CS-7641
Assignment 1- Supervised Learning

Nayeem Aquib

1 INTRODUCTION

In this report, 5 different learning algorithms- Decision trees, Neural networks, Boosting, Support Vector Machines, and k-nearest neighbors, will be used to train and test on 2 different datasets. Both of these datasets were acquired from Kaggle. One is a dataset of videogame sales across different platforms and different regions. Another is a dataset of college applicants with different metrics from their applications including the decision of the college.

2 CLASSIFICATION PROBLEMS

2.1 Dataset - Video Game Sales

2.1.1 Description 1

Using a dataset and a learning algorithm, we will classify whether a video game will be a "success" or a "failure" based on its global sales.

2.1.2 Why is it Interesting

1. I am curious to see which games come up on top and if it matches the hype.
2. To check if based on this data, my model can predict a nintendo games success such as Super Mario Bros: Wonder in future.
3. From a more practical perspective with a business motivation, a solution to this problem could provide some guidance on risk assessment and market strategy for a game publisher like Nintendo before releasing a game.

2.2 Dataset - College Application Dataset

2.2.1 Description 2

Using a dataset and a learning algorithm, we will predict whether an applicant with certain metrics will get in or not.

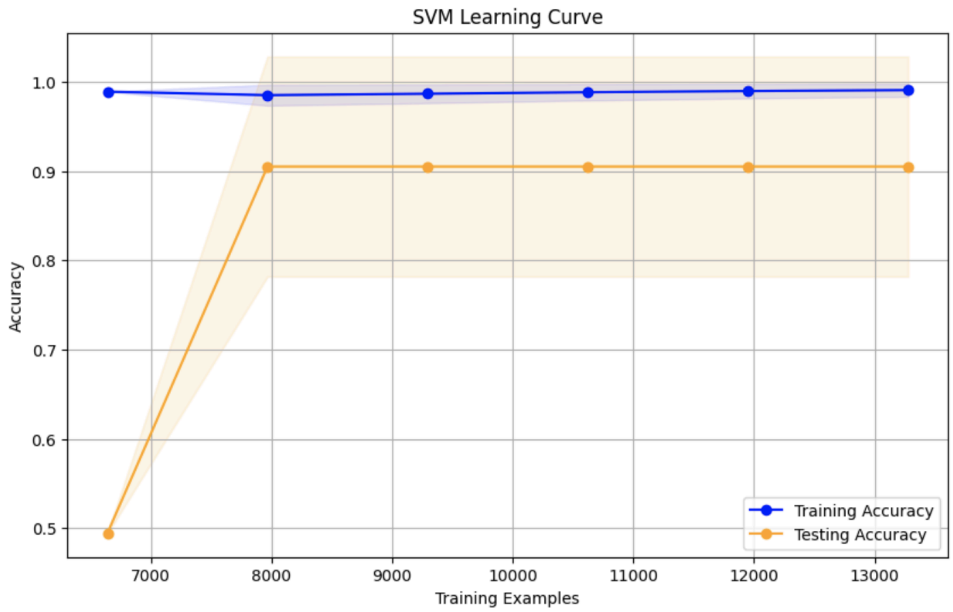
2.2.2 Why is it Interesting

I am curious to know the typical background of applicants who get into a competitive college.

