MinhTrang Nguyen
mnguyen306
CS 7641 – Machine Learning

Project 1: SUPERVISED LEARNING

1. INTRODUCTION

Supervised machine learning is a system of algorithms that develops a prediction model using input dataset and the labeled outcomes. This paper incorporates five techniques of supervised machine learning to solve two chosen problems by training and testing the network using those various method above. My two datasets are the Phishing Website and the Sickness Detection from the openml.org website. The five supervised machine learning techniques are Decision Tree, Neural network, Boosting, Support Vector Network (SVM), and k- Nearest Neighbors (kNN). The performance result is evaluated base on the accuracy, simulation time, and the learning curve of each method.

2. DESCRIPTION OF MY CLASSIFICATION PROBLEMS

My Dataset #1 is Phishing Websites. Cybersecurity has always been my interest. Knowing how to detect a phishing website is a very useful skill to have these days when the internet is become every popular and affect a lot of people's life. Since this is my first time exploring the different supervised machine learning techniques, I purposely pick a clean dataset with no missing data and straight forward feature format. Thus, I would get more time to explore the machine learning methods better. Both features data and output data are integer of -1, 0, or 1, which makes this problem a classification problem. Even though the dataset is clean and simple, it is not a trivial problem to solve at all. The dataset contains 1354 entries, with 9 features, and 3 output classes. It is impossible for me to figure the Phishing rules.

My Dataset # 2 is Sickness Detection. This dataset is the Thyroid disease records of 3543 patients. The dataset triggers my interest is because we are at the middle of the pandemic; being able to detect disease from the patient's background and history catches my attention. If one can develop a structure that can detect Thyroid disease, other applications can be detecting Covid-19, cancers, diabetes, stroke, and other sickness. I would like to know what are the challenges and what kind of data that doctor needs to gather in order to sufficiently detect a sickness. Unlike the 1st dataset, I would like to pick a second datasets which is not as clean this time. This dataset also consists a variety type of input such as string, Boolean, numbers, and missing data values, which makes it becomes both classification and regression problem. I would also like to learn the techniques of preprocessing the data and how to clean up the data. This dataset contains 3543 data entries, with 30 features, and 2 output classes, which has three times more data and three times more features than the first data set. This dataset should be larger, and more challenging than the first one.

Another reason why I choose both datasets above is because of its clear description of the features. As I was searching for datasets for this project on a lot of websites, there are many datasets that has no descriptions of the feature column. These two datasets I picked have a well-described column title. It means as I explore the dataset and learn its feature importance, I understand the feature better and what would make a better set of features if I have to eliminate some feature in order to save computational time and system's memory.

3. SUPERVISED LEARING TECHNIQUES

This effort is to explore each method deeply and to tune the hyperparameters to find the best set up for each method before going to method compare and contrast at the next section.

A. Method 1: Decision Tree

A. Plot learning curve

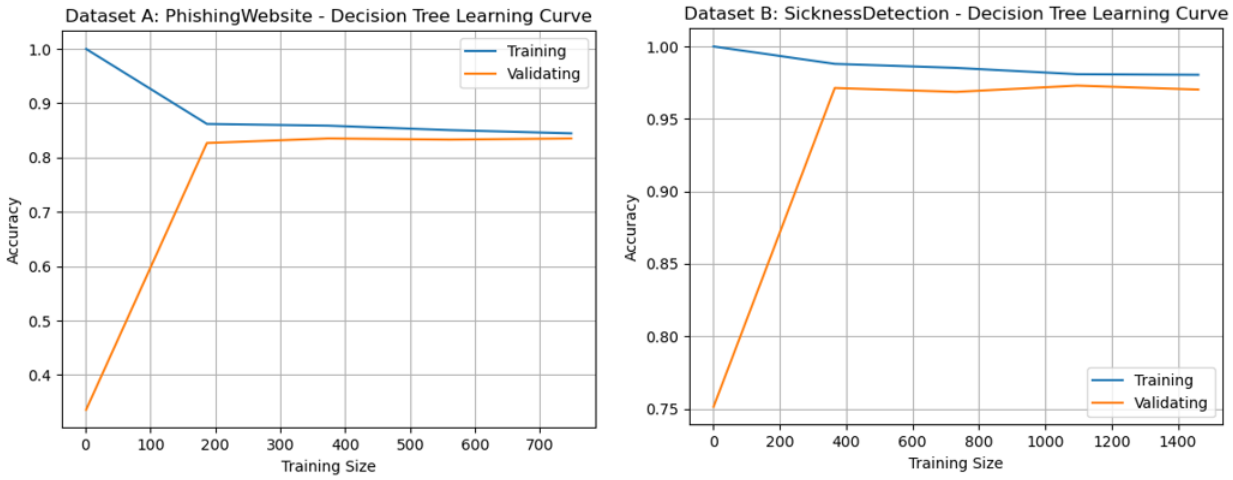

Figure 1: Method 1 - Decision Tree Learning Curve for Dataset A and Dataset B

In figure 1, in both datasets, as training size grow bigger, the gap between the training and validating are very small. It means the amount of data is enough to train, and the bias is very small.

