CS 7641 Assignment 1

Datasets

Fashion Pictures is a MNIST-like dataset containing 10,000 greyscale images that are 28 by 28 pixels, each pixel with a value between 0 to 255 representing the lightness / darkness of that pixel. Each picture has a label indicating the type of clothing that is in the picture. The label is encoded as a number between 0 to 9, representing T-shirt / Top, Trouser, Pullover, Dress, Coat, Sandal, Shirt, Sneaker, Bag, and Ankle Bag, respectively. The dataset is very balanced with 1,000 pictures per clothing type, totaling 10,000 samples.

National Basketball Association (NBA) shot logs contain information on shots made during the 2014-2015 season including who took each shot, where on the floor was the shot taken from, who was the nearest defender, how far away was the nearest defender, time on the shot clock, and various other attributes. The label to be predicted is whether or not each shot was successful. The categorical features such as player name and defender name were one-hot encoded, making the final feature count 479. Since the original dataset was particularly large, I truncated the data to only include 15,993 shots (made by 33 players). This dataset is relatively balanced with ~45% successful shots and ~55% misses.

Why are they interesting?

Computer vision problems are a big component of supervised learning applications today and many datasets such as ImageNet and MNIST have been benchmarks for the computer vision community. Classifying clothing pictures is especially important in today's world of social media and e-commerce. Being able to quickly annotate Instagram pictures and online shop photos is incredibly useful when running e-commerce businesses or providing B2B services for e-commerce. One recent trend in e-commerce has been social commerce, where pictures in social media are analyzed in real-time and shopping suggestions are provided for items similar to those in the picture. In this assignment, we will explore the pros and cons of using different approaches to classifying images and their effectiveness.

Predicting shot probability in basketball is helpful for coaches and athletes to understand the driving factors behind a player's shot performance. It is also useful for analysts, fantasy players, and sports bettors for predicting the outcome of a particular game in real-time. From a Machine Learning standpoint, this problem is particularly difficult since there are likely many other factors that are not directly captured in the data, such as positioning of other players and the physical state of the player. It would be interesting to see how well the different approaches can generalize and pickup signals from the limited feature set.

Experiment Approach

The way that each model is setup is to have 20% of each of the total dataset as holdout for testing. Given that both my datasets are fairly sizable, 20% would leave enough data for training and still give meaningful test results. The split is also stratified to ensure that classes are still balanced between the testing and training sets. For each model, I do a Grid Search with 5-fold cross validation to find the best hyperparameters. I chose 5 folds since it would help prevent picking an overfitted hyperparameter set while keeping the runtime to a reasonable length. For each type of model, I will show the learning curves measuring the training accuracy and the average cross-validation accuracy with respect to dataset size

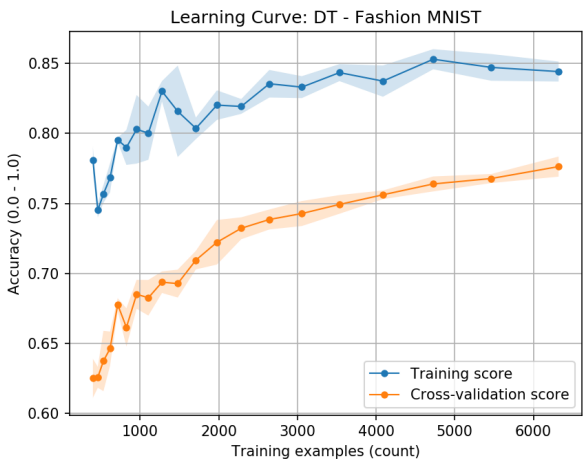
and model complexity. The the best model of each type (based on best cross-validation accuracy) will then be scored on the 20% holdout.

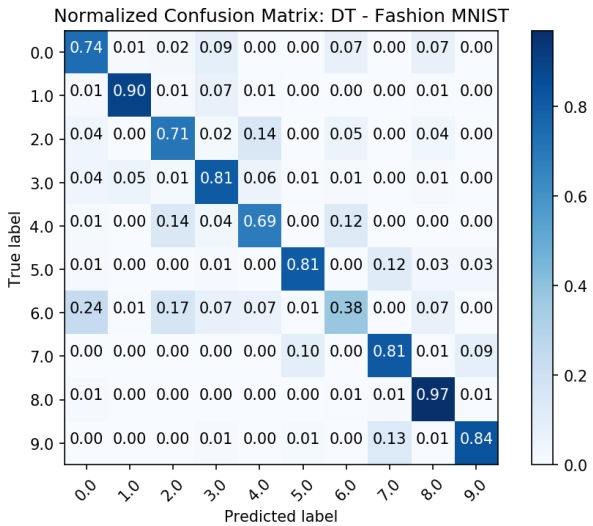
The accuracy metric used was a balanced accuracy, but a regular accuracy metric would have sufficed as well since the datasets are fairly balanced. I chose balanced just as a precautionary measure / good practice. Additionally, I examined confusion matrices for each model to see which labels the models had trouble determining.

Before each model is fitted, standard scaling is done on the features to remove mean and give them unit variance. This is to ensure features with larger magnitudes and variances do not overpower smaller ones, although this is more of a consideration for the NBA dataset than the Fashion Picture one.

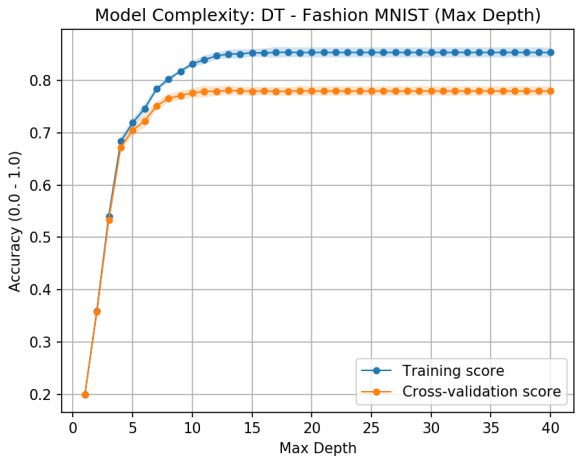
Decision Trees

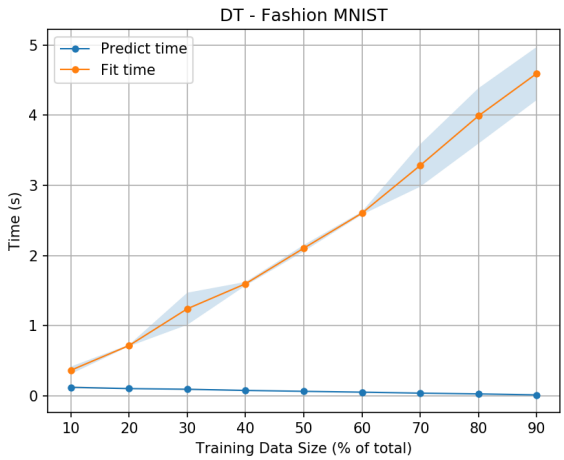
Decision trees were run with both Gini Impurity and Information Gain as the criterion, and pre-pruning was done by fixing the max depth of the trees. The Fashion dataset was modeled reasonably well using decision tree, with the best tree having a test accuracy of **76.5%**. The tree had a maximum depth of 40 and a total of 269 nodes (135 leaves). The final model had entropy as its criterion but there was very little difference between entropy and Gini models overall. This is likely because the two functions are very similar and rarely give different results in practice. However the fitting time for Gini is slightly lower since it's less computationally expensive as it does not require computing logarithms. Looking at the confusion matrix, the label with the most amount of error seems to be in the "Shirt" category (label 6) where the model oftentimes predict "T-shirt/Top" (label 0) or "Pullover" (label 2). This is understandable since, from a visual standpoint, the only differentiating factors are the sleeves, collars or buttons which are small patterns that may be difficult for a tree to capture and generalize.



