Soham Gadgil

CS 4641 Assignment 1 – Supervised Learning

The following paper contains detailed analyses of five supervised learning algorithms – Decision Trees, Neural Networks, Boosting, Support Vector Machines (SVMs), k-nearest neighbors. I will first start off by explaining the chosen datasets. Then, for each algorithm, I will explain the relationship between hyperparameters and accuracy and explore the best values for the hyperparameters by performing 5-fold cross validation. Finally, using the tuned hyperparameters, I will test the accuracy of each algorithm to determine the best one to use for each dataset. Accuracy is metric of choice since both datasets seem to be well balanced. Pythons scikit-learn package was used to implement the algorithms.

Datasets

1) MNIST Dataset: The MNIST dataset is one of the most widely used datasets for training and testing various learning algorithms. It is a large database of grayscale images of handwritten digits 0 – 9. The dataset used here consists of 60000 training samples and 10000 testing samples, both in csv files. Each image is 28X28, for a total of 784 pixels. Each of the training and testing sets has 785 columns, the first column being the label and the rest of the columns being the pixel values. The reason I chose this dataset was that the distribution of the digits in the samples is uniform and there are a lot of training samples available. Also, if each pixel is counted as a feature, there would be a lot of features involved leading to complex learning procedures, which would be interesting to analyze. The dataset was obtained from Kaggle and can be found here.

2) Breast Cancer Dataset: The breast cancer dataset predicts the presence of breast cancer based on six different features (mean_radius, mean_texture, mean_perimeter, mean_area, mean_smoothness). The dataset contains 569 total samples which is divided into 469 training samples and 100 testing samples. The data is in the form of a csv file with the first 5 columns being the features and the last column being the diagnosis. I selected this dataset since the sample size was small which might influence the accuracy. There is a relatively even split of positive and negative cases of breast cancer (357 positive v/s 212 negative). Also, the feature space is much smaller than the MNIST dataset, which would influence the training times of the various algorithms. Also, it is a binary classification dataset, allowing for a smaller set of output values. This was also obtained from Kaggle and can be found here.

Decision Trees

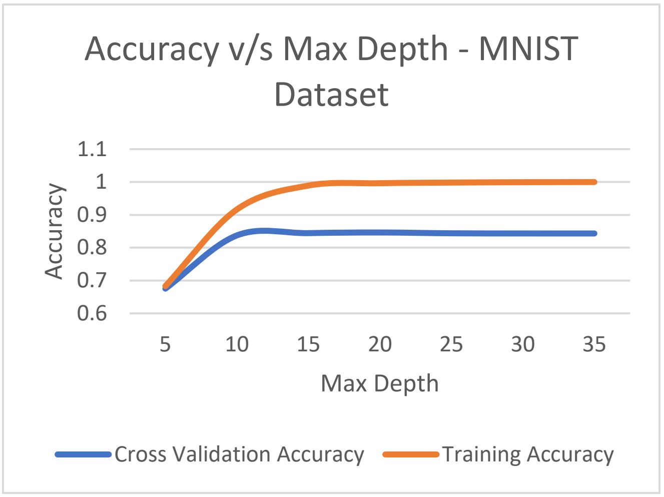
The hyperparameters selected to tune for decision trees were:

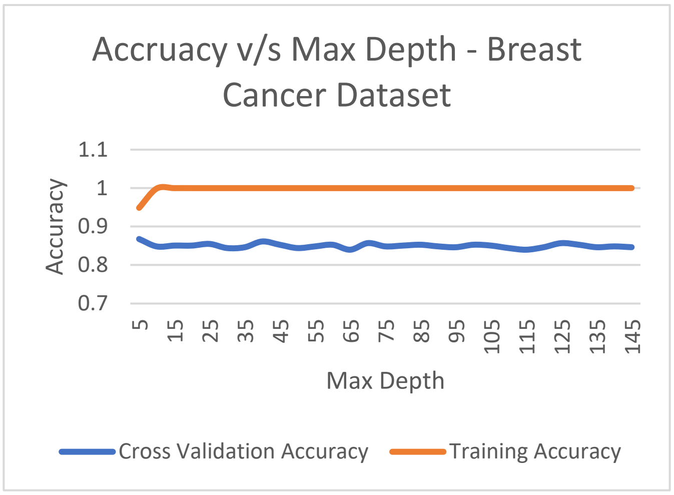
1) Max Depth: The maximum depth of the tree. Pruning is important in decision trees to prevent the tree from becoming too complex and start overfitting.

2) Max Leaf nodes: The decision tree is grown with these number of leaf nodes in a best first fashion. This is another parameter to prevent the tree from overfitting.

Max Depth

Following are the results obtained by changing the maximum depth of the tree:



