William Ma

Professor Charles Isbell

CS 4641: Machine Learning

2 February 2017

Supervised Learning

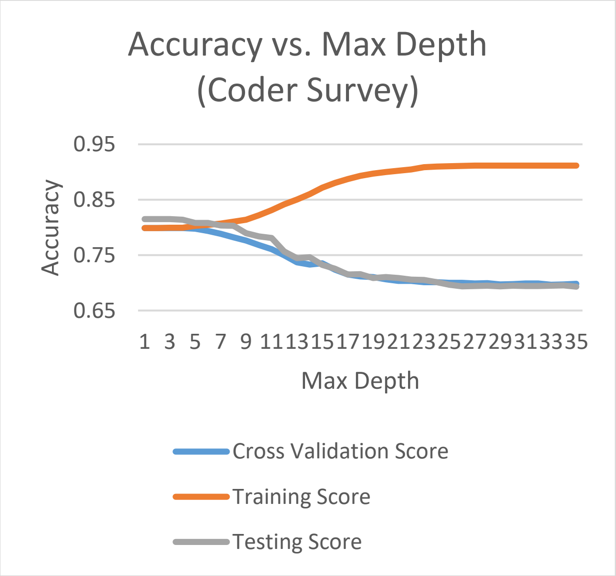
Overview: In the following pages, you will find detailed analyses of five different supervised learning algorithms. For example, I will explore the relationship between the values of various hyperparameters and their corresponding algorithms. However, there are a number of things that will remain invariant throughout the analysis in order to isolate and focus on the topic of analysis; in the previous example, that would be the hyperparameters. One invariant is the datasets used, as described below:

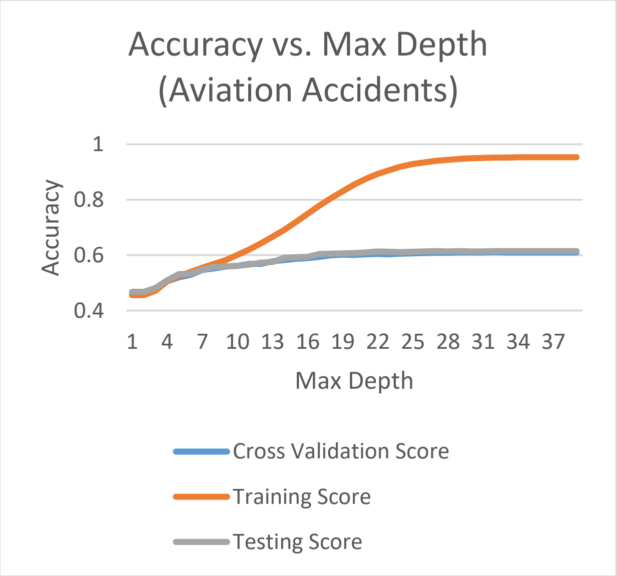
- **2016 New Coder Survey:** This dataset represents the results of Free Code Camp and CodeNewbie's 2016 collaborative survey, intended to reach out to people who are actively learning to code, and attempting to learn about their motivations, their methods, their demographics, and their socioeconomic backgrounds. Free Code Camp and CodeNewbie reached out to them via Twitter and email, and received a flood of responses. 15,655 people responded to the survey's 48 questions; this dataset represents all 15,655 people's responses. We will be using age, income, whether someone is an ethnic minority, and whether someone is a software developer to predict their gender. This dataset is provided under the copyleft Open Database License, and was sourced for this paper from kaggle.com.
- **Aviation Accident Database & Synopses:** This dataset is the aggregation of the National Transportation Safety Board (NTSB)'s data on civil aviation accidents and incidents in the United States, its territories, and international waters since 1962. It includes data on over 70,000 accidents and incidents. Since it is likely that most air accidents will occur near populated areas, we use latitude and the extent of damage to the aircraft to predict a longitude class, divided into 18 classes of 20 degrees each, and consequently the approximate area of the crash. Released by the NTSB under the hopes that it could be used to improve the safety of traveling by airplane, the database is available under a Creative Commons, CC0: Public Domain License, and was sourced from kaggle.com.

All analyses of the five algorithms will be performed on the above two datasets. Furthermore, for analysis of hyperparameters, that is to say, model complexity, our training and test sets will be partitioned from the data in a 70% / 30% ratio, respectively. There will be additional information on the effect of cross validation on accuracy, which will be performed consistently with 10 folds. The analysis will be performed using model complexity curves, followed by learning curves.

Decision trees: Decision tree learning is a highly intuitive supervised learning algorithm that attempts to take a dataset and infer a number of rules or questions, corresponding to if then else constructs, wherein the root node is an initial question, which based upon a result allows one to progress farther down the tree, asking more and more questions and coming closer to an answer. The algorithm is fairly simple to understand, but can be unstable, creating completely different trees for even the most minute changes to the data. Furthermore, decision tree learning may generate overly complex trees for a dataset, potentially leading to overfitting. We use scikit’s DecisionTreeClassifier to implement this type of learning.