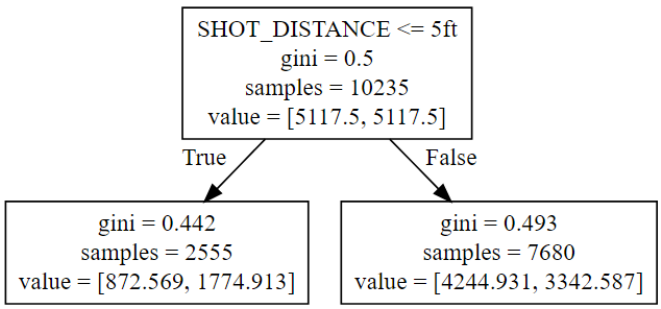


Max depth pruning did not result in a better model as shown in the bottom left graph. I believe this is because vision problems in general have a large and complex feature set (in this case 784 pixels) which require a high level of model complexity to learn. The training time for decision trees is about expected; it looks fairly linear in the training data size is reasonably inline with its theoretical run-time of $O(n_{samples} n_{features} \log(n_{samples}))$ according to scikit-learn documentation.

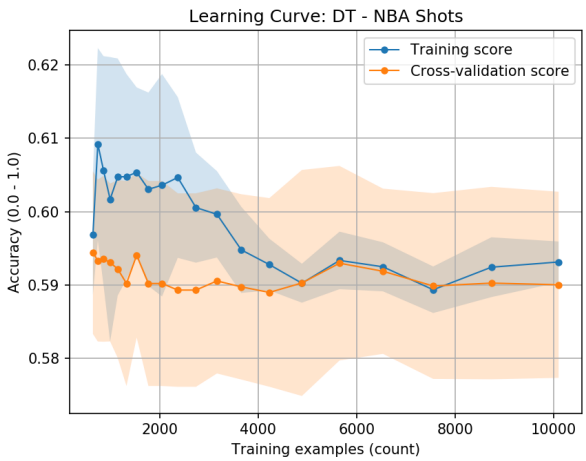


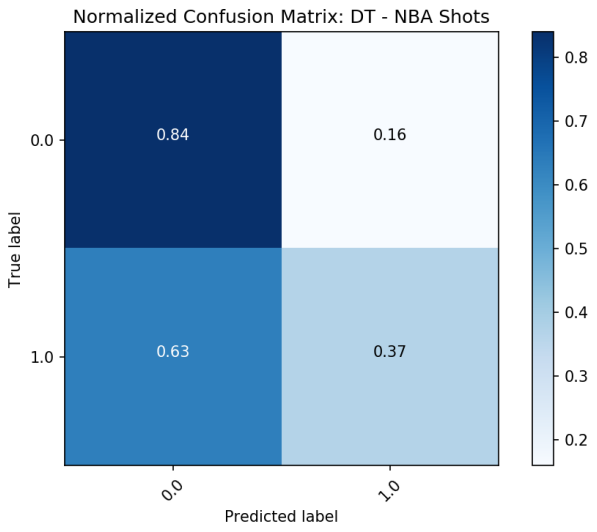


The NBA dataset was surprisingly difficult to predict, but a simple decision tree seemed to have produced the best classifier of all the methods with a testing accuracy of **60.4%**. Unlike the Fashion dataset, the shot data benefited greatly from max depth pruning to the point where the best model was a decision stump. This is because the signal to noise ratio in the data is quite low and it's very easy to overfit the data with additional complexity.

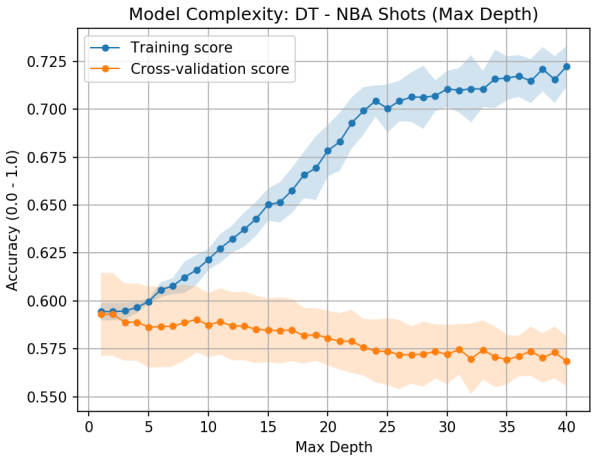


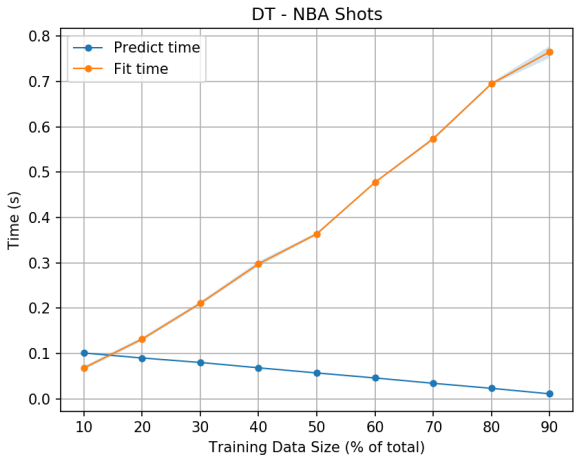
The single factor that had the best predictive power was whether or not the shot distance was less than or equal to 5 feet. Although this is intuitive, I would have expected that other factors would contribute to the predictive power as well such as distance to defender and shot clock. The learning curve shows that with small number of examples, it's easy for the decision tree to overpredict, but with enough samples the cross validation and training accuracy converge. In terms of accuracy, most of the misclassification seem to come from erroneously predicting missed shots as successful ones.





