Author: Michael Pecorino

Introduction

Sports analytics is becoming increasingly interesting for two main reasons. The first has to do with technology. Professional sport leagues are consistently researching ways to bring technology into the game that will capture data in new and exciting ways. For example, in the sport of baseball, Major League Baseball (MLB) is using the Hawk-Eye camera system to track ball and player movement. In the most recent upgrade, the system can even track player's limbs which can allow for people to get answers to questions they didn't even know they had. The National Basketball Association (NBA) uses similar motion-capture technology called STATS. The second driver behind sports analytics is the increase in the legalization of sportsbooks, which are companies that allow people to legally place bets on sporting events within certain states. The sports betting industry is expected to be an $8 billion-dollar industry by 2025. That figure will only grow as sports betting inevitably becomes legal in all 50 states.

For this project, I am focusing on two sports analytics classification tasks:

- Division I NCAA Men's College Basketball (NCAAMBB) school win/loss classification
- National Basketball Association (NBA) team win/loss classification

The two datasets are similar, and I believe this is advantageous to the analysis since being able to directly compare models across the two datasets will provide further insight into the two classification problems. What we will see in this analysis is that the NBA dataset is somewhat harder to classify than the NCAAMB dataset, or at least requires more sophisticated feature engineering in order to get the two problems on an even playing field. This fact will be made evident in the charts and analysis.

Given the recent advancements in technology, these two datasets have a seemingly limitless number of variables that can be engineered for the models. Additional feature engineering may advance the model metrics displayed in this analysis.

Data

Data for both classification problems were taken from Sports Reference: https://www.sports-reference.com/. The table below shows the dimensions of each dataset:

Data	Years	Observations	Variables	Response
NCAA Men's College basketball (NCAAMCB)	2011-2020	51,347	41	Win/Loss
National Basketball Association (NBA)	2013-2020*	7,379	41	Win/Loss

*2015 not available

There are significantly less observations per year in the NBA dataset due to that league having fewer teams. Division I NCAA Men's basketball has 357 schools that each play approximately 25-40 games per year. On the other hand, the NBA only has 30 teams that each playing 82 games per year.

The number of variables collected and features engineered are identical for both datasets, except for two variables. The NBA dataset contains the age of the players while the NCAA dataset contains an indicator for games played on neutral courts. The consistency in the remainder of the variables is to ensure apples-to-apples comparisons throughout the analysis – if the feature sets were vastly different, it would be difficult to determine which classification problem was more difficult. In addition to comparing the classification metrics across the two datasets, the variable similarities allow one to explore the variable importance for each problem and draw conclusions about the differences in styles of play in the two basketball leagues.

Modeling Introduction

The following types of models will be reviewed and compared throughout the analysis:

- Decision tree
- Neural Network
- Support Vector Machine (SVM)
- K-Nearest Neighbors (KNN)
- Boosting, using the XGBoost framework

Model Validation

It is important to discuss the strategy for model validation before model results, because the same validation strategy applies to each model discussed in this analysis. Before going further, it is worth mentioning the point of performing model validation: to prevent overfitting. An overfit model is one that is fit extremely well to a single dataset and cannot generalize well to new data - and performing well on new data is the goal. Using model validation methods during the model fitting stage is an important task to complete for any Machine Learning problem in order to have reasonable assurance that the model will perform well in the future.

A technique known as cross-validation is used to prevent overfitting and determine the optimal model parameters. Cross-validation consists of splitting the model training data into N subsets, also known as folds. Typical values for N include 1, 5, or 10. The cross-validation used in this analysis uses 5 subsets and can be referred to as 5-fold cross-validation.

The validation method is the following: fit a model all subsets except for one and evaluate the model on the subset not used for fitting the model. Below is a description of 5-fold cross-validation:

- Train on subsets [2, 3, 4, 5] and evaluate the model on subset [1]
- Train on subsets [1, 3, 4, 5] and evaluate the model on subset [2]
- Train on subsets [1, 2, 4, 5] and evaluate the model on subset [3]
- Train on subsets [1, 2, 3, 5] and evaluate the model on subset [4]
- Train on subsets [1, 2, 3, 4] and evaluate the model on subset [5]

Therefore, 5 versions of model metrics are produced, and a final cross-validation summary is generated by taking the average of all 5 results. The objective is to use the cross-validation results to find the optimal model hyperparameters, and by considering the average performance across the folds, we mitigate the risk of overfitting the model on any one set of data.

After the best model is chosen from cross-validation results, each model is tested on a validation dataset to get an unbiased estimate of how that model may perform on new data. This validation set will be used to select the best type of model, i.e. decision tree, KNN, Boosting, etc. Once the best type of model is selected, we should always test the model again on a totally independent dataset known as the test set. The model's performance on the test set is the expectation for how the model will perform in the future.

The table below expresses the number of observations in each type of data:

	Training Data (to optimize model hyperparameters)					**Choose best type of model**	**Get estimate of future performance**	
	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Validation Data	Test Data	Total observations
NCAAMCB	4,849	4,848	4,848	4,848	4,848	11,161	15,945	51,347
NBA	724	723	723	723	723	1,549	2,214	7,379

Model Descriptions

Decision Tree Model

Decision trees can model complex problems by asking questions about the data that break the hypothesis space into rectangular regions. In addition to cross-validation, for decision trees a process called pruning can aid with preventing overfitting. There are two types of pruning methods for decision trees: pre-pruning and post-pruning.

- Pre-pruning methods limit the growth of the tree structure. Such methods include, but not limited to, limiting the maximum depth of the tree or setting the number of observations required to make a split.

- Post-pruning is the more computationally expensive option out of the two, since it creates the largest possible tree and then trims the tree down to some optimal structure.

