CS7641 A1: Supervised Learning

Tien Hoang
thoang30@gatech.edu

I. Introduction

In this assignment I explore five common learning algorithms in machine learning: decision trees (dt), neural networks (nn), boosting (boost), support vector machines (svm), and k-nearest neighbors (knn). Those algorithms are well implemented in the scikit-learn Python library. For each algorithm, validation curves for important parameters are studied, hyper-parameters are tuned using grid search and 5-fold cross-validation. Each algorithm performance is compared by three criteria: accuracy score, training time, and testing time.

The two data sets I chose are from Kaggle datasets: "Breast Cancer Coimbra Data Set" (BC) and "Disease Prediction Using Machine Learning" (DP). Both of them are real world data in health care where we can use machine learning to solve the real world problems. BS is a binary classification while DP is a multi-class classification. In addition, the contrast in size of data, type of data, etc will help us understand more about the algorithms.

II. Data sets

A. *Breast Cancer Coimbra Data Set (BS)*

Source: https://www.kaggle.com/datasets/yasserhessein/breast-cancer-coimbra-data-set

There are 10 quantitative predictors, along with a binary dependent variable denoting the existence or nonexistence of breast cancer. These predictors consist of anthropometric measurements and parameters typically obtainable through standard blood tests. Should these predictors prove to be reliable, prediction models could potentially serve as a biomarker for detecting breast cancer.

Quantitative Attributes:

- `Age`: years
- `BMI`: kg/m2
- `Glucose`: mg/dL
- `Insulin`: µU/mL
- `HOMA`
- `Leptin`: ng/mL
- `Adiponectin`: µg/mL
- `Resistin`: ng/mL
- `MCP-1`: pg/dL

Labels:

- `1`: Healthy
- `2`: Patients

B. *Disease Prediction Using Machine Learning (DP)*

Source: https://www.kaggle.com/datasets/kaushil268/disease-prediction-using-machine-learning

This dataset comprises 132 parameters that enable the prediction of 42 distinct types of diseases. The complete dataset is divided into two CSV files: one for training and the other for testing your model. Each CSV file contains 133 columns. Among these columns, 132 represent symptoms that individuals may exhibit, while the last column indicates the prognosis. These symptoms are associated with the 42 diseases that can be categorized based on this set of symptoms. Our task is to train your model using the training data and evaluate its performance using the testing data.

III. Decision Trees

In both datasets, I will use the decision tree algorithm for the classification task. To combat the overfitting problem, the parameter ranges of the decision tree are pre-pruned by using validation curves with 5-fold cross-validation. After that, the hyper-parameters (criterion, max_depth, min_samples_split, max_leaf_nodes) are tuned using grid search technique.

A. *Validation curve*

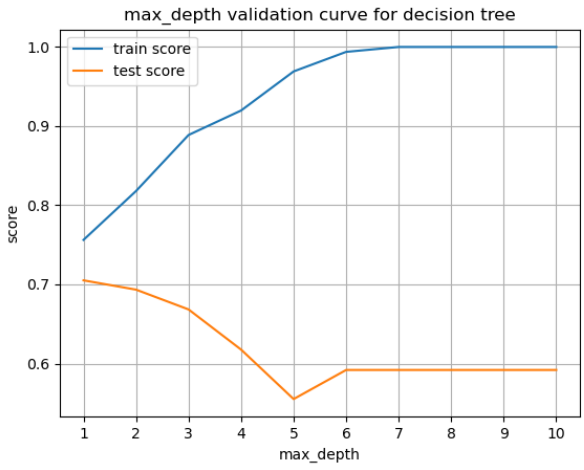

Fig. 1: BC dataset: max_depth

When max_depth is set to small values, the tree undergoes aggressive pruning, leading to a problem of underfitting. This is evident from the fact that both the training

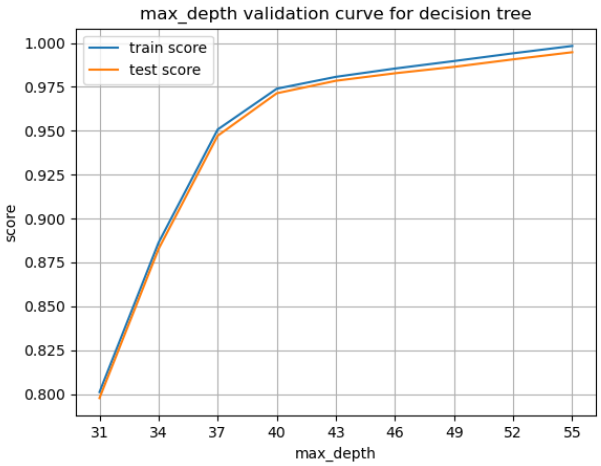

Fig. 2: DP dataset: max_depth

and cross-validation scores remain low. As max_depth increases, the training score steadily rises. Nevertheless, the cross-validation score initially improves but eventually begins to decline, eventually reaching a plateau. Consequently, when max_depth is pushed to higher values, the tree starts to exhibit overfitting tendencies. The situations are similar for two datasets as show in figure 1 and figure 2.