The bottom left graph shows the benefit of pre-pruning on decision trees: the overfitting drastically increases on this dataset as the max depth grows but cross validation accuracy slowly dwindles. In datasets where the complexity of the approximated mapping is lower in complexity, tree pruning is essential to preventing overfitting. As with the Fashion dataset, the training time is inline with expectations that it will be approximately linear in the size of the sample set.

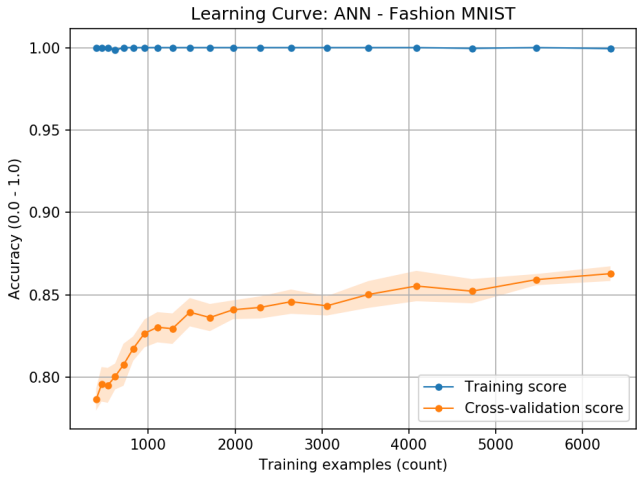


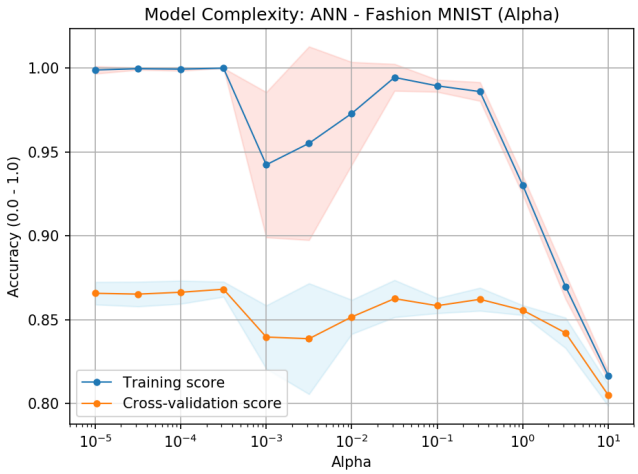


Neural Networks

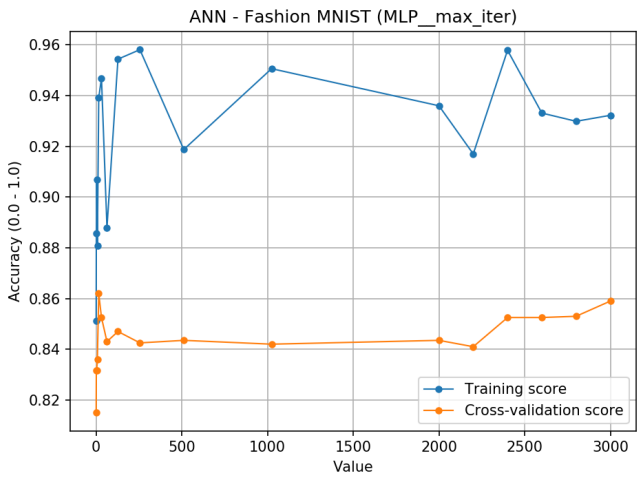
For Artificial Neural Network, I tested a number of learning rates, hidden layer sizes, and activation functions to determine the best setup for each dataset. The two activation functions I tested were ReLU and Tanh since they were both less likely to suffer from the diminishing gradients than the Sigmoid function. Unsurprisingly, ReLU turned out to be the best activation function for both models due to faster learning and being able to create sparser models.

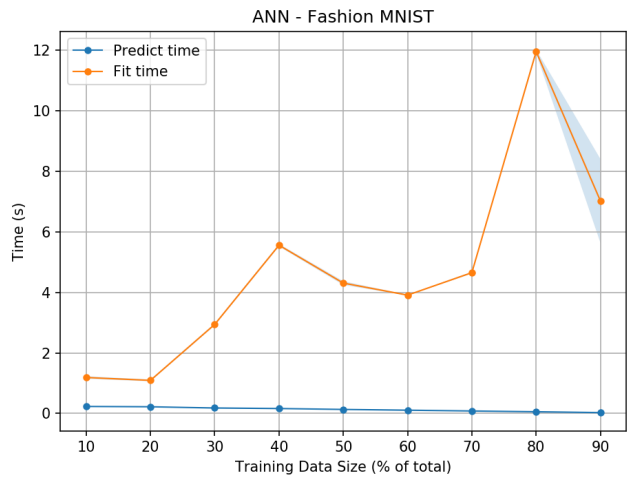
Neural networks worked quite well with the Fashion picture dataset, achieving a testing accuracy of **86.3%**. The best model had one hidden layer with 784 nodes, the same number as the number of features. However, other architectures with 1-3 layers of ~400-1600 nodes also performed with similar accuracy (within 1%). This goes to show that, for this particular problem, various configurations of the neural network can sufficiently learn the mapping. One interesting observation is that the training accuracy is almost always 100% even when using 6,000+ data points (shown in chart on bottom left). This indicates the model architecture is prone to overfitting and isn't generalizing enough during training. In other words, when training accuracy goes to 100%, the classifier has no more ability to learn. It is surprising that even large values of alpha, the L2 regularization parameter, did not help much in addressing this issue, instead it decreased cross validation accuracy with training accuracy (shown in bottom right graph). My guess would be that the complexity of the model is much higher than the complexity of the mapping so it's difficult to prevent overfitting. In future iterations, it would be interesting to see the effect of dropout on the training and test accuracy of these models.



