Assignment 1 Analysis **gth836x; 901929700**
Myles Lefkovitz **mlefkovitz@gatech.edu**

I. The Data:

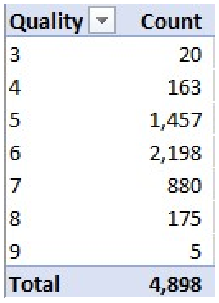
Dataset 1 - Wine Quality data:

As anyone will tell you, wine's quality is subject to one's taste. Despite that, the University of California Irvine's Machine Learning Repository hosts a Wine Quality Data Set[i] with around 5,000 wines, rated on a 1-10 scale. Each wine has 11 (presumably measured) real-valued attributes.

The 11 attributes and their minimum and maximum values are: fixed acidity [3.8 – 14.2], volatile acidity [0.08 – 1.1], citric acid [0 – 1.66], residual sugar [0.6 – 65.8], chlorides [0.009 – 0.346], free sulfur dioxide [2 – 289], total sulfer dioxide [9 – 440], denisty [0.98 – 1.04], pH [2.72 – 3.82], sulphates [0.22 – 1.08], and alcohol [8 – 14.2]. These numbers may mean something to you if you're interested in wine.

The attributes are all real-valued, therefore they can all be normalized/scaled. We don't need to review them here.

As noted by the data set contributor: the classes are not balanced. There are much more 'normal' wines than great or poor ones. Since quality was rated by humans, there no wines received ratings of 1, 2, or 10. You can see a breakdown of the quantities for each quality rating in *Figure 1* & *Figure 2* to the right.



Quality	Count
3	20
4	163
5	1,457
6	2,198
7	880
8	175
9	5
Total	**4,898**

Figure 1. Wine Quality Chart

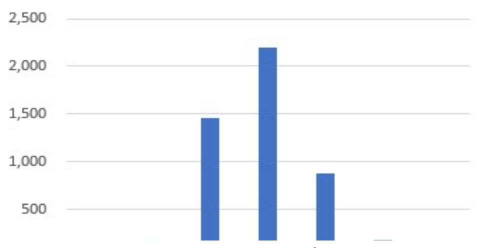


Figure 2. Wine Quality Histogram

If the data set's 7 categories were balanced, we would expect chance to correctly classify a wine 1/7th or around 14% of the time. By reviewing the data, we can see that about 45% of the samples are categorized in the 6/10 quality category. To create a high-performing simplistic model, we could uniformly classify every wine as a 6/10. We should consider a learner successful only if it categorizes wines correctly more than 45% of the time.

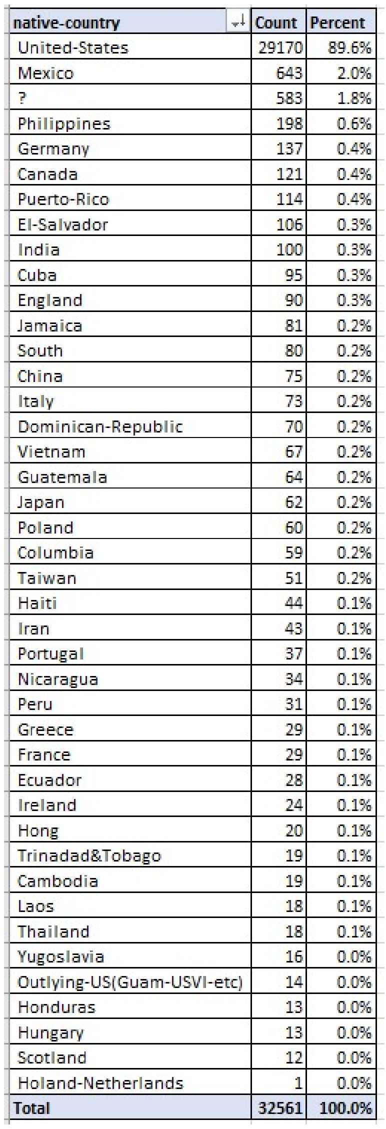
Dataset 2 – Adult Income data:

For my second data set I searched for a classification problem that differed from the wine problem above, which had real-valued factors and many classes. On the University of California Irvine' Machine Learning Repository I found an Adult Income Data Set[ii] that classifies adults into one of two income categories: '>50K', or '<=50K'. The '>50K' category identifies individuals that earned more than $50,000 in the given year, 1994. The '<=50K' category identifies individuals that earned less than or equal to $50,000. $50,000 in 1994 is approximately $81,000 in today's terms. The data has 13 attributes, 5 of which are real-valued, and 8 of which are categorical.

The 5 real-valued attributes and their minimum and maximum values are: age [17 – 90], education-num [1 – 16], capital-gain [0 – 99,999], capital-loss [0 – 4,356], hours-per-week [1 – 99].

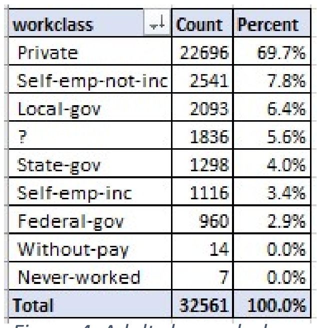
The 8 categorical attributes and a brief description are: workclass (type of employer), education (level of education, duplicated in the education-num real-valued attribute), marital-status, occupation (type of job), relationship (family status), race, sex, native-country. See figures 3-10 below for an examination of these 8 attributes.

The data set's 2 categories, '>50K' and '<=50K', represent approximately 24% and 76% of the instances in the data set, respectively. To create a high-performing simplistic model, we could uniformly classify every individual in the '<=50K' category. We should consider a learner successful only if it categorizes adult incomes correctly more than 76% of the time.



