Christian Boylston
903355452
CS 4641: Machine Learning
Fall 2018

Assignment 1: Supervised Learning

Introduction:

For this analysis, we'll be looking into several different types of supervised learning algorithms. More specifically, the analysis will cover the performance of Decision Tree, Neural Network, Boosted Decision Tree, K-Nearest Neighbors, and Linear Support Vector Machine models on two binary classification problems. All algorithms were implemented using the sci-kit learn library and graphed based on performance by the matplotlib library. First, it will be important to run some small experiments to determine the appropriate hyperparameter values for each model to get a good performance on the two datasets (all hyperparameter experiments will be done utilizing an 80/20 train/test split). With appropriate hyperparameters in place, it should be possible to compare the performance of all of these algorithms on the test sets and draw reasonable conclusions about the effectiveness of each algorithm in classifying the data.

Datasets Background:

Adult Census Income Dataset: This dataset is a collection of information from the 1996 US Census. The data set contains 32,561 instances that each contain 14 attributes. Each instance represents a person that responded to the 1996 US Census. Attributes include things like education level, age, gender, occupation, and income. The models will utilize all of the attributes other than the income attribute to predict the income for an individual. Considering that this is a binary classification problem, the income is simply a boolean where false (0) represents that the person's income is less than or equal to $50,000/year and true (1) represents that the person's income is over $50,000/year. The data was provided by the fine folks at the UC Irvine Machine Learning repository. Most of the attributes were string variables, so some preprocessing was necessary to convert this part of the data into integer values and later some scaling was necessary to reel in some of the continuous attributes. It's also important to note that the data is a skewed with 75% of the instances making less than $50,000/year. This dataset is rather popular on the UCI machine learning repository for good reason. It offers several tens of thousands of data instances that offer real-life information about the anatomy of American income. Leaving many interesting questions for examination. What groups of people are most likely to make over $50,000/year? How much does education affect income? So many of these questions are worth investigating and can hopefully shed some light on understanding socio-economic standing.

E-mail Phishing: Considering how large the adult dataset is, it seemed like an interesting endeavor to look at a smaller dataset and see how well these models classify on significantly smaller datasets. This data set consists of 1250 instances, where each instance is an email that will be marked as a phishing email (0) or a legitimate email (1). Each instance has 10 attributes including the attribute that states whether it is a phishing email or not. Each attribute is represented by neither a -1, 0, or 1 representing false, unsure, or true. Some attributes include web traffic, IP address, and URL length. This dataset is also taken from the UCI machine learning repository, but was slightly modified to remove the negligible number of cases where the email was labeled as "suspicious" rather than simply phishy or legitimate. This data is pretty evenly split and uses categorical variables, so it won't really need as much preprocessing. This dataset also provides a classification more predictable than something that is more prone to variation like someone's income. As such, it should be expected that the classifying algorithms are more accurate at classifying phishy emails than personal income. Nonetheless, this is in fact an interesting problem that can be utilized in software to detect phishing emails. It could provide key insights into understanding what are the tell-tale signs of something "phishy."

Decision Trees:

Decision trees constructed on these on two datasets utilized a limitation on the maximum depth as the chosen means of pruning along with the GINI index as a means of splitting attributes. This is a form of pre-pruning. By limiting the maximum distance from the root node of the decision tree to any given leaf node, the tree isn't able to perfectly fit itself to the training data. With the maximum depth hyperparameter adjusted optimally, the decision tree can better generalize to novel test data since it has been prevented from overfitting the training data. To figure out the best maximum depth value in the actual code for the completed model, the decision tree was run with various values for the maximum depth and these were plotted against some kind of performance metric. The performance metric utilized for the census dataset was its f1 score. The f1 score was chosen because the data is skewed with three times as many people making $50,000/year or less than over $50,000/year and accuracy wouldn't necessarily reflect its ability to correctly classify members of each income group, as simply classifying most every instance as making $50,000/year or under would likely perform pretty well in terms of accuracy on the test and training, but it would generalize terribly when given future instances where the person makes over $50,000/year. The phishing dataset is fit to use accuracy as its performance metric because it's data is relatively balanced requiring accurate classifications of both phishing and legitimate emails to get a good overall accuracy score.In Figures 1 and 2, the results of varying the hyperparameter (maximum depth) for the census dataset and the phishing dataset. It should be noted that for the adult income (census) data set that the graphs should in fact be labeled "Test f1 score" and "Training f1 score."

