Project 1: Supervised Learning

Scott Merrill
smerrill7@gatech.edu

I. Introduction

The class of supervised machine learning algorithms attempt to approximate a function that maps inputs to labeled outputs. A wide range of supervised algorithms exist, each with unique characteristics. The appropriate algorithm for a given problem is thus dependent on the amount and type of input data as well as training and testing constraints. This project explores the properties of several supervised learning algorithms on two datasets to examine the types of problems certain algorithms may outperform. We specifically train Decision Tree, AdaBoost, Neural Network, Support Vector Machine and K-Nearest Neighbor models and explore the effects various hyperparameters have on the bias-variance tradeoff.

II. Data Sets

To evaluate the properties of supervised learning models, we'll look at two NFL datasets that differ broadly in terms of size, number of features, types of features, number of outliers, runtime constraints and balance of positive examples.

A. NFL Scores Dataset

The NFL Scores Dataset [1] contains scores and betting lines of all NFL games dating back to the 1970s. The goal of this dataset is to predict whether a team will cover point spread quoted by Vegas Sportsbooks. Since Vegas spread bets typically only pay -110 (meaning a $110 bet wins $100), correct classification 52.4% of the time is necessary to break even and thus is the target accuracy for our models. Also, because game lines are often set days in advance, training and run-time constraints are ignored in this problem.

The raw dataset contains only NFL scores, game lines and weather conditions and thus significant pre-processing and feature engineering were required to extract explanatory variables. In total, 34 continuous features were created from the underlying dataset. Each of these variables are continuous and were each standardized to have a zero mean and standard deviation of 1. The features are largely correlated and can be broadly categorized as either a metric for wins or points scored.

In total, the dataset contains 6,068 games of which the home team covered the spread on 3,011 occasions. Thus, dataset is almost perfectly balanced with the home team covering 49.62% of the time. Since the dataset is nearly balanced, the Jaccard index is used to compute accuracy score which is simply the percentage of correct predictions.

B. NFL Play-by-Play Dataset

The NFL Play-by-Play Dataset [2] contains the results of every NFL play run from 2010-2019. The objective of this dataset is to predict whether the next play will be a running or a passing play. Further, for models in this domain to be decision useful and allow for live bets to be placed, testing needs to be performed before a team runs a play. There is usually around 30 seconds between plays thus testing time can take no longer than 30 seconds.

The raw dataset contains many continuous and categorical variables that are largely uncorrelated. Further, the variables vary broadly in terms of categorization ranging from game-based metrics that measure score, time and yards to informational categorizations of teams and players. While no features were specifically manufactured from the dataset, pre-processing was still necessary to encode categorical variables. One hot encoding was done to each categorical variable. Additionally, many plays such as kickoffs, field goals, and punts were dropped from the dataset as our model will not be used to predict 4th down or special team plays. Further, only a subset of the dataset from 2016-2019 was considered and all plays from prior seasons were ignored.

The resulting dataset after preprocessing contains 93,471 entries in total with 38,238 (40.9%) runs and 55,233 (59.1%) being passes. Moreover, since the dataset is imbalanced an accuracy measure that takes into account the relative imbalance of the dataset is thus required. Since we have no preference over false positives or false negatives, we'll use the F1 score as our measure for accuracy. The F1 score is a weighted average of the precision and recall; precision measures the number of true positives as a fraction of all positive predictions and recall is the number of true positives as a fraction of true positives and false negatives. Thus, precision penalizes a classifier for falsely classifying a negative example as a positive and recall measures a classifiers ability to correctly classify all positive examples.

C. Additional Comments on Datasets

While both datasets contain information on NFL games and the goal of both is to produce actionable sports bets, the datasets differ significantly in terms of the prediction task, features set, correlation between features, size, number of outliers, balance of positive examples and runtime constraints. These differences will enable for interesting comparisons between supervised learning algorithms; some algorithms may perform better with more data while other models may be better for modeling a noisy dataset with many outliers. Additionally, some algorithms may be impractical given runtime constrains. Moreover, selecting datasets with very different properties may provide insight on when certain models should be preferred and when they should be avoided.

III. Modeling with NFL Scores Dataset

A. Decision Trees

We first attempt to model the NFL Scores dataset using the default decision tree in sklearn which uses the gini index to optimize tree splits and sets no constraints to prune leaf nodes. The learning curve for this default decision tree classifier is shown in Figure 1. The classifier produced high bias as indicated by the low validation scores and high variance as indicated by large deviations between the training and validation curves. The variance of the model is likely a consequence of significant overfitting. With no maximum depth or pruning, the tree building algorithm iterates until each leaf node is pure; that is, each leaf node contains only positive or negative samples. Such results in overfitting as can be seen with the perfect training accuracy. To counter overfitting, pre-pruning is considered by adjusting both the maximum depth and minimum leaf size parameters.

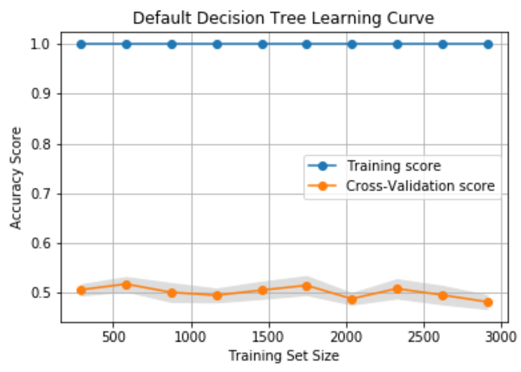

Figure 1

The depth of a tree corresponds to the length of the longest path from the root of the tree to a leaf node. As the depth of the tree grows, the model becomes more complex and is more likely to identify spurious relations between features and output classes. On the contrary, a depth too small may prevent the tree from identifying important patterns. To find the optimal balance between over and under-fitting with regard to the max depth parameter, we plot the validation curve in Figure 2. While increasing the depth improves training accuracy, validation accuracy degrades with increased model complexity indicating overfitting. Thus, smaller depth values will lead to models with lower variance and similar bias to larger depth models.

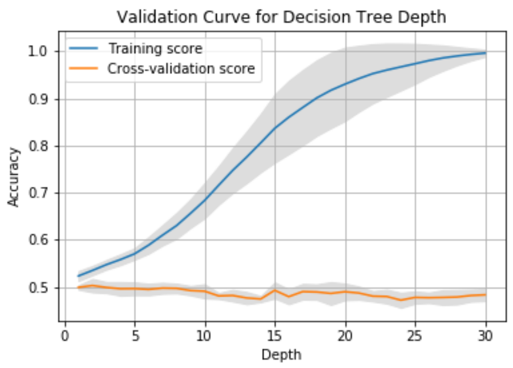

Figure 2

The minimum leaf size parameter constraints the tree from creating leaf nodes with fewer samples than the set value of the parameter and is thus another parameter that can pre-prune a tree and combat overfitting. Larger values of the parameter result in less complex trees with smaller depth while smaller leaf sizes will result in more complex trees that are more likely to overfit. This relationship can be seen from the validation curve shown in Figure 3. With very small leaf sizes, perfect or near perfect training accuracy is observed. As the number of minimum leaf samples grows, training and validation scores begin to converge indicating a reduction in overfitting and a model with less variance. On the chosen dataset, however, bias seems largely invariant to different leaf sizes and thus larger leaf sizes may be preferred as they reduce variance without sacrificing too much accuracy.

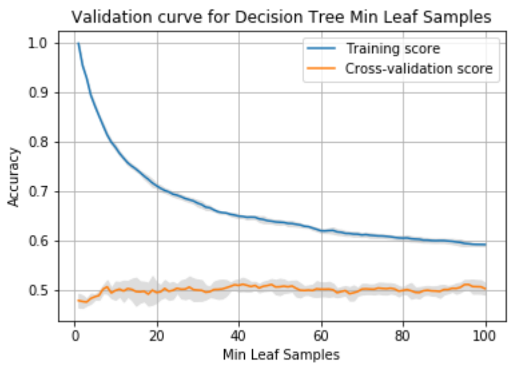

Figure 3

In addition to depth and leaf size, many other hyperparameters can be set which to limit the size of decisions to reducing overfitting. Additionally, the mechanism by which a tree chooses which node to split on may affect the performance of the model. The typical approach is to split on the node with the highest gini impurity but other similarity heuristics such as entropy are also used in practice. We use grid search to optimize all decision tree parameters. The exhaustive search technique identified the optimal depth, minimum leaf sample size, minimum weighted fraction of leaves, minimum splitting samples and splitting criterion to be 5, 10, 0.03, 2 and gini impurity respectively. The learning curve of this optimized classifier is shown in Figure 4. Both the training and validation accuracies begin to converge with increasing sample sizes indicating a significant reduction of variance from the default model. Bias is also slightly improved as validation accuracies are higher irrespective of sample size. Lastly, the validation accuracy appears to increase monotonically with sample size, indicating more data may work to both improve accuracy and reduce variance.