B. Accuracy vs max_depth

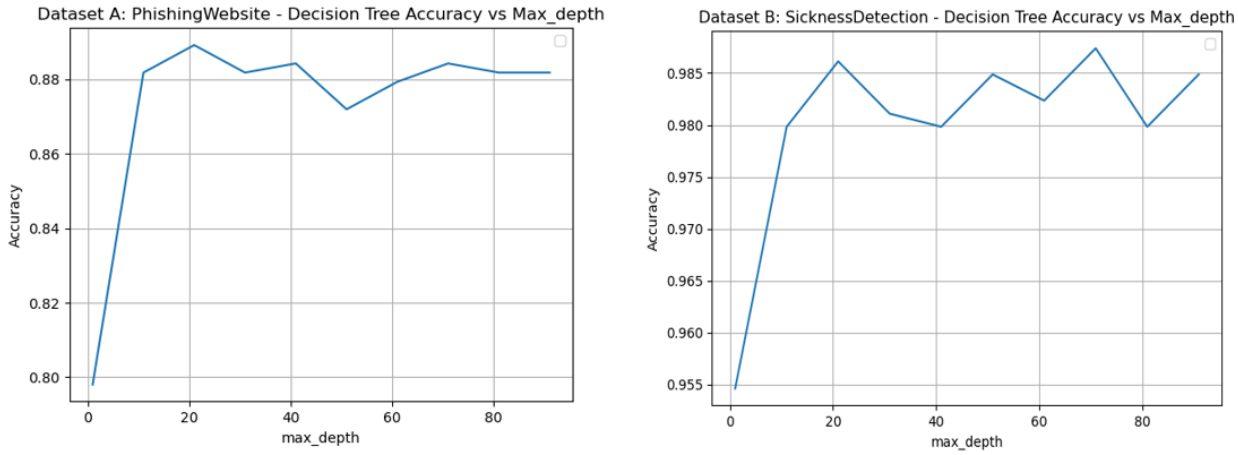

Figure 2: Method 1 - Decision Tree Accuracy vs max_depth on Dataset A and Dataset B

Max_depth is maximum depth of the decision tree. As max_depth grows larger, the nodes could get expanded until each leaf has one item. Setting the right value of max_depth can be used as a pruning technique and could help preventing overfitting data. For both datasets, max_depth=21 yield the best result.

C. Accuracy vs min_sample_leaf

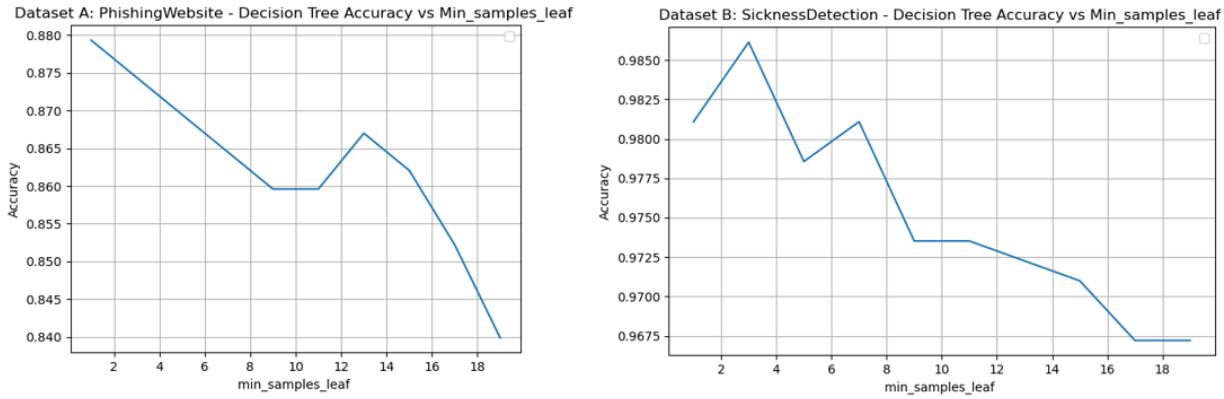

Figure 3: Method 1 - Decision Tree Accuracy vs min_samples_leaf on Dataset A and Dataset B

Min_samples_leaf is the minimum number of samples required to split the node. This is also another technique to prune the decision tree and prevent overfitting. For both dataset, if the min_samples_leaf value is too high, the accuracy decrease it requires too many sample to split the node and many nodes did not get split when it supposed to. The min_samples_leaf for dataset B is 3 instead of 1 help preventing overfitting.

D. Accuracy vs max_features

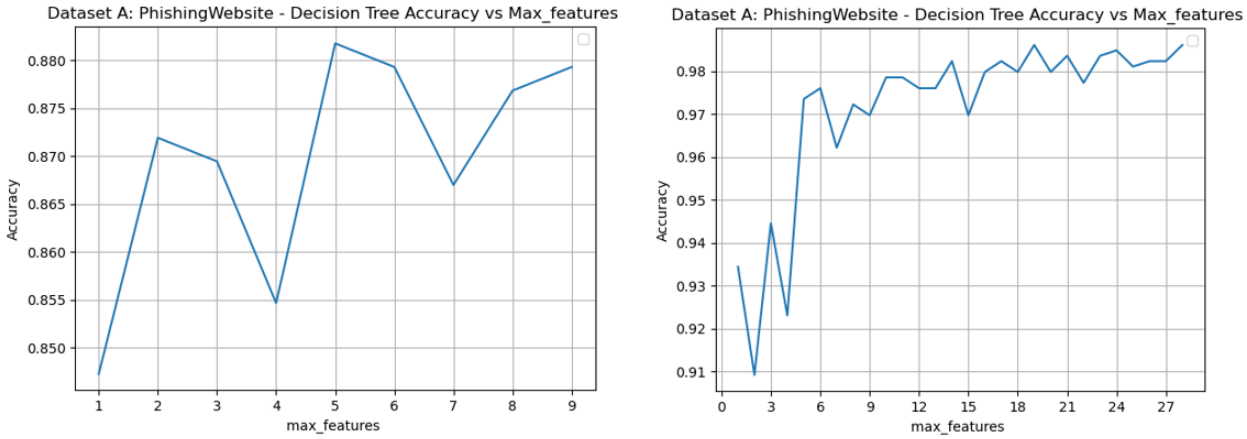

Figure 4: Method 1 - Decision Tree Accuracy vs max_features on Dataset A and Dataset B

Max_features is the number of features consider when searching for the best split. Base on the definition, the larger the number of the features seem to be better in this case. It makes sense because both datasets don't have a large number of features; therefore, it will be helpful to use more number of features.

