Daniel Kurniawan
GTID: 903043201
CS4641 HW1
9/20/17

Supervised Learning

Wine Quality – A dataset that contains information related to the red and white variants of the Portuguese 'Vinho Verde' wine. This dataset is split into two workbooks – one specifically for red wines and one for white wines. For our purposes, we are only concerned with the dataset for white wine, since it has more instances (4898) of data for the 11 attributes. For the most part, our data is made up of chemical contents of the wine such as fixed acidity, volatile acidity, residual sugar, etc. The output that we are trying to predict is the quality of the wine, expressed as an integer between 0-10. Thus, it can be modeled into a classification problem to try and predict the quality of the wine based on its chemical contents. I personally feel that this problem is interesting to assess how well wine quality is supported by its chemical properties – we do not have data on selling prices, brands, grape types, etc and thus we are working with limited data. From an algorithmic standpoint, it is interesting to see how well the machine learning techniques used for wine quality prediction perform with multi-class classification, especially when the classes are not equally balanced.

Pen Digits - The Pen-Based Recognition of Handwritten Digits Dataset consists of about 11,000 numeric writing samples. The dataset was created by collecting around 250 handwritten samples from 44 different writers, written on a pressure sensitive tablet that recorded the location of the pen every 100 milliseconds. The objective with this dataset is to classify a hand-written digit (0-9) based on 8 pen positions (x, y coordinates).

These datasets are interesting to me primarily because they have real-world applicability and their potential for applications for high-accuracy classifiers. With wine quality, there is potential to train a classifier that predicts wine quality, which can then aid a client in deciding whether or not to buy a particular wine. The pen digit classification problem is even more interesting since it deals with handwriting recognition and being able to identify what someone is trying to write.

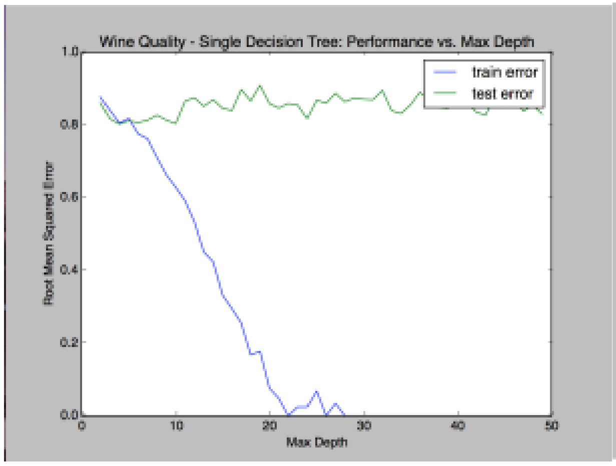
Algorithms

Each of the five algorithms were trained multiple times. For almost all the algorithms, various hyperparameters were tuned to assess performance (e.g. how to prune the decision tree, number of hidden layers and neurons per layer in a neural network, etc). All algorithmic runs were validated using mean square error over the training and test sets, and how accurate the algorithm is in predicting instances in the test set. For both datasets, a standard 70%/30% training and testing set split was used. Finally, before I began algorithm runs, I used the StandardScaler class in sklearn to standardize features by removing the mean and scaling to unit variance. All training times were roughly sub-second.

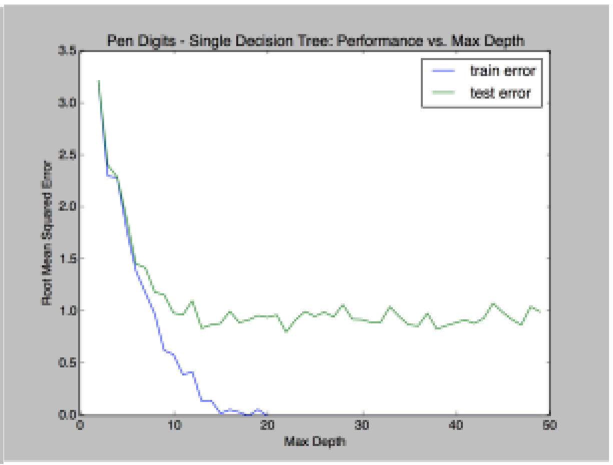
Daniel Kurniawan
GTID: 903043201
CS4641 HW1
9/20/17