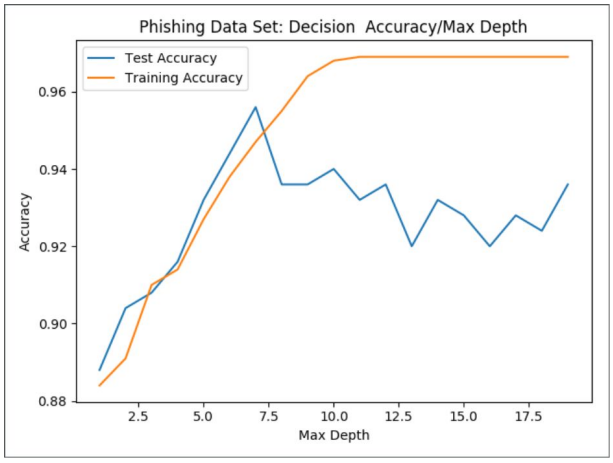


Figure 1: Adult Income Dataset Max Depth **Figure 2: Phishing Dataset Max Depth**

Looking at Figure 1, it seems that the optimal maximum depth for the decision tree model trained on the adult income dataset is nine as this is when the model begins to perform better and better on the training data, but worse on the test data indicating some degree of overfitting. Similarly, this seems to indicate the the best choice of maximum depth for the decision tree model trained on the phishing data is about six as after six the model begins to succumb to overfitting.

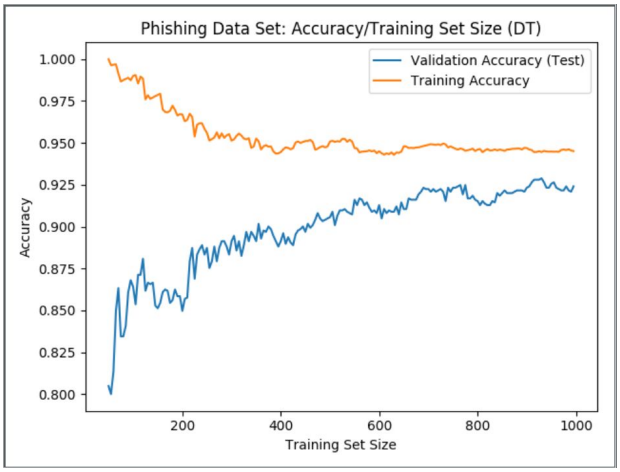


Figure 3: Adult Income Dataset Learning Curve **Figure 4: Phishing Dataset Learning Curve**

Featured above in Figure 3 is the learning curve for the Adult Income dataset model utilizing the Decision Tree (DT) model with the established best maximum depth hyperparameter. This graph measures the f1 score of the model as a function of the training set size. The validation curve was produced by taking 5-fold cross-validation test results and averaging them, while the training curve utilized the average of the training results on this cross-validation. It can be seen in Figure 3 that the model generalizes quite well when the training size is over about 8,000 and only increases slowly and steadily afterwards. The f1 score

for the DT model on the Adult data set appears to top out around 0.65 with sufficient training data, while the accuracy was measured be on average about 0.85. When looking at a learning curve for Figure 4 utilizing the same technique as was explained for Figure 3, it can be seen that around a training size of 700 and above, the model begins to generalize pretty well and starts to classify consistently with around 0.92 accuracy. It's also of interest to note that in both datasets the performance for both the test and training increases at a similar rate as the number of training examples increases except when the training set size is small and the training error is increasing while the testing error is decreasing.

The decision trees were very quick to train and would return a model that would work with a relatively high degree of performance on both datasets. Considering that the decisions trees perform well and train very quickly, they seem like a good starting place for getting a high performance model.

Neural Networks:

When looking at the performance of neural network models, there are two hyperparameters that must be adjusted. The first is the number of neurons in each layer and the second is the number of hidden layers. The graph of the performance of the adult income dataset based on the number of neurons can be seen in Figure 1 while the graph of the performance of the adult income dataset based on number of hidden layers can be seen in Figure 2. Analogously, Figure 3 and Figure 4 shows the variation of these hyperparameters based upon the phishing data set.