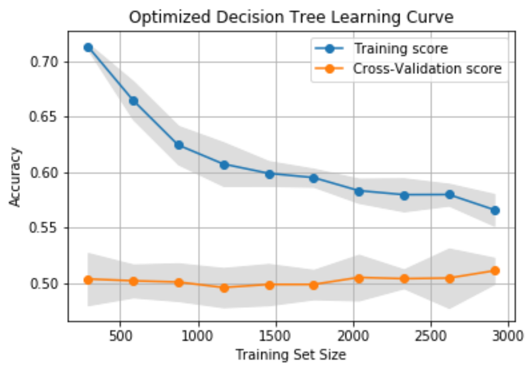

Figure 4

The optimized model was used to make predictions on the test set and resulted in an accuracy score of 51.1%. From the confusion matrix in Figure 5, we note that the model predicts a team will fail to cover the spread a higher percent of the time, however the model is also more accurate with these predictions; when predicting a team will fail to cover the spread it is correct 53.1% of the time. As a result of predicting more teams will fail to cover, however, the model produces many false negatives. Such isn't a huge issue in the scope of this problem as in betting we are indifferent to false negatives and false positives; we are more concerned with maximizing true positives and true negatives.

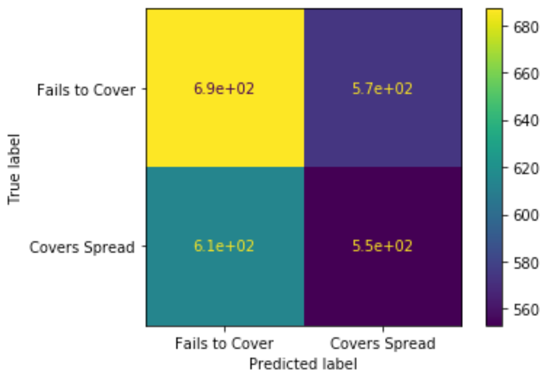

Figure 5

B. *AdaBoost*

The NFL Scores Dataset is next modeled with the default AdaBoost classifier in sklearn which considers averaging over 50 decision tree stumps (i.e. decision trees with max depth of 1). The learning curve for this classifier is shown in Figure 6.

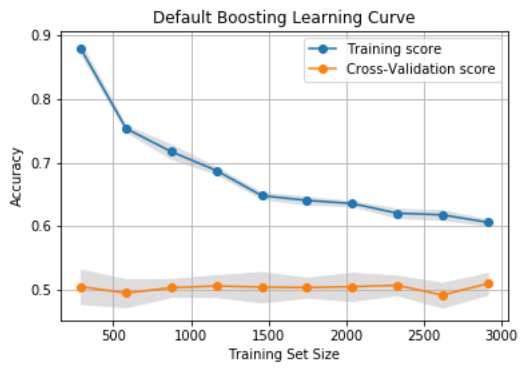

Figure 6

The model produces accuracies on par with the default decision tree classifier. Variance of the model is large with small training sizes as indicated by large differences in the training and validation accuracies. These variances, however, begin to converge as the training set size and thus more data would likely be beneficial to further reduce variance. Without more data, however we will consider reducing this variance by adjusting the number and types of AdaBoost base estimators.

The number of learners in an AdaBoost classifier has a significant impact on overfitting and model complexity. Since misclassified examples from one learner are given more relative weight in subsequent learners, more learners will result in more complex models that are more sensitive to outliers. The NFL Scores dataset likely contains many outliers whereby an underdog unexpectedly wins, or Vegas spreads are quoted inconsistently. Moreover, fewer AdaBoost learners may be beneficial in reducing model complexity and prevent the model from overfitting to the noise in the dataset. To balance overfitting and complexity, the validation curve for the number of estimators is shown in Figure 7. Validation accuracy initially increase with the number of learners, but the benefit of more learners is muted past 20 learners. As the number of learners is further increased, training and validation accuracies diverge indicating a more variance and overfitting.

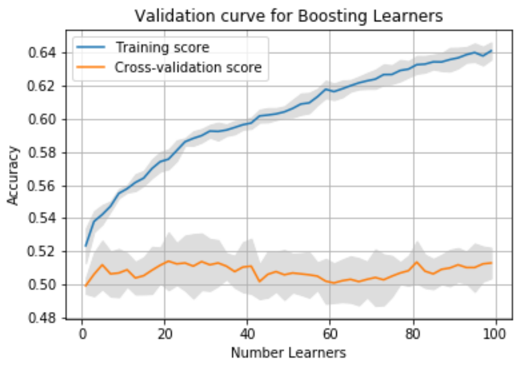

Figure 7

The performance of AdaBoost is also contingent on the quality of the base learners. The base learners in AdaBoost need to be weak learners that can correctly classify examples 50% of the time. Given each learner in AdaBoost is a weak learner, AdaBoost will theoretically produce a more efficient, strong learner. Decision tree stumps that classify whether the home team covers the spread may not all be weak classifiers as some features may not be very predictive when considered in isolation. Moreover, increasing the depth of the underlying decision trees may increase the accuracy of the base learners and improve AdaBoost's performance. Figure 8 shows the validation curve for AdaBoost when the depths of the underlying decision trees are varied. Increasing the depth provides little improvement to accuracy of the model. In addition, large depths result in the base learner's overfitting which carries over to the AdaBoost classifier as a whole; this can be seen by the perfect training accuracies when the depths of the underlying trees exceed 5. Overall, by varying the depth of the underlying decision trees, bias remains constant and thus the underlying trees may still not be weak learners.

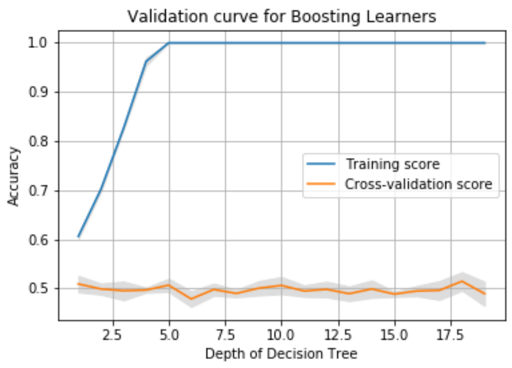

Figure 8

Both decreasing the number of learners and increasing the depth of the underlying decision trees in isolation appear to provide marginal improvements over the default classifier. Additional hyperparameters, such as the learning rate and properties of the underlying base estimators can have significant impact on the performance of the model. Grid search was used to simultaneously find the optimal set of parameters for AdaBoost and its base estimators. The optimal number of estimators and learning rate were found to be 7 and 0.95 respectively. Additionally, the optimal underlying decision tree learner was found to have a max depth of 2 and minimum leaf samples of 20. The resulting learning curve for the optimized classifier is shown in Figure 9. Bias and variance appear slightly improved over the base classifier. Further, accuracy appears to increase as the training set is increased passed 2,500 samples, indicating additional data may further reduce bias and variance of the model.

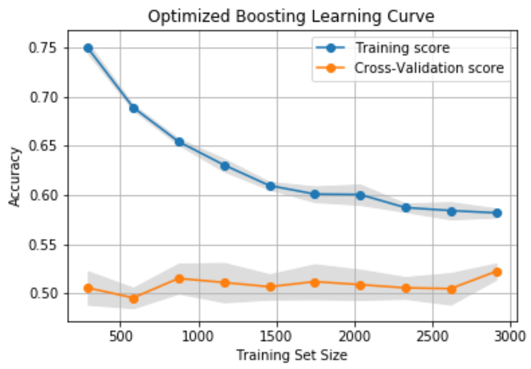

Figure 9

Using the optimized AdaBoost classifier, predictions were next made on the test set which produced an accuracy of 49.3%. The confusion matrix for these predictions is shown in Figure 10. This test accuracy measure falls more than 2 standard deviations worse than the validation accuracy indicating the data in the training set may not be representative of the test set or underlying data generating process.

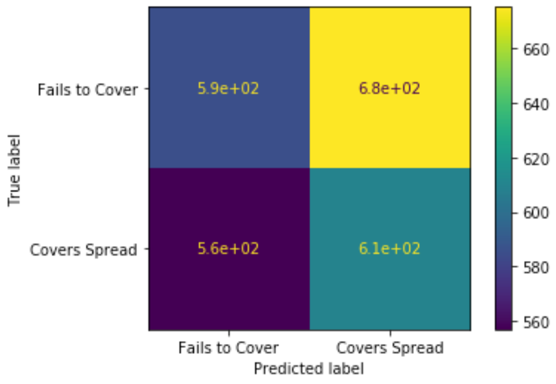

Figure 10

C. *Neural Networks*

We next evaluate the performance Neural Networks using the default MLP classifier in sklearn which considers a network with 1 hidden layer of size 100. The loss curve for this default Neural Network is shown in Figure 11. The default Neural Network results in both high variance and high bias indicated by an increasing validation loss and decreasing training loss. Given the NFL Scores dataset only contains 34 features, a hidden layer of size 100 – more than 3 times larger than the number of features – results in significant overfitting as indicated by the low training loss.

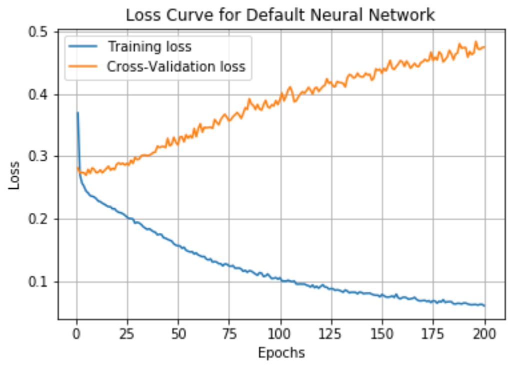

Figure 11

To tune our Neural Network, an optimal architecture that balances overfitting with the ability to generalize needs to be identified. In the default classifier, a single hidden layer of size 100 significantly overfits, which makes sense since a network of this size contains 4,791 trainable parameters.

