CS7641: Machine Learning Assignment 1

By Yumei Wang (ywang4068)

Five supervised learning algorithms: decision tree, K-nearest neighbors (KNN), support vector machines (SVM), neural network and boosting (AdaBoost) were investigated on two different classification problems datasets.

Datasets description

Two datasets can be downloaded from UCI and Kaggle.

1. Wine classification dataset [1] as dataset 1: It was loaded from sklearn datasets[2], and has 178 instances, 13 attributes and 1 output with 3 classes (0-2) and no missing values. All attributes are constituents found in three types of wines and the goal is to determine the origin of wines using chemical analysis. The dataset is balanced with 59 instances of class 0, 71 instances of class 1, and 48 instances of class 2. All attributes are continuous.

The data has been normalized using standscaler from sklearn library for preprocessing (without preprocessing, the test accuracy of all five algorithms were very low). With default parameters, decision tree and Boosted decision tree (AdaBoost) algorithms got the same test accuracy of 86% on test data, and KNN, SVM and neural network algorithms had the same test accuracy of 97%. This makes the dataset interesting. And more interestingly, after hyperparameters tuning, test accuracy of decision tree increased to 92%. Although SVM and neural network algorithms had the same test accuracy, they learned differently observed from the learning curves.

2. Student evaluation dataset [3] (dataset 2): It has 3977 instances, 32 attributes and 1 output with 3 classes and no missing values. It is to rate the instructors by students with 28 course specific questions and additional 4 attributes. The dataset has 509 instances of class 1, 975 instances of class 2 and 2493 instances of class 3. It is not very unbalanced.

Accuracy on test data using the five algorithms with default parameters were very different, ranging from 67% to 97%. AdaBoost and decision tree achieved the highest accuracy, but SVM and neural network did poorly with less than 70% accuracy. But interestingly, after hyperparameters tuning, decision tree, AdaBoost and neural network algorithms had similar test score (98%) but different learning curves, and test accuracy of KNN increased to 85%, but SVM didn't improve much.

Analysis

Dataset 1: models test accuracy with default hyperparameters

algorithm	Decision tree	K-nearest neighbors	Support vector machines	Neural network	AdaBoost
accuracy	0.86	0.97	0.97	0.97	0.86

KNN, SVM and neural network algorithms had the same test accuracy, and decision tree and AdaBoost algorithms had similar performance.

Dataset 2: models test accuracy with default hyperparameters

algorithm	Decision tree	K-nearest neighbors	Support vector machines	Neural network	AdaBoost
accuracy	0.96	0.82	0.67	0.68	0.97

Decision tree and AdaBoost algorithms performed the best, but SVM and neural network did poorly.

Next part will tune hyperparameters to improve the models performance using cross validation (5 fold) and compare different algorithms by examining the learning curves, test accuracy, classification report, training time and prediction time.

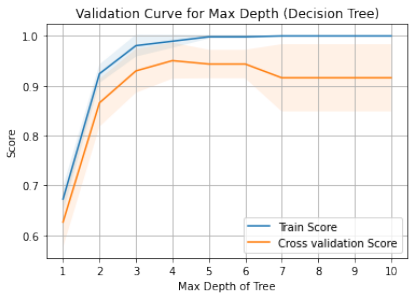
Hyperparameters tuning

Cross-validation curves of each single hyperparameter of five algorithms were plotted and all other parameters were set default values. The scoring metric for cross validation was accuracy. Using the optimal hyperparameters tuned, the learning curves of decision tree for each dataset were plotted.

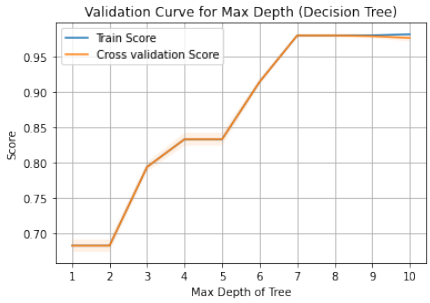
Decision tree

Validation curves for max depth of tree and min samples on leaf of decision tree model were plotted for both datasets. The criterion used to split a node is Gini index.

