An Analysis of Supervised Learning Algorithms on Classification Problems

Abhinav Tirath

I. Dataset Description

In this study, we compare and contrast the results of 5 different supervised learning algorithms: Decision Trees, Neural Networks, Boosted Decision Trees, Support Vector Machines, and k-Nearest Neighbors. To accomplish this, each algorithm was implemented on two datasets.

A. Credit Card Fraud Detection

The first dataset used describes 10,000 separate credit card transactions, classified into two categories: fraudulent (represented by 1) and not fraudulent (represented by 0). The data is highly unbalanced; only 493 of the transactions are classified as fraudulent, as indicated in Figure 1.

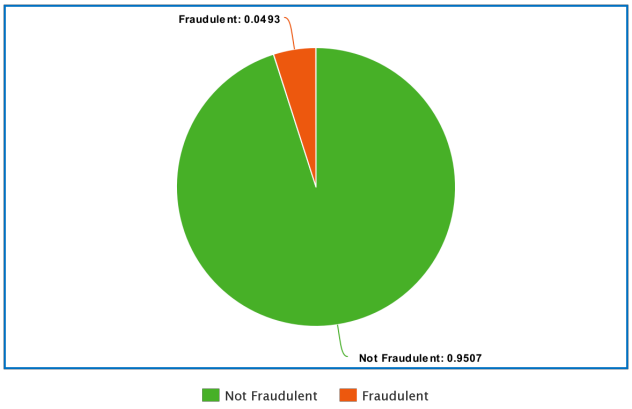


Figure 1. Displays the balance of the credit card data.

The data were collected and analyzed by a research collaboration of Worldline and the Machine Learning Group of ULB. This dataset has 28 features that describe each transaction, and no cell is blank. The number of instances in the dataset was reduced from about 243,000 to 10,000 in order to make the data more balanced and reduce time of computation.

The features are all numerical inputs labelled V1, V2,… V28, and they are the result of a PCA transformation. Unfortunately, the original features and a further description of the data cannot be provided, due to confidentiality issues.

Credit and debit card transactions account for over 50% of all transactions in today's world. For this reason, it is essential that payments be assessed as fraudulent or not fraudulent as soon as possible. If we fail to do so, customers will be charged for purchases that they did not make, and credit and debit card companies will fall to the wayside with lawsuits and chaos. A solution to this problem will ensure the safety of consumer money and credit card use.

Since the dataset is quite unbalanced, most of the algorithms achieved greater than 95% overall accuracy in their predictions. However, the algorithms were not quite as successful in the recall, which describes the percentage of the time that at predicting fraudulent transactions as fraudulent (algorithms quite often classified them as not fraudulent). This is also the most important metric; above all else, we want the fraudulent transactions to be predicted as fraudulent. So, we will primarily use this metric to compare and contrast the results of the algorithms for the credit card fraud dataset.

B. Pima Indians Diabetes

The second dataset used describes diagnostic measurements of 769 Pima Indian females, classified into two categories: has diabetes and does not have diabetes. The data is somewhat imbalanced; 501 of the females did not have diabetes, as indicated in Figure 2.

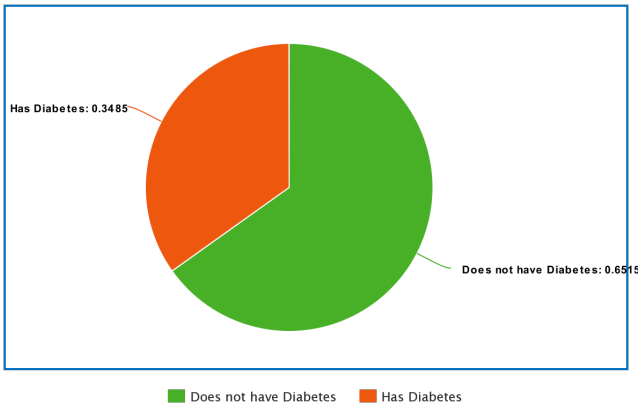


Figure 2. Displays the balance of the Pima Indian Diabetes data.

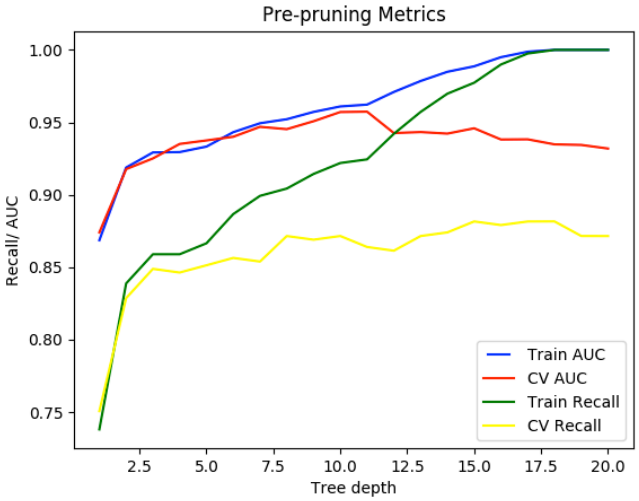
The eight features consist of age, BMI, number of pregnancies, and five diagnostic measurements, such as blood pressure. The data was provided by National Institute of Diabetes and Digestive and Kidney Diseases, but some values were left blank for some patients. These values were filled with the mean of the remaining measurements of the same feature. This was the only adjustment made to the data.

The objective of our analysis of this data is to predict whether a given person has diabetes or not, based on a few relatively simple diagnostic measures. The results can help physicians easily determine whether someone is likely to be diabetic. This has far-reaching implications. Not only will diabetic testing become much cheaper and faster, doctors may even be able to determine if someone has diabetes simply from a physical.