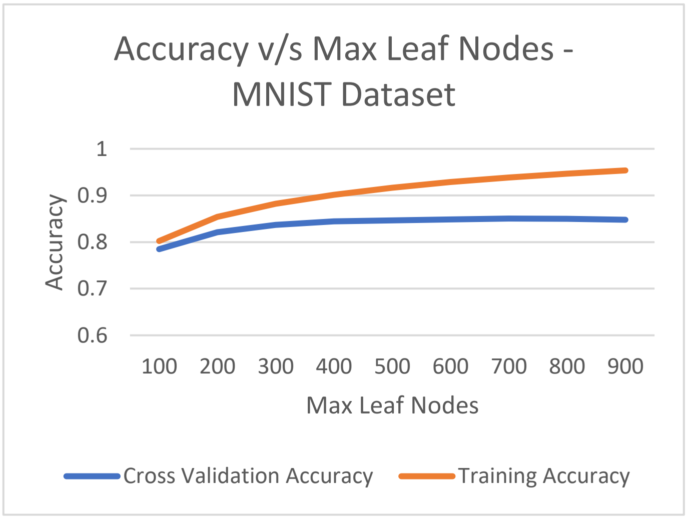


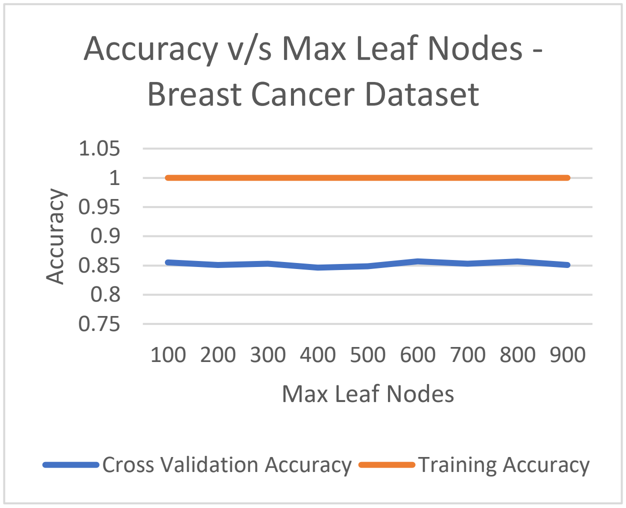
For the MNIST Dataset, it can be observed that both the training and cross validation accuracies increase at the beginning, but they eventually reach an asymptote, showing that adding more nodes to the tree does not provide more information. It can also be seen that although the training and validation accuracies show the same trend, the gap between them increases rapidly, and the training accuracy reaches very high values. This implies that the model is overfitting as the max depth increases. The optimal hyperparameter value chosen was 15, which is when the accuracy starts to remain unchanged. The average training time was ~6 s while the average testing time was ~0.03 s

For the Breast Cancer Dataset, the training accuracy follows a similar trend as in the MNIST Dataset, it increases to a very high accuracy and remains unchanged thereafter. The cross-validation accuracy on the other hand follows a different trend. Apart from some slight variation, it stays the same throughout, indicating that the max depth of the tree has little effect on the accuracy. This model also suffers from overfitting, as evidenced by the gap between the training and the cross-validation accuracies. The parameter value chosen was 5, since it provided the best accuracy for validation and it was before the asymptote. The average train time was ~1.2 ms while the average test time was ~0.38 ms. Interestingly, the runtimes were not dependent on the max depth of the tree and were arbitrary, indicating that the larger depths were excessive and not needed.

Max Leaf Nodes

Following are the results obtained by changing the maximum number of leaf nodes of the tree:





For the MNIST Dataset, the accuracies follow a similar trend to the max depth. However, this graph seems better since the gap between the training and the cross validation accuracy does not grow rapidly. Also, the training accuracy keeps increasing, albeit very slowly, without forming an asymptote. This is a good sign meaning overfitting does not occur as fast as it does with the max depth hyperparameter. The cross validation accuracy still asymptotes towards the end, thus the chosen value for the max leaf nodes was 400, where the gap between the two accuracies is small and the accuracy is starting to remain constant. The average training and testing times are ~6.3 s and ~0.032 s respectively, and they increase with the number of nodes as expected.

For the Breast Cancer Dataset, the training accuracy seems to remain unchanged completely, providing no valuable information about the effect of the number of leaf nodes. However, the cross validation accuracy seems to remain arbitrary, displaying similar but erratic accuracies. The model tends to overfit at some nodes, especially towards the end of the graph. The chosen value of the maximum leaf nodes was 300 since the model starts to show signs of overfitting, owing to the decrease in accuracy after that. The average training and testing times are ~1 ms and ~0.3 ms respectively, showing that decision trees take negligible time to build and test on this dataset.

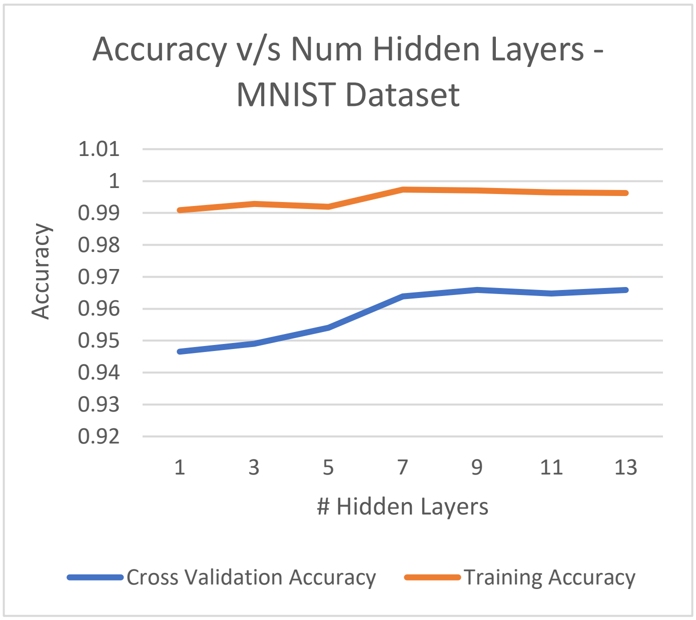
Neural Networks

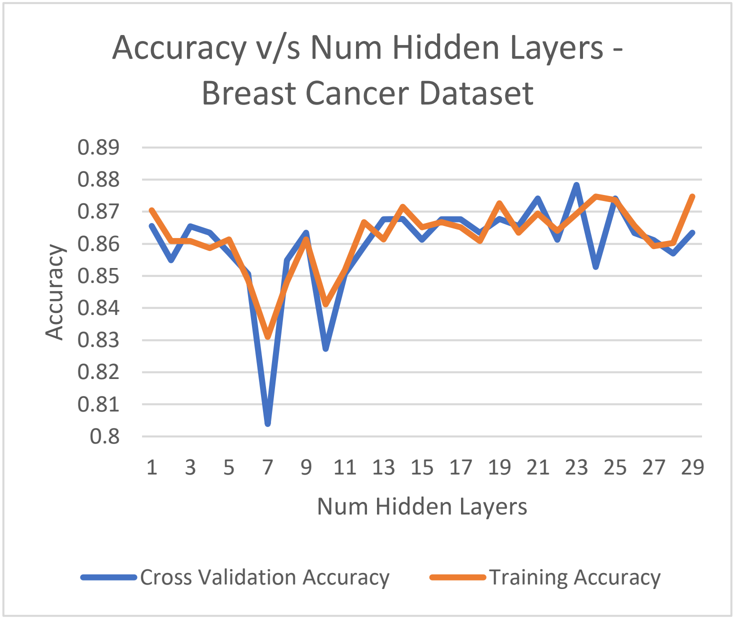
The type of neural network used was a multi-layer perceptron (MLP), a feed forward neural network model. The hyperparameters selected to tune for neural networks were:

1) Number of Hidden Layers: The varying number of hidden layers are an important parameter for tuning an MLP, since it affects the complexity of the algorithm and having more layers possibly allows for more features to be learned. Number of neurons in each layer is the default value i.e. 100.

2) Number of Neurons: Another parameter for tuning is the number of neurons present in each layer. In this analysis, we consider each of the hidden layer having the same number of neurons, but that number can be varied. This affects the number of connections between the different layers, which in turn affects number of computations. Number of hidden layers was kept constant at 4.