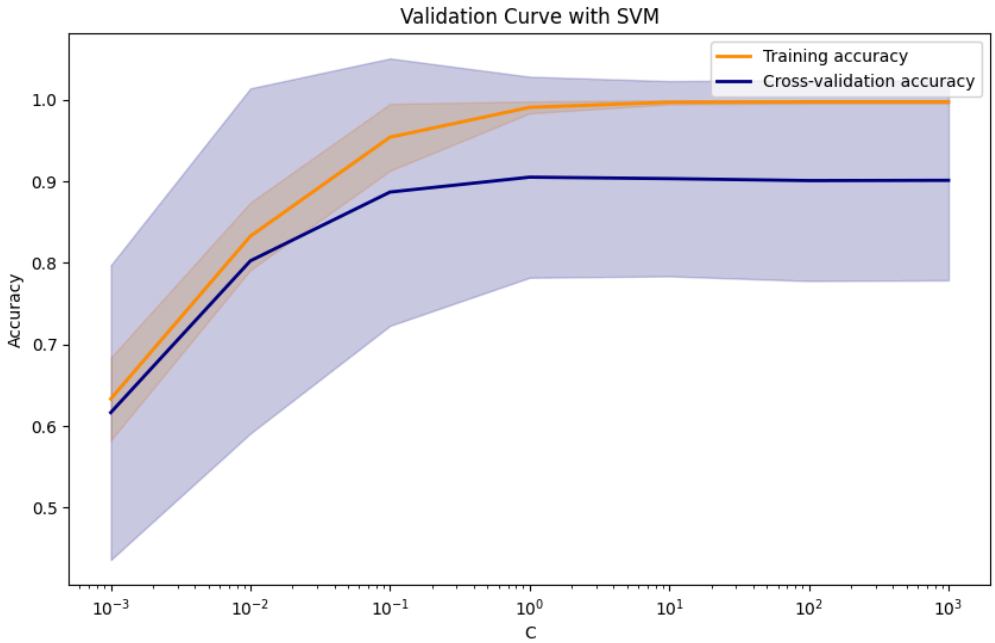
3 THE LEARNING ALGORITHMS ANALYSIS

SVM

In the case of SVM, I tried using different kernels such as linear, rbf, poly, etc. to see which performs the best. Linear and rbf kernel performed quite well on the both training and testing dataset. The accuracy was at 0.99 for both training and testing dataset. The accuracy improved substantially once the number of data points increased to more than 7900. Here's the learning curve:



The linear kernel generally demonstrated much better performance than the poly kernel, which had a training and testing accuracy of 0.79. I think, since the increase in dimensionality by the poly kernel does not correspond to a better division of classes in case of the video games sales data, the simpler linear model performed better due to its lower complexity.
To generate the validation curve, I used a varied C parameter and got the following graph.

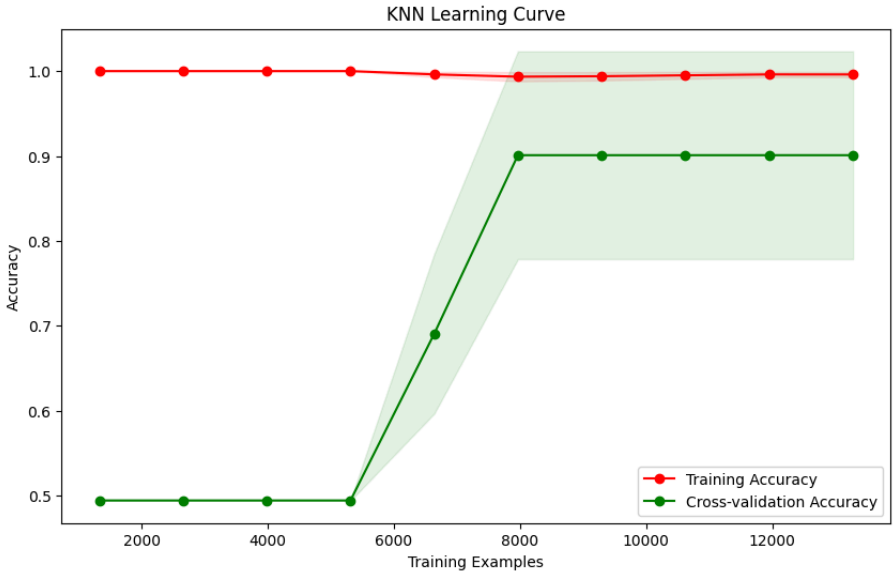


It seems like the model with a C value in between 10 to 100 offers a good balance between bias and variance. This way we can get a model without overfitting.

