Robert Bradshaw

2/2/2017

CS 4641

Assignment 1: Supervised Learning

Let's talk about the datasets I chose to work with. I immediately ran into a rut upon starting this assignment; I spent the first three days working on this assignment by attempting to train a 10000+ attribute dataset with thousands of examples on my school laptop. Needless to say, due to the curse of dimensionality, the multilayer perceptron in WEKA was unable to finish in a reasonable amount of time, so I decided to move on to other datasets. I went hunting for .ARFF datasets in particular using University of California Irvine's repository. The majority of the WEKA classification runs' result buffers can be found in each datasets' respective folders (if you don't feel like retesting slower algorithms).

DATASETS

- Vowel.arff – 990 instances with 14 attributes
- Waveform-5000.arff – 5000 instances with 41 attributes

GENERAL

The first dataset I came across that I found to be interesting was the vowel.arff dataset. This dataset is personally very interesting to me because it attempts to perform vowel recognition by taking in account factors (i.e. attributes) regarding the gender of the person

speaking, the identity of the person speaking, and other attributes that are hidden from people using the dataset; I these assume are related to frequency, noise, the environment in which the sound files were recorded, and other audio-related attributes. From a CS 4641-perspective, I believe this dataset was particularly interesting because it was one of the first datasets I ran where the WEKA SMO algorithm was the worst performer (with default settings) and where KNN was the best performer. The second dataset I used was the Waveform-5000 dataset (which deals with 3 classes of waves and 21 other attributes with continuous values between 0 and 6). I believe this dataset is interesting because there is high variability between the performances of each algorithm relative to one another. I believe these two datasets complement each other well since the machine learning algorithms that perform poorly on one dataset appear to perform way better on the other set; noticing this, I deduced that it would be easiest to elucidate the nuances of machine learning by striving to understand exactly what characteristics of each of these datasets causes them to perform in opposite ways relative to one another on these algorithms.

I'll talk about the general performance of the supervised machine learning algorithms I used next. **10-fold cross validation** was used for every algorithm in these datasets. **70-30 percentage split** was used for the testing data. Learning curves were calculated using the WEKA experimenter's CrossValidationResultProducer and Percent Removed filter applied in the generator (running the experiment with 10, 20, 30, 40, 50, 60, 70, 80, and 90% of the data used for the training sets). The base WEKA settings generally provided decent results for each of the algorithms they provide.

I didn't realize run times would be very important to directly record for this assignment until after I finished collecting data. In the "sup" folder, you can find a multitude of WEKA result buffers saved that include run times of the algorithms.

RAW DATA (Root mean squared error)

KNN

K	Vowel Training	Vowel Testing	Waveform Training	Waveform Testing
1	0.0358	0.0696	0.4192	0.414
2	0.0507	0.0847	0.3527	0.3505
4	0.0862	0.1381	0.3168	0.3163
8	0.19	0.2257	0.3002	0.2988
16	0.255	0.2691	0.2903	0.292
32	0.2808	0.2862	0.2885	0.2922
64	0.2787	0.2743	0.2909	0.2947
128	0.2783	0.2824	0.2968	0.3027
256	0.2796	0.2802	0.3059	0.3122

SMO

C	Vowel Training	Vowel Testing	Waveform Training	Waveform Testing
0.2	0.2717	0.2757	0.3235	0.3266
0.4	0.2692	0.2708	0.3228	0.3264
0.5	0.2685	0.2699	0.3222	0.3266
1	0.2665	0.2677	**0.322**	0.328
1.2	0.2661	0.2672	0.3224	0.3264
1.5	0.2657	0.267	0.322	0.3264
4	0.2646	0.2655	0.3231	0.3261
20	0.2639	0.2643	0.3231	0.3264
100	**0.2638**	0.2641	0.3231	0.3264
150	0.2638	0.2641	0.3231	0.3264
200	0.2638	0.2641	0.3231	0.3264

Neural Net

Hidden Layers	Vowel Training	Vowel Testing	Waveform Training	Waveform Testing
7	0.1709	0.1985	0.3048	0.3286
10	0.1372	0.1585	0.3135	0.3233
14	0.1072	**0.1304**	0.3191	0.321
20	0.1036	0.1348	0.3297	0.3297
28	**0.0799**	0.1171	0.328	**0.3125**
41	0.0816	0.1069	**0.315**	0.3198
56	0.0857	0.1151	0.3257	0.3219
82	0.0753	0.1063	0.3269	

