Author: Michael Pecorino

Plagiarism trigger: I partially took this course in Fall 2020 and so some of the analysis is repeated.

Introduction

Dataset 1: 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, Major League Baseball (MLB) is using the Hawk-Eye [1] 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 an entirely different set of questions to be answered. The National Basketball Association (NBA) and Division I NCAA Men's College Basketball (NCAAMBB) use similar motion-capture technology called STATS [2]. The second driver behind sports analytics is the increase in the legalization of sportsbooks, which are companies that facilitate gambling on sporting events within certain states. The sports betting industry is expected to be an $8 billion-dollar industry by 2025 [3]. That figure will only grow as sports betting inevitably becomes legal in all 50 states. For this project, I am focusing on NCAAMBB school win/loss classification.

Dataset 2: Similar to the technology used to track every play in sporting events, smartphones have built in sensors that describe the orientation and movement of the device. While the phone is in our pocket, it can determine if we are standing, sitting, laying down, walking, walking downstairs, and walking upstairs. Therefore, the classification problem is of the multi-class type. This information can provide stakeholders (smartphone users and perhaps one day hospitals and insurance companies) with the data to study how movement impacts physical and mental health.

Data

Data for the Division I NCAA Men's College Basketball classification problem were taken from Sports Reference [4] and will be referred to as NCAAMBB in the remainder of the analysis. Data for the smartphone sensor classification problem were taken from Kaggle [5] and will be referred to as SENSOR in the remainder of the analysis. Table 1 shows the dimensions and description of the response variable for each dataset. The NCAAMBB response is nearly perfectly balanced, and the SENSOR response is reasonably balanced. Therefore, accuracy will be used to evaluate the models throughout the analysis.

Table 1: Dimensions and response variable

Data	**Observations**	**Variables**	**Response (Numerical Encoding, % of data)**
NCAA Men's College basketball (NCAAMBB)	51,347	41	Win (1, 50.1%) /Loss (0, 49.9%)
Smartphone sensor (SENSOR)	10,299	567	Laying (0, 18.9%), Sitting (1, 17.3%), Standing (2, 18.5%), Walking (3, 16.7%), Walking Downstairs (4, 13.7%), Walking Upstairs (5, 15.0%)

NCAAMBB only uses team level statistics as data from player movement sensors is not readily available to the average person. Thus the number of variables to describe win/loss is reasonable. Relative to NCAAMBB, the engineering behind the SENSOR data requires many more variables in order to capture the orientation and movement of the phone. Data processing steps were identical for both datasets, which entirely consists of normalization for the KNN algorithm so that variables with large magnitude do not dominate the distance calculation.

Since the NCAAMBB dataset describes human decision making in the game of basketball, in which noisy signal is inevitable, it is expected that generalization error for this dataset will be much lower than the more scientific SENSOR data. This hypothesis will tested visually throughout the analysis.

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), and Boosting.

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 and to improve generalization of the models. Performing well on new data is the goal, and overfit model is one that is fit extremely well to the model fitting dataset and cannot generalize well to new data. 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 optimal model parameters. Cross-validation consists of splitting the model training data into N subsets, also known as folds. The validation method is the following: fit a model on N - 1 subsets and evaluate the model on the one subset not used for fitting the model. Typical values for N include 1, 5, or 10. The cross-validation used in this analysis uses N = 5 subsets and can be referred to as 5-fold cross-validation. 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]

The five splits were made using stratified random sampling to ensure that the overall proportion of each response variable class was maintained in each of the five folds. The objective is to use the cross-validation results to find the optimal model hyper parameters. Therefore, for each model hyper parameter test, five versions of model metrics are produced. A cross-validation summary is generated by taking the average performance of all five results. 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. Table 2 provides the number of observations used in each phase of the analysis.

Table 2: Description of data splits

	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
NCAAMBB	4,849	4,848	4,848	4,848	4,848	11,161	15,945	51,347
SENSOR	1,015	1,026	1,047	1,021	1,038	2,205	2,947	10,299

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, e.g. "Is variable x >= 10?". 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. Once all leaf nodes are established, the post-pruning algorithm will query if each node split improves performance on some hold-out validation data. If a node split increases error, it will be pruned away and a leaf node will be created in place of the splitting node.

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 the cross-validation method described in the Data section.

Post-pruning tends to 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 stop the tree growth before it can get 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 to the training data and would not generalize well to unseen data. In fact the smallest alphas could allow the tree to fit the training data perfectly. Very large alphas would underfit because it would be too conservative and would not fit either the training data or any unseen data very well.