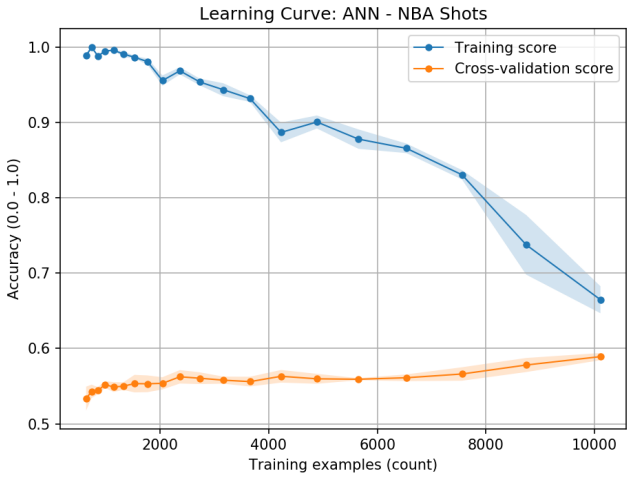


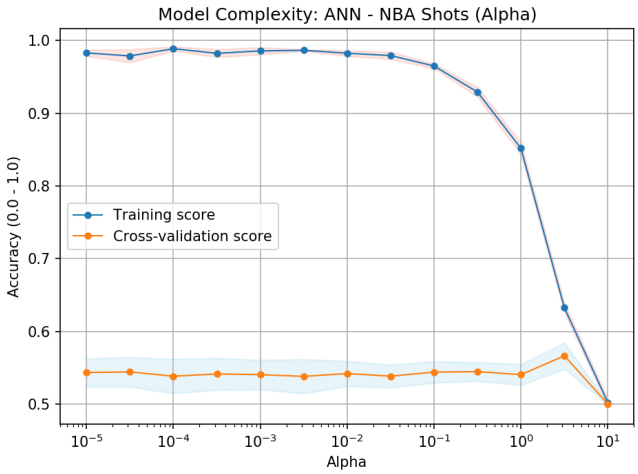
The number of epochs did not drastically change the accuracy of the model after approximately 100 epochs. I believe this is due to two factors: 1) I enabled early stopping so training stops if the cost function does not see an improvement after 10 consecutive epochs, and 2) the optima for the objective function (with 100% accuracy) is found rather quickly. With the Adam optimizer I expected the learning curve to be bumpier since it is stochastic gradient-based, however point 2) previously mentioned may have offset that.



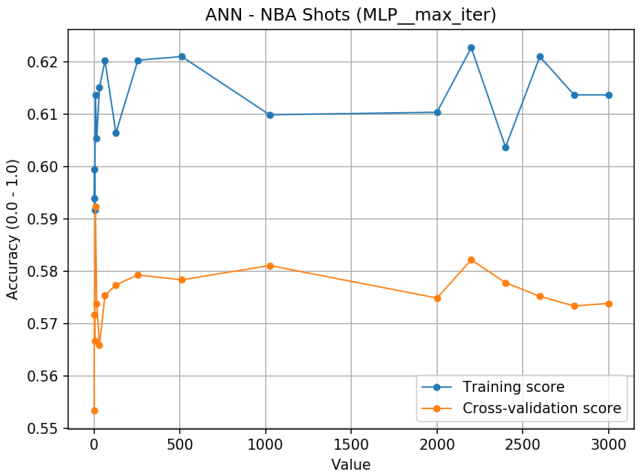


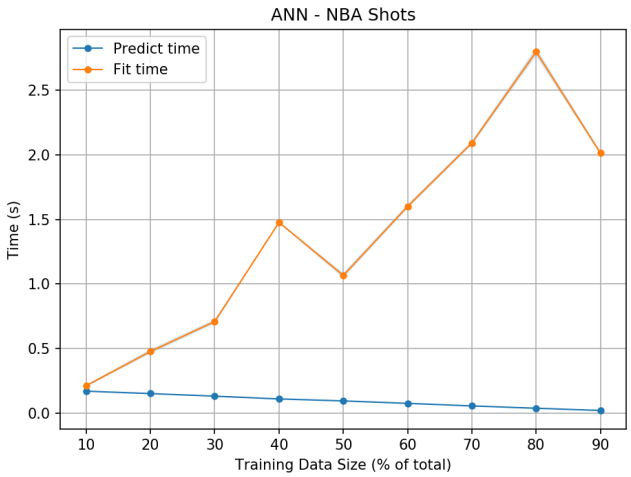
Neural Networks performed decently well for the NBA dataset. The best model had a testing accuracy of **58.1%**, and had 2 hidden layers of 239 nodes each and L2 regularization parameter of 3.16. The NBA dataset showed some interesting results in the learning curve (bottom left graph). As training examples increased, the training accuracy drastically decreased and slowly converged with the cross-validation accuracy. One cause for the convergence at a low accuracy rate is possibly because the Bayes error, the lowest possible error rate given the dataset, is likely high for this dataset . With respect to the regularization parameter Alpha, it did not have a material effect on overfitting until sufficiently large values (between 1 and 10), with a sweet spot at approximately 3 which is reflected in the best model's parameters.





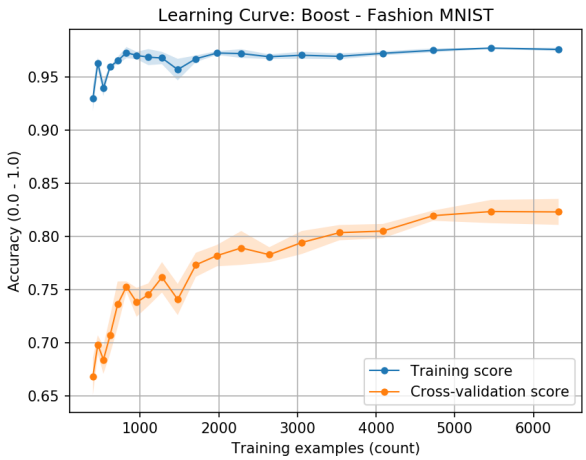
The iteration learning curve looks similar to the one for the Fashion dataset, with the accuracy tapering off fairly early on in the number of epochs. The fit time for this dataset is considerably lower than that of the Fashion dataset. The lower fit time is expected since the NBA dataset has far fewer features and the hidden node counts were based on the number of input features, resulting in smaller networks on average.