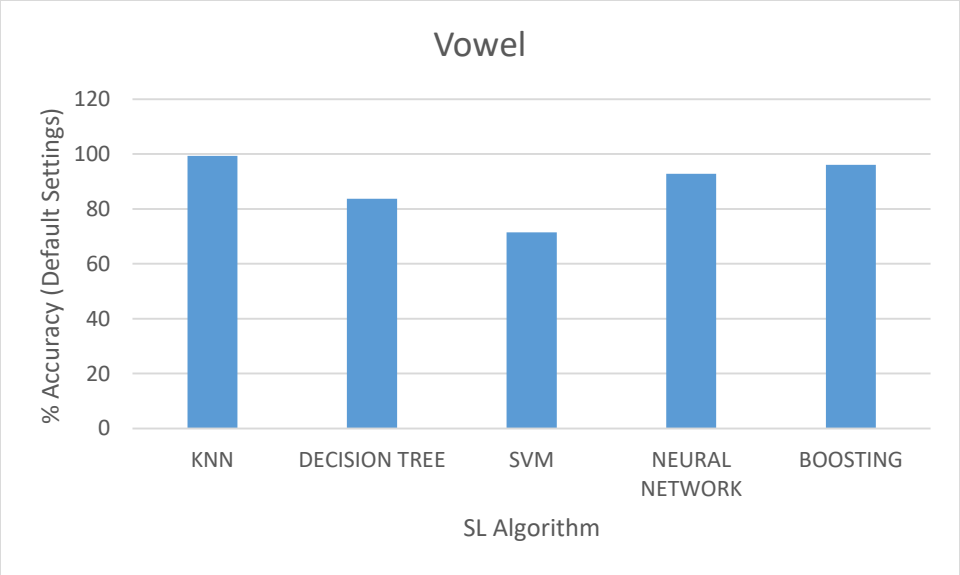
Decision Tree

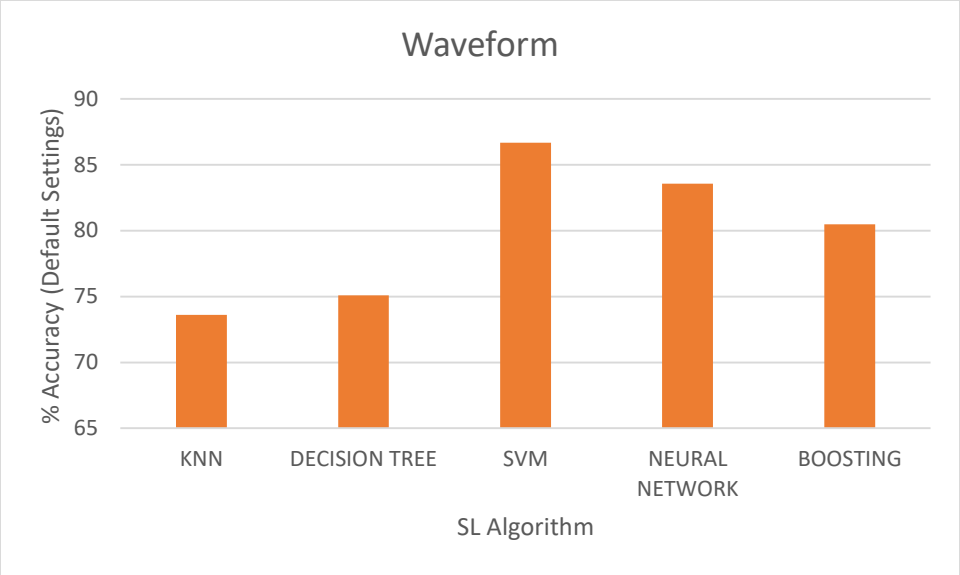
Confidence Factor	Vowel Training	Vowel Testing	Waveform Training	Waveform Testing
0.015625	0.1751	0.1926	**0.3759**	**0.3837**
0.03125	0.1741	0.1933	0.3806	0.3886

0.0625	**0.1727**	0.1933	0.3863	0.4042
0.125	0.1729	0.1918	0.391	0.4052
0.25	0.1712	0.1918	0.3951	0.4114
0.5	0.1686	**0.1871**	0.3955	0.414
0.75	0.1619	0.1872	0.396	0.414
0.9	0.1619	0.1872	0.396	0.414
1	0.1619	0.1872	0.396	0.414