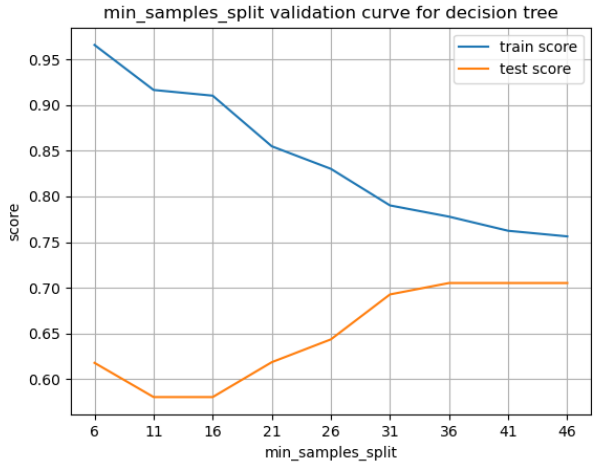

Fig. 3: BC dataset: min_samples_split

As we known, min_samples_split specifies the minimum number of samples in a node before it can be split into child nodes. The primary purpose of this parameter is to prevent the tree from becoming too deep and overfitting the training data. In the BC dataset (figure 3), the train score decreases while the test score increases while the min_samples_split is increased. The figure gives us the idea of how big the min_samples_split should be chosen. On other hand, in the DP dataset (figure 4), both train and test score are at maximum and decrease when the min_samples_split is increased.

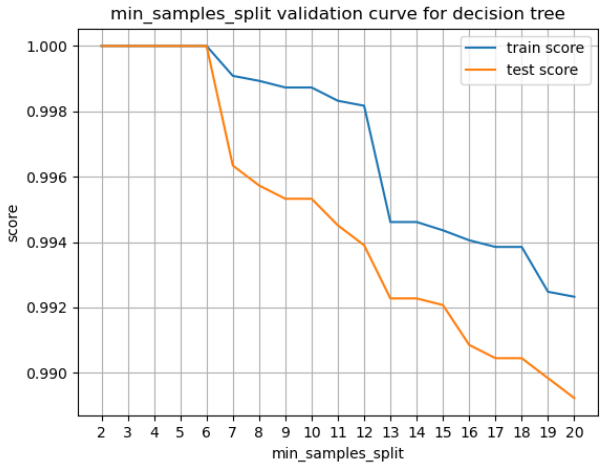

Fig. 4: DP dataset: min_samples_split

B. *Hyperparameter tuning*

From the validation curve, we have the range of the parameters for the grid search. Below are the best parameters set for our two datasets:

BS dataset:

- `criterion`: entropy
- `max_depth`: 3
- `max_leaf_nodes`: 4
- `min_samples_split`: 2

DP dataset:

- `criterion`: gini
- `max_depth`: 41
- `max_leaf_nodes`: 49
- `min_samples_split`: 4

C. *Learning curve*

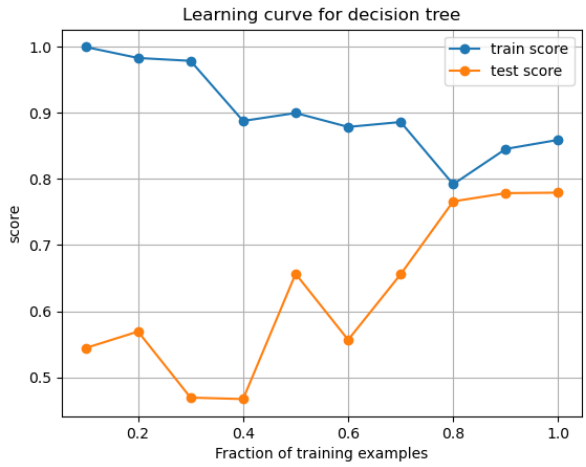

Fig. 5: BC dataset: learning curve

To identify issues such as overfitting or underfitting, learning curves are studied for BC data (figure 5 and DP data (figure 6). The optimal training fraction is around 0.8 as expected.

