Jackson Morgan
902937525
jmorgan45

Description of Problems

For both problems I browsed Kaggle for data sets that will work well in the case of making predictions based on inputs.

Dating Problem

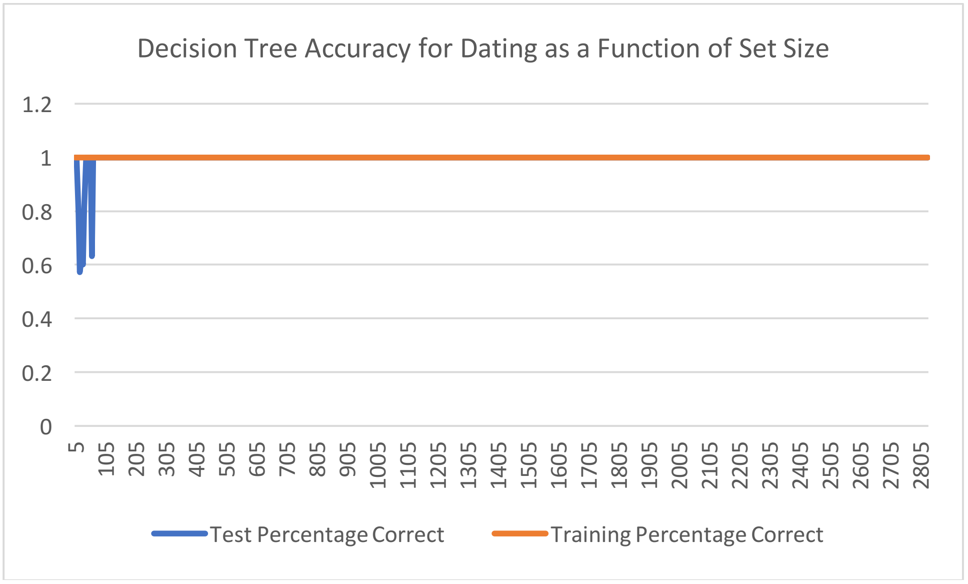
The first data set I found was one of speed dating data (https://www.kaggle.com/annavictoria/speed-dating-experiment). This data contains a large number of attributes that describe an individual and how the potential partner perceives that individual such as attractiveness, salary, race similarity etc. In dating there are a ton of different variables to determine if a possible match will be successful, and the goal is to predict accurately if a match will cause a match to happen. It will be interesting to see how this performs in different algorithms as this data set has a few attributes that are highly correlated to the classification and a wide number of attributes that simply provide noise.

Candidate Funding Problem

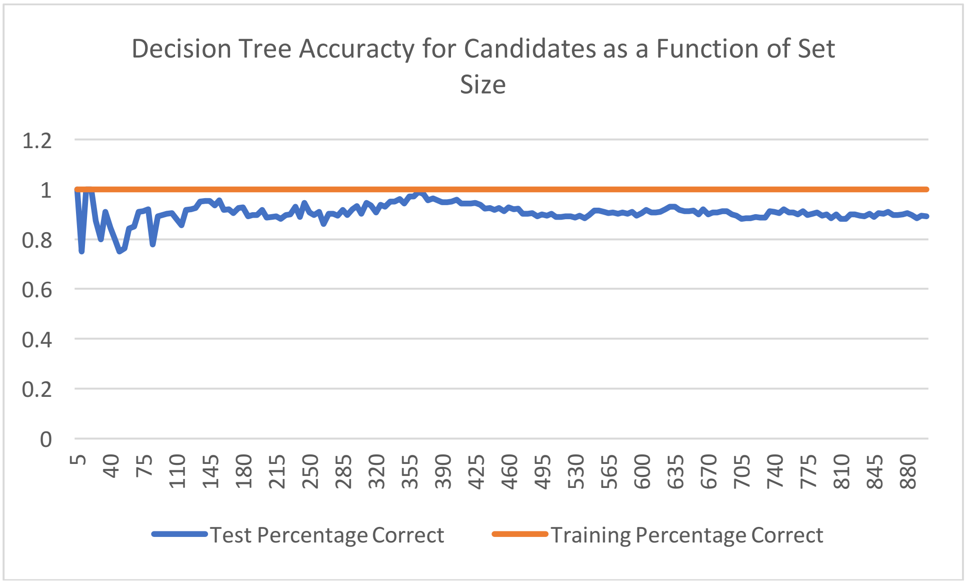
The second data set I found was one of candidate funding (https://www.kaggle.com/danerbland/electionfinance) in various elections across the country. In this case, I'm predicting if a candidate wins or loses the race simply from the amount of funds that he/she receives, not based on party affiliation as that would be a bit more trivial. This data set is interesting as it serves as a foil to the dating problem. Many different attributes add together to create the classification.

