruning and a confidence level of 0.25. The runtimes were quite short taking 0.93 seconds on training time using all of the adult dataset, and 0.06s to run over the testing set for adult.

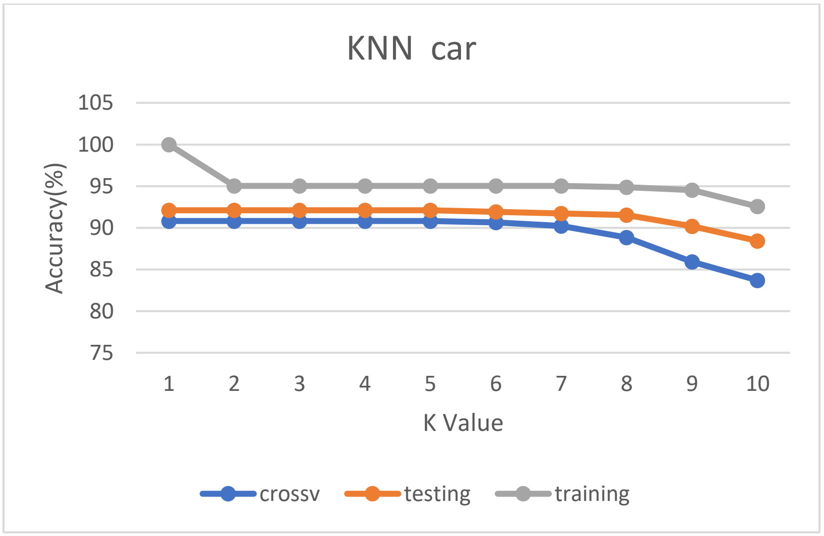
For the adult dataset, the training accuracy jumps up between 10% and 20% most likely due to noise and imbalance in the data. Then the training accuracy exhibits a slight downward trend between 40% and 100%, as the larger the size of the training set, the algorithm must test over more data. The cross-validation and testing accuracy show a slight upward trend, as the more data increases the information gain. No overfitting is occurring since the cross-validation does not decrease at the expense of training accuracy.

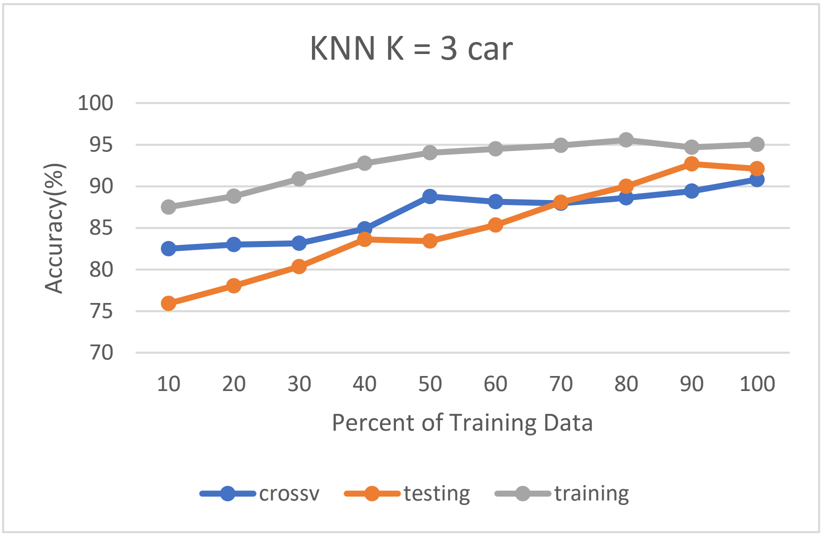
For the car dataset, all three accuracies are trending upwards. However, no overfitting is occurring as the cross-validation accuracy continues to increases throughout the interval.

K-Nearest Neighbors (unweighted) – lazy.IBk









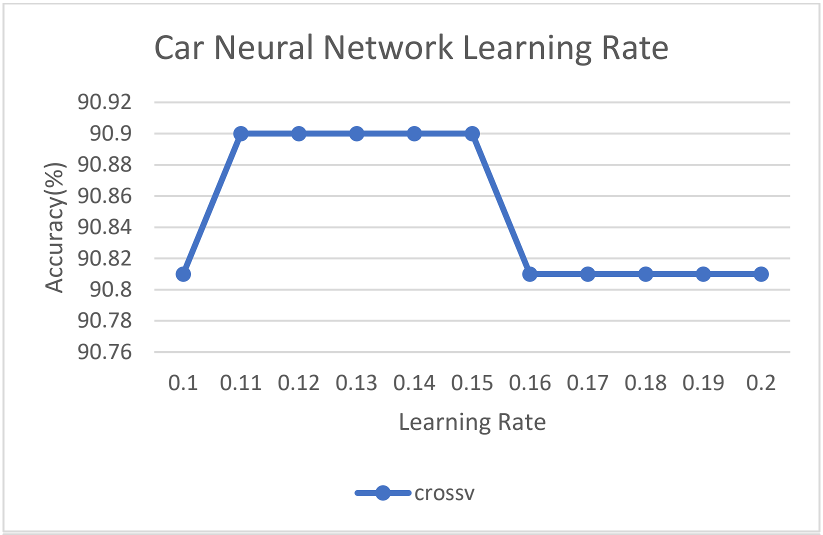
Running the algorithm with different K values and comparing the cross-validation accuracy, we were able to find optimal k values for each dataset. For the car dataset, for K = 1, …,5 the cross-validation remained unchanged; however, I chose K = 3, since K = 1 is likely to overfit, and K = 5 is more likely to underfit. We also see that as K increasing past 5, the accuracy decreases; this is because as K increases KNN is likely to underfit the training data, as more neighbors get to "vote" the accuracy variance decreases, but tends to ignore trends in the data. For the adult dataset, the highest cross-validation accuracy of 83.5% was attained at K = 35.

