Assignment 1

Aydin Atay, CS 7641, Fall 2018
Georgia Institute of Technology
Computer Science Department,

Introduction

I was born in Bulgaria where almost everyone used to make their own wine. I too learned how to make home-made wine, and it is extremely difficult to make good wine.

Why is it interesting: Reproducing the same wine year after year is almost impossible, it depends on that year's temperatures, precipitation, natural events, fertilizing, when the grapes are picked and so on and so forth. I picked a wine dataset because I want to know if predicting the wine quality based on variables is possible. On the other hand, wine taste is a subjective quality which also varies from a consumer to another.

The second dataset I choose is a car quality dataset. The dataset has relatively small number of attributes, e.g. *number of doors, number of seats, maintenance cost, lug boot,* and *safety* which should give an idea of what is important for a buyer while choosing a car.

Why is it interesting: It would be quite interesting if a buyer could definitely make a selection based on only 5 factors! The choices of this kind are quite subjective based on many factors so I am suspicious having big success without having more data about users' backgrounds. In other words I would expect different buyer groups to show different behavior so composite data detached from buyer groups should be less useful.

In general, I see a slight similarity between these two dataset: both are about the quality of a product based on some attributes related to its production and testing. A friend of mine is using www.brightcellars.com which uses some kind of algorithm trying to predict a customer's wine choice based on a 7 question quiz. They are asking the questions below:

1. *What chocolate type do you like*
2. *What tea type do you like*
3. *What alcoholic drink do you like*
4. *What fruit juice do you like*
5. *What do you pair with wine? fine dining, friends, beach etc*
6. *How adventurous are you with new food and drink?*
7. *Red or white or something in between?*

Datasets

Both datasets were downloaded from Weka datasets suggested on Piazza. These dataset are in original Weka .arff format and both come with a training and test datasets. The download link is below:

http://www.cs.ubc.ca/labs/beta/Projects/autoweka/datasets/

More information on Weka .arff dataset format can be found here:

https://www.cs.waikato.ac.nz/ml/weka/arff.html

Datasets and other information can also be found in

https://github.com/designer16tr/CS7641

Wine Dataset: This dataset is has 12 attributes of which the last is a class attribute. There are 3429 instances in the training set and 1469 instances in the test set . This is a reduced set of only white wines.

Car Dataset: This dataset has 6 attributes and 1 class. There 1210 instances in the training set and 518 instances in the test set.

Methodology:

Learning Curve Experiments: In Weka I used GUI to perform experiments. During experiments 10 fold cross validation was used with changing the sample size of the training set from 5% to 100%. In Weka Explorer :

a) I used Meta Classifier -> FilteredClassifier.
b) From the FilteredClassifier configuration tab I can choose the classifier and the filter.
c) As filter I use unsupervised -> instance -> Resample.
d) From Resample configuration tab I chose noReplacement = True and vary the sample size from 5% to 100%. I use an excel sheet to plot training performance % versus training data % and training time and test performance
e) Supplied test set, load the test.arff
f) Start modeling and measuring on train set,
g) after training and train run finishes, run model on test set

- I repeated the experiments for all the algorithms using the above procedure.
- Selection of algorithm was done at step b)
- For Each algorithm a training set and a separate test set were used. The training set was available in the original repository.

Model Complexity and Parameter Optimization Experiments: For each algorithm I experimented on several parameters and after I saw which parameters had the highest impact on results I conveyed more experiments and added to the report. Some parameters (e.g. kernel selection in SVM) had extremely long run times but not much impact so I didn't include the results.

Decision Tree Experiments:

Weka is using a modified version of C45 algorithm written in Java, so they call it J48. The learning curve experiments were conducted as described in "Methodology" above. For parameter optimization the Confidence Factor 'C' and Minimum Number of Objects 'M' were changed. After adjusting parameter values the model was built and tested on training set, then the model was tested on the test set.