Number of Hidden Layers

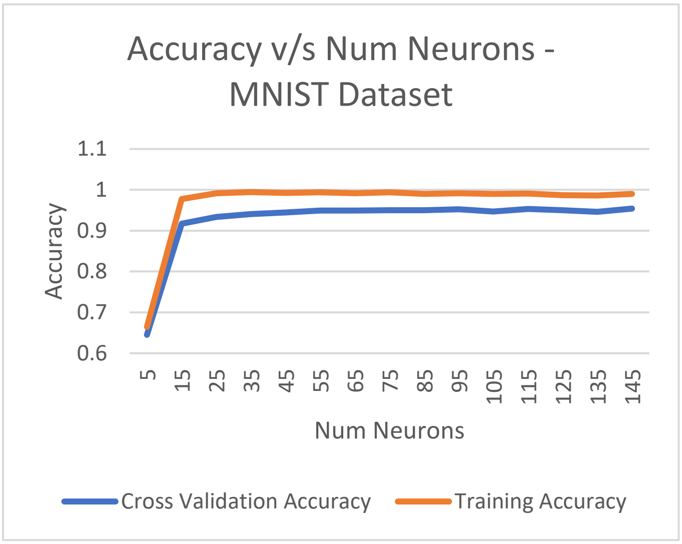


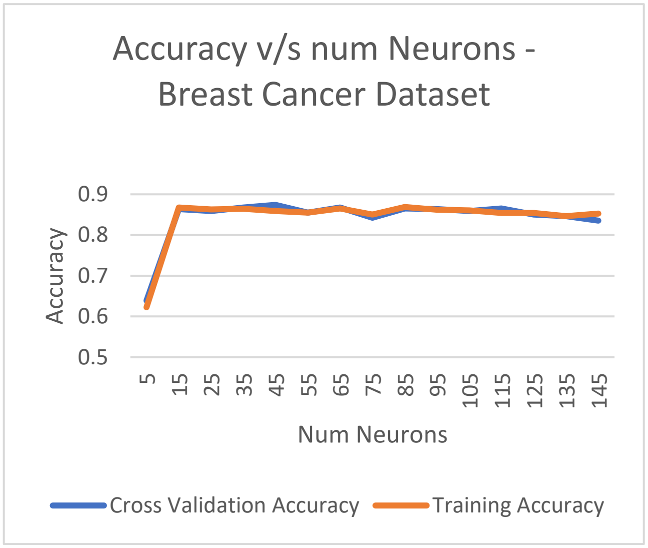


For the MNIST Dataset, we observe two asymptotes, one at the start, where the accuracy is somewhat unchanged, but there is a big gap between the cross validation accuracy and training accuracy. Then, the accuracy increase and again asymptotes towards the end, but the gap between the two accuracies decreases, meaning adding more layers actually helps the algorithm. This is understandable since the feature space is huge, the number of layers needed to capture relevant information is higher. The chosen value was where the later asymptote starts, which is 7. The average training time, which is ~85.73 s is much higher than the average testing time, which is 0.1 s, indicating that neural networks take a long time to train, but inference time is comparatively very small. Another thing to note is that the accuracy is significantly higher than that for decision trees but the training time is much more for neural networks.

For the breast cancer dataset, the MLP starts overfitting very quickly, even for a small number of hidden layers. The accuracy takes a sharp dip around 7 hidden layers, which mean that even 7 layers are excessive. This can be explained by the fact that the feature set and the number of samples are quite small, so not many layers are needed to learn the patterns in the dataset. Adding more layers will tend to start picking idiosyncrasies in the data and try to fit the training set too well, which is not what we want. Towards the end, the gap between the cross validation accuracy and the training accuracy also starts increasing. The chosen value for this hyperparameter was 5, where the accuracy is relatively high, and the two accuracies agree with one another. The average training and testing times were ~1.3 s and ~3.4 ms.

Number of Neurons





For the MNIST dataset, there is a big jump in accuracy at the start and then it stabilizes, eventually forming an asymptote. The gap between the cross validation accuracy and the training accuracy also increases at the start and then remains relatively constant. For the optimal value, we chose the number of neurons where the graph becomes asymptotic, which is 55. The average training and testing times are ~54.75 s and ~0.08 s respectively.

For the Breast Cancer dataset, there is a very similar trend seen. The accuracy increases at first, but then remains unchanged. The progression towards the asymptote is steeper than that in the MNIST dataset. Also, the gap between the cross validation and the training accuracies is negligible throughout the graph, meaning the testing data agrees with the training data. The chosen value for the number of neurons is 55, which is where the cross validation accuracy is high and the gap between the accuracies is very small. The average training and testing times are ~0.5 s and ~0.001 s respectively.

