Supervised Learning

Exploration of supervised learning algorithms

Roshan Gajurel
Georgia Institute of Technology
Rgajurel3@gatech.edu

Abstract—This paper explores five supervised learning algorithms and compares them using two different data sets. `

I. Introduction

Among the various techniques available for supervised learning, this paper focuses on five of the widely used techniques:

- Decision trees
- Neural networks
- Boosting
- Support Vector Machines
- k-nearest neighbors

We will explore these algorithms using two different datasets and analyze how it behaves under a variety of circumstances. We will analyze the behavior of these datasets based on the impact caused by change in parameters of the learning algorithms. Scikit Learn libraries was used exclusively to run all the analysis and graphs. We will also compare the outcomes of different learners and discuss it. Accuracy is used to determine how well a classifier did in contrast to error rates, which can be derived from the accuracy.

II. Exploration using Breast Cancer Wisconsin (Diagnostic) Data Set

Breast Cancer Wisconsin is one of the most popular datasets used to explore different learning algorithms. Since this is our first exploration, we are starting with something that is not overwhelming but not trivial at the same time. Also, it is one of the early datasets that was able to prove the usefulness of learning algorithms in the real world. The models derived from this dataset was used in the University of Wisconsin Hospitals to diagnose malignant vs benign cancerous cells [1]. At the time it was fascinating to see a computer predict cancerous cells with such accuracy. It makes you wonder about other areas that could benefit from these learning algorithms. This dataset consists of 569 datapoints with 32 features of which 357 are benign and 212 malignant. The derived models from this dataset are easy to visualize which will allow us to understand the robustness of these algorithms.

A. Decision Trees

Decision trees are simple to understand and visualize. Once the models have been built the cost of use these trees are logarithmic, hence robust. However, they are susceptible to overfitting, small variation in data can completely change the tree and can create biased trees if some classes dominate.

We will analyze the data of the tree by plotting various graphs that demonstrate the characteristics of the tree. We have used GINI index to split the attributes since GINI performed slightly better compared to information gain. GINI measures how often a randomly chosen data point would be incorrectly labeled if the data point was randomly labeled. Whereas, information gain is measured by entropy.

First let us look at the learning curve of the tree.

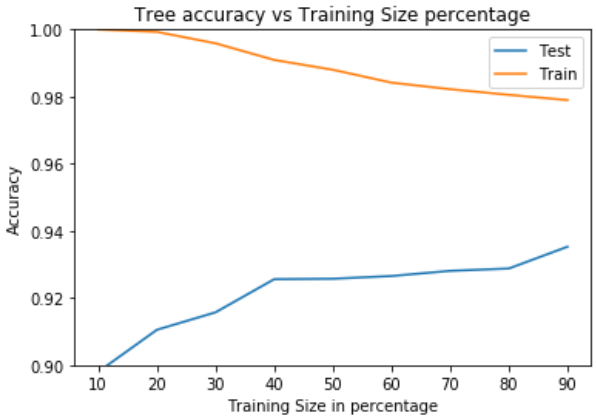


Fig. 1 Decision Tree Learning Curve

We can see that as the training size increases, the accuracy of the test data increases, which means the error rate decreases. By looking at the curve we can tell that the model still has a higher variance. Increasing the training data would make the model better because it would reduce the error rate.

Next, we use two pre-pruning methods and compare each other. Pruning is the method used to reduce the size of decision tree by removing the nodes of the tree that might be overfitting. The first method is to vary the depth and the second method is to vary the node size of the trees. To make the graph less volatile, we took the average of the accuracy score versus depth and max leaf size over ten iterations. Below we can see that both tree depth and leaf node size effect the accuracy. When they are one the model does not perform well but gets better as the depth and number of node increase. However, at one point we can see that the increase in the number of nodes and depth has no effect in the accuracy and this point can considered optimal pruning parameters for this data set.