Pruning: We implement pruning for a decision tree algorithm in order to reduce the complexity of the tree and therefore overfitting. Pruning for the below analysis was done by enforcing a max depth; though there exist other methods for pruning the decision tree, the improvement in accuracy was negligible, and for the sake of brevity they have thus been omitted from the analysis. Max depth will thus be the only method of pruning.

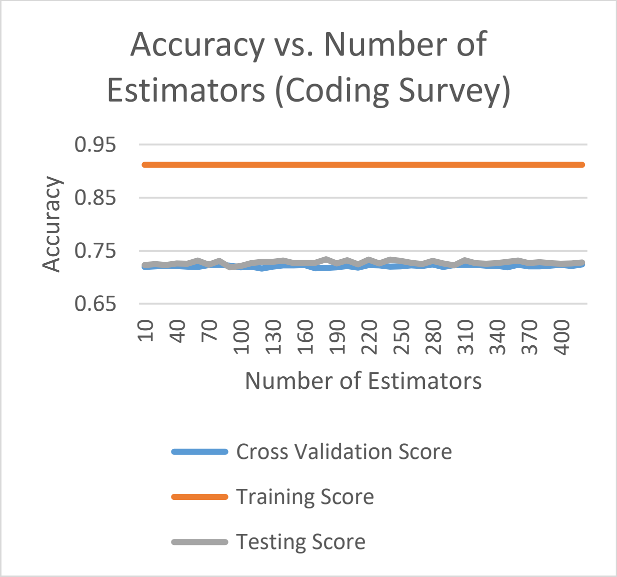


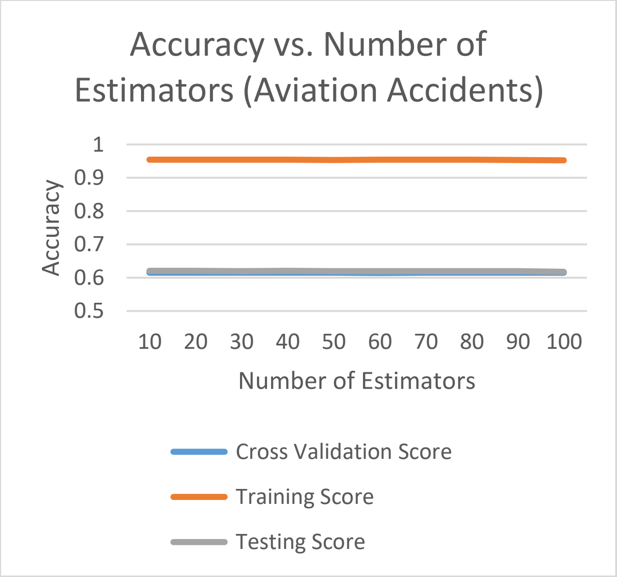


Above we can see the effect of pruning on the accuracy of our decision tree algorithm, which seems to have a different effect on each of our datasets. One will quickly note that while the algorithm’s accuracy on the coder survey decreases as max depth increases, when it comes to aviation accidents that is not the case; in fact it increases asymptotically. Why might this be the case? The answer lies in the fact that while the behavior of the training and testing scores differ from one another, the training score's behavior is consistent across the two graphs. This would imply that although both models are overfitting as our max depth increases, the impact this has on the accuracy of predicting aviation accidents is negligible as compared to that of the coder survey, as the afflicted branches seem to not even become relevant in the final decision.

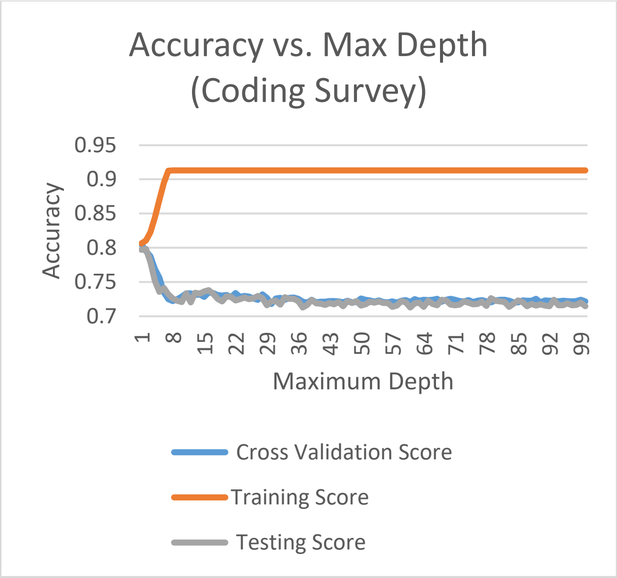
When it comes to picking the optimal values of max depth for our datasets, I chose to go with the best intersection point of all three scores, which for the coder survey was a max depth of 6, and for aviation accidents was a max depth of 8. The hope is that these intersections will offer the most consistent performance in a real-world scenario.