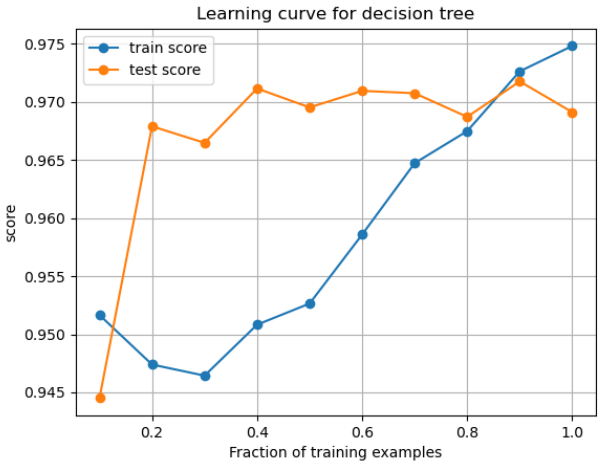

Fig. 6: DP dataset: learning curve

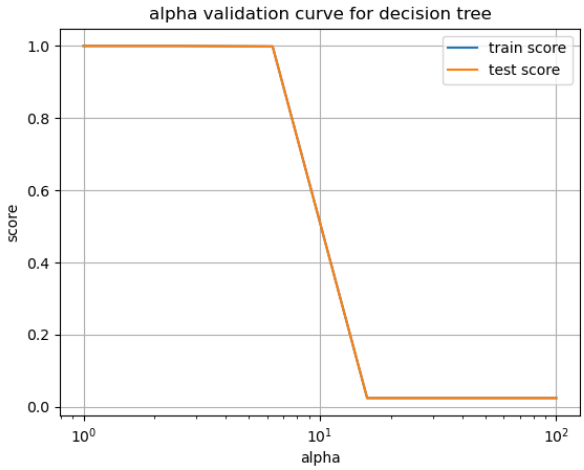

Fig. 8: DP dataset: alpha

IV. Neural Networks

In the report, I explore the Multi-layer Perceptron classifier (MLPClassifier class in sklearn library) for both datasets. As the default parameter, ReLU activation fuction is used. The hidden layer sizes for the last one should match with the number of outputs (2 for BS dataset, and 42 for DP dataset). The alpha and learning rate are tuned using validation curves and grid search.

A. *Validation curve*

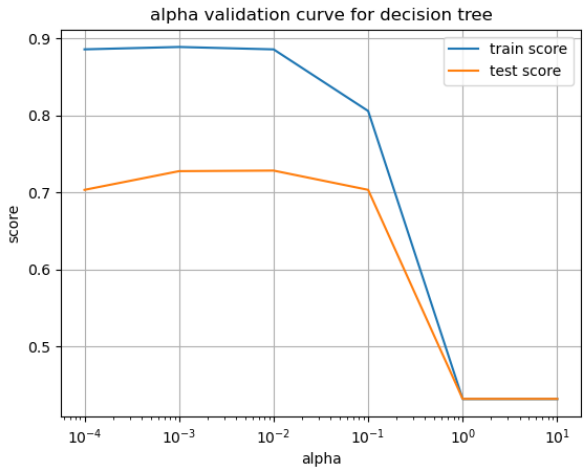

Fig. 7: BC dataset: alpha

We tune alpha which is the strength of the L2 regularization term in neural network algorithm. Figure 7 and figure 8, both show the similar pattern that the training score and testing score are reduced to zero if the alpha increases to a threshold.

Learning rate schedule determines the size of the steps taken during the optimization process when adjusting the model's weights and biases to minimize the loss function. Properly tuning the learning rate is essential for training neural networks effectively. Figure 9 and figure 10 show the learning rate curve for BC and DP datasets respectively.

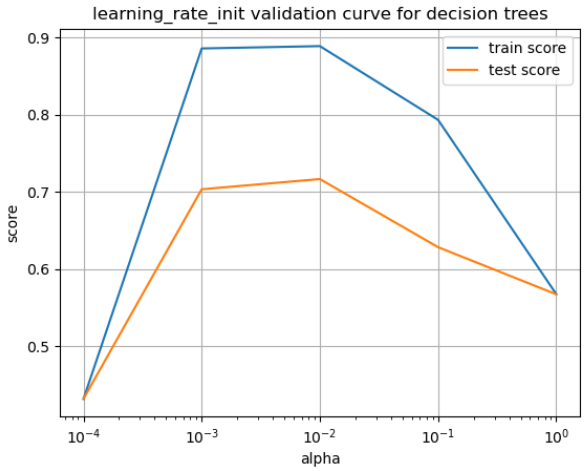

Fig. 9: BC dataset: learning rate

B. *Hyperparameter tuning*

From the validation curve, we have the range of the parameters for the grid search. Below are the best parameters set for our two datasets:

BS dataset:

- `alpha`: 0.067
- `learning_rate_init`: 0.001

DP dataset:

- `alpha`: 0.001
- `learning_rate_init`: 0.01

C. *Learning curve*

The learning curves are studied for BC data (figure 11 and DP data (figure 12). Figure 11 shows that the both train and test score decrease then increase together as the