Post-pruning can provide slightly better predictive performance since leaves/nodes can be peeled away from the tree to optimize the performance, whereas a pre-pruning method may not necessarily grow to the optimal tree structure. Based on that key feature, post-pruning is used in this analysis.

Post-pruning is accomplished using the cost complexity parameter, or CCP, of decision trees. This parameter is also referred to as "alpha", and that is the terminology chosen hereafter. The larger the alpha, the more leaves and nodes will be pruned from the tree. Very small alpha would overfit, could fit the training data perfectly, and would not generalize well to unseen data. Very large alpha would underfit, be overly conservative, and would not fit either the training data or any unseen data very well.

Whether pre-pruning or post-pruning, the objective is to choose the alpha that optimizes the model performance on unseen data. This is accomplished by using cross-validation.

Neural Network

Neural Networks can represent hypotheses similar to a decision tree, but has the added ability of finding more complex decision boundaries relative to the rectangular regions formed by decision trees. Their framework is often compared to how neurons fire in the brain and how thoughts are generated or memory is accessed. Signal about the Machine Learning problem's response variable propagates forward and backward through the Network in a loop. Neural Networks are complex and usually unexplainable, but perform very well on hard tasks like object and speech recognition. When it comes to more basic regression or classification problems, Neural Networks can still perform about well as the next best model.

Support Vector Machine (SVM)

The objective of the SVM algorithm is to find the best decision boundary that maximizes the distance between the positive and negative samples. This line is often much more complex than the typical algorithm used in linear or logistic regression. For that reason, SVM's are on the longer side in terms of training time and scoring time.

K-Nearest Neighbors (KNN)

The Nearest Neighbors algorithm does exactly as the name implies – it finds the nearest neighbors to the input data and uses those nearest neighbors to inform the prediction. Taking this a step further, the model can consider 100% of the data as neighbors, and weight each neighbor's contribution to the prediction based on their similarity to the input data. The similarity-weighted approach is the one taken here.

This KNN algorithm is referred to as instance-based learning, or a lazy learner, which means a model of the input data is only created when a prediction is required. This means that training a model in the traditional sense, like for decision trees, is non-existent. It is as easy as storing the training data to be used later for prediction. On the other hand, this algorithm can be quite computationally expensive for prediction since the algorithm would have to search through the entire dataset to find the N nearest neighbors to the input data.

For some Machine Learning problems, this type of model can be quite powerful because the modeling framework can be customized to the input data. This is different from other models in this analysis because the others train a single model on the historical data and that model is used on all new data inputs. The trained model cannot adjust itself to

the input data. However, this model is very sensitive to the features used to calculate similarity. It is important to perform necessary steps like scaling and dimensionality reduction.

Boosting (XGBoost)

Boosting is an extension of the single decision tree in which multiple decision trees are linked together in what is referred to as an ensemble model. Effectively the process is that a decision tree is fit, the model makes errors, and then a subsequent tree model is fit that attempts to fix the errors of the previous tree. This process continues for some optimal number of trees as determined by cross-validation.

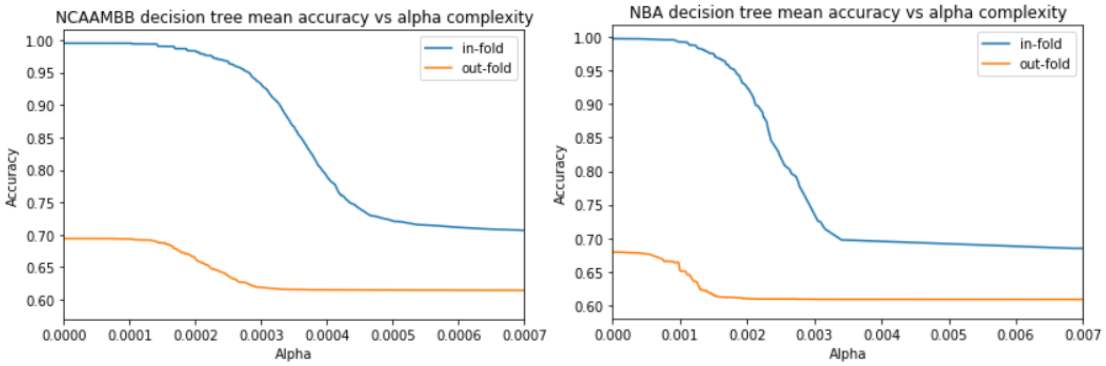
Using trees in an ensemble is often better because it reduces the bias of a single tree. Despite using cross-validation to get the optimal decision tree, the fact that the result is a single tree limits the modeling to a single perspective. Boosting, on the other hand, introduces multiple perspectives, which overall can perform much better on new data compared to the single perspective.

Learning Curves

We can plot what is referred to as a learning curve that shows how model performance changes over some variable, i.e. model hyperparameters, size of the training data, etc. For typical classification problems, and for some hyperparameters, the typical learning curve is a monotonically increasing performance on training data, with a U-shaped curve for validation data. The U shape on the validation data comes from the fact that some model settings will underfit while others will overfit, and that there is some point in the middle that performs optimally on new data.