Considering our dataset contains only 6,000 entries, overfitting was inevitable. Simpler architectures are thus considered. However, with two few hidden layers and nodes in each hidden layer, our model may lose the ability to generalize. Moreover, 9 different network configurations were tested to determine the architecture that best balances over and under-fitting. The optimal network size was found to contain a single hidden layer of size 3 and the loss curve for this network configuration is shown in Figure 12.

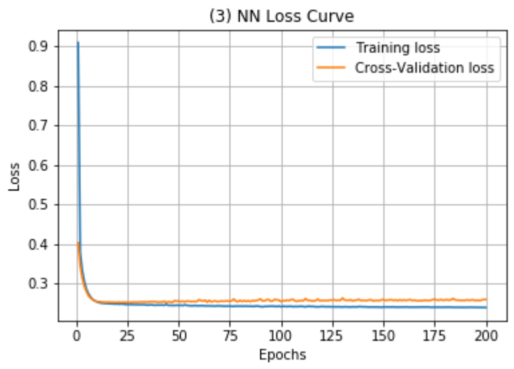

Figure 12

The loss curves show low variance as indicated by the proximity between the training and validation curves. With this reduced variance, however our model suffers from increased bias; the training loss is much larger than that observed from the default neural network. Moreover, to add complexity to our model and improve generalization ability, we consider adjusting alpha, the L2 regularization parameter. This parameter penalizes sparse models; that is models which contain many parameters whose values are close to zero. Moreover, larger values for alpha will encourage smaller neural network weights and simpler models while smaller values will encourage larger weights allowing for more complex decision boundaries. Thus, smaller values of alpha may improve bias at the expense of increased variance and overfitting. The validation curve for alpha are shown in Figure 13 . Surprisingly, decreasing alpha further increases bias of the model indicating more complex network representations fail to improve generalization ability. As expected with larger values of alpha, overfitting and variance are reduced.

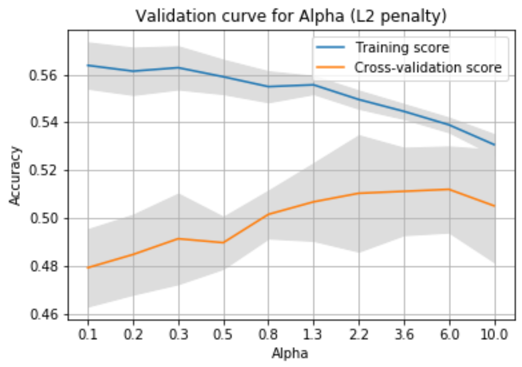

Figure 13

Many more parameters exist when defining a neural network that may help reduce bias. Random search was used to find the set of hyperparameters that maximizes validation accuracy. It was found that for a network with a single hidden layer of 3 nodes, the optimal batch size, alpha value and activation function were 32, 1.51 and sigmoid respectively. The learning and loss curves for a Neural Network with these parameters is shown in Figures 14 and 15. The model is significantly improved over the default classifier as overfitting is properly addressed. However, the model still suffers from high bias which appears to be a decreasing function of the training set size. This indicates that more data may further improve model performance.

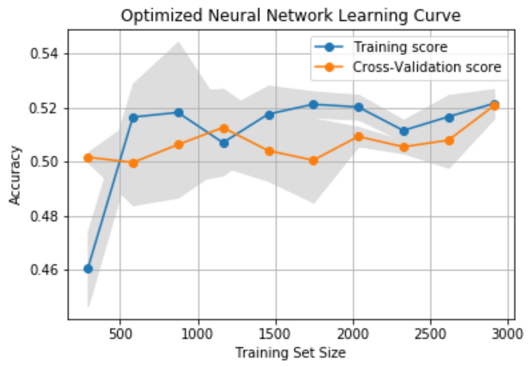

Figure 14

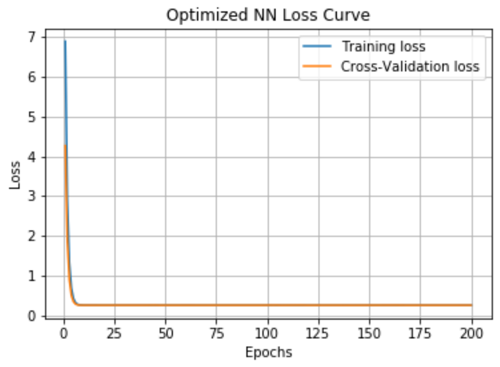

Figure 15

The trained model was evaluated using the test set and the confusion matrix for which is shown in Figure 16. From the confusion matrix we note that the model almost always predicts a team will cover the spread and is correct less than half the time on these predictions. A model that always (or almost always) predicts a team will cover the spread is uninteresting. It is clear the model is still overgeneralizing which is likely the result of the simplistic network configuration. Moreover, additional data is likely necessary as such will enable more complex network representations to be constructed allowing for more interesting models and perhaps reduce bias.

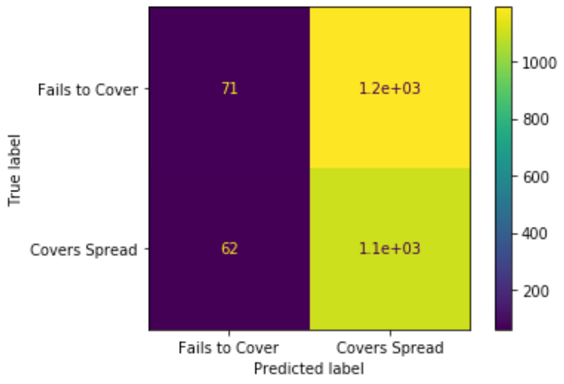

Figure 16

D. Support Vector Machines (SVM)

The default SVM classifier in sklearn uses the Radial Basis Function (RBF) kernel. This classifier was trained on the NFL Scores Dataset and the learning curve for which is shown in Figure 17. The default SVM classifier results in a model with high variance as indicated by large deviances between training and validation. The model also shows significant bias with an accuracy around 50% irrespective of the sample size.

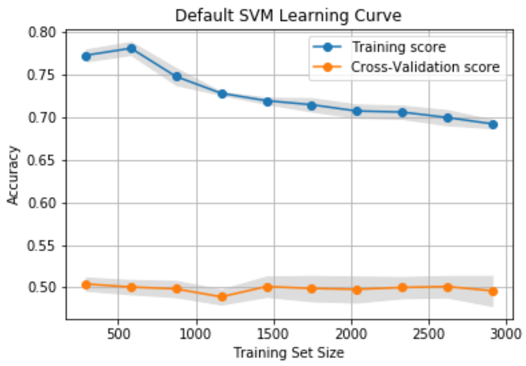

Figure 17

To tune an SVM, an optimal architecture that balances variance and bias needs to be identified. Given our dataset is complex with many features and we know little about how these features aid in prediction, we'll fit an SVM on all kernel function offered in sklearn. In particular, a linear kernel, a 3rd degree polynomial kernel, a sigmoid kernel and the default RBF kernel are considered. The training accuracy curves as a function of training iteration are shown in Figure 18. The kernel that most efficiently minimizes loss (maximizes accuracy) is the RBF kernel and is thus the kernel that will be used when optimizing the hyperparameters C and gamma to address overfitting and bias.

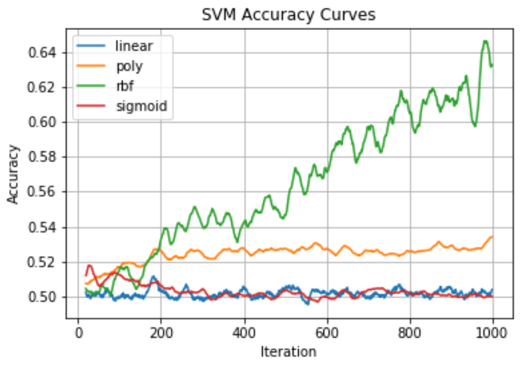

Figure 18

The regularization parameter (C) effectively allows adjustments to the size of the margin of the chosen hyperplane. There is an inherent tradeoff between the amount of misclassified training examples and size of the margin. Larger margins (smaller values for C) will result in more misclassification and less overfitting while the reverse is true for smaller margins (higher values for C). The validation curve for C is shown in Figure 19. As expected, smaller values for C resulted in much less overfitting and appear more optimal than larger values in the noisy dataset.

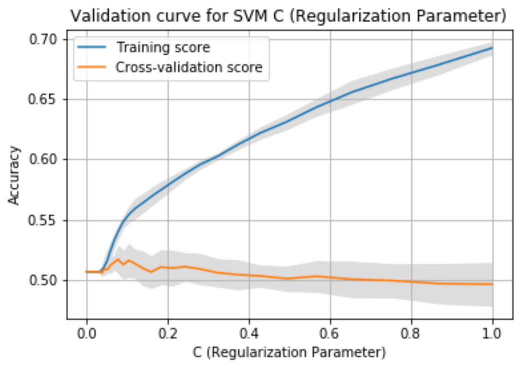

Figure 19

Another parameter which may reduce overfitting is gamma which allows us to adjust the weights more distant points have on the location of the decision boundary. With smaller values of gamma, examples far from the decision boundary have more influence on the location of the boundary. Such will result in a less complex model with more bias but less variance. When gamma is large, close values to the decision boundary carry more weight than more distant points exposing the model to more overfitting while potentially reducing bias. The validation curve for gamma is shown in Figure 20. With small values of gamma, overfitting is small as indicated by the proximity of the training and validation scores and bias is minimized. Thus, the validation curve for gamma also suggests less complex models are more appropriate which makes sense as complex models would be more prone to the outliers and noise present in the dataset.