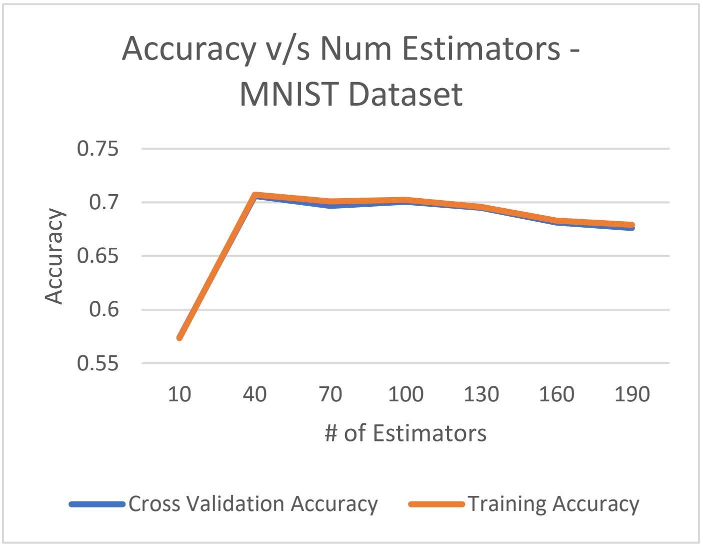
<u>**Boosting**</u>

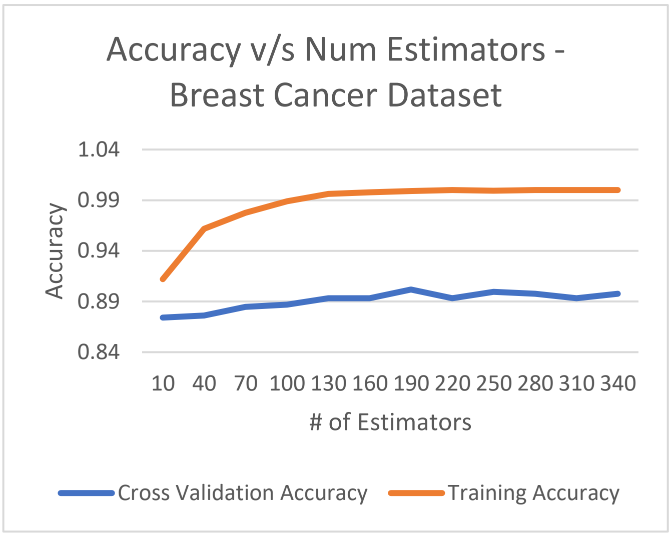
The type of boosted decision tree used was the Adaptive Boosting, using the AdaBoostClassifier, which weights the incorrectly classified instances higher than the correctly classified ones. The hyperparameters tuned for AdaBoost were:

1) Number of Estimators: This is the maximum number of estimators at which the boosting is terminated. This hyperparameter was chosen since stopping the boosting algorithm forcefully would affect the accuracy.

2) Max Depth: To remain consistent, since this is a boosted decision tree, pruning is still important to ensure the tree does not grow to become overly complex.

Number of Estimators

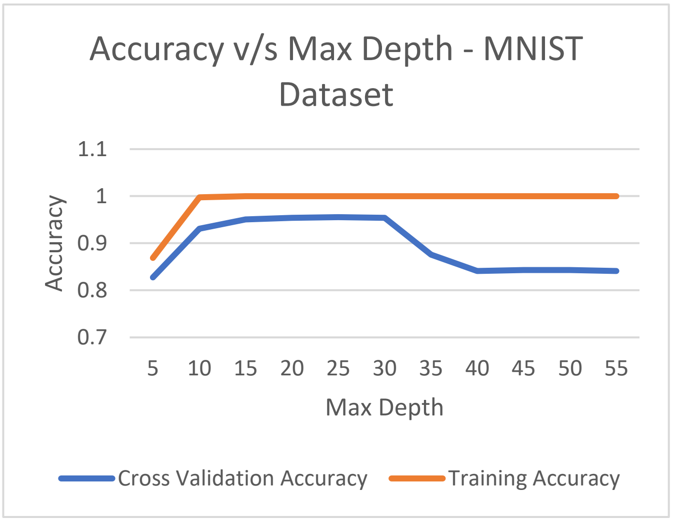


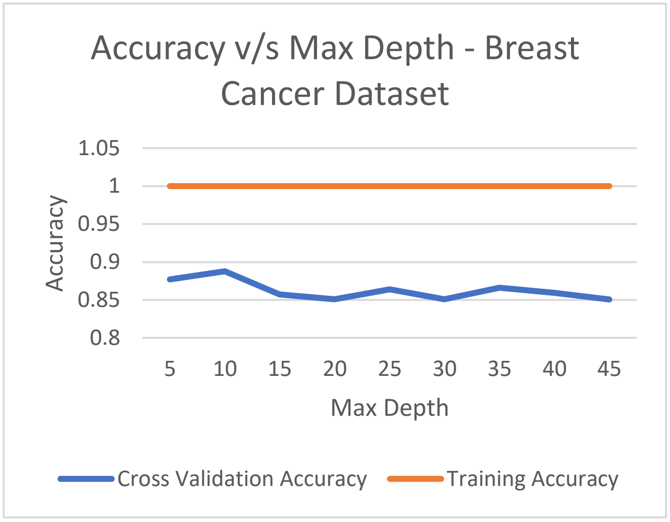


For the MNIST Dataset, the accuracy starts low and jumps up steeply. Thereafter, the accuracy starts decreasing showing that the model is overfitting. Also, the training and the cross validation accuracies are practically indistinguishable meaning that the model adapts to the training and the testing sets equally well. Since our goal is to minimize overfitting, the chosen value for the number of estimators is 40, just after the accuracy jump and before overfitting starts becoming prominent. It can also be observed that the highest accuracy obtained is lower than the accuracies of all models seen so far. The average training time is 50.6 s and the average testing time is 0.5 s. It is observed that the training and testing time increase rapidly with the number of estimators (5.1 s to train for 10 estimators vs 92.6 s to train for 190 estimators). This is another reason to keep the number of estimators small.

For the Breast Cancer Dataset, the cross validation accuracy shows a similar trend to the one in the decision tree algorithm although it increases gradually early on, before becoming steady. On the other hand, the training accuracy shows a big jump in the accuracy before ending in an asymptote. As a result, the gap between the two accuracies increases by a lot. The number of estimators chosen is 130, which is about where the training and cross validation accuracies stop improving and remain relatively constant. The average training time is 0.183 s and the average testing time is 0.015 s. Even here, the runtimes increases rapidly as the number of estimators increase, although the values are much smaller than the MNIST Dataset, possibly because of the lesser number of samples and dimensionality.

Max Depth





For the MNIST dataset, it can be seen that the cross validation accuracy increases at first, remains constant, and then decreases sharply, while the training accuracy increases and reaches an asymptote, implying that the model starts overfitting. This is also evidenced by the fact that the gap between the two accuracies starts off small but for larger values of the max depth, the gap becomes significant. A deeper tree can fit more complicated functions, but increased flexibility can cause overfitting , and generalization performance may suffer, as we see in this case. Since we want to minimize overfitting, the chosen value of the max depth is 15, where the accuracy of the model stops improving. The runtime variation is interesting. The average training time was ~187.7 s which increased at first, up until a max depth of ~25 (814 s), but then decreases sharply until the end (Max depth of 55 has a train time of 7.14 s). This can be because at the depth that the decision tree starts overfitting, it has learnt the data too well so increasing the depth will not provide any extra information about the dataset. The average testing time was ~0.22 s, much lower than the average training time, showing that once the tree has been made, classifying a new sample takes a short time.

For the Breast Cancer dataset, the cross validation accuracy increases at first, but then the accuracy decreases, showing that the model is overfitting. The Training accuracy remains very high and constant throughout, resulting in a big gap between the cross validation and the training accuracies. The value chosen for the max depth is 10, where the model starts decreasing in training accuracy. The average training time is ~12.9 ms and the average testing time is 1.13 ms, both of which are much less than the time taken for the MNIST dataset, mostly because of the lower dimensionality and the smaller number of samples.