E. Feature Importance

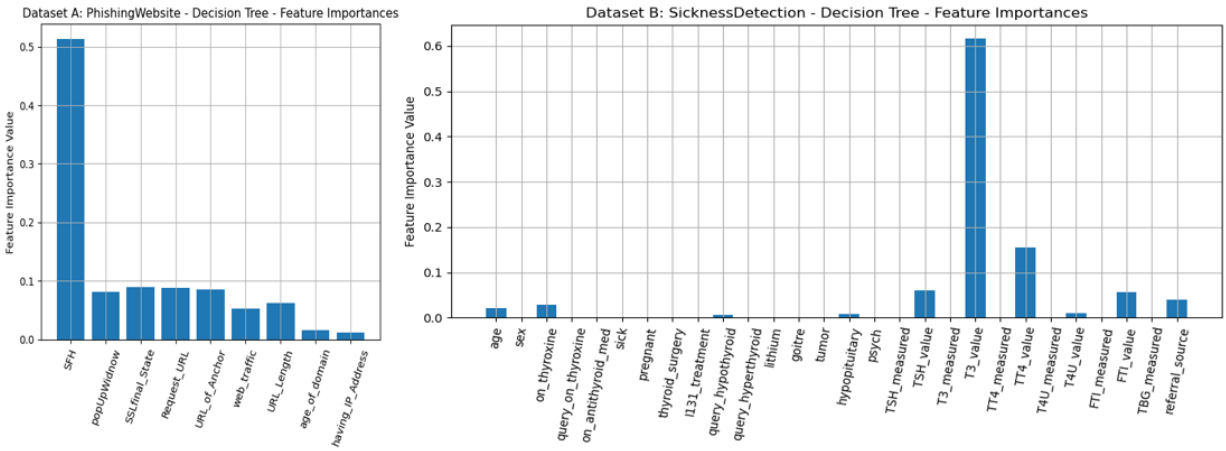

Figure 5: Method 1 - Decision Tree Feature Importance on Dataset A and Dataset B

Another interest tool I learn from this project is the feature importance ranking tool. For dataset A, the SFH (Server Form Handler) feature is the most stand out. For dataset B, the T3_value feature is the most importance. For this project, if I want to improve my simulation time by cutting off some features, I can pick the top 10 features from the feature importance result to do that. It's a very useful tool.

B. Method 2. Neural Network

For the second method – Neural Network, I use the MLPClassifier which implements a multi-layer perceptron (MLP) algorithm. MLP is a non-linear model using backpropagation to train the data.

1. Plot learning curve

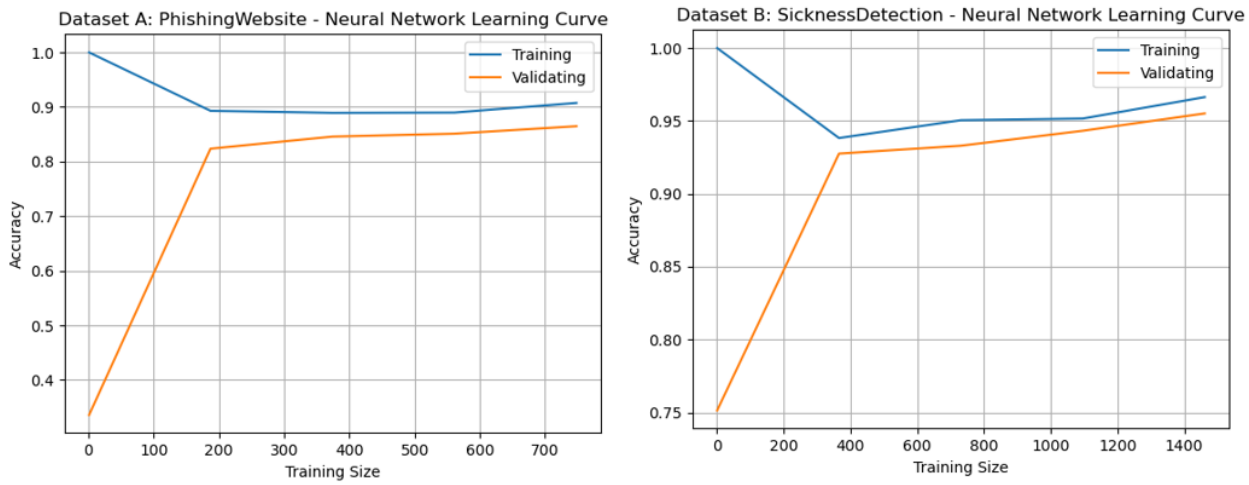

Figure 6: Method 2 – Neural Network Learning curve on Dataset A and Dataset B

In figure 6, in both datasets, as training size grow bigger, the gap between the training and validating are very small. It means the amount of data is enough to train, and the bias is very small.

2. Accuracy vs hidden_layer_sizes

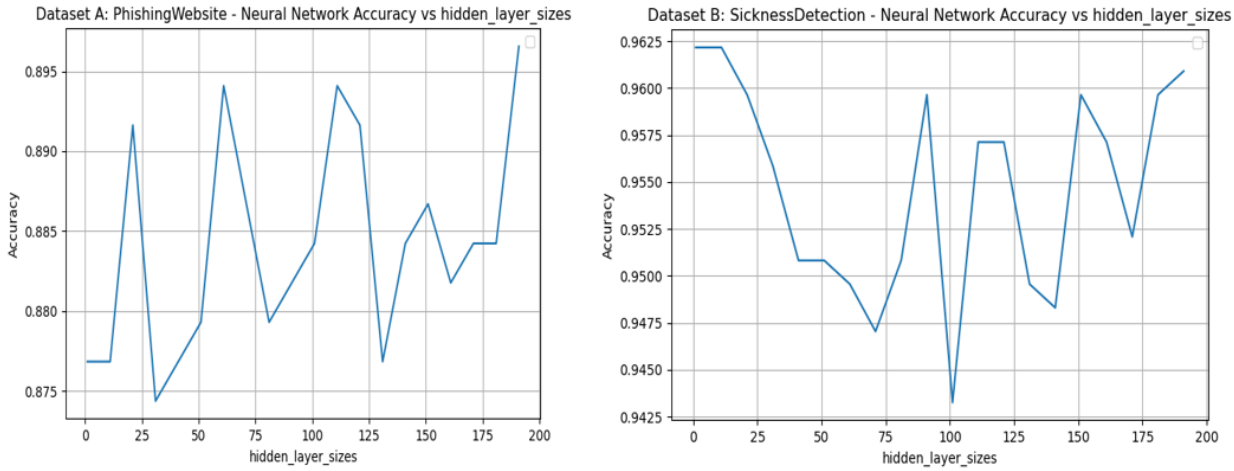

Figure 7: Method 2 – Neural Network Accuracy vs hidden_layer_sizes on Dataset A and Dataset B

The hidden_layer_sizes is the number of hidden layers in the neuron network. For both cases, increasing the number of hidden layer yield fluctuated result. For dataset A, number of hidden layer = 19, or 60, or 110 yield the best performance. For dataset B, number of hidden layer = 10, or 80, or 110 yield the best performance.