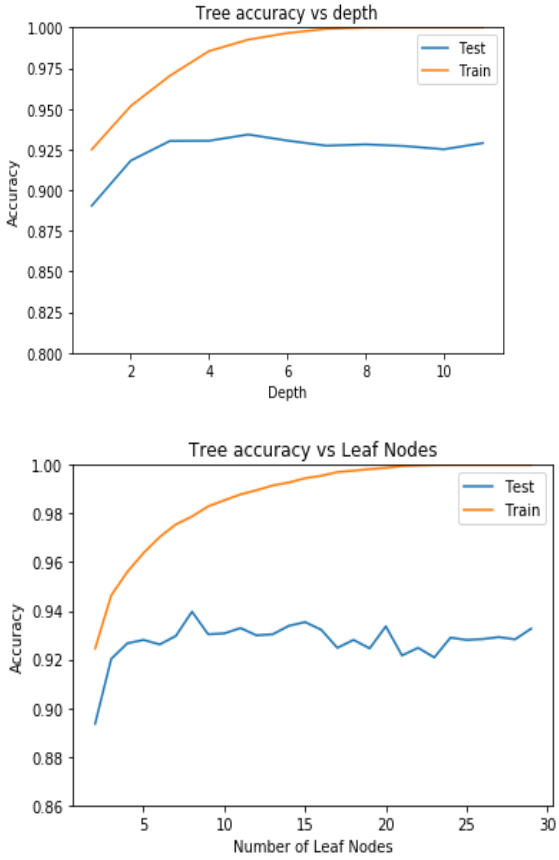


Fig. 2. Decision Tree Pruning methods

Based on the graphs, the highest accuracy is around 94%. We might be able to increase this if we had more data, however we can use cross-validation to train the model better. After performing a **10-fold cross-validation**, we were able to get the best crossvalidation score of **95.3%** which is a decent increase. We also saw an accuracy of 93.706% against a never seen test data on the same model. By running different iterations of all the hyperparameters we found that these were the ones that were able to best minimize the error rates for this data set, splitting criteria of GINI, maximum depth of 4 and maximum leaf node of 8. The wall time for running grid search for the above parameters was 22 seconds which is a lot faster than some other classifiers.

B. Neural Network

For this dataset we used a multi-level perceptron (MLP) which trains using backpropagation. It can learn non-linear models in real time using partial fit. However, since the hidden layers can have more than one local minimum, random weight initialization can lead to different validation accuracy.

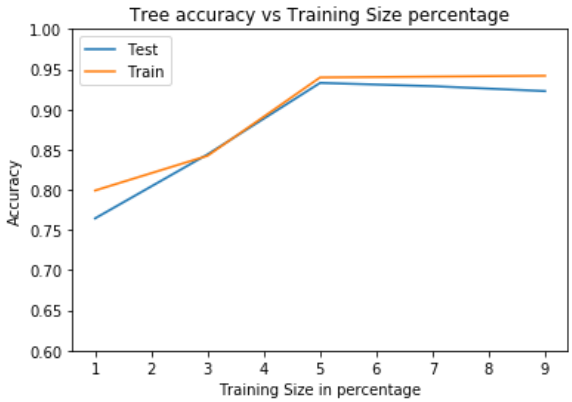
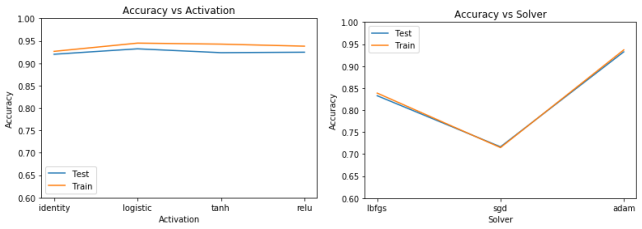
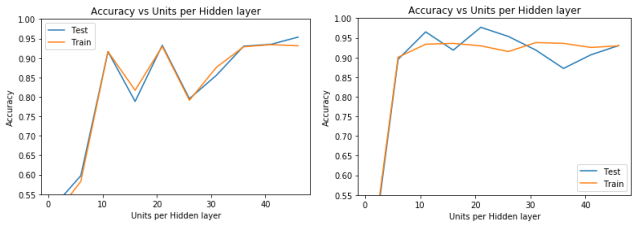


Fig. 3 Neural Network Learning Curve

Looking at the learning curve we can see that both training and testing accuracy increases when the training size is increased. Which is different than decision tree. However, we see that after train test split of about 50% the model starts to overfit.

We used the following parameters to tune the model and test which would perform better over others, activation, solver, hidden layers, preprocessor and scalars.





1 hidden layer *3 hidden layers*

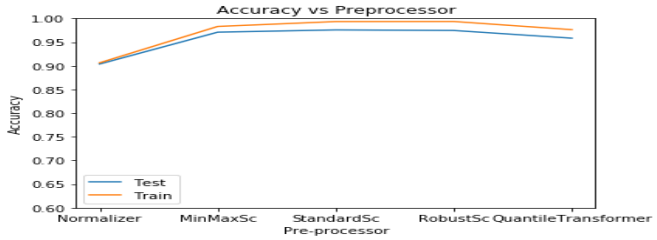


Fig. 4 Neural Network parameters

As we can see that activation method did not have a significant impact on our dataset, however the adam solver performed the best among the solvers. When we experimented with hidden layers, increasing the number of units per layer did not have a significant impact after fifty and the models performed similar on one and three layers. Among the preprocessors, standard scalar and robust scalar performed the best.