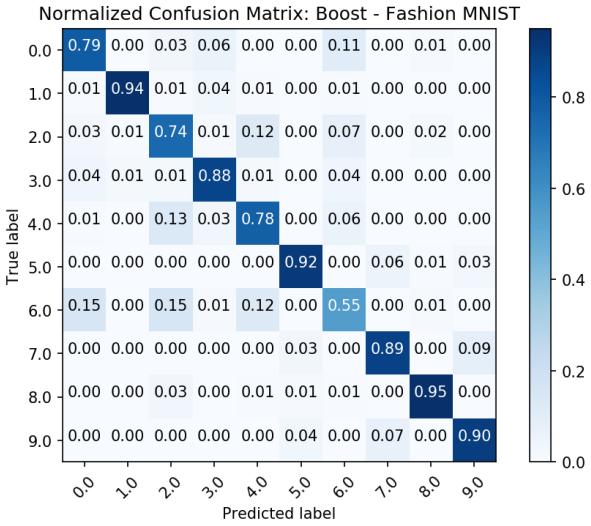




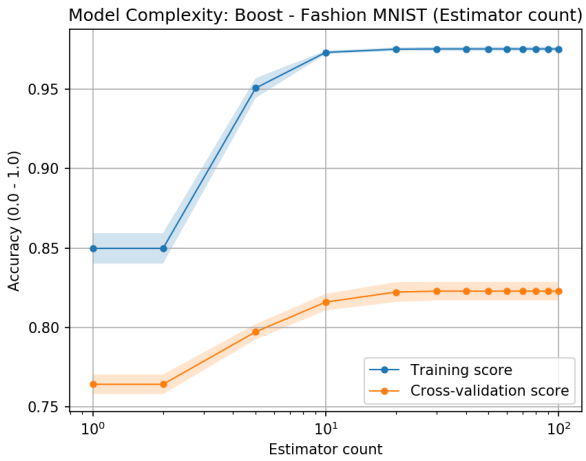
Boosting

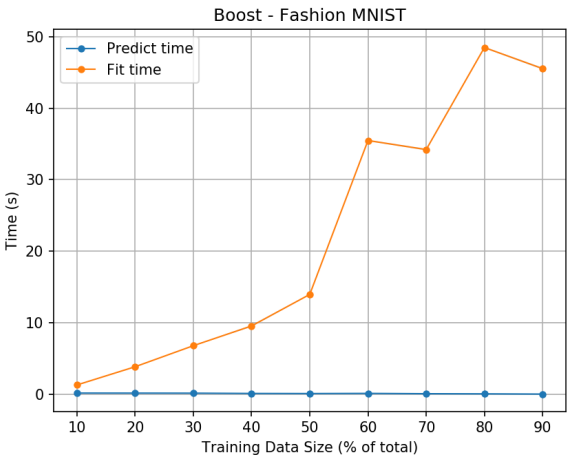
The Boosting model experiment was performed using 1-100 Decision Trees of maximum depth 1-40. For the Fashion dataset, the learning curve is similar to that of Decision Trees but with better accuracy overall. The the final Boosting model achieved **82.9%** test accuracy with 20 trees of depth 16. Looking at the confusion matrix, the mislabeling of "Shirts" (label 6) has gotten better than Decision Tree but still not great (~55% accuracy).



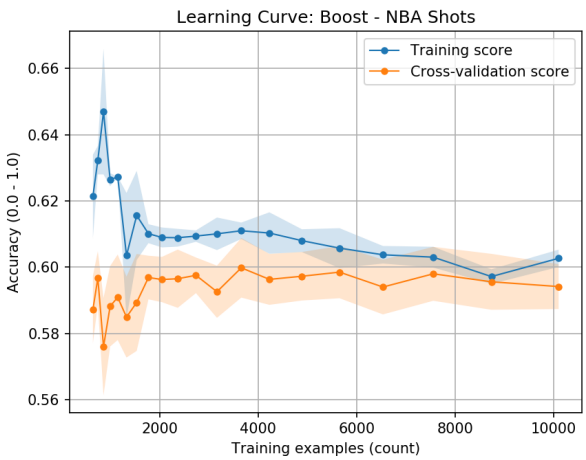


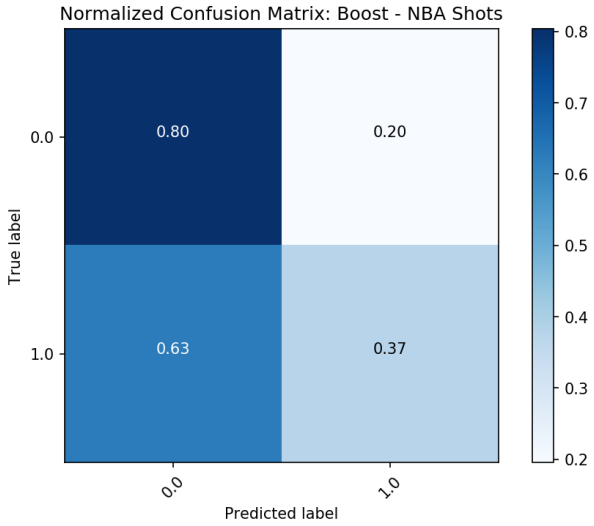
In terms of model complexity, the number of estimators / trees seemed to improve the model performance up until a certain point (~15 trees) where it begins the plateau. This is likely because the leftover variance after the ~15th tree is very difficult for the trees and features to capture and generalize effectively. The other complexity metric tested was the maximum depth of trees. This did not prove to be an important factor for this dataset for the same reasons mentioned in the Decision Trees section. The fitting time is substantially longer Decision Trees since the algorithm needs to fit many trees sequentially during training.



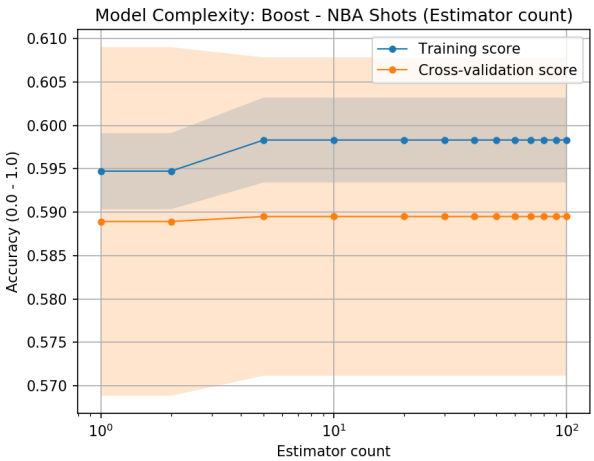


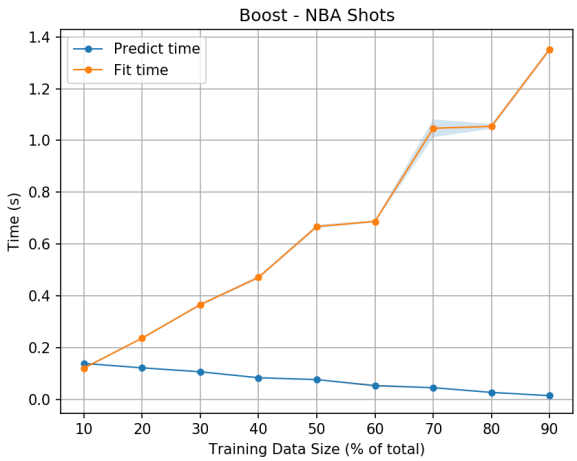
Boosting performed similarly to Decision Trees for the NBA dataset. The best Boosting model had testing accuracy of **59.0%**, and, similar to the final DT model, it on the simpler side with just 5 trees with max depth of 3. The confusion matrix looks similar as well, where the successful shots were well predicted but the majority of the misses were mislabeled as successful.





Increasing the estimator count increases the performance until about ~5 estimators, at which point the performance tapers off. Also, for each point on the estimator curve, the cross validation score range is extremely large which indicates other parameters are likely better levers for performance than estimator count. This makes sense as we know the best model so far has been a decision stump which is just one estimator. The learning curve (bottom right) shows that the fit and predict times are approximately proportional to the number of samples processed.