Decision Trees w/ Pruning

In building a decision tree, I used sklearn's DecisionTreeClassifier class with the goal of pruning the tree using the max_depth parameter. I first ran an experiment to try and find what the best value for max_depth should be – also known as how much to prune the tree. What I found was that as max_depth increases (the deeper the tree) our error rates decrease. While there is some variance in the test error, we can see that in general the error rate for the test set slightly decreases. One potential reason for this variance is that we have relatively small datasets and a larger number of classes, so it is quite difficult to build an accurate decision tree. For example, with the wine quality dataset, we only have ~5000 instances mapped to 10 different classes. Regardless, I wanted to minimize any kind of overfitting and thus chose max_depths of 43 for wine quality and 22 for pen digits. Training time was basically negligible. One interesting experiment I would consider for the wine quality dataset is to make the output classes binary by establishing a threshold above which the wine is considered 'good' (could be if quality > 7 then the wine is labeled 'good') and otherwise the wine is considered 'bad.' I think that with fewer classes the accuracy of the decision tree would increase significantly (more on this later).



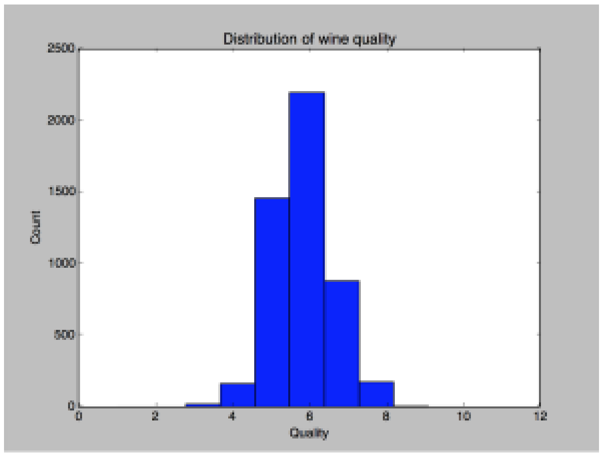
Accuracy: 57.54%

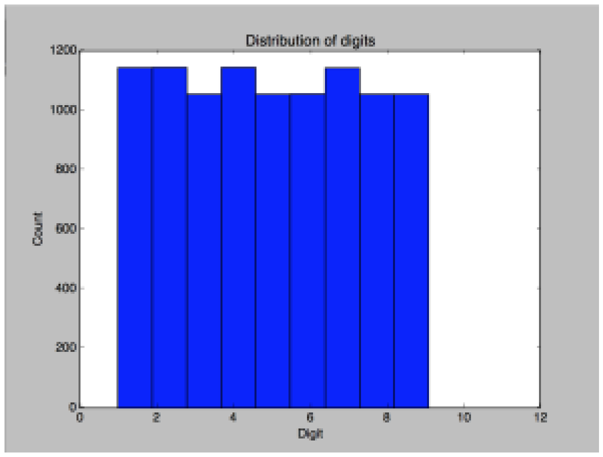


Accuracy: 95.63%

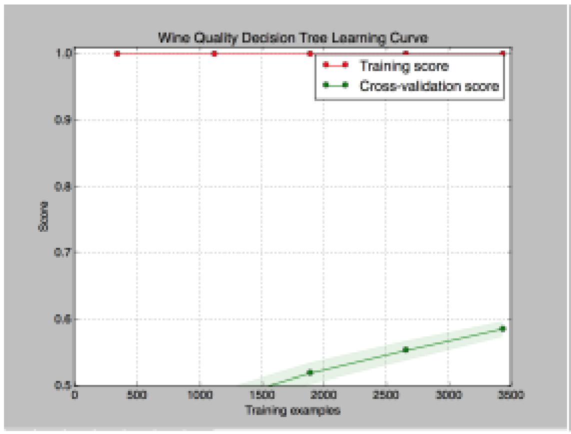
My reported accuracy for the decision trees are 59.4% for wine quality and 94.3% for the Pen Digit dataset. I think that with fewer classes the accuracy of the decision tree would increase significantly. While both problems have the same amount of output classes (integers 0-9) I believe that there are several key reasons why the wine quality dataset performs far worse. First, the wine quality dataset contains significantly less instances than the pen digit dataset (5000 vs. 11,000). Secondly, the distribution of classes is far more balanced in the pen digit dataset, illustrated below:

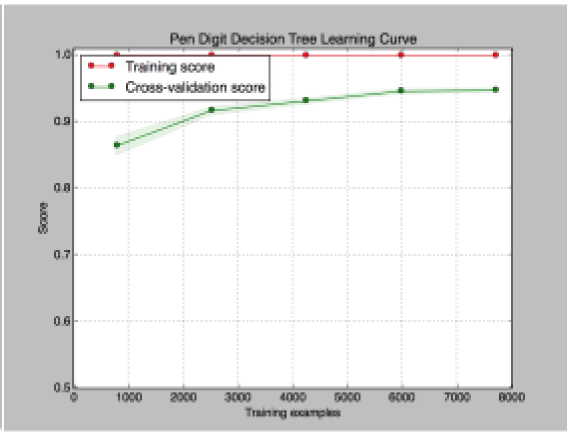
Daniel Kurniawan
GTID: 903043201
CS4641 HW1
9/20/17





We see here that the distribution of classes in both datasets is extremely different. While the pen digit has better representation of all classes, the wine quality dataset lacks representativeness, highlighted by the fact that in a set of 5000 instances, only 20 are labeled with quality of '3.' This makes it harder for the model to accurately predict/classify the correct quality of the wine.



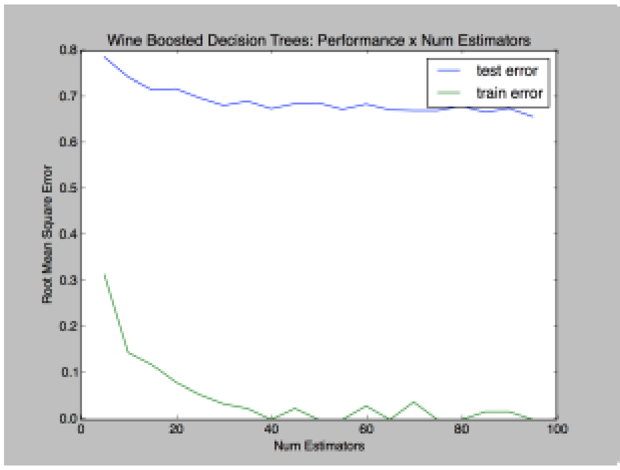


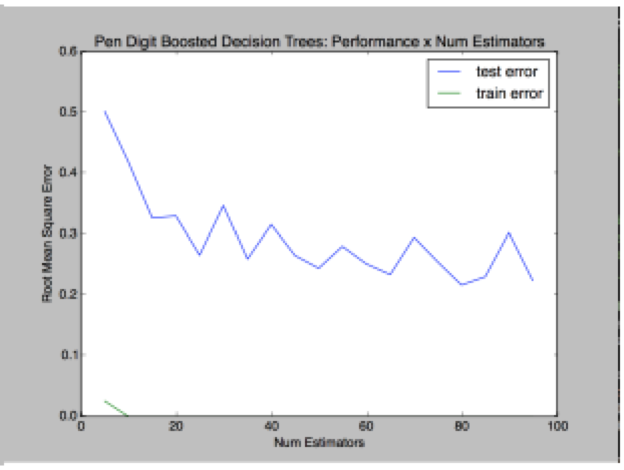
Lastly, I wanted to display plots of the learning curves for pruned decision trees. It makes sense that training accuracy remains 100% due to pruning, however in the wine quality dataset we see that there is much larger variance in the data denoted by the large gap between validation score and training score. Moreover, for wine quality, we need many more data samples to generalize well to the dataset and reduce variance. In the pen digit one, we see that the gap between training and validation is smaller indicating low variance, and that with more training samples the model can generalize well to new instances.

Daniel Kurniawan
GTID: 903043201
CS4641 HW1
9/20/17

Boosted Decision Trees

I chose to use the AdaBoostClassifier which is a meta-estimator that begins by fitting a classifier on the original dataset and then fits additional copies of the classifier on the same dataset but with a small twist - the weights of incorrectly classified instances are adjusted such that subsequent classifiers focus more on difficult cases. Max_depth was set to 10 for all cases.