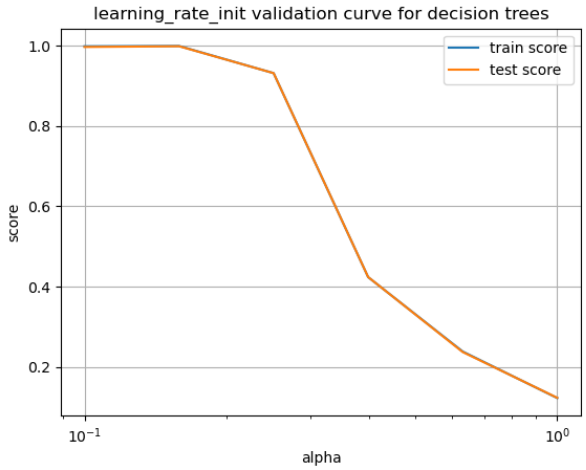

Fig. 10: DP dataset: laerning rate

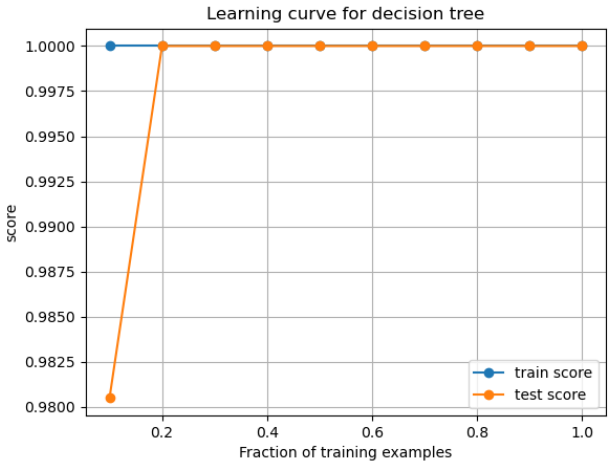

Fig. 12: DP dataset: learning curve

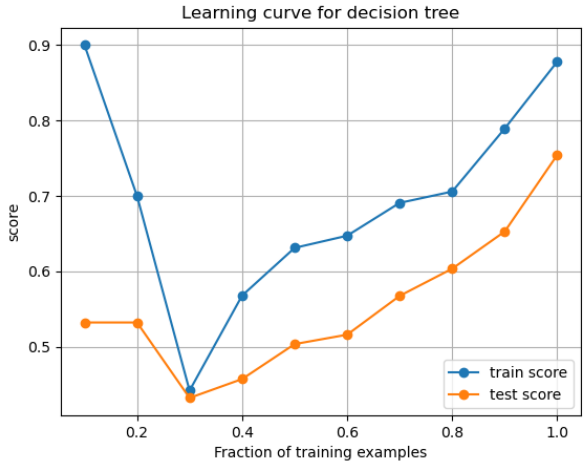

Fig. 11: BC dataset: learning curve

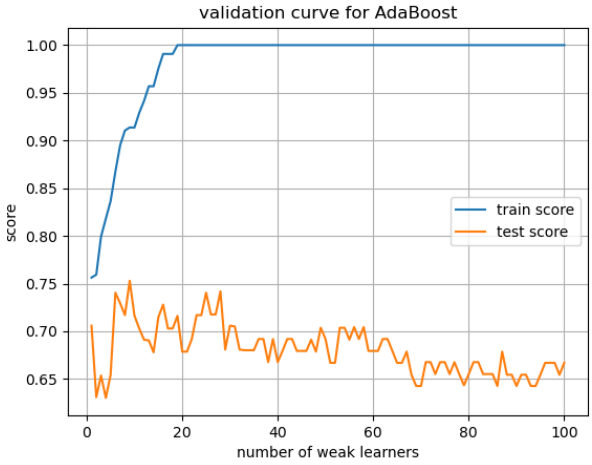

Fig. 13: BC dataset: Number of weak learners

training sample increasing. On the other hand, figure 12 show both train and test score almost equal to 1.

V. Boosting

In both datasets, I will use the decision stumps as weak learners for AdaBoost. AdaBoost, short for Adaptive Boosting, is a popular ensemble machine learning algorithm which works by assigning different weights to data points during training. Hyperparameters in AdaBoost include the number of weak learners (iterations), the type of weak learner used (e.g., decision tree stumps), and the learning rate (controls the contribution of each weak learner to the final prediction).

A. *Validation curve*

Figure 13 and figure 14 show the train and test score as function of number of weak learners. For both the datasets, the training accuracy first increases with an increase in the number of weak learners and then plateaus. In the figure 13, the test score first increases then fluctuate and decrease. The optimal number of weak learners is around 8. In figure 14 the test score increase and then plateaus as the train score. The optimal number of weak learners is around 34.

B. *Learning curve*

For BC dataset (figure 15), test score increases with an increase in the training data size. This can be explained by a property of boosting according to which it can fail to perform well if the given data is insufficient. For DP dataset (figure 16), the training and cross-validation scores converged to a low value. This might be because of the presence of noise/outliers in the data.