3. Accuracy vs max_iter

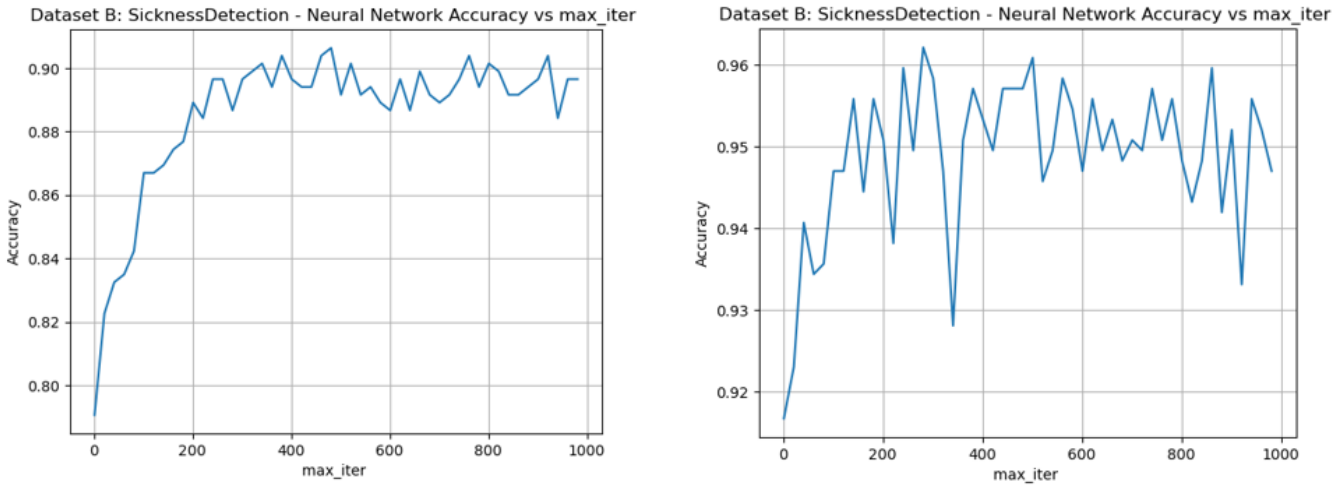

Figure 8: Method 2 – Neural Network Accuracy vs max_iter on Dataset A and Dataset B

Max_iteration is the number of iteration that solver will continue to iteration, unless it converged. If this number is too small, the solver will never reach convergent. In both of my datasets, max_iter need to be > 200 to have a better performance.

4. Accuracy vs learning_rates

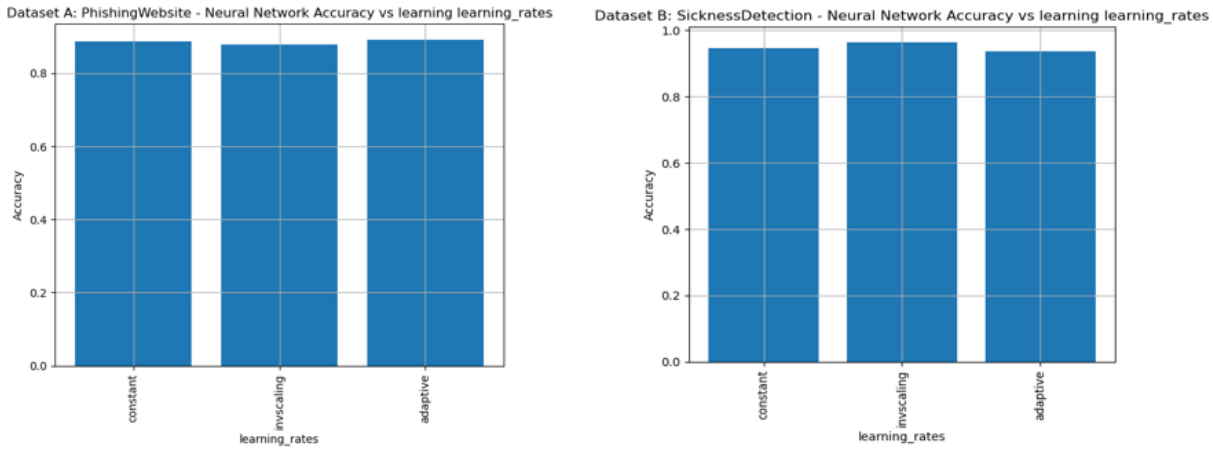

Figure 9: Method 2 – Neural Network Accuracy vs learning_rates on Dataset A and Dataset B

All three learning rates yield very similar result in both of my datasets. I assume that learning rate has a lower impact than other hyper parameter in this Neural Network set up.