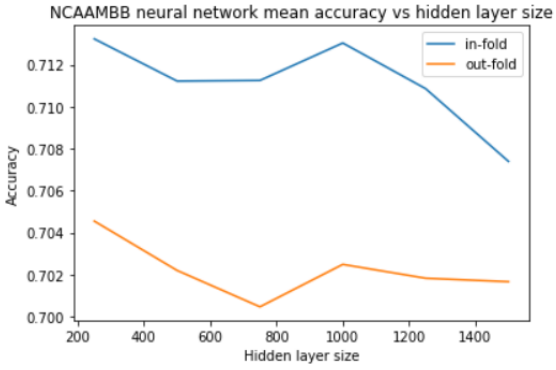
Decision Trees

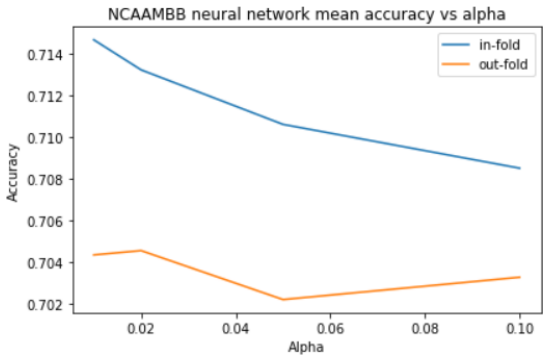
The plots below show the mean accuracy from cross-validation for the complexity parameter alpha. The decision tree algorithm performs optimally for both datasets at the lowest complexities. As alpha increases, more pruning is applied to the tree. The extra pruning makes the tree too simple and it underfits the data, and this is visible through the performance drop. A simple tree would perform poorly on both the in-fold and out-fold sets. The fact that the best out-fold performance is at the lowest alpha speaks to the complexity of the data. Even when the training data is completely overfit with 100% accuracy, the tree would still perform optimally on new data. This implies that the problem of classifying wins/losses in basketball is very difficult and requires complicated splits in the data to make accurate classifications.

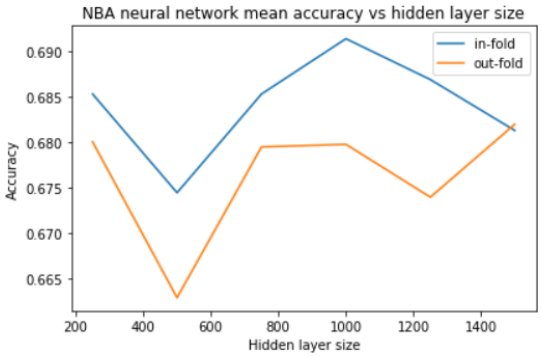


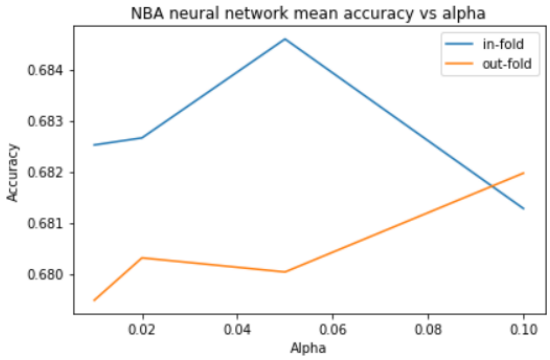
Neural Network

For Neural Networks, learning curve is plotted for some of the hyperparameters, like hidden layer size and alpha regularization. For each hyperparameter, the learning curves are quite noisy. There is no obvious signal about optimality from the shapes of the learning curves, and in every plot except for the hidden layer plot for the NBA dataset, the overall change in the magnitude of the accuracy is immaterial.