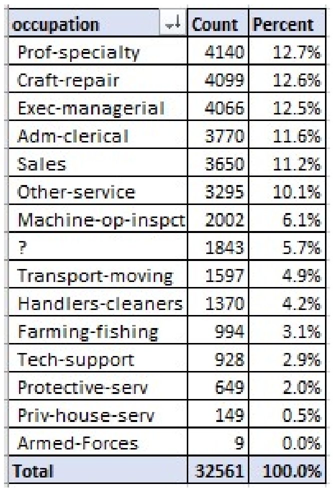
native-country	Count	Percent
United-States	29170	89.6%
Mexico	643	2.0%
?	583	1.8%
Philippines	198	0.6%
Germany	137	0.4%
Canada	121	0.4%
Puerto-Rico	114	0.4%
El-Salvador	106	0.3%
India	100	0.3%
Cuba	95	0.3%
England	90	0.3%
Jamaica	81	0.2%
South	80	0.2%
China	75	0.2%
Italy	73	0.2%
Dominican-Republic	70	0.2%
Vietnam	67	0.2%
Guatemala	64	0.2%
Japan	62	0.2%
Poland	60	0.2%
Columbia	59	0.2%
Taiwan	51	0.2%
Haiti	44	0.1%
Iran	43	0.1%
Portugal	37	0.1%
Nicaragua	34	0.1%
Peru	31	0.1%
Greece	29	0.1%
France	29	0.1%
Ecuador	28	0.1%
Ireland	24	0.1%
Hong	20	0.1%
Trinadad&Tobago	19	0.1%
Cambodia	19	0.1%
Laos	18	0.1%
Thailand	18	0.1%
Yugoslavia	16	0.0%
Outlying-US(Guam-USVI-etc)	14	0.0%
Honduras	13	0.0%
Hungary	13	0.0%
Scotland	12	0.0%
Holand-Netherlands	1	0.0%
Total	32561	100.0%

Figure 3. Adults by Native Country



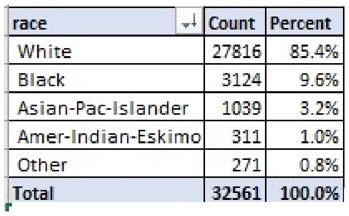
workclass	Count	Percent
Private	22696	69.7%
Self-emp-not-inc	2541	7.8%
Local-gov	2093	6.4%
?	1836	5.6%
State-gov	1298	4.0%
Self-emp-inc	1116	3.4%
Federal-gov	960	2.9%
Without-pay	14	0.0%
Never-worked	7	0.0%
Total	32561	100.0%

Figure 4. Adults by work class



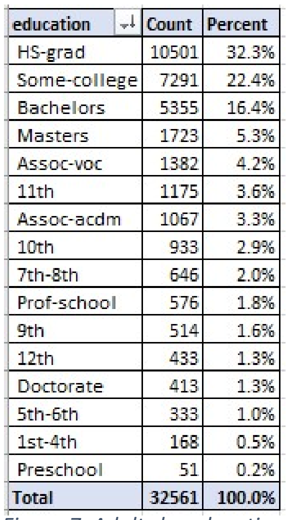
occupation	Count	Percent
Prof-specialty	4140	12.7%
Craft-repair	4099	12.6%
Exec-managerial	4066	12.5%
Adm-clerical	3770	11.6%
Sales	3650	11.2%
Other-service	3295	10.1%
Machine-op-inspct	2002	6.1%
?	1843	5.7%
Transport-moving	1597	4.9%
Handlers-cleaners	1370	4.2%
Farming-fishing	994	3.1%
Tech-support	928	2.9%
Protective-serv	649	2.0%
Priv-house-serv	149	0.5%
Armed-Forces	9	0.0%
Total	32561	100.0%

Figure 5. Adults by occupation



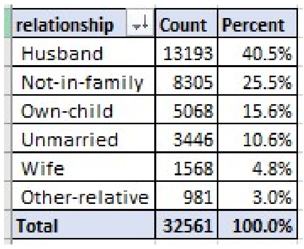
race	Count	Percent
White	27816	85.4%
Black	3124	9.6%
Asian-Pac-Islander	1039	3.2%
Amer-Indian-Eskimo	311	1.0%
Other	271	0.8%
Total	32561	100.0%

Figure 6. Adults by race



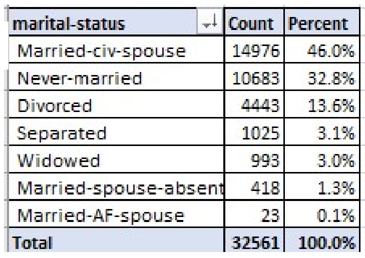
education	Count	Percent
HS-grad	10501	32.3%
Some-college	7291	22.4%
Bachelors	5355	16.4%
Masters	1723	5.3%
Assoc-voc	1382	4.2%
11th	1175	3.6%
Assoc-acdm	1067	3.3%
10th	933	2.9%
7th-8th	646	2.0%
Prof-school	576	1.8%
9th	514	1.6%
12th	433	1.3%
Doctorate	413	1.3%
5th-6th	333	1.0%
1st-4th	168	0.5%
Preschool	51	0.2%
Total	32561	100.0%

Figure 7. Adults by education



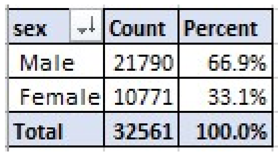
relationship	Count	Percent
Husband	13193	40.5%
Not-in-family	8305	25.5%
Own-child	5068	15.6%
Unmarried	3446	10.6%
Wife	1568	4.8%
Other-relative	981	3.0%
Total	32561	100.0%

Figure 8. Adults by relationship



marital-status	Count	Percent
Married-civ-spouse	14976	46.0%
Never-married	10683	32.8%
Divorced	4443	13.6%
Separated	1025	3.1%
Widowed	993	3.0%
Married-spouse-absent	418	1.3%
Married-AF-spouse	23	0.1%
Total	32561	100.0%

Figure 9. Adults by marital status



sex	Count	Percent
Male	21790	66.9%
Female	10771	33.1%
Total	32561	100.0%

Figure 10. Adults by sex

II. The Learners

For both data sets I followed a similar procedure:

1. Load the data and clean any missing parts
2. Find a good algorithm that implemented the methods described
3. Identify popular hyperparameters for each learner
4. Run automated cross validation (generally 3 or 5-fold) to identify the best hyperparameters
5. Graph the results and review, iterate if necessary by adding hyperparameters

When evaluating a model's performance, we should consider: the score against training data, the mean score against cross validated data, the score against a held-out test set, we should review the learning curve (the aforementioned scores plotted against the number of training examples) for bias/variance, and we should consider the model's training time.

