Assignment #1 (sbakshi35)

Data

The objective of this assignment is to utilize well-established supervised learning techniques on two datasets that are 'interesting'. The datasets that I have chosen for this assignment are as follows:

1. Breast Cancer Wisconsin (Diagnostic) Data Set: This is a UCI ML dataset from the medical field which is suited for classification tasks. There are 569 instances in this dataset, and a relatively large number of attributes (32) for a dataset with relatively fewer instances. The features are characteristics computed from digitized images of fine needle aspirates of breast mass. Contrary to what is the norm for medical datasets, this set is relatively balanced, with ~63% benign cases and the rest as malignant (treated as 1 in the dataset).
   This dataset is quite commonly used for testing the performance of supervised classification algorithms, and can be used as a baseline for analyzing the performance of the five chosen algorithms. In the rest of the assignment, this dataset is referenced as BCW.
2. Rice (Cammeo and Osmancik) Data Set: Part of the UCI ML Data Repository, the rice dataset is a larger dataset compared to the Breast Cancer one, with 3810 instances, but has fewer features (8 features describing the characteristics of the rice grain). The two classes of rice, Cammeo and Osmancik, are very similar in terms of characteristics – both types of grains are wide and long, and have a glassy and dull appearance, and therefore, the classification problem is not trivial.
   I chose this dataset primarily to present a contrasting dataset compared to the Breast Cancer dataset. This dataset is more balanced, with ~57% of the cases being rice of the Osmancik variety (treated as 1 in the dataset). In the rest of the assignment, this data set is referenced as RCO.

Overall, both these datasets can provide valuable insights into the performance of the five chosen classification algorithms – decision trees, neural networks, support vector machines, k-nearest neighbors and boosting. For both the datasets, the features are scaled and a train-test split is used to generate the training and test sets.

Decision Trees

The first algorithm we will use is decision trees. In the implementation on scikit-learn, Gini impurity is used as the default criterion for measuring the quality of split at every node. There are multiple hyper-parameters that can be tuned for use within the scikit-learn decision tree framework. In this assignment, I have chosen the two parameters that help prevent overfitting in decision trees, since decision trees are quite prone to overfitting (especially large trees with more depth). The first parameter is the max depth of the tree, which is a pre-pruning technique. The second parameter is the α (regularization) parameter for minimal cost-complexity pruning, and is a post-pruning technique. It finds the node which forms the 'weakest link', and such nodes are recursively removed. Higher values of α leads to more nodes being pruned out from the tree.

We first look at the validation curves for each of the parameters for each dataset:

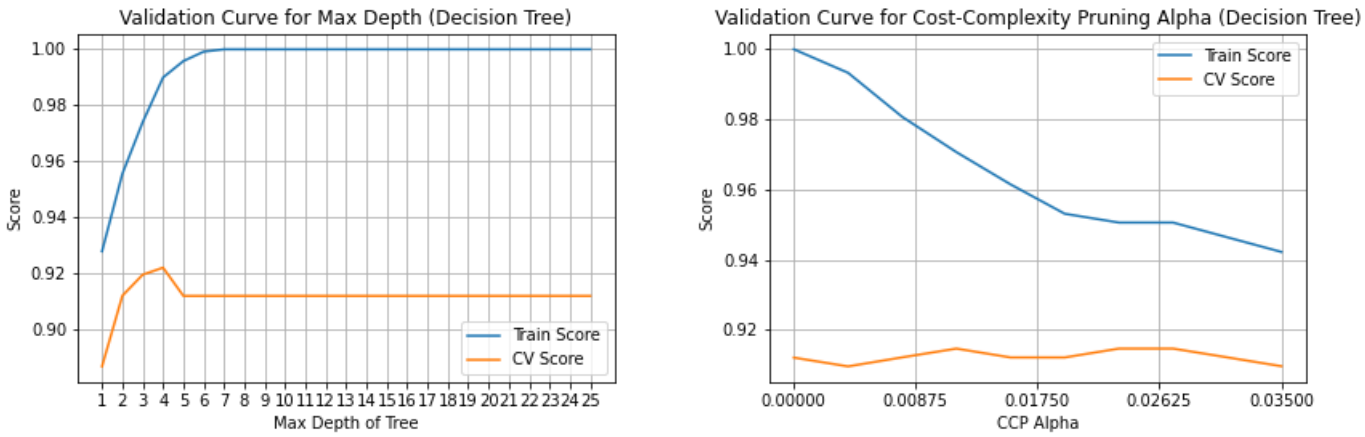


Fig. 1. Validation curves for the BCW dataset.

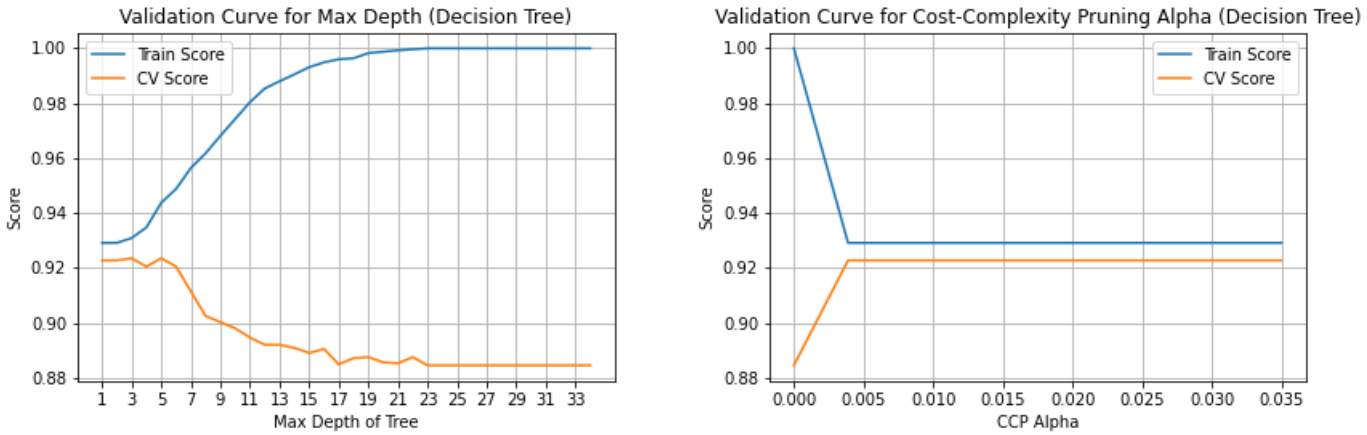


Fig. 2. Validation curves for the RCO dataset.

In both the datasets, the training score increases with the max depth parameter. This is expected as the model starts to overfit as the tree becomes larger. However, as seen in the cross-validation score curves, they either remain at the same level or decrease and then stabilize. This also indicates that the model is overfitting for higher values of max depth.

For the CCP α parameter, the training score becomes lower at higher values of α. This is expected as an increase in in the parameter value leads to more pruning in the tree. The cross-validation scores remain relatively stable over the range of chosen α.