Support Vector Machines (SVMs)

C-Support Vector Classification (SVC) is used for this analysis. The multiclass support is handled according to a one-vs-one scheme. Following are the hyperparameters that were tuned for this algorithm:

1) C and Gamma: The C value is the penalty parameter for the soft margin cost function. and Gamma is the kernel coefficient for rbf, poly, and sigmoid kernels. The value of C controls the influence of each individual support vector, involving trading error penalty for stability. This means that the model considers it alright to misclassify a few labels to prevent fitting to the data too well and doing poorly on unseen data. Gamma controls the tradeoff between error due to bias and variance in the model. Having a high value of Gamma gives low bias and high variance and vice versa. Thus, SVMs are very sensitive to these hyperparameters, making them important to tune. The kernel used was the default rbf kernel.

2) Kernel Type: The kernel type used by the SVM is an important hyperparameter since it will affect the way the input data gets transformed into the representational space. The kernels tested here are linear, poly, rbf, and sigmoid. The C and gamma values obtained above are used with these kernels. Usually the non-linear kernels require many more resources to train than the linear kernel. Having a high value of C gives low bias and high variance and vice versa.

Using the data without preprocessing proved to be too time consuming. Running MNIST with the rbf kernel took an average of 1458 s, and the accuracy was poor (~0.1). This can be attributed to the high dimensionality of the MNIST dataset. Thus, the data was scaled so that the values were between 0 and 1.

C and Gamma

Grid search was used to find the best values for C, the penalty parameter of the error term and Gamma, the kernel coefficient. This grid search was used only on part of the training set since using all the samples proved to be too time consuming and did not affect the optimal values. The following tables show some interesting accuracies obtained by different combinations:

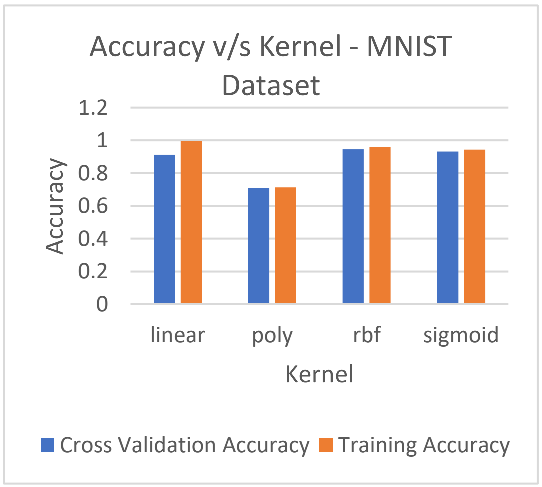
MNIST Dataset		
C Value	**Gamma**	**Accuracy**
10	0.0001	0.685
10	0.001	0.84
1	0.001	0.633

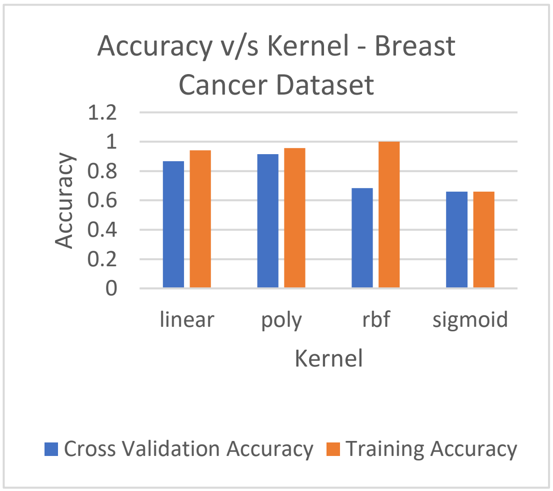
For the MNIST Dataset, the values tested for the MNIST dataset were C = [10, 1] and gamma = [0.001, 0.01, 0.1]. The best values were found out to be C = 10 and gamma = 0.001. As it can be seen, the accuracies vary a lot based on the values chosen. The accuracies are relatively lower since they were used on a smaller training set

Breast Cancer Dataset		
C Value	**Gamma**	**Accuracy**
5	0.01	0.7
5	0.1	0.66
1	0.001	0.68

The values tested for the Breast Cancer Dataset were C = [5, 10, 1] and gamma = [0.1, 0.01, 0.001]. The values selected were C = 5 and gamma = 0.01 since they gave the highest accuracy.

Kernel Type





For the MNIST dataset it is observed that the poly kernel has a low accuracy which can be because of the high dimensionality of the MNIST data, which makes it difficult to separate it using a polynomial, and polynomial kernels are suited for normalized data. The degree used was the default value, which is 3. The linear kernel also has the cross validation accuracy on the lower side which is intuitive since the MNIST dataset is certainly not linearly separable, there being 10 classes of digits that can be formed. The rbf and sigmoid kernels are close on accuracies. Gaussian kernels tend to provide good performance under smoothness assumptions. Rbf kernel is chosen here, since it has a slightly higher accuracy than the sigmoid kernel. Another interesting point is that the linear kernel, which takes ~97.39 s for trianing is much lower than the training times of other kernels, showing that non-linear kernels require more resources. In-fact the poly kernel takes ~1274.6 s to train, almost 13 times than the linear kernel takes.

For the Breast Cancer dataset, we see the opposite. The poly kernel actually performs the best while the rbf and sigmoid kernels have poor accuracies. This implies that the data can be linearly separable but using more complex kernel might start overfitting the data. The poly kernel, which takes ~9.32 s to train, has a much larger training time than any other kernel. However, its accuracy is also the highest, which is why poly is chosen as the desired kernel.