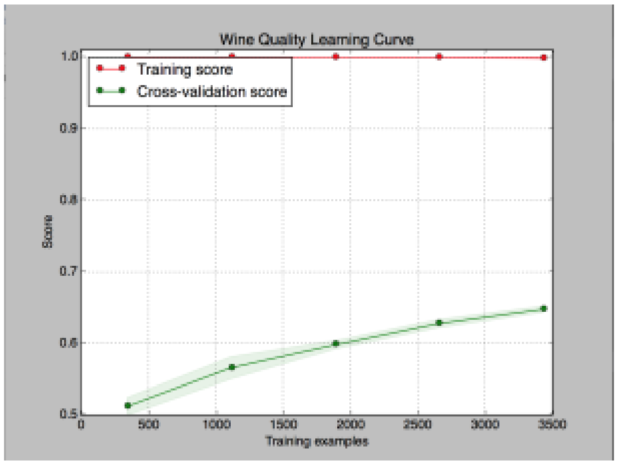




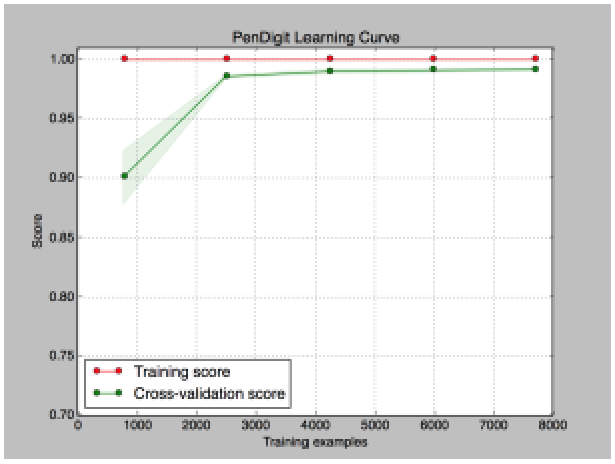
I'd like to note that after running the graphs of root MSE vs. number of estimators in decision trees and boosting, I realized that root mean square error does not mean anything for the context of the pen digit problem, since it is measure of the spread of the y values. In our case, whether a 4 is mis-classified as a 6 or a 9 does not matter, and thus the metric that we should really be looking at is classification accuracy and confusion matrices, which I will be using to assess performance for the next few algorithms. For the wine quality dataset, root MSE can still apply because it's a factor of how far away the predicted value is from the true quality of the wine.

The first comparison I tested for boosting was performance (root mean square error) based on the number of estimators used in boosting. For the wine quality dataset, anything larger than 40 as the number of estimators barely added any value in decreasing the test error of the model. For the pen digit dataset, I notice a bit of variation in the test error as the number of estimators increases, however anything above 22 does not make much of a difference. Using 40 as the number of estimators for the wine quality dataset and 22 for the pen digits, I then plotted a learning curve displaying the accuracy of the boosted decision tree based on the number of training examples, to see how the smaller wine quality dataset fared against the larger digit dataset.

Daniel Kurniawan
GTID: 903043201
CS4641 HW1
9/20/17



Accuracy: 66.34%



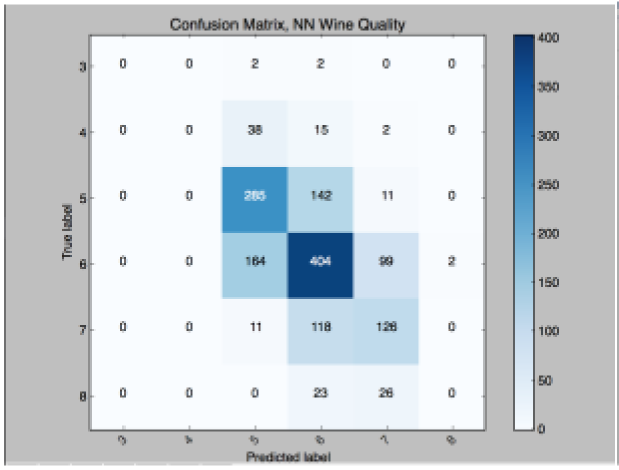
Accuracy: 99.30%

As seen above, the accuracy score for the wine quality classifier increases as the number of training examples increases. This was originally one of my hypotheses – as I mentioned in the previous section (see: Decision Trees w/ Pruning) I believed that a primary cause of bad performance in the wine quality dataset is that there is not enough data – and that there is not enough representativeness of all possible classes in the dataset. Moreover, the training score has peaked at 100%, which is a sign of overfitting, and there is a huge gap between the cross-validation score and the training score which shows high variance scenario. One potential way to fix this is to reduce the complexity of the model (which we will do following neural networks) or gather more data. For the pen digit dataset, accuracy has smoothed out at around 99%, so increasing the number of training examples won't have much of an impact on the performance of this classifier, other than helping the model generalize well with the data. There is potential that the classifier is overfitting the data due to a peaked training score of 100%, however variance is quite low as indicated by the small gap between training and validation curves.