1. Max depth of tree



Dataset 1

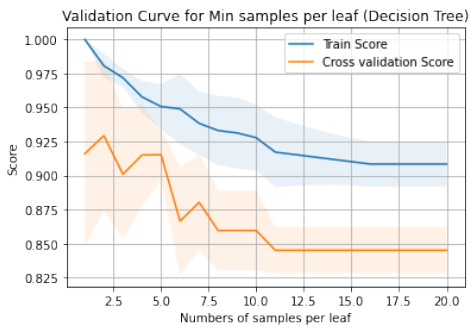


Dataset 2

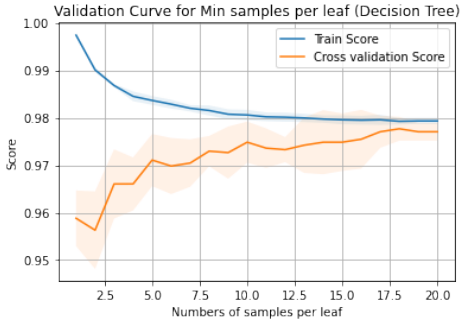
The larger the tree (the higher max depth) was, the higher training accuracy would be, but would be more likely to result in overfitting. Pruning would help to avoid overfitting, but too small max depth would lead to underfitting due to bias.

For dataset 1, the optimal max depth of tree was 4 when both scores were close and the highest. For dataset 2, the optimal max depth of tree was 7 as both scores were the same highest.

2. Min samples on leaf



Dataset 1

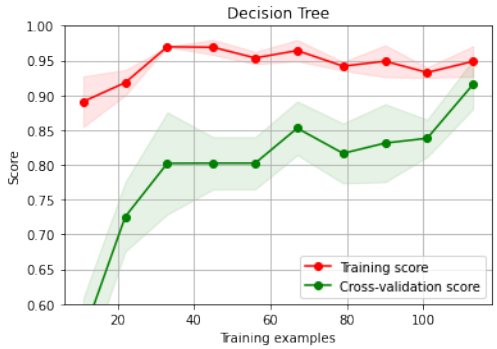


Dataset 2

Less samples on a leaf would make a larger tree, and have higher training accuracy, but cross validation would help to find the best number of samples on leaf to avoid overfitting or underfitting. For the two datasets, the validation accuracy behaved differently or oppositely as min number of samples on leaf increased.

For dataset 1, the validation accuracy also decreased, but for dataset 2, the validation accuracy increased as min number of samples on leaf increased, so for dataset 1, the optimal min samples on leaf was 2 with the highest validation accuracy. For dataset 2, optimal min samples on leaf was 18 when the training and validation accuracy converged.

Learning curve



Dataset 1

Dataset 2

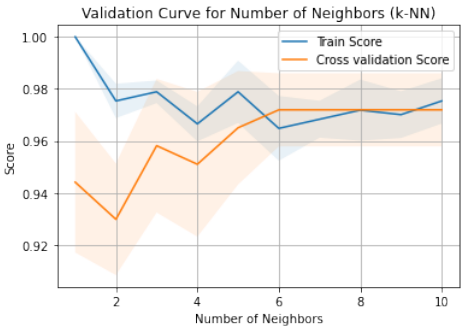
For dataset 1, the training accuracy increased first and then decreased and the highest score was above 95%, and the test accuracy continued to increase as more training examples added and the highest score was 92%, which indicated the model performed well on the training data, and good at the test data, but could be improved further if more data added.

For dataset 2, both the training and validation accuracy started from 0.90 and increased as more data added and finally the two converged at 0.97, which proved the model learned the structure of the training data fast and generalized very well and predicted correctly on the test data.

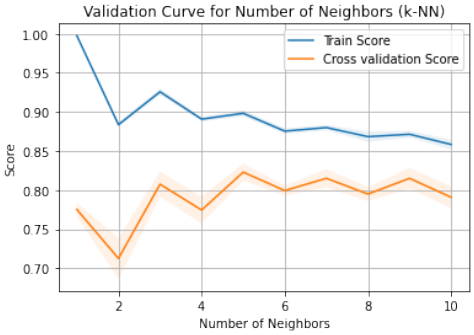
K-nearest neighbors

Validation curves for number of neighbors (k) and power of distance metric parameter (p) of K-nearest neighbors (KNN) model were plotted for both datasets.

1. Number of neighbors (k)



Dataset 1

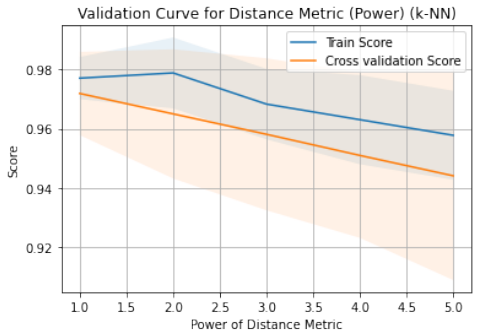


Dataset 2

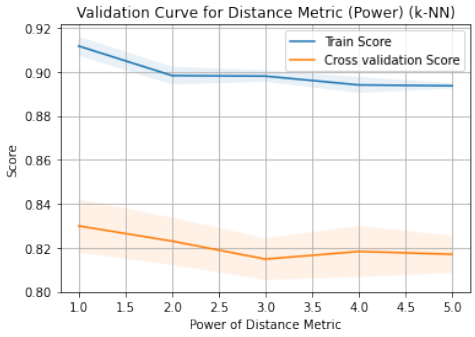
For KNN algorithm, the highest training accuracy is always when number of neighbors (k) was 1 and decreases as number of neighbors increased, but lower k tends to overfitting because the model picks only the values that are closest to the data sample and fails to generalize well on the test data.