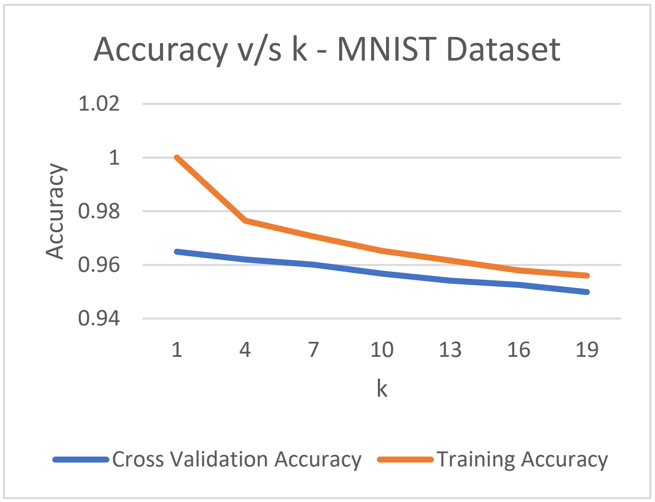
k-Nearest Neighbors

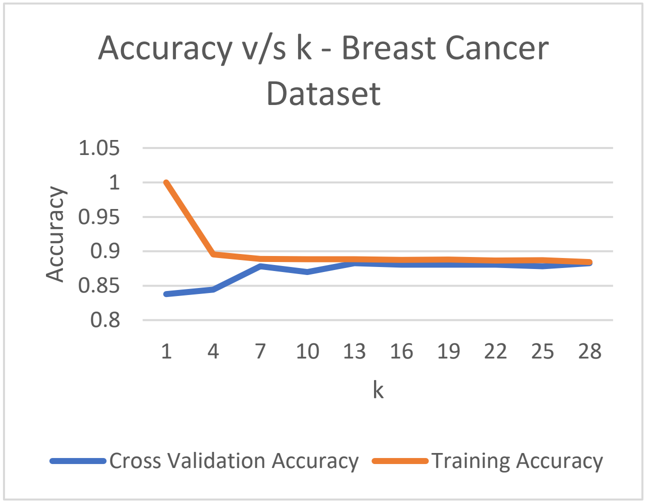
The following hyperparameters were tuned for the kNN algorithm:

1) Number of nearest neighbors (k): The obvious hyperparameter to tune here is the actual value of k, the number of neighbors included in a vote for classification. The value of k can affect the complexity and accuracy of the algorithm

2) Distance Metric: The distance metric used to calculate the distance to the nearest neighbors. The metrics used are Euclidean, Manhattan, Chebyshev, and Minkowski

Number of Nearest Neighbors (k)

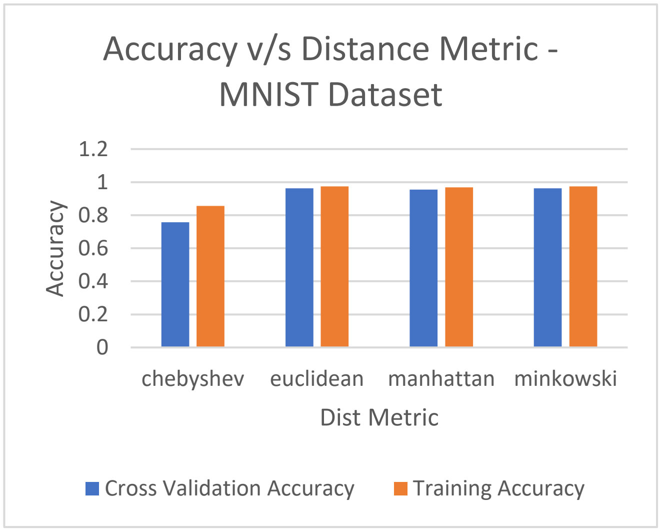


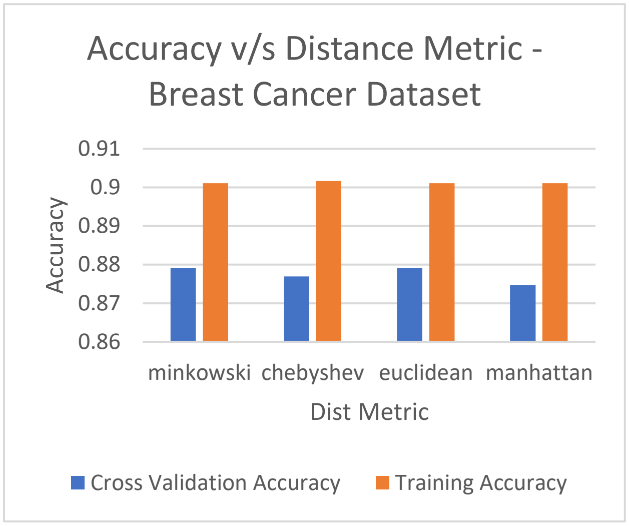


For the MNIST dataset, it can be observed that both the cross validation and training accuracies decrease. The cross validation accuracy decreases steadily while the training accuracy shows abrupt decays at first. The training accuracy is very high for 1 nearest neighbor since the data point being tested always lies in the training set . This can be because a higher k may result in neighbors which is are far from the data point getting selected to take part in the voting, resulting in misclassification. Also, the gap between the two accuracies can be seen to be decreasing, which is a sign that the model might not be suffering from overfitting a lot. The chosen value of is 1, where the cross validation accuracy is still high and the gap between the two accuracies starts decreasing. The average training time, which is ~3.8 s, is lower than the average testing time, which is ~16.6 s, showing that calculating the distances between the nearest neighbors takes time, another reason why k should be kept small.

For the Breast Cancer Dataset, the training accuracy shows a sharp decrease at first, understandably because 1 nearest neighbor is a trivial training case. After that, it reaches an asymptote. The cross validation accuracy increases at the start and also asymptotes as the number of nearest neighbors (k) increases. The chosen value for k is 13, where the gap between the two accuracies is small and the cross validation accuracy starts remaining unchanged. The average training time is ~1 ms and the average testing time is ~0.05 s. Again, the testing time is higher than the training time.

Distance Metric





For the MNIST Dataset, the chebyshev metric does poorly as compared to the other three, which have almost the same cross validation and training accuracy. If we look closely, the manhattan metric accuracy is slightly lower than the euclidean and minkowski metrics. To choose the optimal metric, I looked at the runtimes for the two metrics. Euclidean metric had sligthly lower training (3.2 s) and testing (15.4 s) times than minkowski (3.3 s for training and 15.6 s for testing). Thus, euclidean distance metric was chosen.

For the Breast Cancer dataset, the first thing to notice is that the Training accuracy is much higher than the cross validation accuracy meaning that the model does not work better on unseen data. Again, the Euclidean metric had the highest accuracy and the lowest runtimes, which is why it was chosen as the optimal metric.