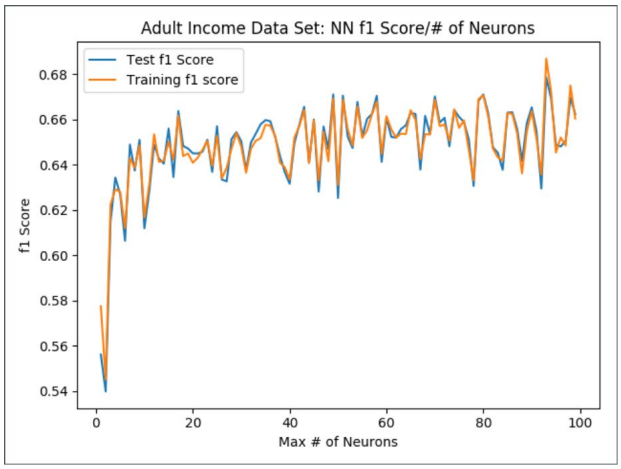


Figure 1: Adult Dataset f1 score/# of Neurons

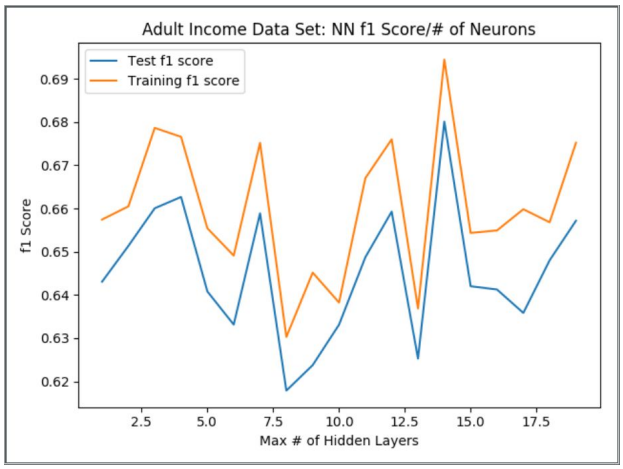


Figure 2: Adult Dataset f1 score/# of Hidden Layers

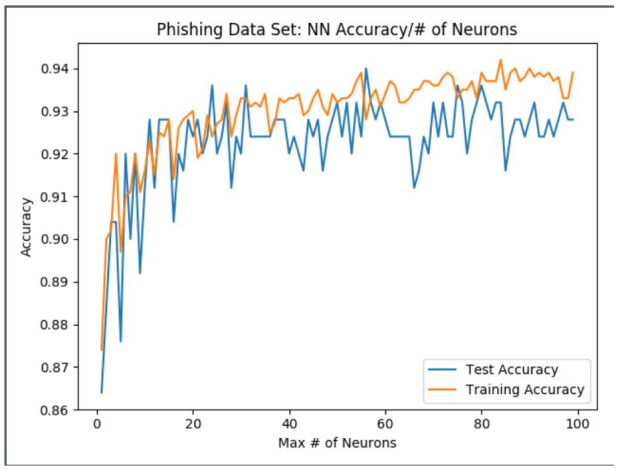


Figure 3: Phishing Dataset Accuracy/Neurons Layers

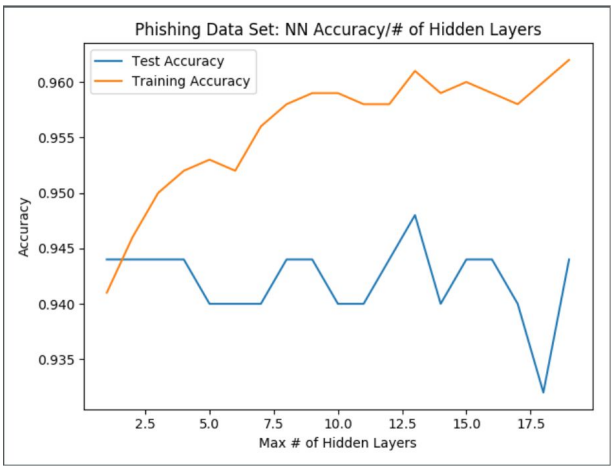


Figure 4: Phishing Dataset Accuracy/Hidden

As it appears in Figure 1, it seems that 95 represents a good estimate for the number of neurons in each layer to maximize the f1 score for the neural net representing the Adult Income data set. Figure 2's depiction of the f1 score as a function of the number of hidden layers for the Adult Income data set does not lend itself to as simple conclusions as the data from Figure 1. The function oscillates up and down with a good bit of irregularity over its entire domain. This seems to suggest that tweaking the number of hidden layers wasn't an incredibly reliable way of increasing f1 score performance of the model. So, the number of hidden layers chosen was two simply because during the experiment it performed comparable to any other number of hidden layers and significantly decreased training time when compared to networks with a greater number of hidden layers. Figure 3 displays that once the number of neurons got over about 30 for the Phishing data set model, the accuracy remained pretty high with some minor variation, so just going from maximum accuracy on the test set, 55 neurons per layer were chosen for the model. Figure 4 shows that the number of hidden layers did very little to change the performance of the phishing data set model, so choosing a small number of hidden layers (two) seemed most prudent to get a speedier training time and a model that performed well.