For dataset 1, optimal number of neighbors was 6 and for dataset 2, optimal number of neighbors was 5 when the validation accuracy was the highest.

2. Power parameter (p)



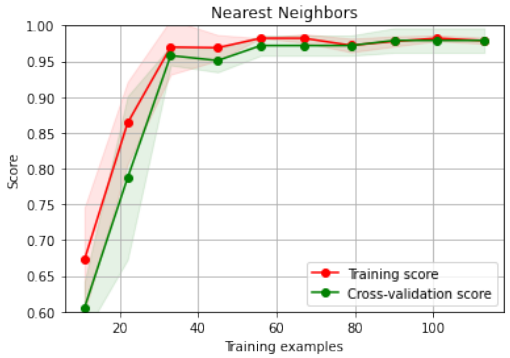
Dataset 1



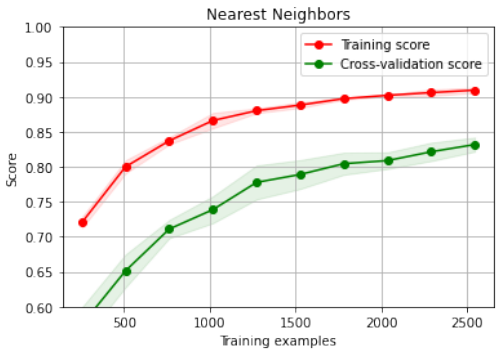
Dataset 2

For both datasets, optimal p value was 1, which is equivalent to using manhattan_distance (l1), when the training and validation accuracy were both the highest.

Learning curve



Dataset 1



Dataset 2

For dataset 1, both the training and validation accuracy increased as more data added and finally the two converged at 0.97, which proved the model learned the structure of the training data fast and generalized well and predicted correctly on the test data.

For dataset 2, both the training and validation accuracy increased as more data added, but didn't converge at last and the test score was lower than 0.85. Increasing the k value didn't improve the test score or overcome the overfitting, but adding more data might help to improve it as the two scores were both increasing and didn't converge.

Support vector machines

Validation curves for kernel functions, gamma and C of support vector machines model were plotted for both datasets. The scoring metric was accuracy.

1. Kernel functions

Dataset 1: kernel functions

Kernel	linear	poly	rbf	sigmoid
Test accuracy	0.96	0.90	0.98	0.99

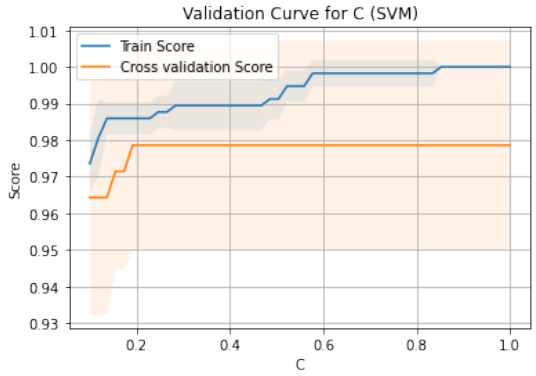
Different kernel functions got similar test score.

Dataset 2: kernel functions

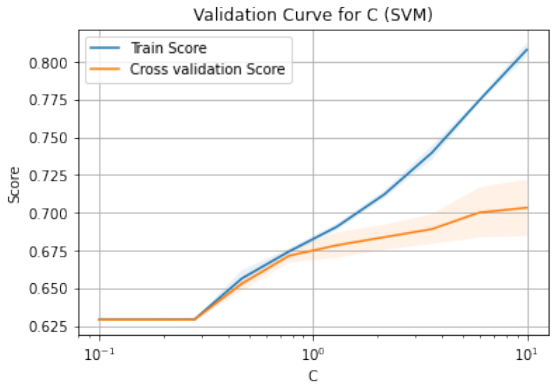
Kernel	linear	poly	rbf	sigmoid
Test accuracy	0.63	0.64	0.67	0.48

Except for sigmoid kernel, the other three performed similarly.