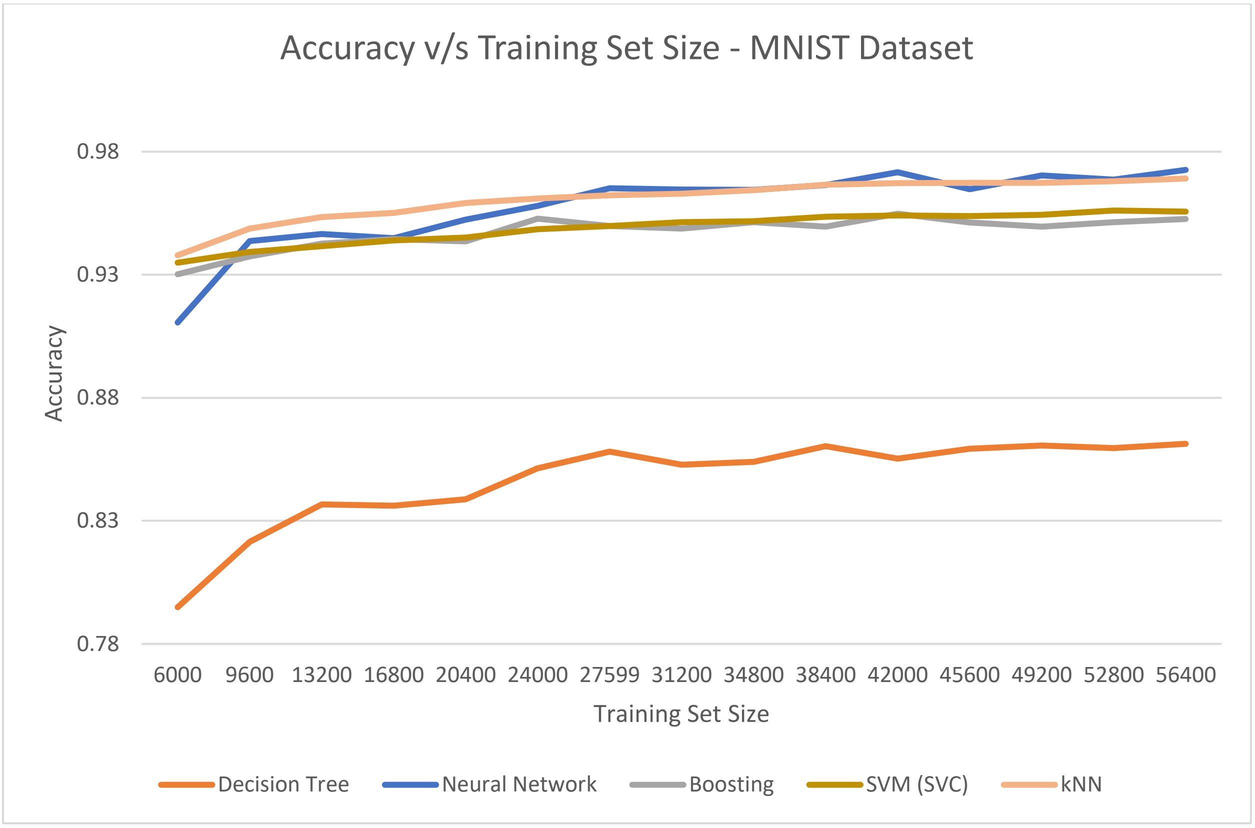
Testing

To summarize the analyses above, following are the hyperparameters chosen for testing purposes:

	Decision Trees	**Neural Networks**	**Boosting**	**SVMs**	**kNN**
MNIST Dataset	Max Depth = 15, Max Leaf Nodes = 400	# Hidden Layers = 7, #Neurons = 55	# Estimators = 40, Max Depth = 15	Kernel = rbf # Iterations = 1100	K = 1, Metric = Euclidean
Breast Cancer Dataset	Max Depth = 5, Max Leaf Nodes = 300	# Hidden Layers = 5, #Neurons = 55	# Estimators = 130, Max Depth = 10	Kernel = poly # Iterations = 110000000	K = 13, Metric = Euclidean

Now that we have the optimal hyperparameters, the next step is to test the different algorithm on the testing set i.e. unseen data and observe how each algorithm performs and develop learning curves. The training set size is varied to analyze how accuracy is affected by the amount of data provided to the model. The testing set size is kept the same throughout. The following graphs show the performance of various algorithms using the hyperparameter values obtained above.

MNIST Dataset



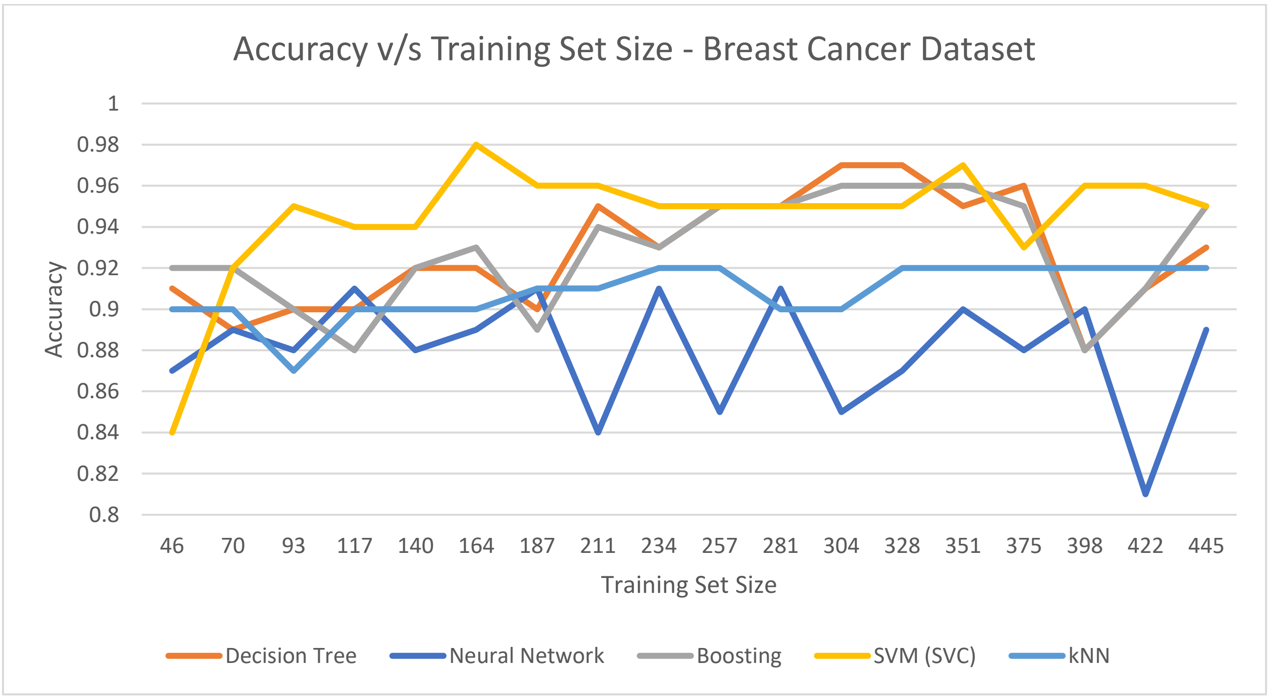
The first thing that can be observed is that decision trees give a much lower accuracy as compared to the other algorithms. This can be attributed to the high dimensionality of the MNIST dataset. Since pruning is applied, the decision tree might not be able to learn all the information needed to classify unseen data accurately. The depth of the tree is cut short, not allowing it to expand fully since it can lead to an overly complex tree resulting in overfitting. Also, the accuracy increases rapidly for small training sizes and then remains relatively unchanged.