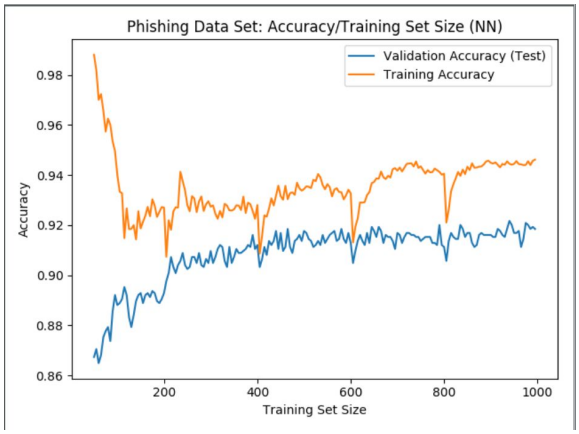


Figure 5: Adult Income Dataset Learning Curve **Figure 6: Phishing Dataset Learning Curve**

Featured above in Figure 5 is the learning curve for the Adult Income dataset model utilizing the Neural Network (NN) model with the established best number of neurons/layer and number of hidden layers. This graph measures the f1 score of the model as a function of the the training set size. The validation curve was produced by taking 5-fold cross validation test results and averaging them, while the training curve utilized the average of the training results on this cross-validation. It can be seen in Figure 5 that the model generalizes quite well when the training size is over about 8,000 and only marginally increases thereafter. The f1 score for the NN model on the Adult data set appears to top out around 0.66 with sufficient training data, while the accuracy was measured be on average about 0.85. When looking at a learning curve for Figure 6 utilizing the same technique as was explained for Figure 5, it can be seen that around a training size of 300 and above, the model begins to generalize pretty well and starts to classify consistently with around 0.91 accuracy. It's also of interest to note that in both datasets the performance for both the test and training increases at a similar rate as the number of training examples increases once the model begins to generalize slightly better after 100 training examples.

The neural networks were one of the slowest models to train. They performed slightly better than the SVM with the help of some scaling and normalization, but were still rather slow to train and required some of the greatest amount of experimentation to properly adjust the relevant hyperparameters.

Boosted Decision Trees:

Much like the regular decisions trees, setting the maximum depth is going to be the chosen form of pruning and will also be the hyperparameter to be tweaked by the model. The boosted decision tree also has one more hyperparameter known as the number of estimators. For the sake of brevity, experiments on this hyperparameter are excluded as altering the number of estimators greater the default number did little to increase performance on either data set. Thus, the default value of 50 will be used for both datasets. Looking below at Figures 1 and 2, it can be seen that the Adult Income dataset test performance begins to peak around a maximum depth of 9 and then begins overtraining on the training set thereafter, while the phishing data set seems to perform best when the maximum depth is set at five.