We complete a grid search using 4-fold cross-validation over these two parameters to determine the best combination of hyper-parameters. These 'optimal' hyper-parameters are presented in the following table, where Px stands for the predicted values and Ax stands for the actual values:

Dataset	Max. Depth	CCP α	Accuracy (Test Set)	Confusion Matrix		
BCW	3	0.0039	0.9649		PF	PT
				AF	106	2
				AT	4	59
RCO	3	0	0.9300		PF	PT
				AF	467	51
				AT	29	596

The learning curves for the decision tree classifier are presented for each dataset as well. The interesting outcome here is that for the BCW dataset, the learning curve shows a large gap between the training and cross-validation scores. This means that there is high variance in the model, and this implies that we need more data, or we need to choose a model with fewer features (less complex model) if possible. However, in the case of the RCO dataset, the gap between the train score and the CV score is relatively lower at higher percentage of training examples, which means that since the final converged score can be higher, there might be bias in the model, and it could benefit from increased model complexity.

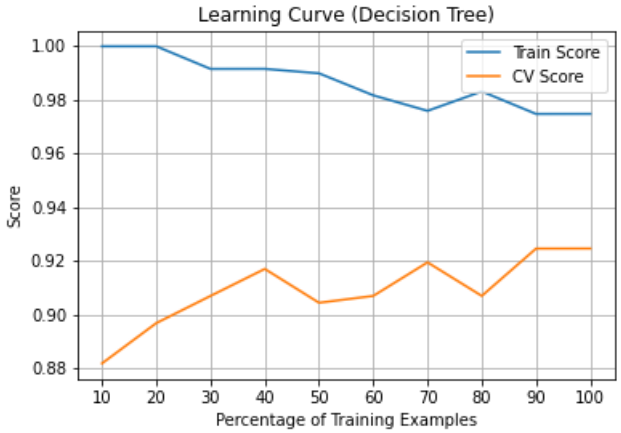


Fig. 3. Learning Curve for Decision Tree on BCW.

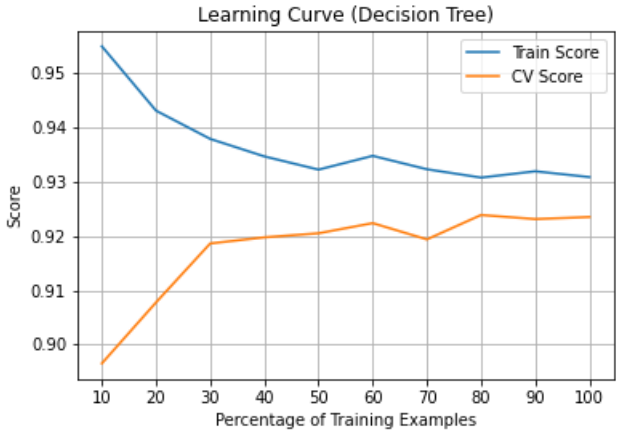


Fig. 4. Learning Curve for Decision Tree on RCO.

Neural Networks

For the neural network classifier, we use a multi-layer perceptron network with two hidden layers, each with 4 nodes. The activation function used for the MLP classifier is the ReLU unit. The solver used in this assignment is Adam, a stochastic gradient descent optimizer.

Two important hyper-parameters that affect the performance of the MLP classifier significantly are the learning rate and the regularization parameter. Validation curves for these parameters on the BCW and RCO datasets are shown below:

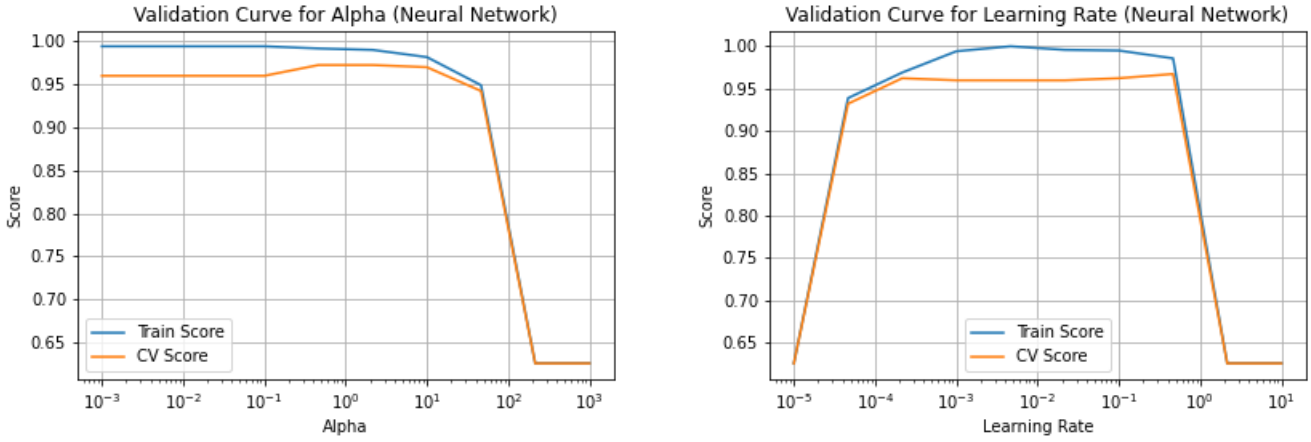


Fig. 5. Validation curves for the BCW dataset.

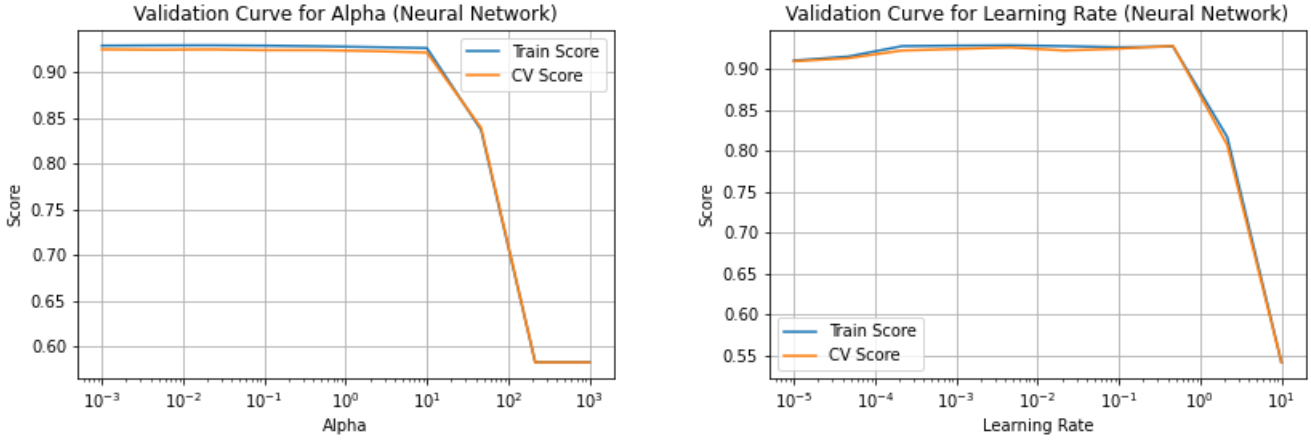


Fig. 6. Validation curves for the RCO dataset.

Both sets of validation curves are shown on semi-log X-Y plots. It can be seen that for both the datasets, high α value tends to be detrimental to the performance of the MLP classifier as high α introduces more bias in the model. For the learning rate parameter, low learning rate leads to slow convergence and low training/CV scores, and high learning rate leads to instability and non-converging results as well.

Grid search again helps us determine the best set of hyper-parameter values for the MLP classifier. The results are shown in the table below. The MLP classifier performs very well on the BCW dataset, with over 99% accuracy on the test set and only a single misclassification as seen in the confusion matrix. On the other hand, the MLP classifier surprisingly does not do well on the RCO dataset, failing to outperform even the simple decision tree algorithm.