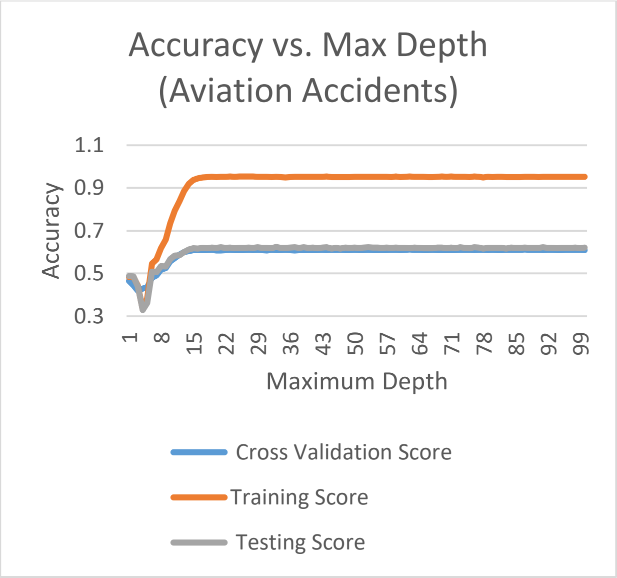
Adaptive boosting: We now experiment with boosted decision trees, in particular adaptive boosting, or AdaBoost, as included in the scikit library. AdaBoost, a modified decision tree algorithm, has a hyperparameter in the the number of estimators, which limits the maximum number of estimators allowed before boosting ends. We experiment with this number below, at the default max depth, which is none.





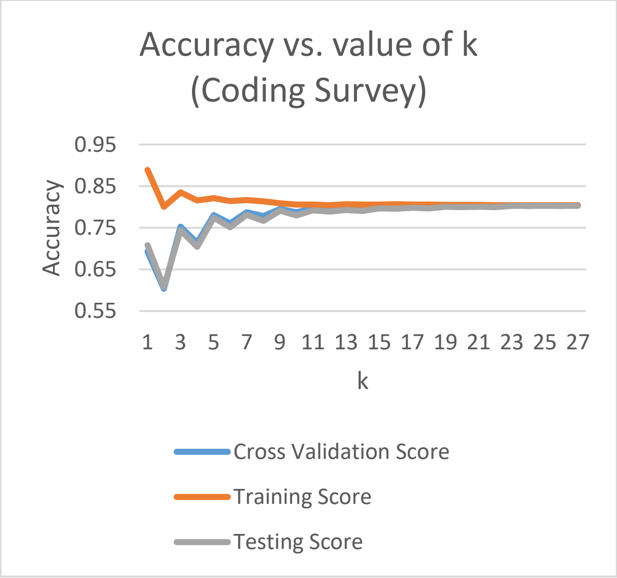
Perhaps sadly, analyzing the effect of the number of estimators on our accuracy yielded no useful insights; it seemed to have almost no effect on the number of estimators. Therefore, we can only recommend the default value for the number of estimators, which is 50. We thus move on below to analysis of pruning on an adaptively boosted decision tree. For the same reasons as for our nonboosted decision tree algorithm, and to maintain consistency, we again use max_depth as our hyperparameter of choice to implement pruning.

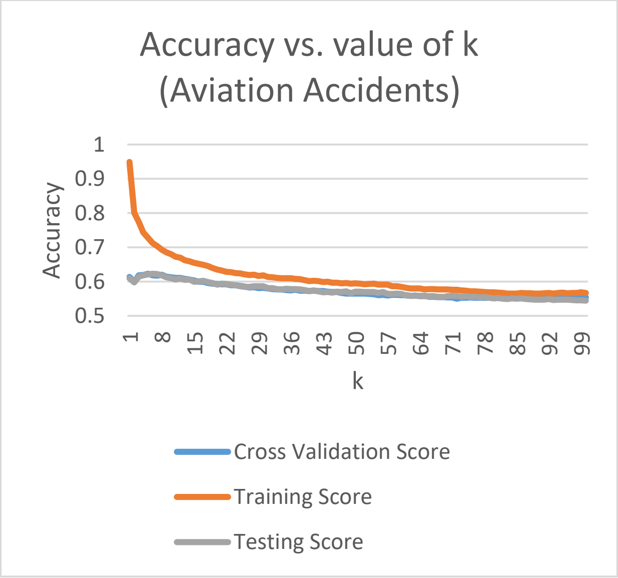




With the above, we can clearly see asymptotic trends on accuracy as we increase our maximum depth. Since ideally, we'd like to minimize overfitting, and therefore our model's susceptibility to unseen data, we would ideally pick maximum depth values which correspond to the inflection points in the graph where the curves begin to become asymptotic. For the coding survey, this corresponds to a max_depth of 8, and for aviation accidents this would be 15.