5. Accuracy vs batch_size

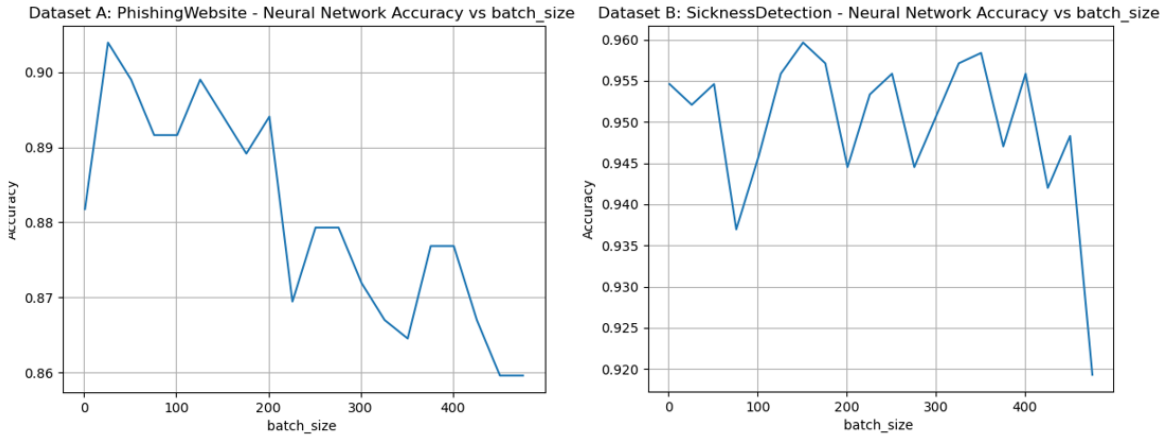

Figure 10: Method 2 – Neural Network Accuracy vs batch_size on Dataset A and Dataset B

Batch_size is the size of mini batches for the solver. In dataset A, since there is 1354 data point, a mini batch size of less than 200 is good. As the batch_size increase, dataset A did not have enough data to train, which leads to the accuracy decreases. In dataset B, as long as the mini batch size is < 450, the solver still can keep up with a decent accuracy value.

C. Method 3. Boosting

I'm using the AdaBoostClassifier for this method. AdaBoost is designed to adjusting the different weight on data id it is incorrect in order to focus more on those difficult cases.

1. Accuracy vs learning_rate

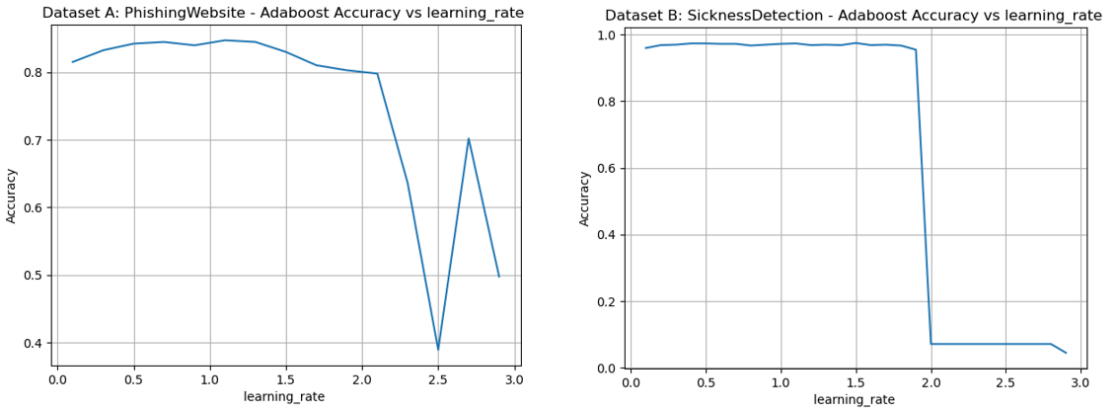

Figure 12: Method 3 – Boosting Accuracy vs learning_rate on Dataset A and Dataset B

The learning rate control how much data from the previous existing model going to contribute to the new model. Therefore, when this number is too high, the performance goes down because it won't care much about previous steps anymore. In both datasets, learning rate of < 2 would provide a good accuracy. A learning rate of > 2 can decrease the performance dramatically.

2. Accuracy vs n_estimators

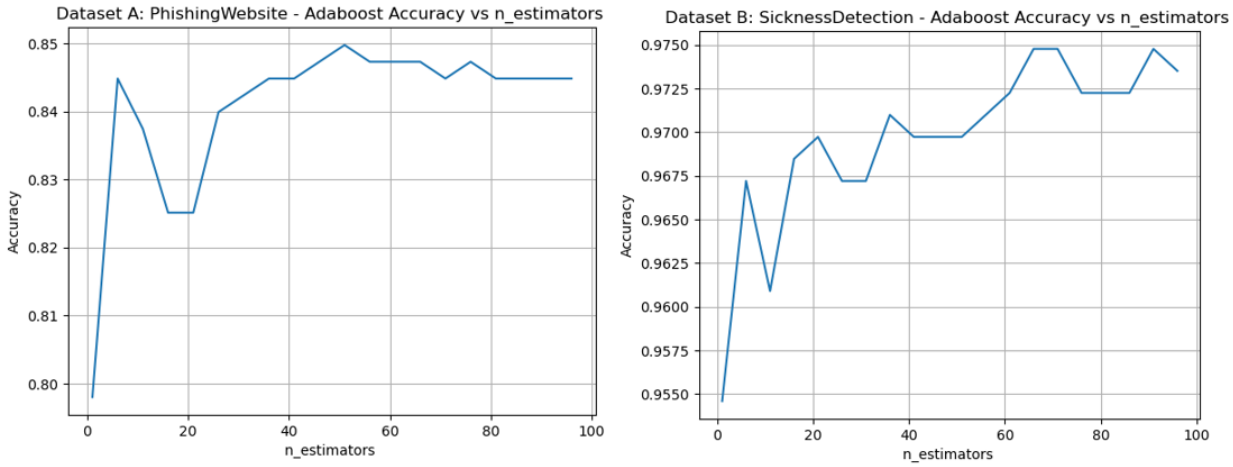

Figure 13: Method 3 – Boosting Accuracy vs n_estimators on Dataset A and Dataset B

N_estimators is the maximum numbers of estimators to eliminate boosting. In dataset A, a n_estimators of 50 would work best. In dataset B, a n_estimators of 62 would work best for this algorithm.