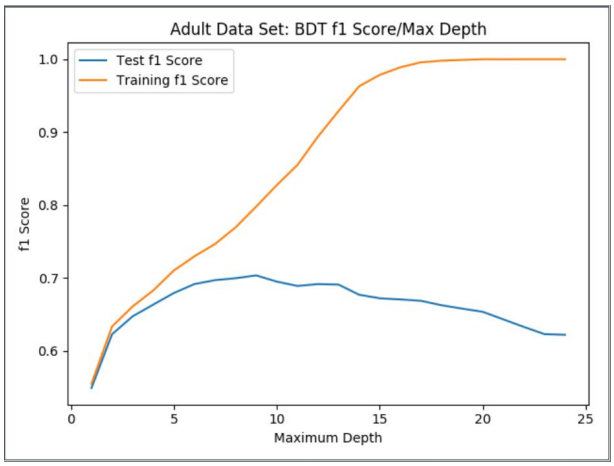


Figure 1: Adult Dataset f1 Score/Max Depth

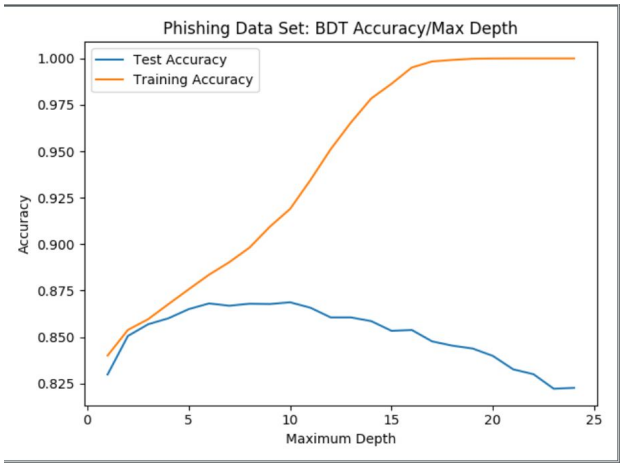


Figure 2: Phishing Dataset Accuracy/Max Depth

Figure 3: Adult Income Dataset Learning Curve

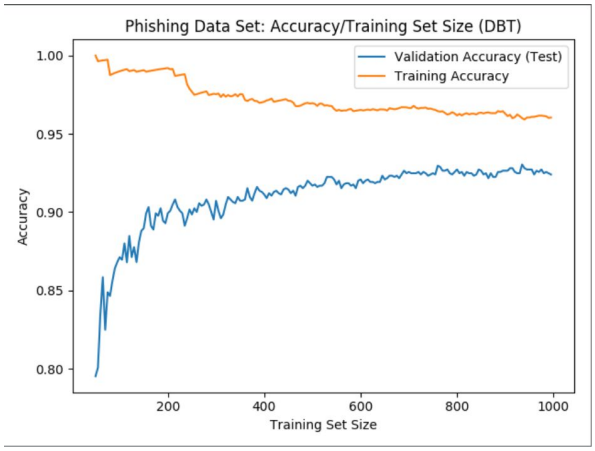


Figure 4: Phishing Dataset Learning Curve

Featured above in Figure 3 is the learning curve for the Adult Income dataset model utilizing the Boosted Decision Tree (BDT) model with the established best maximum depth. This graph measures the f1 score of the model as a function of the the training set size. The validation curve was produced by taking 5-fold cross validation test results and averaging them, while the training curve utilized the average of the training results on this cross-validation. It can be seen in Figure 3 that the model generalizes quite well when the training size is over about 5,000 and only increases slowly and steadily thereafter. The f1 score for the BDT model on the Adult data set appears to top out around 0.70 with sufficient training data, while the accuracy was measured be on average about 0.87. When looking at a learning curve for Figure 4 utilizing the same technique as was explained for Figure 3, it can be seen that around a training size of 300 and above, the model begins to generalize pretty well and starts to classify consistently with around 0.93 accuracy thereafter. It's also of interest to note that in both datasets the training accuracy decreases and the testing accuracy increases as the as the training set size increases. This shows

that the model is beginning to generalize better with more data by utilizing boosting and max depth to classify better and combat overfitting the training data.

It should be noted that the BDT's train relatively fast on the training data and perform the best in terms of f1 score and classification accuracy. Thus, far it appears to be the best method for a classification model.

K-Nearest Neighbors:

For the K-Nearest Neighbors Algorithm, it was important to find the appropriate value for the hyperparameter "k" in order to get the best performance out of the K-Nearest Neighbors model. Looking at Figure 1 for the Adult Income dataset where the f1 score is graphed as a function of the number of neighbors, it appears that 13 is the number of neighbors that maximizes the f1 score, so this is the value that will be used for the Adult income dataset model. Now, looking at Figure 2 for maximizing accuracy as a function of the number of neighbors, it appears that five is the best choice for the number of neighbors, as after five both training and test accuracy begin to drop precipitously as a result of the oversaturation of more and more distant neighbors increases.

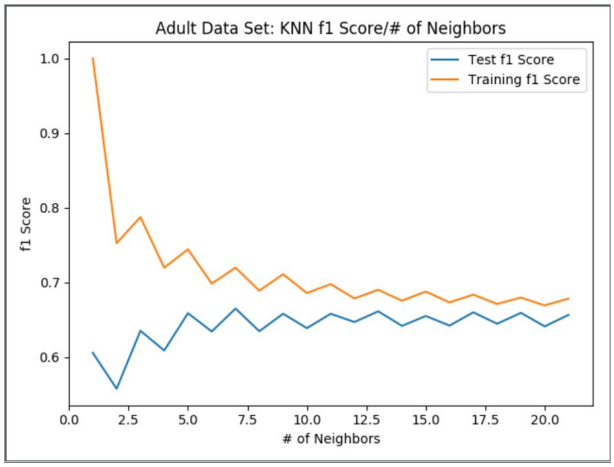


Figure 1: Adult Dataset f1 Score/# of Neighbors(k)