Dataset	α	Learning Rate	Accuracy (Test Set)	Confusion Matrix
BCW	2.1544	0.1	0.9942	<table><tr><td></td><td>PF</td><td>PT</td></tr><tr><td>AF</td><td>108</td><td>0</td></tr><tr><td>AT</td><td>1</td><td>62</td></tr></table>
RCO	0.1	0.1	0.9221	<table><tr><td></td><td>PF</td><td>PT</td></tr><tr><td>AF</td><td>447</td><td>71</td></tr><tr><td>AT</td><td>18</td><td>607</td></tr></table>

If we look at the learning curves for the two datasets, this discrepancy in performance can possibly be explained. For the BCW dataset, we can see that both the training and cross-validation scores converge at a high score for high percentage of training examples. This means that the model has neither bias nor variance, and would be regarded as a classifier with good performance on the dataset. However, on the RCO dataset, both scores converge to a relatively lower value, indicating that the model has bias, and a model with higher complexity will possibly perform better on the given dataset.

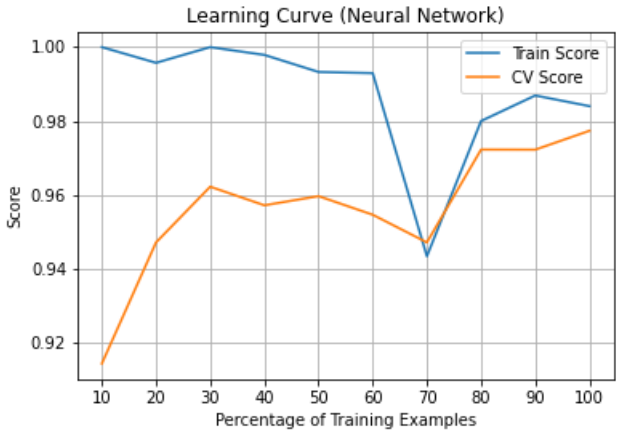


Fig. 7. Learning Curve for Neural Network on BCW.

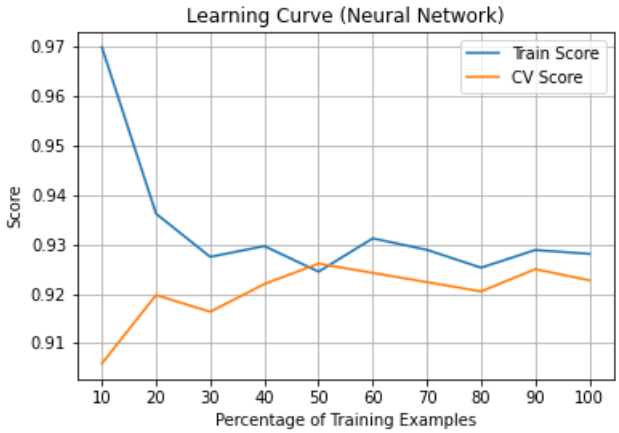


Fig. 8. Learning Curve for Neural Network on RCO.

We can also look at the loss curves and the training and cross-validation score curves versus the number of epochs. This shows how the training of the neural network progresses with epochs.

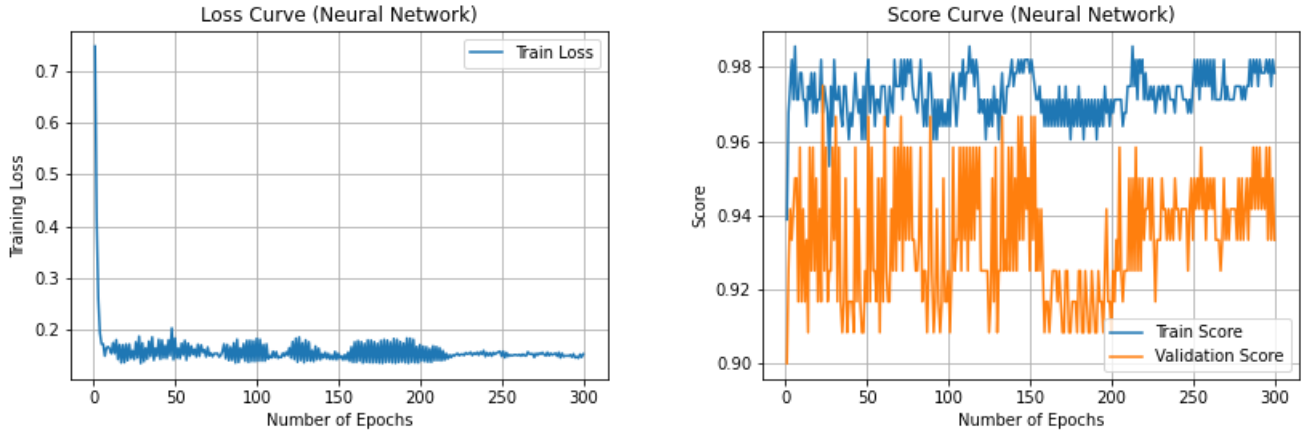


Fig. 9. Loss and score curves for the BCW dataset.

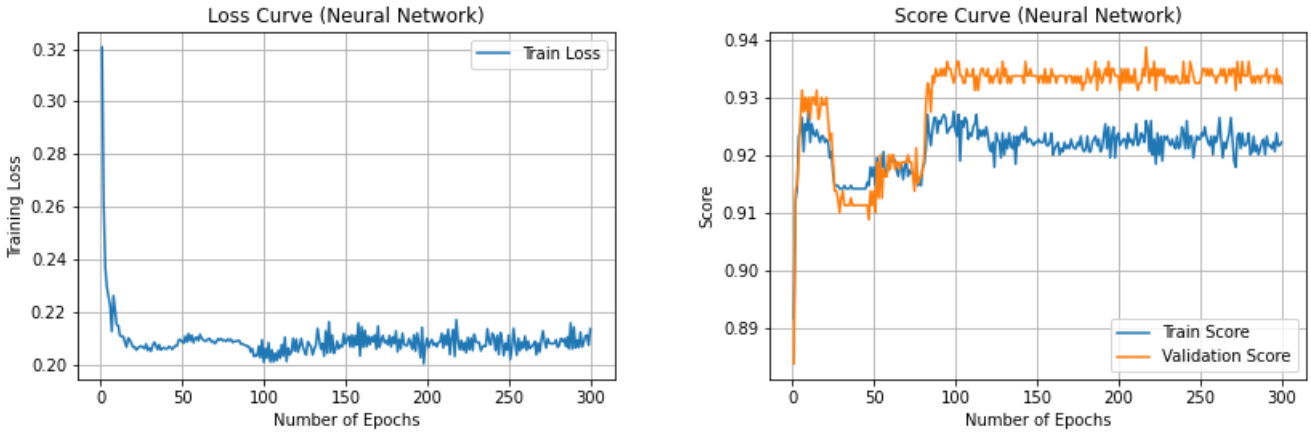


Fig. 10. Loss and score curves for the RCO dataset.

One interesting point to note in these curves is that the validation score is more than the train score in the case of the RCO dataset. Additionally, the fact that the scores converge to a relatively low value also portray the bias in the model.