2. Regularization parameter C



Dataset 1

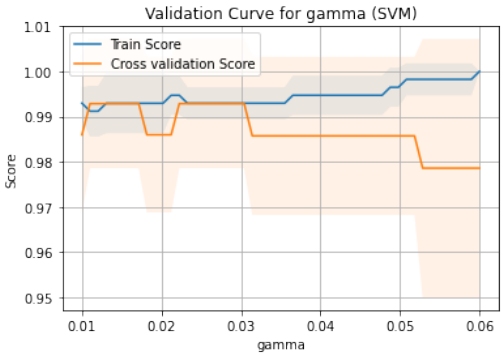


Dataset 2

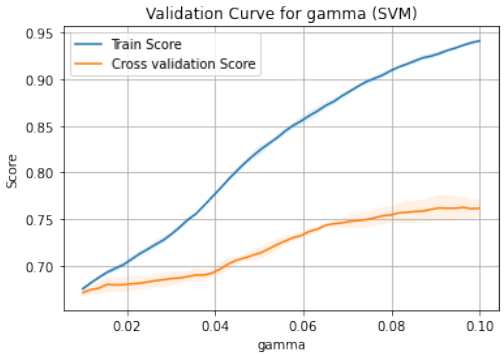
The C parameter trades off correct classification of training examples against maximization of the decision function's margin. For larger values of C, a smaller margin will be accepted if the decision function is better at classifying all training points correctly. A lower C will encourage a larger margin, therefore a simpler decision function, at the cost of training accuracy[4].

For dataset 1, the training and validation accuracy were very close and increased with C, and the optimal C was 1 when the training and validation accuracy both were the highest. For dataset 2, the training and validation accuracy increased with C, and when C was 10, the training accuracy was still increasing but validation accuracy was still low. This would suggest the model didn't capture the important patterns of the data.

3. Gamma



Dataset 1



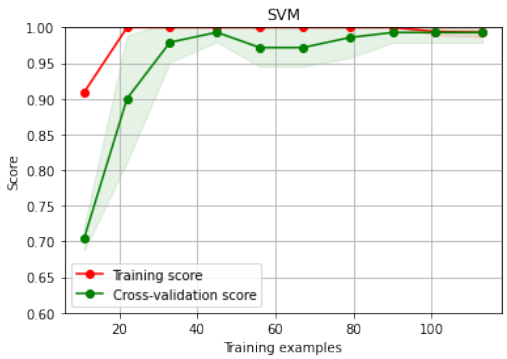
Dataset 2

Gamma parameter of RBF defines how far the influence of a single training example reaches, with low values meaning 'far' and high values meaning 'close'[4]. A high gamma value means only the closest points to the decision boundary will carry the weight leading to a smoother boundary. One the other hand a low gamma value means even the points far away from the decision boundary have a weight leading to a more wiggly boundary[5].

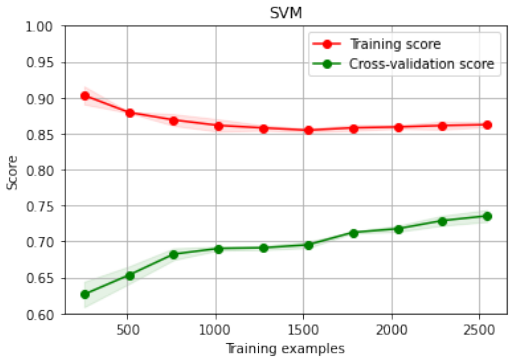
For dataset 1, the training and validation accuracy varied very little with gamma, but 0.015 was chosen as the optimal gamma value when the training and validation accuracy were the same.

For dataset 2, when gamma is very small (0.02), the model is too constrained and cannot capture the complexity or "shape" of the data, however, when gamma is 0.1, the training accuracy was very high but validation accuracy was still low. This showed overfitting due to high variance. To avoid overfitting, 0.04 was chosen as the optimal gamma value.

Learning curve



Dataset 1



Dataset 2

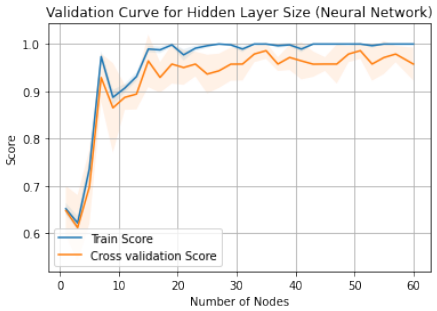
For dataset 1, the training and validation score were very similar with training examples added. Both increased and finally converged at 1.0, the model performed very well.