Dataset 1 – Wine

1. Decision trees – pruned

As a first step, I searched for an algorithm that would create a decision tree. Jonathan Tay posted his own algorithm on the Piazza classroom forum. I found it easy to implement so I stole it. The hyperparameter that seems most meaningful in this algorithm is 'alpha', which controls the pruning rate.

I developed a list of possible alphas on a logarithmic scale, and iterated until I was confident that the list was inclusive of all reasonable values. Cross validation settled on alpha=0.06 as the best hyperparameter. See the model complexity chart in figure 11 below.

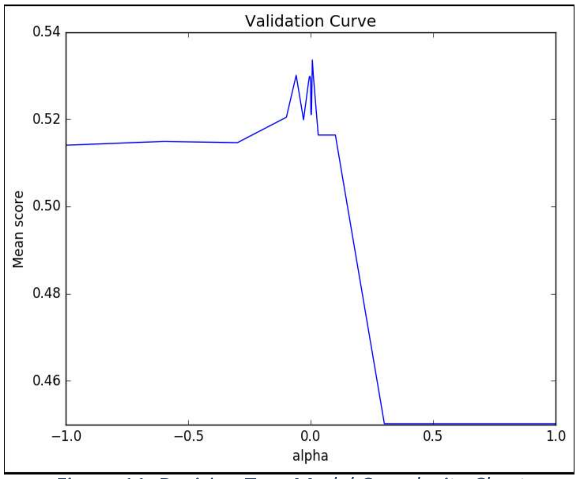


Figure 11. Decision Tree Model Complexity Chart

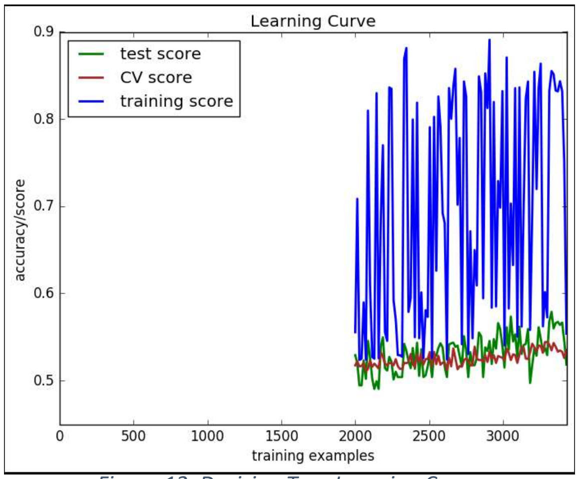


Figure 12. Decision Tree Learning Curve

When evaluating a model's performance, we should consider: the score against training data, the mean score against cross validated data, the score against a held-out test set, we should review the learning curve (the aforementioned scores plotted against the number of training examples) for bias/variance, and we should consider the model's training time.

This model's performance against training data is 57%
This model's performance against CV data is 53.3%
This model's performance against test data is 51.1%
This model took approximately 15 minutes to run

By reviewing the learning curve in figure 12 above, we can see that the model exhibits high bias. We can tell, because though the training score jumps wildly with different training sizes, the lows are consistent and fairly close to the test score.

Knowing that the model shows high bias, we can tell that adding more training examples, or simplifying the model won't improve it. This makes conceptual sense, since we determined the pruning factor 'alpha' that minimized CV error. The pruning factor alpha is an attempt to regularize or reduce the impact of additional features. Since this model exhibits high bias, if we wanted to improve it we would need to search for more features for the existing instances. For example, we could see if price was a strong indicator of quality.

2. Neural Networks

As a first step to implementing a neural network, I searched for an algorithm. Scikit-learn has a popular algorithm. I found it easy to implement so I stole it. The hyperparameters that seem most meaningful in neural networks are 'alpha', which controls the learning rate and the hidden layer size, which controls the number of nodes in the hidden layer. I opted for a simple model that would avoid overfitting, so I only evaluated a single hidden layer.

I developed a list of possible alphas on a logarithmic scale, and iterated until I was confident that the list was inclusive of all reasonable values of alpha. I also developed a list of possible hidden layer sizes, throwing random numbers at the algorithm. Cross validation settled on alpha=0.001 and hidden layer size = 12 as the best set of hyperparameters. See the model complexity chart in figure 13 below.

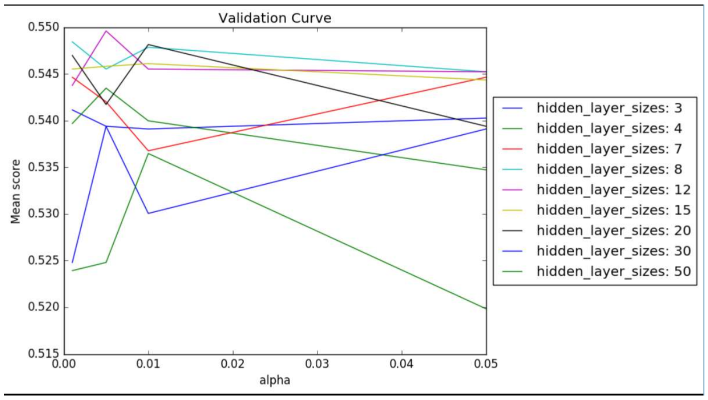


Figure 13. Neural Network Model Complexity Chart

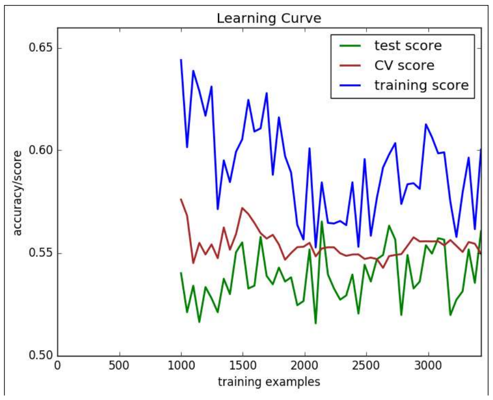


Figure 14. Neural Network Learning Curve

When evaluating a model's performance, we should consider: the score against training data, the mean score against cross validated data, the score against a held-out test set, we should review the learning curve (the aforementioned scores plotted against the number of training examples) for bias/variance, and we should consider the model's training time.