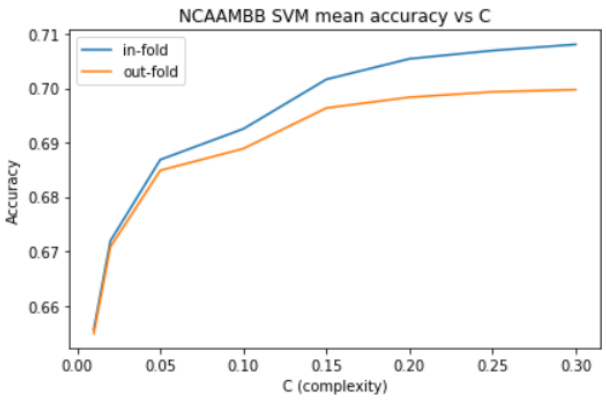


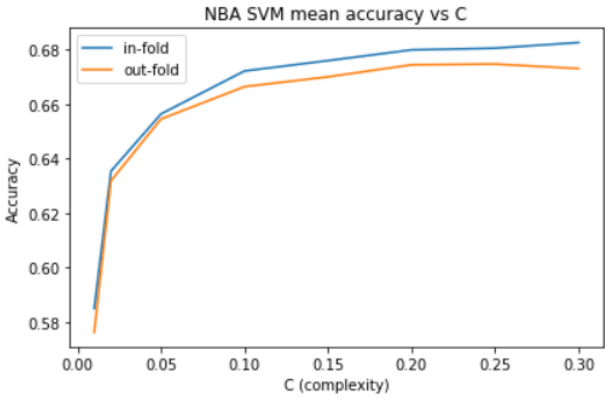




SVM

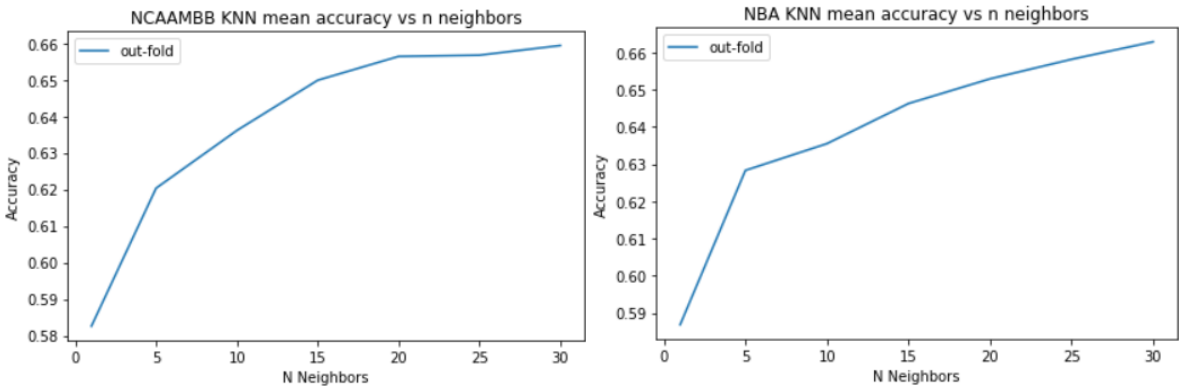
SVM is all about fitting a line to separate the data. If not constrained, the model can fit the training data perfectly and would generalize very poorly on new data. For SVM the complexity of the decision boundary is interesting to plot for a learning curve. It is obvious that as the complexity increases, the performance on the validation data begins to plateau, and would eventually decrease as the performance on the training data increases. Based on the plots, we can likely choose complexity values of 0.20 for both datasets, as complexity values beyond that are not providing much more model power.





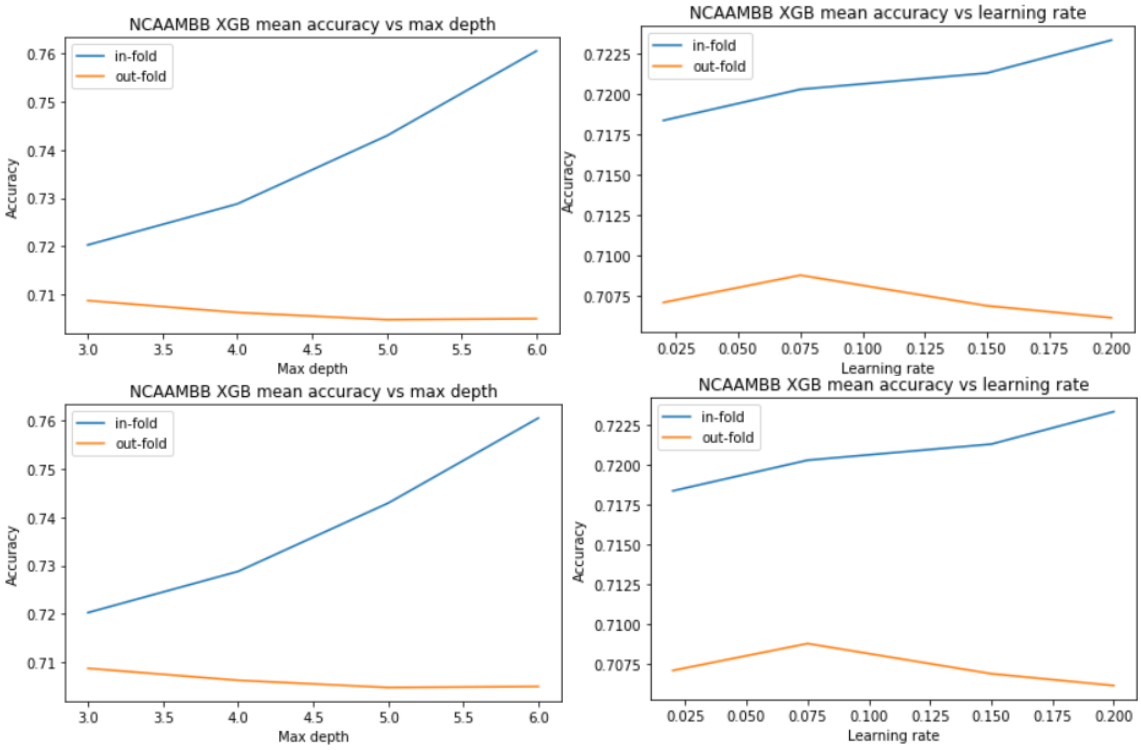
KNN

For the KNN algorithm, it makes sense to plot the number of neighbors on the learning curve. The plots below show the average model performance on the out-fold data for neighbors from 1-30, in steps of 5. The neighbors plot is often referred to as the scree plot, also known as the elbow plot, because the shape is like an arm and an elbow and typically, the optimal number of neighbors is around the elbow. For the NCAMBB dataset, the performance on the out-fold dataset begins to level off after 20 neighbors. It is very likely that choosing 20 would be optimal when considering the extra computation time needed for 30+ neighbors. For the NBA dataset, the performance on the validation data is still rising, and thus, we should test values beyond 30 to ensure an optimal model.

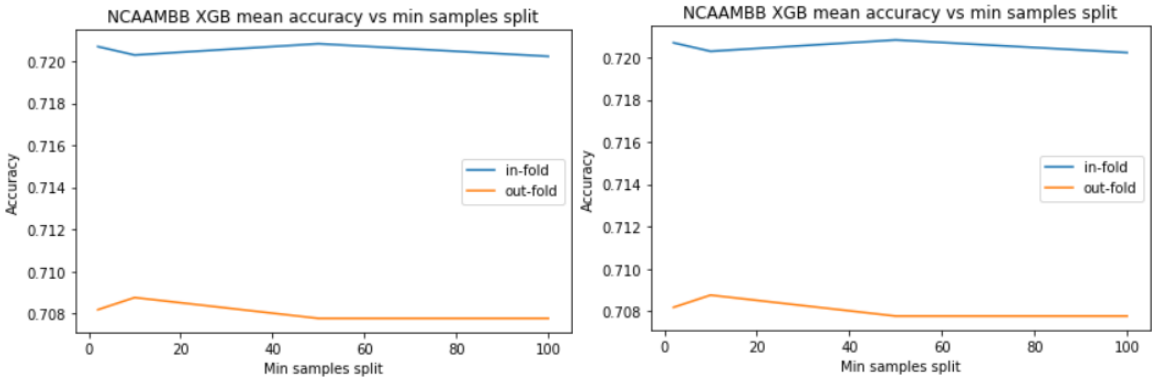


Boosting

The model output for the XGBoost model is the most interpretable as it exhibits the typical curves for classification problem. This speaks to the power of the XGBoost model, in that it can take an obviously complicated dataset and find more signal about the response variables compared to the other algorithms. The plots below include the learning curves for the max depth and the learning rate. For both plots, the out-fold data exhibits the typical U-shaped pattern of increasing performance followed by decreasing performance. It's also shown that the in-fold data is monotonically increasing.