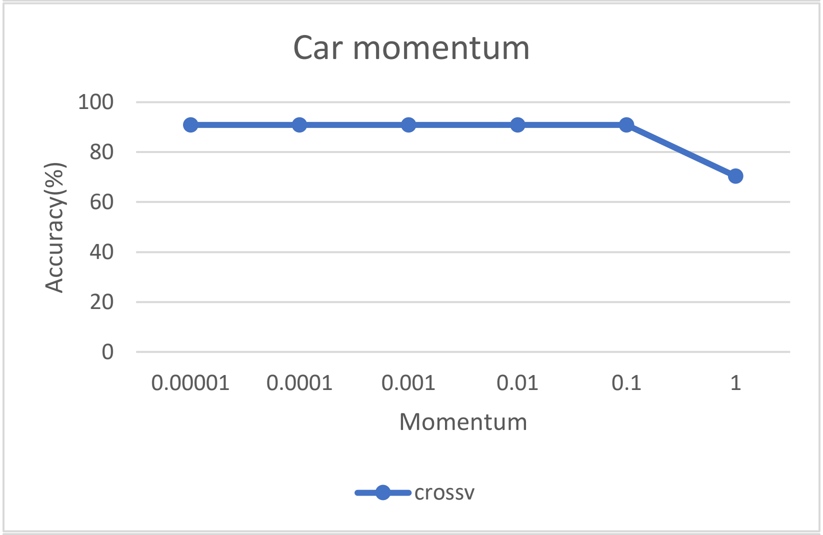
If we look at the learning curve for adult when K = 35, we see for low percentages of the training data there is a slight upward trend in the training accuracy. While this trend is not particularly significant (only changing by a percent), it is most likely caused by the imbalance of the data, and at such low percentages some attribute values are unlikely to be represented which can cause such discrepancies, after 40% we see the general downward trend in the training accuracy as the training data size increases. The cross-validation and testing accuracy also exhibit a general upwards trend indicating that the model is not overfitting based on training data. KNN is probably not the best algorithm for this dataset as it will treat all 14 attributes evenly which could possibly help contribute to the error, as intuitively it is unlikely that each attribute matters equally.

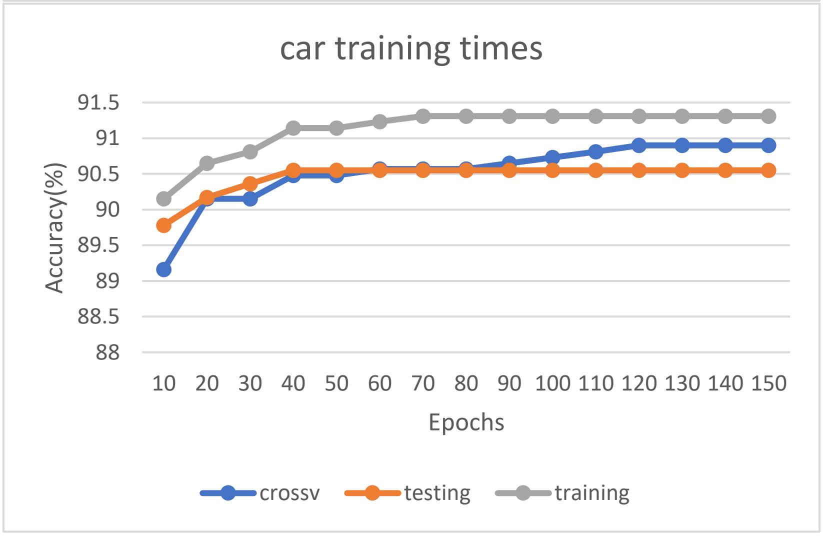
For the car data set, the learning curve for K = 3, the upward trend in the training accuracy is likely due to the imbalance of the data. We also see an upward trend in cross-validation accuracy and testing accuracy, showing the model is not overfitting and is generalizing better to the test set as we use more of the training data. I think KNN does a lot better here than on the adult dataset(92.1% testing accuracy vs 83.74%, 90.81% cross-validation accuracy vs 83.5%), as there are fewer attributes and it is more likely that they are closer in weight at least from a practical standpoint.