3. Feature Importance

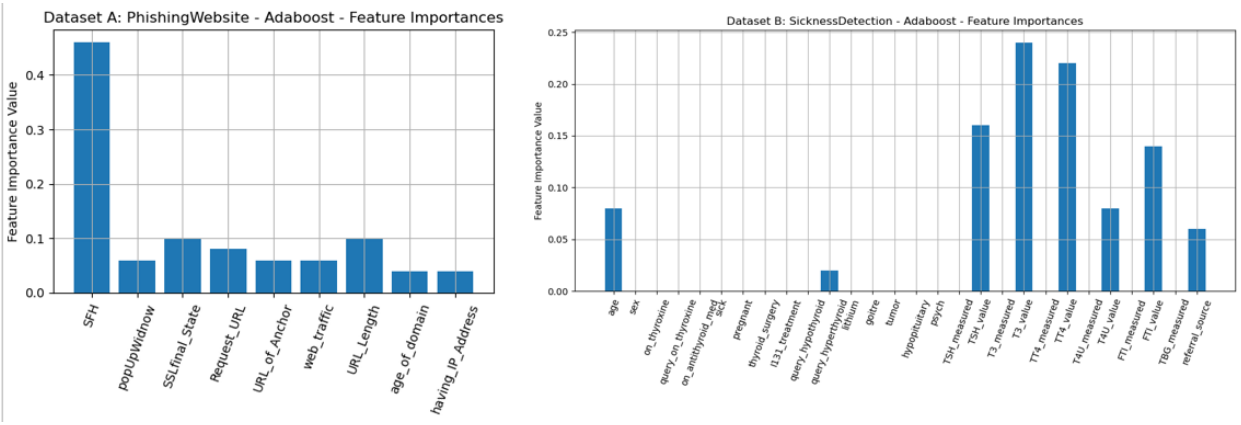

Figure 14: Method 3 – Boosting Feature Importance on Dataset A and Dataset B

Feature importance in this technique also agrees with the Decision Tree technique.

D. Method 4: Support Vector Machine (SVM)

In this method, I'm using the SVC (Support Vector Classification) function kernel to solve both datasets.

1. Accuracy vs C_param

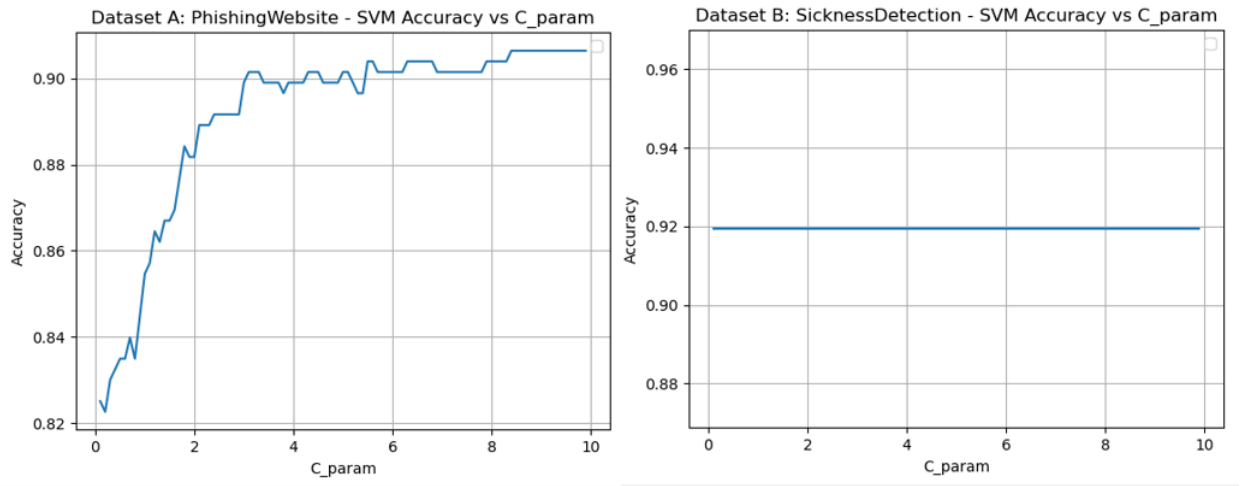

Figure 16: Method 4 – SVM Accuracy vs C_param on Dataset A and Dataset B

The C parameter is the regularization parameter. Since both of my data is not noisy, I can set C parameter to a value higher than 1 to increase the accuracy.

2. Accuracy vs max_iter

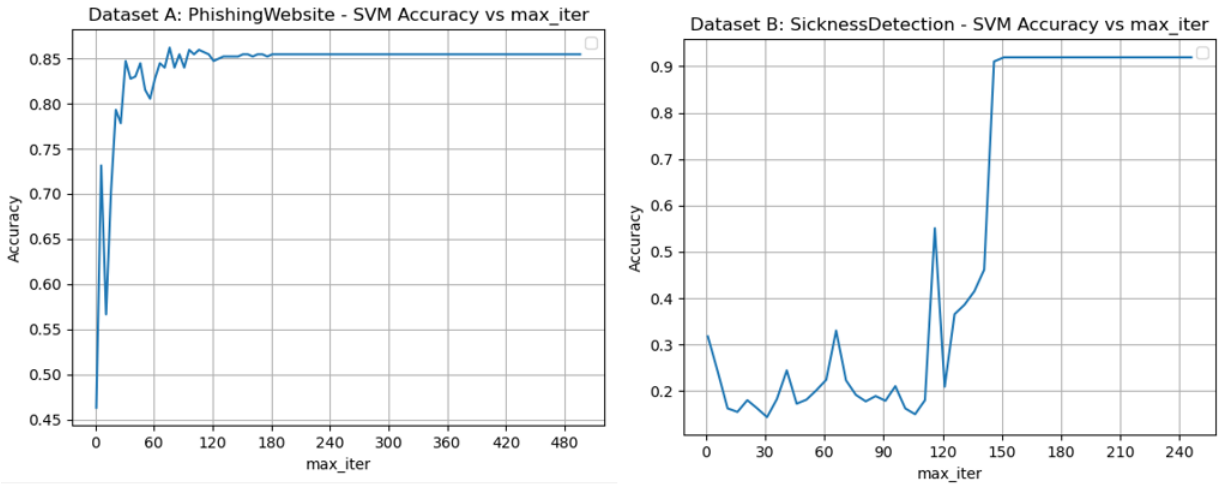

Figure 17: Method 4 – SVM Accuracy vs max_iter on Dataset A and Dataset B

Max_iter is to set the hard limit of the solver. If the hard_limit is set too low, the solver will stop too early. Therefore, in both of my cases, a max_iter of > 150 is a good set up to make sure that the solder have enough iterations to go through all cases needed.