Finally using the best performing parameters, we performed a 10-fold cross-validation which had the best score of 92% on test data. However, by using a standard scalar to remove the mean and scaling to unit variance of the training data we were able to achieve an accuracy of 97.6% on the test data, which is better than the decision tree. To add to the accuracy, the results of grid search using various hyper-parameters returned a best F1 score of **98.61%.** These were the best parameters used: alpha of 1, hidden layer sizes of 500, adam as the solver, and robust scalar as the scalar. One thing to note is, I had to run the algorithm overnight. So, even though the accuracy is better than other algorithms, the cost of training the model is very high.

C. Boosting:

AdaBoost from the skit-learn library was used as the boosting algorithm with decision trees as the base estimator. AdaBoost fits a classifier on the original dataset and then fits other copies on the same dataset but adjusts the weights of incorrectly classified instances so that it focuses more on the difficult scenarios subsequently. Let's look at the learning curve for AdaBoost.

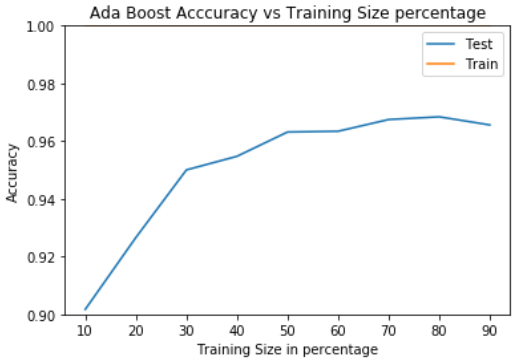


Fig. 5 Ada Boost Learning Curve

We can see that it classifies the training data perfectly and we can also see that it does better with the testing data as the test size increases however after 80% of the data there does not seem to be any progress in the accuracy and the model bias increases at this point.

Let us look at how the model behaves when different boosting parameters are changed.

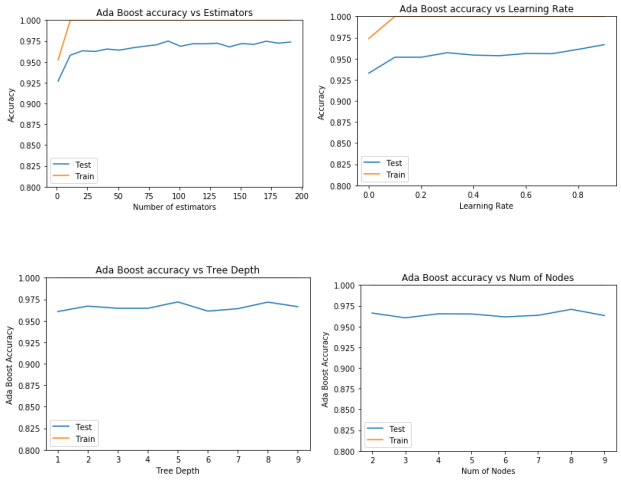


Fig. 6 Ada Boost parameters

For this dataset 90 seems to be the optimal number of estimators and the model performs better when the learning rate is close to 1. One interesting thing to note is that tree depth and number of nodes of the decision tree is not significant. If we look at fig 2, decision trees did not perform well when the depth and number of nodes were low. This tells us that, for this dataset, we can be very aggressive with pruning and still get good accuracy from our model.

Next, wen we ran a 10-fold cross-validation, the model returned a best cross-validation score of 96.94%. And using grid search, the optimal parameters were able to return the accuracy score of **98.601%.** Which is what the neural network predicted, we were able to find the optimized model a lot faster using grid search for AdaBoost compared to neural network. The wall time of grid search was 6 min 58secs.

D. Support Vector Machines

Support Vector Machines (SVMs) are another supervised learning method which is effective in high dimensional spaces even when the number of dimensions is greater than the number of samples. Also, it can use different kernel functions can be used as decision functions based on the dataset. However, SVMs are expensive because it uses 5-fold cross-validation to perform probability estimates. CSupport Vector Classification (SVC) from the scikitlearn was used for this dataset. Since the breast cancer dataset has higher number of features compared to the size of the data SVM should perform well.

Let's look at the learning curve first. This was made using all the default values for the parameters of the classifier.