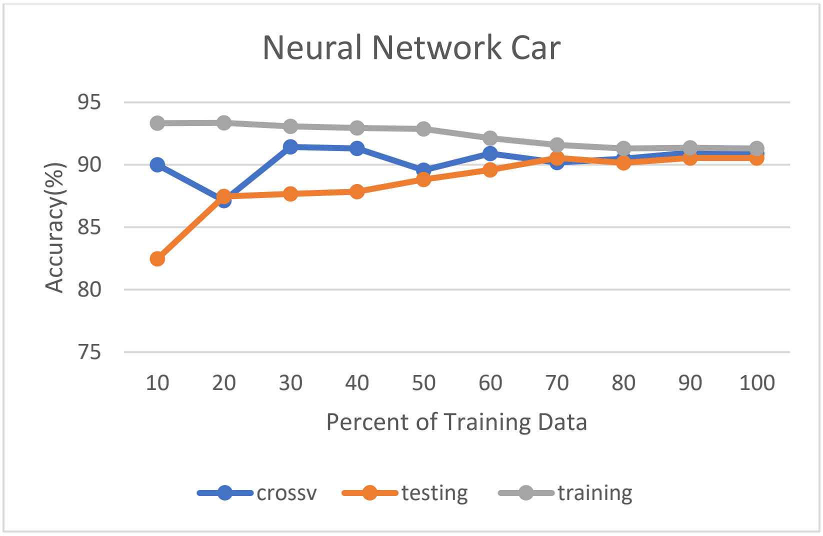
As KNN is a weak learner, training time is proportional to the training set, but essentially nonexistent (0s to 0.02s). However, the testing time takes significantly longer for higher values of k, especially on cross-validation.

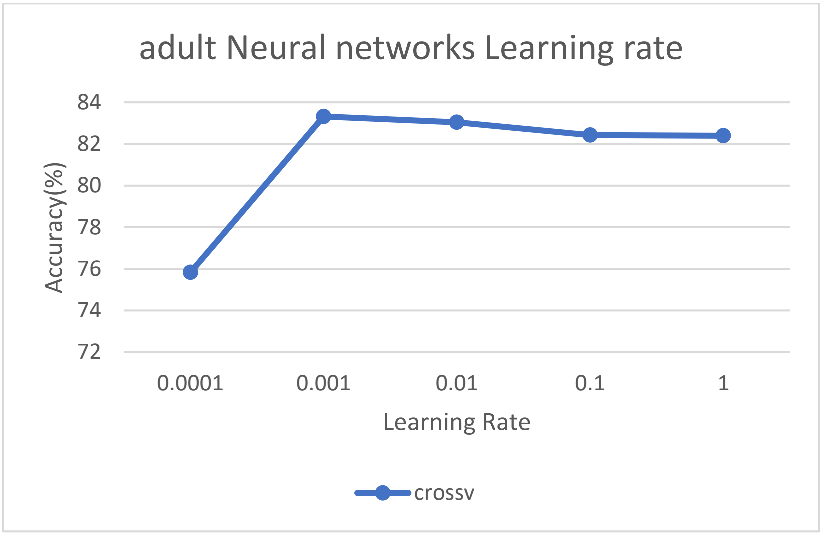
Neural Networks – functions.MultilayerPerceptron