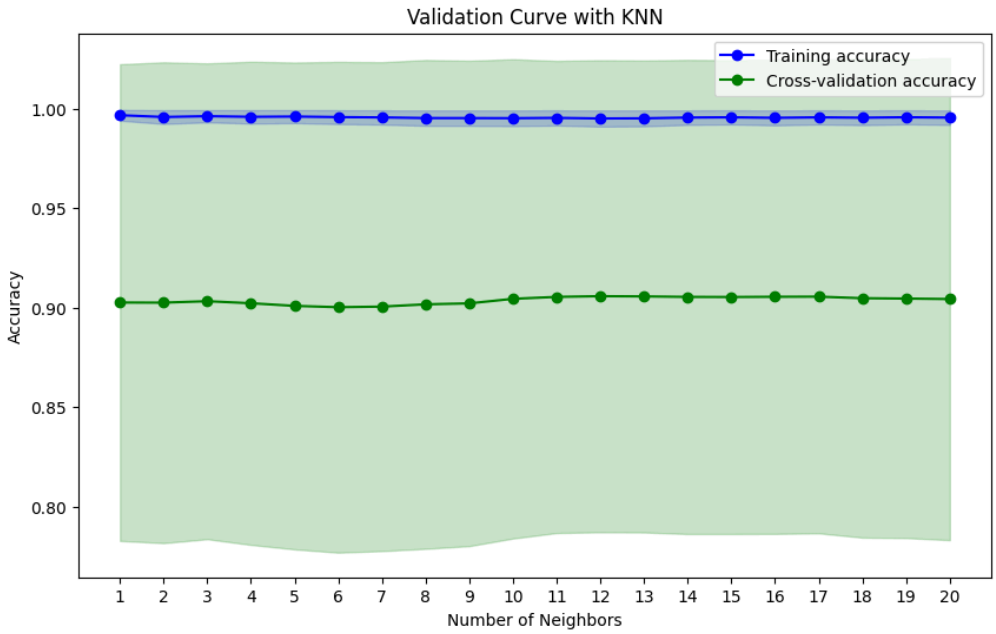
KNN

In case of KNN, first I tried with different number of neighbors(3, 5, and 10) to see which performs the best. In all cases, they performed near identical with a training and testing accuracy of 1.0 and 0.99 respectively. I think this is due to the dataset having regions where the class distribution is very homogeneous. So changing the number of neighbors did not affect the prediction outcome because all or most neighbors belong to the same class. For example, games developed by a major publisher like Nintendo often have high sales figures across all regions, making these regions of the dataset homogeneous in terms of their success. Also, if most blockbuster games by Nintendo are clustered together in the high-sales region of the feature space, a few nearest neighbors would be enough to classify a new game in this region as a success. Similarly, indie games with lower sales figures might cluster in another part of the feature space, where again, only a few neighbors are needed to classify these games accurately.

So I settled down with 5 neighbors and generated the following learning curve.



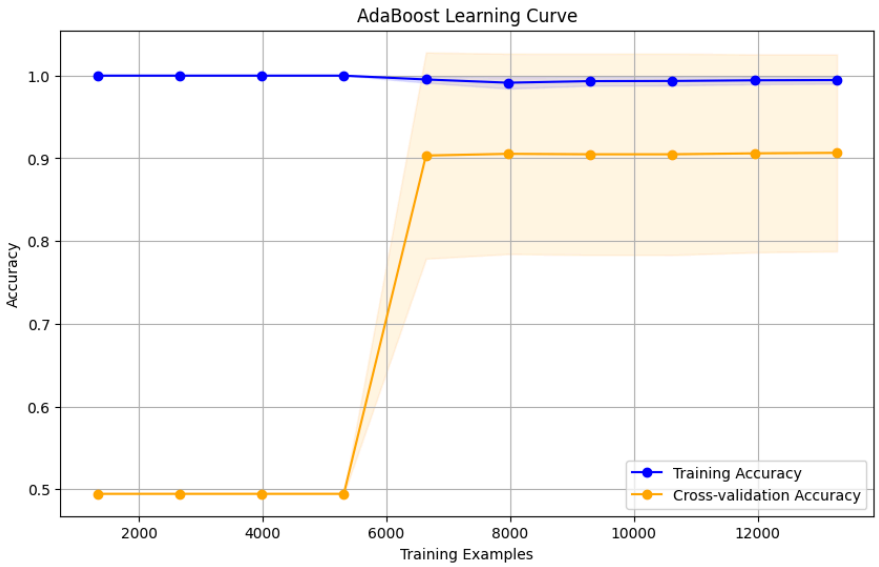
To generate the validation curve, I used a varied number of neighbors from 1 to 21 and got the following graph.