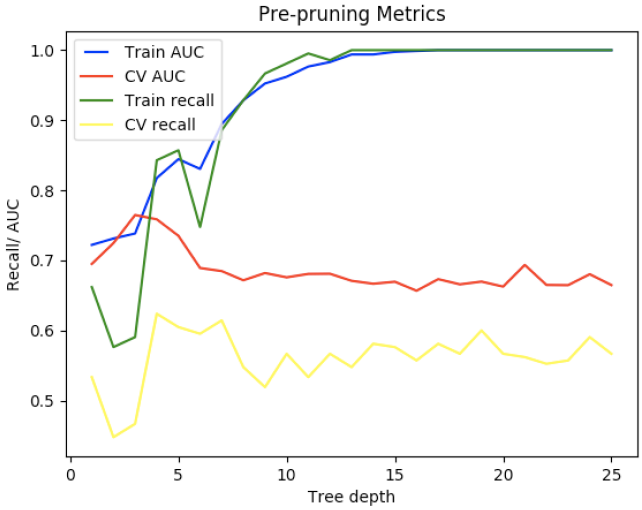
II. Decision Trees

With decision trees, we first prepruned the maximum depth of the tree using cross validation, fit the tree to the data with the calculated maximum depth, and postpruned the tree by removing the children of nodes with class counts of less than some threshold.

Figure 3. These metrics display recall and Area Under Curve (AUC) versus the maximum depth of the tree in the process of prepruning. AUC is a common metric used in binary classification to assess accuracy. A higher AUC is preferred.



a. Credit Card Data

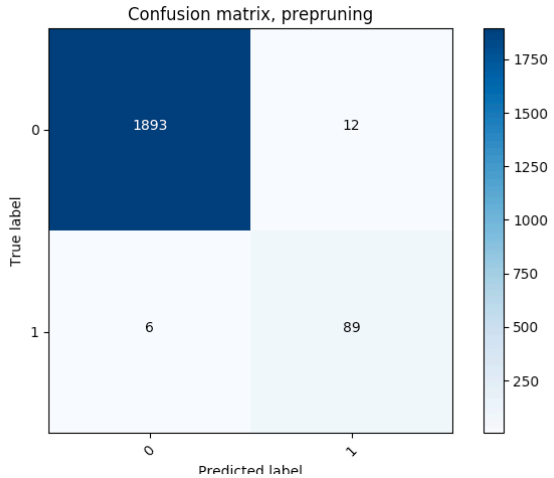


b. Pima Indians Diabetes Data

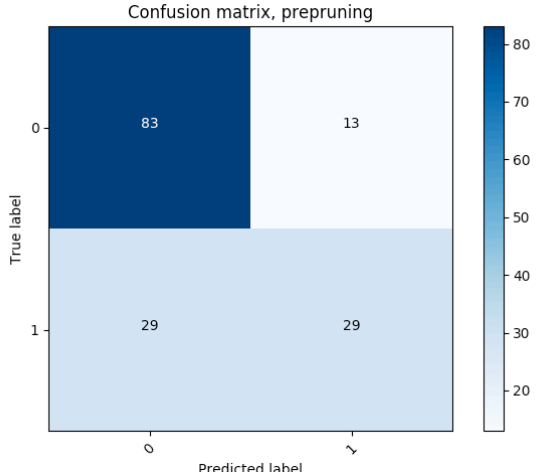
In both diagrams of Figure 3, Train AUC and recall converge to 1 when their maximum depth is allowed to be high enough. This is a case of overfitting; the data will perform best on the data with which it trained. As such, we see that recall and AUC of the cross-validation set do not correspond or correlate with the recall and AUC from the training set. We know that our cross-validation data gives more insight into our results on our test data. For the credit card data, we are trying to maximize recall, so a maximum depth of 18 will be chosen, according to figure 3a. Since we are trying to maximize accuracy for Pima dataset, we take the maximum of the cross-validation AUC function, which, according to Figure 3b, is 3. It is also apparent that, in both

curves, there is not much variance after a certain depth. This indicates that only minor adjustments in classifications are being made, and we would likely get similar results for any of these values as maximum depth. Note that the cross-validation recall and AUC of the credit card data is much higher than those of the diabetes data.

Figure 4. Confusion Matrices indicate the true values of your test data and your predictions for them.



a. Credit Card Data

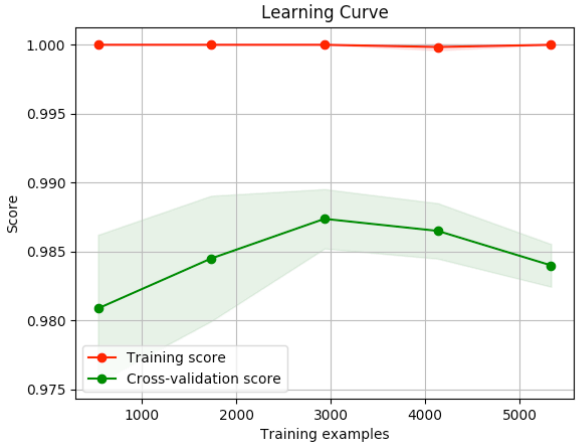


b. Pima Indians Diabetes Data

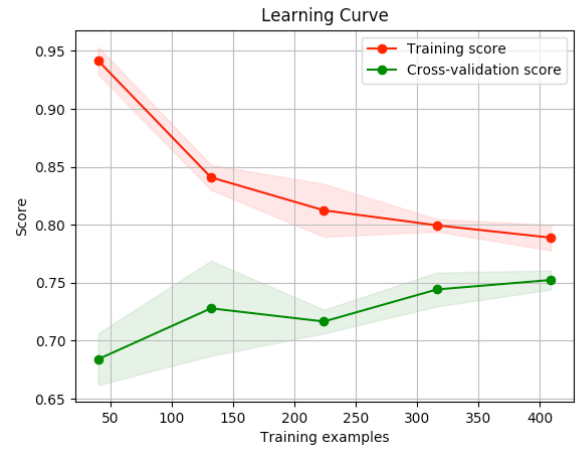
Here, we can see the results on our test data after we have trained our algorithm. Figure 4a shows that the algorithm classifies non-fraudulent instances correctly with high frequency, but classifies fraudulent instances correctly at a significantly lower rate, as expected. This is partially attributable to the fact that the dataset is unbalanced. In Figure 4b, we see that the recall of 0 is relatively high. The recall of 1, however, is quite low at 50%. When comparing Figure 3b with Figure 4b, there are clearly higher rates of CV recall than 50%. We forewent a high recall to get the highest possible AUC.