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 error information propagates backward through the Network to create the learning loop. Neural Networks are complex and usually uninterpretable, 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 as well as simpler models.

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 boundary can be 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. The goal of SVM is to find linear separability and uses kernel transformations in order to increase the separability of the data. SVM is primarily built for binary classification, and so for the SENSOR data, multiple binary classification models are used in a 1 vs. all method (e.g. walking vs. not walking, sitting vs. not sitting). In this case, the argmax (most probable) label is assigned as the class.

K-Nearest Neighbors (KNN)

The Nearest Neighbors algorithm does exactly as the name implies – it finds the k nearest neighbors to the input data and uses those nearest neighbors to inform the prediction. Nearest neighbors are determined by similarity to the input data using some distance metrics. The number of neighbors and distance metric are the two key parameters to optimize. The model can disregard k entirely and let the model consider 100% of the data as neighbors and weight each neighbor's contribution to the prediction based on their similarity to the input data. We can also combine the first two approaches and use similarity-weighted k neighbors, which is the approach taken in this analysis.

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 has 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. KNN adjusts 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 eventually dimensionality reduction, but not performed here) to prevent a subset of variables from dominating the distance calculation.

Boosting

Boosting is the idea of fitting a sequential ensemble of weak learners that each try to correct the errors of the previous models in the sequence. Each sequential model is provided the training data with the observations weighted by how successful the ensemble has been at predicting the class. The more errors the ensemble is making, the higher the weight will be for the observation.

In this analysis, the decision tree model is used as the base model for the boosting ensemble. The boosting process continues for some optimal number of models (trees) as determined by cross-validation. Using trees in an ensemble is often better than a single decision tree because it reduces the bias of a single tree. Even if cross-validation is used to optimize the 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.

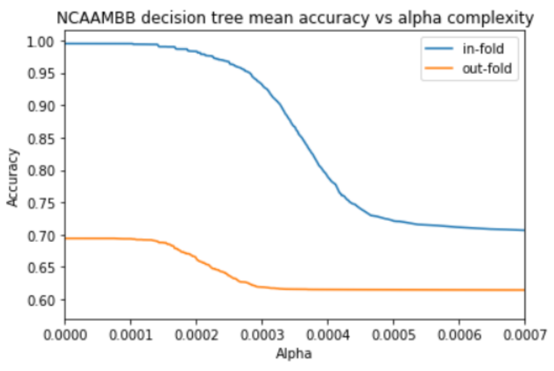
Since the boosting algorithm applies more to the observations with errors, they can easily overfit to the training data. Therefore, it is important to tune hyperparameters in cross-validation to ensure the ensemble can generalize to new data.

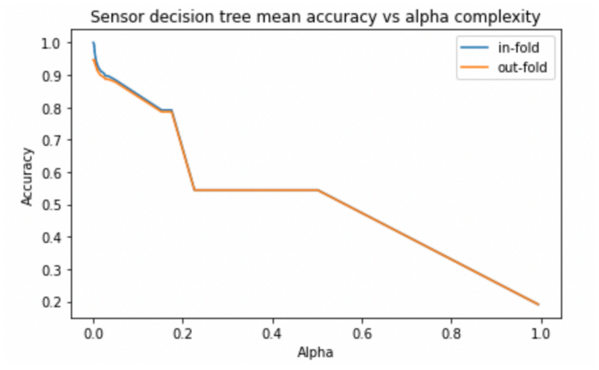
Learning Curves

We can plot what is referred to as a learning curve that shows how model performance changes over some variable, i.e. 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 (performance can improve), some will be in an optimal zone, and others will overfit (performance is getting worse).

Decision Trees

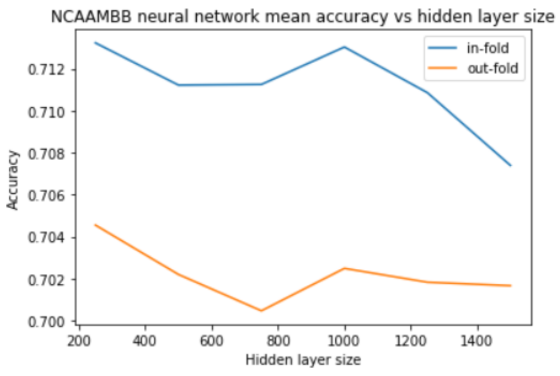
Since post-pruning was the focus for the decision tree model, we can plot the mean accuracy from cross-validation for the complexity parameter alpha. The decision tree algorithm performs optimally for both datasets at the lowest complexities. As alpha increases, more pruning is applied to the tree. The 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. One finding to point out right away is that the initial hypothesis is holding true - the SENSOR data is much more predictable than NCAAMBB. We will see this three throughout the analysis.