Wine dataset

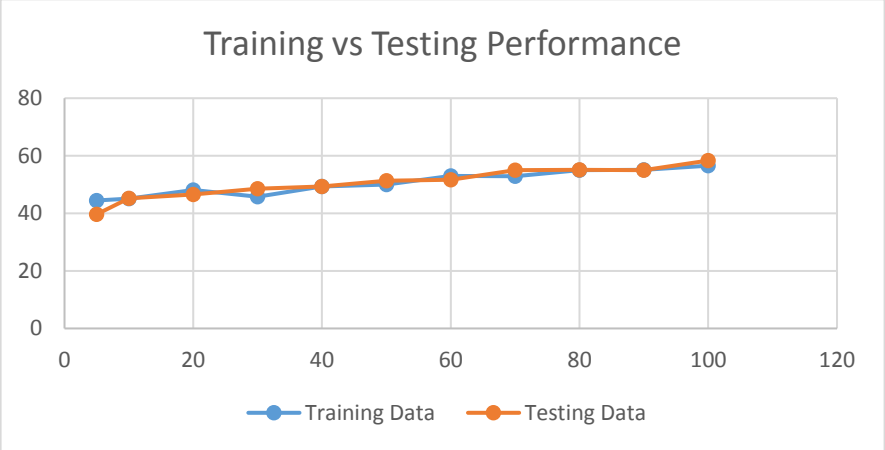

Figure 1a. Decision Tree Learning Curves

Confidence Factor: C	Minimum Number of Instances per leaf: M	Number of Leaves	Size of tree	Build Time	Training Performance (% correct)	Testing Performance (% correct)
0.1	2	388	775	0.15	55.85	56.09
0.1	3	264	527	0.11	56.05	54.53
0.2	2	477	953	0.11	56.14	58.2
0.2	3	381	761	0.12	56.13	55.68
0.3	2	531	1061	0.16	56.61	58.34
0.3	3	428	855	0.14	55.85	56.2
0.4	2	534	1067	0.12	56.58	58.34
0.4	3	440	879	0.1	55.91	56.02

Figure 2a. Model Complexity and Parameter Optimization on Decision Tree

Car dataset

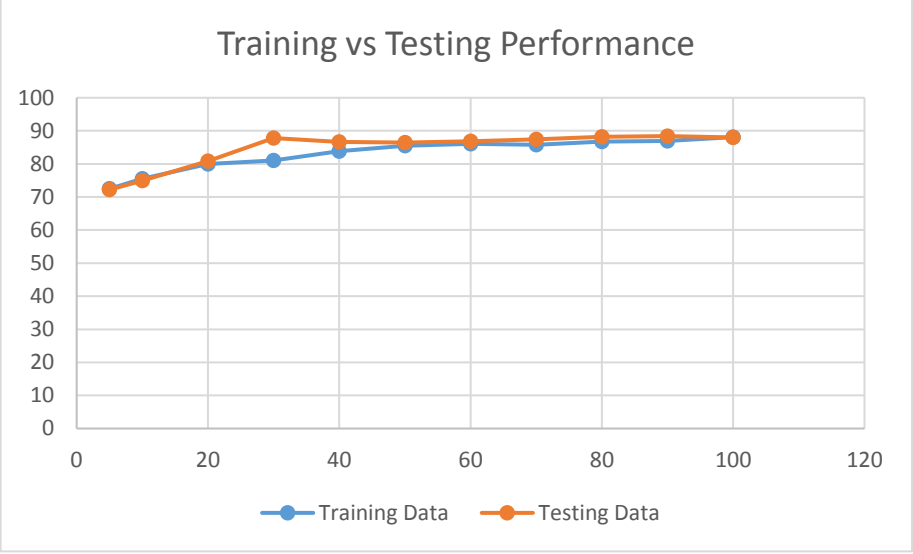

Figure 1b. Decision Tree Learning Curves

Confidence Factor: C	Minimum Number of Instances per leaf: M	Number of Leaves	Size of tree	Build Time	Training Performance (% correct)	Testing Performance (% correct)
0.1	2	62	86	0.04	87.67	87.84
0.1	3	60	83	0.001	87.52	87.65
0.2	2	72	100	0.04	89.09	87.65
0.2	3	67	93	0.01	88.34	88.03
0.3	2	101	140	0.01	90.25	89.77
0.3	3	94	130	0.01	88.17	89.75

Figure 2b: Model Complexity and Parameter Optimization on Decision Tree

Decision Tree Experiment Findings:

In the wine set there are more instances and attributes compared to car data set but error % is very high regardless of what parameters are optimized. During learning curve experiments pruning was kept fixed at 0.25 confidence factor so when I tried to optimize it I expected decrease in error % but didn't happen. Increasing instances per leaf actually increased error. The class contains 10 values of which some has almost no effect, so this is another factor for high error.