These results are neither the worst, nor the best in comparison to other algorithms. One advantage for decision trees with these datasets is that they work well with little input data, and our diabetes dataset has only 779 total instances. One disadvantage to decision trees in our case is that they are unstable, meaning that small amounts of data can greatly affect the accuracy of a decision tree.

Figure 5. Learning curves show the accuracy of your dataset vs. the number of instances used to train it.



a. Credit Card Data



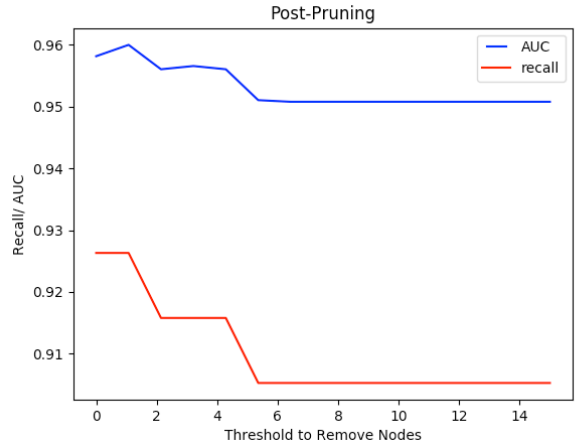
a. Pima Indians Diabetes Data

In Figure 5a, we see that both the training score and the cross-validation score are very high. Once again, this result is partially due to the imbalance in the data. The reason that cross-validation curve begins to dip is because the algorithm begins to overfit the data. However, the change in accuracy

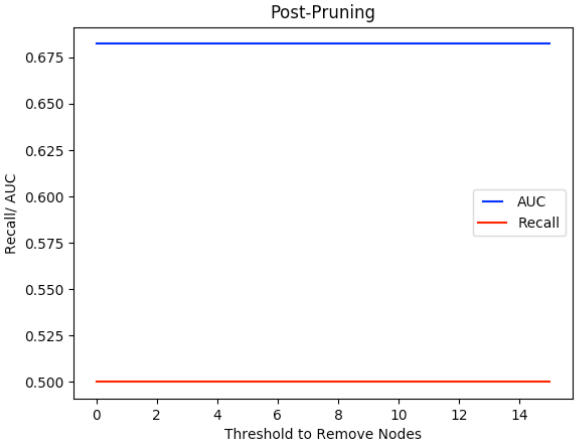
is very minor. The distance between the curves and the training score indicate overfitting and high variance.

In Figure 4b, the training curve looks as it should; the cross-validation rises with the number of training examples, and the training score decreases, indicating low bias and variance.

Figure 6. The plots indicate the AUC and recall against the threshold to remove nodes from the tree, as described in the introduction to this section.



a. Credit Card Data



b. Pima Indians Diabetes Data

In Figure 6a, we see that the highest AUC value is yielded by a threshold of 0 or 1. The algorithm arbitrarily chooses 0, so no nodes are removed. Similarly, the algorithm will not remove any nodes for the diabetes data. This is due to the fact that both the AUC and recall curves are completely flat in Figure 6b, likely due to the fact that the maximum depth of the decision tree is only 3, according to Figure 3b. As a result, postpruning did not make a difference to either decision tree.

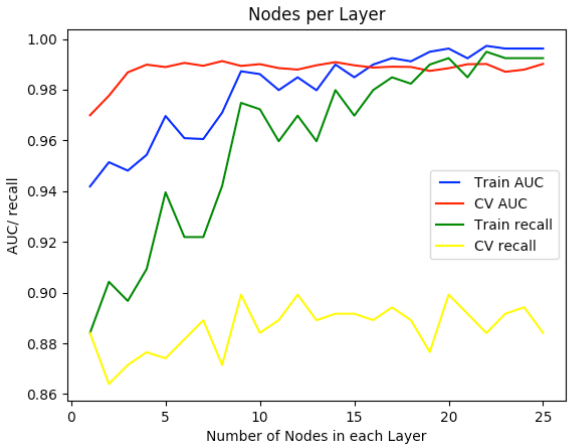
Moreover, the most important features in the decision tree algorithm for the credit card data is V17, followed by V14 and V13. For the diabetes dataset, only three features were used to split: Plasma Glucose Concentration, BMI, and Age, in decreasing level of importance.

Overall, the decision tree implementation could use some work. Its predictions based on its confusion matrices were about average, but the runtime was very short in comparison to the other algorithms. The implementation could be improved in two ways: by making it less susceptible to overfitting and by prepruning a different hyperparameter. Since we prepruned maximum depth, our postpruning made no difference, as it also prunes depth.

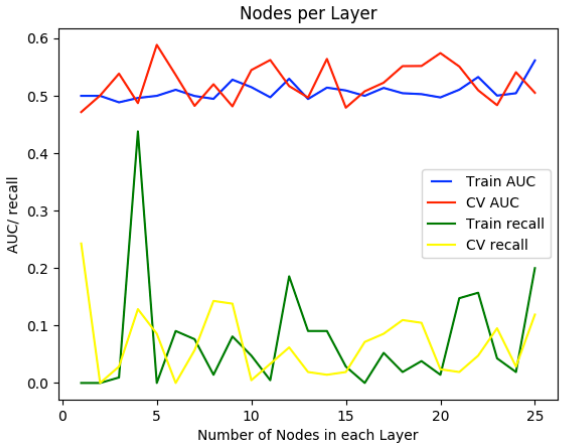
III. Neural Networks

Initially, the ideal number of layers were tested independently, and it was found that the credit card data worked best with two layers, and the diabetes data worked best with one layer. Then, cross-validation was used to tune our determine the ideal number nodes in each layer. We then train the neural net with ideal hyperparameter and obtain our results.