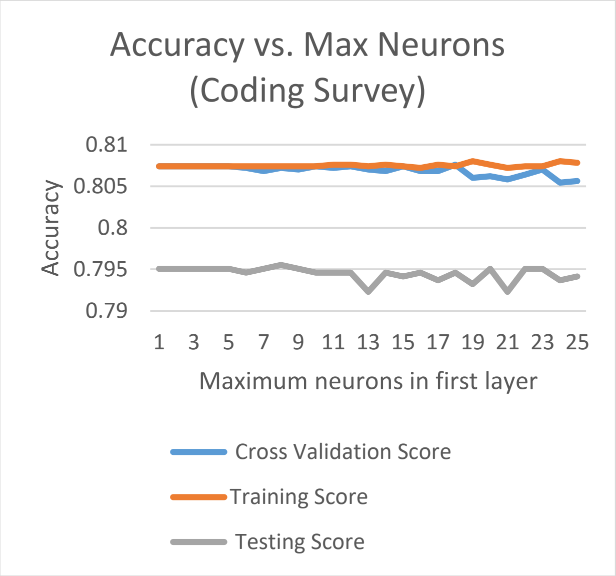
k-nearest neighbors: We now move on to the k-nearest neighbor learning algorithm, a comparatively lazy algorithm compared to the others discussed in this paper, but performs excellently in a number of problem domains for its simplicity. The algorithm makes predictions for a point of given input by looking at the k nearest data points and making a decision on what to predict based off of those k nearest points. The potential for tuning, then, and thus the hyperparameter, lies in the value of k that is used for a particular dataset.

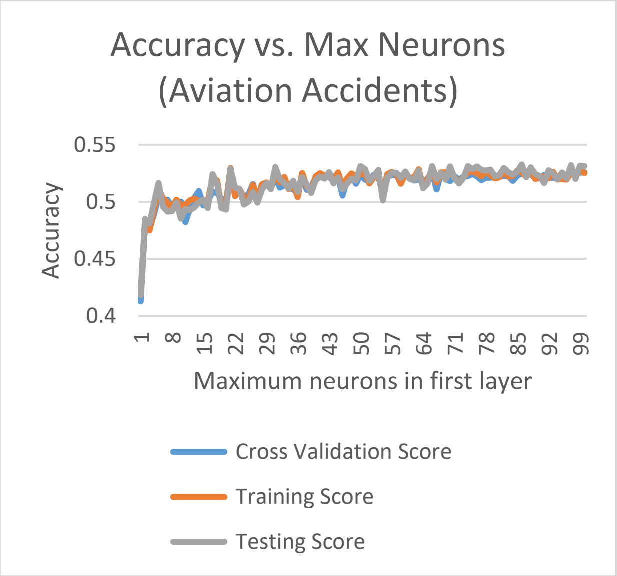




The above demonstrates the relationship of k to the accuracy of our predictions. For the coding survey, accuracy scores in every category converged to about 80%, so the point at which they became near-indistinguishable from each other determined my value of k for that one in particular. Thus, the value of k I picked for the coding survey was 20. On the other hand, for our aviation accidents dataset, the values do converge, but at a lower accuracy than is potentially possible. In fact, there is a small local maximum at around k=6, after which overfitting reduces accuracy asymptotically to about 54%. This local maximum, however, gets us about 62%, a substantial improvement compared to that overfitted 54%. Thusly, the value of k I selected for the aviation accidents dataset was 6.

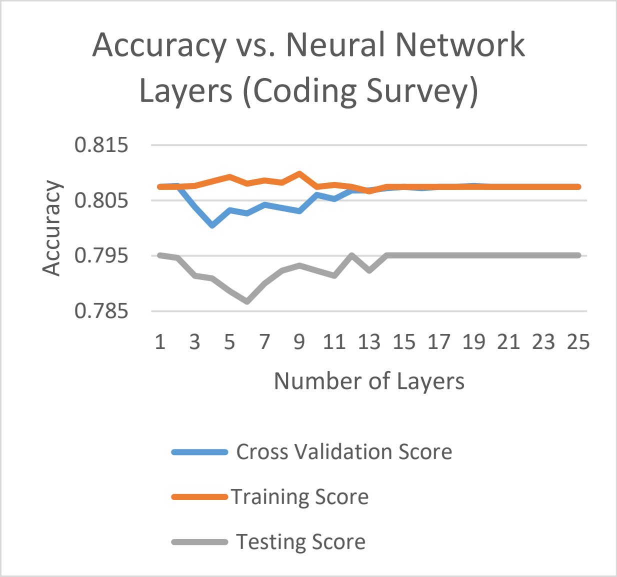
Neural networks: Neural networks represent a very powerful supervised learning algorithm, with incredible complexity and predictive power. For the purposes of our analysis, we use scikit's multilayer perceptron (MLP), a feedforward neural network model through a weighted directed acyclic graph, wherein the edges are reinforced or reduced by each piece of data that is added to be learned from. That is to say, based on the amount of error for each piece of data introduced, the graph is iteratively updated and improved to best cover the new complete set. There are a number of layers, with an input layer, an output layer, and a varying number of hidden layers in between. This varying number of hidden layers represents a point for tuning, and a hyperparameter. However, there exists another hyperparameter and optimization in the maximum number of nodes, or neurons, that are allowed in each layer. This is where we begin.