For dataset 2, the training accuracy decreased as more data added, while the validation accuracy increased but they didn't converge, which should be caused by high variance. More data might help to improve the model performance.

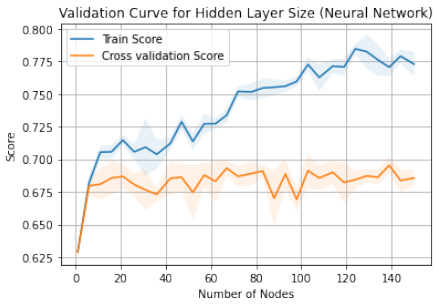
Neural network

Validation curves for hidden layer size, activation functions and learning rate of neural network model were plotted for both datasets.

1. Hidden layer size



Dataset 1



Dataset 2

For dataset 1, the training and validation score fluctuated using a single layer neural network when increasing number of nodes. But a single layer of 20 nodes would got higher than 90% accuracy and 35 nodes got the highest score.

For dataset 2, the training and validation accuracy increased as more nodes added, but validation score didn't improve much and both scores were low. This one layer neural network was too simple and underfit. More layers and (/or) more nodes should be added to capture the complexity of the data.

2. Activation function

Dataset 1: activation

activation	identity	logistic	tanh	relu
Test accuracy	0.98	0.97	0.97	0.96

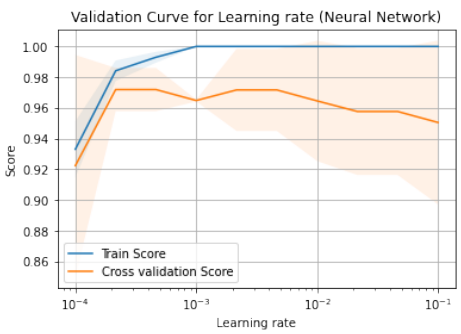
Different activation got similar test scores.

Dataset 2: activation

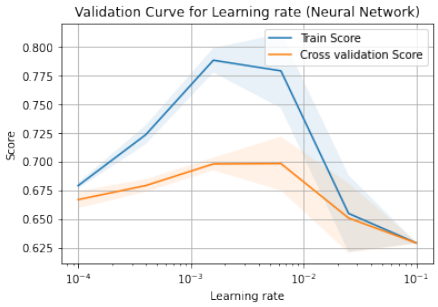
activation	identity	logistic	tanh	relu
Test accuracy	0.62	0.69	0.69	0.70

Different activation got similar test scores. Relu was a little better.

3. Learning rate



Dataset 1

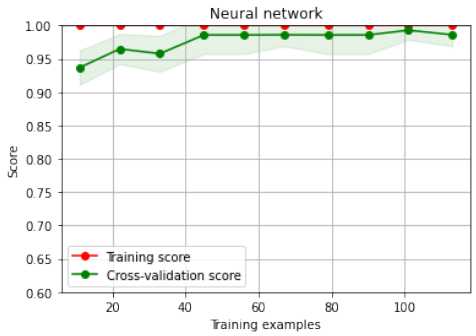


Dataset 2

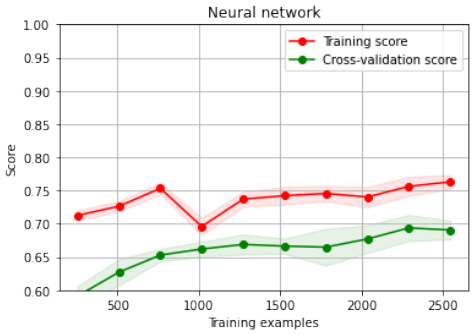
The learning rate controls how much to change the model in response to the estimated error each time the model weights are updated. A large learning rate can cause the model to converge too quickly to a suboptimal solution.

For both datasets, training and validation accuracy increased first, then plateaus and finally decreased as learning rate increased. For both datasets, learning rate of 0.002 was the optimal choice.

Learning curve



Dataset 1

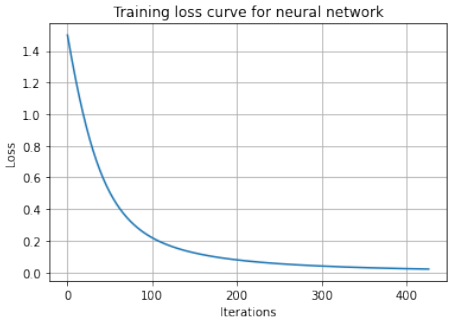


Dataset 2

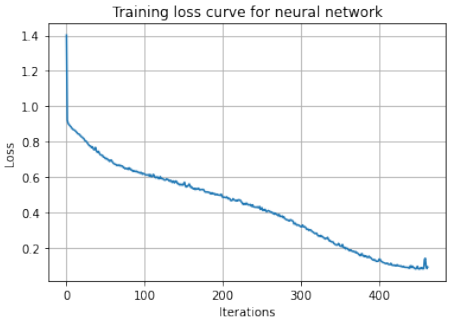
For dataset 1, the training accuracy was always 1.0 and validation accuracy increased from 0.93 to 0.97 and converged at last, which showed the model had a great performance.