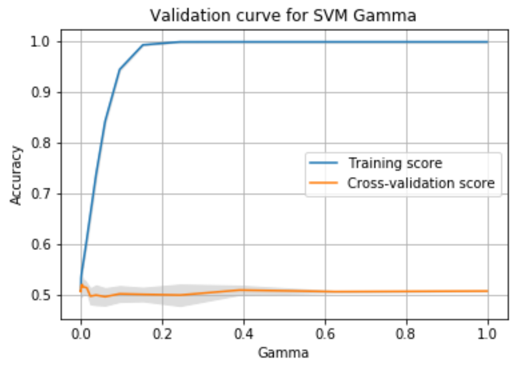

Figure 20

Finally, we use randomized search to tune C and gamma simultaneously. The optimal values for C and gamma were found to be 0.11 and 0.009 respectively. The learning curve for an SVM model using the RBF kernel and these parameters is shown in Figure 21. Overall, simplifying the model to reduce overfitting resulted in both lower variance and bias in comparisons to the default SVM classifier. In addition, the training and validation curves appear to both be increasing in parallel as the training set is increased, indicating that more training samples may improve model performance.

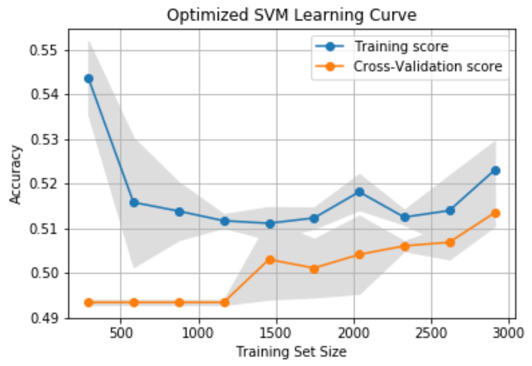

Figure 21

When classifying examples on the test set, the trained model correctly predicted 51.5% of games. The confusion matrix for the classifiers predictions is shown in Figure 22. From the confusion matrix we see the model predicts around 2/3 of the time that a team will cover the spread and is only correct 50% of the time on these predictions. However, when the model predicts a team will not cover the spread, it is correct over 56% of the time. Since a sports bettor only needs to predict with 52.4% accuracy to break even on Vegas spread bets, one potential strategy may be to bet on games in which the optimized SVM model predicts a team will not cover the spread. Since the model is correct on these bets 56% of the time, the strategy is theoretically profitable. However, more data is necessary to verify the low false negative rate of the model seen on the test set.

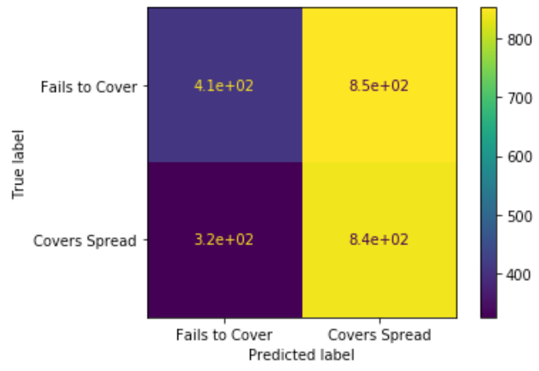

Figure 22

E. K-Nearest Neighbor (KNN)

The default KNN classifier in sklearn, which averages the 5 closest datapoints based on the Euclidean Distance, is fit to the training set of the NFL Scores Dataset and the learning curve for this classifier is shown in Figure 23. The default KNN classifier produces significant variance and bias that appear constant irrespective of training size.

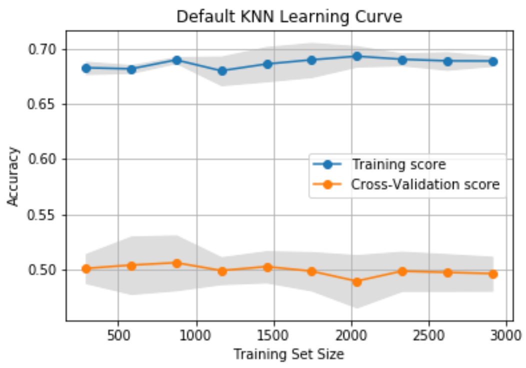

Figure 23

With 34 different features and a noisy database of only around 6,000 samples, the similarity between the 5 closest neighbors may be occurring spuriously. Thus, increasing the number of neighbors may allow for more averaging, less overfitting, and a more accurate model. To test this hypothesis, the validation curve for the number of neighbors is generated and shown in Figure 24. As can be seen from the validation curve, overfitting is inversely related to the number of neighbors. With the evaluation of a single neighbor, the model is perfectly fit to the training set; this results in a model with the most bias and variance. As the number of neighbors increase, the training and validation scores begin to converge, and variance is reduced. Other than a local maximum at around 12 neighbors, increasing neighbors appears to have little effect on validation accuracy.

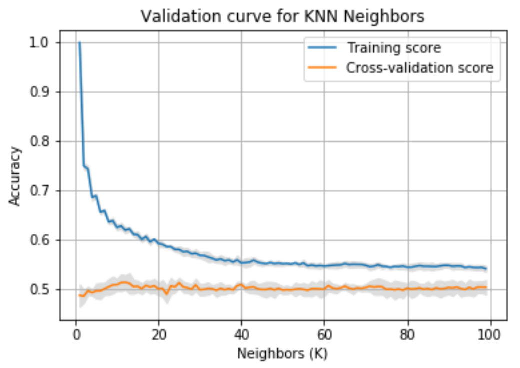

Figure 24

The high dimensionality and noise present in dataset likely make KNN a poor choice to predict NFL spreads; even with significant averaging, the model failed to impress. However, several other parameters exist when defining a KNN model which may improve performance. Grid search was used, and it was found that the optimal values for the number of neighbors, power parameter and weighting mechanism were 11, 3 and uniform weighting respectively. The learning curve for the optimal KNN classifier is shown in Figure 25. The optimized model still has high variance; however, bias appears to be improved significantly over the default KNN classifier. In addition, the validation accuracy appears to be increasing monotonically with the training set size indicating more data may further improve the accuracy of the model.

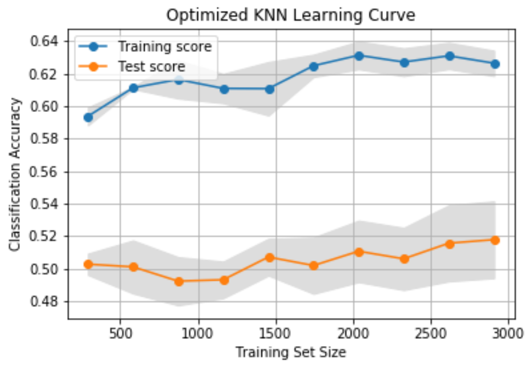

Figure 25

The optimized classifier was finally used on the test set resulting in an accuracy of 49.1%. The confusion matrix of these predictions is shown Figure 26. The model produces many more false positives than true positives or true negatives which is a testament to the high variance that still remains in the model. While additional data can reduce this variance, a significant amount of data is likely required due to the high dimensionality of the dataset.

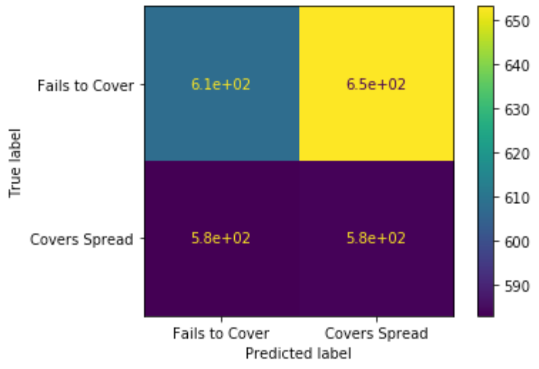

Figure 26

F. Model Comparisons

Figure 27 shows a comparison of the validation accuracy for the optimal models with respect to training size. As can be seen, all models seem to improve in terms of validation accuracy as the training size increases, indicating all models may perform better with more data. In addition, the accuracies of each model on the test set for Decision Tree, AdaBoost, Neural Network, SVM, and KNN were 51.1%, 49.3%, 48.4%, 51.5% and 49.1% respectively. Thus, the validation scores of each of these models was better than or equal to the test set accuracies indicating models may be overfitting and the training set may not be representative of the entire class of NFL game spreads.

Decision trees provided the most consistent results as both validation and test scores was 51.1%. This suggest that Decision trees may be most robust when the training set isn't representative of the underlying data generating process. On the contrary, Neural Networks had both the worst performance on the test set and the largest deviation between validation and test set accuracies indicating these models are least robust when trained on unrepresentative data. Neural Networks are thus likely better applied to larger datasets as with more data the probability that the training set is a representative sample of the true data generating process is higher.

AdaBoost had the second largest deviation between validation and test accuracy which also may be explained by the noise present in the training set. By giving more weight to misclassified outlier examples the algorithm may be overfitting to this noise which doesn't scale well when tested out of sample.