This model's performance against training data is 60.7%
This model's performance against CV data is 55.9%
This model's performance against test data is 53.4%
This model took approximately 51 minutes to run

By reviewing the learning curve in figure 14 above, we can see that the model exhibits high bias. We can tell, because though the training score jumps wildly with different training sizes, the lows are consistent and fairly close to the test score.

Knowing that the model shows high bias, we can tell that adding more training examples, or simplifying the model won't improve it. Since this model exhibits high bias, if we wanted to improve it we would need to search for more features for the existing instances. For example, we could see if price was a strong indicator of quality.

3. Boosted Decision Trees

As a first step to implementing a boosted decision tree, I searched for an algorithm. Scikit-learn has a popular algorithm. I found it easy to implement so I stole it. I also had to select a weak learner. I stuck with the same decision tree algorithm I used in my decision tree analysis. The hyperparameters that seem most meaningful in boosted trees are 'alpha', which controls the pruning rate, the learning rate, which controls how rapidly weak-learners are applied, and the number of estimators.

I developed a list of possible learning rates on a logarithmic scale, and iterated until I was confident that the list was inclusive of all reasonable values. I also developed a list of possible numbers of estimators, throwing random numbers at the algorithm. Cross validation settled on alpha=0.06 (the result from our weak learner) and learning rate = 0.05 and number of estimators = 150 as the best set of hyperparameters. See the model complexity chart in figure 15 below.

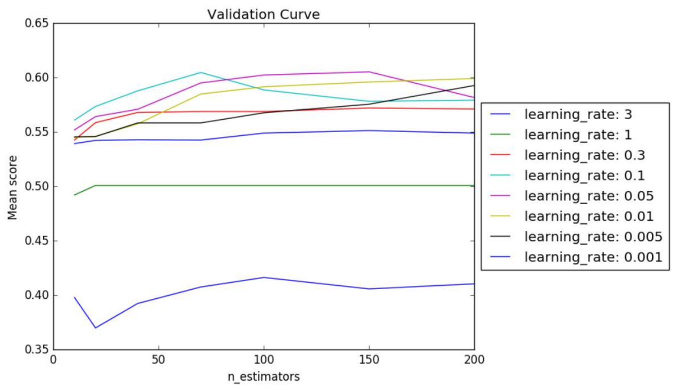


Figure 15. Boosting Model Complexity Chart

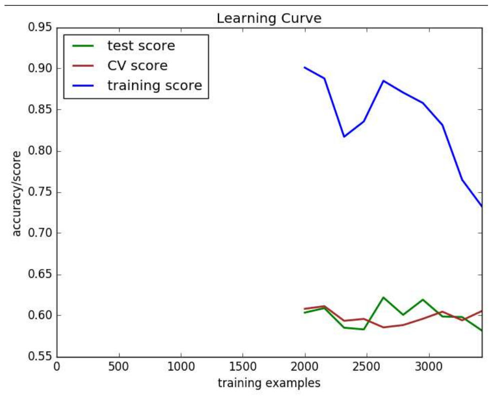


Figure 16. Boosting Learning Curve

When evaluating a model's performance, we should consider: the score against training data, the mean score against cross validated data, the score against a held-out test set, we should review the learning curve (the aforementioned scores plotted against the number of training examples) for bias/variance, and we should consider the model's training time.

This model's performance against training data is 73.2%
This model's performance against CV data is 60.5%
This model's performance against test data is 58.2%
This model took approximately 5 hours and 16 minutes to run

By reviewing the learning curve in figure 16 above, we can see that the model exhibits high variance. We can tell, because though the training score jumps wildly with different training sizes, the lows are fairly far from the test scores.

Knowing that the model shows high variance, we can tell that adding more training examples, or simplifying the model might improve it. Since this model exhibits high variance, if we wanted to improve it we would need to search for more training examples. For example, we could solicit another survey, or just buy a bunch of wine, drink it, and rate it ourselves.

4. Support Vector Machines

As a first step to implementing a support vector machine, I searched for an algorithm. Scikit-learn has a popular algorithm. I found it easy to implement so I stole it. The hyperparameters that seem most meaningful in SVMs are the 'kernel', the 'gamma', and the 'C'.

I developed a list of possible hyperparameters, and iterated until I was confident that the list was inclusive of all reasonable values. Cross validation settled on the RBF kernel, gamma = 0.5, and C = 10 as the best set of hyperparameters. See the model complexity chart in figure 17 below.

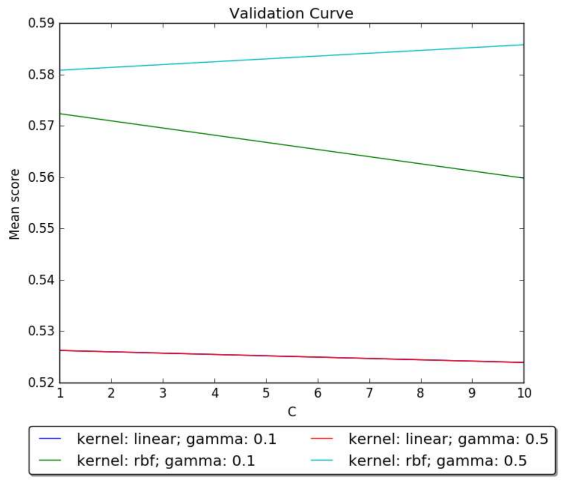


Figure 17. SVM Complexity Chart

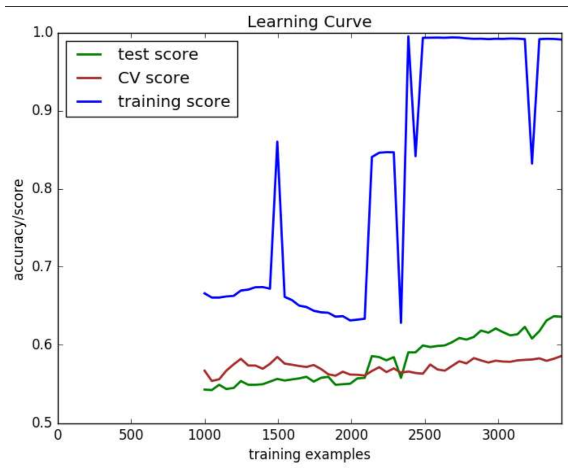


Figure 18. SVM Learning Curve

When evaluating a model's performance, we should consider: the score against training data, the mean score against cross validated data, the score against a held-out test set, we should review the learning curve (the aforementioned scores plotted against the number of training examples) for bias/variance, and we should consider the model's training time.