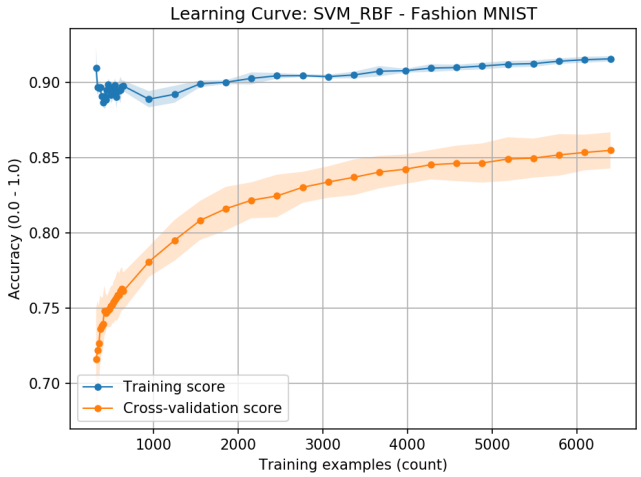


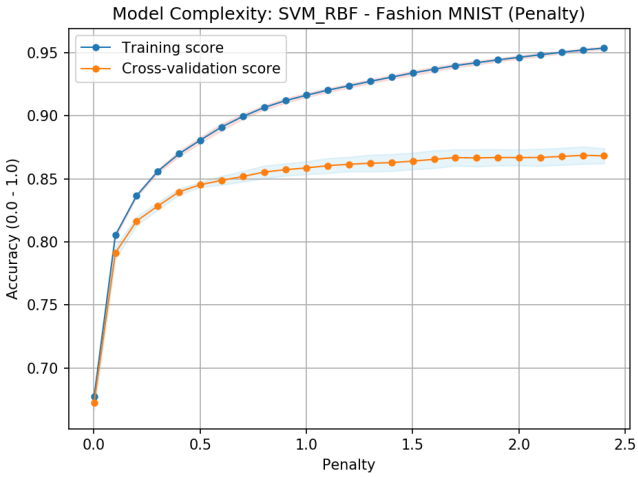


Support Vector Machines

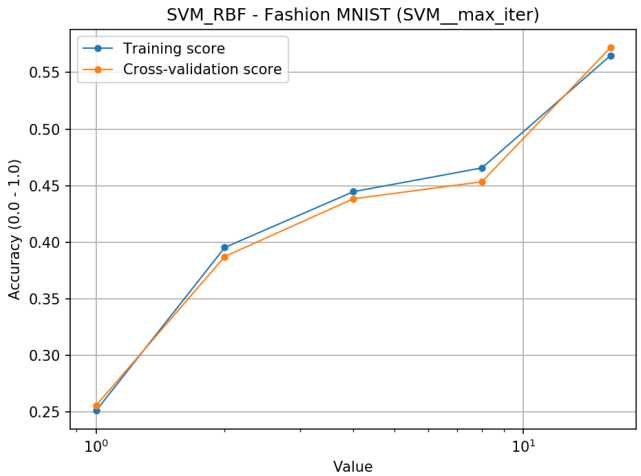
Support Vector Machine experiments were run with varying penalty terms, max iterations, stopping tolerances, kernel coefficients, and two kernels (Linear and Radial Basis Function). The penalty term "C" determines the amount of regularization in the model, and the max iteration / stopping tolerance determines at what point the training of the learner stops. The kernel coefficient "Gamma" was only applicable for the RBF kernel and was a fixed constant for the majority of the top performing RBF models.

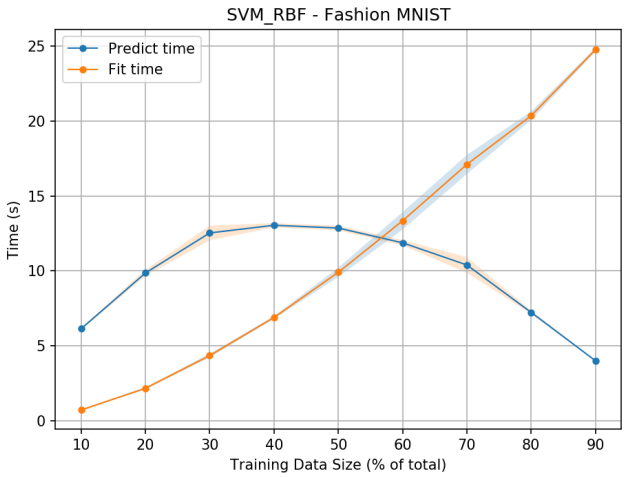
The best performing setup for the Fashion dataset had testing accuracy of **86.6%** which was better than the best Linear kernel model which had 82.5% accuracy. It also beat out Neural Network for the best model on this dataset. It used the RBF kernel with a 1.25 penalty parameter, no iteration cap, and a stopping tolerance of 10^{-8}. The stopping condition was the strictest which let the model run for a large number of iterations before stopping. The penalty parameter was near the high end of the range, meaning it was on the less regularized side. I believe the model setup was ideal for this dataset because with a large dataset and multiple classes, a relatively non-regularized RBF model that is allowed to train for a large number of iterations would be able to capture the main characteristics of the mapping.



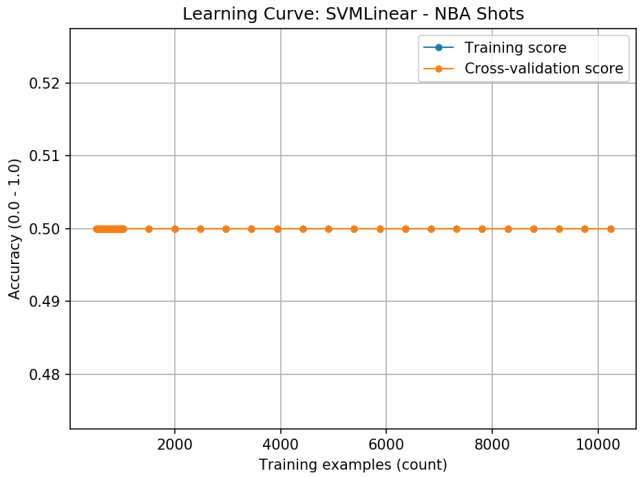


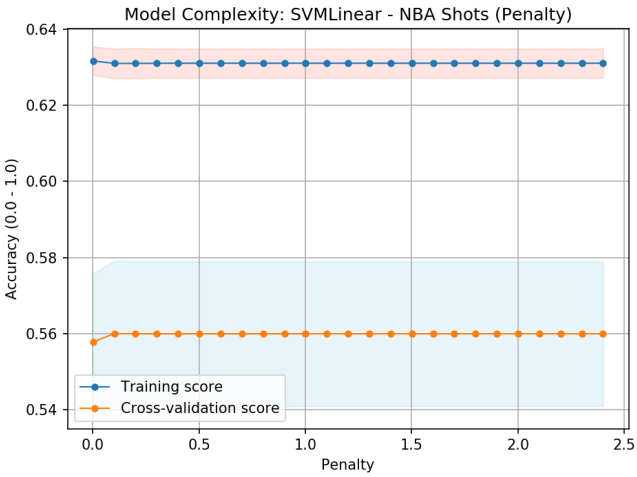
With respect to the number of iterations for RBF kernel, we see that the accuracy tends to improve slightly with each number of iteration whereas the linear kernel almost immediately tapers off (shown in image output folder). Another interesting thing to point out is the relationship between training time and predict time as the number of training sample increases. Unlike most other models, it seems the predict time is actually comparable with the predict time. In fact, they're almost the same when the fit and predict sample sizes are evenly split (50-50).