Figure 8. Recall and AUC vs. the Number of Nodes in each layer



a. Credit Card Data

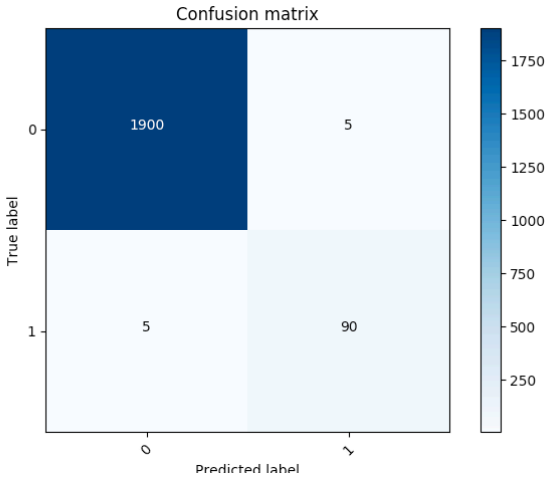


b. Pima Indians Diabetes Data

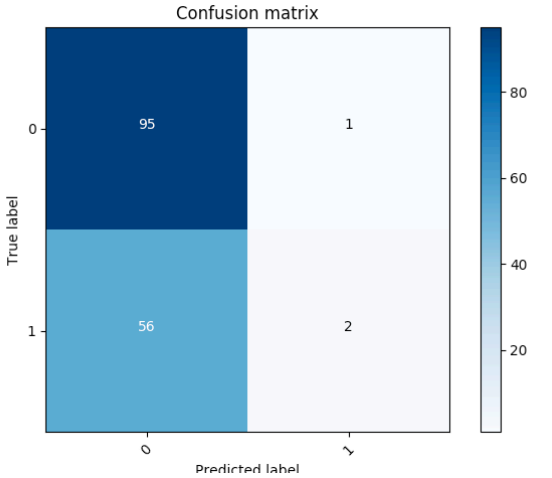
In Figure 7a, we can see that both our AUC and our recall for our cross-validation set are maximized at around 20 nodes per layer. We also observe that, after nine nodes per layer, neither the CV AUC nor the CV recall vary much; however, the Train AUC and recall consistently increase, as a result of fitting the data even more.

In Figure 7b, we observe that each curve is quite erratic, and never consistent. This indicates that it will be difficult to determine which number of nodes per layer will realistically perform the best on the testing data. However, we do see that the recall rate in both the training and cross-validation sets are generally much lower than their respective AUCs. This indicates that the model suffers from high bias, and it is classifying the majority of instances as 0. Still, the algorithm will choose the number of nodes with the highest cross-validation AUC, regardless of the cross-validation recall value.

Figure 8. Confusion Matrices for the Neural Networks.



a. Credit Card Data



b. Pima Indians Diabetes Data

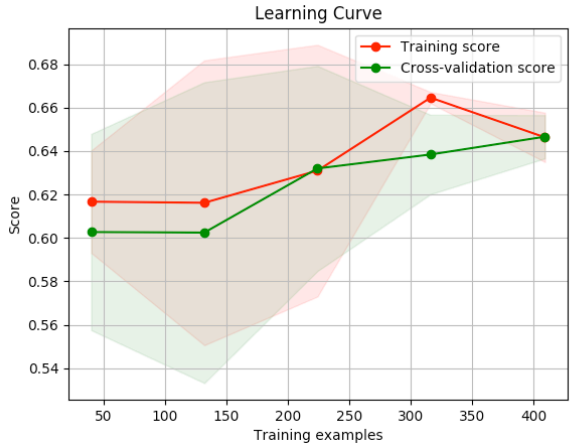
From Figure 8a, the neural network has done an excellent job in classifying whether credit card transactions are fraudulent or not (only missing a total of 10 values), likely due to its high fault tolerance. According to our recall metric, neural networks performed slightly better than decision trees.

From Figure 8b, we see that neural network does very poorly, and only predicts a value of 1 3/154 instances. This is due to the imbalance of the data; the network has essentially determined that the best way to maximize the AUC is to guess 0 for nearly every instance. As a result, the recall score is very low, and the network has high bias.

One reason for the disparity in the confusion matrices is that neural networks work much better with larger datasets than with smaller datasets.

Figure 9. Learning curves for Neural Networks.

a. Credit Card Data



a. Pima Indians Diabetes Data

In Figure 9a, the learning curve is about what we expect it to be. However, the slight distance between the curve indicates slight overfitting and possibly high bias.

In Figure 9b, we find the opposite. The erratic behavior of the training score curve and the relatively low value of both curves indicate underfitting and high variance.

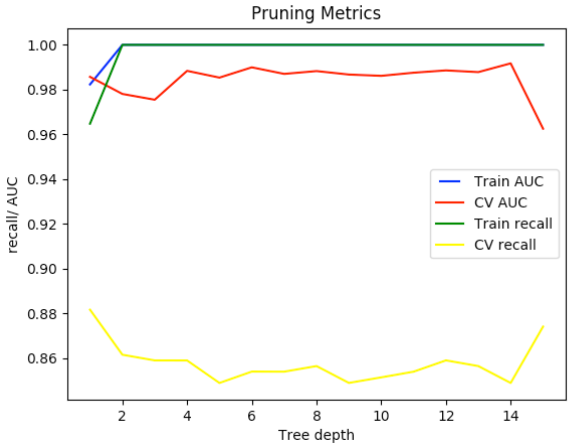
Initially, my neural network algorithm tested how many layers were ideal before how many nodes to put in each layer, but the results were extremely erratic and inaccurate. Instead, we determined the best number of layers by running a series of tests and choosing the best results.

In redoing this, we would add another hyperparameter to tune, such as the solver, to increase the overall performance of the neural network. We would also not rely on just the AUC score for the diabetes dataset. Rather, we would have some combination of the AUC and recall scores, so all the predictions would not be 0.