E. Method 5: K- Nearest Neighbor (kNN)

This method implements prediction base on the k number of nearest neighbor vote.

1. Accuracy vs n_neighbors

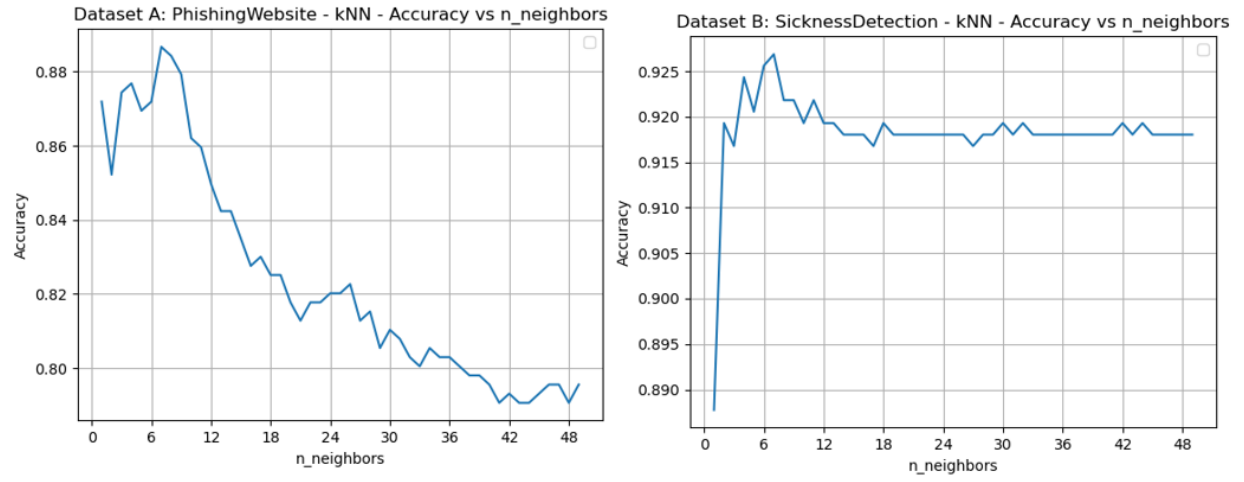

Figure 19: Method 5 – kNN Accuracy vs n_neightbors on Dataset A and Dataset B

The n_neighbors is the most important hyper-parameter of this technique. Tuning the number of neighbors can make a huge different on the performance. For dataset A, n_neighbors = 7 yield the best accuracy of 0.887 compare to n_neighbors = 99, the accuracy would be 0.79, which is 14% decrease in accuracy. In dataset B, k_neighbor = 7 yield the best accuracy of 0.9258.

2. Accuracy vs leaf_size

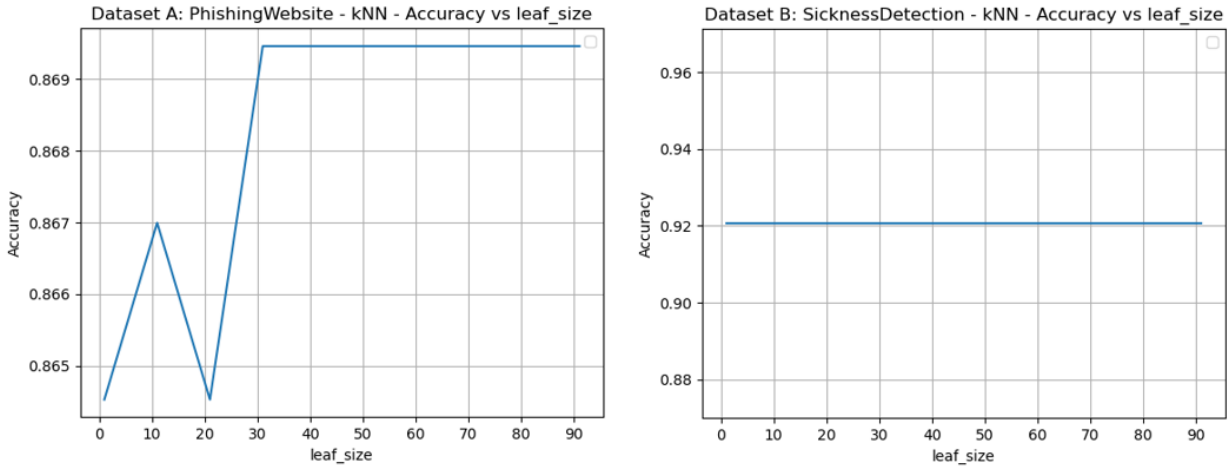

Figure 20: Method 5 – kNN Accuracy vs leaf_size on Dataset A and Dataset B

Leaf_size is the number of leaf pass to the tree. It can affect the speed of the query. For dataset A, leaf_size < 30 would worsen the performance. However, in dataset B, leaf_size seems not making any different in the performance.

4. METHOD COMPARE AND CONTRAST

A. Best Performance

The best performance analysis was developed by comparing 5 methods accuracy with its best set up. The best set up was gather from the hyperparameters tuning described in the previous part. According to Table 1, the neural network method produces the best accuracy (0.937) for Dataset A. Base on the max_iter analysis above, dataset A has enough amount of data for the neural network to learn the prediction model well. Another reason neural network performs better in this case is when I look at the feature importance, beside one feature that is very importance, all other features share similar importance values which makes this problem contain more information dimensions. Since the problem is more complicated, neural network method can connect and discover the features relationship to output better.

For dataset B, the decision tree with pruning method produces the best accuracy (0.986). Decision Tree is doing better for dataset B is because this dataset is clumsier with a lot of features. The features are also non-linear separable which make the Decision Tree method perform better than the SVM and the kNN method.