The minimum samples required to make a split is also interesting to plot on a learning curve. As the minimum samples increases, we expect to see for the in-fold curve to decrease. The reason is because the model would become increasingly constrained and not make splits that would improve performance, so performance would trend down as the hyperparmater increases. We start to see that as the hyperparameter approaches 1000, but not enough values were tested to make the trend obvious. We also faintly see the U shape on the out-fold data similar to the other hyperparmaters tested.



Overall Model Performance

Cross-validation was used on the training data to optimize model hyperparameters. The validation data is used to choose the best model type i.e. Neural Network or Boosting. The test data is used to get an unbiased estimate of how the final model will perform on new data. The tables below show the model performances on validation and test data. The performance on the training data is included for reference.

For the NCAAMBB dataset, Boosting had the best performance on the validation data and so was chosen as the best model. The accuracy on the unbiased test data was 72.23%

Table 3: Model performance for training, validation, and test data for the NCAAMBB dataset

Dataset	Decision Tree	Neural Network	SVM	KNN	Boosting
Training	70.03	70.41	69.30	-	71.58
Validation	72.09	72.62	64.48	69.00	**73.66**
Test	71.05	71.50	63.89	68.50	**72.23**

For the NBA dataset, Neural Network

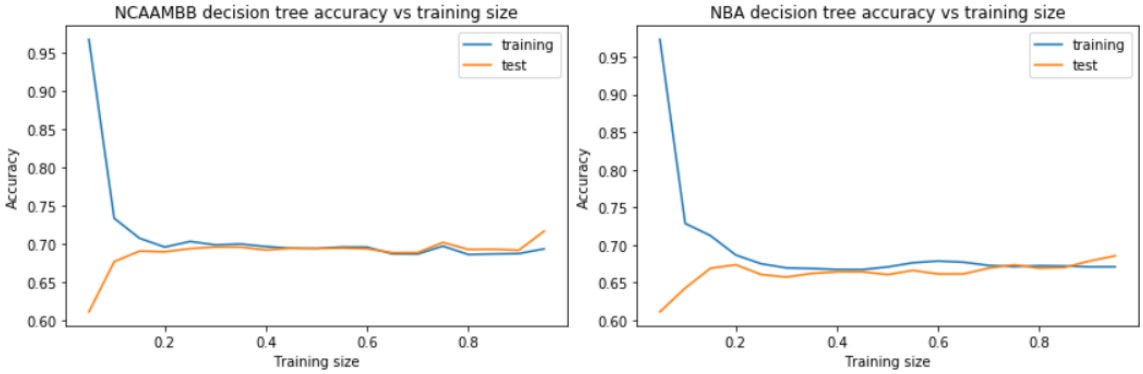
Table 4: Model performance for training, validation, and test data for the NBA dataset

Dataset	Decision Tree	Neural Network	SVM	KNN	Boosting
Training	68.03	68.20	67.34	-	71.52
Validation	66.37	**68.30**	67.72	65.33	66.17
Test	65.99	**67.34**	65.94	65.18	68.25

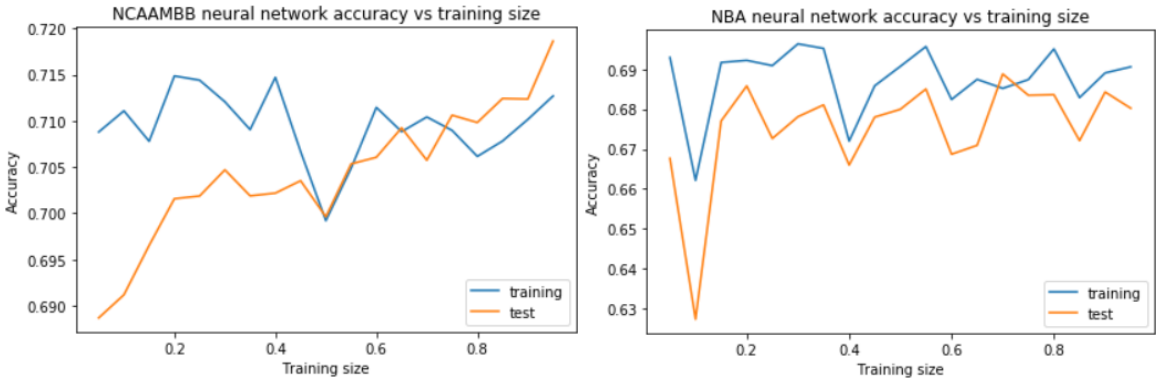
Model Performance by Training Size

We should check how each model performs as a function of how much data is used to train the model. The reason this is important is because certain models may perform better after more data is collected. For this experiment, 100% of the data was used and the data was split into training and testing data in 5% increments, with a minimum of 5% of the data and a maximum of 95% of the data.