For dataset 2, both training and validation accuracy were low and didn't improve much as more training data added. Two or three layers with 20 nodes were tested but both didn't increase the performance, so it suggested the model was not able to capture the complexity of the data.

Loss curve



Dataset 1



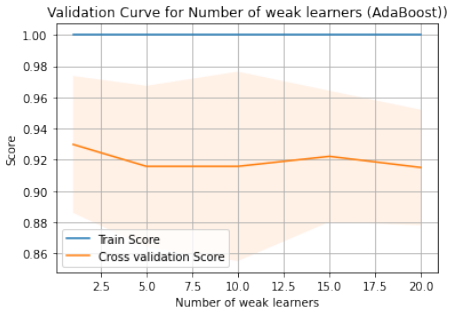
Dataset 2

For both datasets, the loss decreased after each iteration, and loss of dataset 1 was smooth and almost decreased to 0 at last, indicating the model performed well. However, the loss of dataset 2 was not smooth and decreased slowly to 0. Combined with the learning curve, it indicated the model performed poorly.

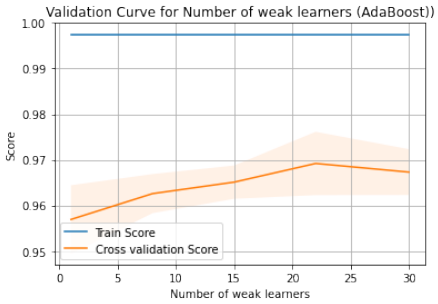
Boosting

AdaBoost was used in this experiment and the base estimator was decision tree with the optimal hyperparameters tuned above. Validation curves for number of weak learners and learning rate of AdaBoost model were plotted for both datasets.

1. Number of weak learners



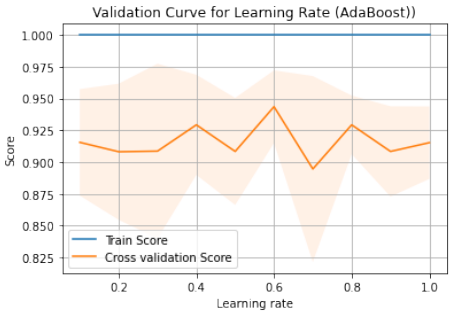
Dataset 1



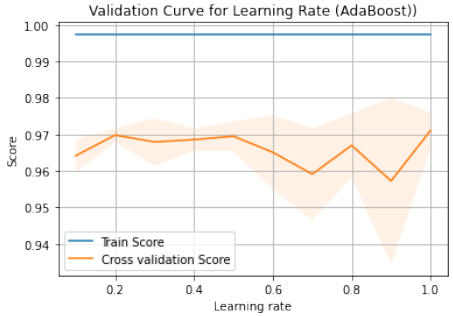
Dataset 2

For both datasets, single decision tree already got the best accuracy for training data and number of weak learners had little effect on the validation accuracy. GridSearchCV got the best number of weak learners of 5 for both datasets.

2. Learning rate



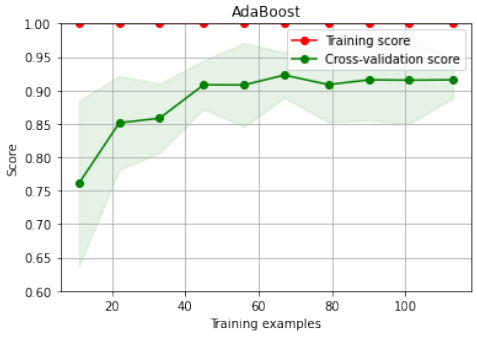
Dataset 1



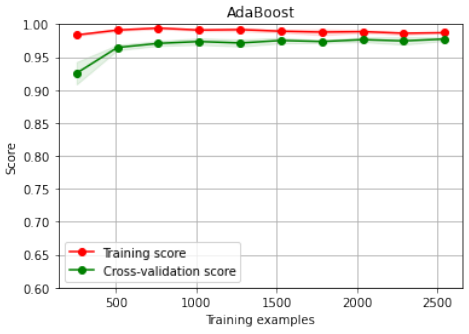
Dataset 2

For both datasets, different learning rate didn't affect the training accuracy, and had little effect on validation accuracy (although got up and down). For dataset 1, learning rate of 0.1 was chosen and for dataset 2, learning rate of 0.2 was chosen as the best.