Findings

Decision Trees

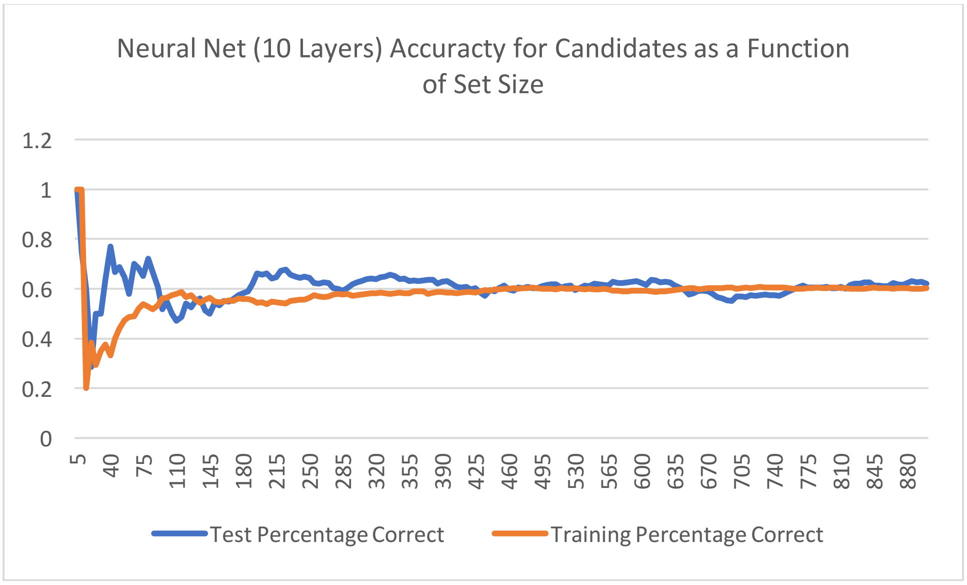


Time: 00:04.9

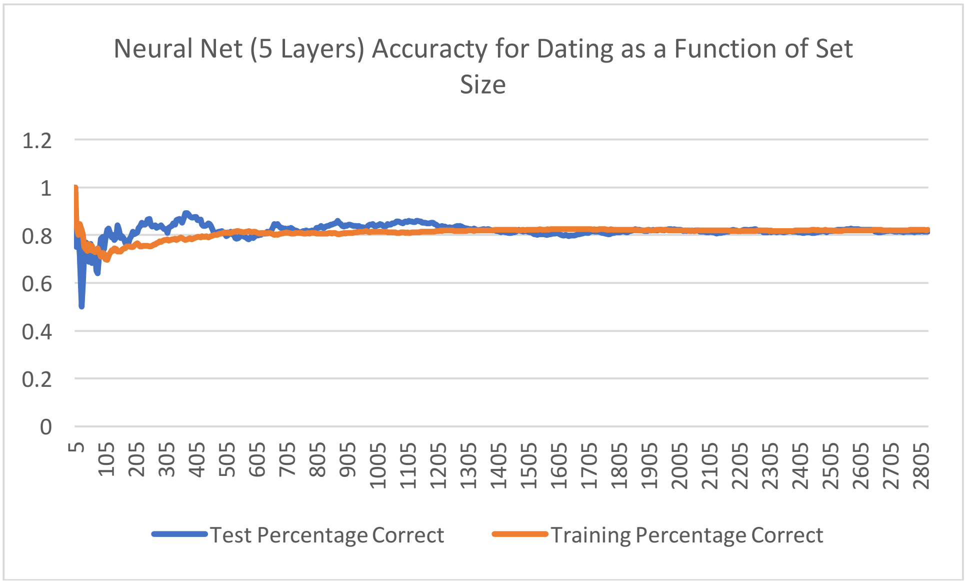


Time: 00:59.4

Neural Network

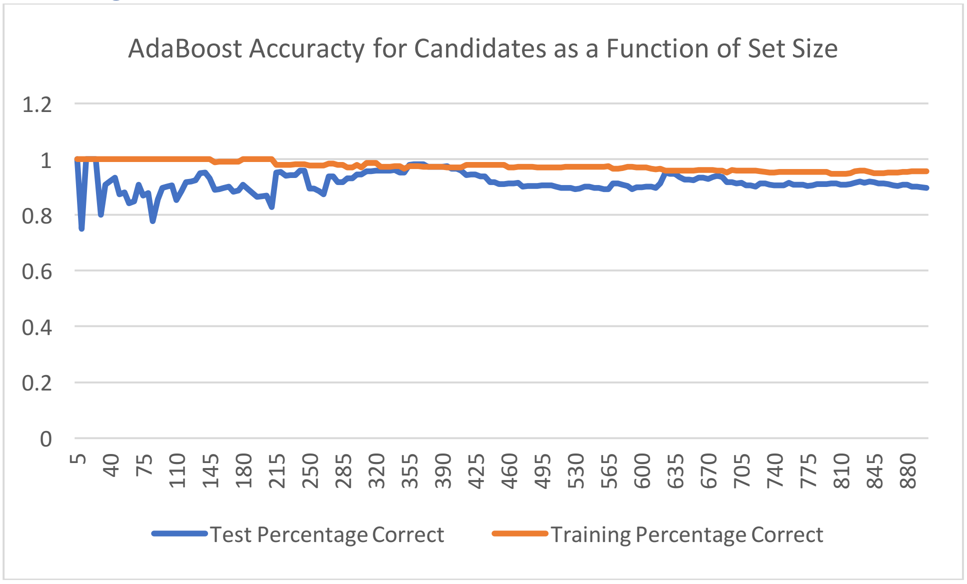


Time: 00:10.9

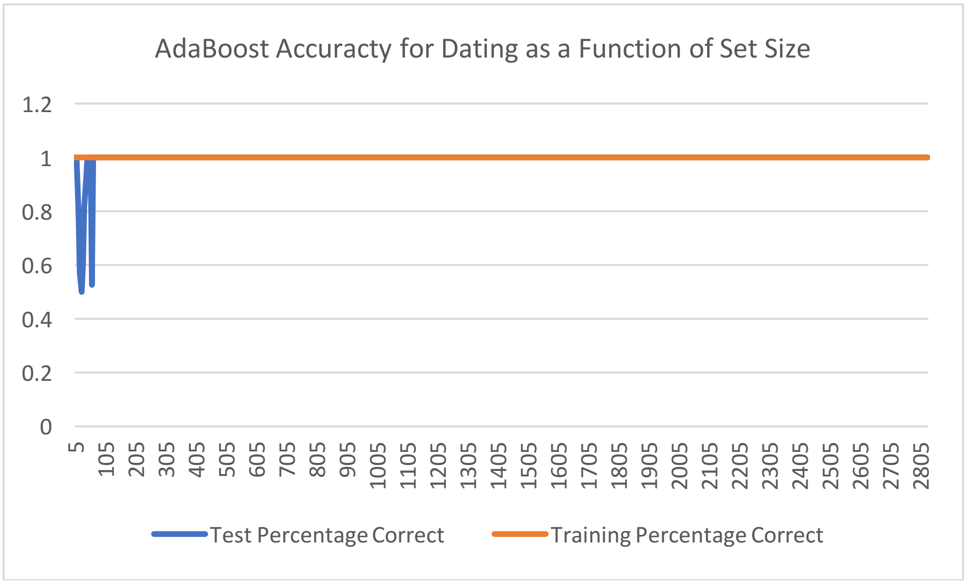


Time: 02:00.0

Boosting

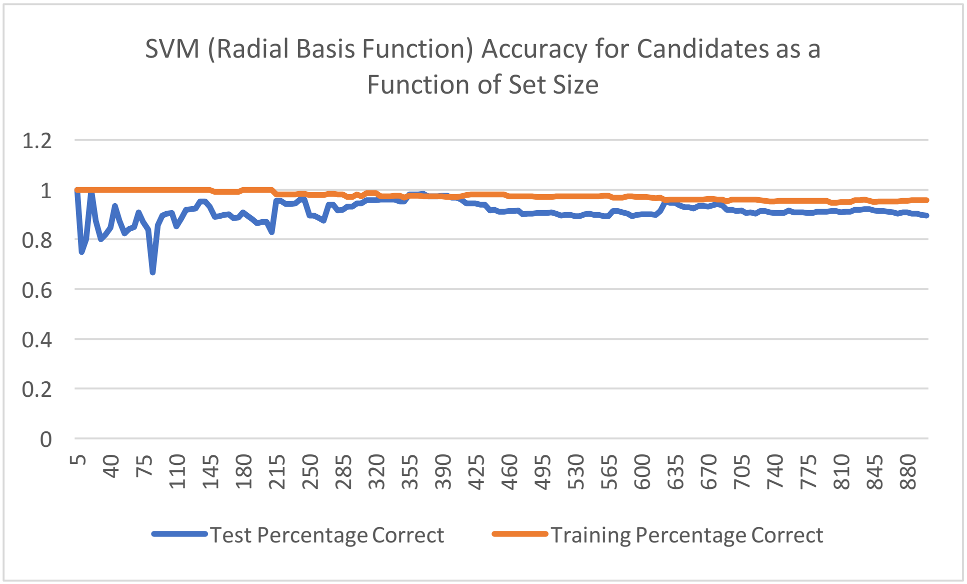


Time: 01:37.4

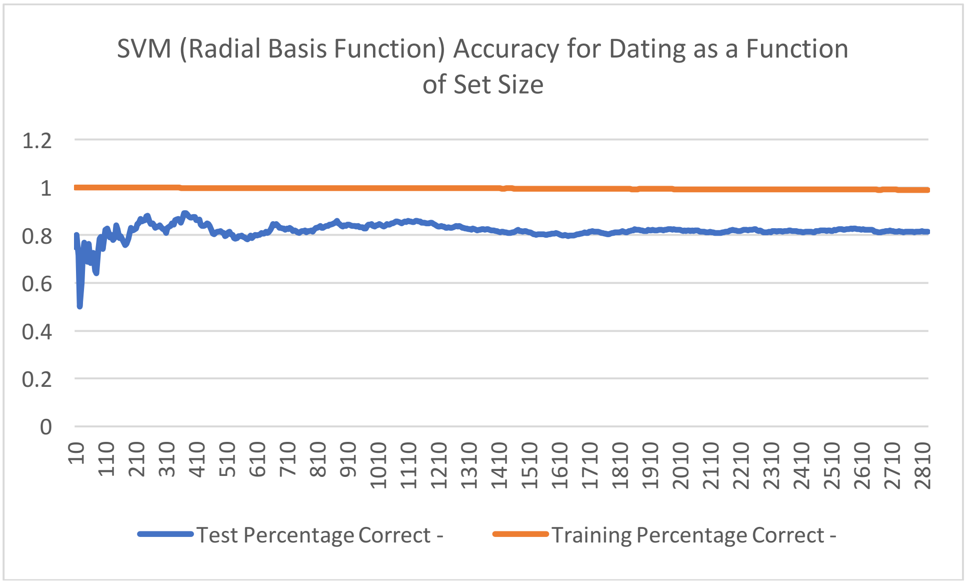


Time: 13:32.7

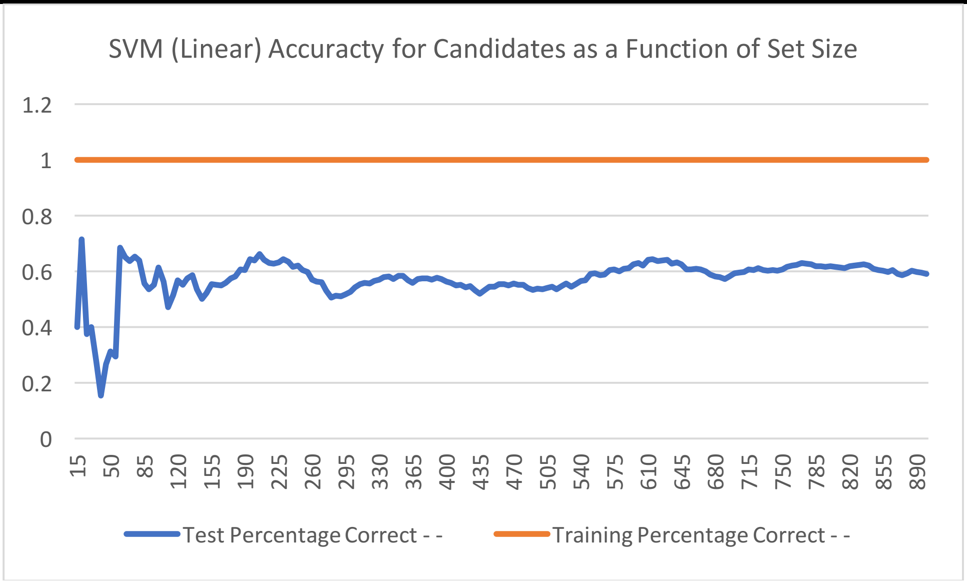