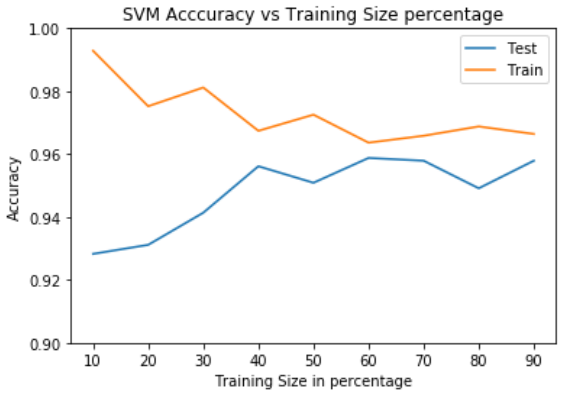


Fig. 7 SVM Learning Curve

Just like decision trees, we can see that as the training size is increased the model gets better. However, I ran the above algorithm multiple times and I could see that it over fit on some of the models.

Next, I ran the SVM by changing different parameters of the classifier like, Kernel and Gamma.

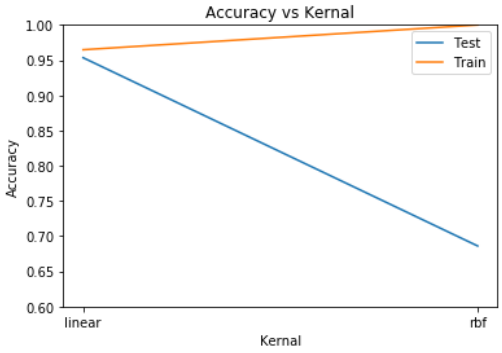


Fig. 8 SVM Accuracy vs Kernel

I tested the data set against the linear and rbf kernel. With linear kernel, we can see that the testing data performed well however, with rbf it did not. The accuracy went below 70% with rbf kernel. This is probably because we can see that the model starts to overfit. Next, we look at the impact of gammas.

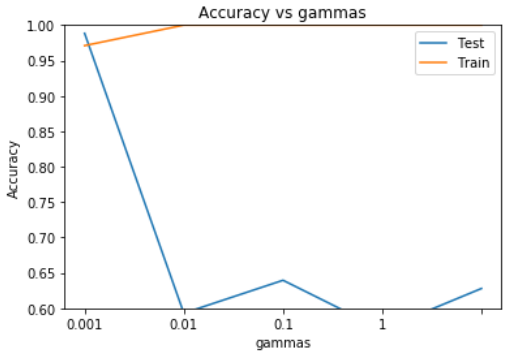


Fig. 9 SVM Accuracy vs gamma

As we can see that, that the model performs better when the gamma is low. So, a linear kernel with a low gamma would optimize the SVM for this dataset.

This was verified by running a grid search with additional parameters like decision function with 'ovo', and 'ovr', class weight as balanced and None, probability as True, False including kernel and gamma sets. Grid search validated our assumption that linear kernel with a low gamma was better. Additionally, it also returned probability set to true, C as 1.0 and ovr as the decision function shape would produce the optimal model. The best model had an accuracy of **95%** with the test data, and a 10fold cross-validation best score was **96%** which is lower than Neural Network and SVMs but is slightly better than decision trees.

E. k-Nearest Neighbor(k-NN)

KNeighbourClassifier from scikit-learn was used as the classifier. KNN is different than other model in that it does not create a model as other classifiers, rather it stores the training data and then it queries the stored data for the nearest neighbor of the data point to make predictions. As a result, training the data is faster but prediction takes longer and can be very costly for larger datasets.

Let's look at the learning curve first.

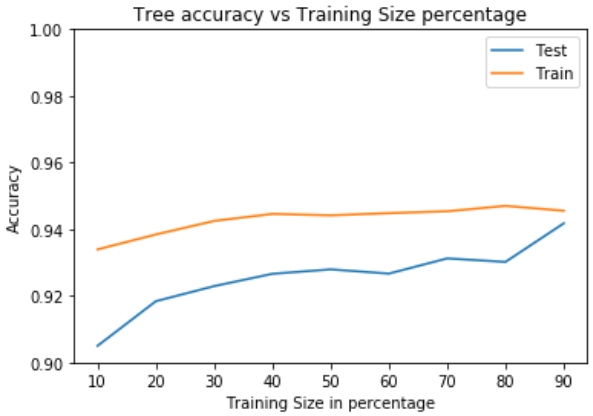


Fig. 10 k-NN learning curve

From the learning curve we can see that as the number of training data increases the accuracy of the model increases as well and the error rate of the model decreases. We need to be careful about data with noise. It can reduce the accuracy of the model. One way of managing this is by increasing the number of neighbors. The below graph shows the effect of number of neighbors on accuracy.