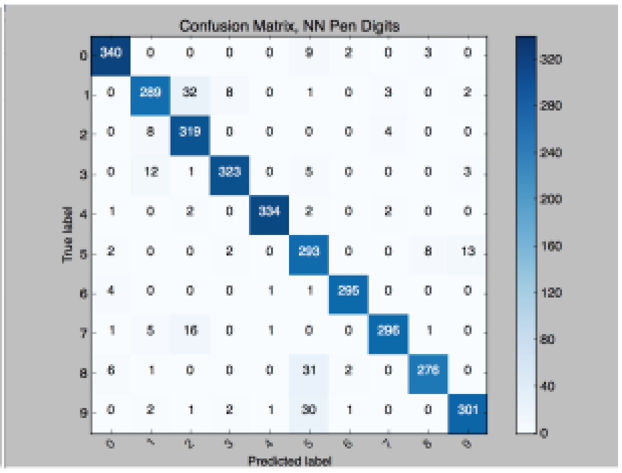
Neural Networks

To build a neural network for these two datasets, I used sklearn's neural network class to build a multi-layer perceptron classifier. Generally, the school of thought is to use a few layers and neurons as possible to reduce training time, model complexity, and make the solver's life easier. Reducing the number of neurons from layer to layer is also recommended so that each layer can learn higher and higher level features, and force the network to figure out how to encode enough information to produce the correct result. With the wine dataset, I mapped to a larger output than input for the first few layers.

Daniel Kurniawan
GTID: 903043201
CS4641 HW1
9/20/17

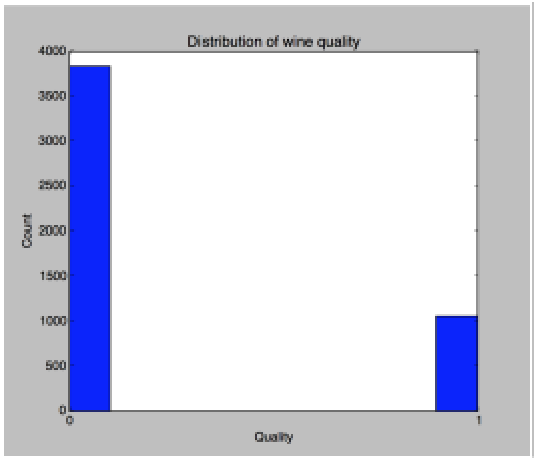


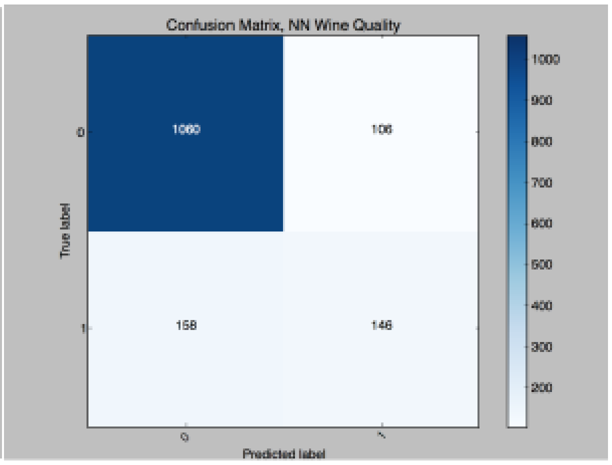
Accuracy: 55.34%



Accuracy: 94.36%

The confusion matrices shown above illustrate true vs. predicted labels in the test set and depict just how important data quality is in building an accurate classifier. In the wine quality dataset, we see that most the test set is comprised of instances with a class of either 5,6, or 7 whereas classes 3,4, and 8 barely have any representation in the testing set. Because the classes are extremely imbalanced, it is hard to perform multi-class classification on the wine dataset. On the other hand, the pen digit dataset has roughly equal representation of all classes, and thus it is much easier to build an accurate classifier. Due to consistently low performance of several classifiers on the wine dataset, I wanted to take a different approach and build a neural network using a threshold rather than performing multi-class classification. I turned this problem into a binary classification problem where 1 is mapped to 'good wine' and 0 is 'bad wine.' I expect this to perform better since now there are only two classes to predict, despite the presence of an imbalanced dataset.





Accuracy: 82.17%

As seen above, the classes are still imbalanced, but the reported accuracy is much higher (82.17%). This is mainly because now there is less classes to predict, and representation of these two classes in the dataset is better than before (even though there are significantly more 'bad' wines, there is still a good proportion of the dataset labeled as 'good.'