This model's performance against training data is 99.1%
This model's performance against CV data is 58.6%
This model's performance against test data is 63.6%
This model took approximately 9 minutes to run

By reviewing the learning curve in figure 18 above, we can see that the model exhibits high variance. We can tell, because though the training score jumps wildly with different training sizes, the lows are fairly far from the test score.

Knowing that the model shows high variance, we can tell that adding more training examples, or simplifying the model might improve it. Since this model exhibits high variance, if we wanted to improve it we would need to search for more training examples. For example, we could solicit another survey, or just buy a bunch of wine, drink it, and rate it ourselves. Additionally, with training scores above 99% it's clear that the model is overfitting the data. There's likely room to remove some of the attributes that add only noise, or increase regularization to reduce overfitting.

5. k-nearest neighbors

As a first step to implementing k-nearest neighbors, I searched for an algorithm. Scikit-learn has a popular algorithm. I found it easy to implement so I stole it. The hyperparameter that seems most meaningful in kNN is the number of neighbors.

I developed a list of possible neighbor sizes. Cross validation settled on number of neighbors = 1 as the best hyperparameter. See the model complexity chart in figure 19 below.

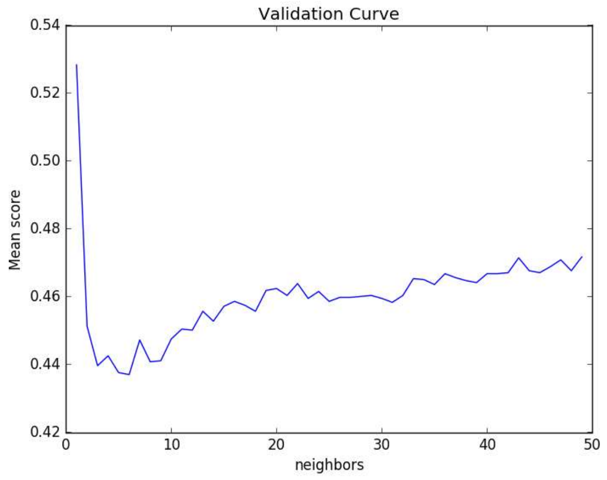


Figure 19. SVM Complexity Chart

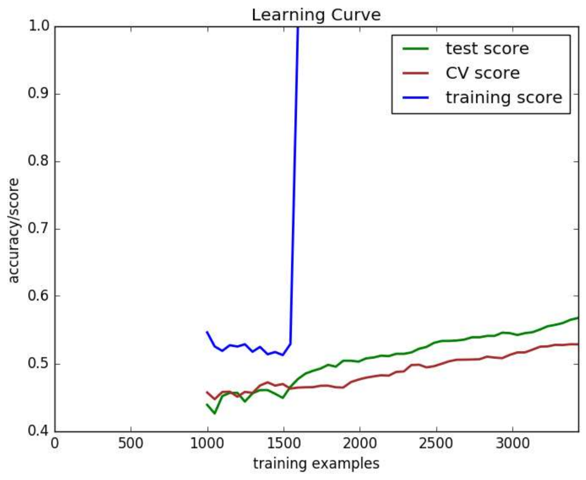


Figure 20. SVM Learning Curve

When evaluating a model's performance, we should consider: the score against training data, the mean score against cross validated data, the score against a held-out test set, we should review the learning curve (the aforementioned scores plotted against the number of training examples) for bias/variance, and we should consider the model's training time.

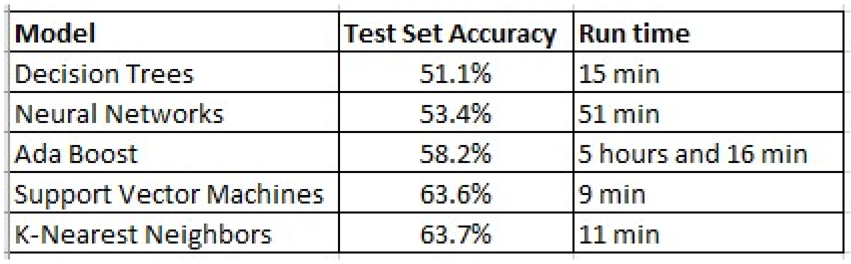
This model's performance against training data is 100%
This model's performance against CV data is 57.7%
This model's performance against test data is 63.7%
This model took approximately 11 minutes to run

By reviewing the learning curve in figure 20 above, we can see that the model exhibits high variance. We can tell, because the training set scores are very far from the test scores.

Knowing that the model shows high variance, we can tell that adding more training examples, or simplifying the model might improve it. Since this model exhibits high variance, if we wanted to improve it we would need to search for more training examples. For example, we could solicit another survey, or just buy a bunch of wine, drink it, and rate it ourselves. Additionally, with training scores above 99% it's clear that the model is overfitting the data. There's likely room to remove some of the attributes that add only noise, or increase regularization to reduce overfitting.

Summary

As a performance summary against Wine Quality data, the figure below shows the performance of each model:



Model	Test Set Accuracy	Run time
Decision Trees	51.1%	15 min
Neural Networks	53.4%	51 min
Ada Boost	58.2%	5 hours and 16 min
Support Vector Machines	63.6%	9 min
K-Nearest Neighbors	63.7%	11 min

While all of the models performed better than the 45% accuracy we would expect from a simplistic model that picks the most popular class for every instance, some fared much better than others.

It's clear that the kNN and SVM models performed much better, much more quickly, and due to their high variance, they have room for improvement with the addition of more data, the reduction of features, or regularization.