Learning curve



Dataset 1



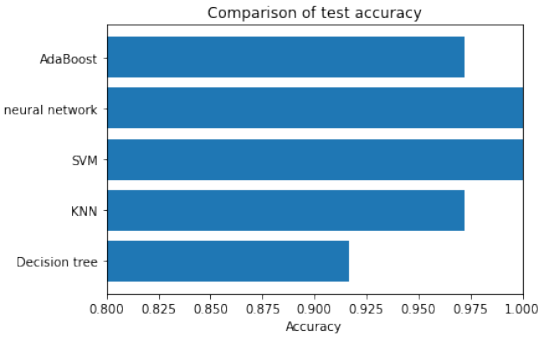
Dataset 2

For dataset 1, compared to single decision tree, AdaBoost performed better as the training accuracy was 1.0 and the validation accuracy increased gradually and plateaus at 0.92, although there was a little overfitting.

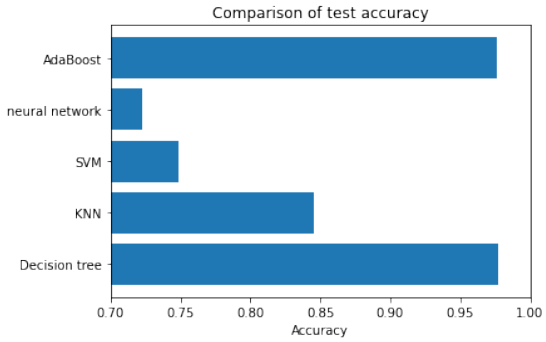
For dataset 2, the single decision tree performed better than AdaBoost. It might be because the boosted model take the noise of the data and didn't generalize perfectly.

Comparison of different algorithms

Test accuracy



Dataset 1



Dataset 2

Dataset 1: After hyperparameters tuning, all algorithms performed well on the test data and especially neural network and SVM got 100% accuracy. Decision tree can be improved if more data are added.

Dataset 2: After hyperparameters tuning, five algorithms behaved differently. Decision tree and AdaBoost had the highest accuracy (98%) as they learned fast and predicted well. KNN was the in the middle. Neural network and non-linear SVM performed poorly as both can't learn the patterns of the data and did bad at the training data.

Classification report

Precision, recall, f1-score in addition to accuracy of different algorithms were reported in the following tables for both datasets.

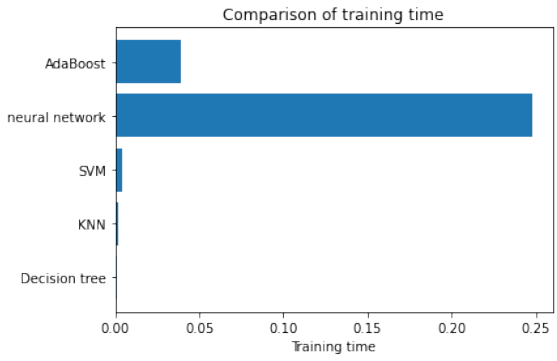
	class	**Dataset 1**			
		precision	recall	f1-score	accuracy
Decision tree	0	0.93	0.93	0.93	0.92
	1	0.86	0.92	0.89	
	2	1.00	0.89	0.94	
KNN	0	0.93	1.00	0.97	0.97
	1	1.00	0.92	0.96	
	2	1.00	1.00	1.00	
SVM	0	1.00	1.00	1.00	1.00
	1	1.00	1.00	1.00	
	2	1.00	1.00	1.00	
Neural network	0	1.00	1.00	1.00	1.00
	1	1.00	1.00	1.00	
	2	1.00	1.00	1.00	
AdaBoost	0	0.93	1.00	0.97	0.97
	1	1.00	0.92	0.96	
	2	1.00	1.00	1.00	

Dataset 1: All algorithms achieved high scores for precision, recall and f1-score in addition to accuracy.