In the car dataset there is less data but the class contains only 4 values which is factor for better performance. Regardless, like in the wine dataset the confidence factor didn't change the error % very significantly.

Neural Network Experiments:

Weka is using an ANN algorithm which they call MultilayerPerceptron. According to Weka documentation the default layer parameter 'a' is calculated as 'a' = (attribs + classes) / 2. The learning curve experiments were conducted as described in "Methodology" above. For the parameter optimization experiments the default values were used. The optimization was made on Learning Rate and Momentum values.

Wine dataset

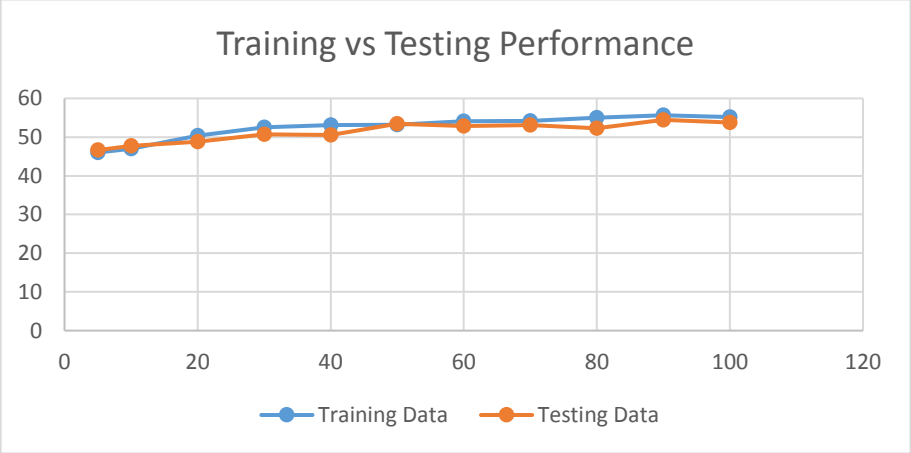

Figure 3a. ANN Learning Curves

Layers	Learning Rate	Momentum	Build Time (s)	Training Performance (% correct)	Testing Performance (% correct)
6	0.1	0.1	12.03	55	57.68
6	0.2	0.1	12.71	55.18	58.94
6	0.3	0.1	12.2	55.26	59.2
6	0.1	0.2	13.35	54.97	57.66
6	0.2	0.2	11.78	54.74	59.32
6	0.3	0.2	11.84	55.35	58.5

Figure 4a. Model Complexity and Parameter Optimization on ANN

Car Dataset

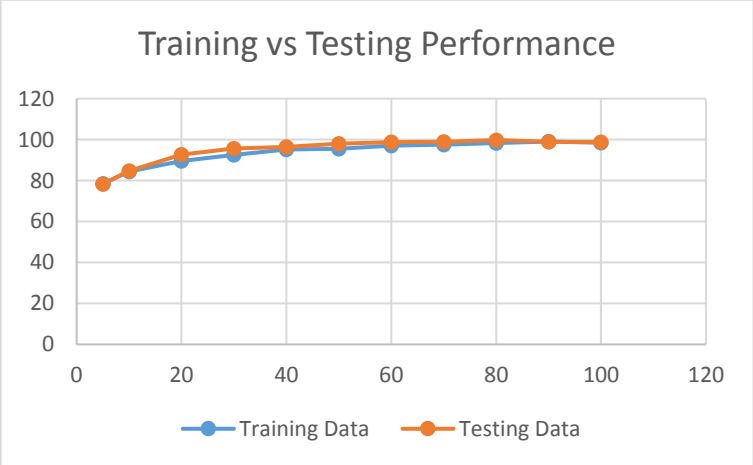

Figure 3b. ANN Learning Curves

Layers	Learning Rate	Momentum	Build Time (s)	Training Performance (% correct)	Testing Performance (% correct)
3	0.1	0.1	4.82	98.68	99.42
3	0.2	0.1	7.25	98.84	100
3	0.3	0.1	7.12	98.76	99.23
3	0.1	0.2	6.8	98.93	99.23
3	0.2	0.2	6.64	99.01	99.61
3	0.3	0.2	6.37	98.76	99.23

Figure 4b. Model Complexity and Parameter Optimization on ANN

Neural Network Experiment Findings:

In the wine set the error % was slightly higher compared to Decision Tree results. Initially I tried different hidden layer numbers, e.g. 2,4, 6, 8 .. but the effect was minimal so I decided to stay at default layer numbers and change Learning Rate and Momentum. The error rate is slightly effected by Learning rate but not momentum. The build time was quite slow but testing is fast.