We also try out the single layer neural network, where the number of hidden units in the layer is treated as a validation parameter. For this case, we also use the regularization term α as the other hyper-parameter. The validation curves are shown below. The validation curves in this case are not very conclusive as the remain quite stable over the entire range of chosen parameter values.

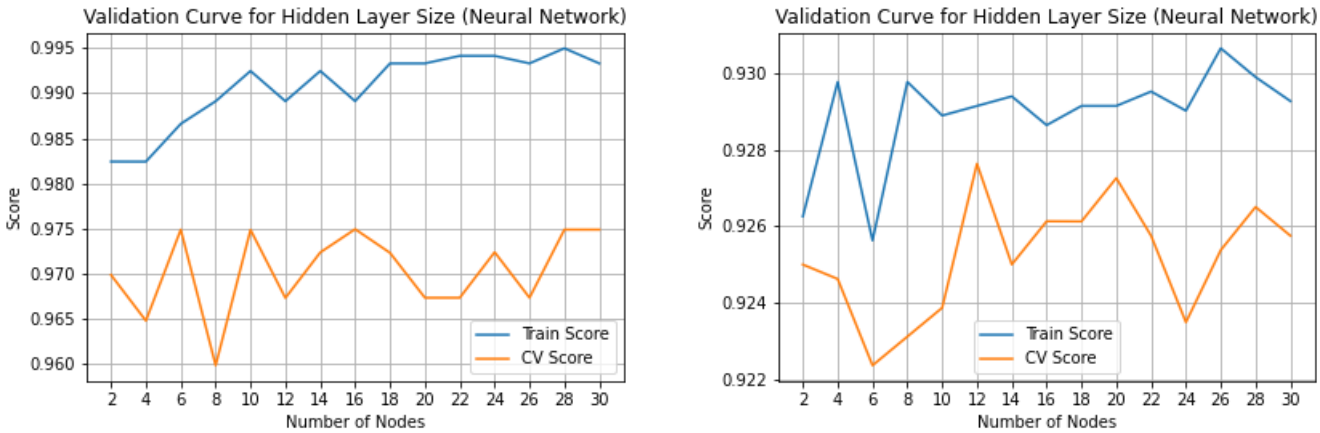


Fig. 11. Validation curves for the BCW and RCO dataset respectively.

The results for the single hidden layer neural network are shown in the table below. For the BCW dataset, the single layer network performs exactly the same as the two-layer network, and even has the same value of α. For the RCO dataset, the results are slightly better than the two-layer network, with a better distribution of false positives and negatives.

Dataset	α	Hidden Units	Accuracy (Test Set)	Confusion Matrix
BCW	2.1544	16	0.9942	\ PF PT AF 108 0 AT 1 62
RCO	0.4642	12	0.9318	\ PF PT AF 472 46 AT 32 593

Finally, the learning curves are also presented for this new model. For the BCW dataset, the learning curve shows convergence at high scores for high percentage of training examples, which indicates that the model is a good classifier without bias or variance. For the RCO dataset, the learning curve shows bias in the model as the train and CV scores have converged to a relatively low value with high percentage of training examples. This can probably be solved by using a model of higher complexity. This is expected as a more complex two-layer model also faced the same issue.

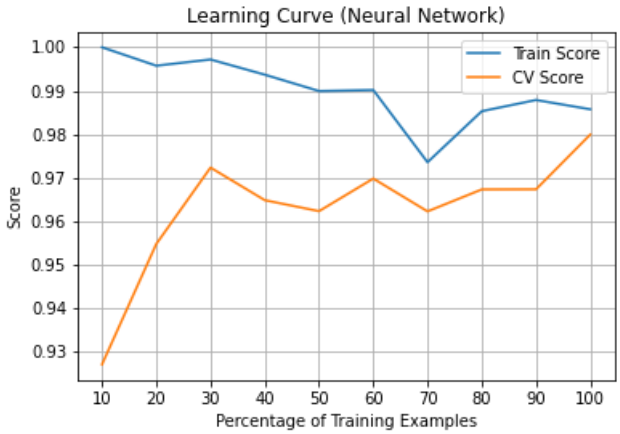


Fig. 12. Learning Curve for Neural Network on BCW.

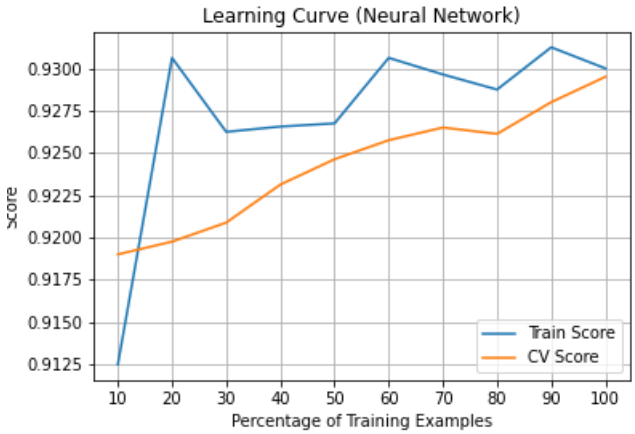


Fig. 13. Learning Curve for Neural Network on RCO.

k-Nearest Neighbors

For the k-NN classifier, we use the scikit-learn implementation with uniform weights for prediction. Just like the other classifiers, we look at the validation curves for the two dataset for the two chosen hyper-parameters: the most obvious choice in this case is the actual number of neighbors k, while the second hyper-parameter that I have chosen is the power of the distance metric used while calculating the distance to neighbors. This is also an interesting hyper-parameter, especially when dealing with high dimensional data (larger number of features) due to the curse of dimensionality.

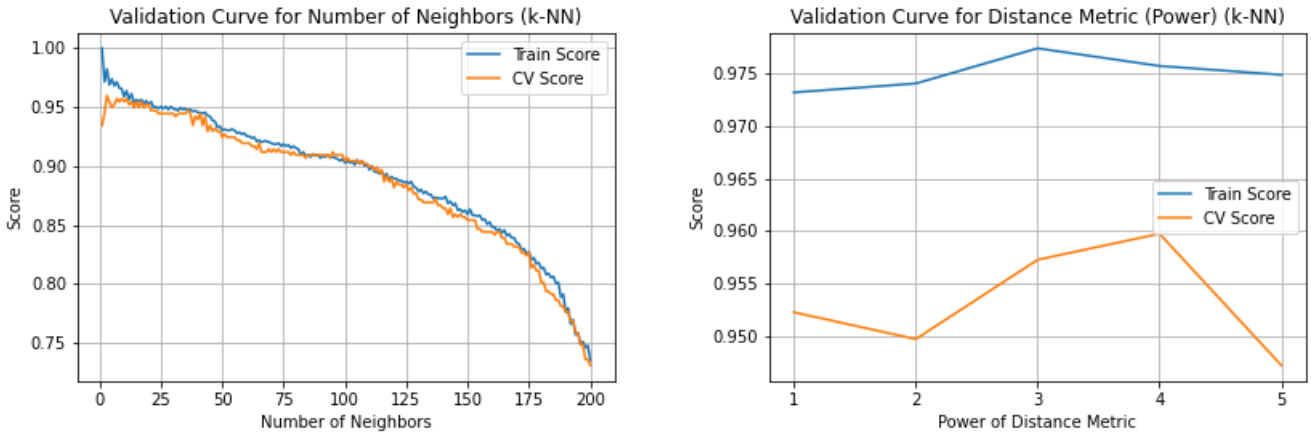


Fig. 14. Validation curves for the BCW dataset.