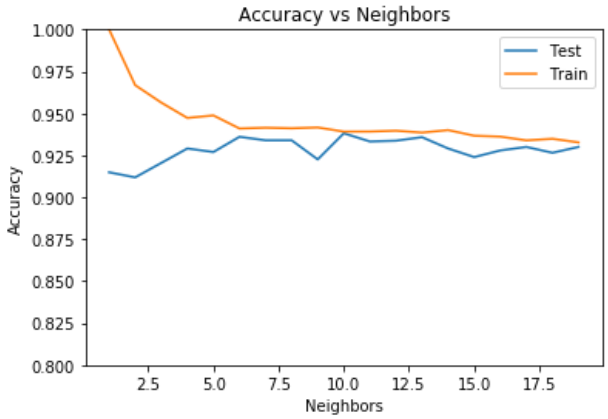


Fig. 11 k-NN learning curve

As we can see that increasing the number of neighbors increases the accuracy for around 10 neighbors and then there does not seem to be any significant increase in the accuracy after that.

Since the dataset is small k-NN had a very fast wall time. Even under grid search with different neighbors and 10-fold cross validation it was able to find the best model in less than 1 second. However, the accuracy from k-NNs was the worst compared to all other classifiers. It returned a best cross validation score of **93.19%** against test data. The best parameters for this dataset was 8 neighbors, leaf size of 30 and uniform weight.

III. Exploration using Credit Card Fraud Detection Data Set

Credit card fraud detection data set was the second dataset we choose for exploring these algorithms. This dataset can be found in Kaggle. It contains anonymized credit card transactions of customers that has information about if the transaction was fraudulent or not. This is a realworld dataset collected over 2 days in September 2013 from European credit card companies. Applying supervised learning algorithms on this dataset will be interesting just because of its real-world application and the impact it can create. A well-trained model that can predict fraudulent behavior with accuracy can create a new method for companies to track fraudulent behavior which would not have been possible with this efficiency without these learning algorithms. This dataset contains 284, 807 data points of which only 492 are fraudulent cases. Since the dataset is highly unbalanced with fraud accounting for only .172% of the dataset. First thing we will do is make the dataset more balanced by having the ratio of fraudulent to non-fraudulent 1:10 or 5500 datapoints on different algorithms.

A. Decision Trees

Just like before let's start with looking at the learning curve for the dataset. The first curve is

from 1500 datapoints with a fraud to not ratio of 1:2.

Fig. 12 Decision learning curve 1

Here on the first graph we can see that the training and test accuracies are converging as the training size increases. We can tell from the graph that if it had more data it would have performed better. So, let's increase the number of datapoints to 5500 data points with a ratio of 1:10.

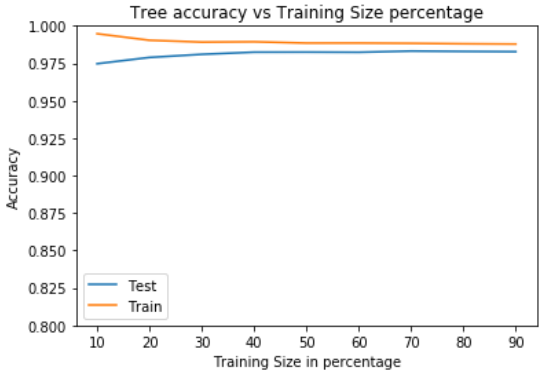


Fig. 13 Decision learning curve 2

In this case that the tree performs much better. The accuracy on test data has increased from 94% to around 97%.

Next, lets look at how the tree performs under pruning.

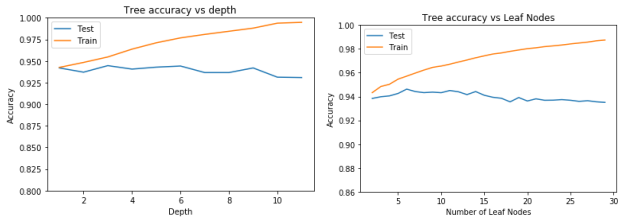


Fig. 14 Decision learning pruning parameters

The outcome of the decision tree classifier is comparable to breast cancer dataset. The increase in depth increases the bias of the model and it can better classify the training data however the effect on testing set does not increase significantly after the depth of 3. Similar results are seen with the increase in node, 8 seems to be the most optimal max node size for this dataset.

Comparing the performance of the smaller (1500) and larger dataset (5500) is quite interesting. The best cross-validation score for the smaller dataset was 94.5% however it went up to **98.5** percent on the larger dataset which validates that increasing the number of training data improves the model. The only draw back was that it took longer to train the model.