Support Vector Machine



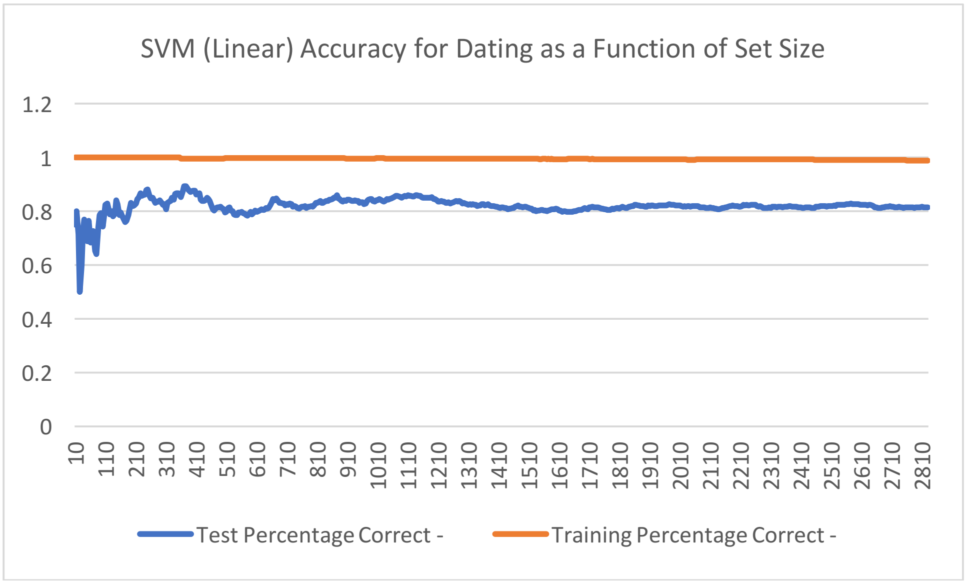
Time: 0:01:42.253859



Time: 0:05:24.162989

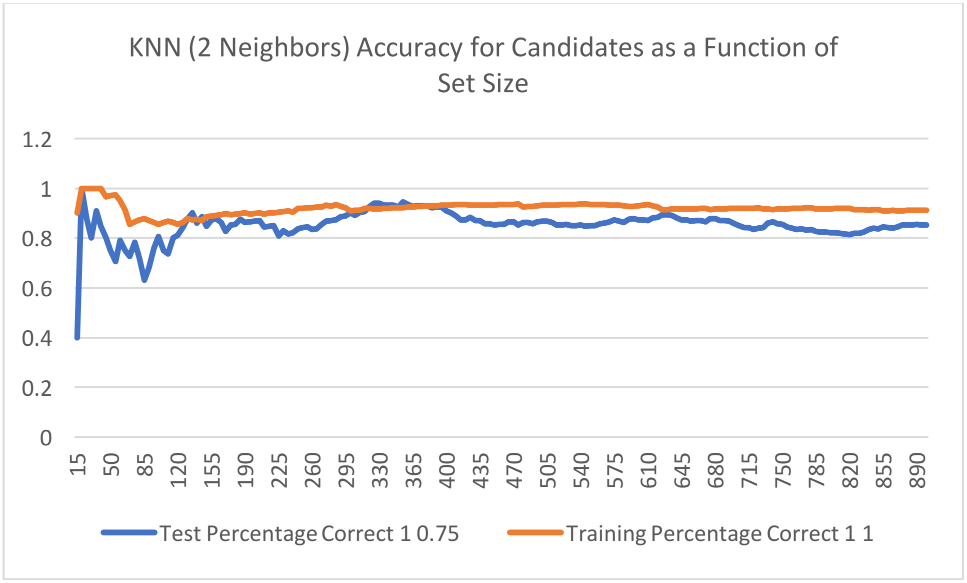


Time: 00:07.1

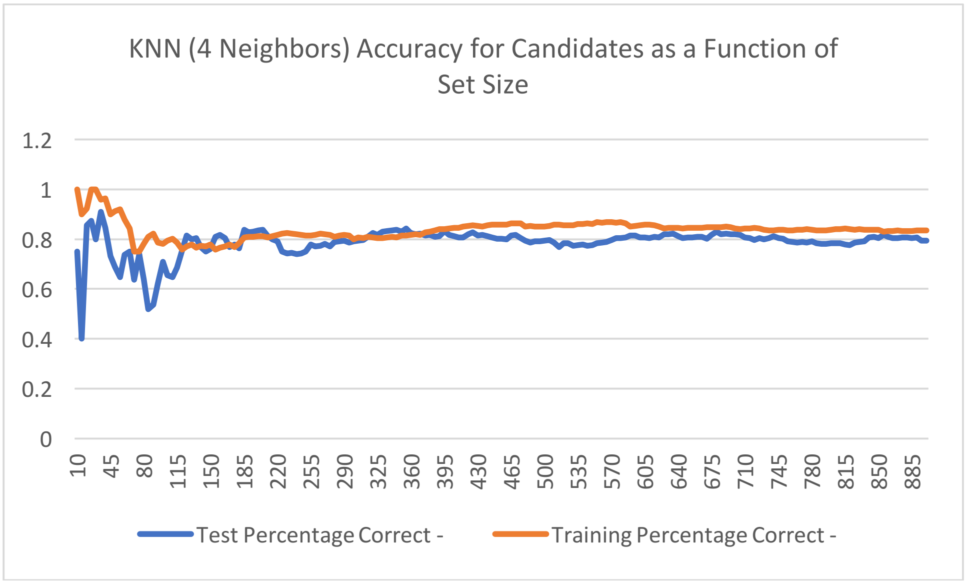


Time: 04:56.5

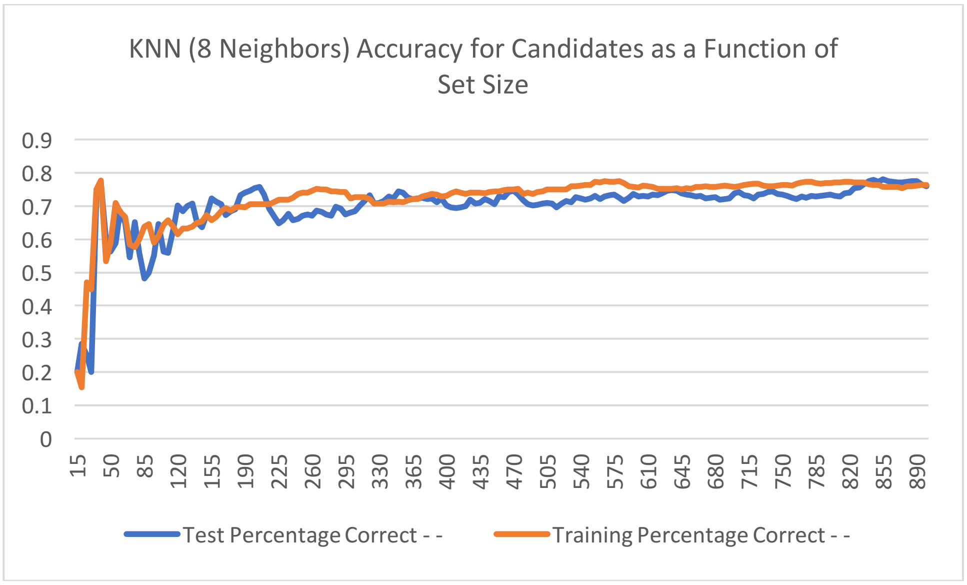
K-Nearest Neighbor



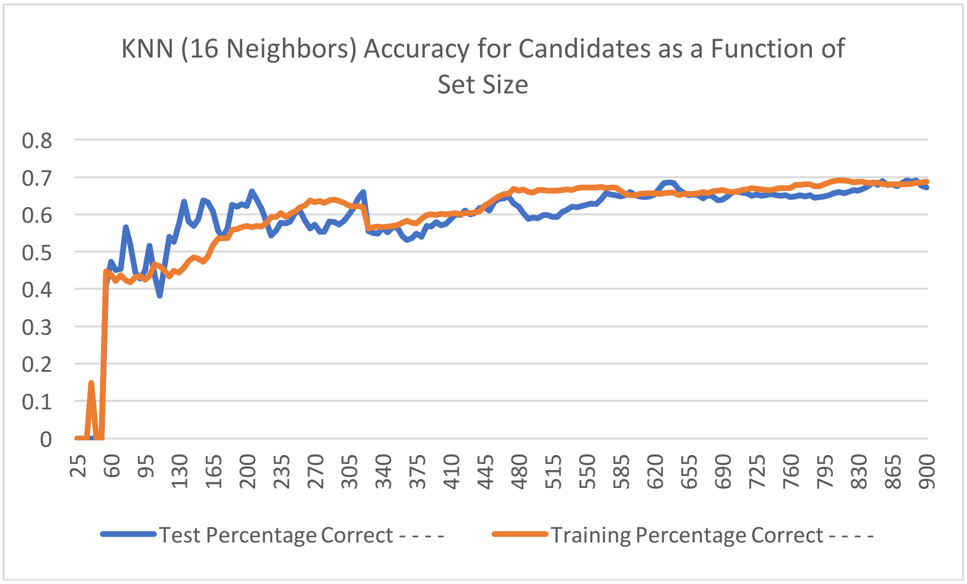
Time: 00:27.1



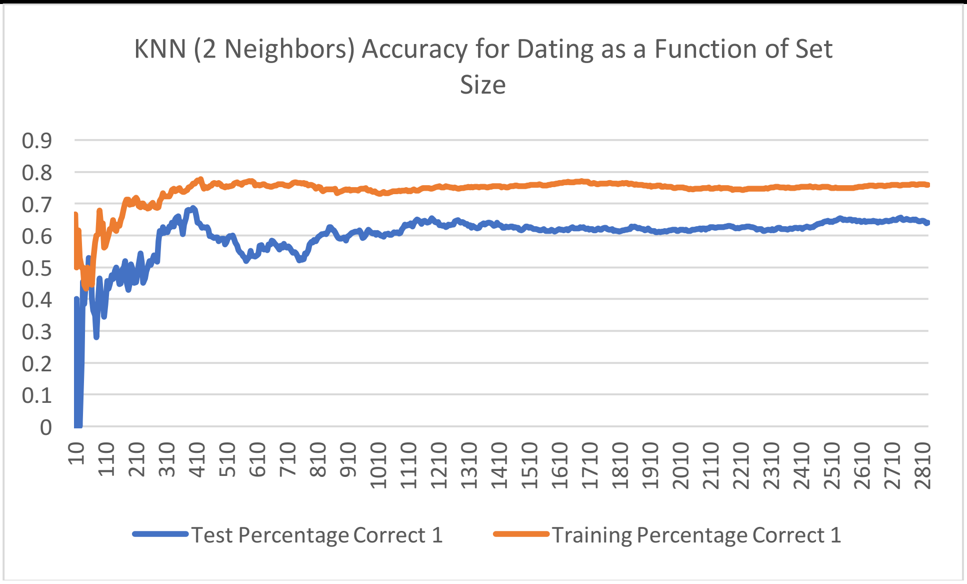
Time: 00:32.3