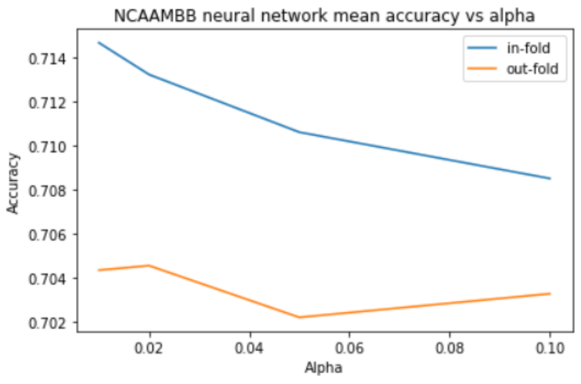


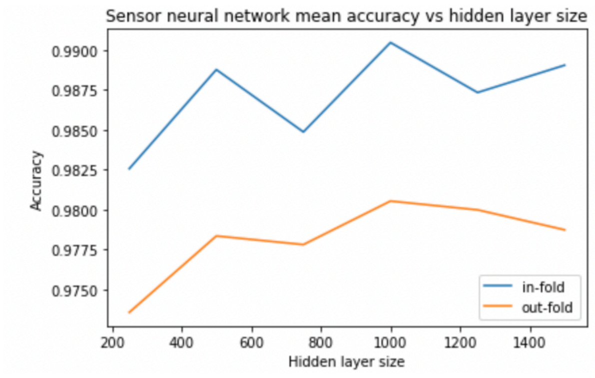


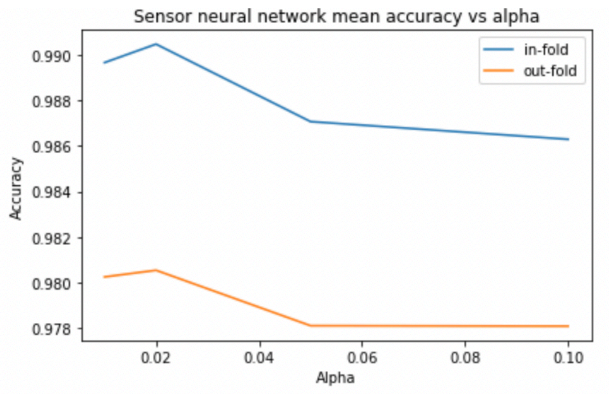
Neural Network

For Neural Networks, the learning curve is plotted for some of the hyperparameters, like hidden layer size (left plots) and alpha regularization (right plots). There is a clear difference in the optimal hidden layer size for the two datasets. The noisy NCAMBB dataset, which can easily be overfit, requires a smaller amount of hidden layers. On the other handle, the complex, mostly noise free SENSOR data, gives optimal performance with 1,00 hidden layers. Performance by alpha has the same general shape for both datasets, with the optimal value being around 0.2.