In the car set KNN performed great. Regardless what values I chose the error was < 2, and occasionally <0.1 but the build time is still quite slow.

Boosting Experiments

Weka's boosting algorithm is called AdaboostM1. As the weak learner the J48 Decision Tree algorithm was used. The learning curve experiments were conducted as described in "Methodology" above (Figure 5). The tree size depends on the Confidence factor and number of objects on the branches so to optimize parameters I experimented with Confidence Factor 'C' and number of iterations (Figure 6).

Wine dataset

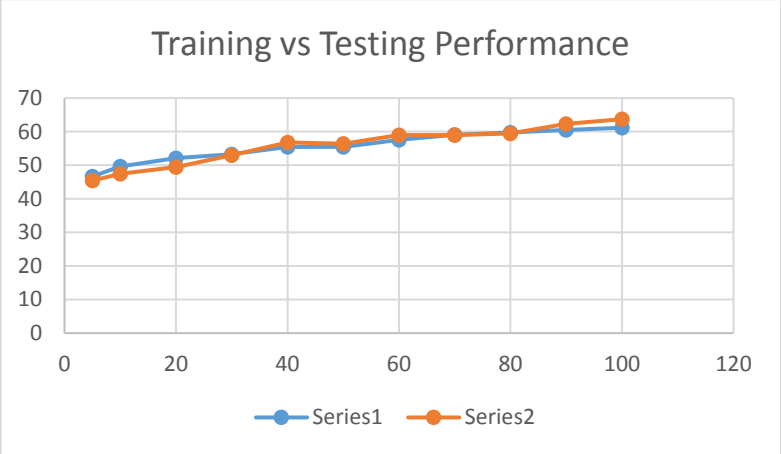

Figure 5a. Adaboost Learning Curves

Confidence Factor: C	Iterations	Number of Leaves	Size of tree	Build Time (s)	Training Performance (% correct)	Testing Performance (% correct)
0.25	10	445	889	1.35	60.96	61.47
0.25	25	438	875	3.25	64.1	62.97
0.4	10	443	885	1.23	61.94	61.74
0.4	25	421	841	3.08	63.11	63.85

Figure 6a. Model Complexity and Parameter Optimization on AdaboostM1

Car Dataset

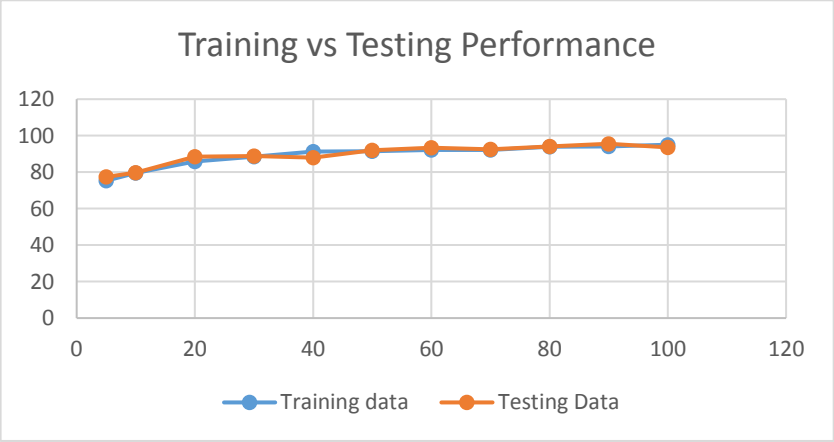

Figure 5b. Adaboost Learning Curves

Confidence Factor: C	Iterations	Number of Leaves	Size of tree	Build Time (s)	Training Performance (% correct)	Testing Performance (% correct)
0.25	10	148	204	0.04	94.88	94.4
0.25	25	149	205	0.11	95.45	95.17
0.4	10	132	181	0.04	95.12	95.17
0.4	25	193	269	0.08	95.21	95.37

Figure 6a. Model Complexity and Parameter Optimization on AdaboostM1

Boosting Experiment Findings:

In the wine set performance slightly increased. Compared to Decision tree alone the higher sampling numbers reduced the testing errors, also build times were lower as well. I decided to alter Confidence Factor and iterations. Increasing confidence factor and iteration numbers both reduce testing errors and improve learning curves.