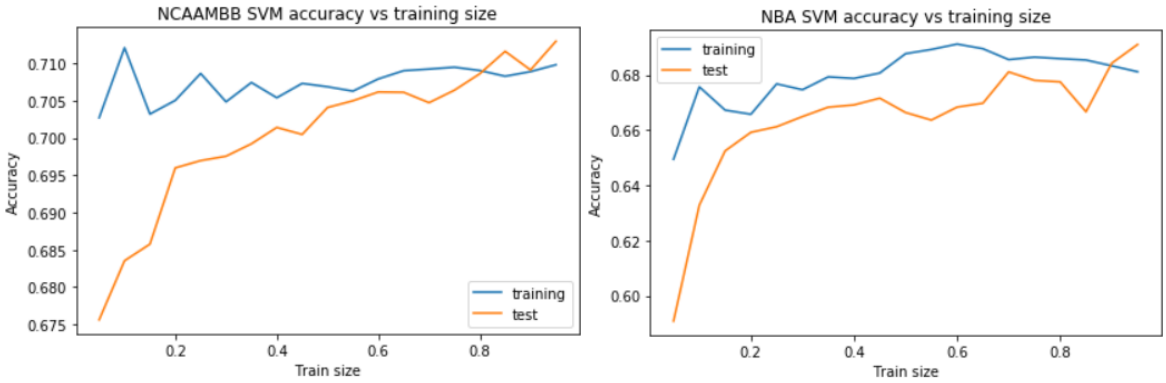
The decision tree model does not require a lot of data to reach optimal performance on unseen data. The plateau is reached very quickly. We should not expect the decision tree to get better as more data is added.



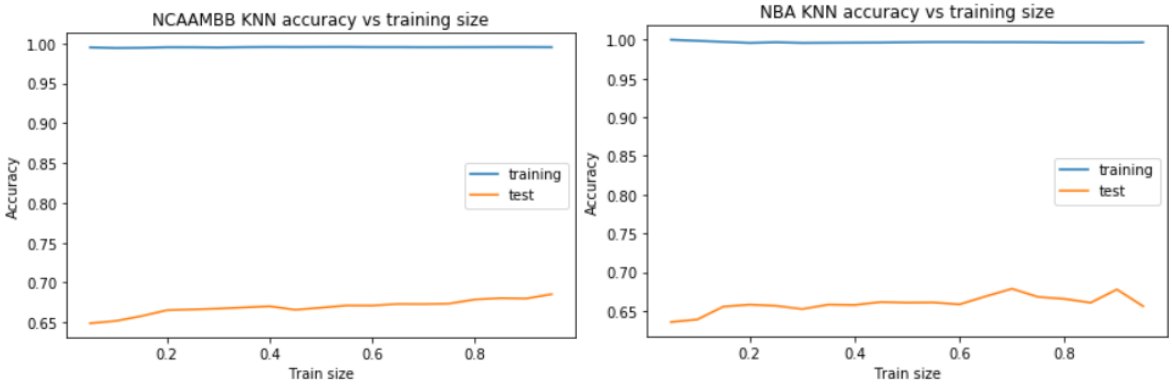
The Neural Network model on the NCAAMBB dataset shows an increasing trend on the test data as more data is used to train the model, whereas the NBA dataset is mostly flat after about 20% of the data is used for training. This is interesting because the Boosting model was chosen for the NCAAMBB dataset. This decision should be monitored as more data is added.



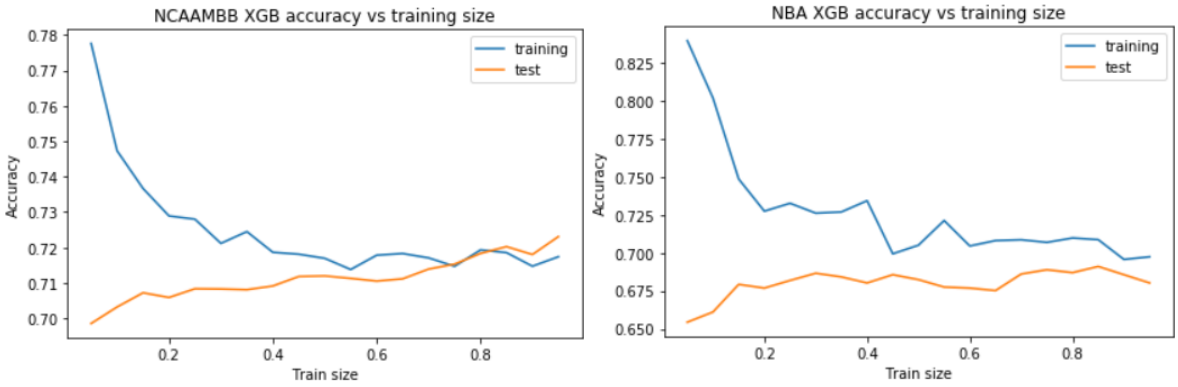
Both datasets show a positive trend in the SVM model as more data is added to the training set. We should expect SVM to perform better in the future when more data is available.



The KNN training score is not really relevant here as KNN training is not really comparable to the other models. The test data score has a steady increase as more data is added to the training data. These curves are very dependent on the numbe rof neighbors used (30). The curves would be steeper, or flatter, if a different number of neighbors was used.



The XGBoost model trends upwards as more data is added to the training set. This speaks to the power of the model. As more data is added, the boosting mechanism can continue to add model power.