#	Method	Dataset A: Phishing Websites		Dataset B: Sickness Detection	
		Best Performance set up	Best accuracy	Best Performance set up	Best accuracy
1	Decision Tree	Max_depth=21, Min_leaf=1, max_feature=5	0.892	Max_depth=21, Min_leaf=3, max_feature=28	**0.986**
2	Neural Network	hidden_layer=23, learning_rate='adaptive', batch_size=16	**0.937**	hidden_layer=12, learning_rate='invscaling', batch_size=150	0.961
3	Adaboost	learning_rate=1.2 , n_estimator=52	0.851	learning_rate=1.5, n_estimator=65	0.975
4	SVM	C_param=8.2 , max_iter=180	0.912	C_param=[1:10] , max_iter=>150	0.921
5	kNN	n_neighbor= 7, leaf_size=31	0.889	n_neighbor=7 , leaf_size=10	0.922

Table 1: Best Performance on Dataset A and Dataset B

B. Learning Curves (Training/ Testing Error rate)

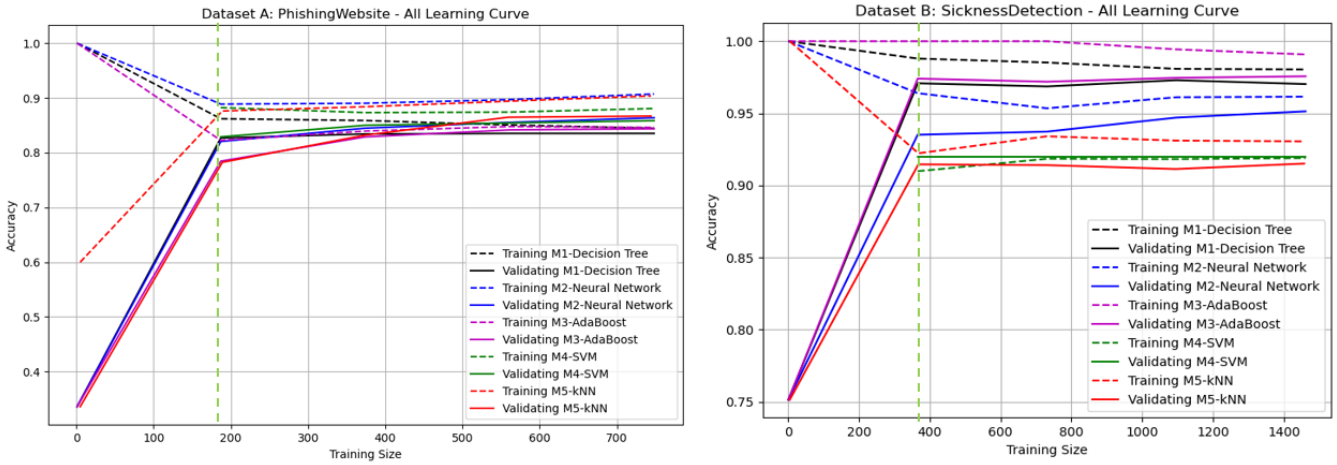

Figure 21: Learning Curves of all methods on Dataset A and Dataset B

The training and testing errors rate concept is similar to learning curve. Figure 21 is the compare and contrast of the accuracy performance as a function of training size yielded from five techniques. For dataset A, all methods' learning curve agree that the performance is best when the training size is greater than 185. In dataset B, all methods' learning curve agree that the performance is best when the training size is greater than 285. I notice that all methods have a similar break point where all agrees on how much training data would benefit the model. All methods agree that both datasets have enough training data for the model to learn proficiently. In dataset A, the gap of the training and validation curve for both datasets are very close together means my methods' set up are not overfitting and there is no noticeable bias issue. In dataset B, all the training score (dash line) are higher than validating scores (solid line), the model may suffer from over fitting problem. However, those gaps are small enough for me to move forward with those result. Both of these datasets may suggest that the problems I pick are easy problems or my problem is not very noisy.

C. Simulation Run Time

#	Method	Dataset A: Phishing Websites		Dataset B: Sickness Detection	
		Best Performance set up	Sim time (s)	Best Performance set up	Sim time (s)
1	Decision Tree	Max_depth=21, Min_leaf=1, max_feature=5	0.00315	Max_depth=21, Min_leaf=3, max_feature=28	0.003865
2	Neural Network	hidden_layer=23, learning_rate='adaptive', batch_size=16	3.55481	hidden_layer=12, learning_rate='invscaling', batch_size=150	0.90198
3	Adaboost	learning_rate=1.2 , n_estimator=52	0.26275	learning_rate=1.5, n_estimator=65	0.17109
4	SVM	C_param=8.2 , max_iter=180	0.01462	C_param=[1:10] , max_iter=>150	0.043017
5	kNN	n_neighbor= 7, leaf_size=31	0.01787	n_neighbor=7 , leaf_size=10	0.054514

Table 2: Simulation Time (seconds) on Dataset A and Dataset B

In term of simulation time, the neural network takes the longest to train for both data set while the decision takes the shortest to train. It makes sense since the decision tree is known to be very simple and straight forward compare to the neural network. In decision tree algorithm, each branch would have a smaller number of data as it goes deeper, which makes decision tree's computation similar to a binary search. Decision tree also performs pruning which help to cut down a lot of data points to process. Neural network involves optimization algorithm to find a set of weights, which can be high multidimensional. Therefore, neural network is computationally more expensive than decision tree and other methods.

D. Which method is the best?

In my opinion, for dataset A, even though Neural Network method yields the highest accuracy value, it took hundreds and thousands time more than other methods. Since all other accuracy value are close to each other, I would pick the SVM method to be the best method to use for dataset A. SVM's accuracy is 0.912, which is 2.6% less than Neural Network's accuracy, and SVM's simulation time is only 0.01462 second, which is 243 times faster than the Neural Network method. The SVM method is a good balance for both accuracy and simulation time. For dataset B, the Decision Tree method wins both accuracy and the simulation time analysis. Therefore, Decision Tree would be the best for dataset B problem.

5. REFERENCE

- https://www.openml.org/d/4534
- https://www.openml.org/d/38
- https://scikit-learn.org/stable/modules/tree.html
- https://scikit-learn.org/stable/modules/neural_networks_supervised.html
- https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.AdaBoostClassifier.html
- https://scikit-learn.org/stable/modules/svm.html
- https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html