The SVM model resulted in the best test set accuracy and a validation accuracy that only slightly exceeded test set accuracy. Thus, these models appear appropriate for modeling high dimensional data with limited samples.

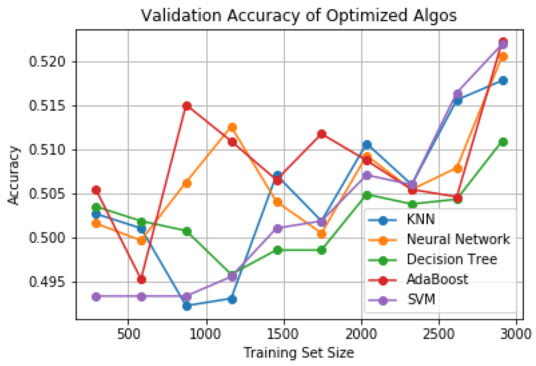

Figure 27

Figure 28 shows the training time of each model. As can be seen, the Neural Network classifier training time took significantly longer than any other algorithm even with a small dataset and a single hidden layer of 3 nodes. The SVM model also displayed much larger training times than Decision Trees, KNN, and AdaBoost and training time appears to be increasing at the fastest rate with respect to the training set. Moreover, with larger datasets, Neural Networks and SVM's may require significant training time constraints. Given the context of this problem, however, neither training nor testing time is of significant importance as Vegas oddsmakers quote game lines days in advance.

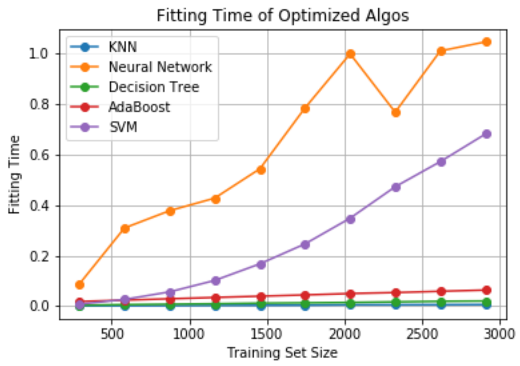

Figure 28

IV. Modeling with NFL Play-by-Play Dataset

A. Decision Trees

A Decision Tree with default hyperparameters was fit to the NFL Play-By-Play dataset; the learning curve for this classifier is shown in Figure 29. The large deviation between the training and validation curves indicates high variance in the model which is likely explained by overfitting. The validation score, as computed by the F1 score shows some bias and hovers around 0.70 and 0.71 regardless of the training size indicating additional data may not improve performance.

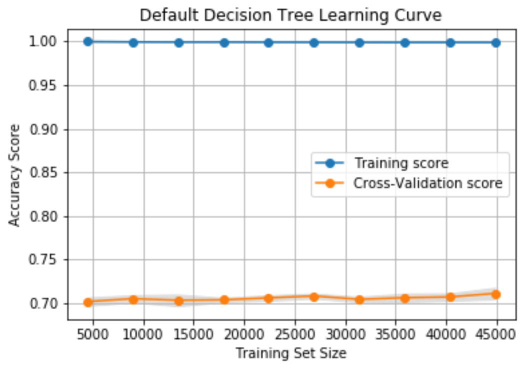

Figure 29

To reduce overfitting, we consider limitations on the maximum depth of the tree. The validation curve for the tree depth is shown in Figure 30. At lower depths both bias and variance is minimized. However, as the depth increases beyond 5, the training and validation accuracies begin to diverge, and the tree begins to overfit.

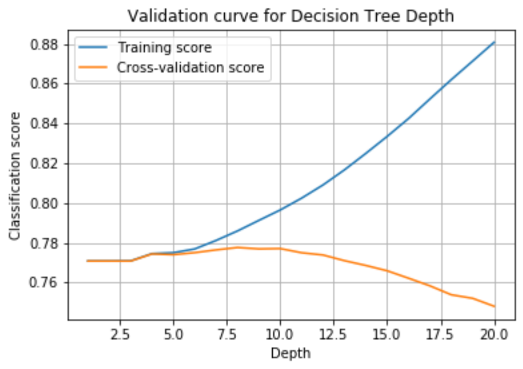

Figure 30

Given our dataset is unbalanced, we will also consider reducing overfitting by adjusting the minimum weighted fraction of weight's parameter. This parameter constraints the tree from producing a leaf node unless some minimum weighted percentage of total training examples are classified. This metric is weighted by the proportion of positive and negative examples in the dataset. Since the NFL Play-By-Play dataset contains roughly 40% run plays and 60% pass plays, this metric should result in a less biased pruning method. The validation curve for this parameter is shown in Figure 31. The training and validation scores are distant at small weights but as weights increase both variance and bias are reduced. When the parameter is 0 no pruning occurs and the decision tree is able to perfectly classify all the training examples.

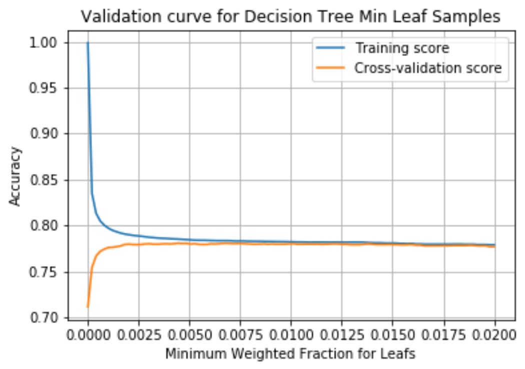

Figure 31

As was done on the previous dataset, we next train all hyperparameters simultaneously as to further improve validation performance and minimize variance in our model. The optimal depth, weighted fraction of leaves, minimum samples for a split and splitting criterion were 8, 0.0039, 2 and the gini criterion respectively. The learning curve for the optimized tree is shown in Figure 32. In comparisons to the default classifier, overfitting, variance and bias are all reduced significantly. Both the training and validation scores remain bounded as training examples are increased passed 20,000. Thus, the addition of more data will unlikely improve the model further.

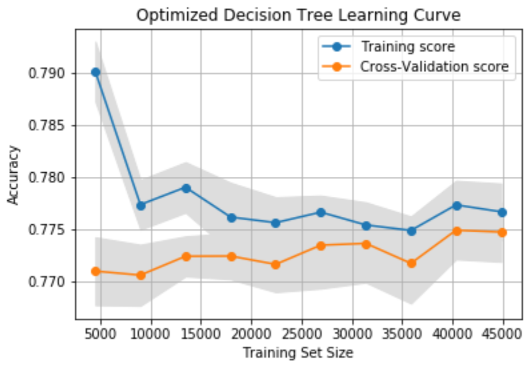

Figure 32

When used to classify examples in the test set, the optimized classifier produced an F1 score of 78% and the confusion matrix shown in Figure 33. The model predicts many more pass plays than run plays and as a consequence produces many false positives. Given our indifference between false positives and false negatives in the scope of this problem, this is not a big issue.

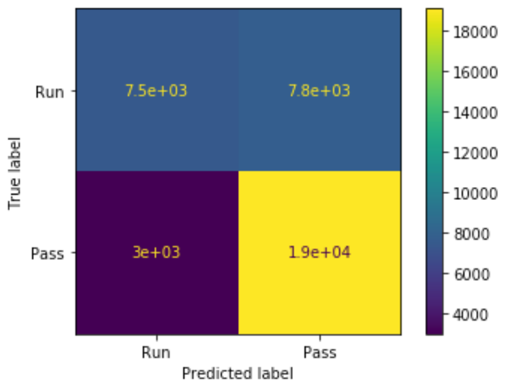

Figure 33

B. *AdaBoost*

We next consider the application of AdaBoost on the NFL Play-By-Play Dataset. The learning curve for the default AdaBoost classifier consisting of 50 decision tree stumps is shown in Figure 34. Both the learning curve and validation curves converge quickly after 10,000 samples indicating minimal variance and overfitting. This demonstrates an important property of AdaBoost; when the base learners in AdaBoost are weak learners that don't overfit the dataset, AdaBoost is robust to overfitting due to the averaging effect. Moreover, the decision tree stumps likely have these properties. In addition to low variance and overfitting, the model has less bias than the optimal decision tree classifier. Bias, however, doesn't appear to be reduced as the training set size exceeds 15,000, indicating more data may not improve performance.

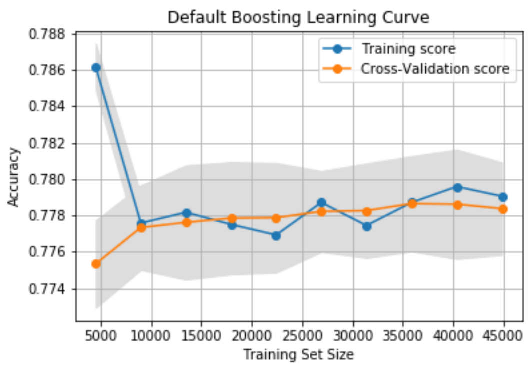

Figure 34

Since overfitting doesn't appear to be a huge issue, we'll experiment with adding complexity to the model by increasing the number of learners. With our dataset that contains few outliers we would expect model accuracy to improve with the number of learners without sacrificing much in terms of variance. The validation curve for the number of learners is shown in Figure 35. As we'd expect the validation accuracy of the model improves with the number of learners while variance and overfitting only appear to increase after 80 learners are added.