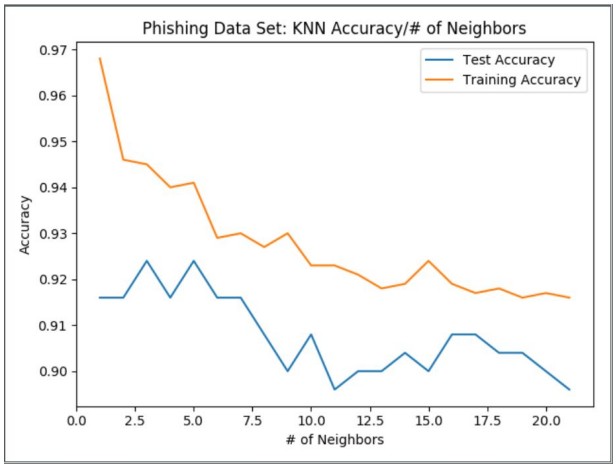
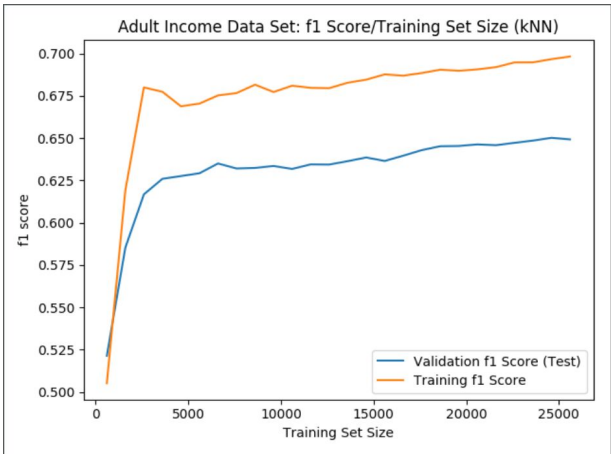


Figure 2: Phishing Dataset Accuracy/# of Neighbors(k)



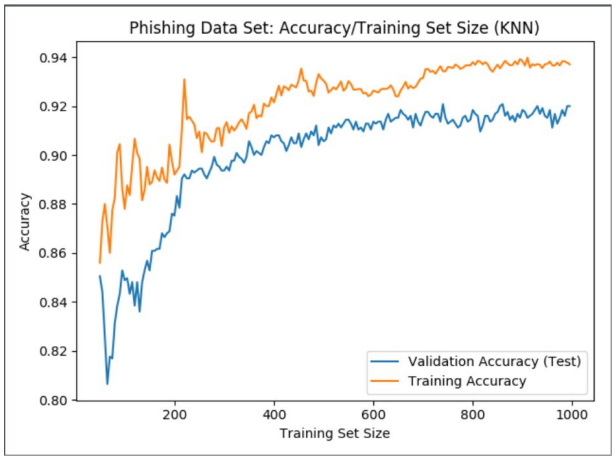


Figure 3: Adult Income Dataset Learning Curve **Figure 4: Phishing Dataset Learning Curve**

Featured above in Figure 1 is the learning curve for the Adult Income dataset model utilizing the kNN neighbors model with the established best k-value. This graph measures the f1 score of the model as a function of the the training set size. The validation curve was produced by taking 5-fold cross validation test results and averaging them, while the training curve utilized the average of the training results on this cross-validation. It can be seen in Figure 1 that the model generalizes quite well when the training size is over about 7,500 and only marginally increases thereafter. The f1 score for the kNN model on the Adult data set appears to top out around 0.64 with sufficient training data, while the accuracy was measured be on average about 0.84. When looking at a learning curve for Figure 4 utilizing the same technique as was explained for Figure 3, it can be seen that around a training size of 500 and above, the model begins to generalize pretty well and starts to classify consistently with around 0.91 Accuracy. It's also of interest to note that in both datasets the performance for both the test and training increases at a similar rate as the number of training examples increases. This is dissimilar from some models where the test error increases as the training error decreases.