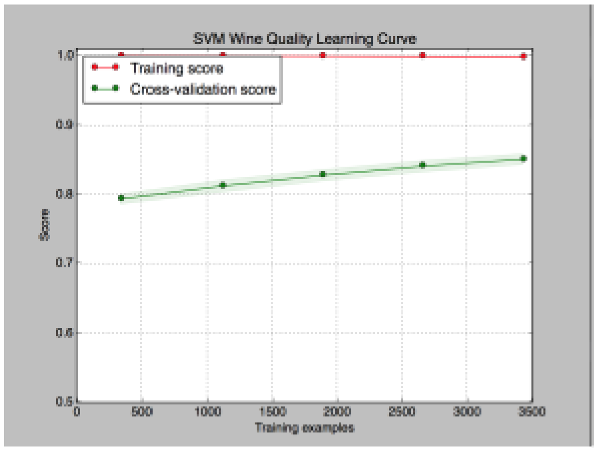
One interesting piece in developing neural networks for wine quality is that I found that performance was generally better when using the hyperbolic tangent activation function. The tanh function is also sigmoidal ("s"-shaped), but instead outputs values that range (-1, 1). Thus, strongly negative inputs to the tanh will map to negative outputs. Additionally, only zero-valued inputs are mapped to near-zero outputs. These properties make the network less likely to get "stuck" during training, and may be the reason why performance over the datasets was better.

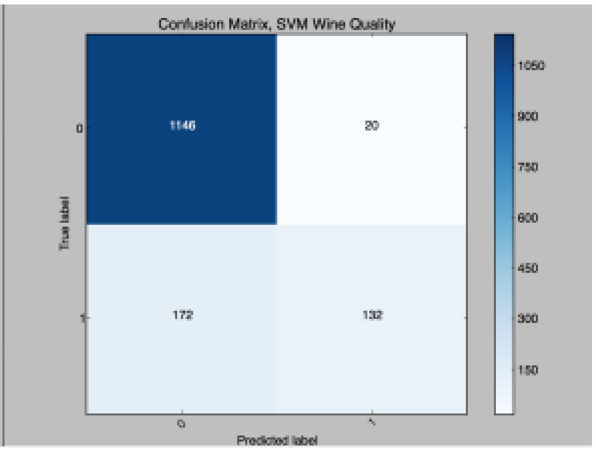
Another note: The solver 'adam' works well on relatively large datasets (with tens of thousands of training samples or more) in terms of both training time and validation score. For small datasets, however, 'lbfgs' can converge faster and perform better, so I used lbfgs as the solver. As a quick comparison, accuracy for lbfgs and a tanh activation function was around 82% whereas adam was 79.1% for the wine dataset, and for pen digits the accuracy for lbfgs and tanh was 94.36% whereas adam took significantly longer time to train (>5 minutes) and produced an accuracy of 63%.

Support Vector Machines

For the rest of the algorithms (SVM, kNN), I will be performing binary classification over the wine dataset instead of multi-class classification. I chose to do this because performance has been consistently bad over the wine dataset for the previous three algorithms (decision trees, boosting, and neural networks) due to imbalances in the dataset and I believe it will also be interesting to assess the performance of the wine quality classifier if the problem was approached a different way. Another key reason for this change in analyzing the problem is due to the high variance scenario (seen by the huge gap in training and validation curves) which was seen through the learning curves for decision trees and boosting – the best way to combat high variance is either to gather more data or to reduce model complexity. I chose to reduce model complexity by having the classifier predict from 2 classes compared to 7.