The accuracies of the other algorithms seem to be clustered together at the top and the graphs seem smooth without any sudden peaks of troughs, and the algorithms steadily increase in accuracy as the training set size is increased, since the models observe more data points and can learn more. Another interesting thing to note is that the SVM and Boosting algorithms have the almost identical learning curves. This can suggest that they learn similar information about the dataset, leading them to classify unseen data similarly. The learning curve for neural network increases almost linearly in accuracy at first, suggesting that adding more data certainly helps the network to learn more, possibly because it sees more samples of the different classes. If we look closely, there are certain segments where the accuracy drops slightly even with the training size increasing. This shows that the curse of dimensionality plays some part in the accuracy. The dataset has a lot of features since each pixel in the image can be a feature. However, the effect is not as pronounced since the trend of the accuracy still seems to be increasing. The kNN algorithm seems to remain well behaved throughout, increasing slowly as the training size increases.

Considering the training and testing times mentioned above, the order of runtime for the algorithm is SVM > Neural Networks > Boosting > kNN > Decision Tree (Highest to Lowest). It is understandably that SVM takes the longest runtime since it requires the most resources to perform the computations. For SVM, I tested with a small range of values while doing the grid search. The range of values can be extended to check if the accuracy can be improved by testing some more values of C and Gamma. kNN has a higher testing time that training time, since most of the work for distance calculation is done during inference.

While choosing the best algorithm, three things were considered – the accuracy of the algorithm, the runtime, and the behavior in the learning curve. Based on this, it seems like Neural Network and kNN are good candidates since they have a better accuracy than the others, with kNN having a slightly better accuracy for larger training sizes. Based on the runtimes, neural networks have a higher training time, while kNNs have a larger testing time. However, the behavior of kNN with the training set size seems to be more stabilized than that of the Neural Network. Thus, kNN is proposed as the best algorithm for the MNIST dataset.

Breast Cancer Dataset



First thing that can be noticed is that the learning curves are more erratic than those in the MNIST Dataset. In MNIST, most curves are smooth, without any sudden rises or falls. However, in this dataset the accuracies for some algorithms vary a lot as the training set size changes. This can be attributed to the small number of samples available. There are only 445 training samples available, thus the training sizes tested are also small. As more training data is added, the algorithms can learn a lot more about the dataset resulting in sharp changes in accuracy, as noticeable in the learning curves above. Another thing to note is that although the curves are erratic, they all hover around the same accuracy range, which was not the case in MNIST, where decision trees performed distinctively worse than the other algorithms.

Both Boosting and Decision Tree algorithms have similar learning curve patterns, suggesting that giving more importance to misclassified data points does not help much in the is dataset. Also, both the curves show a sharp dip in accuracy towards the end, which might be because the additional data added is noisy and can lead to the overfitting. The kNN algorithm is the smoothest of all the algorithms and does not show any sudden changes in accuracy. Also, the accuracy seems to reach an asymptote towards the end, suggesting that adding more samples does not affect the accuracy. The learning curve for the Neural Network seems to be the most erratic. It seems to be relatively stable for smaller training samples but then it shows big changes in accuracy for the rest of the learning curve. Even the accuracy seems to be the lowest of all. This might be because at the beginning, the number of samples are too small to learn anything substantial and adding more samples helps the neural network learn more. However, as more samples keep getting added, the neural network might start learning the oddities of the data, causing it to overfit and perform poorly on the testing set. If a neural network other than a multilayer perceptron is used while testing various ways the network is structured, we might be able to improve the accuracy. SVM seems to do poorly at the beginning when the training set size is small. However, it quickly learns as more data is added and the accuracy becomes higher than all the other algorithms. Overall, it seems to have the highest accuracy which remains relatively stable.

Considering the training and testing times mentioned above, the order of runtime of the algorithms was SVM > Neural Network > Boosting > Decision Tree > kNN (Highest to Lowest). Decision Tree and kNN algorithms had similar runtimes but the training time of decision tree was slightly higher than that of kNN. As with MNIST, SVM took the longest time out of all the algorithms. However, since the total data is small, the difference in runtime is not as drastic as that in MNIST.

While choosing the best algorithm, three things were considered – the accuracy of the algorithm, the runtime, and the behavior in the learning curve. In terms of accuracy, SVM, Boosting, and Decision Trees all have comparable accuracies. SVM performed the best in accuracy and the behavior was relatively stable, expect for the case where the number of training samples was small, but that would be expected. Even though SVM too the longest time to train, since the dataset is quite small, runtime is not that biggest concern. Thus, SVM is chosen as the best performing algorithm.

Since the Breast Cancer Dataset presents a binary classification problem, we can further analyze the F1 score and the confusion matrix for the best algorithm selected i.e. SVM.

Confusion Matrix

	Predicted Positive	**Predicted Negative**
Actual Positive	59	2
Actual Negative	3	36

Thus, the F1 score is 0.959, the Precision is 0.9516, and the Recall is 0.967. The high F1 score and the confusion matrix suggest that one class does not dominate the other and that the dataset is evenly distributed. The Recall is a bit higher than the Precision, which means the SVM model is good at predicting the presence of breast cancer.

Conclusion

In this paper I analyzed different algorithms and their effect on different datasets. I realized that tuning the hyperparameters is very important and it can lead to large changes in accuracy. In this paper I have tuned ~2 hyperparameters for each algorithm. Ideally, we would want to tune all possible hyperparameters to extract best performance. I also learned that the algorithms are sensitive to the dataset they are being trained on, so it is important to consider all algorithms while testing a dataset. An algorithm which performs the best on one dataset might perform the worst on another dataset. I used accuracy as the performance metric here since my datasets were well balanced but for datasets which are biased towards one class may give a good accuracy but will have a low F-Score.