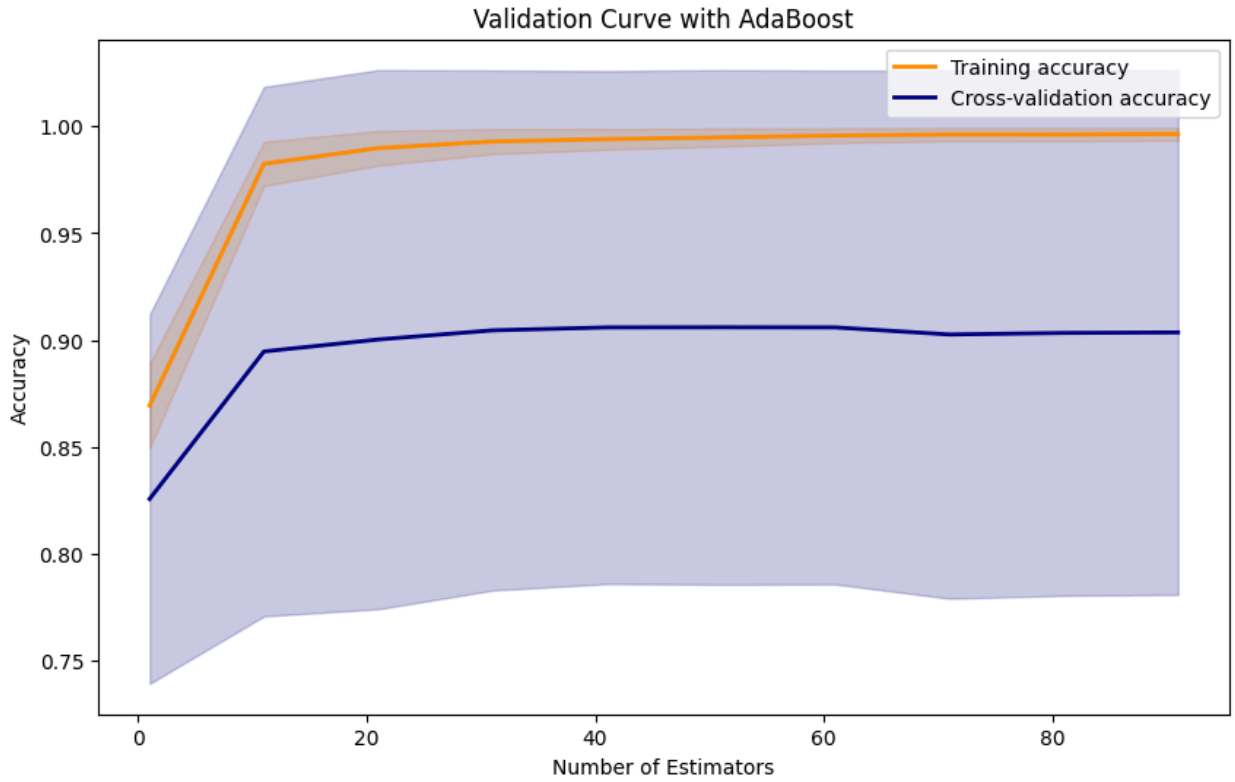
One noticeable thing is that unlike the learning curve, the validation curve does not indicate a clear sign of overfitting, as the cross-validation accuracy does not drop significantly with the increase of n-neighbors which is a sign of good generalization. This is probably due to the quality of the data. The data might be distributed in such a way that the addition of more neighbors does not change the classification decision for most points, probably because the nearest neighbors tend to come from the same class or the class boundaries are relatively clear.