Boosting

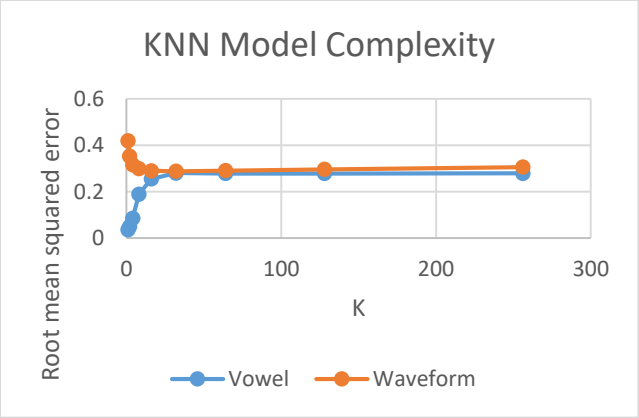
Confidence Factor	Vowel Training	Vowel Testing	Waveform Training	Waveform Testing
0.015625	0.1306	0.1508	0.3551	0.3467
0.03125	0.1209	0.1532	**0.3517**	**0.3549**
0.0625	**0.1159**	**0.15**	0.3582	0.3655
0.125	0.1202	0.1371	0.3594	0.3552
0.25	0.1228	0.1447	0.3618	0.3662
0.5	0.123	0.1402	0.3496	0.365

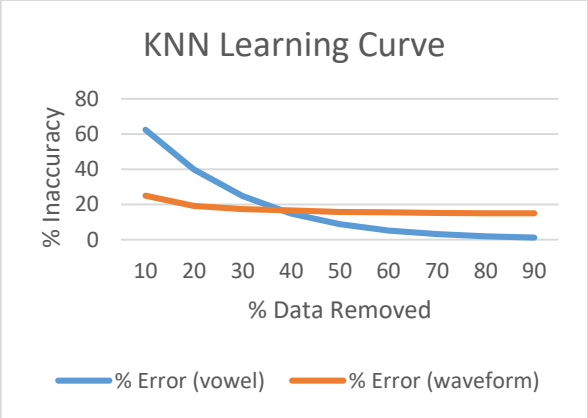




KNN

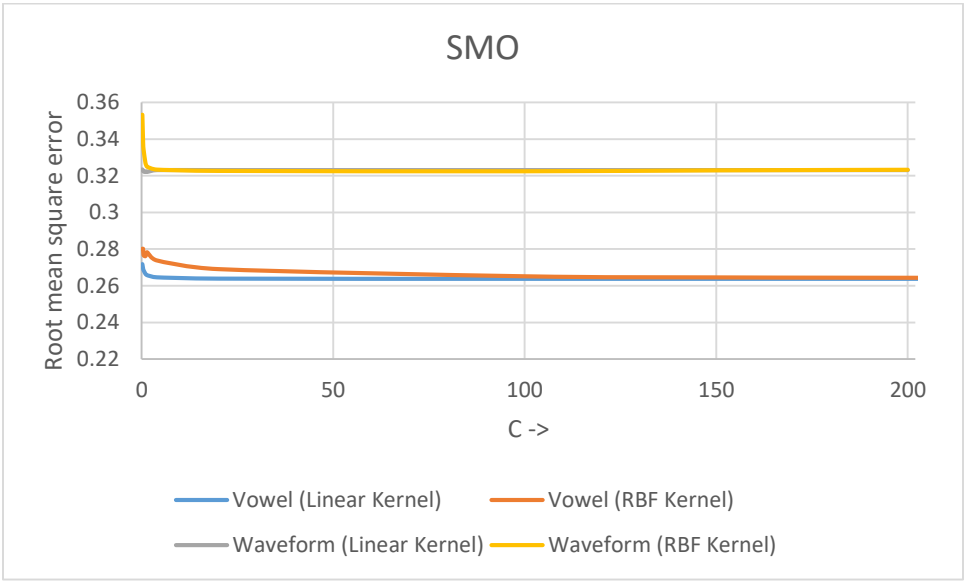
KNN runs extremely fast (a maximum of 3 seconds for even extremely high values of K on large datasets like Waveform-5000). Something that totally surprised me was the KNN (with K = 1) was the most accurate out of all the other algorithms for the Vowel dataset! This is due to this dataset having extremely low bias and variance. An increase in K results in logarithmic growth of error since class boundaries become less pronounced. The waveform dataset performs in the exact opposite manner; as a result of high variance in this large dataset, higher K values logarithmically decrease the classification error. This is due to the high variance in this relatively larger dataset. Taking in more neighbors into consideration helps reduce the adverse effects of this variance due to noisy data.