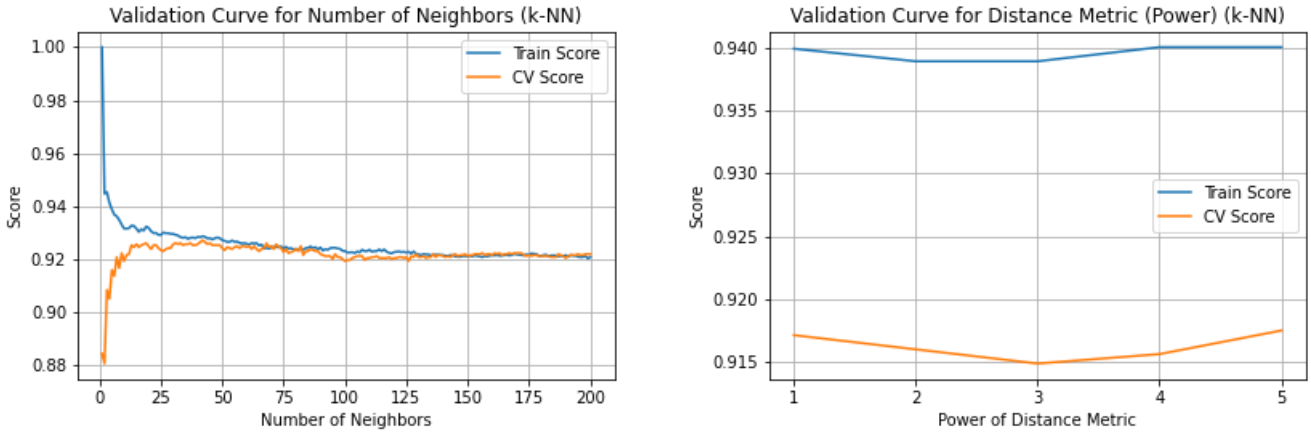


Fig. 15. Validation curves for the RCO dataset.

The results for the validation curves for the hyper-parameter k can be explained as follows: for low values of k, the value of the training score is higher as there is probably overfitting (also seen by the low validation scores). However, as k increases more bias is introduced in the model, and therefore, the scores keep dropping. For the parameter p, the BCW dataset has a higher number of features in the dataset, and therefore, lower powers might be better, whereas it is the opposite in the case of the RCO dataset.

The results of grid search on the two datasets are summarized below. The k-NN classifier actually manages to perform quite well for both the datasets, with very few misclassifications.

Dataset	k	p	Accuracy (Test Set)	Confusion Matrix		
					PF	PT
BCW	3	1	0.9708	AF	106	2
				AT	3	60
					PF	PT
RCO	49	5	0.9361	AF	470	48
				AT	25	600

Next, we look at the learning curves for the classifier for these two datasets.

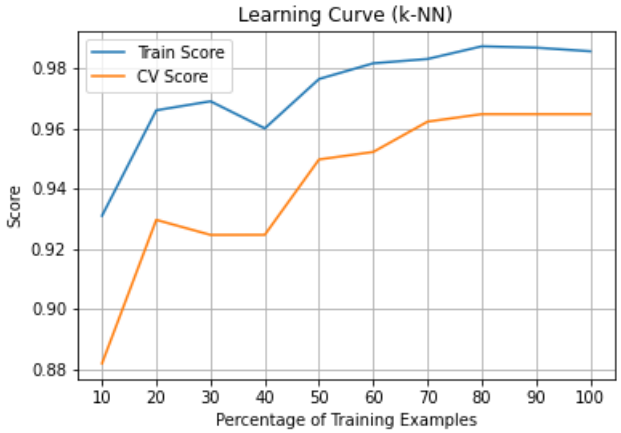


Fig. 16. Learning Curve for k-NN on BCW.

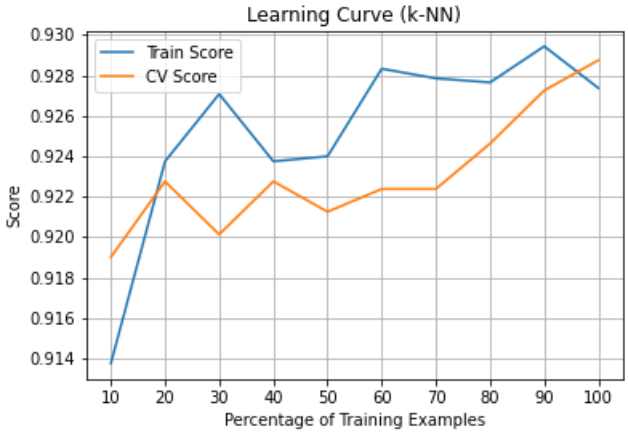


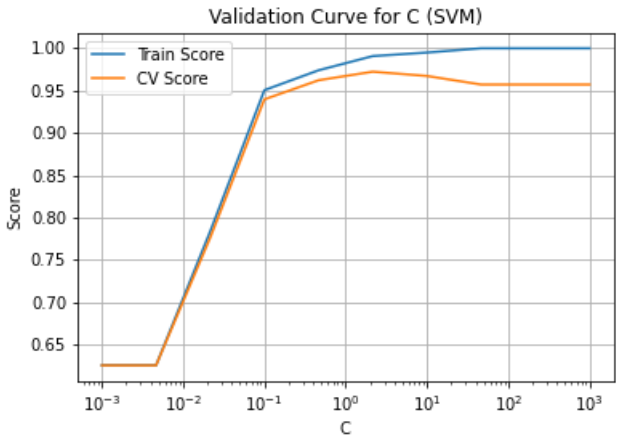
Fig. 17. Learning Curve for k-NN on RCO.

In this case, we see contrasting results for the two datasets. For the BCW dataset, the learning curve shows that the although the accuracy scores are high for both train and CV at maximum percentage of training samples, the scores have not converged enough and the CV score is relatively lower than the training score. This points to the fact that the classifier might be suffering from high variance, and more data might be helpful for training the k-NN model. For the RCO dataset, the learning curve shows that the although the accuracy scores have converged, the convergence score is not high enough, which means that there might be some bias in the model itself.

Support Vector Machines

Support Vector Machines can also be used as classifiers. In this case, we can use the hyper-parameters to be the regularization parameter or penalty term C, which represents a squared l2 penalty, as well as the kernel type in the SVM, which is one of the most important features of the SVM. For the kernel type, we carry out the validation procedure differently compared to other hyper-parameters. In order to plot the validation curve for the different kernels, the training set is further divided to create a validation set.

For the parameter C, the plots are as expected as at lower values of C, the training and testing scores are small as the strength of the regularization is large and there is bias in the model. With increasing C, the regularization strength keeps going down, and therefore, beyond a point, the model starts overfitting for the training data, thus leading to a drop in CV score accuracy. For the kernels, both linear and RBG kernels seem to perform the best for the two datasets. Surprisingly, the validation score is higher than the training score for the RCO dataset.