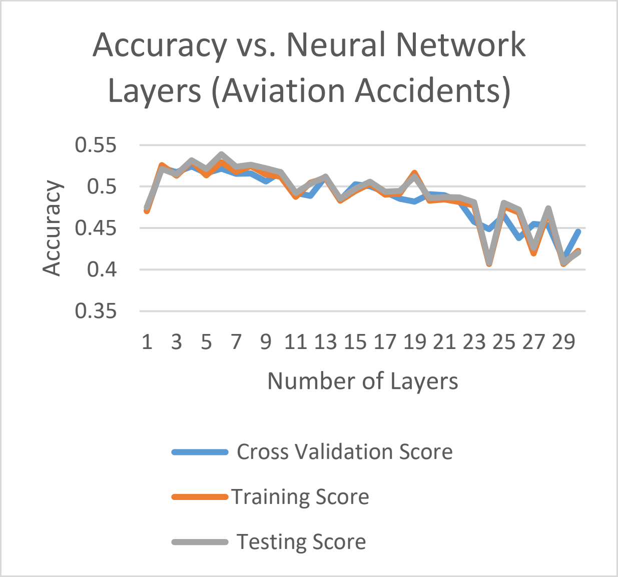




From the above, it may be difficult to see a significant relationship between the accuracy of our algorithm and the maximum number of neurons that we allow. However, when it comes to the graph on the left, we'd like to see the best accuracy we can, so ideally we'd pick the point of divergence as where we'd like to settle our maximum number of neurons. Thus, we pick 18 as the ideal number for maximum neurons. As for the aviation accidents dataset, we can see that the graph seems to grow almost logarithmically, so we ideally want a best compromise between accuracy, which grows slower and slower as we add more neurons, and runtime, which will increase substantially as we add more neurons. Therefore, we pick 45 as the optimal number of neurons.

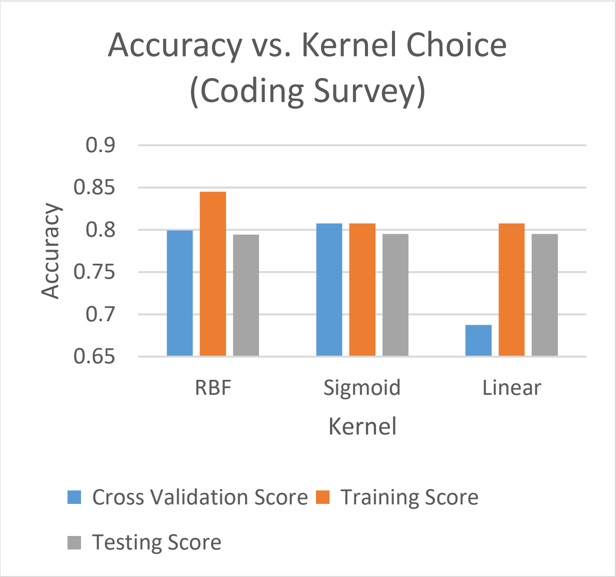
However, as we stated before, this maximum number of neurons at the first layer is not the only place we can optimize. We can also optimize by specifying the number of layers at which this maximum number is enforced. Analysis of this hyperparameter is described below.

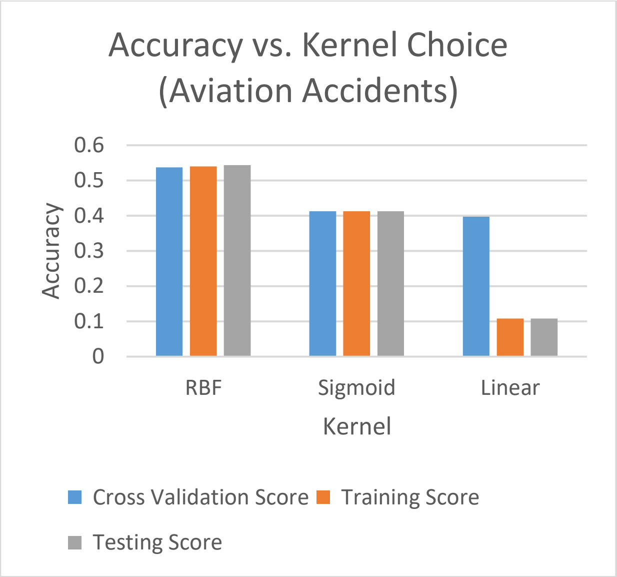




The relationships shown above show clearly where overfitting as a result of excessive layers begins to hurt the accuracy of the algorithm, or cease to improve it, as is the case for the coding survey. Since we would like to minimize this overfitting, for the graph on the left we pick our number of layers as the point at which the cross validation and testing score become asymptotic. For the graph on the right, corresponding to aviation accidents, since accuracy seems to be oscillating in a slowly decreasing fashion, we'd ideally want to use the number of layers at which accuracy is highest, while the scores still line up. The number of layers that correspond to this description is 6.

Support vector machines: The final supervised learning algorithm we will discuss is that of the support vector machine, that categorizes the training data by finding the best fit parameters for a specifically chosen kernel function that will try to map various spaces of the data as belonging to a certain category. The support vector machine we will be using is scikit's SVC implementation, of which we have picked 3 different kernel functions to test, the radial basis function kernel (RBF), the sigmoid kernel, and the linear kernel. This choice of kernel will be our hyperparameter to test with our datasets.