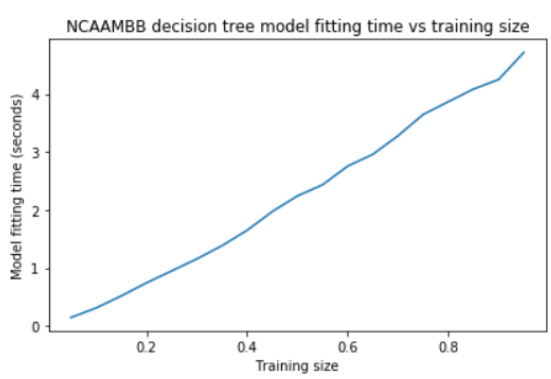
Training and Scoring Times by Training Size

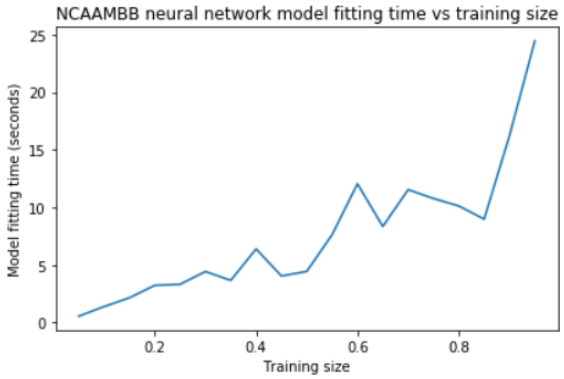
Considering just how a model performs on new data, while a significant factor, is not the only consideration for building a Machine Learning model. Modelers are also interested in how much time is required to train a model and how much time is required to score the model on new data. For simplicity, only the curves for the NCAAMBB dataset are included. The same insights can be applied to the models for the NBA dataset.

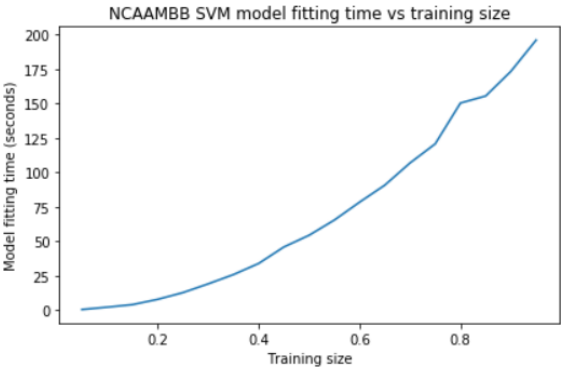
Training Times

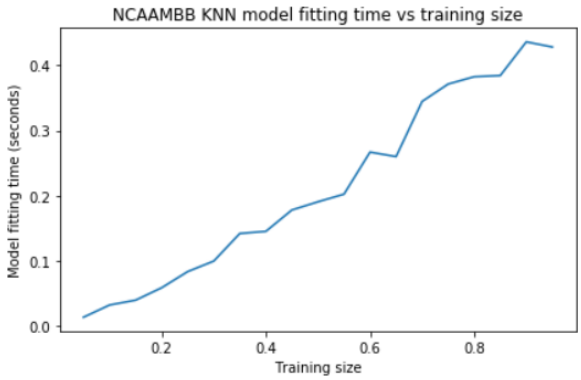
It is important to be aware of the different requirements around training times so that a modeler can set expectations for modeling iterations for hyperparameter tuning, or for retraining an existing model on new data. The plots below show the training times for each type of model, as a function of the training size. The training sizes used range from 5% to 95% of the data. Any data not used for training is used for validating the model. For example, if 45% of the data is used for training, the remaining 55% of the data is used for validation. Note: the KNN model is not included here as model training for that algorithm is not comparable to the other types of models for the reasons mentioned above in the KNN model description. KNN is included in the scoring times section.

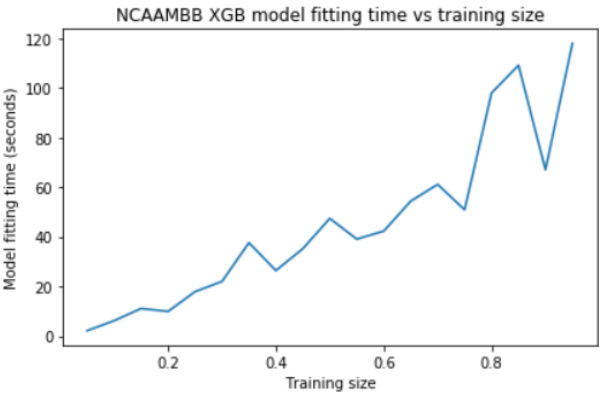
The plots below show that the single Decision Tree is the fastest model to train, followed by Neural Networks, Boosting, and SVM. Also, the Decision Tree model displays a linear trend on the training time, indicating that we can expect the training time to increase proportionately if new data is added. The Neural Network plot is mostly linear up until using 90% of the data. This could indicate an exponential relationship in which the model training would take much longer if data is continuously added. The plot looks similar for Boosting, and the exponential trend is much more obvious for the SVM model.











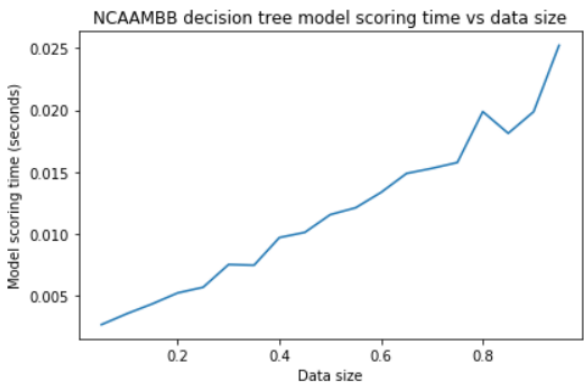
Boosting provided the best model performance in terms of accuracy, and the training time is not a concern. The scoring times for each model are check next.