Daniel Kurniawan
GTID: 903043201
CS4641 HW1
9/20/17



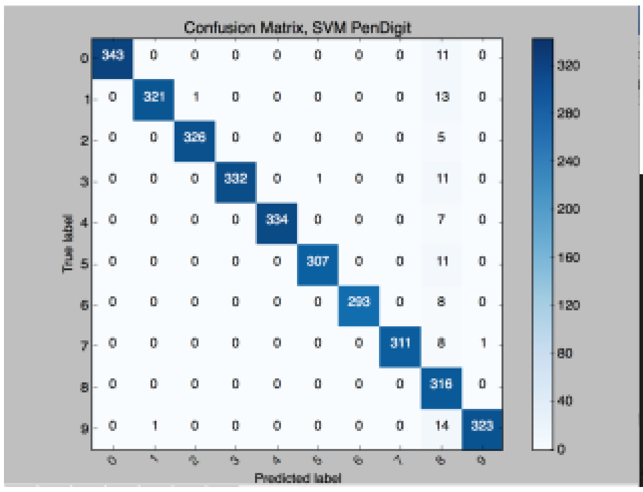


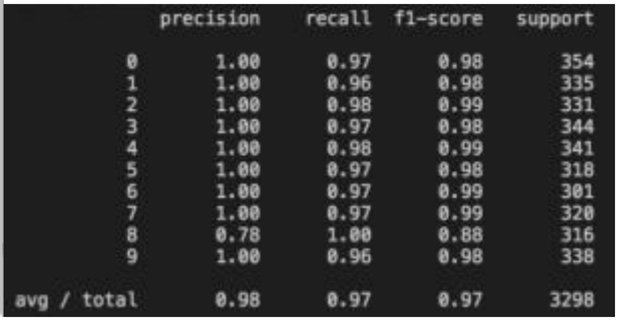
Accuracy: 86.93%

The first thing I looked at was performance of SVM's on the wine quality dataset. Now that the problem has been turned into a binary classification problem, the accuracy is much higher than before, coming in at almost 87% accurate, and mean square error was around .1306. The learning curve shows that adding more training samples would help the model generalize well with the dataset. Another thing to note is that the gap between validation score and training score has decreased, which shows variance is decreasing due to reduced model complexity (two-class classification compared to seven-class) when compared to the learning curve in boosting. One potential experiment worth looking at with SVM's is to play around with the class_weight and sample_weight parameters – which are used to better handle unbalanced datasets.

For the pen digit dataset, SVM works by using a one vs. one scheme to handle multiclass classification, meaning a series of SVMs for each pair of digits are created, quadratic optimization is used is to determine class probabilities, and the digit with the highest probability is chosen as the prediction.

Daniel Kurniawan
GTID: 903043201
CS4641 HW1
9/20/17





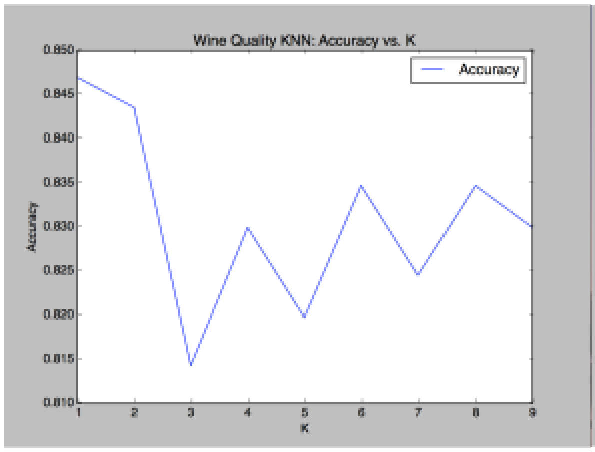
```
             precision    recall  f1-score   support

          0       1.00      0.97      0.98       354
          1       1.00      0.96      0.98       335
          2       1.00      0.98      0.99       331
          3       1.00      0.97      0.98       344
          4       1.00      0.98      0.99       341
          5       1.00      0.97      0.98       318
          6       1.00      0.97      0.99       301
          7       1.00      0.97      0.99       320
          8       0.78      1.00      0.88       316
          9       1.00      0.96      0.98       338

avg / total       0.98      0.97      0.97      3298
```

Accuracy: 97.21%

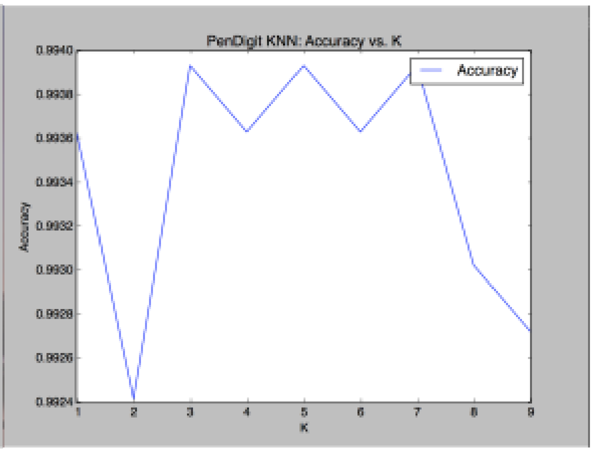
The reported accuracy of support vector machines on the pen digit dataset is very high, and the classification report shows high scores and equal representation of all classes in the test set.

kNN

The last algorithm I performed against both datasets was the K-Nearest-Neighbor. The first metric I wanted to look at was accuracy, especially to check how well kNN performed in the newly transformed wine quality binary classification problem. Turns out, the performance is slightly worse than SVMs, with highest accuracy coming in when k=1. In general, k-NN assumes locality and smoothness. If the data doesn't support this (the bias we are making in using k-NN is not appropriate) then lower k will be better. This is seen in the wine quality accuracy vs. K graph on the left below.