From the graph on the left, we can see that the kernel choice seems to have a variety of effects on the dataset. To an untrained observer, it may be difficult to pick between the sigmoid and RBF kernels, as they seem to have around the same accuracy, with the RBF having a much higher training score. However, a larger training score does not offer the same benefit, as say, a higher cross validation score would, since a larger training score may simply imply a large amount of overfitting. It is for this reason that the sigmoid kernel was my kernel of choice for the coding survey dataset. For the aviation accidents dataset however, it is much more clear, and RBF is picked handily over the other kernels, reliably performing substantially better in all scores.

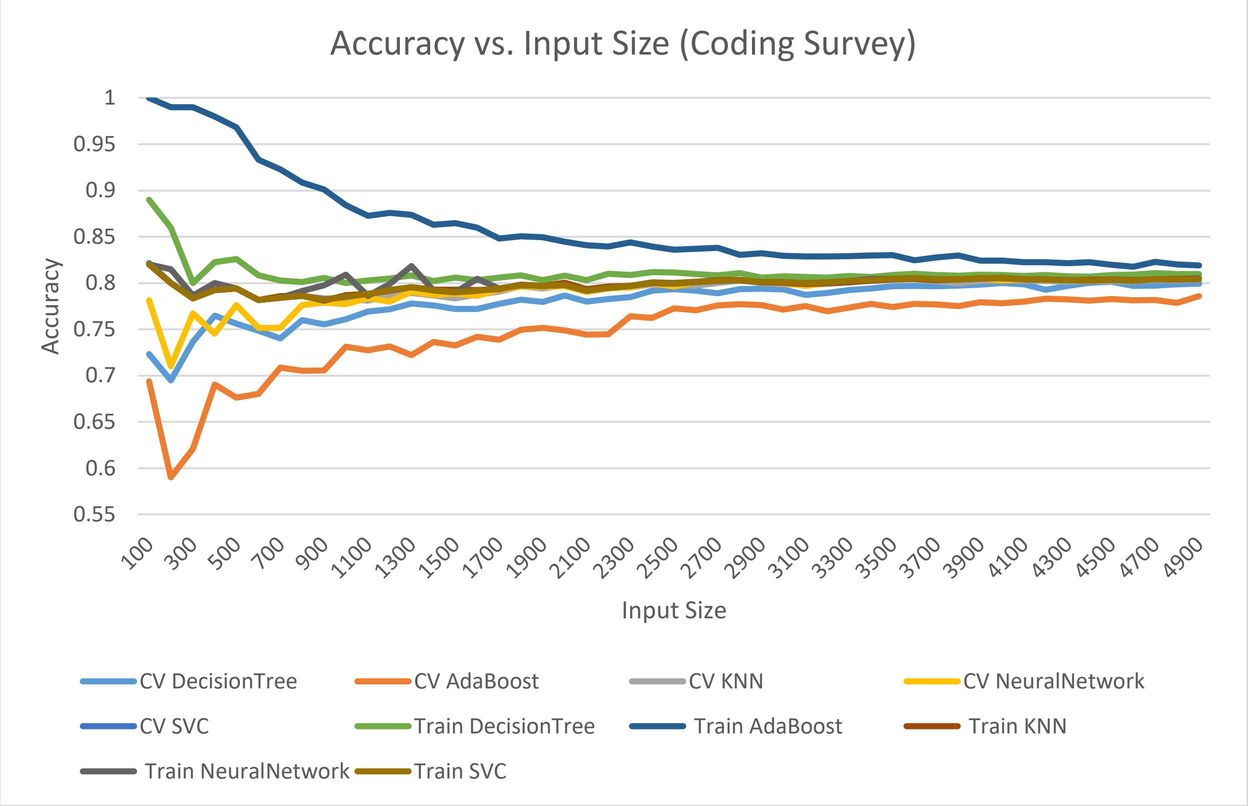
Now allow us to aggregate all of our results for each of the algorithms in the table below:

Classification Algorithm	**Hyperparameters**	
	Coder Survey	**Aviation Accidents**
Decision Tree	Max_depth = 6	Max_depth=8

AdaBoost Decision Tree	n_estimators = 50,
	max_depth = 8
		n_estimators = 50,
		max_depth = 15

k-nearest neighbor	k = 20	k = 6
Neural Network	nodes = 18, layers = 14	nodes = 45
Support vector machine	Sigmoid kernel	RBF kernel

Learning: With these hyperparameter values, we can now analyze our learning curves.