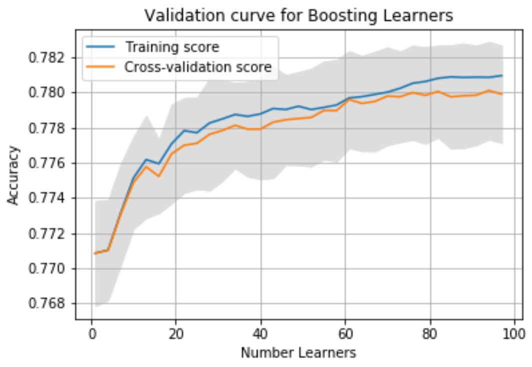

Figure 35

To add further complexity, we will modify the depth of the underlying decision trees. Figure 36 plots the validation curve for AdaBoost when modifying the depth of the decision trees used in the algorithm. As seen in the validation curve, AdaBoost begins to overfit the data as the depth of the underlying decision trees increases. This makes sense as if the underlying base learners in AdaBoost overfit the data, then AdaBoost may not provide immunization from overfitting. Thus, using decision trees with larger depth that overfit the training data will consequently result in AdaBoost overfitting to the training set.

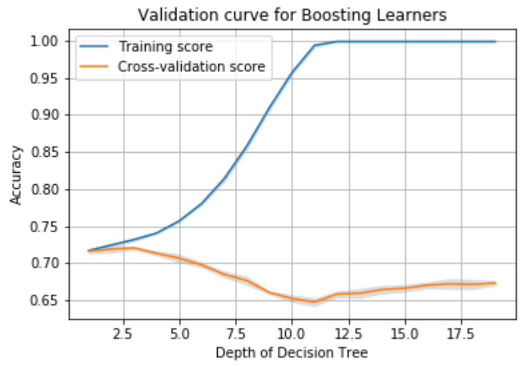

Figure 36

We next look to train all of the hyperparameters simultaneously as to further improve validation performance and minimize variance in our model. The optimal number of estimators and learning rate were found to be 95 and 0.95 respectively. The optimal depth, weighted fraction of leaves minimum samples for a split and splitting criterion for the base learners were 2, 0.001, 2 and the gini criterion respectively. The learning curve for the optimized model is shown in Figure 37 and displays less bias than the default classifier at the expense of slightly more variance. This is as expected since complexity was added to the model with the addition of more estimators and a deeper maximum tree depth. However, even with this increased variance, the optimized model seems to handle overfitting well as both the training and validation accuracy appear to be converging with increasing sample size.

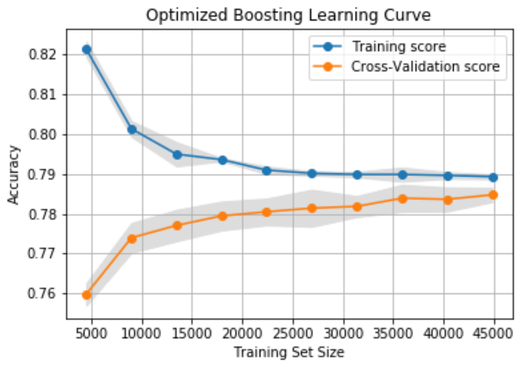

Figure 37

The optimized AdaBoost algorithm was used to classify examples on the test set, and the confusion matrix for the tree is shown in Figure 38. The algorithm performed quite well on the test set producing an F1 score of 78%, however like the decision tree the model produced many false positives. Given both models have produced many false positives, such may indicate that many of the outliers present in the data are run plays.

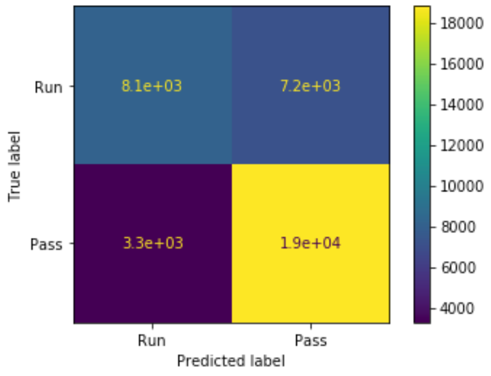

Figure 38

C. Neural Networks

The default sklearn Neural Network classifier with a single hidden layer of 100 neurons was fit to the NFL Play-By-Play training set; the learning and loss curves are shown in Figures 39 and 40 respectively. While the training and validation accuracies remain close regardless of the sample size in the learning curve, the two curves diverge with increasing epochs in the loss curve; the results from these charts provide ambiguous results as the learning curve indicates the model may be underfitting while the loss curve displays potential overfitting. Moreover, due to the conflicting messages from the learning and loss curves both simpler and more complex network representations are considered.

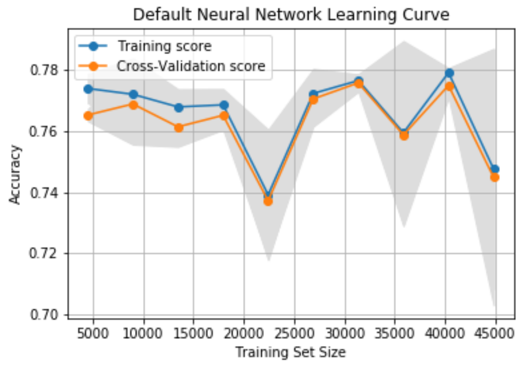

Figure 39

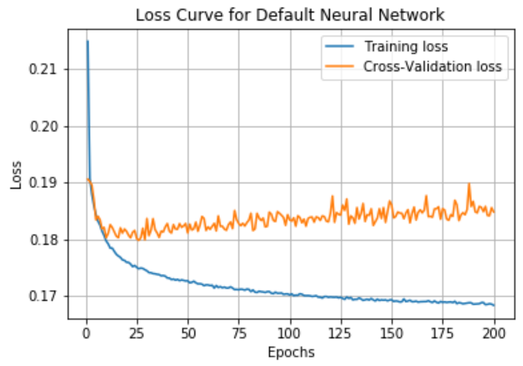

Figure 40

In total 5 different network architectures were considered, and the optimal configuration was found to be a simpler one with a single hidden layer of size 24. The simplicity of the optimal model is particularly surprising; a network of this size contains only 3,553 tunable parameters which is small given our dataset contains almost 100,000 entries. With such a large dataset our network should be able to handle more complex network representations, however this was not found to be the case. The loss curve for this neural neural network configuration is shown in Figure 41. Training and validation losses narrowed from the default classifier indicating overfitting was reduced. This is an expected consequence of a smaller, less complicated network.

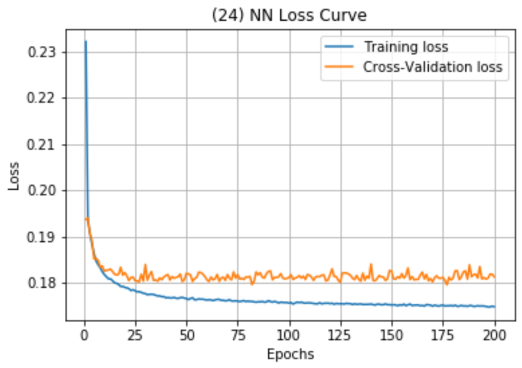

Figure 41

We next evaluate the impact of alpha on this neural network architecture. In sklearn, alpha corresponds to the L2 regularization term that penalizes sparse models where only a few parameters are non-zero. Thus, larger values of alpha will penalize more complex networks that generate more curved decision boundaries while smaller values of alpha will result in more complicated models that may improve accuracy at the expense of overfitting. The validation curve for alpha is shown in Figure 42 further indicating that as alpha is increased, simpler models are preferred thus reducing variance and overfitting. Added complexity however fails to have a significant impact on bias as the validation score remains relatively stable when alpha exceeds 0.25.

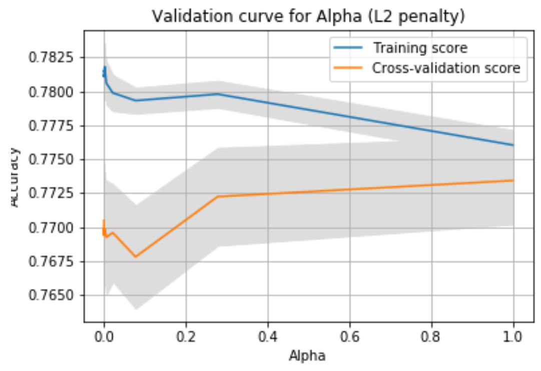

Figure 42

Using random search all hyperparameters were tuned simultaneously to maximize validation accuracy. The optimal parameters for the batch size, alpha, activation function and solver were 128, 0.198, sigmoid activation and Adams optimizer respectively. The learning curve and loss curves for a Neural Network with these parameters is shown in Figures 43 and 44. The model shows both less bias and variance than the default network as both training and validation loss are lower in absolute value and remain relatively close with each epoch. Given the network size was decreased and alpha was increased, simpler models appear to outperform more complex ones on this dataset.