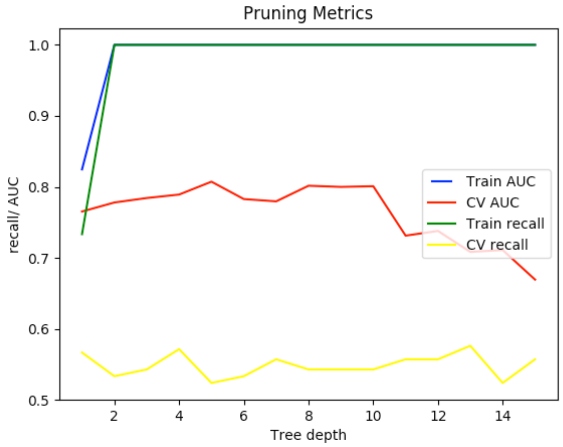
IV. Boosted Decision Trees

For the boosted decision tree, we pass in a DecisionTreeClassifier and preprune its maximum depth, in a similar prepruning process as we did for our decision tree. Then, we train the boosted tree with our ideal maximum depth and obtain the results.

Figure 10. Recall and AUC vs. the Maximum Depth of the DecisionTreeClassifier



a. Credit Card Data



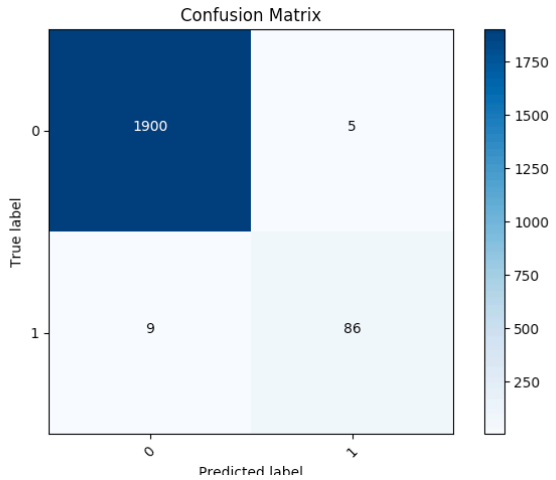
b. Pima Indians Diabetes Data

In both Figures 10a and 10b, after a maximum tree depth of just two, both the training AUC and recall are 1. This shows how little information decision trees fit to the data. We also see that there is a sharp decline in the cross-validation AUC in both graphs, indicating that overfitting has occurred.

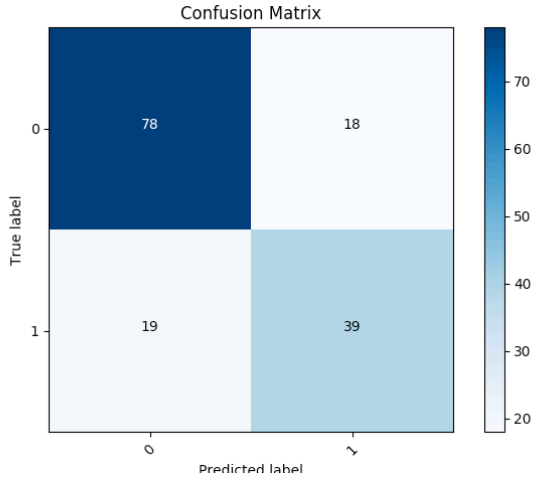
For Figure 10a, the ideal maximum tree depth

for the cross-validation recall is actually just one, which is significantly less than it was for decision trees. Furthermore, the maximum depth that maximizes cross-validation AUC in Figure 10b is five, an increase from three previously.

Figure 11. Confusion Matrices for Boosted Decision Trees



a. Credit Card Data

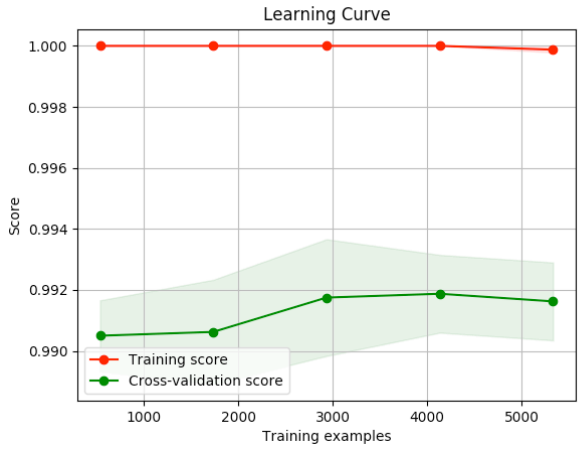


b. Pima Indians Diabetes Data

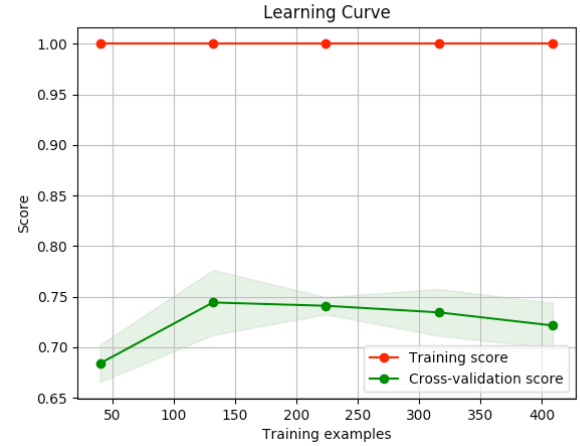
In Figure 11a, we can see that boosted decision trees has done very well in classifying non-fraudulent instances as non-fraudulent, but has not done very well in correctly classifying fraudulent instances, in comparison with the other algorithms. This, again, is a failure to fully account for the imbalance in the data.

In Figure 11b, we see that, although it has been the worst algorithm so far in determining the 0 case, it has by far exceeded the past two algorithms in its predicting that people have diabetes correctly. This is assuredly due to the focus that boosting focuses on previously misclassified values. Additionally, like decision trees, boosted decision trees can produce accurate results from smaller datasets.