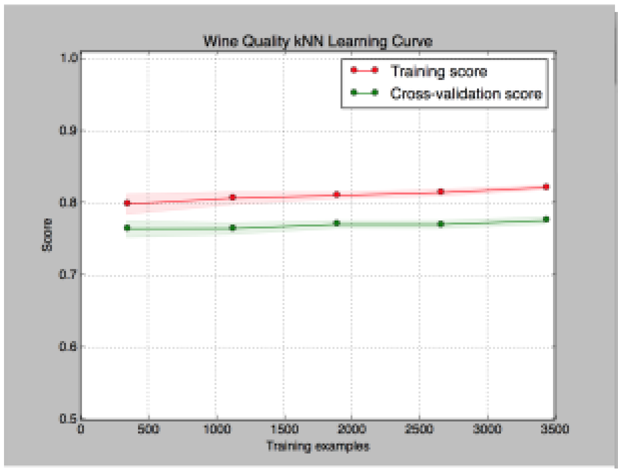
Accuracy: 84.69% (k=1)

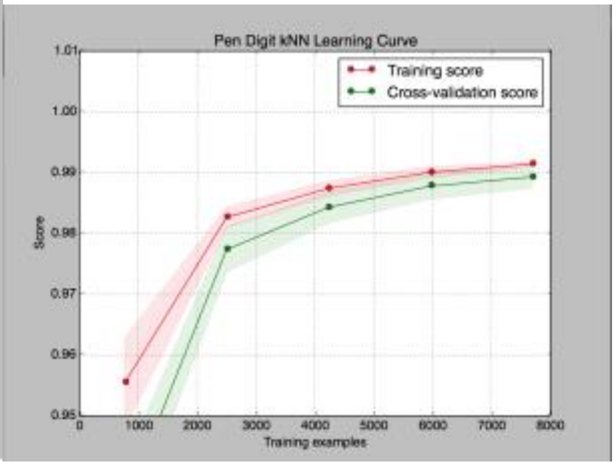


Accuracy: 99.39% (k=3)

Daniel Kurniawan
GTID: 903043201
CS4641 HW1
9/20/17

On the flip side, for Pen Digits k-NN performed extremely well, coming in with greater than 99% accuracy for all values of k=1-9. The highest accuracy was achieved when k=3,5, and 7 at a value of 99.39%. Given that the dataset has already achieved such high accuracy, there isn't much marginal improvement in increasing the value of k.





I wanted to run one more experiment for k-NN and that's how the accuracy responds depending on the number of training examples. My hypothesis was that the more data you have, the higher your accuracy will be. For both sets I was pleased with the relatively low variance indicated by the small gaps between the training and validation curves, and since training score does not peak at 100% as seen in previous gaps it tells me that the classifier is not severely overfitting the data.

Conclusion

Wine Quality – Performing multi-class classification algorithms on this dataset can prove to be extremely challenging, due to poor data quality (mainly representativeness). However, transforming the dataset to use a threshold above which a wine is labeled 'good' or 'bad' makes this problem much more manageable for a model to accurately predict whether a wine is good or bad. After running pruned decision trees, AdaBoost, and Neural networks again on the binary classification problem (wine quality) I achieved accuracy metrics of 81.63%, 86.7%, and 81.29% respectively. Once the problem was turned into binary classification, the issues of high variance were greatly reduced, which makes sense as model complexity is reduced by turning the problem into a two-class classification rather than multi-class. SVMs had the highest performance for this dataset, with an 86.93% accuracy.

Pen Digits – Classification on this dataset performed extremely well despite the nature being a multi-class classification problem. Classification performance seemed to be generally above

Daniel Kurniawan
GTID: 903043201
CS4641 HW1
9/20/17

93% accuracy, with k-NN being the best performer at 99.39%, and low variance in training and validation scores (also less likely to overfit the data). I can see that the performance on this dataset is high compared to the wine quality dataset since the pen digit set is much more balanced, with many and equal instances of each class being represented in the dataset. If anything, this assignment taught me how important data quality is when running machine learning algorithms, as it is much tougher to work with an unbalanced dataset compared to a balanced one.