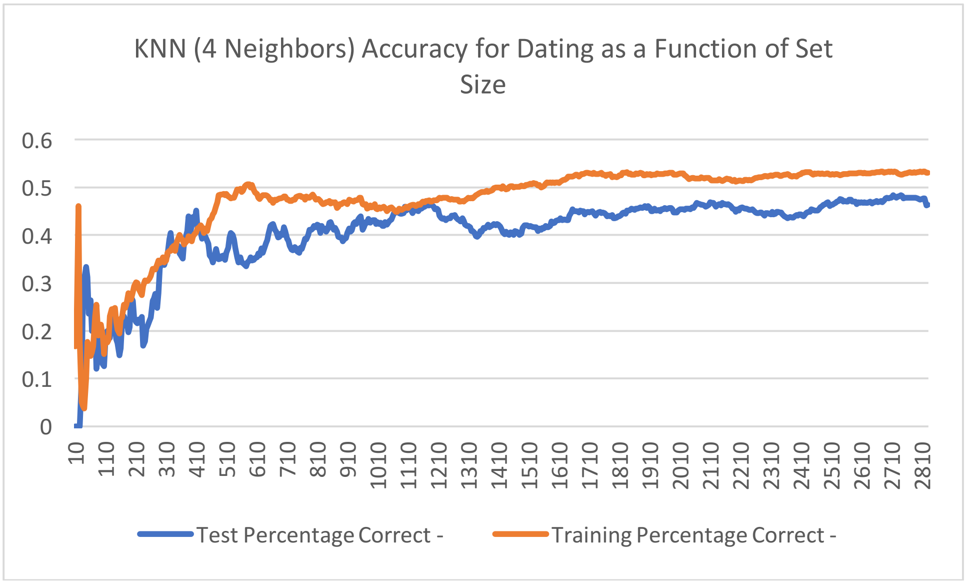
Time: 00:32.2



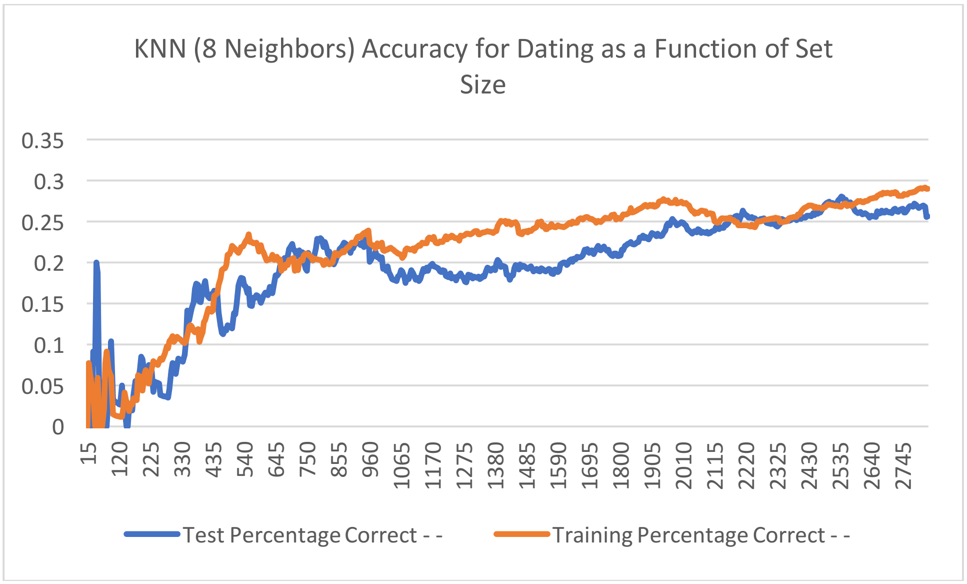
Time: 00:27.3



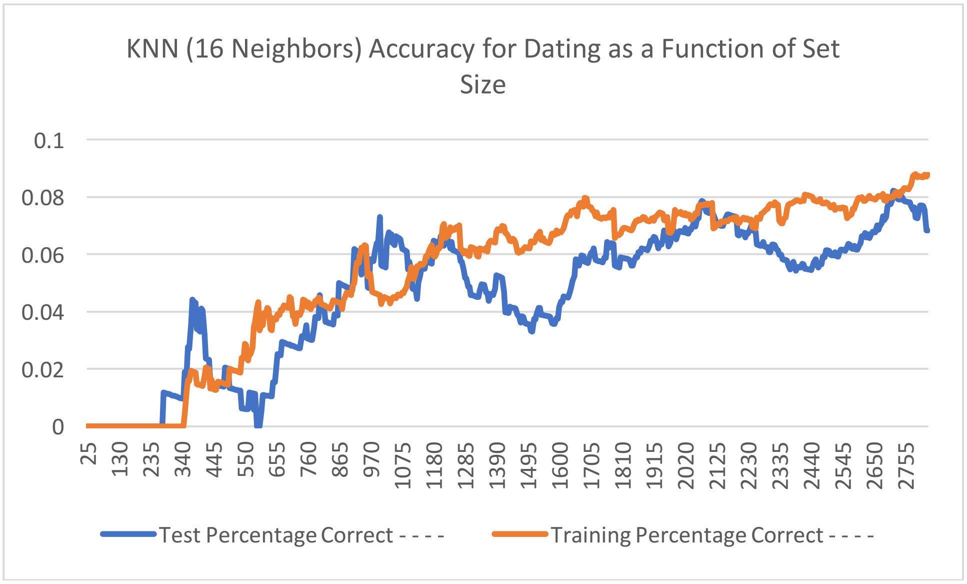
Time: 07:24.0



Time: 08:01.9

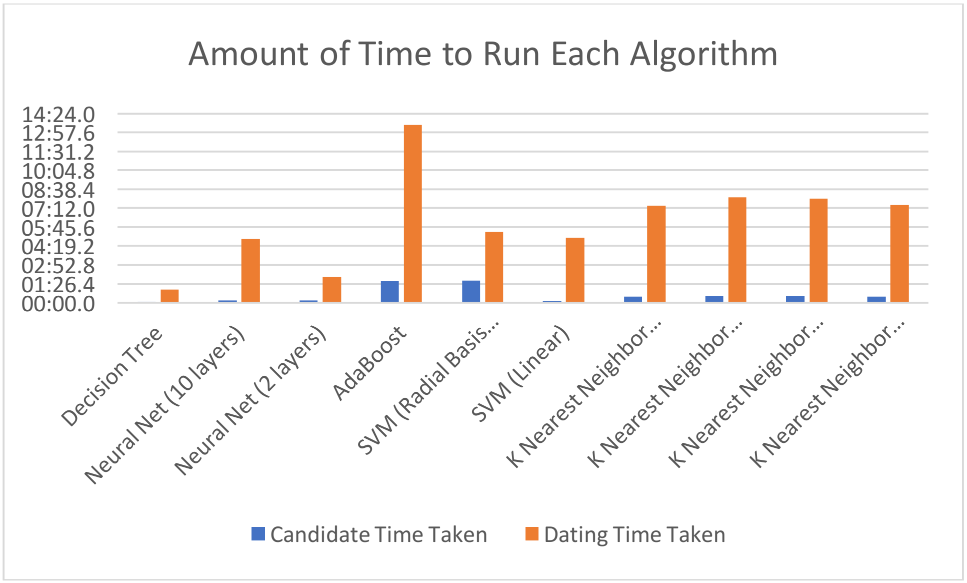


Time: 07:56.8



Time: 07:25.1

Amount of Time Taken



Algorithm Name	Candidate Time Taken	Dating Time Taken
Decision Tree	00:04.9	00:59.4
Neural Net (10 layers)	00:10.9	04:50.9
Neural Net (2 layers)	00:11.7	02:00.0
AdaBoost	01:37.4	13:32.7
SVM (Radial Basis Fuction)	01:42.3	05:24.2
SVM (Linear)	00:07.1	04:56.5
K Nearest Neighbor (2 Neighbors)	00:27.1	07:24.0
K Nearest Neighbor (4 Neighbors)	00:32.3	08:01.9
K Nearest Neighbor (8 Neighbors)	00:32.2	07:56.8
K Nearest Neighbor (16 Neighbors)	00:27.3	07:25.1

Analysis

Decision Tree

The most striking thing about the decision tree is algorithm is the 100% accuracy given to the data-set on dating. In order to explain this phenomenon, let's compare the two data sets involved:

In the candidate data-set, there was the possibility of having incongruent funds given to losing candidates. Often times candidates can be given large sums of money to campaign to a constituency that simply does not like their ideas. The accuracy of the candidate data set might have been higher for a decision tree if party affiliation and state were included. Such a decision tree would have probably branched on both of those as they have a strong correlation to the outcome. For example, a candidate running as a Republican in Alabama would have a higher chance at success than one running as a democrat. However, often times the amount of money reflects the election, and this allowed the accuracy of the predictions to hover around 90%. Though given party it might have been higher.