Adaboosting

I tried around different numbers of estimators or weak learners. At around 50 estimators, the model achieved an accuracy score of 0.99. So I settled at 50 estimators and generated the following learning curve:



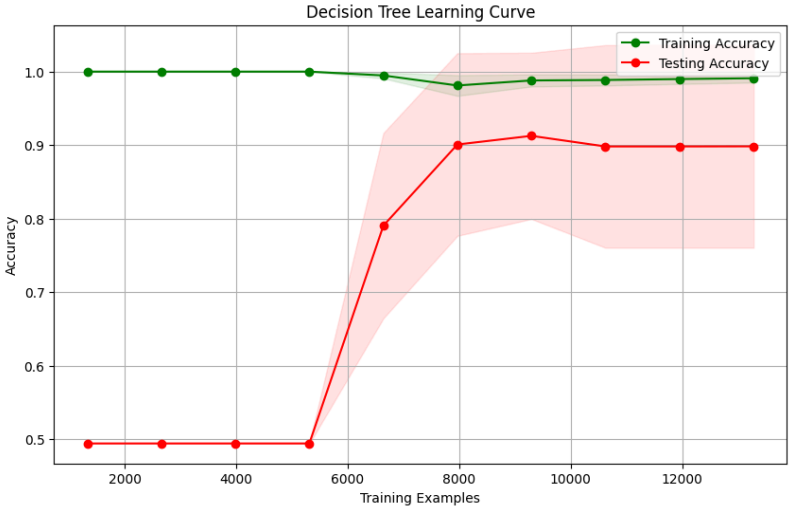
The model accuracy increases significantly once there are more than 6400 data points. To generate the validation curve, I used a varied number of estimators from 1 to 101 at every 10th interval and got the following graph.



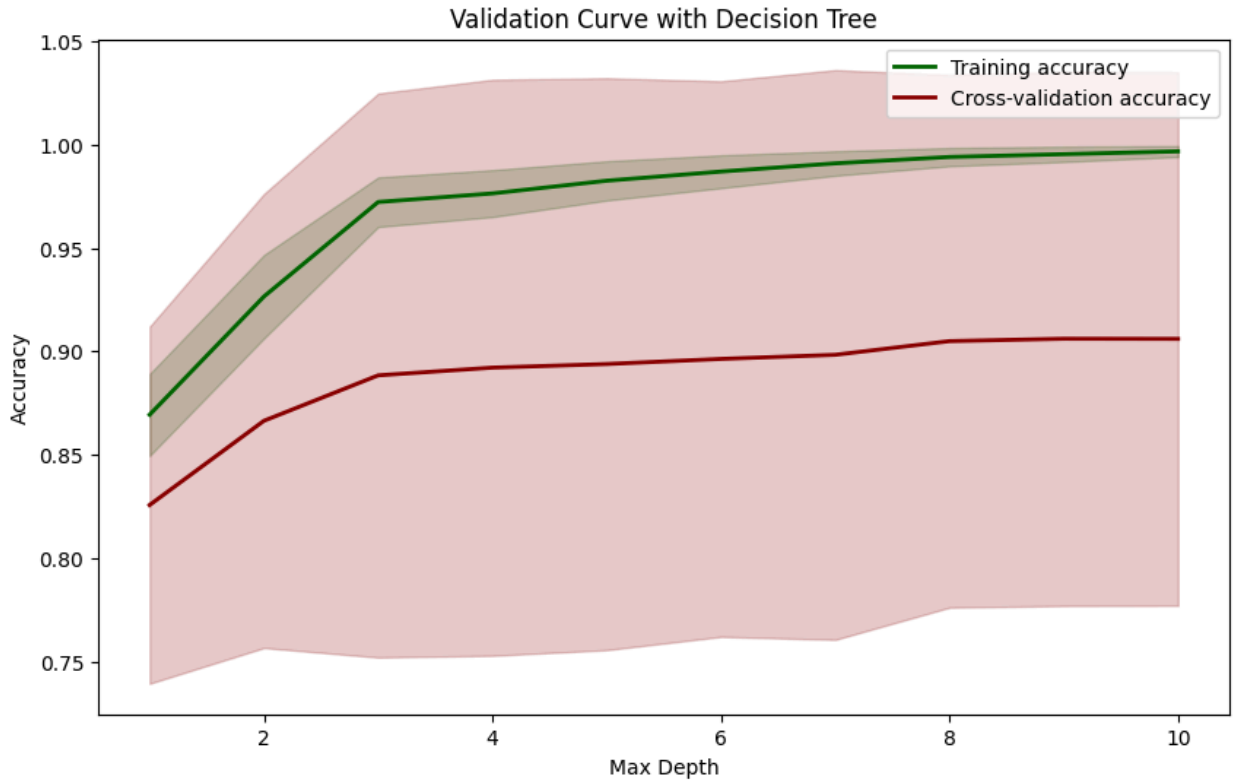
Initially, I thought that 50 estimators was the optimal number. However, from the validation graph the optimal number seems to be 20.