In the car set boosting didn't bring any improvements. Changing confidence factor or iterations practically resulted in no change in almost no decrease in error %. Having less values to change in class might be a factor, or may be because the accuracy is already quite high.

SVM (Support Vector Machines) Experiments

Weka has an SVM algorithm which is called LibSVM. The learning curve experiments were conducted as described in "Methodology" above (Figure 7). SVN is an algorithm which tries to separate classes into hyperplanes by calculating distances between instances close to boundaries. In my optimization experiments I used all available kernels to see how they behaved with the default values and tried to interpret what it meant.

Wine Dataset

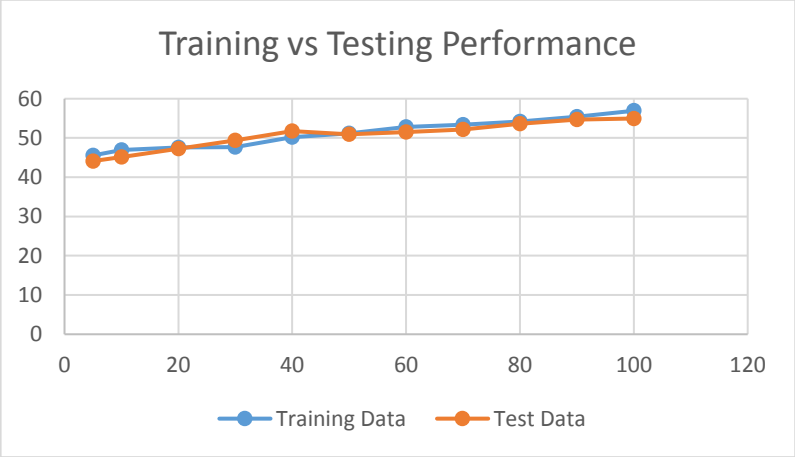

Figure 7a: SVM Learning Curves

Kernel	degree	gamma	Coeff	Build Time	Training Performance (% correct)	Testing Performance (% correct)
Linear u*v	3	1/maxIndex	0	99.96	52.87	50.58
Polynomial (gamma*u*v + C)^degree	3	1/maxIndex	0	732.92	45.26	48.06

Radial Basis Function exp(-gamma*|u-v|^2)	3	1/maxIndex	0	3.67	57.01	54.6
Sigmoid tanh(gamma*u*v +C)	3	1/maxIndex	0	0.68	45.11	44.31

Figure 8a: Model Complexity and Parameter Optimization on SVM

Car Dataset

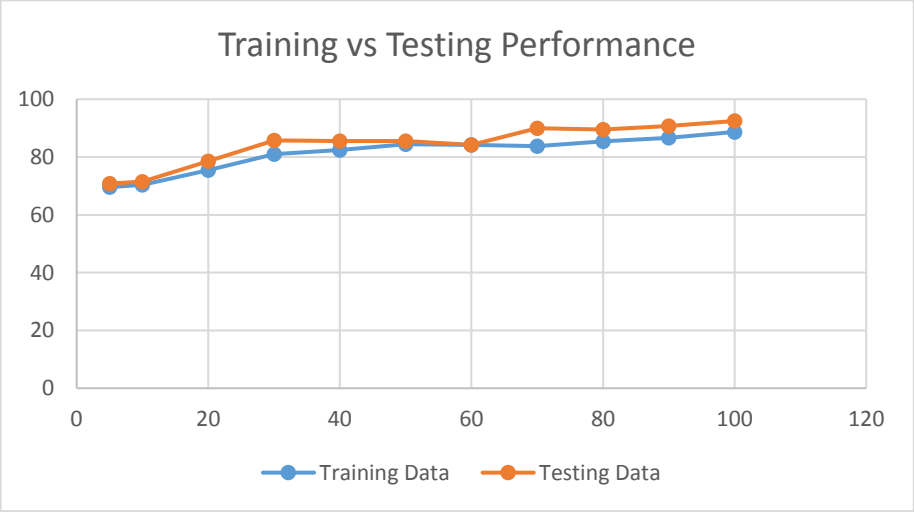

Figure 7b: SVM Learning Curves

Kernel	degree	gamma	Coeff	Build Time	Training Performance (% correct)	Testing Performance (% correct)
Linear u*v	3	1/maxIndex	0	0.03	91.65	96.14
Polynomial (gamma*u*v +C)^degree	3	1/maxIndex	0	0.06	69.67	70.85
Radial Basis Function exp(-gamma*|u-v|^2)	3	1/maxIndex	0	0.09	88.68	92.47
Sigmoid tanh(gamma*u*v +C)	3	1/maxIndex	0	0.14	81.65	86.29