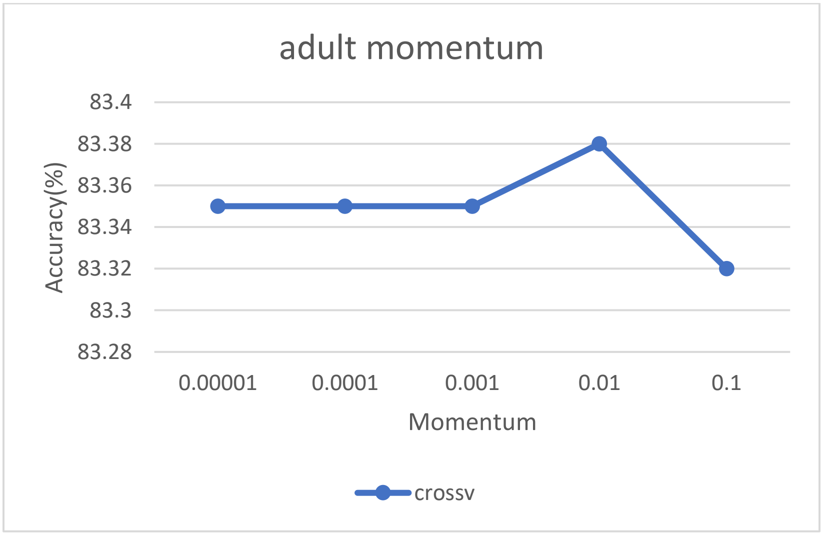


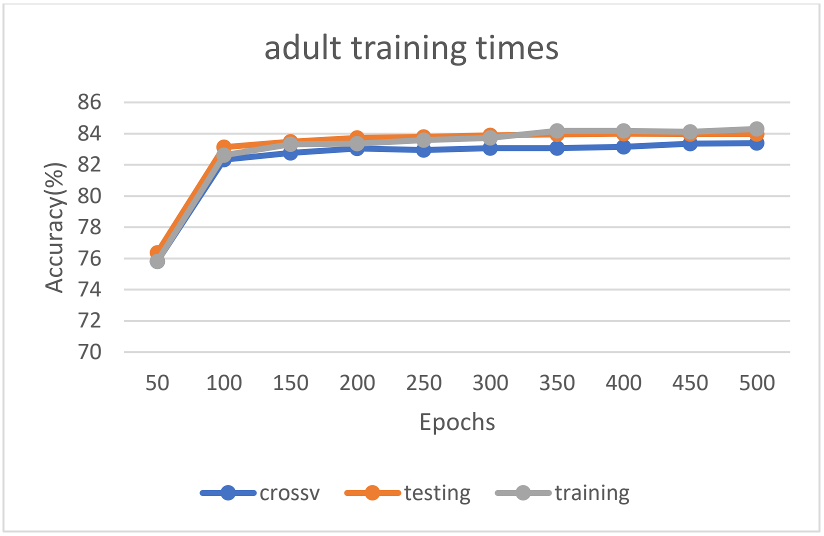


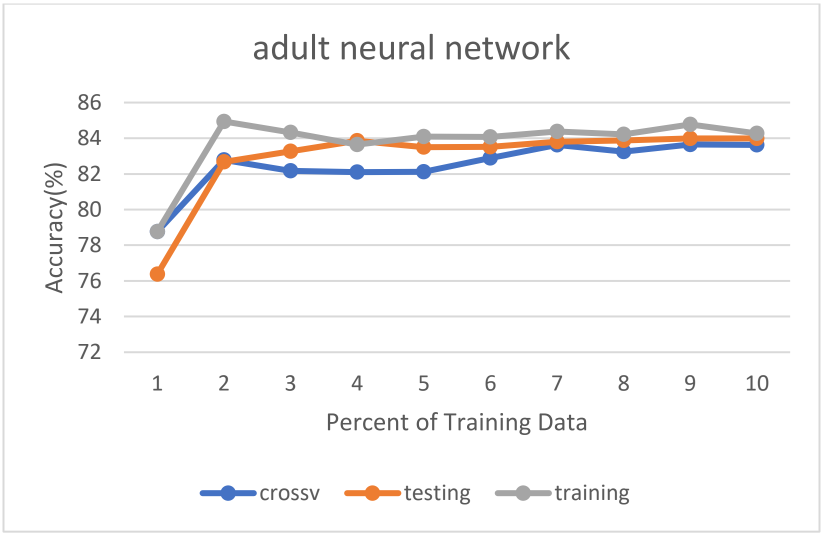



Neural Network Car
Accuracy(%)
95
90
85
80
75
10
20
30
40
50
60
70
80
90
100
Percent of Training Data
crossv
testing
training


adult Neural networks Learning rate
Accuracy(%)
84
82
80
78
76
74
72
0.0001
0.001
0.01
0.1
1
Learning Rate
crossv


adult momentum
Accuracy(%)
83.4
83.38
83.36
83.34
83.32
83.3
83.28
0.00001
0.0001
0.001
0.01
0.1
Momentum
crossv





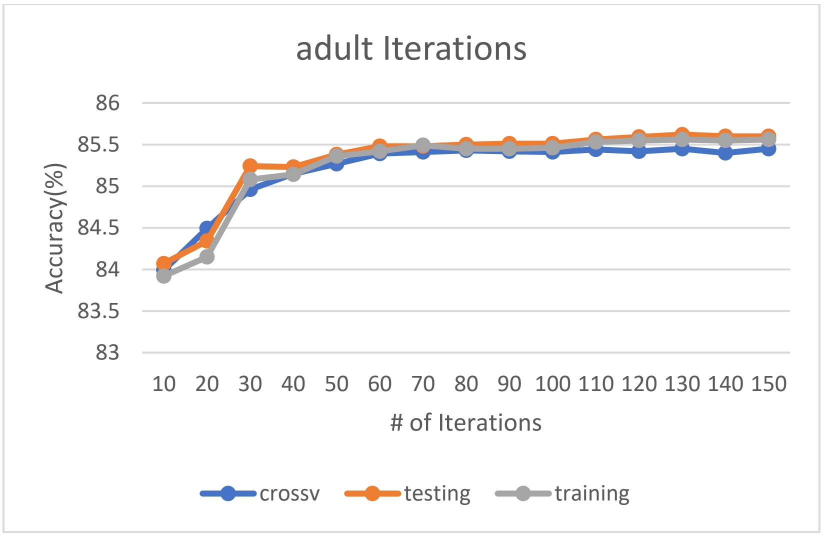
For this algorithm, I fixed the hidden layers of the Neural Network to 2, and the number of epochs was set to 500 (for the Percent training data vs accuracy graphs) to reduce the running time of the algorithm. First with momentum set to 0.1, I varied the Learning Rate magnitude (or value for car), using cross-validation to find an optimal value. Then given that learning rate, the same process yields a momentum value. Then I plotted the learning curves using the optimal Learning Rate and Momentum.

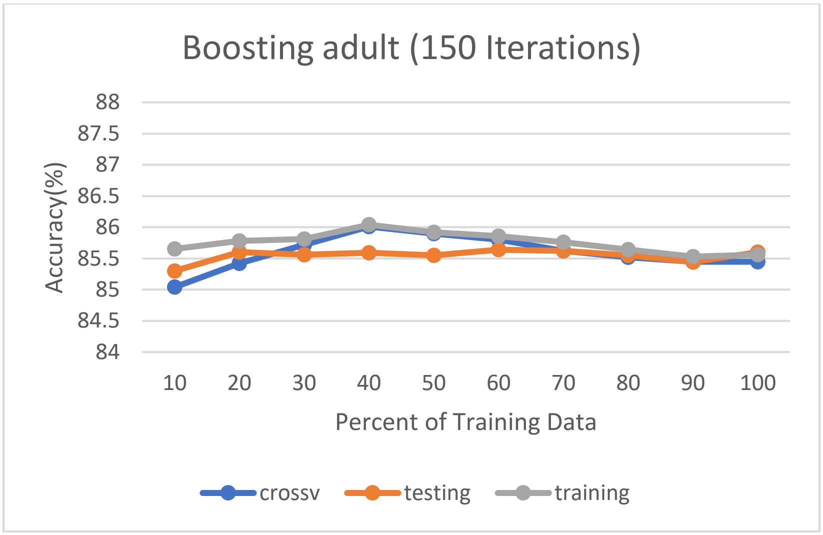
For the car dataset, I got a Learning Rate of 0.13. As this dataset is significantly smaller than adult, I was able to run the algorithm for actual Learning Rate values, instead of just the magnitudes. Using that learning rate, I got a value of 0.001 for the momentum. Using those values, the learning curve exhibits very typical behavior as the training accuracy decreases with the training size (more values to “test” over), and the cross-validation and testing accuracy show upward trends indicating information gain as more data is added. I also took 100% of the training data and the optimal learning rate and momentum, and varied the number of epochs the algorithm is allowed to train through. You can see that all three accuracies increase as the number of epochs increases, which is expected assuming the algorithm has reached a minimum.