B. Neural Network

We are using a dataset with size of 5500 for neural network since they can perform well on larger datasets. However, it also takes longer to get results from these datasets. Let us look at the learning curve first.

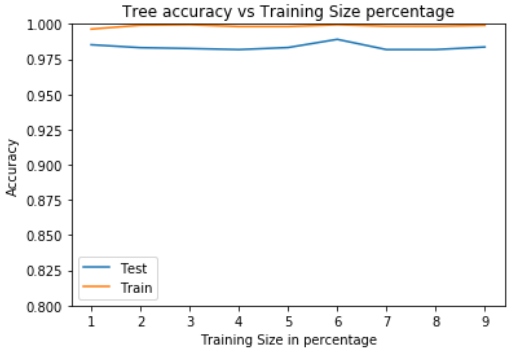


Fig. 14 Neural Network learning curve

We can observe that this curve is different compared to other learning curves we have seen. We can see that test and training size do not seem to converge and are relatively unaffected by the increase in training size. It is interesting that the neural network can perform similarly by training on 10% and 90% of the data.

The same parameters that were tuned for breast cancer data was used for Neural Network as well which are, activation, solver, alpha, hidden layer and pre-processor.

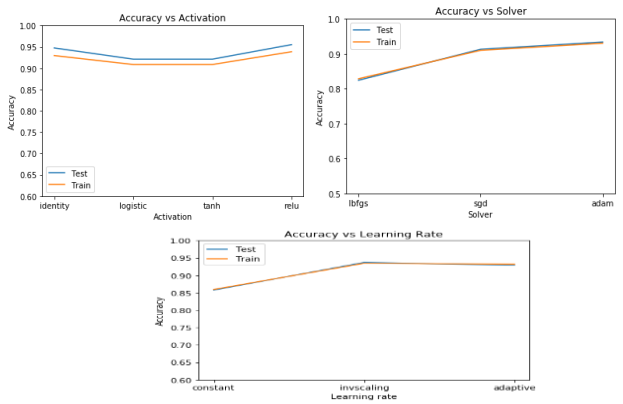


Fig. 15 Neural Network parameters

We can see that logistic and tanh activation perform slightly worse than identity and relu but not very significantly. Which is comparable to our first analysis. We can also see that adam is still the best solver for this data set as well like before. Invscaling seems to be the better learning rate compared to constant and adaptive for this dataset.

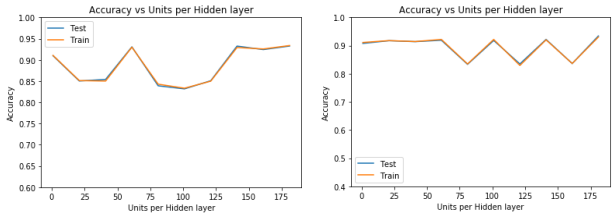


Fig. 16 Neural Network hidden layers

The first graph above shows the accuracy when the units per hidden layer is changed on one layer, the second graph is the same with 3 layers. Looking at the graphs it seems the number of units are per hidden layers does not affect this dataset for the values since the graph seems to be random. The outcome is similar with 3 layers as well, however the graph seems to perform stable for the first 60 units per layer.

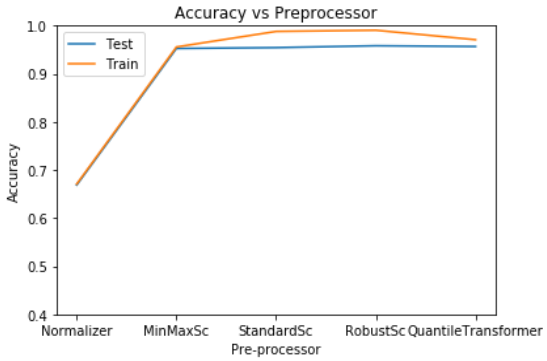


Fig. 17 Neural Network preprocessors

Performing some pre-processing however has normalized the data better and the accuracy is much better when Standard Scalar and Robust Scalar were used. This behavior is similar to what we observed with the breast cancer dataset. When 10fold cross validation was performed without preprocessor the accuracy was 91% and when Standard Scalar was used the accuracy jumped to **98.4%** on test data. The performance of this model is comparable to that of decision trees. This result is different compared to the breast cancer dataset because neural network had better accuracy compared to decision trees. The significance of this cannot be stressed enough. Performing a grid search on this dataset on my machine was not feasible just because of the amount of time it takes. This is a huge limitation for neural network over other learners. Given the processing power we would have been able to create a model that would perform better.