Figure 12: Learning Curves for Boosted Decision Trees



a. Credit Card Data



b. Pima Indians Diabetes Data

Both learning curves both indicate a possibility that of high bias and overfitting. This is substantiated by the fact that training scores are consistently about 1, and by the face that cross-validation scores begin to decrease with more training data. Another possible cause is that, due to cross-fold validation, the depth of the tree was optimized to work with less data.

If we were to remake this algorithm, we would add the number of estimators to be another hyperparameter to tune with cross-validation. This would result in increased overall accuracy of the algorithm.

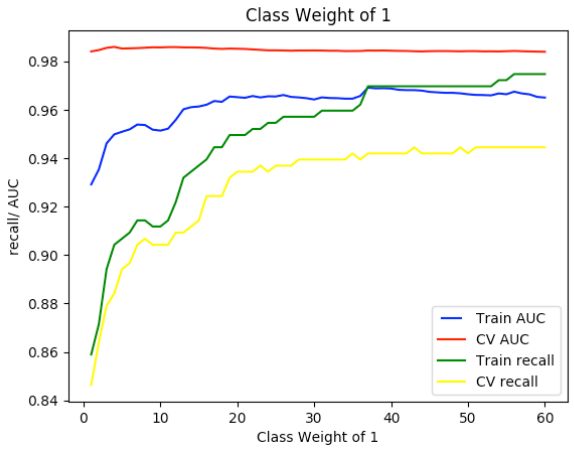
The most important features for the credit card boosted decision tree are V3 and V4. For the diabetes dataset, they were BMI, Plasma Glucose

Concentration, and Age, in that order.

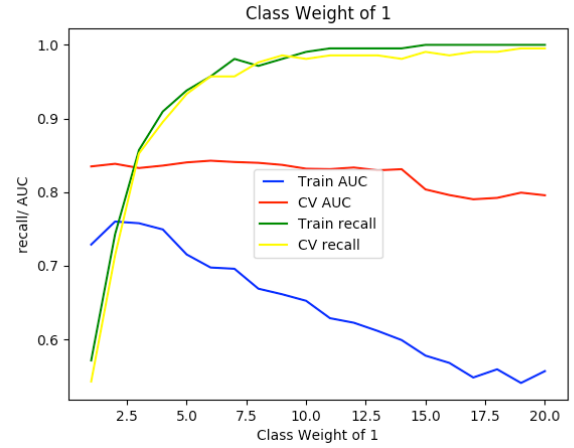
V. Support Vector Machines

For support vector machines (SVM), we use cross-validation to select the ideal weight of the class of 1. Since we have less instances of 1 in both our datasets, increasing our its weight should increase the recall score. Then, we train the SVM with the ideal weight of class 1.

Figure 13: Recall/ AUC Score vs. The weight of class 1. Note that the scales are different in each plot.



a. Credit Card Data

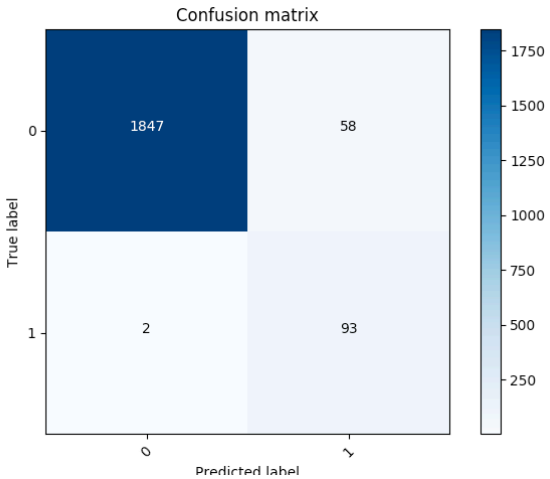


b. Pima Indians Diabetes Data

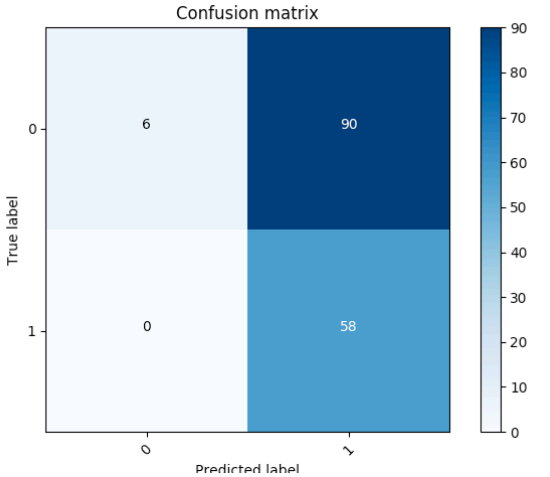
In both of these curves, the train and cross-validation recall scores increase greatly as the class weight of 1 increases. This is expected since we are placing more 'value' in correctly classifying 1, due to the imbalance with 0.

In Figure 13a, train AUC is increasing or staying level, but it is decreasing in Figure 13b. This is because the data is much more imbalanced in the credit card data than in the diabetes data. As a result, whereas a large weight of class 1 makes the data more balanced in the credit card data, a large weight of class 1 in the diabetes makes it almost 'ignore' the classification of 0.

Figure 15: The Confusion Matrices for Support Vector Machines



a. Credit Card Data



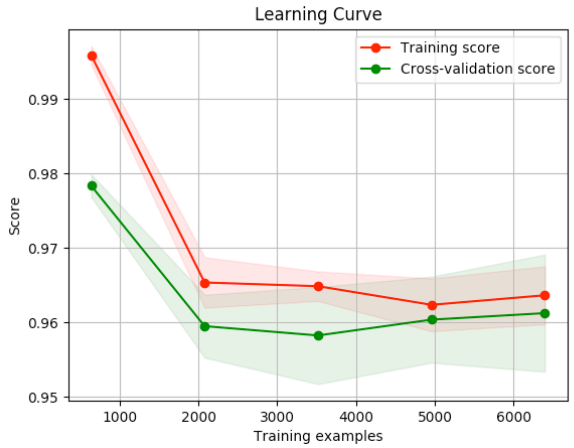
b. Pima Indians Diabetes Data

From Figure 15a, it is easy to see that our SVM classified more of the instances with value 0 incorrectly, which is by far the least accurate it has been. The tradeoff, however, is that almost all of its predictions with instances of true value 1; only 2 are incorrectly classified as 0. This is largely due to the large weight given to class 1.