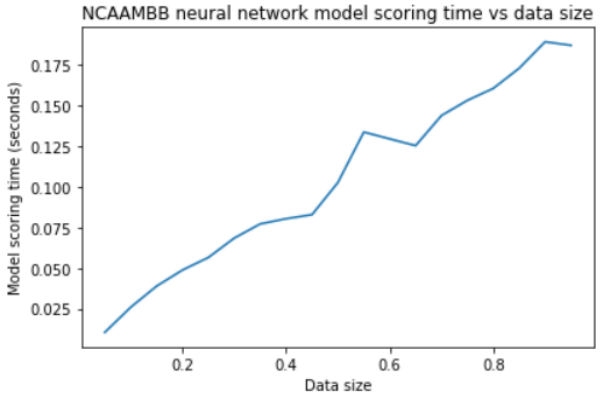
Scoring Times

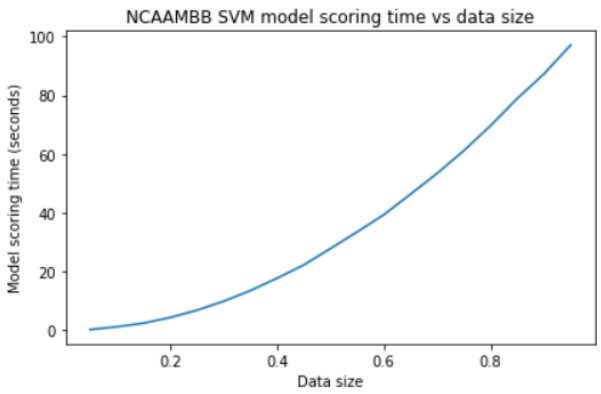
The amount of time required to score the model is perhaps even more important. While training the model has some endgame to it and eventually stops, scoring the model on a production system can continue in perpetuity. Depending on the application, the model may need to score thousands of new observations per second. For some extreme applications, like predicting stock market movement every one second, if the scoring time is too long then the prediction can already be outdated.

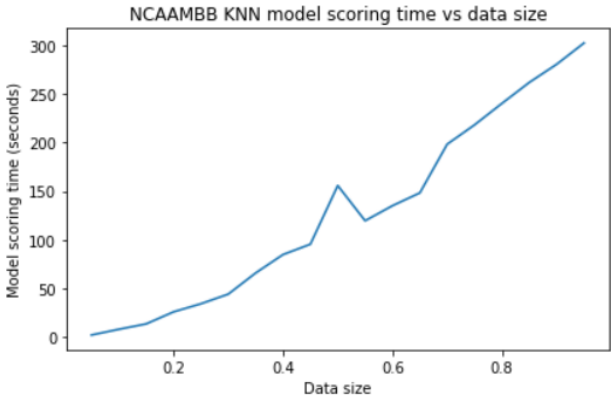
The plots below show the scoring time for all five models as a function of size. Technically, the scoring times can be compared after just scoring the models on a single observation. I have chosen to use the same format as above for the training time so that we can see the trend of scoring on a batch of new observations and reduce any variability that comes with scoring on a single observation.

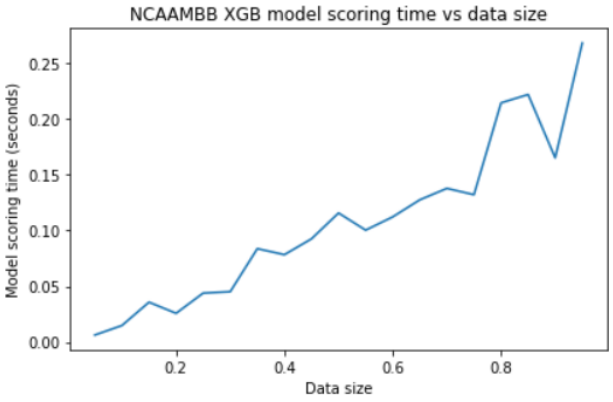
The plots below show that the Decision Tree, Neural Network, and Boosting model are similar, fast, and display a linear trend in the amount of time required for training the model as the size of the data increases. The SVM model is 400x slower, and as expected, the KNN model is an additional 5x slower on top of SVM, or 2000x slower than the Decision Tree, Neural Network, or Boosting model. As mentioned in the KNN description, the model is slow because the model is a lazy learner, holding off any computations until a prediction is needed. Besides that, KNN models search through 100% of the data to find the nearest neighbors.











Conclusion

Various models were created for the NCAAMCB and NBA datasets to predict which school, or team, would win the game. The best model was the Boosted trees using XGBoost. Decision trees are a good approach to capture the non-linearity that comes from human decision making in a game like basketball, and adding in the boosting feature provides a model that generalizes well to new data.

From the comparisons of NCAAMBB and NBA datasets, it is clear that both problems are difficult to solve. Much of this has to do with the human element and its non-deterministic nature. There are some factors that simply cannot be mapped to data, at least not with the current state of technology. For example, the mental state of the team's best player simply cannot be captured in data. Also, the data can change while the game is being played. For example, payers can get injured and stop playing or foul out of the game. An alternative approach would be to predict the outcomes of the games after each play, constantly updating the model with real time data.

While the human element impacts both types of problems, the results show that the NBA games are more difficult to predict. There are a few reasons for this:

- Players can trade teams in the NBA and that is not something that was considered for this analysis as it requires advanced feature engineering. If trades can be captured effectively in the model, then perhaps model performance would be better.

- College teams have a much larger home court advantage than NBA teams. NBA players become accustomed to traveling for away games and, as professionals, can perform well when the fans are against them. College level play is different because after all, the players are college kids. Traveling to another school can be tiring, and the home team's student body attending the game can really provide an edge to their school.

- There are 350+ schools and only 30 NBA teams. The variability in talent in college basketball is much larger than in the NBA. For many college games, the prediction is easy because one team is clearly better than the other. Predicting NBA games is naturally harder because the ability of the players on the opposing teams tends to be more similar relative to college level play.