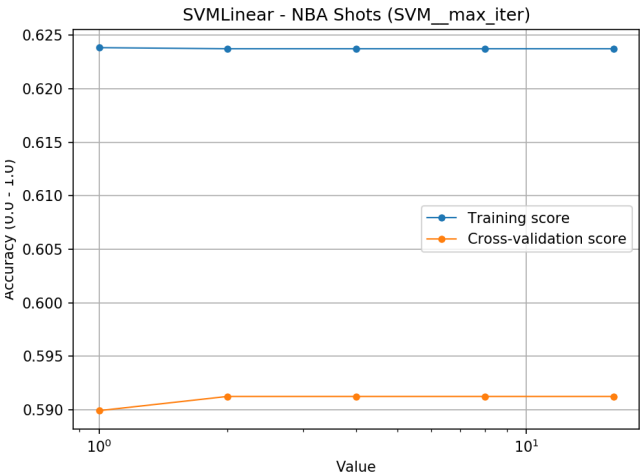


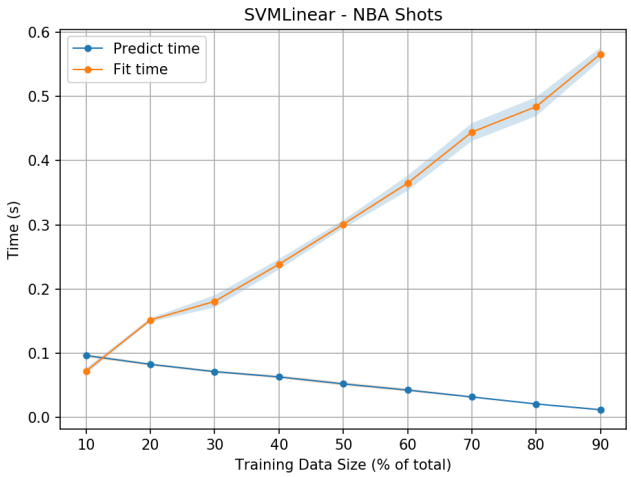
For the NBA dataset, the Linear kernel proved to be better than RBF with a testing accuracy of **59.1%** vs. RBF's 57.9%. The parameters for the Linear kernel model had a penalty parameter of 0.5 and max iteration of 79. The lower penalty parameter makes sense for this dataset as it's easier to overfit so a larger margin would be better for generalization. The linear kernel also makes sense because, as we've seen from the Decision Tree classifier, one or two linear boundaries are sufficient to achieve ~60% accuracy. The learning curve on the bottom left looks peculiar as it's always at 50% accuracy regardless of training examples. I believe that's a result of the Linear kernel trying to find the best fitting hyperplane when the data isn't linearly separable, which always happens to split the data in portions that are about half correctly labelled. The bottom right chart shows that, on average, the model accuracy isn't very sensitive to the penalty term, although the range of cross-validation accuracy is very high for each point. This indicates Linear kernels are much more sensitive to the other parameters in the setup than it is to the penalty term.





Similar to the penalty term, the max iteration parameter did not have a large impact on the accuracy of the model; after a certain point (~2 iterations), the performance stabilizes. The time chart on the bottom right shows that the fit time and predict time are directly proportional to the number of samples.



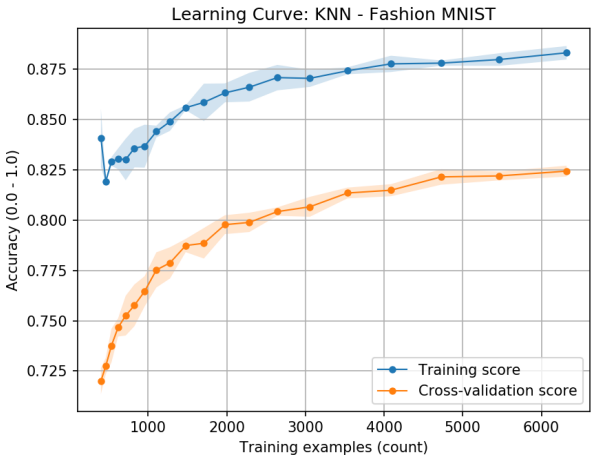


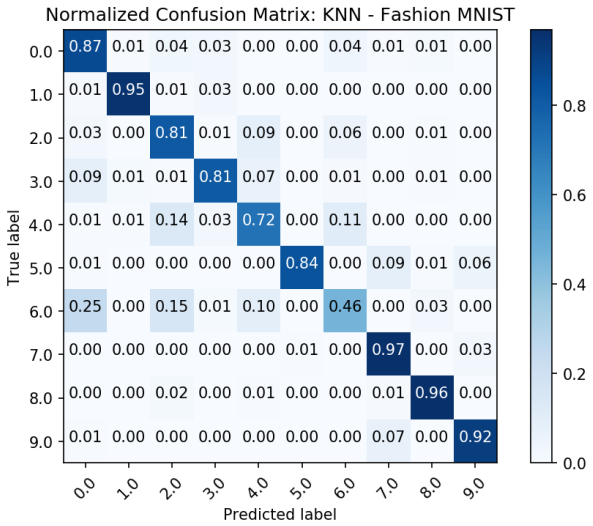
k-Nearest Neighbors

The k-NN experiments were run with 1-50 neighbors, three types of distances (Manhattan, Euclidian, Chebyshev), and two types of weights (uniform, distance). Both Manhattan and Euclidean distances produced good models for both datasets, whereas Chebyshev distance did not perform as well. I believe this is because Chebyshev distance only captures the difference in the most distant dimension and was not using the difference in all the other dimensions. Both Manhattan and Euclidean distance captures differences in all the dimensions, which is more informative for the algorithm. The weighting by distance or uniform did not make a significant difference in model performance as the top models for both datasets used either quite interchangeably.