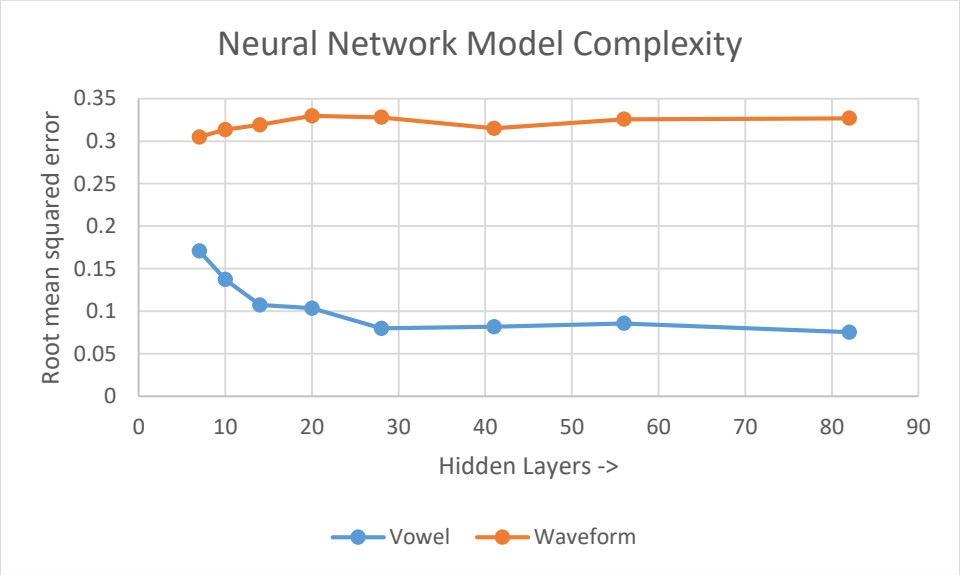
SMO (SVM)

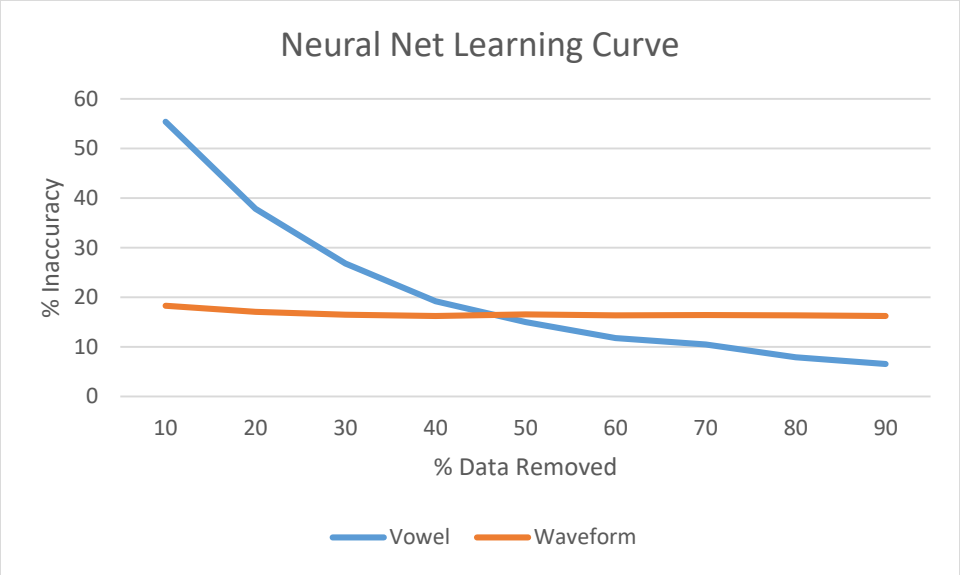
The C variable for SMO (WEKA Support Vector Machine) is basically the degree to which you want to avoid classifying examples incorrectly. This is done by moving the adjusting the margin of a hyperplane that separates the positive/negative data points as examples such that you classify examples for a specific dataset more accurately. I initially used two kernel algorithms for these datasets (linear kernel and a normalized polynomial kernel). However, the normalized poly kernel was negligibly different from the regular linear kernel, so I decided to apply to the RBF Kernel provided by WEKA: **K(x,y) = e^-(0.01* <x-y,x-y>^2)**



The RBF Kernel causes the Vowel datasets performance to diverge at a lower C value, and the Waveform dataset's performance to diverge with a much higher C value. The RBF Kernel also tends to result in lower classification accuracy relative to the default linear kernel on both of the datasets. This is due to the datasets' easier linear separability. Overall, the linear kernel produces the lower % error for the SMOs for both of the critical values of model complexity curves.