Decision Trees

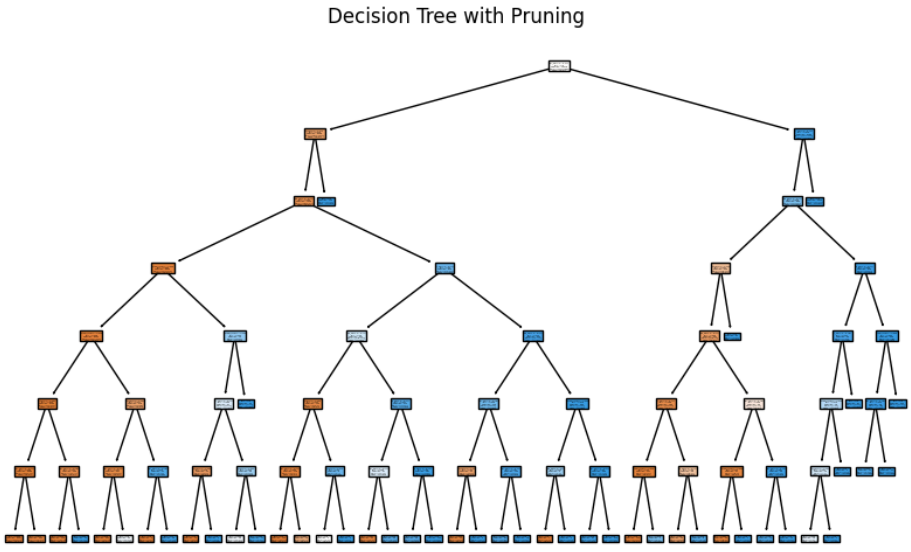
I tried different max_depths and although the accuracy was lower(0.85) at max_depth 1,at max_depth 7, 0.99 accuracy was achieved. Following is the learning curve:



At around 8000 data points, the accuracy plateaus.
To generate the validation curve, max_depth was varied from 1 to 10.



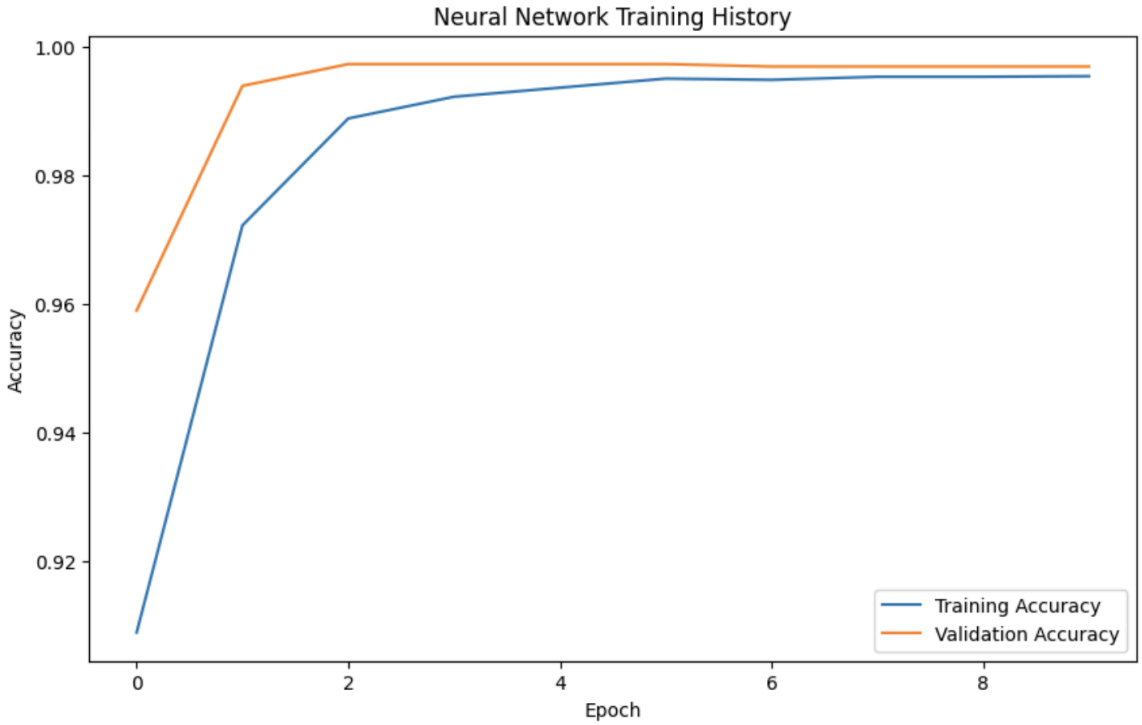
The optimal number of max depth seems to be 7 in this case.