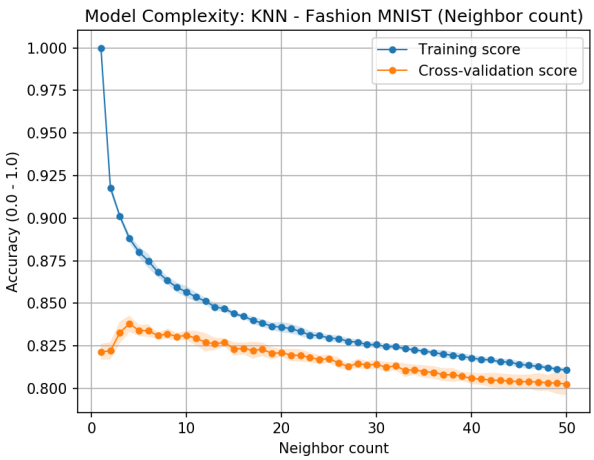
The best model for the Fashion dataset uses 4 neighbors, Manhattan distance, and uniform weighting. It achieved a testing accuracy of **83.3%**. The low number of neighbors makes sense for this dataset since there are a large number of classes and the distance would intuitively measure the difference between the pixel values (which are bounded), making the clusters fairly close together. This point is further explored later on in the complexity graph. One drawback of approach is that the distance would likely be influenced by overall brightness and contrast of the picture which could lead to inaccuracies. The learning curve on the bottom left shows that with more training data, the k-NN model continues to become more accurate but does not learn as quickly as SVM RBF or Neural Network. This makes sense as generally k-NN is an instance based learning method and is very data-hungry. In the confusion chart, the model is having

trouble classifying "Shirts" (label 6) which is understandable since, from a picture point of view, the pixel distributions would be similar to that of "T-shirts" (label 0) and "Pullovers" (label 2).

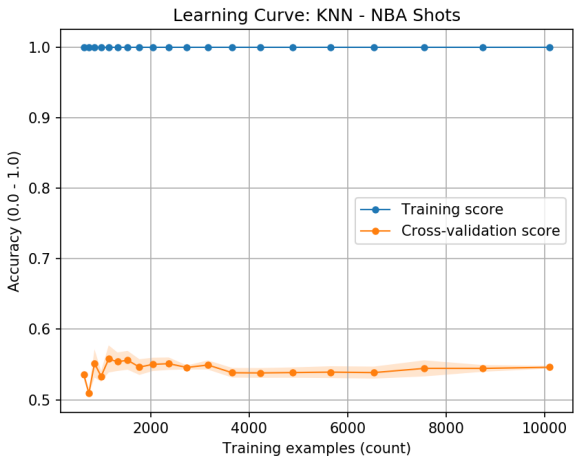


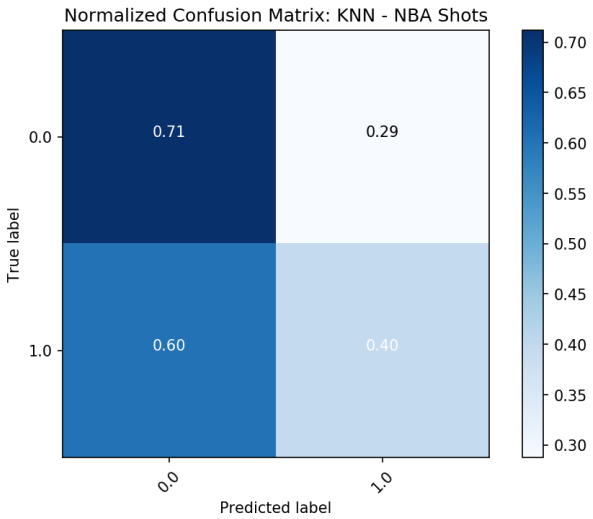


One interesting observation about the model complexity graph (bottom left) is that as the neighbor count grows, the performance gets worse. This is because there are large number of classes and they are likely to be fairly close together (based on the fact that the distance is only based on raw pixel values). When taking into account farther neighbors, there is a higher likelihood of misclassifying the image. In terms of timing, k-NN is the only algorithm that generally has higher predicted time than fit time. This is because each time a new data point is to be classified, the distance between that data point and every training instance needs to be calculated.

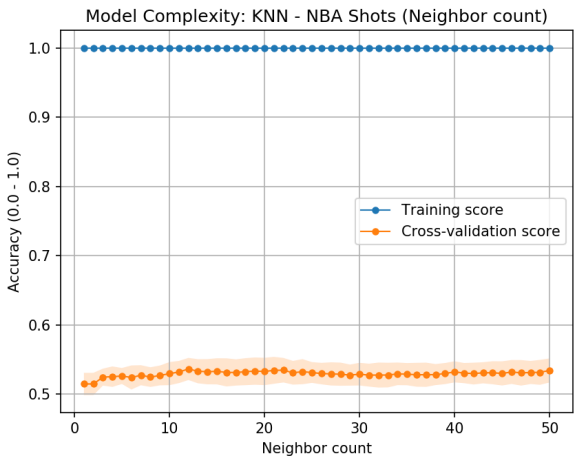


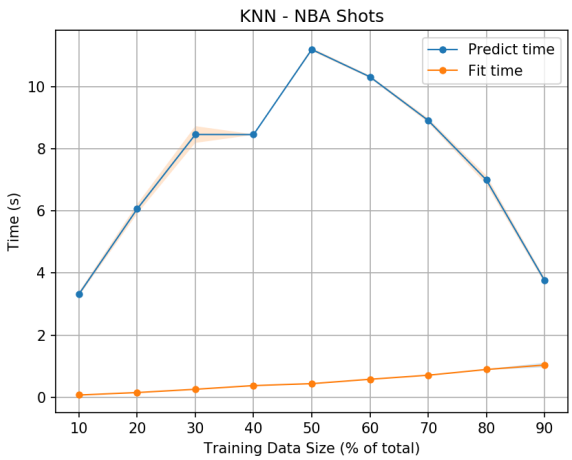
For the NBA dataset, the k-NN algorithm performed relatively poorly, with the best model achieving a testing accuracy of **55.5%**. The optimal parameter choices were 40 neighbors, Manhattan distance, and distance weights. The reason I think it performed poorly is because the dataset is noisy with only a few features that actually contribute to the prediction (e.g. shot distance). Since the distance takes into account all the dimensions, the large number of non-useful features drown out the signal that comes from the one or two useful ones. As such, k-NN would not be suited for these kinds of datasets where the predictive power is concentrated in a small minority of the features.





The learning curves for both number of training samples as well as the neighbor count show that 1) neither have a substantial impact on performance after a certain point and 2) overfitting to 100% training accuracy occurs extremely frequently with this dataset. I believe the second point is because there are only two classes and there are a relatively large number of features. The timing curve is similar to that of the Fashion dataset: predict time is substantially higher than fit time due to distance computations.





Conclusion

The different algorithms explored in the assignment all have their own pros and cons, and some are more suited for certain datasets than others. The Fashion image classification dataset was best modeled by high complexity non-linear models such as Neural Networks and SVM RBF, whereas the binary NBA shot dataset was best modeled by tree based or linear split-based methods such as Decision Trees, Boosting (with few shallow trees), and Linear SVM. Another takeaway was that k-NN, the instance based method, performs poorly on datasets that have features that are highly skewed in terms of predictive power. SVM was also interesting because it had a large number of parameters and could be configured to model a variety of different datasets and relationships. In the future, I would experiment more with certain aspects of each of these algorithms, such as introducing dropout and convolutions for Neural Networks, and trying ensemble methods for Decision Trees.