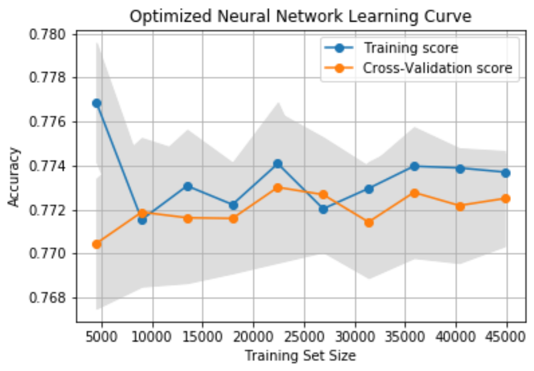

Figure 43

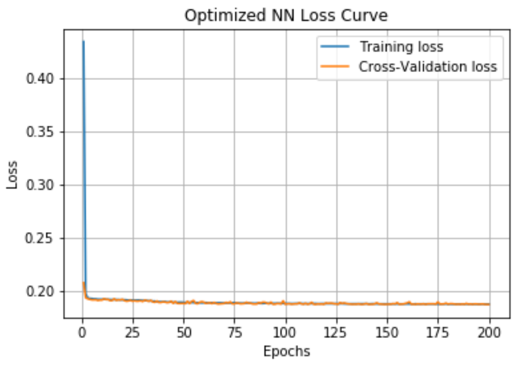

Figure 44

On the test set, the optimized Neural Network showed consistent results also obtaining an F1 score of 77%. Unlike the previous models, the Neural Network produces a similar number of false negatives and false positives.

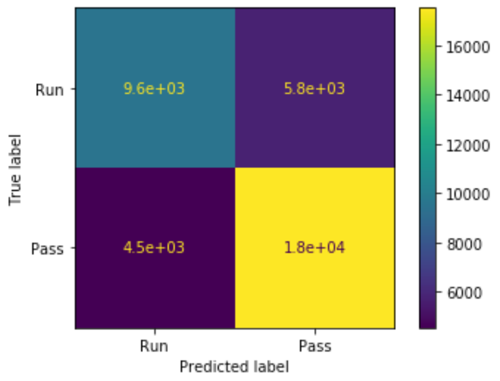

Figure 45

D. *Support Vector Machines*

An SVM model was fit on the NFL Play-By-Play Dataset with the default parameters in sklearn. The learning curve for the default model is shown in Figure 46. The learning curve shows relatively consistent results on training set sizes less than 35,000, however increasing the training set beyond this point results in reduced accuracy. Such is likely the consequence of the model failing to converge in the constrained number of iterations when the training set is incremented beyond this point.

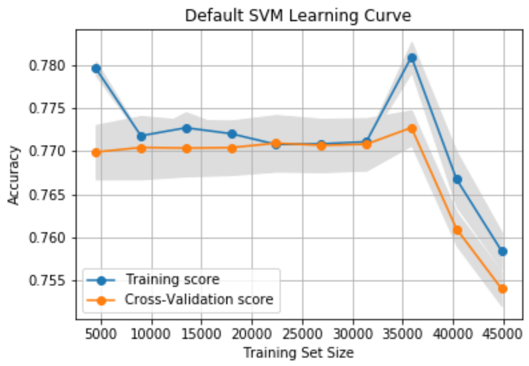

Figure 46

Prior to tuning hyperparameters, the appropriate kernel function was selected by evaluating the training accuracy curves as a function of training iteration. These curves are shown in Figure 47 and indicate the RBF kernel provides the quickest convergence and most efficiently maximizes accuracy. Moreover, this kernel will be the considered when optimizing our model.

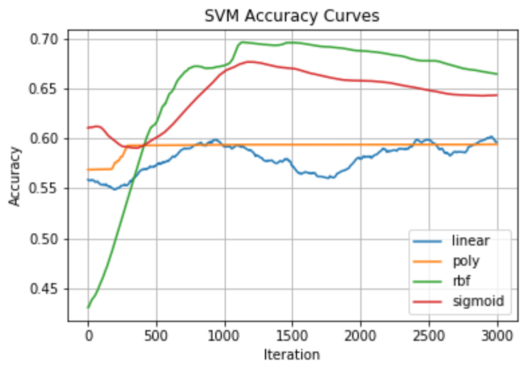

Figure 47

With our kernel selected we continue to specify C and gamma. Both C and gamma provide metrics for adjusting the complexity of the SVM. C allows adjustments to be made to the margin of the hyperplane and gamma determines the weights each support vector carries in determining the optimal hyperplane. Moreover, with higher values of C and smaller values of gamma, margins are smaller and support vectors closer to the dividing hyperplane carry more weight. Such results in a more complex model. In contrast smaller values of C and larger values of gamma results in larger margins and more distant support vectors carrying more weight. This allows for less complex model that may be less prone to overfitting but may not be complex enough to generalize well.

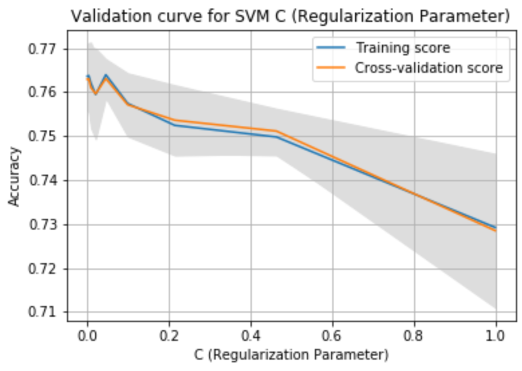

Figure 48

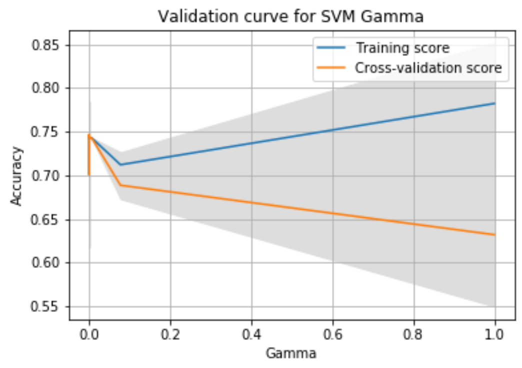

Figure 49

The validation curves for C and gamma are shown in Figures 48 and 49 respectively. Smaller values of both C and gamma appear to be preferred over larger values. These results contradict each other as smaller values of C would indicate a preference for less complex models while smaller values of gamma would be biased towards more sophisticated models. Given the low variance and overfitting seen in the default SVM classifier, our model can likely handle more complexity. Thus, the preference for smaller values of C shown in the validation curve may simply be a consequence of more complex models failing to converge within the set maximum number of iterations and thus producing lower validation accuracies.

To find an optimal balance between C and gamma, randomized search was used to tune both parameters simultaneously. The optimal values for C and gamma were found to be 0.11 and 0.009 respectively. The learning curve for the optimal SVM model is shown in Figure 50. The optimized model displays similar bias and variance as the default model for training set sizes below 35,000. However, the model is more robust to larger training set sizes. Even so, however, additional data will not likely improve performance of the optimized model since validation accuracies appear to remain constant.

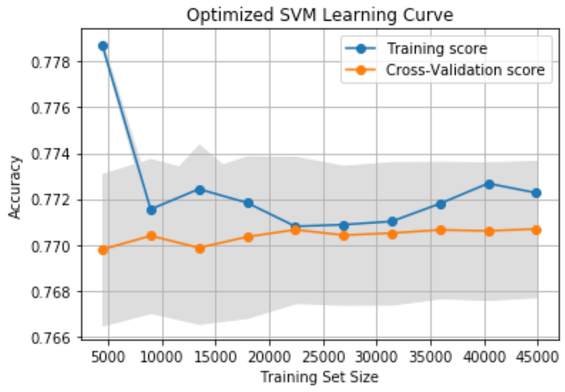

Figure 50

Applying this optimal SVM model on the test set resulted in an accuracy score of 77% and the confusing matrix shown in Figure 51. While the F1 score is comparable to other models, the overall accuracy score is much lower (only around 60%). The model's performance was likely hindered due to both the complexity and size of the dataset. Such resulted in the SVM failing to converge and thus suboptimal models. Training on a subset of the training set or applying feature reduction techniques may be beneficial to improve the performance of the model.

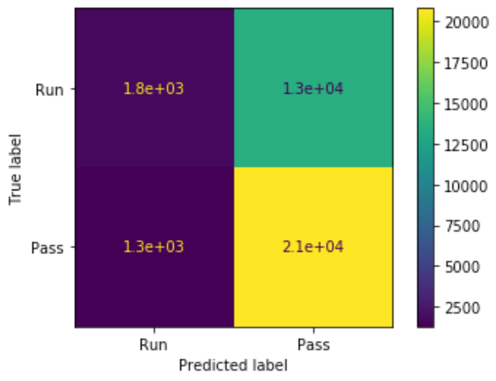

Figure 51

E. K-Nearest Neighbor

The default KNN classifier which uses Euclidean Distance and averages over 5 neighbors was fit to the second dataset; the learning curve for which is shown in Figure 52. The default classifier results in high variance, bias and overfitting. Given the high dimensionality of the dataset with 48 features and its size of 100,000 entries, such results aren't surprising. Since the model only considers the 5 closest neighbors – or 0.0005% (5/100,000 * 100) of the dataset – to make predictions, the model is particularly prone to outliers and irrelevant features.