VI. SVM

Experiments were conducted using Support Vector Machine model (SVC class in scikit-learn library with default 'rbf' kernel). The hyperparameters we studied in this report are regularization C parameter and kernel coefficient gamma. The strength of the regularization

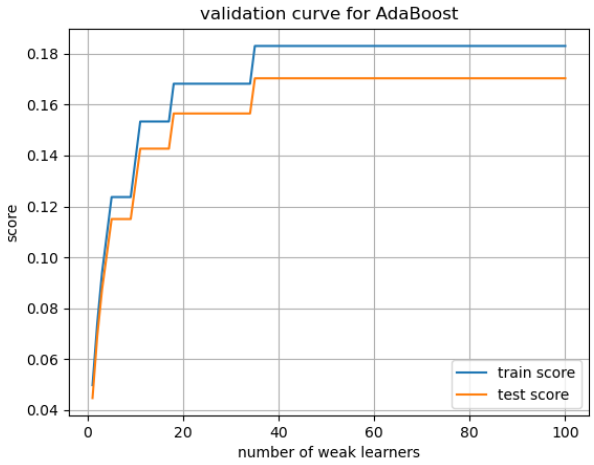

Fig. 14: DP dataset: Number of weak learners

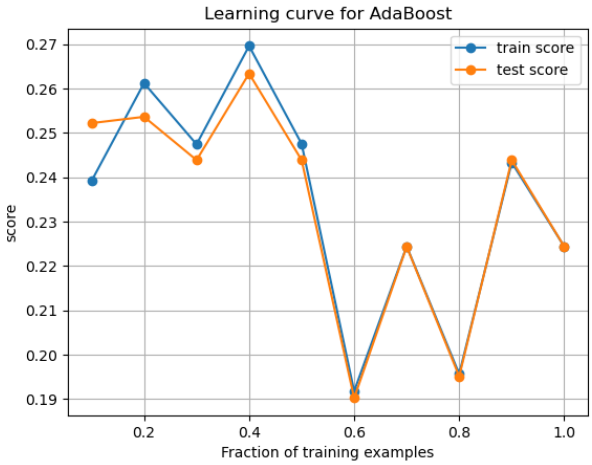

Fig. 16: DP dataset: learning curve

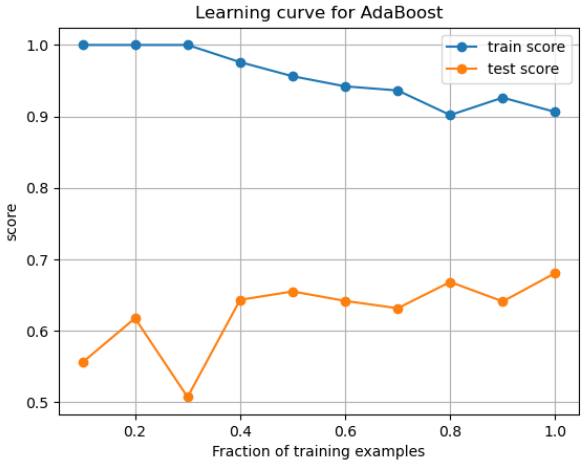

Fig. 15: BC dataset: learning curve

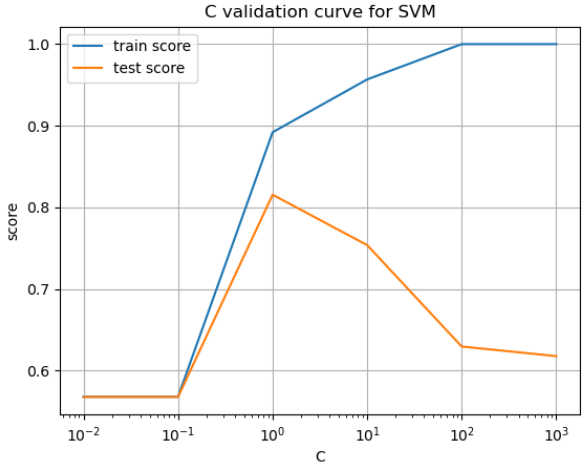

Fig. 17: BC dataset: C parameter

(squared l2 penalty) is inversely proportional to C. Both C and gamma must be positive.

A. *Validation curve*

The C parameter is a crucial hyperparameter that controls the trade-off between maximizing the margin (distance) between the decision boundary and data points and minimizing classification errors. Figure 17 show the C validation curve for the BC dataset, the train and test scores first increase as C increase. At C = 1.0, the test score begins to decrease while the train score sill increase which shows the overfitting.

The gamma parameter is a crucial hyperparameter that controls the shape of the decision boundary when using the Radial Basis Function (RBF) kernel. Tuning the gamma parameter is necessary because it has a significant impact on the performance and behavior of the SVC, especially when dealing with non-linearly separable data. Figure 18 show the gamma validation curve for the BC dataset which has similar shape as the C validation curve. The DP dataset has the train and test score equal 1.0 all the time which are not shown here.