Figure 8b: Model Complexity and Parameter Optimization on SVM

SVM Experiment Findings

In wine dataset SVM learning curves the initial error is not very high so improvement was moderate, however the testing set error rarely was lower than the training set error %. I chose to try different kernels to see their ultimate effect under default values. The error in polynomial kernel was highest. That was expected because there are a lot of attributes and some are very scattered so separating the hyperplanes must be less efficient, but in general all error rates were comparable to other algorithms.

In car dataset error rate was surprisingly higher compared to other algorithms. Especially the polynomial kernel ended up with almost 30% error which was the absolute highest among all experiments for car dataset. SVM appears to be sensitive to number of attributes.

KNN (K Nearest Neighbor) Learning Curves

For KNN Weka is using the IBk algorithm The learning curve experiments were conducted as described in "Methodology" above. For parameter optimization I used the number of K neighbors, the rest were as used as default values.

Wine Dataset

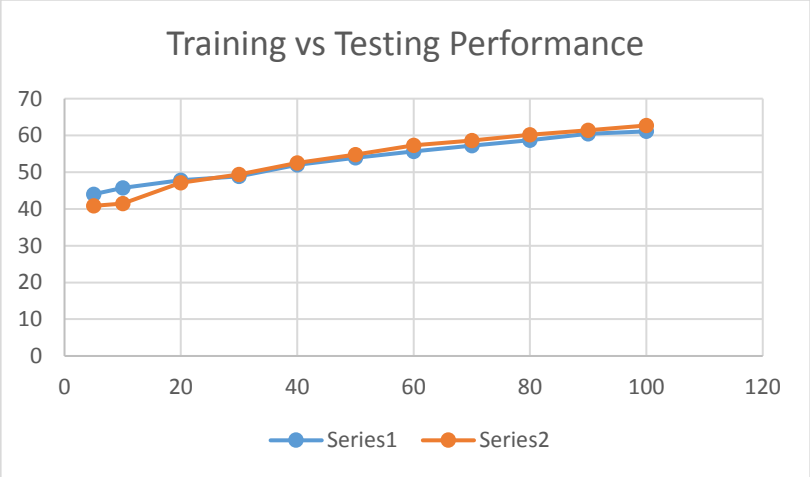

Figure 9a: KNN Learning Curves

K	Distance Weighting	Training Performance (% correct)	Testing Performance (% correct)	Training Time
1	no	61.1	62.7	0.001
10	no	54.24	52.69	0.001
50	no	54.65	52.55	0.001
1	1/distance	61.1	62.7	0.001
10	1/distance	64.3	63.6	0.001
50	1/distance	65.94	64.5	0.001