C. Boosting

As with the breast cancer dataset, training set has an accuracy of 100% with the credit card dataset as well. Also, we can see that boosting does well even with less information. As the training size increases, accuracy increases but the increase is not as significant as other classifiers.

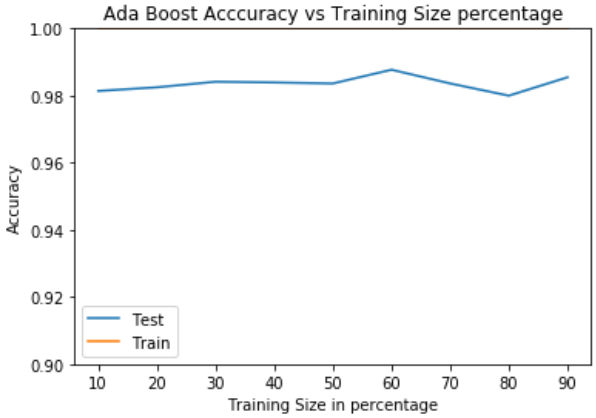


Fig. 18 Boosting learning curve

Next, we will see determine effect of the number of estimators and learning rate on accuracy.

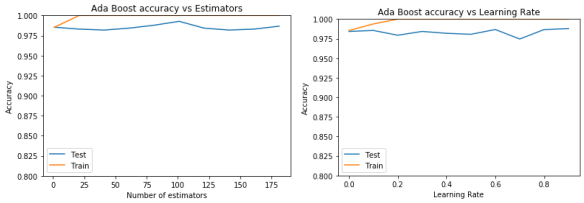


Fig. 19 Boosting estimators and learning rate

As we can see, for this dataset, 100 looks like the optimal number of estimator but it does not make any significant difference. The change in learning rate does not significantly affect the accuracy either.

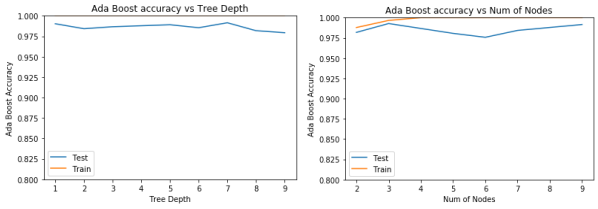


Fig. 20 Boosting tree depth and nodes

The analysis on tree depth and number of nodes is similar. Very aggressive pruning has little to no effect on accuracy. This result is very similar to what we saw for breast cancer data. Another thing common between the datasets is that they both performed very well. By using 10-fold crossvalidation and the best hyperparameters selected from our analysis, the model achieved an accuracy of **98.76%** on the test data. Which is better than neural network. By performing a grid search, it was observed that the optimal parameters were 2 for max depth, 3 for maximum leaf nodes, 1 for the learning rate and 800 for the number of estimators.

D. Support Vector Machines

Let's start with the learning curve for the SVMs.

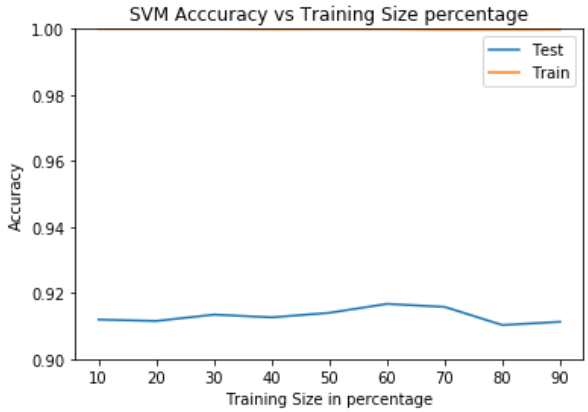


Fig. 21 SVM learning rate

The learning curve for SVM on the credit card dataset is different than that of breast cancer dataset. With breast cancer dataset we saw that the curve was converging, and we could tell that the increase in the number of data size increased the accuracy of the model. However, we do not see similar relationship here. We can see that the model does well with very small amount of data. This could be because of the difference in the size of data points. It tells us that SVMs can be trained with relatively smaller training size and still have a high accuracy. As

we can see from the graph the accuracy of the model is almost 92%.

Next let's look at the effect of kernel and gamma.