However, dating data apparently has a much stronger correlation, which is especially well treated by a decision tree. This makes sense as in practice there is a very strong correlation between whether an individual finds the other individual attractive, and if a match will be made. Decision trees will often work better for simpler problems such as the one defined above and is, therefore, not as affected by noise than the other algorithms below.

A simple decision tree by far took the smallest amount of time to process. Unlike the other algorithms. Decision trees are not concerned with updating weights, or aggregating weak learners, and therefore can be performed faster.

Neural Net

Neural Networks provided the least accurate outcomes of the 5 algorithms resulting in an accuracy of about 60% and 80% for the candidate data set and dating data set respectively.

The candidate data set was particularly difficult to even get to 60% accuracy, as I needed to tweak the number of layers. I found very little change by tweaking the number of nodes within the layers, but found that the accuracy of the neural net peaked with around 10 layers. According to Jeff Heaton, author of "An Introduction to Neural Networks," "the optimal size of the hidden layer is usually between the size of the input and size of the output layers." It just so happens that the input data set for the candidate data has 10 attributes, and therefore, at least for this set, the maximum accuracy of the neural net could be achieved by setting the number of layers equal to the number of input nodes.

This case is not true for the dating data set. While this dataset technically yields 70 input attributes, adding more layers saw little improvement on the overall accuracy of its predictions. Therefore, for efficiency's sake, I set the number of layers at 2 which is faster to train than a 10 layer net as fewer weights need to be updated each iteration. The difference between the two data sets that causes this is probably the relative number of attributes that matter. In the candidate data set, spending is split up into 10 categories that culminate in overall spending, and therefore, each category is important to the overall accuracy of the prediction. However, in the dating data-set only a few attributes that describe the attractiveness of an individual matter. So, weights associated with those attributes are focused on at the expense of weights for less appropriate attributes.

With a duration of about 5 times more than decision trees, and an accuracy less than decision trees, neural nets do not seem to be a good fit for these data sets. Neural nets are often designed to find **complex** relationships in data, and these data sets may not have nearly the complexity of relationships that would necessitate a neural net.

AdaBoost

At least for the dating data set, AdaBoost did not provide any improvement over a regular decision tree, but took a significantly longer time to run. AdaBoost employs a number of weak learners such as decision stumps (one level decision trees) to analyze an overall dataset. However, because the dating data has such a strong correlation between whether an individual found another attractive and the success of a match, a larger number of decision stumps does little to augment the information already created by the regular decision tree. However, because each needs to be created, AdaBoost required more than 10 times the amount of time to run. Originally, I ran AdaBoost with 50 estimators, however that proved to be egregiously long. I then ran it with 10 estimators which came out to a reasonable time of 13 minutes.

AdaBoost did, however, slightly improve the accuracy of the candidate data set. While the full data set yielded a similar accuracy at about 90%, for other data set sizes, accuracy was slightly improved. What didn't work for the dating data set, worked here. Because of the large number of attributes that contribute to outcome of the candidate data set, a large number of weak learners provided a boon to the accuracy.

Support Vector Machines

Support Vector Machines with radial basis function yielded the highest accuracy for the candidate training set, yet a relatively low accuracy for the dating set. An explanation for the low accuracy can be seen by looking at the accuracy of the training set which is nearly 100%. Because lines were drawn to fit the vectors of the training set, the classifier used the vectors for data points that didn't matter as much. This noise then affected the accuracy of the test set because vectors that are one classification could be labeled as another classification as they might share similar attributes where the attributes don't matter.

Yet, as all attributes matter for the candidate training set, and there is a correlation between the amount of money spent and the success of the candidate, support vector machines were able to accurately draw lines to classify the groups.

Finally, the same success can't be claimed for the SVM algorithm using a linear kernel. While the dating set had a similar accuracy probably due to the reason cited above, the candidate data set performed significantly lower at about 60%. Yet, the training set performs at 100% accuracy, meaning that it is easier to get overfitting with linear kernels than with a radial basis function.

K Nearest Neighbor

The most surprising finding regarding the K nearest neighbor algorithm was that the number accuracy decrease in both cases with an increase in the number of neighbors to detect, but upon further thought, it made sense for both of these data-sets. A data set would improve in accuracy given the requirement of more neighbors if there were a large number of outliers in the data. This is not the case here as both datasets have a relatively good connection between attributes and the actual classification. When this is the case, checking more neighbors would be a flaw as the check could accidentally reach into a region that's classified as something else. This is entirely dependent on the subset of data that is used, and this is why the accuracy of the test data follows a more sporadic curve as the number of items in the set increases than other algorithms.

Another thing to note with K-nearest-neighbor is that the time to execute does not expand with the number of neighbors being detected. This is because the running time is the same due to the fact that the number of neighbors is a constant. While the algorithm might be making more comparisons. This is negligible compared to the number of data points that need to be processed.

Overall findings of this study have indicated that for a data set like the dating data set, where there is a strong link between a few attributes and a classification, a decision tree is the best option in both speed and accuracy. A data set like the candidate data set with multiple attributes all contributing to the classifier in the same way, a Support Vector Machine performs well due to its ability to correctly group the vectors that represent all the attributes.