Dataset 2 – Income

1. Decision trees – pruned

As a first step, I searched for an algorithm that would create a decision tree. Jonathan Tay posted his own algorithm on the Piazza classroom forum. I found it easy to implement so I stole it. The hyperparameter that seems most meaningful in this algorithm is 'alpha', which controls the pruning rate.

I developed a list of possible alphas on a logarithmic scale, and iterated until I was confident that the list was inclusive of all reasonable values. Cross validation settled on alpha=0.06 as the best hyperparameter. See the model complexity chart in figure 21 below.

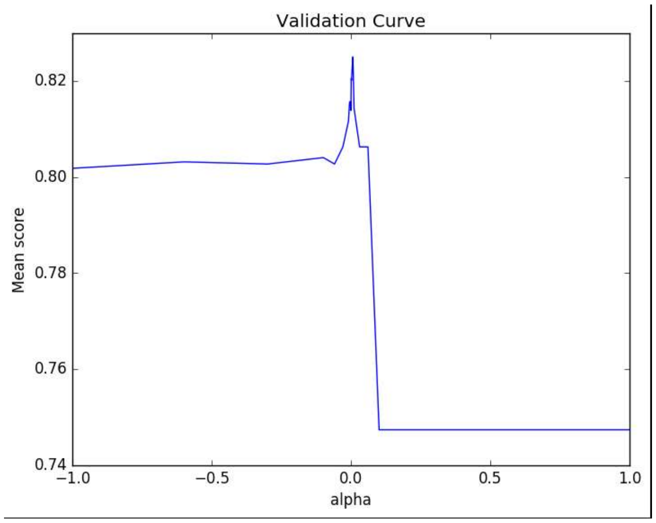


Figure 21. Decision Tree Model Complexity Chart

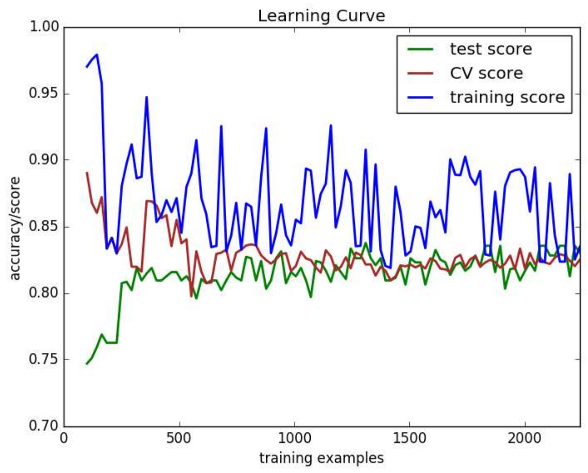


Figure 22. Decision Tree Learning Curve

When evaluating a model's performance, we should consider: the score against training data, the mean score against cross validated data, the score against a held-out test set, we should review the learning curve (the aforementioned scores plotted against the number of training examples) for bias/variance, and we should consider the model's training time.

This model's performance against training data is 83.5%
This model's performance against CV data is 82.5%
This model's performance against test data is 83%
This model took approximately 4 minutes to run

By reviewing the learning curve in figure 22 above, we can see that the model exhibits high bias. We can tell, because though the training score jumps wildly with different training sizes, the lows are consistent and fairly close to the test score.

Knowing that the model shows high bias, we can tell that adding more training examples, or simplifying the model won't improve it. This makes conceptual sense, since we determined the pruning factor 'alpha' that minimized CV error. The pruning factor alpha is an attempt to regularize or reduce the impact of additional features. Since this model exhibits high bias, if we wanted to improve it we would need to search for more features for the existing instances. For example, we could see if region within the US was a strong indicator of income.

2. Neural Networks

As a first step to implementing a neural network, I searched for an algorithm. Scikit-learn has a popular algorithm. I found it easy to implement so I stole it. The hyperparameters that seem most meaningful in neural networks are 'alpha', which controls the learning rate and the hidden layer size, which controls the number of nodes in the hidden layer. I opted for a simple model that would avoid overfitting, so I only evaluated a single hidden layer.

I developed a list of possible alphas on a logarithmic scale, and iterated until I was confident that the list was inclusive of all reasonable values of alpha. I also developed a list of possible hidden layer sizes, throwing random numbers at the algorithm. Cross validation settled on alpha=0.005 and hidden layer size = 3 as the best set of hyperparameters. See the model complexity chart in figure 23 below.

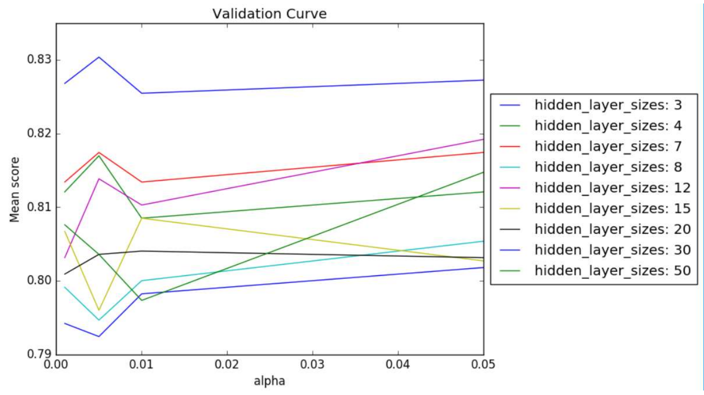


Figure 23. Neural Network Model Complexity Chart

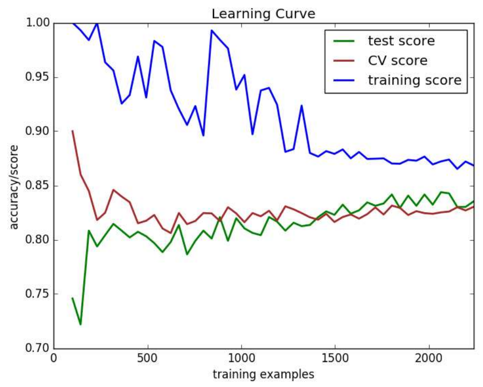


Figure 24. Neural Network Learning Curve

When evaluating a model's performance, we should consider: the score against training data, the mean score against cross validated data, the score against a held-out test set, we should review the learning curve (the aforementioned scores plotted against the number of training examples) for bias/variance, and we should consider the model's training time.