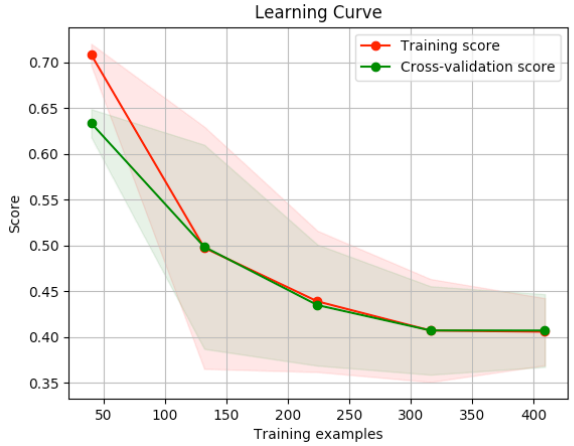
By its nature, ROC AUC (the type of AUC we are using) puts little emphasis on false positives. Since both of our datasets are fairly imbalanced, this tends to work very well for our needs. However, it is apparent from Figure 15b that our SVM predicts a label of one for almost every

instance, and is highly biased. This is due to the fact we maximize using the AUC and the large class weight of 1.

Figure 15: The Learning Curves for Support Vector Machines



a. Credit Card Data



b. Pima Indians Diabetes Data

It is difficult to describe why the cross-validation scores are mostly decreasing as the number of training examples increase in Figures 15a and 15b. If we consider the possibility that a higher proportion of instances in class 1 are encountered as we increase the number of examples, a large weight on one class results in lower overall accuracy. Since this is what the curve is measuring, this is most likely why our cross-validation curves are decreasing.

While implementing this SVM, we had to keep increasing the test for the class weight of 1 for the credit card data because the cross-validation recall of 1 kept increasing without significant changes to the cross-validation AUC.

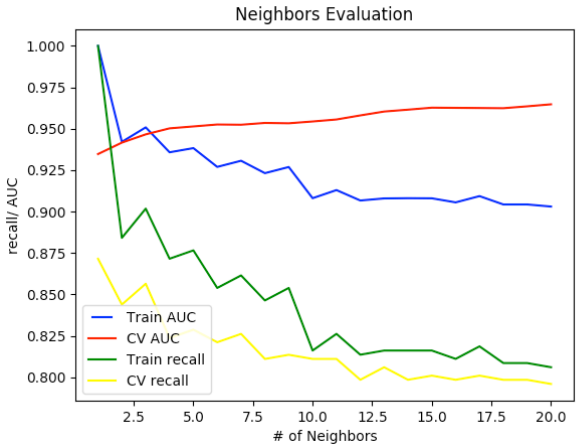
If we were to implement SVM once again, we would find a choose a metric other than AUC with whichto maximize the diabetes data. We would use a metric that takes false positives into greater account. This would prevent the SVM from simply predicting a value of 1 nearly every time.

The features with the highest coefficients for the credit card data were V11, V4, and V28. For the Pima Indians data, they were Age, Plasma Glucose Concentration, and Triceps Skinfold Thickness.

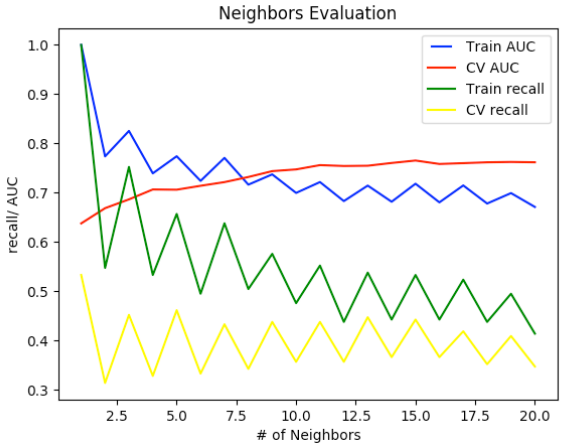
VI. K-Nearest Neighbors

For k-Nearest Neighbors (KNN), we use cross-validation to select the ideal number of neighbors. We then determined the type of weight function to apply, but the results are so insignificant that they are omitted. Finally, we train our algorithm with the ideal number of neighbors and obtain the results.

Figure 16. Recall and AUC vs. the Number of Nodes in each layer



a. Credit Card Data



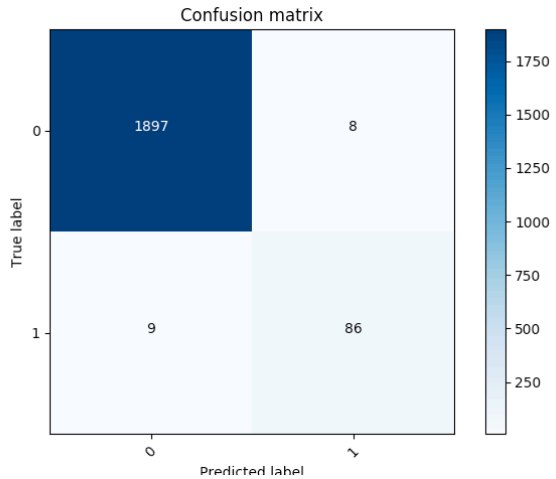
b. Pima Indians Diabetes Data

In both these graphs, the downward trend of the train AUC and recall indicate that we are underfitting our dataset, leading to higher variance as we increase our number of neighbors.