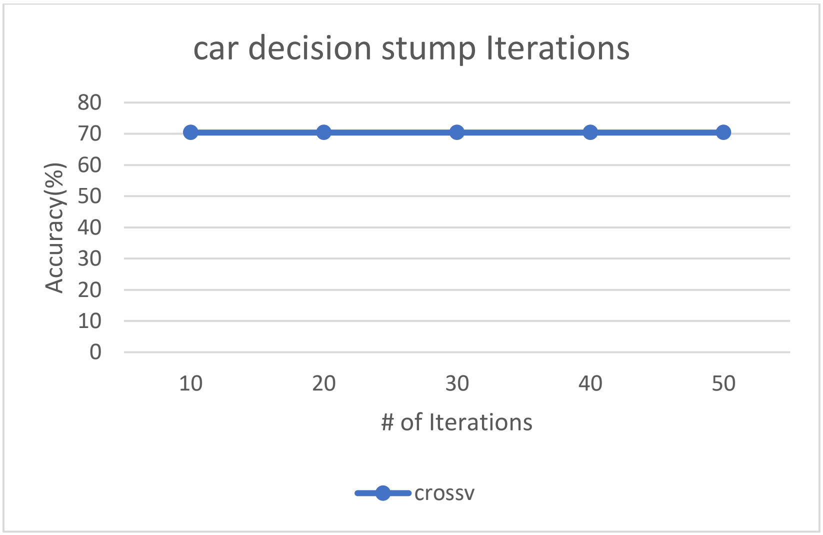
For the adult dataset, I ran the algorithm on 10% of the training data, as 100% would have taken too long (30x more instances than car). With a momentum of 0.1, I found an optimal Learning Rate of 0.001 using cross-validation, and similarly an optimal momentum of 0.01. I plotted the learning curve based on the 10% of the training data. I would attribute the initial jump in the graph to the inconsistency in balance between such a small percentage of data and the whole testing set, and the curve still displays the expected upward trend in cross-validation and testing accuracy. I also took 10% of the training data and the optimal learning rate and momentum and ran the algorithm varying the number of epochs. You can see at the beginning that the algorithm is cut short by the number of epochs, explaining the higher testing accuracy than training accuracy. As the number of epochs increase the algorithm gets closer to a minimum and thus all three accuracies increase.

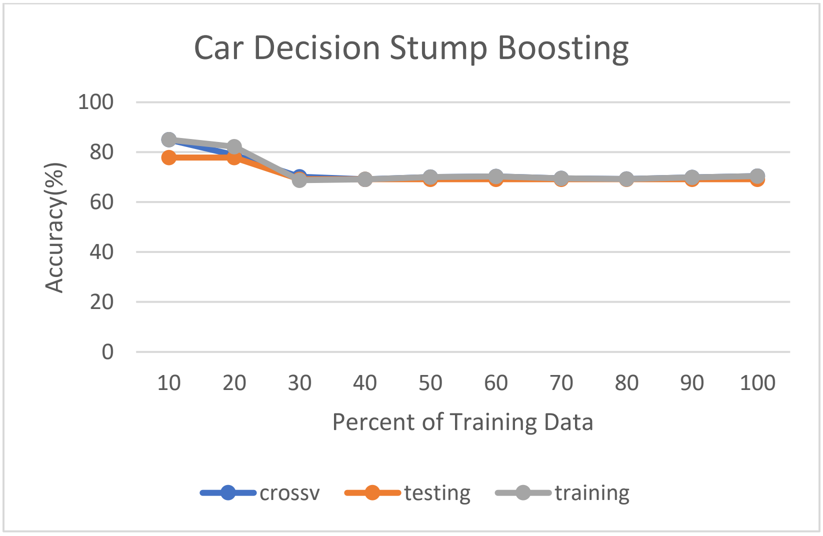
Neural Networks are eager learners and are therefore expected to have much longer training times. Also, from the graphs, we can see as the Learning rate is too small or too large the cross-validation accuracy decreases. Since the number of epochs is fixed (500), for too small a learning rate the algorithm will take too long to reach a minimum, thus decreasing the accuracy. For higher learning rates, you are more likely to overshoot the minimum. Similarly, for momentum, the goal here is to not get stuck in a local minimum; too small a value and you will get pushed back to the local minimum, and too large a value and you will overshoot the global minimum.