This model's performance against training data is 86.8%
This model's performance against CV data is 83%
This model's performance against test data is 83.5%
This model took approximately 49 minutes to run

By reviewing the learning curve in figure 24 above, we can see that the model exhibits high variance. We can tell, because though the training score jumps wildly with different training sizes, the lows are fairly far from the test score.

Knowing that the model shows high variance, we can tell that adding more training examples, or simplifying the model might improve it. Since this model exhibits high variance, if we wanted to improve it we would need to search for more training examples.

3. Boosted Decision Trees

As a first step to implementing a boosted decision tree, I searched for an algorithm. Scikit-learn has a popular algorithm. I found it easy to implement so I stole it. I also had to select a weak learner. I stuck with the same decision tree algorithm I used in my decision tree analysis. The hyperparameters that seem most meaningful in boosted trees are ‘alpha’, which controls the pruning rate, the learning rate, which controls how rapidly weak-learners are applied, and the number of estimators.

I developed a list of possible learning rates on a logarithmic scale, and iterated until I was confident that the list was inclusive of all reasonable values. I also developed a list of possible numbers of estimators, throwing random numbers at the algorithm. Cross validation settled on alpha=0.06 (the result from our weak learner) and learning rate = 0.01 and number of estimators = 150 as the best set of hyperparameters. See the model complexity chart in figure 25 below.

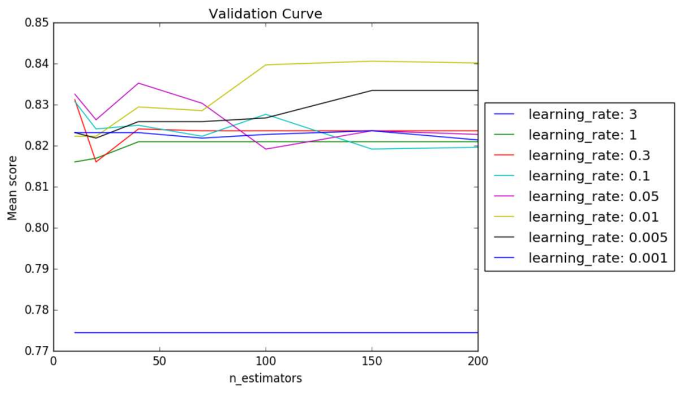


Figure 25. Boosting Model Complexity Chart

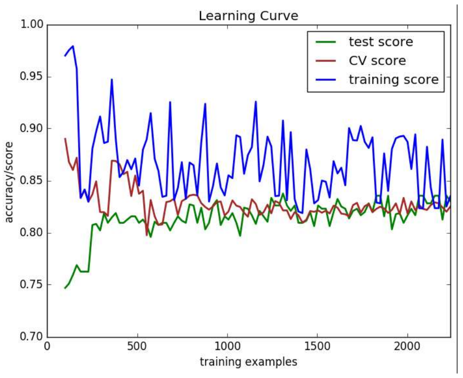


Figure 26. Boosting Learning Curve

When evaluating a model’s performance, we should consider: the score against training data, the mean score against cross validated data, the score against a held-out test set, we should review the learning curve (the aforementioned scores plotted against the number of training examples) for bias/variance, and we should consider the model’s training time.

This model’s performance against training data is 86.9%
This model’s performance against CV data is 84%
This model’s performance against test data is 83.3%
This model took approximately 2 hours to run

By reviewing the learning curve in figure 26 above, we can see that the model exhibits high variance. We can tell, because though the training score jumps wildly with different training sizes, the lows are fairly far from the test score.

Knowing that the model shows high variance, we can tell that adding more training examples, or simplifying the model might improve it. Since this model exhibits high variance, if we wanted to improve it we would need to search for more training examples.

4. Support Vector Machines

As a first step to implementing a support vector machine, I searched for an algorithm. Scikit-learn has a popular algorithm. I found it easy to implement so I stole it. The hyperparameters that seem most meaningful in SVMs are the ‘kernel’, the ‘gamma’, and the ‘C’.

I developed a list of possible hyperparameters, and iterated until I was confident that the list was inclusive of all reasonable values. Cross validation settled on the linear kernel, gamma = 0.1, and C = 10 as the best set of hyperparameters. See the model complexity chart in figure 27 below.

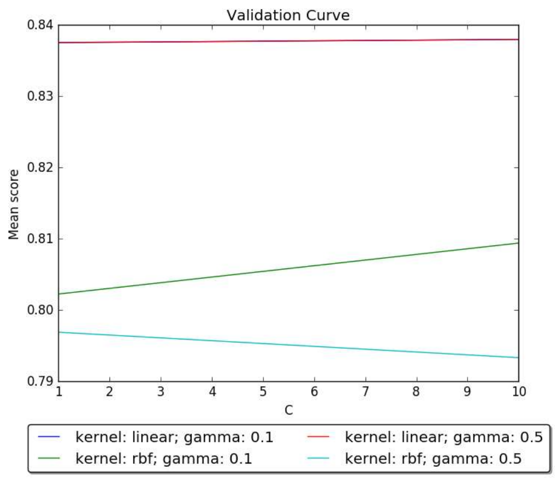


Figure 27. SVM Complexity Chart

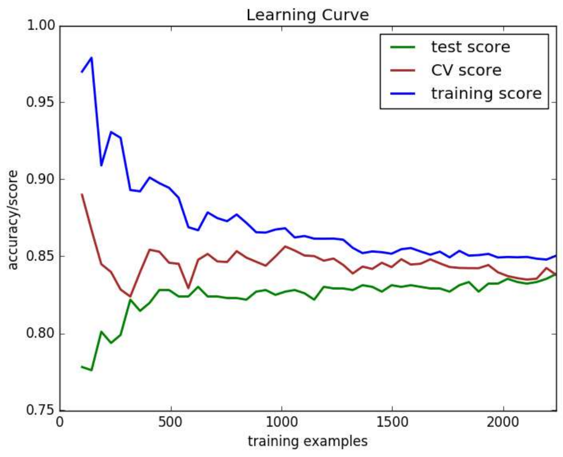


Figure 28. SVM Learning Curve

When evaluating a model's performance, we should consider: the score against training data, the mean score against cross validated data, the score against a held-out test set, we should review the learning curve (the aforementioned scores plotted against the number of training examples) for bias/variance, and we should consider the model's training time.