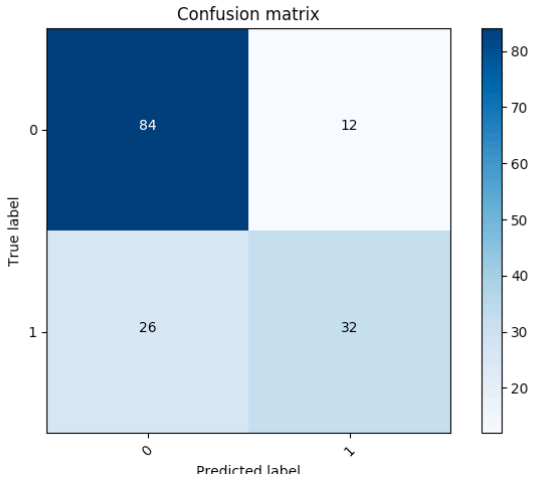
In Figure 16a, we see that 1 neighbors maximizes the cross-validation recall function, indicating that our algorithm will have low variance.

On the other hand, 15 neighbors maximizes the cross-validation AUC function in Figure 16b. So, this algorithm will have high variance.

Figure 17. Confusion Matrices for k-Nearest Neighbors



a. Credit Card Data



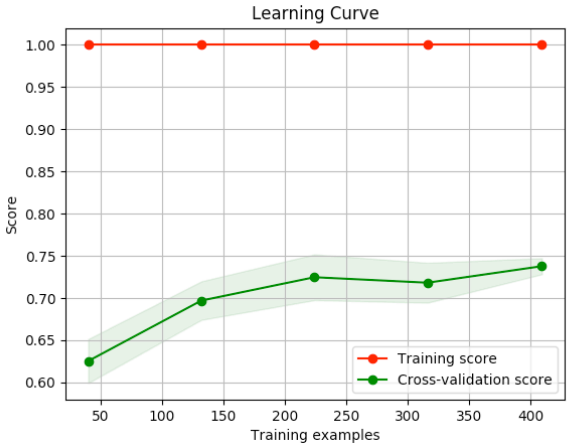
b. Pima Indians Diabetes Data

From Figure 17a, we can see that our k-nearest neighbors implementation for credit card data was about average; it was not very high in either category. The k-nearest neighbor algorithm isn't performing well because it is not robust to noisy or imbalanced data.

From Figure 17b, we determine that our implementation performed quite well in predicting the correct values when compared to our other algorithms. Note that this implementation was neither the best at predicting true labels correctly nor false labels correctly. It is working well with this dataset because it has very low bias, and it does not make many infer about the data.

Figure 18. Learning Curve for k-Nearest Neighbors

a. Credit Card Data



b. Pima Indians Diabetes Data

Both of the training curves in Figure 18 are about what we expect normally. The cross-validation score rises as we increase our training examples because the algorithm is given more information.

In Figure 18b, overfitting and high variance can be observed, due to the distance between the curves and training score.

If we were to redo our implementation of k-nearest neighbors, we would simply tune another hyperparameter, such as the leaf size in hope that the resulting algorithm would be significantly improved.

VII. Results

A. *Credit Card Fraud Detection (CC)*

The algorithms most successful for this dataset were clearly the neural network and the SVM. The neural network was effective because can easily learn rich relationships and works well with large datasets. The SVM was effective mostly because we tuned the weight of the class 1 to make the dataset more balanced.

Overall, although the neural network classified more data correctly, the SVM had a significantly higher recall score. With this dataset, the recall score is the most important metric because we would we want to correctly determine which transactions may be fraudulent first and foremost. SVM classified almost 98% of all fraudulent transaction correctly, much higher than any other algorithm.

The most important features were V17, V11, V3, and V4. Other important features include V28, V14, and V13.

B. *Credit Card Fraud Detection (CC)*

The algorithms most successful were k-nearest neighbors and the boosted decision tree. The k-nearest neighbor was effective because it does not infer much from the dataset, so it essentially lets the data speak for itself. The boosted decision tree was effective because it focuses on previously incorrect predictions. As a result, it classified over 67% of all actual diabetic persons correctly. The algorithm also works well with small datasets. The boosted decision tree is the most effective because its accuracy is by far the highest.

The most important features plasma glucose concentration, age, and BMI.

References

1.17. Neural network models (supervised). (2017). Retrieved from http://scikit-learn.org/stable/modules/neural_networks_supervised.html

Confusion Matrix. (2017). Retrieved from http://scikit-learn.org/stable/auto_examples/model_selection/plot_confusion_matrix.html

Credit Card Fraud Detection. (2018, March 23). Retrieved from https://www.kaggle.com/mlg-ulb/creditcardfraud/home

Nearest Neighbors Classification. (2017). Retrieved from http://scikit-learn.org/stable/auto_examples/neighbors/plot_classification.html

Pima Indians Diabetes Database. (2016, December 13). Retrieved from https://data.world/data-society/pima-indians-diabetes-database

Plotting Learning Curves. (2017). Retrieved September 22, 2018, from http://scikit-learn.org/stable/auto_examples/model_selection/plot_learning_curve.html

Pruning Decision Trees. (March 26). Retrieved September 21, 2018, from https://stackoverflow.com/questions/49428469/pruning-decision-trees

Saxena, R. (2017, February 19). Building Decision Tree Algorithm in Python with scikit learn. Retrieved from http://dataaspirant.com/2017/02/01/decision-tree-algorithm-python-with-scikit-learn/

Sklearn.svm.SVC. (2017). Retrieved from http://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html

Two-class AdaBoost. (n.d.). Retrieved from http://scikit-learn.org/stable/auto_examples/ensemble/plot_adaboost_twoclass.html

Zakka, K. (2016, July 13). A Complete Guide to K-Nearest-Neighbors with Applications in Python and R. Retrieved from https://kevinzakka.github.io/2016/07/13/k-nearest-neighbor/#parameter-tuning-with-cross-validation