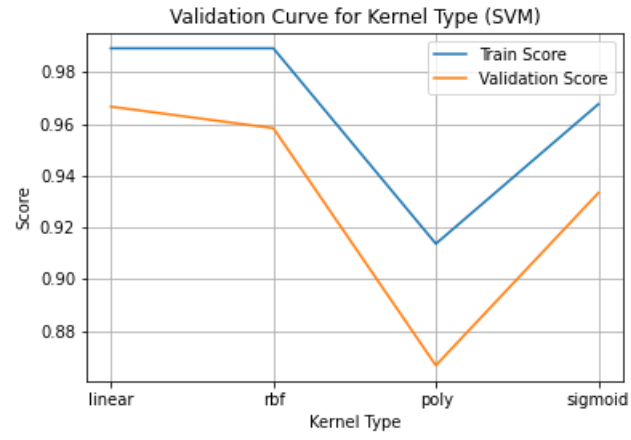


Fig. 18. Validation curves for the BCW dataset.

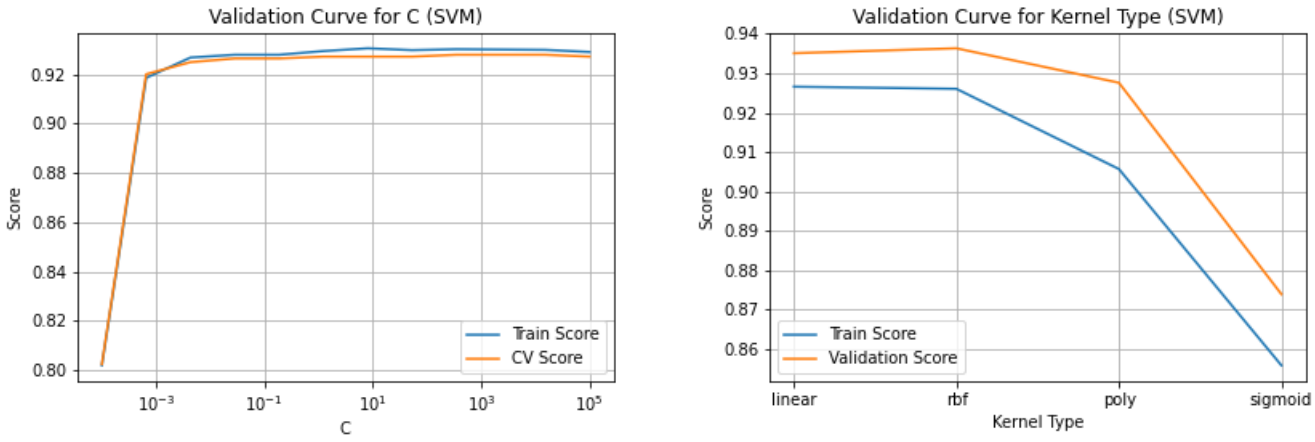


Fig. 19. Validation curves for the RCO dataset.

The results of the grid search over the type of kernel and the regularization parameter are shown in the table below. The SVM classifier performs really well for the BCW dataset, with a test set accuracy of around 98.8% and only two misclassifications. This could indicate that the data is actually linearly separable. The RCO dataset also performs well, with an accuracy score of 93.09%.

Dataset	C	Kernel	Accuracy (Test Set)	Confusion Matrix		
BCW	0.02154	linear	0.9883		PF	PT
				AF	108	0
				AT	2	61
RCO	1.2328	rbf	0.9309		PF	PT
				AF	470	48
				AT	25	600

Next, we look at the learning curves for the classifier for these two datasets.

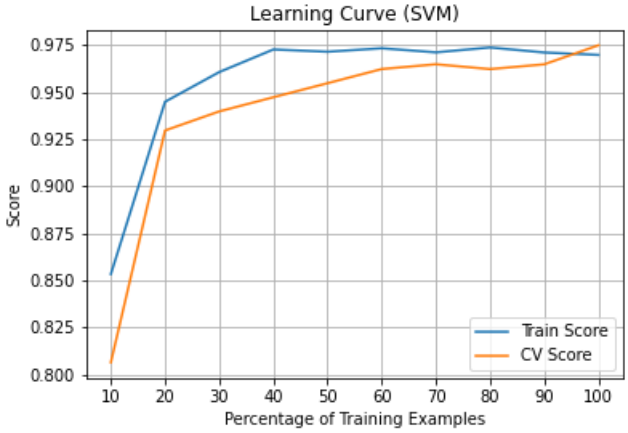


Fig. 20. Learning Curve for SVM on BCW.

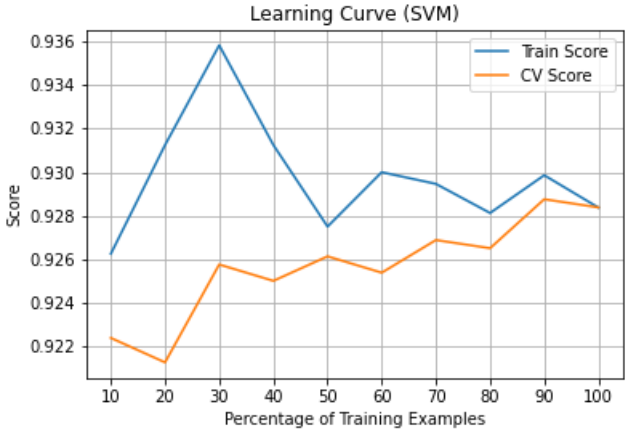


Fig. 21. Learning Curve for SVM on RCO.

For the BCW dataset, we look at the learning curve and see that the train score and CV score have converged to a high value for maximum percentage of training samples, which points to the fact that the SVM classifier is performing well without bias/variance, and that the data might be linearly separable in the feature space. For the RCO dataset, we see that the train score and CV score have converged to a relatively low value for maximum percentage of training samples, which points to the fact that the SVM classifier is biased, and a more complex model could help increase its performance.

Boosting

The final classifier that we show here is the boosting method. We use the AdaBoost classifier with a single leaf decision tree (decision stump) as the weak learner. The hyper-parameters chosen for this classifier are the number of weak learners and the learning rate.

The validation curves for both these hyper-parameters are shown below. With increasing number of weak learners, the model starts to overfit the training data, and therefore the training score goes up for both datasets. However, the CV score either remains stable or goes down after a certain number of weak learners. For the learning rate, both very low and very high learning rates are detrimental to the scores as lo learning rates lead to slow convergence and high learning rates lead to instability.

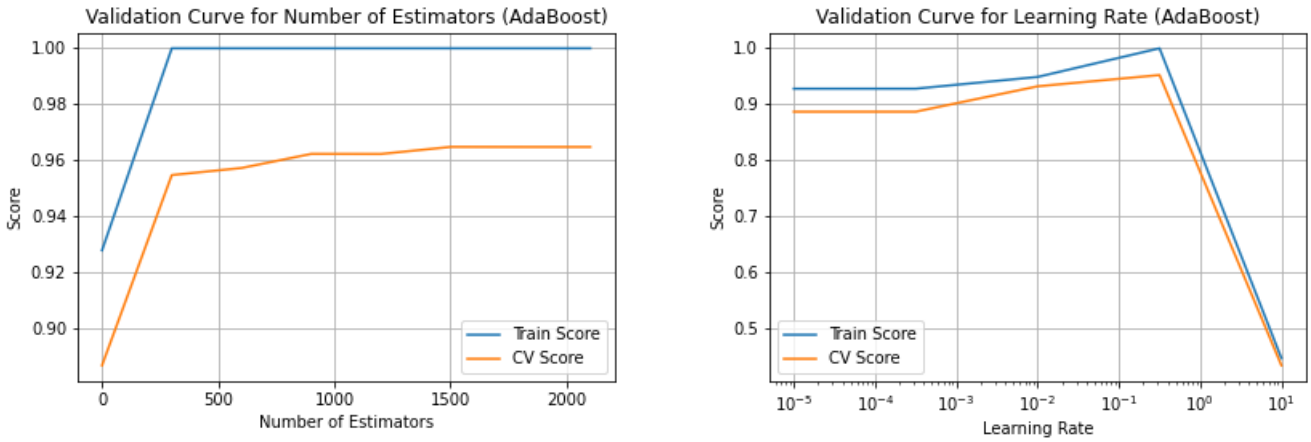


Fig. 22. Validation curves for the BCW dataset.