Figure 10a: Model Complexity and Parameter Optimization on KNN

Car dataset

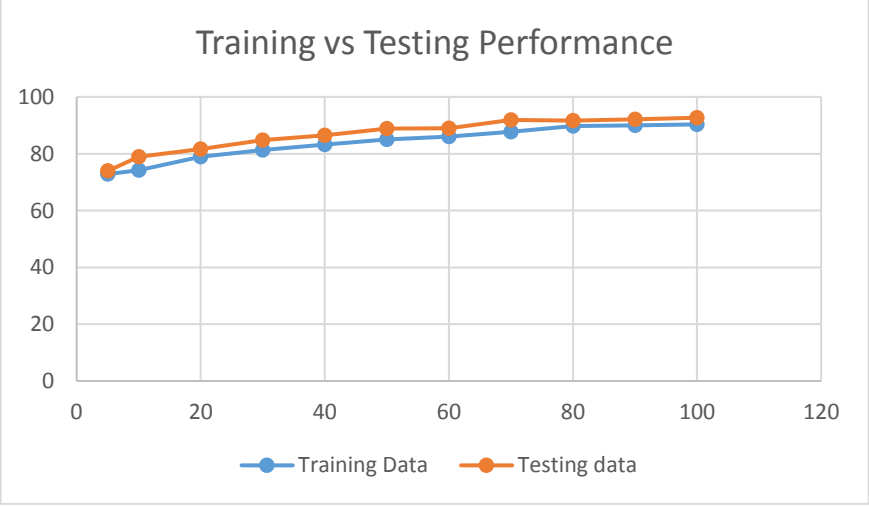

Figure 9b: KNN Learning Curves

K	Distance Weighting	Training Performance (% correct)	Testing Performance (% correct)	Training Time
1	no	90.33	92.66	0.001
10	no	84.88	88.22	0.001
50	no	76.28	77.8	0.001
1	1/distance	90.33	92.66	0.001
10	1/distance	85.7	89	0.001
50	1/distance	78.76	79.92	0.001

Figure 9b: KNN Learning Curves

KNN Experiment Findings

In wine dataset the training and testing errors were quite lower. The testing mostly produces lower error rates compared to training. I decided to optimize K and the distance weighing which resulted in noticeable improvement and using the latter resulted in almost 65 success rate which was the highest for wine set in my experiments among all algorithms. That must be based on the fact that KNN algorithm depends on distance of the close neighbors so closer neighbors should have higher weight/impact.

In car dataset error rate was surprisingly higher. Contrary to the wine dataset the error % increased on both increasing the K and using distance weighing. I have no clear understanding why but I guess that data itself is very subjective. By that I mean users made choices that are not consistent with other users choices.

Summary

Looking at the different algorithms I can say that in general all of them produced comparable results while applied to same datasets. There were some surprises discussed in "Finding" parts for each algorithm but those must be caused by way algorithms are designed, which sometime is sensitive of erroneous / arbitrary user behavior, e.g. while some users prefer an attribute others strongly reject it with no pattern.

Here is a comparison of maximum training versus testing accuracy with optimization results.

Wine			
Algorithm	**Training accuracy (% success)**	**Testing Accuracy (% success)**	**Optimization (% success)**
Boosting	61.18	63.72	63.85
KNN	61.1	62.7	64.5
Decision Tree	56.65	58.34	58.34
SVM	56.93	54.96	54.6
Neural Networks	55.64	54.46	59.32

As a reference I took the learning curve success on test dataset. **For wine dataset above** the best method was Adaboost which with minor optimization of parameters reached 63.85% success. The low success rate of neural networks on wine set could be the scattered data with so many attributes and so many class values.

Car			
Algorithm	Training accuracy (% success)	Testing Accuracy (% success)	Optimization (% success)
Neural Networks	99.09	99.81	100
Boosting	95.04	95.56	95.37
KNN	90.33	92.66	92.66
SVM	88.68	92.47	96.14
Decision Tree	88.1	88.41	89.77

As a reference I took the learning curve success on test dataset. For **car dataset above** the best method was Neural Networks which with minor optimization of parameters reached maximum 100% success and in general stayed above 99%. Because the car dataset has fewer attributes and only 4 values in the class the NN was able to show the maximum success. On the opposite side the decision tree didn't do as well as NN but if still around 90%. The surprising side was that usually NN is better when there is more data but in our case more data in the wine set didn't result in better predictions.

Conclusion

In this study five machine learning algorithms were explored and two separate datasets were used in running training and testing experiments. For wine set which has more attributes, single class but 10 class values boosting and KNN gave the best results. For car dataset Neural Networks and Boosting gave better results. In general the size of the dataset, the number of attributes, and class values are important parameters. When training performance is low some parameter optimization can improve performance. For example in car dataset a change in SVM kernel resulted in 8.4% improvement in performance.

What Can Be Done Further?

Parameter optimization looks promising, but in some cases it is a complex task. Another promising method is reduction / selection of attributes. For example in wine dataset some attributes have negligible effect on the outcome and Weka has a method of suggesting which attributes provide more information gain. I did preliminary tests and discarding a few low information gain attributes slightly improved overall performance so it could be worth experimenting.

References

Weka youtube videos
https://www.youtube.com/user/WekaMOOC/about

Ian Witten Eibe Frank Mark Hall Christopher Pal, Data Mining 4th Edition,
https://www.elsevier.com/books/data-mining/witten/978-0-12-804291-5

Gatech CS7641 Machine Learning course videos
https://classroom.udacity.com/me