B. *Hyperparameter tuning*

From the validation curve, we have the range of the parameters for the grid search. Below are the best parameters set for our two datasets:

BS dataset:

- `C`: 1.0
- `gamma`: 0.1

DP dataset:

- `C`: 0.1
- `gamma`: 1

C. *Learning curve*

For BC dataset (figure 19), the accuracy increases with the number of training examples and is still increasing when the training data size is at its maximum. The test accuracy hasn't converged to the training accuracy,

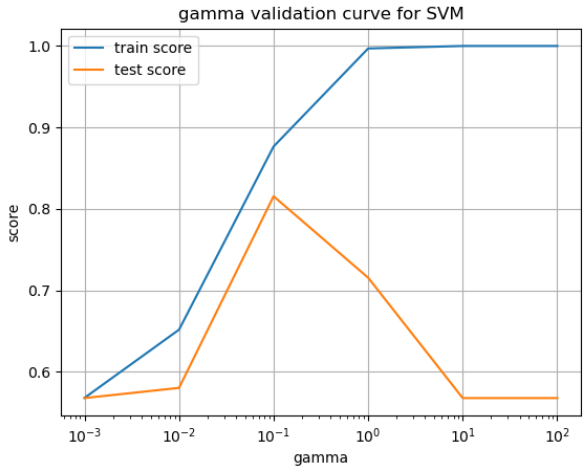

Fig. 18: BC dataset: gamma parameter

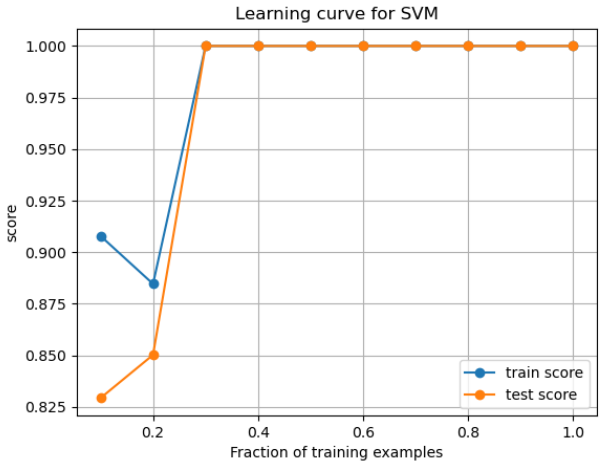

Fig. 20: DP dataset: learning curve

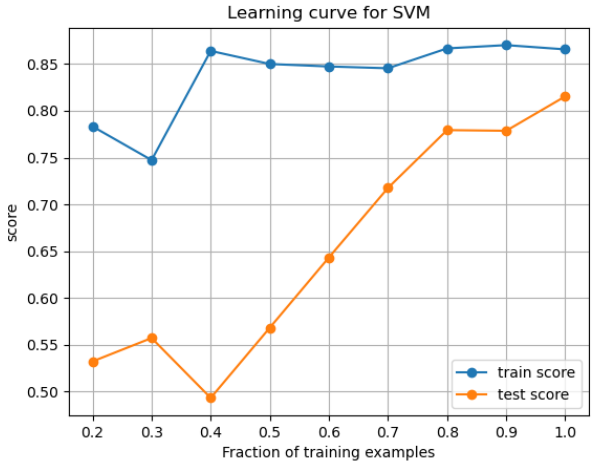

Fig. 19: BC dataset: learning curve

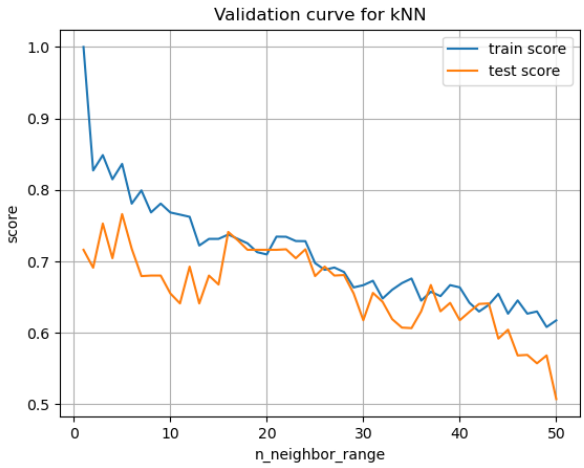

Fig. 21: BC dataset: number of neighbors

indicating high variance. This suggests that test accuracy may improve further if more training data is added. For DP dataset (figure 20), the training and test error converge to a 1.0 which indicate the good performance.

VII. KNN

K-Nearest Neighbors (knn)is a type of instance-based learning, where the model makes predictions by finding the K training examples that are closest to a new, unseen data point and then making predictions based on the majority class (for classification) or the average (for regression) of those K neighbors.

In this report, I only study the validation curve for number of neighbors (k) which also the most importance parameter of the knn.

A. *Validation curve*

The choice of k should be guided by the characteristics of the data and the desired trade-off between model complexity and generalization ability. For low values of k, the train score is high whereas test score is low, indicating that the model is suffering from high variance (overfitting). As k increases, train score drops and test score increases. After a certain point, both the scores start dropping because of high bias (underfitting) (figure 21). The optimal number of neighbors for BC and DP database are 8 and 34 respectively.

B. *Learning curve*

For BC dataset (figure 22, both train and test score increase with an increase in the number of training examples. As the train score is much greater than the test score for the maximum training data size, adding more training data will most likely increase generalization as the classifier is suffering from high variance. For DP dataset (figure 23, the train and test score increase together with an increasing number of training examples which show the good performance.