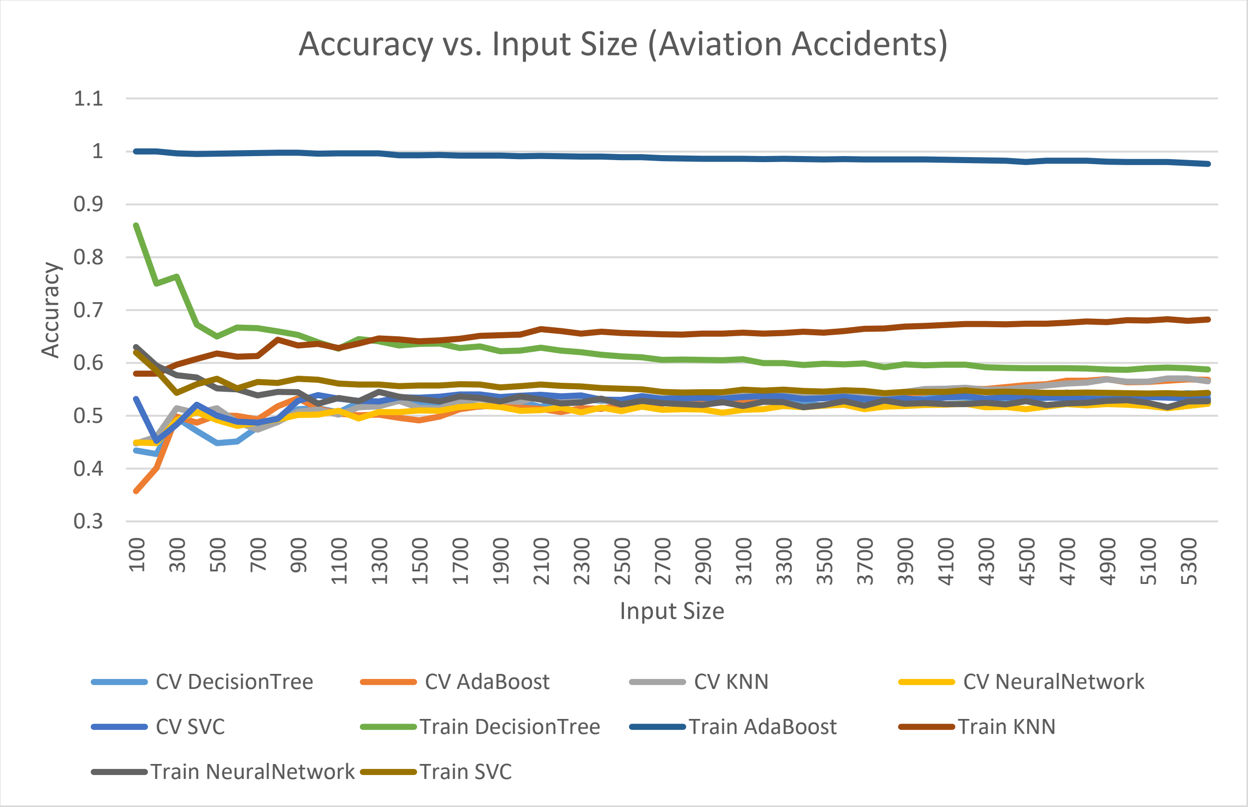


The above graph contains all the learning curves for all five classification algorithms; that is to say, for each of our algorithms, there are two curves corresponding to the cross validated score of the algorithm, and training score of the algorithm, each graphed to determine the relationship between those scores and the size of the input.

One thing we will note right away about the results of this learning curve is that our dataset clearly exhibits a large bias. That is to say, the accuracies for all of our algorithms seems to converge towards a single asymptote at around 80%; the fact that this is evident at around the 5000 data point mark tells us that the dataset suffers from high bias; adding more data will not improve the accuracy of our algorithms very much, if at all.

With the optimization of hyperparameters we performed on the various classification algorithms for this dataset, with the learning curve above we can now pick a hopefully ideal algorithm for predicting this data set in the manner that we have. However, since all the algorithms seem to converge in accuracy, it may be hard to distinguish directly from the graph which algorithm would be best. On the basis of accuracy, the best performer is actually a tie between the KNN, Neural Network, and SVC algorithms.

Because there is a tie, accuracy-wise, what comes next is judging the algorithms based on how fast they ran. The fact of the matter is that KNN runs, based on personal anecdote, exponentially faster than either Neural Network or SVC algorithms, so I must recommend KNN as the ideal algorithm for this dataset, achieving parity in performance with far more expensive (in both time and space) algorithms.



Above we find the same type of graph containing learning curves as we did for our coding survey dataset; in fact, we seem to see largely the same relationship across the two. Both graphs show the performance of various algorithms as a function of the input size, and both of them seem to be converging asymptotically, for the most part.