Neural Networks



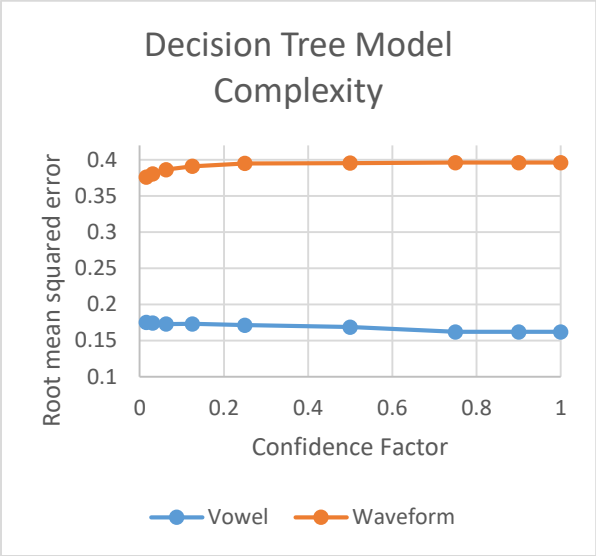


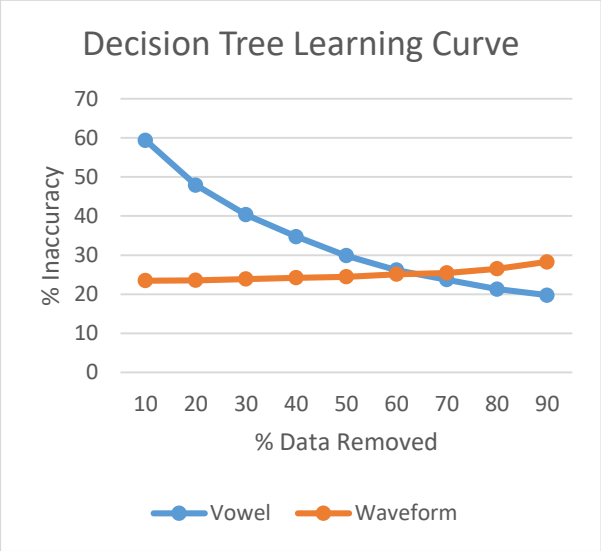
For the neural network, I used the multilayer perceptron algorithm provided by WEKA. To analyze the model complexity, I varied the number of hidden layers on each run. Knowing how neural networks work somewhat from my experience with them in CS 3600, I paid especially close attention to the runs where the number of hidden layers used were multiples of the number of attributes that a dataset contained. Vowel had 14 attributes -- so I paid close attention to runs with 7, 14, 28, and 56 hidden layers. Waveform-5000 had 41 attributes so I paid attention to 10, 20, 41, and 82 hidden layers. As suspected, the critical values of the model complexity curves for these algorithms occurred at a multiple of the number of hidden layers. Vowel's critical number of hidden layers was 28 (2 * # of attributes), and Waveform-5000's critical number of hidden layers was at 41 exactly (# of attributes – the default number of hidden layers). The learning curve for Vowel's neural net performance indicates that the bias in the data set is being overcome by the increase in training data size – which is expected behavior for a neural network. Waveform-5000 is a pretty hard dataset to train it appears, given its learning curve. Extremely biased data can result in the phenomenon with the learning curve we're seeing in Waveform's near-stagnant results. This is contrasted with Vowel's neural network learning

curve, which indicates that our Neural Net had a very easy time digesting and disseminating (due to some kind of “intuitive” linear separability) through all of the attributes Vowel had to offer.