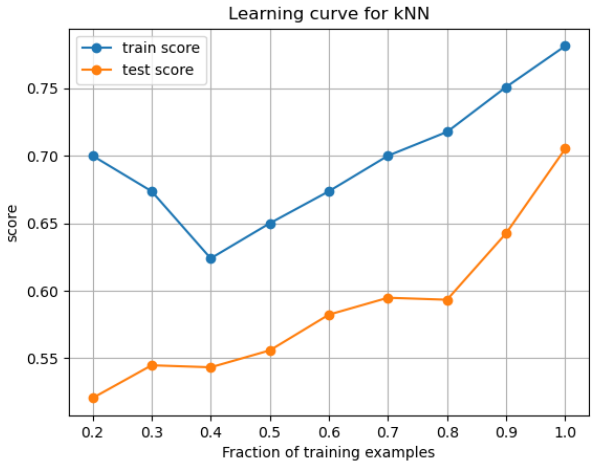

Fig. 22: BC dataset: learning curve

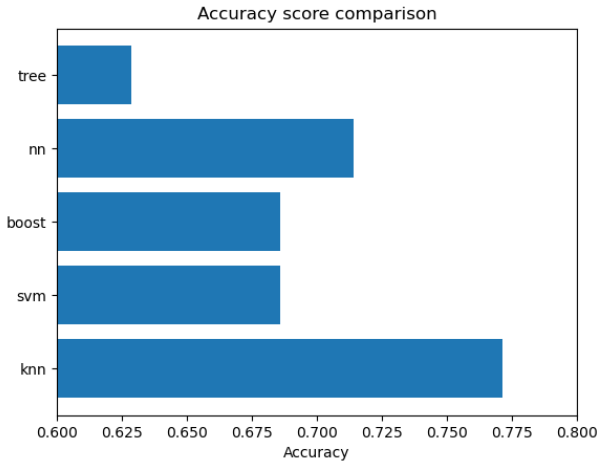

Fig. 24: BC dataset: Accuracy score

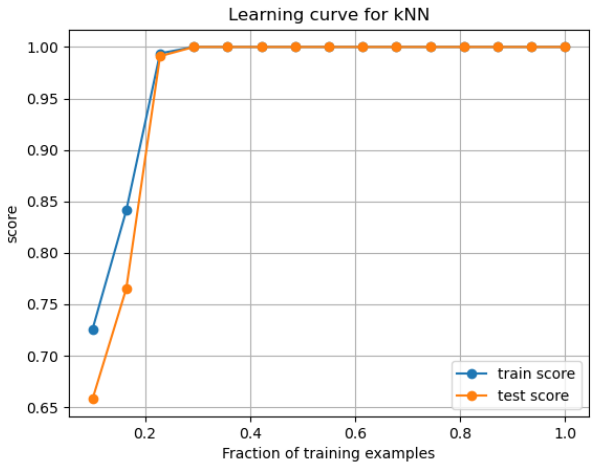

Fig. 23: DP dataset: learning curve

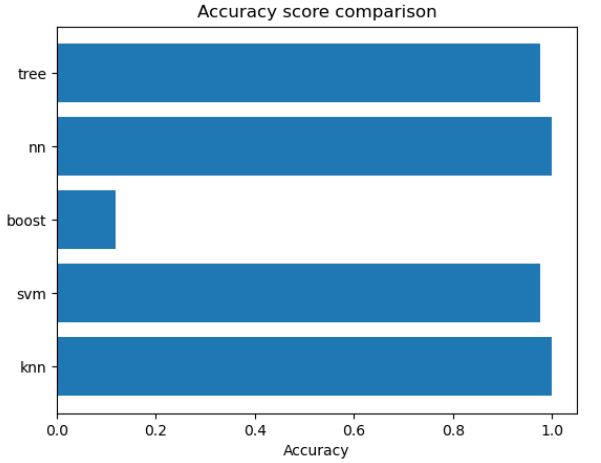

Fig. 25: DP dataset: Accuracy score

VIII. Comparison

A. *Accuracy score*

Here we use accuracy score, a common evaluation metric in machine learning which measures the overall correctness of predictions made by a classification model, for the comparison among the algorithms.

$$Accuracy = \frac{Total Numberof Predictions}{Numberof CorrectPredictions}$$

BC Dataset (figure 24): Decision tree has the lowest test accuracy because it is the least expressive among the 5 models. KNN has the best accuracy and neutral network is the second best. Adaboost and SVM have the similar accuracy. The overall accuracy is low (the best one still less than 80%) which can explain by the lack of training examples (116 entries).

DP Dataset (figure 25): The overall accuracy is high. The neural network and knn algorithm give 100% accuracy, while the decision tree and the svm give 97.6% accuracy. The boost algorithm is completely failed for this problem.

B. *Train time*

Decision trees and neural network have long training time due to the grid search. Boosting and knn have very fast training time due to no grid search in the implementation. SVM training involves quadratic optimization, which is faster than iterative algorithms like neural network backpropagation or boosting. Neural networks are the slowest due to the computational expense of gradient computations in backpropagation.