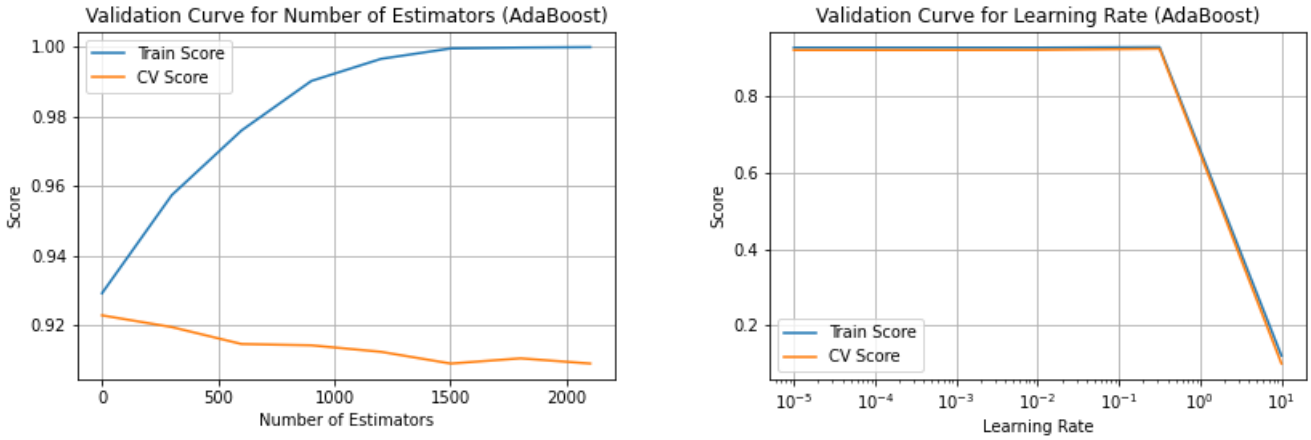


Fig. 23. Validation curves for the RCO dataset.

Results are shown for the boosting model on the two datasets. For the BCW dataset, for a learning rate of 0.316 and with approximately 1200 weak learners, the AdaBoost classifier has a test set accuracy of 98.25%. For the RCO dataset, for a learning rate of 0.01 and with approximately 900 weak learners, the AdaBoost classifier has a test set accuracy of 93.18%. The performance on the BCW dataset is especially good with only 3 misclassifications.

Dataset	Number of Estimators	Learning Rate	Accuracy (Test Set)	Confusion Matrix
BCW	1201	0.3162	0.9825	⟍ PF PT AF 106 2 AT 1 62
RCO	901	0.01	0.9318	⟍ PF PT AF 466 52 AT 26 599

We also look at the learning curves, which are shown below. For the RCO dataset, the learning curve shows convergence of the train score and the CV score at the highest percentage of training examples, and therefore, the AdaBoost classifier suffers from high bias, which means introducing more complexity in the model would be beneficial to increasing its performance. For the BCW dataset, the learning curve shows a gap between the train score and the CV score at the highest percentage of training examples, and therefore, the AdaBoost classifier suffers from high variance, which means more data would be needed to close the gap between the train and

CV scores. It is also interesting to note that in the case of the BCW dataset, the training score always remains at 1 for all percentages of training examples. This is because with enough weak learners, the training data will be fit perfectly by the model.

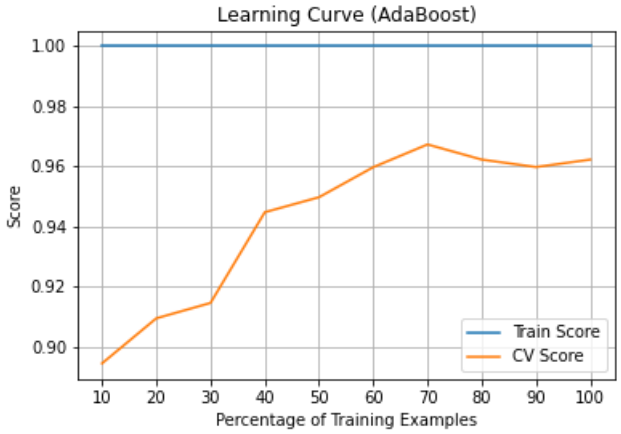


Fig. 20. Learning Curve for Boosting on BCW.

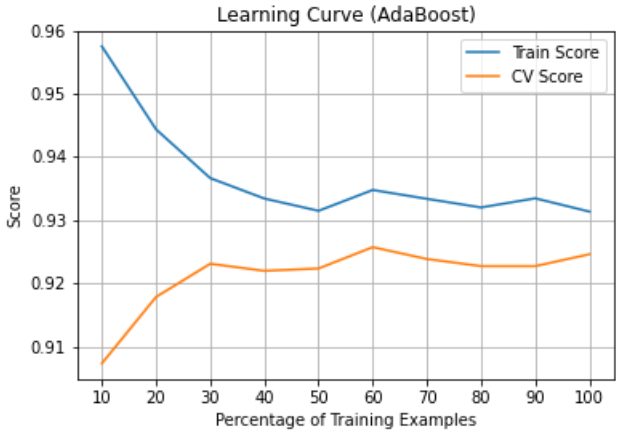


Fig. 21. Learning Curve for Boosting on RCO.

Time and Performance Analysis

The performance and efficiency of the classifiers are now compared. For the training time, the entire time taken to complete the grid search is included since searching for the best hyper-parameter is also a part of the training process.

We first look at the training times for the different classifiers. For the BCW dataset, the SVM takes the least amount of time to train. This might be because of the fewer number of training examples in the dataset, which leads to the quadratic optimization problem being solved very quickly. Decision tree is also trained quite fastest as expected as the max depth of the tree is controlled using the max depth hyper-parameter. The relatively high training time for k-NN is mainly because of the large number of points to be searched in the grid (number of neighbors). Both AdaBoost and neural network training times are also relatively high because the neural network needs to use backpropagation for gradient computation, while for AdaBoost, the significant number of weak learners takes a toll on the training time.

For the RCO dataset, the most surprising element is the fact that the SVM takes the most amount of time to train. This might be because of the large number of training examples in the dataset, which leads to the optimization problem being solved slowly. The rest of the classifiers behave in a similar fashion compared to the BCW dataset. Inference time for decision trees is also low as expected since tree traversal is a trivial process. For the k-NN classifier, since the distance to each neighbor has to be computed, the process is slow. AdaBoost is the slowest in this case as the number of weak learners is high, and inference is completed for each weak learner. Finally, both SVM and the NN classifiers are relatively faster since SVM only uses a linear function, and the NN classifiers only need a single forward pass for each test case.