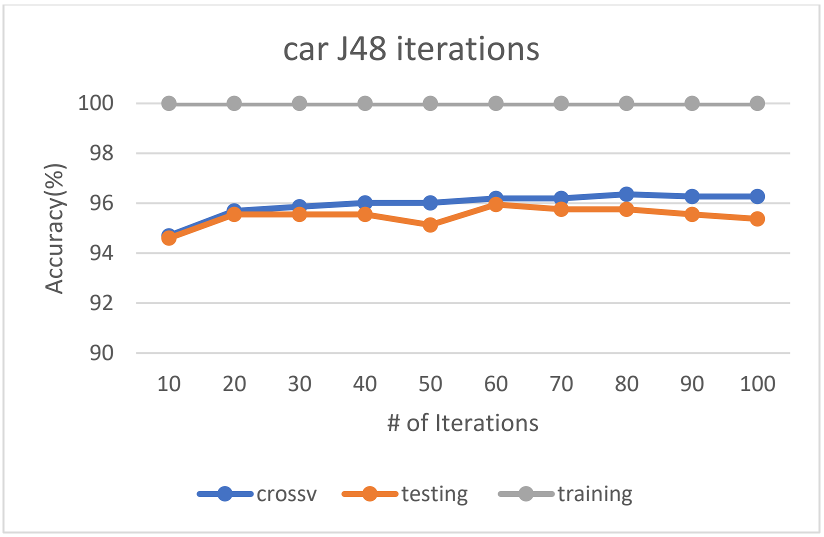
Boosting

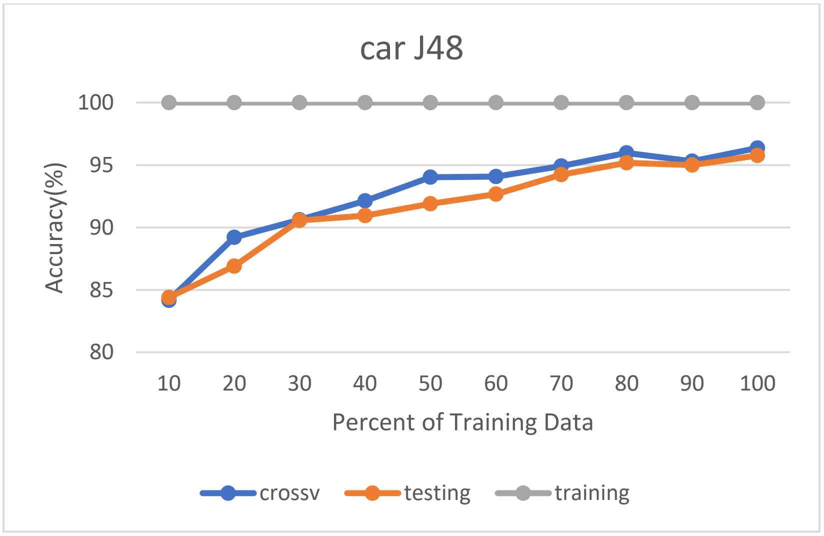



Boosting adult (150 Iterations)
Accuracy(%)
88
87.5
87
86.5
86
85.5
85
84.5
84
10
20
30
40
50
60
70
80
90
100
Percent of Training Data
crossv
testing
training


car decision stump Iterations
Accuracy(%)
80
70
60
50
40
30
20
10
0
10
20
30
40
50
# of Iterations
crossv


Car Decision Stump Boosting
Accuracy(%)
100
80
60
40
20
0
10
20
30
40
50
60
70
80
90
100
Percent of Training Data
crossv
testing
training