This model's performance against training data is 85.5%
This model's performance against CV data is 83.8%
This model's performance against test data is 83.9%
This model took approximately 5 minutes to run

By reviewing the learning curve in figure 28 above, we can see that the model exhibits high bias. We can tell, because it is consistent and fairly close to the test score.

Knowing that the model shows high bias, we can tell that adding more training examples, or simplifying the model won't improve it. This makes conceptual sense, since we determined the pruning factor 'alpha' that minimized CV error. The pruning factor alpha is an attempt to regularize or reduce the impact of additional features. Since this model exhibits high bias, if we wanted to improve it we would need to search for more features for the existing instances. For example, we could see if region within the US was a strong indicator of income.

5. k-nearest neighbors

As a first step to implementing k-nearest neighbors, I searched for an algorithm. Scikit-learn has a popular algorithm. I found it easy to implement so I stole it. The hyperparameter that seems most meaningful in kNN is the number of neighbors.

I developed a list of possible neighbor sizes. Cross validation settled on number of neighbors = 10 as the best hyperparameter. See the model complexity chart in figure 29 below.

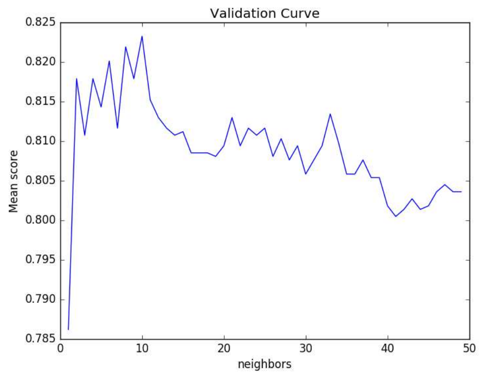


Figure 29. SVM Complexity Chart

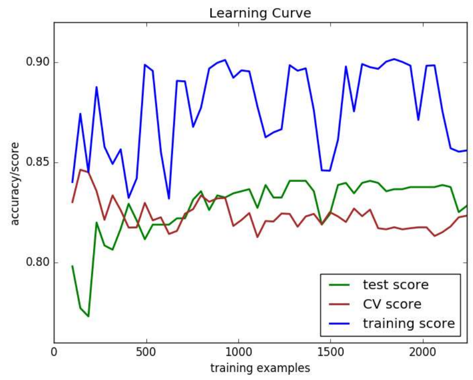


Figure 30. SVM Learning Curve

When evaluating a model's performance, we should consider: the score against training data, the mean score against cross validated data, the score against a held-out test set, we should review the learning curve (the aforementioned scores plotted against the number of training examples) for bias/variance, and we should consider the model's training time.

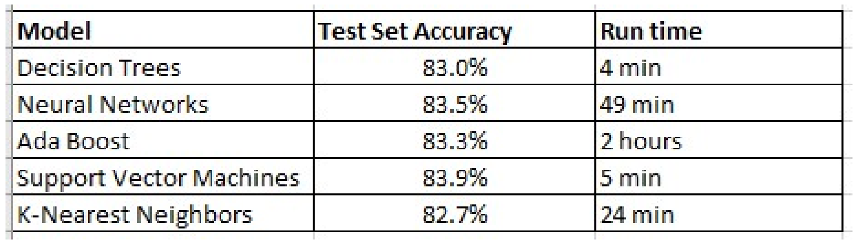
This model's performance against training data is 82.2%
This model's performance against CV data is 82.3%
This model's performance against test data is 82.7%
This model took approximately 24 minutes to run

By reviewing the learning curve in figure 30 above, we can see that the model exhibits high variance. We can tell, because the training set is far from the test score.

Knowing that the model shows high variance, we can tell that adding more training examples, or simplifying the model might improve it. Since this model exhibits high variance, if we wanted to improve it we would need to search for more training examples. For example, we could solicit another survey, or just buy a bunch of wine, drink it, and rate it ourselves. Additionally, with training scores above 99% it's clear that the model is overfitting the data. There's likely room to remove some of the attributes that add only noise, or increase regularization to reduce overfitting.

Summary

As a performance summary against Adult Income data, the figure below shows the performance of each model:



Model	Test Set Accuracy	Run time
Decision Trees	83.0%	4 min
Neural Networks	83.5%	49 min
Ada Boost	83.3%	2 hours
Support Vector Machines	83.9%	5 min
K-Nearest Neighbors	82.7%	24 min

All of the models were similarly accurate and all of the models performed better than the 76% accuracy we would expect from a simplistic model that picks the most popular class for every instance. Since accuracy isn't much of a consideration, I'd select SVM next time, since it produced the highest accuracy in a relatively short time (only one model was quicker, and decision trees were less accurate).

All of the models are good choices, but the SVM model seems like the best choice as it works fairly quickly, and due to the high variance, has room for improvement with the addition of more data, the reduction of features, or regularization.

III. Final Thoughts

About the problems

The Wine Quality problem is technically interesting in that the different learners responded with wildly different results and times. I think I was really lucky to select this data as my first data set. I think there's some room to improve the accuracy of the top models (SVM and KNN) with more thorough analysis into the overfitting problem. It's likely that no function that generalizes exceptionally well exists, given the subjective nature of the data (personal taste determines the quality of a wine).

The Adult Income problem was a great second problem. With fewer classes and a lot of categorical data to wrangle, it's very interesting to me that the learners all performed so similarly.

About the algorithms

In comparing the performance of the 5 algorithms (decision trees, neural networks, boosting, SVMs, and KNNs) it looks like SVMs should be the first algorithm I try the next time I need to attack a supervised learning problem. The SVM learner performed accurately and quickly against both data sets.

---

[i] http://archive.ics.uci.edu/ml/datasets/Wine+Quality

[ii] http://archive.ics.uci.edu/ml/datasets/Adult