Support Vector Machines:

A linear SVM was implemented for this experiment, which left only the value of C and gamma as possible hyperparameters. Experimenting with different values of C and gamma did nothing to improve the performance of the model for both datasets. As such it seems wasteful to space including graphs displaying their triviality in this circumstance. One issue that was encountered with the Adult Income dataset was that the data was skewed towards incomes less than $50,000/year and this caused the SVM to be biased towards labeling the vast majority of instances as making less than or equal to $50,000/year. Considering that three quarters of the instances did in fact fall into this category, the accuracy of the model wasn't terrible, but it did a horrendous job at properly labeling people that actually made more than $50,000/year. To counteract this, a new hyperparameter was selected, which was the use class weights. Using the weights to add more precedence to people that made more than $50,000/year, helped to calibrate the results to a more respectable number. The weights that produced the best results were weighing people making over $50,000/year twice as heavily as those making equal to or less than that amount. No such weights were need for the phishing data set as it was quite well balanced already.

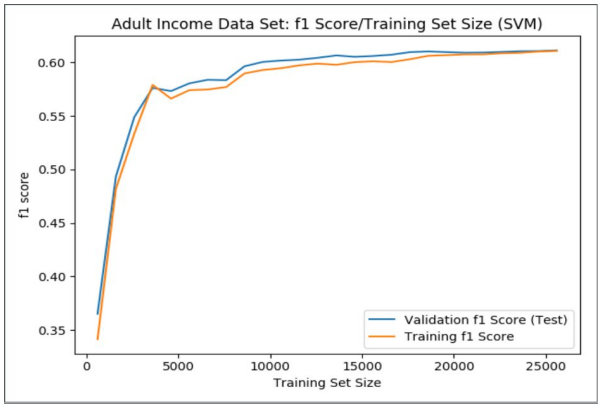


Figure 1: Adult Income Dataset Learning Curve

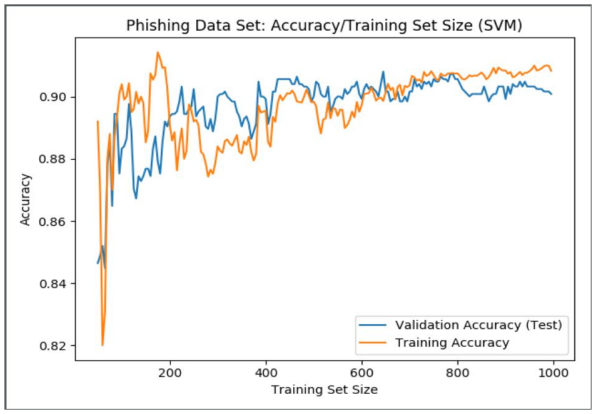


Figure 2: Phishing Dataset Learning Curve

Featured above in Figure 1 is the learning curve for the Adult Income dataset model utilizing the linear kernel SVM model with the established best class weights. This graph measures the f1 score of the model as a function of the the training set size. The validation curve was produced by taking 5-fold cross validation test results and averaging them, while the training curve utilized the average of the training results on this cross-validation. It can be seen in Figure 1 that the model generalizes quite well when the training size is over about 5,000 and only takes another leap in performance after getting a training size of about 10,000. After this leap, the increase in performance is only marginal. The f1 score for the linear kernel SVM model on the Adult data set appears to top out and remain around 0.61 when sufficient training data has been provided, while the accuracy was measured be on average about 0.82. When looking at the learning curve for Figure 2 utilizing the same technique as was explained for Figure 1, it can be seen that around a training size of 200 and above, the model begins to generalize pretty well and starts to classify consistently with around 0.90 accuracy from there on. This definitely marks a shift from some other the models utilized on the phishing dataset, as it appears that the linear SVM can generalize better on less training data than some of the other models. Although, the Adult dataset doesn't seem to be able to display this same ability to be able to generalize on a smaller training set. This may be the result of the datasets themselves. Phishing emails like have some features that very much correlate with the fact that they aren't legitimate. Given the information about the email, a person would likely be able to accurately classify if an email was legitimate or not, but given some information about a person, it might be more difficult to correctly classify whether they make above $50,000/year or not because income is far more complex than spam/phishy email identification. As such it appears, that classification problems where the values of certain features map pretty consistently to a certain outcome are more suitable for classification with a separating hyperplane.

Another thing to note about the SVM model was that it was incredibly slow to train. Training the Adult Income dataset on the SVM model took an incredibly long time without doing a lot of preprocessing on the dataset. It took almost 24 hours to train without scaling the data minimums and maximums along with normalizing the data. After this was done and some

dimensionality reduction, the SVM performed better and faster, but was still the slowest model to train.