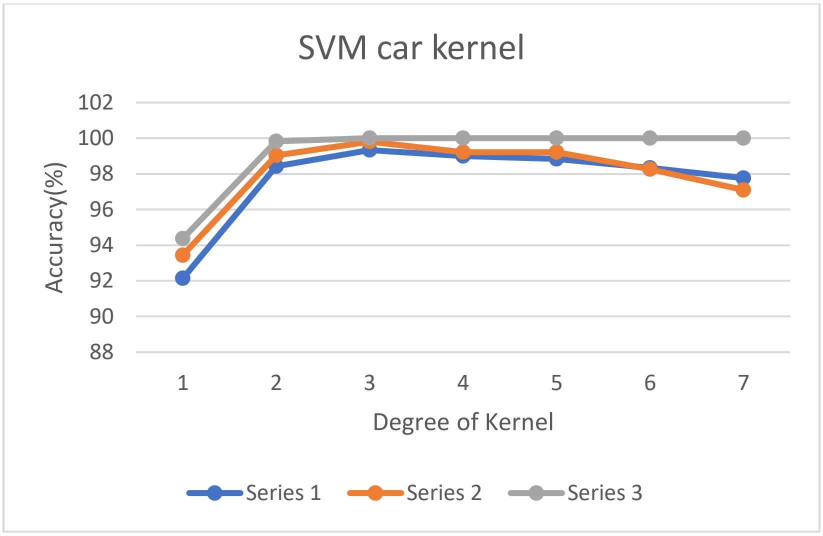


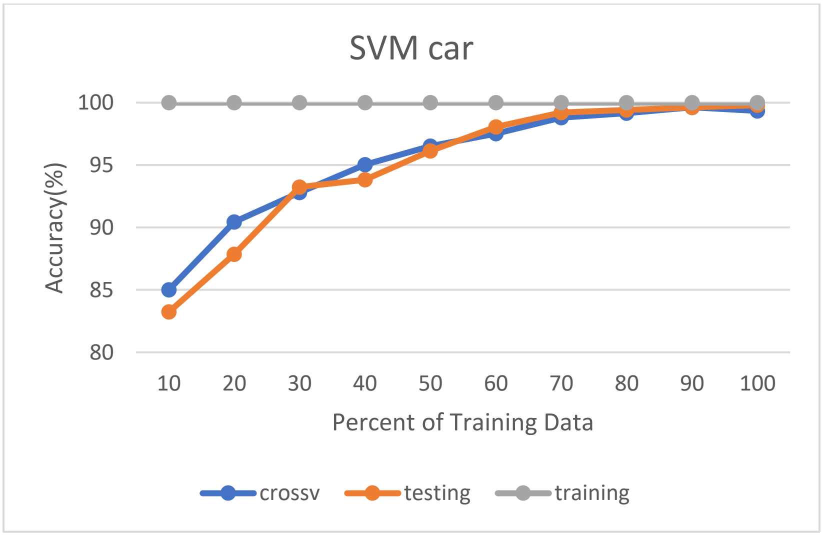


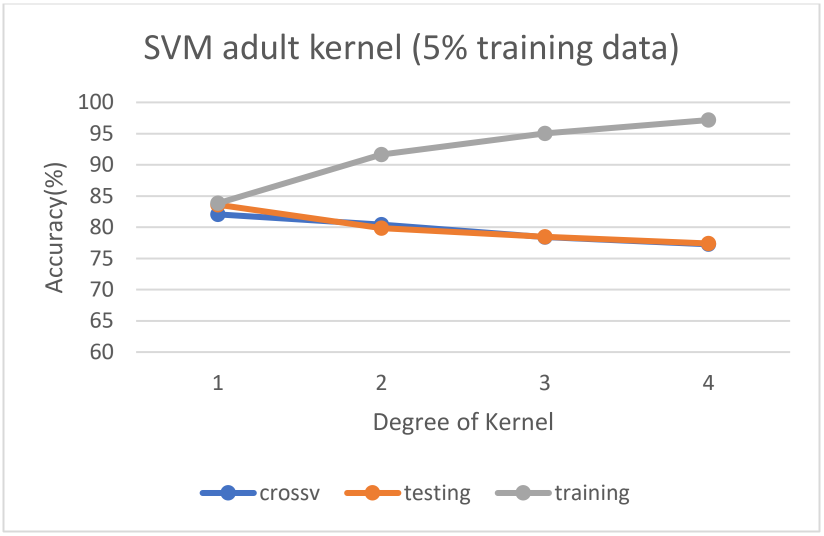
For the adult training set, first, I ran 100% of the training data using Decision Stump as the weak learner, while varying the number of iterations. The increasing training accuracy curve is again most likely due to noise or the imbalance of the data. The testing and cross-validation accuracies increase while the number of iterations increases. This doesn't show any of the signs of overfitting. Then I ran the algorithm for 150 iterations, varying the percent of training data used; the curve is mostly flat with a slight upward trend of cross validation and testing accuracy.

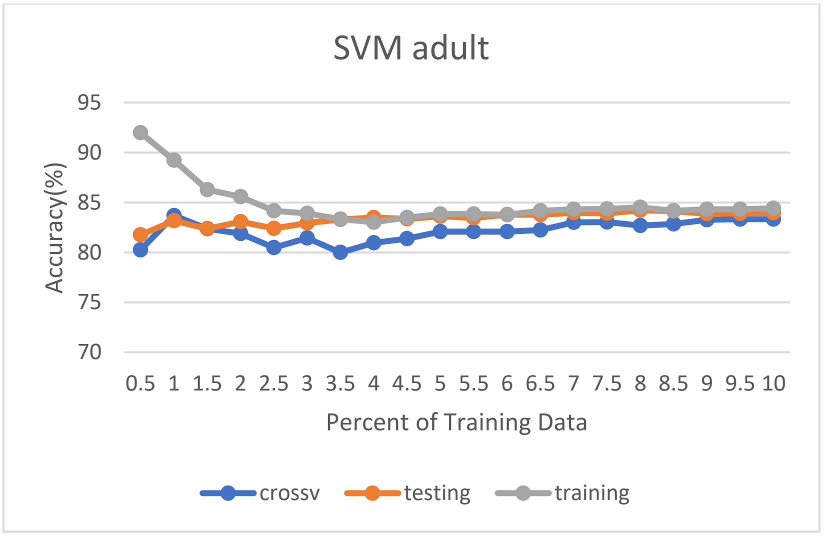
For the car dataset, using Decision Stump as the weak learner, the cross validation did not depend on the number of iterations, and overall the algorithm did rather poorly, attaining a cross validation accuracy of 70.38% and a testing accuracy of 69.13% using the full training set. So, I decided to use J48 instead of Decision Stump to improve upon the algorithm. With J48 (confidence level 0.25, pruning) on 100% of the training data, we see a typical learning curve, as the cross-validation and testing accuracy both increase with the number of iterations, while the training accuracy is constant at 100%. The cross-validation attains a maximum at 80 iterations of 96.13%. Using 80 iterations, the learning curve with respect to training size is shown; as well, we get the testing and cross-validation accuracies increase with the size of the training data while the training accuracy is constant at 100%. As boosting is supposed to, this improves upon the J48 data above, taking 90.23% cross validation accuracy to 96.36% and 89.01% testing accuracy to 95.76%.