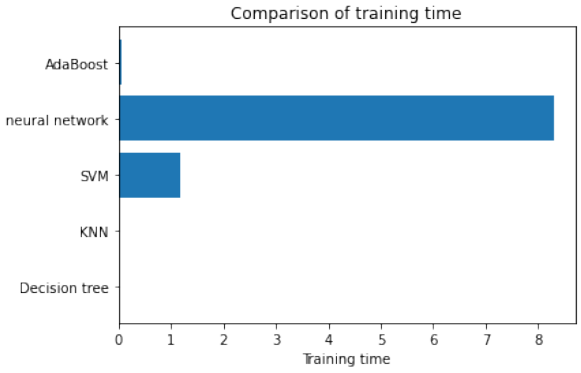
	class	Dataset 2			
		precision	recall	f1-score	accuracy
Decision tree	1	1.00	1.00	1.00	0.98
	2	1.00	0.91	0.95	
	3	0.96	1.00	0.98	
KNN	1	0.76	0.65	0.70	0.85
	2	0.77	0.75	0.76	
	3	0.89	0.93	0.91	
SVM	1	0.63	0.36	0.46	0.75
	2	0.90	0.41	0.46	
	3	0.74	0.97	0.84	
Neural network	1	0.56	0.26	0.36	0.72
	2	0.63	0.57	0.60	
	3	0.77	0.88	0.82	
AdaBoost	1	1.00	1.00	1.00	0.98
	2	0.99	0.92	0.95	
	3	0.97	1.00	0.98	

Dataset 2: Decision tree and AdaBoost got high scores for precision, recall and f1-score in addition to accuracy, but SVM and neural network performed poorly for class 1 and class 2, which might be because the number of instances of class 1 and class 2 are fewer than class 3.

Training time



Dataset 1

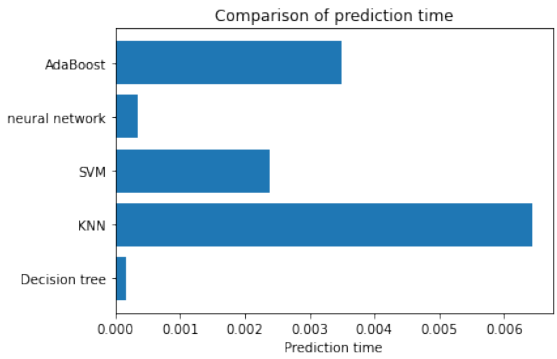


Dataset 2

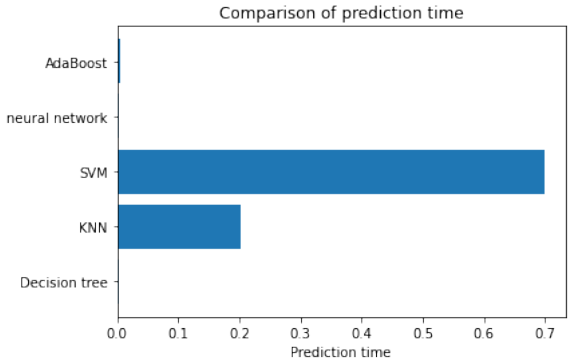
Dataset 1: Neural network took the longest time to train the data due to its iterative nature of the optimization and backpropagation to adjust weights each iteration, and then AdaBoost whose training time depends on number of weak learners. Decision tree takes longer than SVM because it didn't capture the complexity of the training data and learned slowly. KNN was the fastest as it doesn't learn but just stores the training dataset.

Dataset 1: Neural network was the slowest to train, and then SVM as quadratic optimization is faster than iterative searching. KNN and decision tree were very fast. Although the max depth of tree is 7, the model learned the intrinsic patterns of the data very well.

Prediction time



Dataset 1



Dataset 2

Dataset 1: KNN took the longest time to predict because it calculates all the distances between predict target and training data point. Neural network was faster to predict than to train. Decision tree was fast because the max depth of tree was small.

Dataset 2: Non-linear SVM took longer time than KNN mainly because it failed to learn the training data well and depends on the number of support vectors. Neural network and decision tree were the fastest.

Prediction time is relatively shorter than training time, so training time would be the most important to consider when comparing different algorithms performance. Combining test accuracy, training time and prediction time, for dataset 1, SVM was the best model due to its 100% accuracy and short training time. For dataset 2, decision tree was the best model due to its highest accuracy and shortest training time.

Reference:

1. https://archive.ics.uci.edu/ml/datasets/Wine
2. https://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_wine.html#sklearn.datasets.load_wine
3. https://www.kaggle.com/datasets/yatishbn/uci-turkiye-student-evaluation-data-set
4. https://scikit-learn.org/stable/auto_examples/svm/plot_rbf_parameters.html#:~:text=Intuitively%2C%20the%20gamma%20parameter%20defines,the%20model%20as%20support%20vectors.
5. https://amagash.github.io/pages/exploration/supervised-learning/svm.html