C. *Query time*

Performing queries on a decision tree entails navigating the tree structure. In both datasets, the trees are shallow, resulting in quick querying. Neural network queries are also efficient since the computation of weights is the most computationally intensive part,

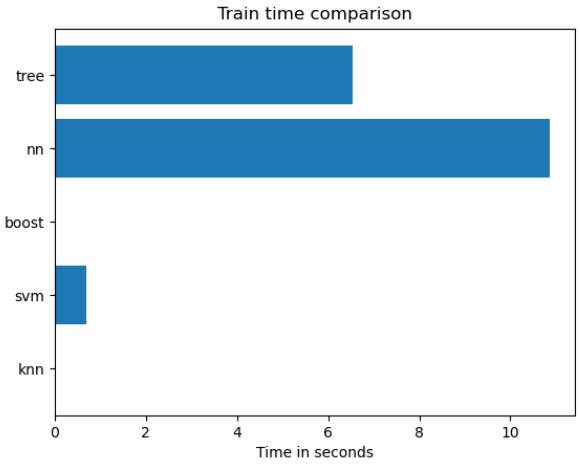

Fig. 26: BC dataset: Train time

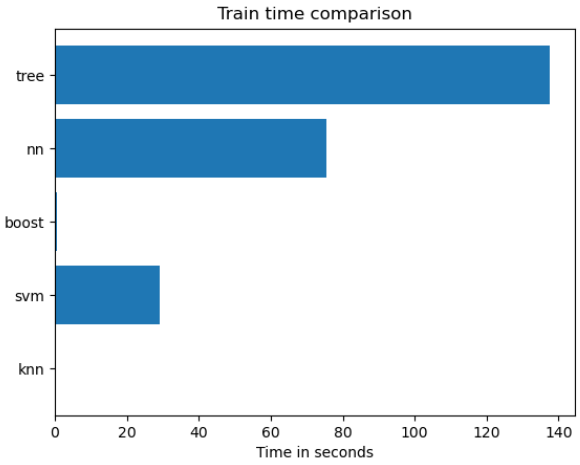

Fig. 27: DP dataset: Train time

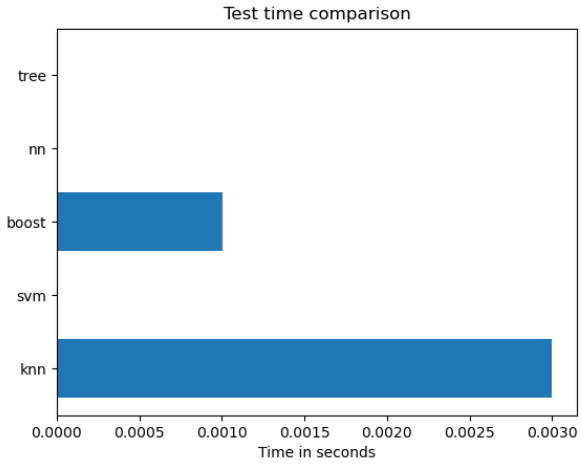

Fig. 28: BC dataset: Test time

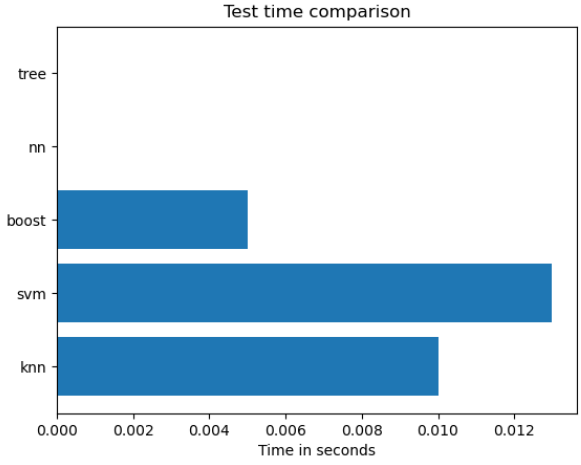

Fig. 29: DP dataset: Test time

and the forward pass is relatively fast. AdaBoost exhibits slower query times for dataset 1 due to the large number of learners. SVMs perform inference swiftly because they merely need to compute the value of a linear function. On the other hand, kNN inference is the slowest as it necessitates distance calculations with respect to every point in the training data.

IX. Conclusion

In this report, I study five common learning algorithms in machine learning: decision trees (dt), neural networks (nn), boosting (boost), support vector machines (svm), and k-nearest neighbors (knn). By studying two different data, we see that the choice of the learning algorithm depends on the specific problem, the characteristics of the data, and the trade-offs between model complexity, training and query speed. Different algorithms are more suitable for different scenarios, datasets. Thus, for a typical machine learning problem, we should try with different algorithms and find the best one through experimentation and validation. An important insight from the breast cancer dataset is that achieving higher accuracy in machine learning would require a larger amount of data.