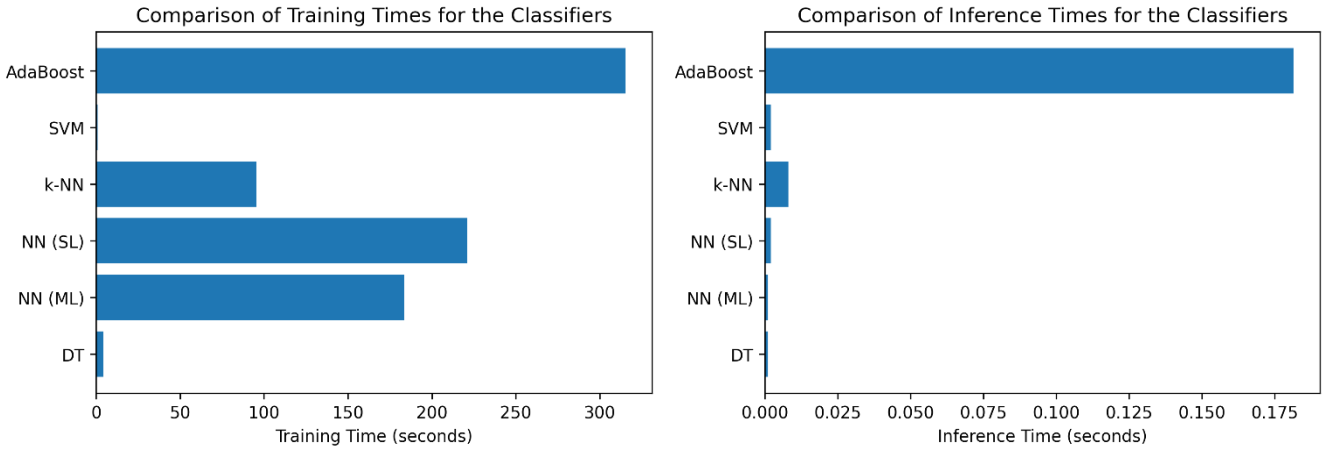


Fig. 22. Training/Inference Times for the BCW Dataset.

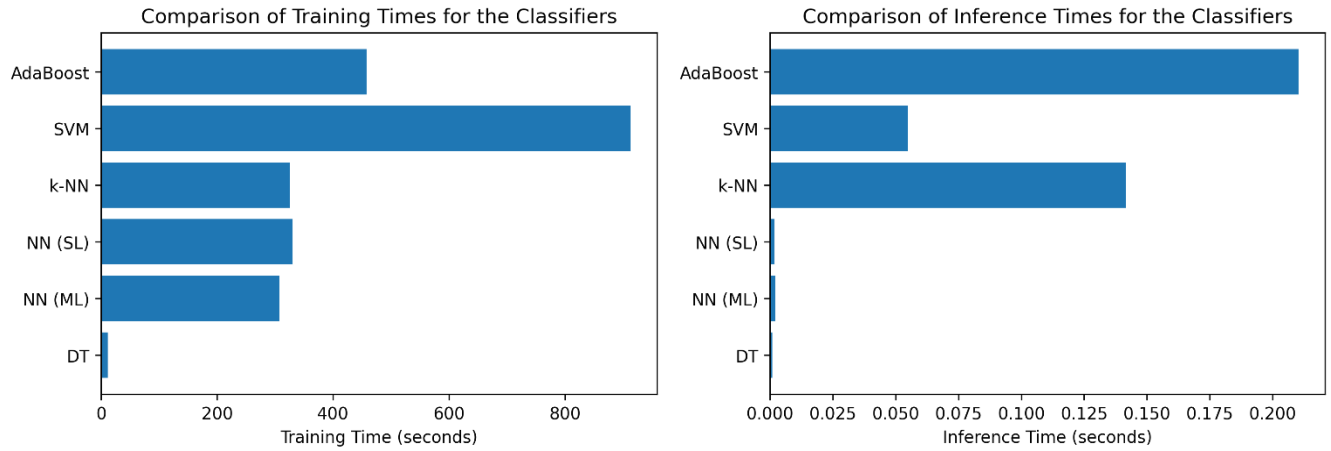


Fig. 23. Training/Inference Times for the RCO Dataset.

For the BCW dataset, decision tree has the lowest accuracy score, mainly because of low model complexity with low depth decision trees. Both neural network models (single layer and multi-layer), on the other hand, are the most complex models and perform almost perfectly on the test set. The SVM classifier with a linear kernel performs the second best among all the algorithms, which could signify that the data is linearly separable in the feature space. AdaBoost and k-NN also perform reasonably well on the dataset, but as seen from their learning curves, they would have benefitted with more available data.

For the RCO dataset, the k-NN classifier surprisingly performs the best out of the six classifiers. This might be due to the fact that all the proposed classifiers have bias, and the k-NN classifier probably uses the most complex model with 49 nearest neighbors. The performance of the multi-layer neural network is also exceptionally poor, with its accuracy score almost 1% lower than the next best classifier. It is possible that in this case, more data is required for the neural network model as the scores in the learning curve have not completely converged, as seen in Fig. 8.

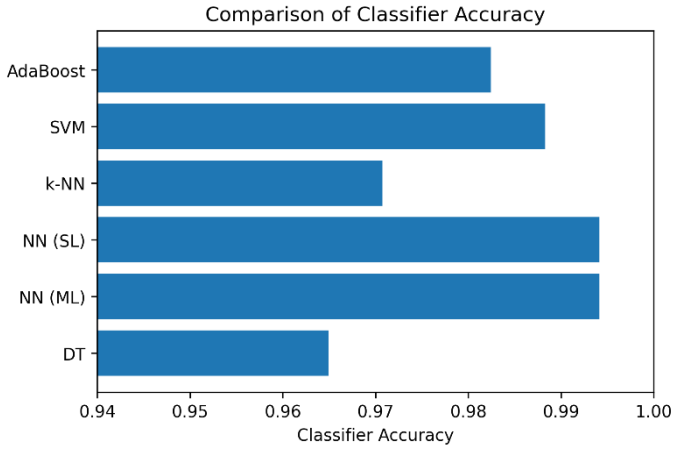


Fig. 24. Classifier Accuracy Comparison for BCW.

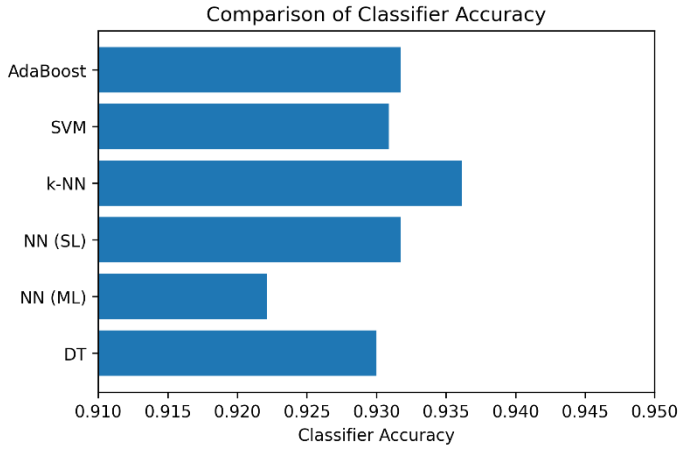


Fig. 25. Classifier Accuracy Comparison for RCO.