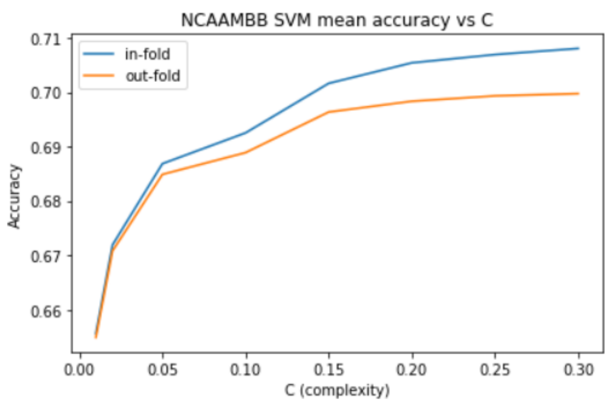


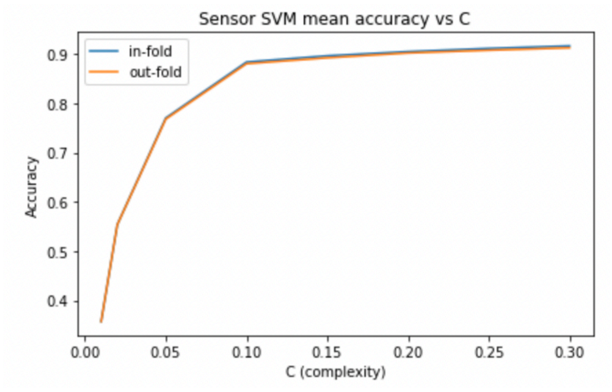




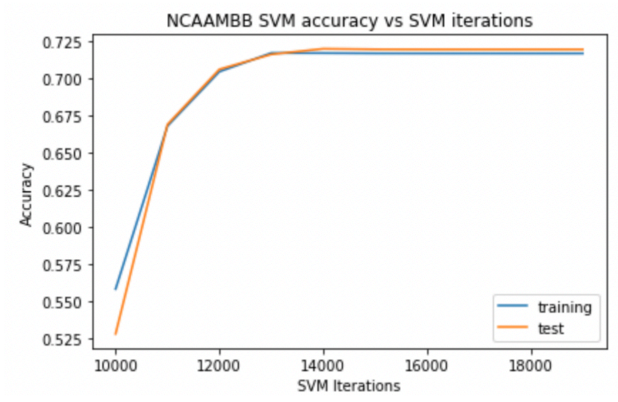
SVM

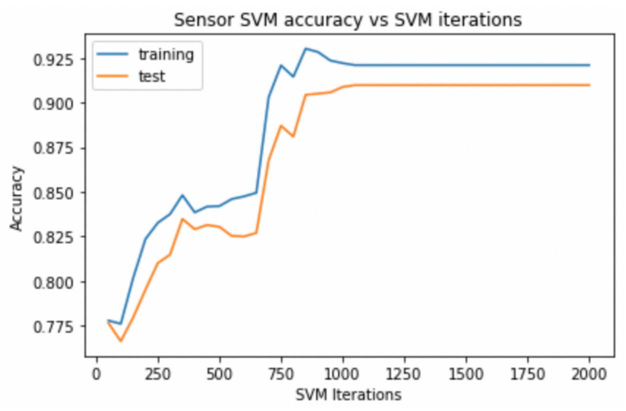
SVM is all about fitting a line to separate the data. If not constrained, the model can become very complex and fit the training data perfectly, generalizing very poorly on new data. For SVM the complexity of the decision boundary is interesting to plot for a learning curve. For the NCAAMBB dataset, 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 (effect not shown to better see rise and plateau). The SENSOR data performs very well at high complexity.





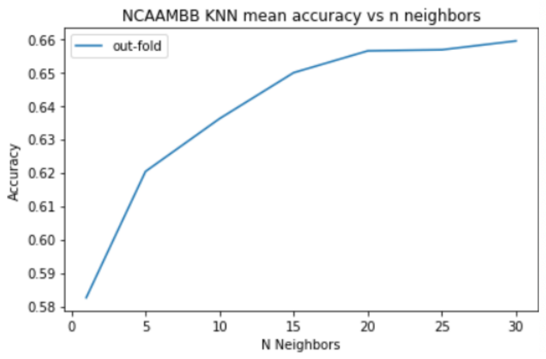
SVM performance is sensitive to the number of iterations allowed. The figures below suggest that for each dataset, there is an x number of iterations for which fitting beyond this number is unnecessary. For the NCAMBB dataset, performance flattens out at around 14,000 iterations and the Sensor dataset flattens around the 1,100th iteration.