Support Vector Machines – functions.SMO (complexity = 1, epsilon = 1 e-12)







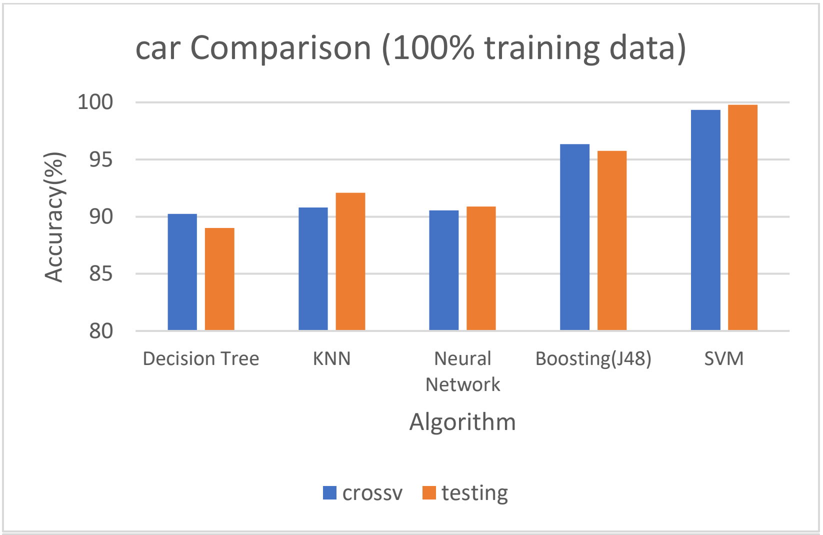


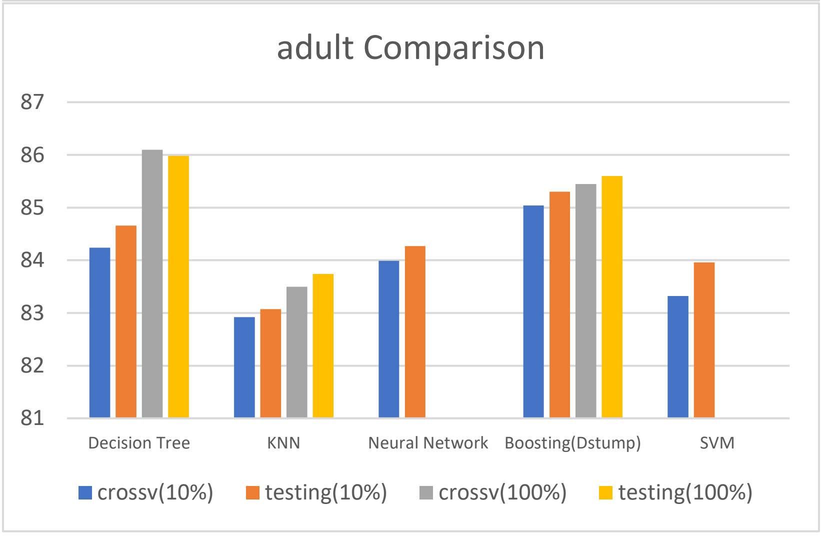
Using 100% of the training data, I first varied the degree of the polynomial Kernel. For the car dataset, we see there is a maximum at degree 3 with 99.33% cross-validation accuracy and 99.8% testing accuracy, after that the model begins to overfit as the training accuracy is a 100% and the cross-validation accuracy begins to decrease. For the adult dataset, I used 5% of the data, as the runtime for a greater percentage was too long for any degree greater than 1. We see that the cross-validation accuracy decreases and the training accuracy increases with respect to the degree of the kernel. This suggests overfitting for degree > 1.

For the car dataset, the learning curve shows the training accuracy constant at 100%, and the cross-validation and testing accuracy trending upwards as more training data is being used. The model is not overfitting based on training data size, as the cross validation has an upward trend. The testing accuracy and cross-validation attain very high maximums at 99.63% and 99.8%, respectively.

For the adult dataset, the learning curve is plotted for 5% of the training data. The training accuracy is shown to decrease as the training data size increase, which is expected since there is more data to test over. The testing accuracy and cross-validation are shown to only have a very slight upward trend. The model doesn't show any signs of overfitting.

Overall





The car dataset was best suited by the support vector machine algorithm, achieving a cross-validation accuracy of 99.33% and 99.8% with a polynomial kernel of degree 3. For a small dataset like car, SVM is great; however, we saw that for adult the runtime was impractical. We also saw with Neural Networks the runtime can be a huge problem when your training set is large. A close second would be the boosting using the J48 tree which takes way less time, as its runtime is proportional to the J48 runtime whereas with SVM depending on the degree of the kernel it can take a lot longer.

For the adult dataset, it's a little harder to compare. For 100% of the dataset the J48 decision tree had the highest cross validation accuracy of 85.98% and the highest testing accuracy of 086.1%. However, I was unable to perform the Neural Network algorithm or the SVM algorithm on the full training set. Given more time, I would definitely run these algorithms again on the full training set; I think this is by far the more interesting and difficult(accuracies were significantly lower) classification problem.