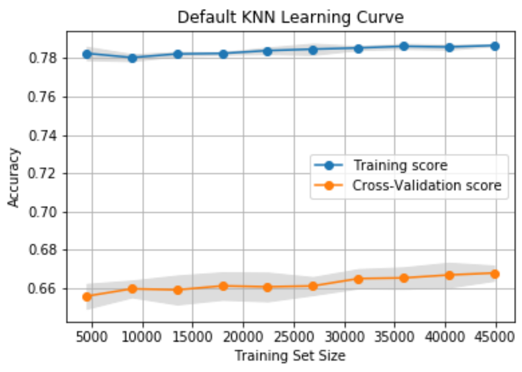

Figure 52

Allowing our KNN model to consider more learners may be beneficial as to reduce overfitting and improve validation accuracy. We test this hypothesis by plotting the validation curve in Figure 53. The validation accuracy improves while variance reduced with the addition of more neighbors. Beyond 300 neighbors, however, reduction to variance and bias are marginal.

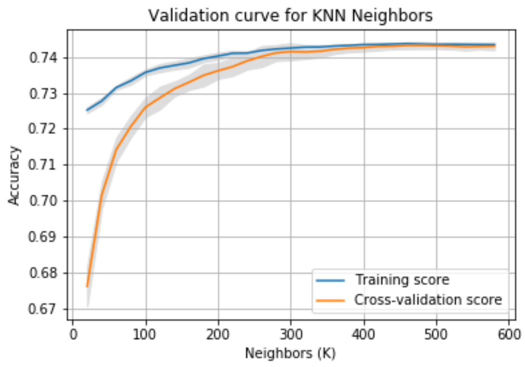

Figure 53

The accuracy of KNN relies heavily not only on the number of neighbors but also the weighting method used for the neighbors and the metric used to compute distance. Thus, randomized search was used to train all of these hyperparameters simultaneously. The optimal number of neighbors weighting metric and distance metric were found to be 296, uniform weighting and Manhattan distance respectively. The learning curve for this model is shown in Figure 54. The optimized model significantly reduced bias, variance and overfitting over the default model. Further, as training size increases, validation score seems to increase monotonically indicating that more data will further improve model performance.

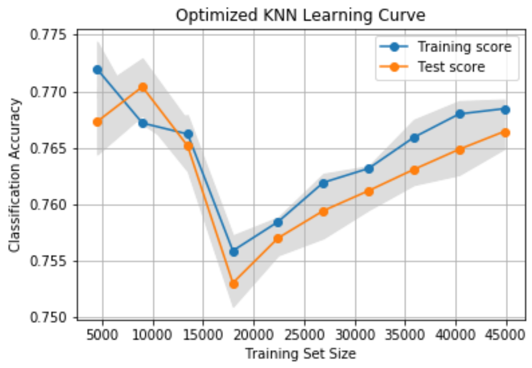

Figure 54

Predictions were made using the optimized KNN model on the test dataset which resulted in an accuracy score of 77%. The confusion matrix for the model is shown in Figure 55. The model performed surprisingly well out of sample considering the high dimensionality of the dataset. The large size of the dataset likely countered the dimensionality issues as more data allows for more neighbors to be averaged thus reducing the effects of outliers and noise.

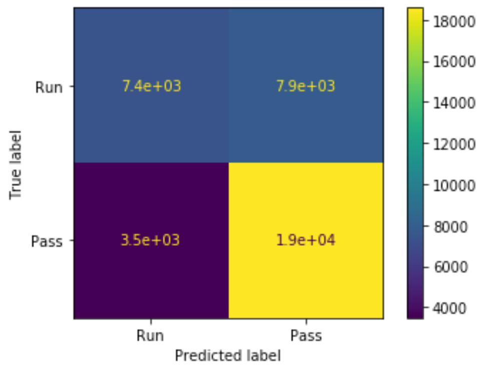

Figure 55

F. Model Comparisons

Figure 56 shows a comparison of the validation accuracy for the optimal models with respect to training size. The validation accuracy for both KNN appears to be increasing with respect to training set size suggesting that when a dataset doesn't contain much noise more comparable neighbors can be identified with larger datasets and thus model performance can be improved. AdaBoost also seems to perform better with increasing training set sizes which may be explained by more accurate weak learners on larger datasets.

When evaluated using the test set F1 scores for the Decision Tree, AdaBoost, Neural Network, SVM and KNN classifiers were 78%, 78%, 77%, 60% and 77% respectively. Moreover, aside from the SVM, all models produced similar test scores to their validation scores indicating the data in the training set was representative of the data in the test set. The SVM model's poor performance is likely a result of the model not converging on an optimal solution. As more training data is fed to an SVM, the optimization becomes increasingly difficult and convergence requires many more optimization iterations.

Decision Trees, AdaBoost and Neural Networks all had test accuracies that were consistent with their validation accuracies and all appear to be well equipped to handle large high dimensional datasets that don't contain many outliers.

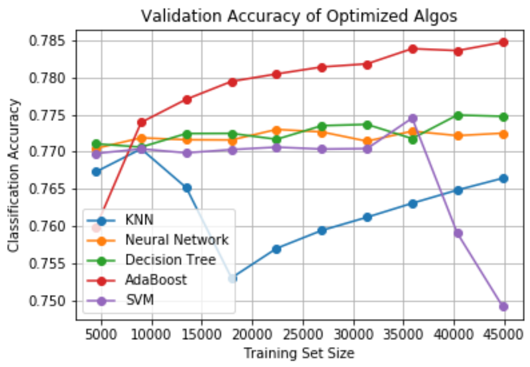

Figure 56

Figure 57 shows the training and Figure 58 the testing time of each model. The training time of the SVM explode with increased training set sizes, highlighting the fact that the optimization steps required become increasingly complex as the amount of training data is increased. Moreover, when training time is of significant importance and training sets are large, SVM's may not be appropriate model selections.

Testing times for KNN far exceed those seen in other algorithms and grows linearly with the amount of training data. Such occurs because computing differences between datapoints is done at testing time; and with 100,000 entries many distance computations need to be performed at runtime. Moreover, when testing time is of significant importance and datasets are large, KNN may not be an appropriate algorithm.

With the goal of predicting whether an NFL play is a pass or a run, a model can be trained in advance and thus train time is not a constraint. However, for the model to be decision useful, a prediction has to be made before a team runs a play; thus, query time is extremely important. Thus, KNN is likely impractical for the given problem. The performance of Decision Trees, AdaBoost and Neural Networks were all similar in terms of accuracies and query time. Any one or a combination of all these models are appropriate to predict NFL plays.

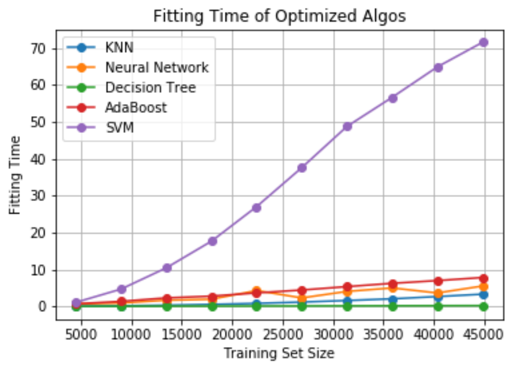

Figure 57

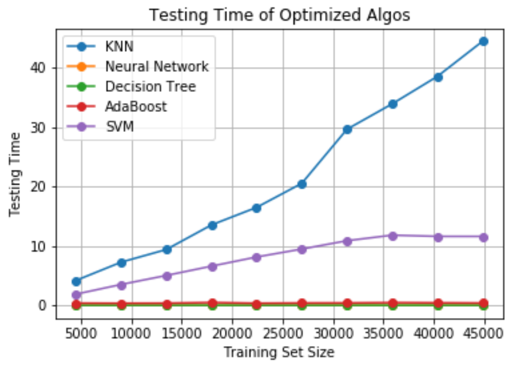

Figure 58

V. Conclusion

Based on the performance of each classifier on the two datasets, it's clear that different models may be preferred in different situations. The appropriate model to choose is highly dependent on the size and amount of noise in the dataset. Training and testing constraints may also be significant factors in the model selection process. When modeling data with many outliers, all classifiers perform relatively poorly, however KNN and AdaBoost do a particularly poor job modeling noisy data. From our experiments, Decision Trees are most robust to noisy data however still provide poor performance. Moreover, when working with a noisy dataset, the best solution may be to collect more data.

When few outliers exist, each classifier performs significantly better. In large datasets with relatively little noise, KNN performs better, but testing time can be slow. Both training and testing of an SVM can be slow on large datasets. In addition, training poses significant risk that the model doesn't converge leading to inconsistent accuracy scores between validation and test sets. Thus, additional precaution is necessary when training an SVM with large amounts of data. Decision Trees, AdaBoost and Neural Networks all perform well on datasets with these characteristics and can be trained and queried much more efficiently.

References

[1] Crabtree, T. (2020). NFL Scores and Betting Data. Retrieved from https://www.kaggle.com/tobycrabtree/nfl-scores-and-betting-data

[2] Horowitz, M. (2018. Detailed NFL Play-by-Play Data 2009-2018. Retrieved from https://www.kaggle.com/maxhorowitz/nflplaybyplay2009to2016

[3] Pedregosa, F (2011) Scikit-learn: Machine Learning in Python, pp. 2825-2830

[4] Chollet, F. (2015) keras, GitHub. https://github.com/fchollet/keras

[5] Sarle, W. (2001, May 21). AI FAQ/neural Nets Index. Retrieved September 20, 2020, from http://www.faqs.org/faqs/ai-faq/neural-nets/