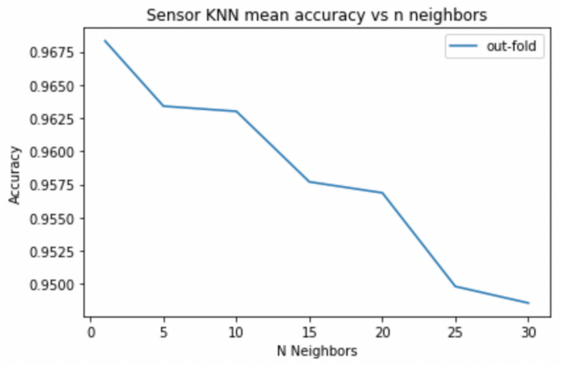




KNN

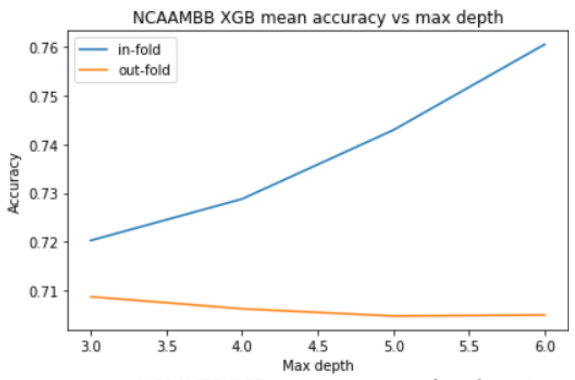
For the KNN algorithm, we can plot the number of neighbors on the learning curve. There is no need to show the in-fold performance for KNN because since KNN is a lazy learner, the training folds would be queried on itself resulting in 100% accuracy. The plots below show the average model performance on the out-fold data for neighbors from 1-30, in steps of 5. For the NCAMBB dataset, the performance on the out-fold dataset begins to level off after 20 neighbors. It is likely that choosing 20 would be optimal when considering the extra computation time needed for 30+ neighbors. We can see right away that performance on NCAAMBB for KNN is slightly lower than other algorithms. This may be due to using unimportant basketball statistics variables in the distance metric or having, quite literally, "noisy neighbors" that give mixed classifications. For the SENSOR dataset, the performance on the validation data is best with 1 neighbor. This speaks to the physics embedded in the the smartphone sensor data. Data is highly reproducible due to physical laws like gravity and the general shape of the human body, so the best estimate for a query is whichever historical data point is most similar.

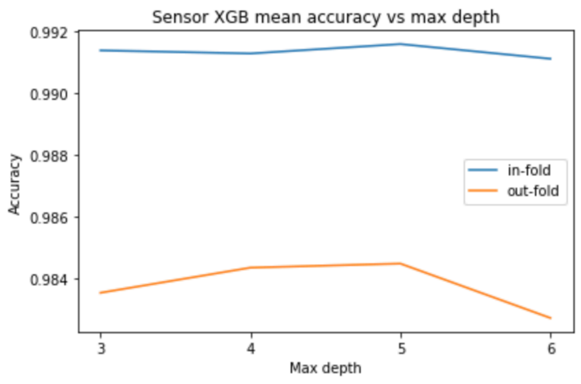


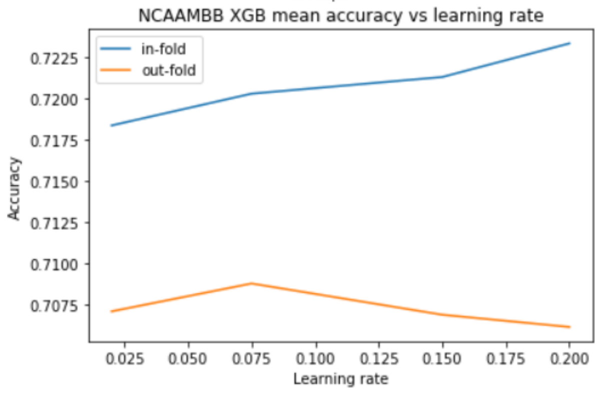


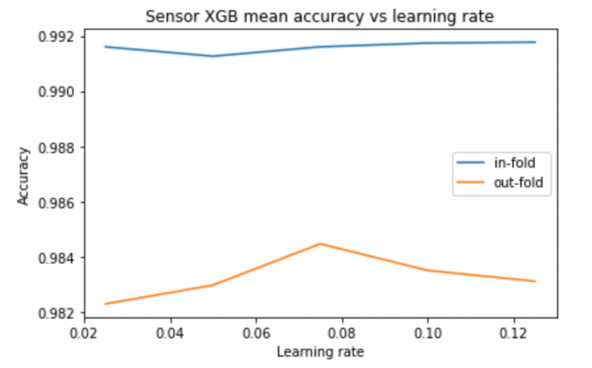
Boosting

While there are many hyperparameters to tune for boosted decision trees, the plots below show the max depth (left plots), learning rate (right plots), and minimum samples to make a split (bottom plots). Again, we see that the NCAAMBB can be easily overfit at model max depths. In fact, a max depth of just 3 was optimal for that dataset, whereas a max depth of 5 was optimal for the SENSOR data. The U-shaped learning curve called out earlier is evident in the SENSOR data. Both datasets have optimal learning rates in the .075-.08 range.