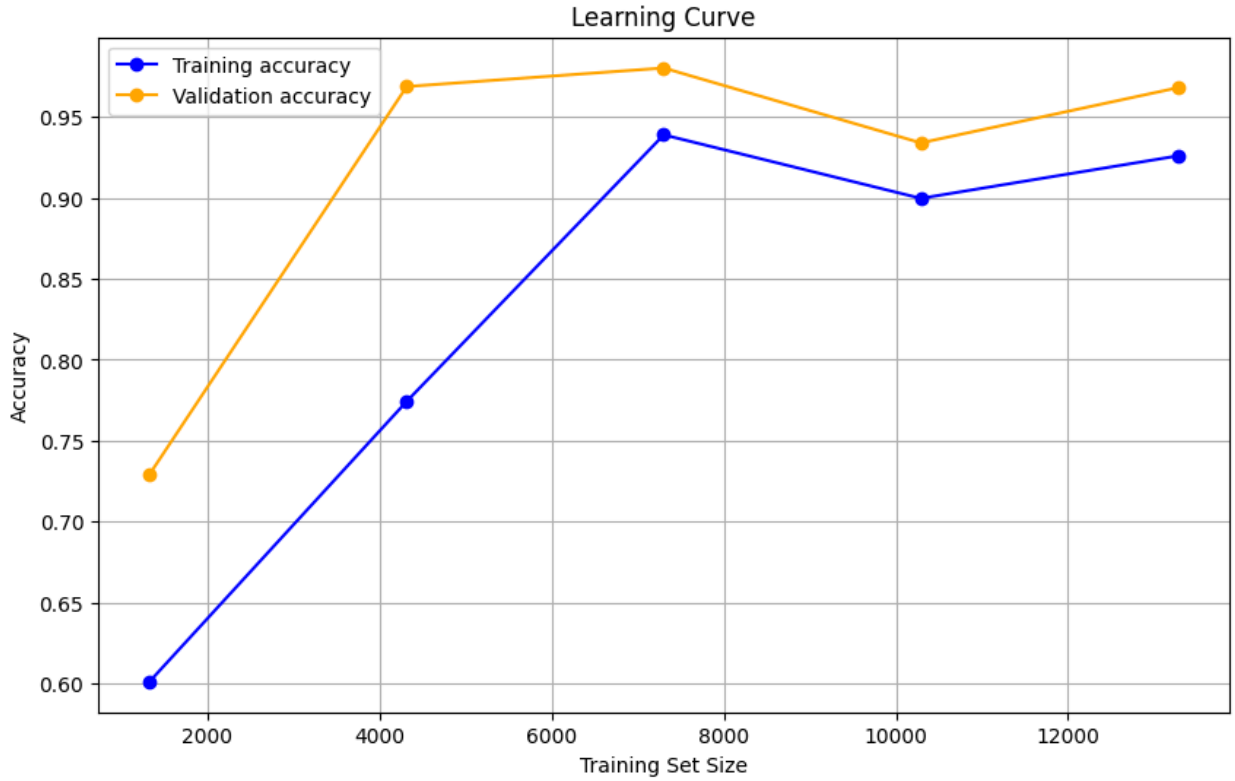


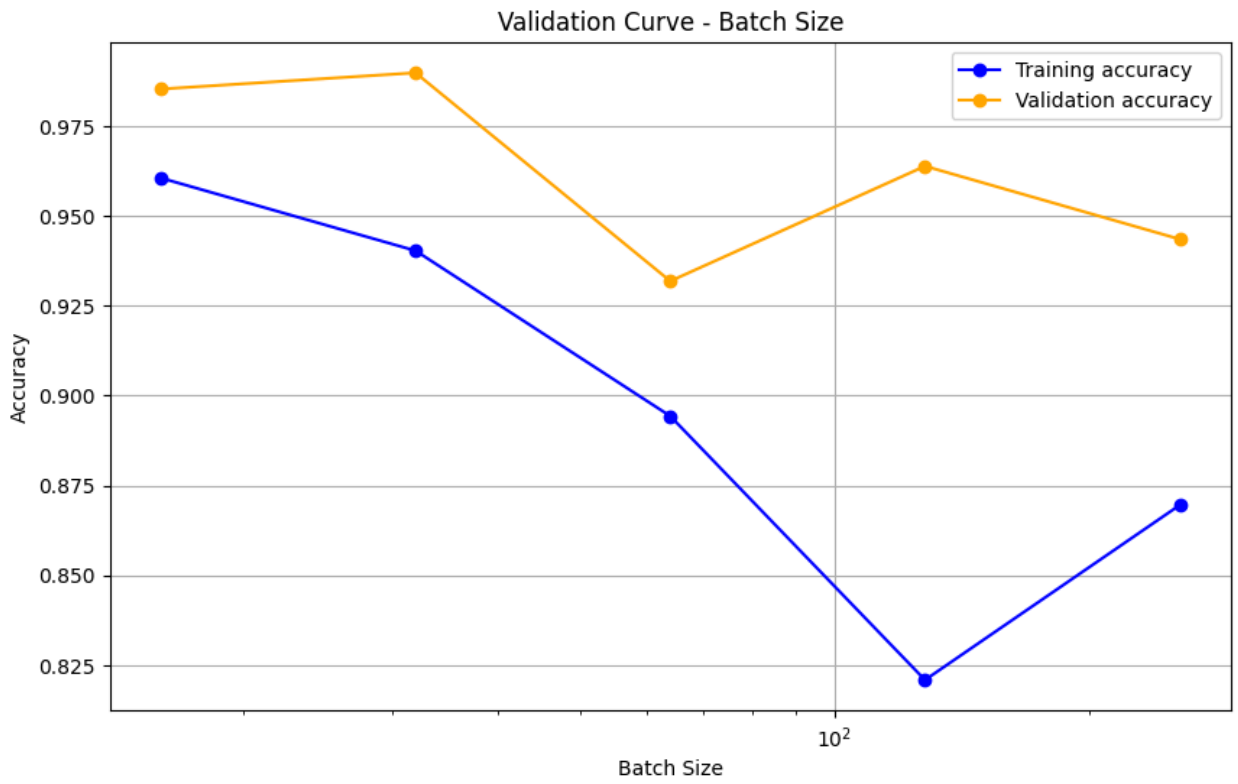
Neural Networks

In case of neural network, I tried around different optimizers like SGD and adam. The

adam performed(0.99) significantly better than SGD(0.85). The loss function was set to 'binary_crossentropy,' since it was a binary classification problem.







From the learning curve we can see that with 4200 or more data points, the accuracy plateaus and the model sort of generalizes. However, for me the bigger revelation was in the validation curve, where I learned that the smaller batch sizes performed much better on this dataset. I think that the smaller batch sizes introduced more noise in the learning process, which allowed the model to generalize better and achieve higher accuracy on the validation dataset.

4 Compare and contrast Results

Out of all of the algorithms, SVM with a linear kernel and KNN performed the best with an accuracy of 0.99 combined with a much faster training speed than any other algorithms. This could be due to the data being not too complex. Other algorithms were able to reach the accuracy of 0.99 as well. However, they took longer to train.

5 Conclusion

The most important thing that I learned is, how important validation curves are to show if the model is overfitting or underfitting. Also, how useful they can be to find out the optimal values for the hyperparameters. This was especially eye opening during the experimentation of adaboosting.