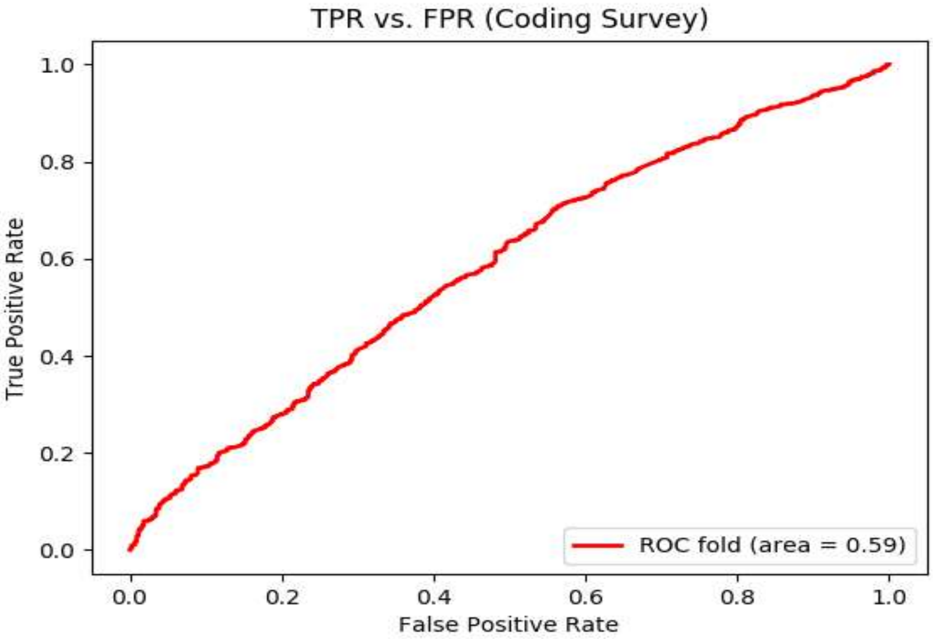
The main difference, that one will notice immediately, is that AdaBoost's training accuracy seems to soar well above the others. The reason for this is an interesting perk of the dataset; while there are 18 classes that the data is split into, much of it is concentrated around just a few classes. This perhaps ought to be expected, just by the nature of the data, since air accidents are more likely to occur around populated areas, which describe a rather limited set of longitudes in comparison to the total range.

On this basis, we can largely ignore that anomaly and focus on the cross validation scores that seem to converge just above 50%, more precisely, at 57%. For largely the same reasons as the previous dataset, we can say that this dataset also suffers from high bias. However, this claim is much more tenuous on this dataset than it is on the former, since the accuracy of the algorithms doesn't seem to converge nearly as quickly or consistently. It may very well be that this dataset in fact suffers from high variance, and more data would result in a more accurate and complete model. The issue that arose with AdaBoost adds further credence to this claim.

On such a basis, attempting to come up with an ideal classification algorithm for this dataset is far more questionable than it was on the last dataset, owing to the vast potential error from the potentially high variance. However, from the data collected from our model complexity curves, we can at least give an ideal algorithm for the data that we do have available.

On the basis of accuracy, it seems that there is a tie between AdaBoost and KNN. This tie is again resolved in the same manner that the last tie was resolved, by looking at the time complexity of the algorithms. KNN, yet again, runs much faster than AdaBoost, and AdaBoost suffers heavily from certain specific cases of high noise, many of which were encountered with this dataset, so the algorithm that I must recommend for this dataset, with the seemingly limited data that I have available, would be KNN.

Noise and Error: This high noise is not only present in the aviation accidents dataset, but was also encountered in the coding survey dataset, and indeed throughout the analysis, so I was inclined to take a look at just how noisy the data really was. To do so, I began with the arguably simpler case in the coding survey dataset, as the classification I was performing on that dataset was only predicting gender, a binary result in the case of this analysis. I used this coding survey dataset to construct a receiver operating characteristic curve (ROC) with the KNN algorithm, predicting whether or not someone is female:



As you can see above, even with our best performing algorithm, our ratio of true positive rate (TPR) to false positive rate (FPR) was just marginally better than random chance, implying a substantial amount of noise in the dataset. I wanted to further quantify just how much noise there was, so I went ahead and constructed a confusion matrix:

88 True Positives	**346** False Negatives
252 False Positives	**1466** True Negatives

As we can see, there seems to be some major unbalancing of the data set. An algorithm could simply answer no for attributes it receives, just answering someone is male and not female, and still perform reasonably well.

This was a revelation to me, as I finally noticed, perhaps unsurprisingly, that the data, being a survey of prospective coders, was heavily biased towards males. This would cause some heavy error in the algorithms whenever it had to predict a female coder. Therefore, trying to predict gender in this dataset may very well be a fool's errand unless there are somehow some deterministic attributes that reliably can determine gender, which frankly, I find unlikely.

Conclusion: I began this paper with the intention of getting an idea of the strengths and weaknesses of different classification algorithms. I wanted to know what they depended on, what properties of the data they were susceptible to, what strengthened them, etc. I must say, going through such a wide variety of topics in such depth in such a short amount of time not only allowed me to learn all of these properties, but exposed to me some new things as well, in particular the weakness of the two data sets that I had used. Perhaps a change in choice of attributes was in order, or what we were predicting, but it seemed like there was so much noise and interference and bias and whatnot that it was at risk of clouding the entire analysis.

Fortunately, it did not, and while there were definitely some issues, I was still able to glean a number of insights about the data and about the algorithms themselves. In the future I will be able to take all these insights and the potential shortcuts that I've now devised in picking better data sets and better methodology for analysis of algorithms.