Analysis:

Overall, it seems that all models were not created equal. The Boosted Decision Tree was able to consistently outperform the other models on both datasets. Out of the box, the Gradient Boosted Decision Tree was able to consistently perform better than the other models that had been tuned with the appropriate hyperparameter values. It seems that much of this success can be attributed to the practice of boosting, which made it possible for the model to learn intricate, non-linear decision boundaries with very little need for any formal preprocessing. Adjusting the maximum depth hyperparameter was able to slightly improve results, but wasn't really necessary for getting good performance from the model. Achieving an f1-score of 0.7 and a classification rate of 87% is rather impressive for such a complex problem as the one posed by the Adult income dataset.

On the other hand, the SVM performed the worst of all the models on the data set achieving a classification rate of 83% on the Adult Income dataset with an f1 score of 0.61 and a classification rate of 90% on the Adult Income dataset. This performance speaks more likely to user error than a weakness in the model. The SVM requires a significant amount of preprocessing in order to perform optimally. Without scaling in outliers and normalizing the data, the SVM succumbs to the curse of dimensionality rather quickly. Despite much experimentation with how to process the data, the best preprocessing procedure was clearly not used as evidenced by the superior performance of more naive classifiers. The SVM would likely be able to perform better using a non-linear kernel that isn't subject to the same issues as the linear kernel. The SVM model also was very prone towards bias leading often to a significant number of misclassifications of instances that were labeled as making more than $50,000k/year. The linear SVM is a powerful model, but requires a significant understanding of your dataset to be able to perform optimally, which makes it in many ways less advantageous to other models that perform well with little extra work.

All of the other models fall somewhere in between these two models in terms of performance. The Decision Trees were able to classify the Adult Income dataset with an accuracy of 85% and an f1 score hovering around 0.65, while it accurately classified the phishing data 92% of the time. Despite not being as accurate as the boosted version, the decision trees performed quite well and with a good bit of speed. It is possible that maybe with the utilization of a less naive pruning procedure, better results could be achieved. It seems that much like the boosted decision trees, the decision tree model can perform rather well on many datasets with little need for extensive preprocessing.

Neural nets represented a similar issue to SVM's. Despite their power as a model, they underperformed when compared to much more naive models. Their performance on both models was slightly less impressive than the decision tree models. This discrepancy between the power

of the model and the performance of the model likely boils down to the datasets selected. Neural networks can be prone to overfitting on smaller data sets and neither of these open source datasets even exceeded 50,000 entries. The neural network model would likely be able to perform better on significantly larger datasets where it could use more training examples to reach greater generalization.

Finally, there's K-Nearest Neighbors (kNN) model, which was right in the middle of the pack in terms of performance. For the Adult dataset, it was able to classify accurately 84% of the time and achieved an f1 score of about 0.64, while for the phishing dataset it achieved a classification accuracy of around 92%. K-Nearest Neighbors struggles with some of the same dimensionality issues as the linear SVM. It seems that since the data sets weren't incredibly massive and each had less than 20 attributes, kNN was able to perform relatively well and even better than some more sophisticated models of classification. kNN might have been able to outperform the linear SVM as a result of the training data not being linear for the adult dataset. It must be noted that kNN was also very slow to train, but it's performance was greatly improved through normalization and some outlier scaling. It is likely that on a larger dataset or a dataset with a greater number of features kNN would experience significant performance drops in terms of speed likely rendering it useless for most practical purposes on exceptionally large datasets.

Conclusion:

Overall, one of the main takeaways from these experiments is that different data sets are going to require different models. For example, higher dimensional data would likely show some of the performance errors in SVM and KNN models that weren't readily apparent on the chosen dataset. Likewise, larger datasets may have been best classified by neural networks. As it stands though on these two data sets, the boosted decision tree was the best model be all metrics on both datasets.

For future experiments it might be more interesting to experiment with datasets that are rather different from these two like a dataset that has a massive number of variables for instance or just a simply bigger datasets of the same kind as those selected. In terms of the datasets selected for classification, the Adult Income dataset proved to be a far more interesting problem to classify as there weren't as many features that were as heavily weighted for classification as in the phishing email dataset. There's far more noise and variation in someone's income bracket classification based on certain features than in simply classifying if a given email is spam. If this experiment were to be completed on different datasets, it would be an even more interesting exercise to look at larger, noisier data sets and see which algorithm is able to take the cake, but for now Gradient Boosted Decision Trees take the cake.

References:

Becker B. and Kohav R. (1996). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.

Abdelhamid N. (2016). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.