Decision Trees

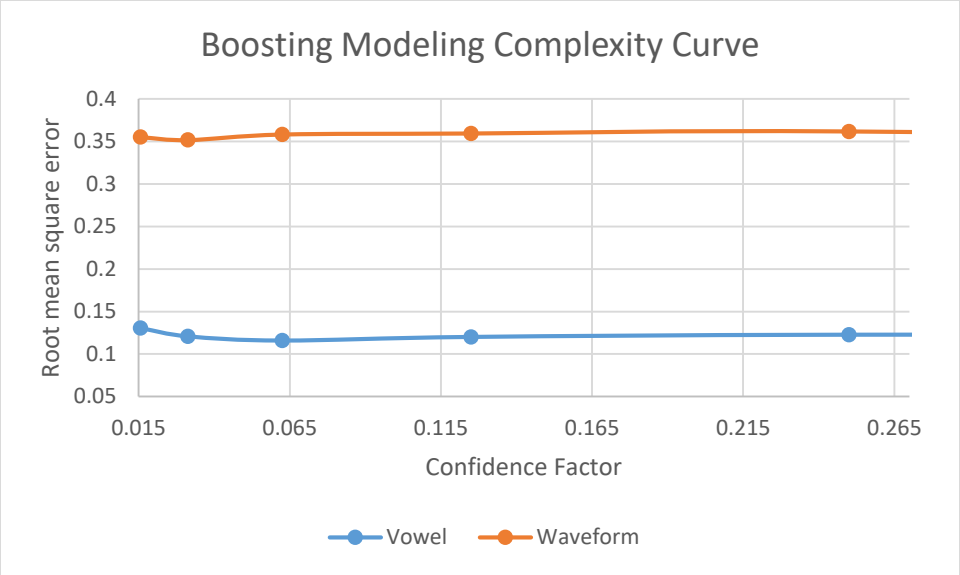
The decision tree type used is WEKA’s J48.





The critical confidence factor value for the decision tree algorithm on Vowel is 0.0625. For Waveform-5000 it is as close to 0 as possible (less pruning actually has the effect of increasing accuracy of prediction). I found this surprising – but after plotting the learning curve it is clear that the decision tree fails pretty hard on the Waveform-5000 dataset due to some kind of latent data noisiness and equivocal information gain on each node when building the decision tree.

Boosting



Boosting consistently provided better results for the datasets for all confidence factor values (which is to be expected). Boosting tends to dramatically increase accuracy at the cost of an increased risk of overfitting.

Something very interesting to me was that boosting reversed the trend found on the training data with the J48 algorithm (where error actually increased in tandem with pruning confidence).

General Comparisons and Analysis

Waveform-5000 was generally harder to train than the Vowel dataset due to the noisiness of the data. Neural Nets and KNN had the best results on Waveform-5000. Neural nets are capable The noisiness of the Waveform data made it hard for Decision trees to make meaningful splits, even despite pruning. Implementing boosting with the decision tree dramatically improved the performance of Waveform-5000 as pruning confidence increased. A reason why Neural Nets probably did better than SVM is that neural networks produce fixed-size parametric models; the scope of their search space is proportional to the sheer number of

hyperparameters that neural networks can accommodate. One thing I noticed about SVMs is that they always end up converging to a global minimum for the Waveform and Vowel datasets. KNN performed very well as K increased (as well as run time) due to it gradually acclimating to the noisiness of the Waveform data.

For the Vowel dataset, Neural Networks performed the absolute best (despite the abhorrent run times they tend to have, it was able to do great work with vowel), indicating a high degree of linear separability of the dataset. Neural networks have an advantage over SVMs because you don't have to guess very stringent hyperparameters for neural networks and it'll still be able to run. I'm sure there is a Kernel out there that would have resulted in much higher accuracy for these datasets (I like to think of them as pseudo-heuristics). Since I'm not a machine learning savant, I have no idea what this one would be. The effects of overfitting on Vowel were especially egregious with an increase in K for KNN due to latent biases in the data.

I utilized cross validation using 10 folds for these algorithms to help eliminate bias at the expense of run time. As evidenced by the comparisons between training error and testing error in the raw data table, cross validation consistently produced much more accurate results for both of these datasets.

I played around with learning rates and momentum for the neural network but the default WEKA settings yield the best results.

Waveform struggled a lot with decision trees (as pruning confidence increased and approached 1 the run times dramatically increased as well).

You can define which algorithm ran the best based off of how accurate it was in relation to the amount of time it took your computer to train on it (just like how we analyze space vs time complexity in algorithm creation).