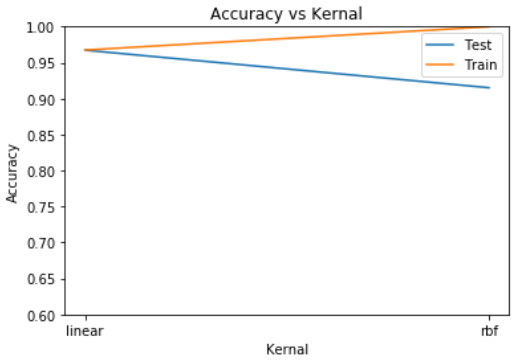


Fig. 22 SVM accuracy vs kernel

We can see that just like breast cancer dataset linear kernel performed better on this dataset as well as compared to rbf. And as we can see below the change in accuracy with the change of gamma was not significant.

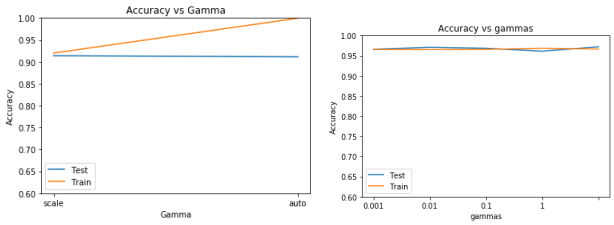


Fig. 23 SVM accuracy vs gammas

By performing a 10-fold cross-validation grid search, best model had an accuracy of 96% with the test data, and a 10-fold cross-validation best score was **95%**.

E. K-Nearest Neighbor

Lets us begin the analysis by looking at the learning curve of the data. We can see that the accuracy increases by a small number, but it does increase with the increase of training data. This is different than we have seen on other classifiers in that when the number of training size is small the training set usually performs well. This is since the default number of neighbors for the algorithm is 5 and since it averages the five nearest neighbors, training data does not have enough datapoints to make a good prediction.

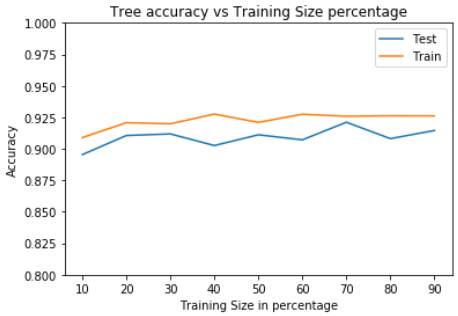


Fig. 24 k-NN learning curve

We can also see that the accuracy increases initially as the number of neighbors increases however after 8 neighbors there does not seem to be any significant increase in accuracy.

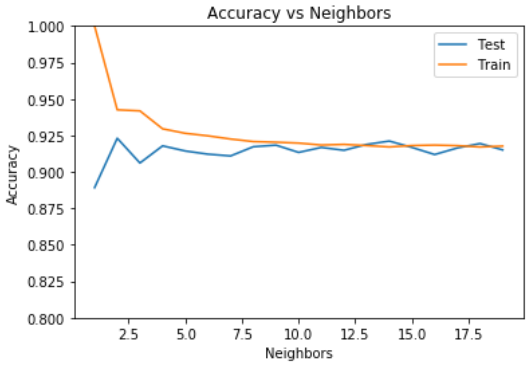


Fig. 25 k-NN accuracy vs neighbors

Our findings were verified by running a 10-fold cross-validation grid search with number of neighbors as one of the hyper parameters. We saw that it returned a best cross validation score of 91.9% and the same model returned a score of **90.45%** on a never seen test dataset. The wall time

for a grid search took 19.7 seconds and returned 4 as the optimal size of nearest neighbors with a leaf size of 30.

IV. Conclusion

We analyzed the breast cancer data set and credit card fraud detection dataset on five different learning algorithms. If accuracy is used to determine the best classifier, for breast cancer dataset, neural network performed the best with an accuracy of 98.61% followed by an accuracy of 98.60% by boosting. SVMs had an accuracy of 96% followed by 95 and 93 for decision trees and k-NN respectively. Even though it was almost a tie between neural network and ada-boost, with decision tree as the base estimator, I would conclude that boosting was better just because of the time it took to train the neural network compared to boosting.

When analyzing the credit card fraud detection data set, boosting had the best accuracy thus the least error rate with an accuracy of 98.7% followed by decision trees with 98.5 which was surprising. Decision trees performed better than neural networks slightly which had an accuracy of 98.4 followed by 95 for SVMs and 90 for k-NNs. It was interesting to see that for a binary classification problem, given a large enough training data, decision trees can perform as well as other more complex classification algorithms and is far less costly.

REFERENCES

[1] Street, W. Nick, William H. Wolberg, and Olvi L. Mangasarian. "Nuclear feature extraction for breast tumor diagnosis." Biomedical Image Processing and Biomedical Visualization. Vol. 1905. International Society for Optics and Photonics, 1993.