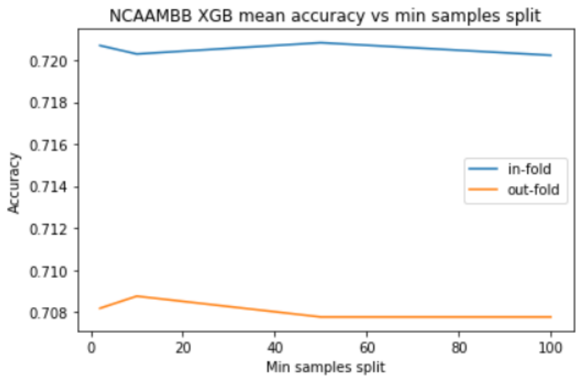


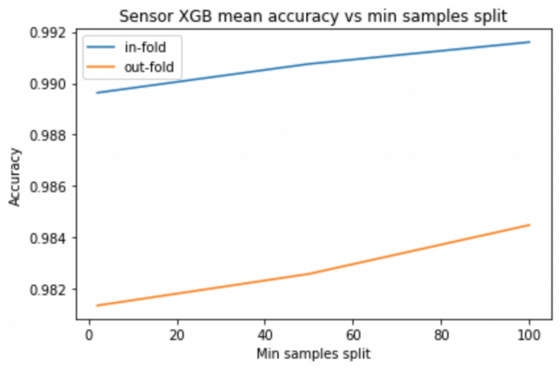






The minimum samples required to make a split is also interesting to plot on a learning curve. As the minimum samples increases, we expect to see the in-fold curve to decrease. The reason is because the model would become increasingly constrained and not make splits that would improve performance, so performance would trend down as the hyperparameter increases. We do see this outcome in the NCAAMBB dataset. However, there appears to be a somewhat linear relationship between min samples split and the performance in the SENSOR data, for both the in-fold and out-fold samples. While this result makes sense for the out-fold data, the result for the in-fold data may be clouded by the results from other hyper parameters, like max depth. Each plot keeps the other hyper parameters fixed and that effect could be showing here, e.g. 100 samples required could be helpful for max depths of 5 but not max depths of 3.





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%. SVM and KNN suffer from the noisy classifications contained in this dataset.

Table 3: Accuracy for training, validation, and test data for the NCAAMNBB 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 SENSOR dataset, Neural Network had the best performance on the validation data and so was chosen as the best model. The accuracy on the unbiased test data was 94.57%. The Neural Network was most effectively able to capture the physical properties embedded in the sensor data.

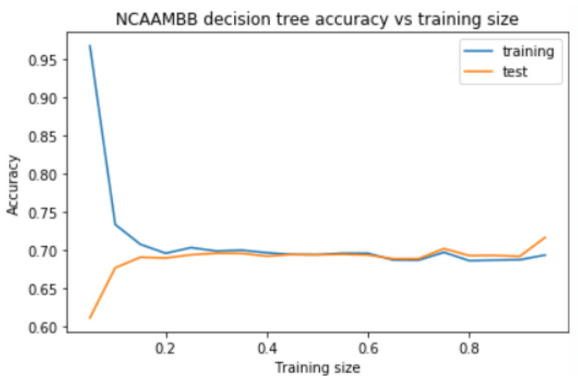
Table 4: Accuracy for training, validation, and test data for the SENSOR dataset.

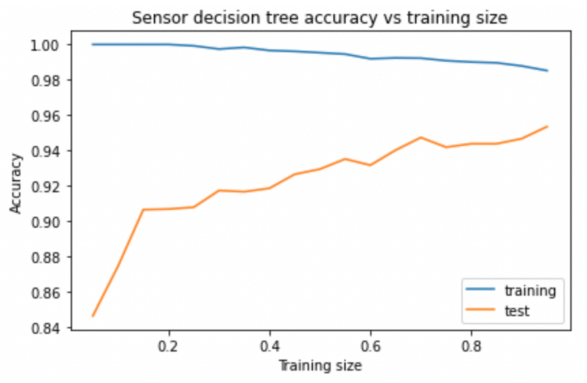
Dataset	**Decision Tree**	**Neural Network**	**SVM**	**KNN**	**Boosting**
Training	99.07	98.93	91.35	-	99.63
Validation	99.09	**97.41**	91.25	96.83	98.68
Test	87.00	**94.57**	89.92	86.70	93.96

Model Performance by Training Size

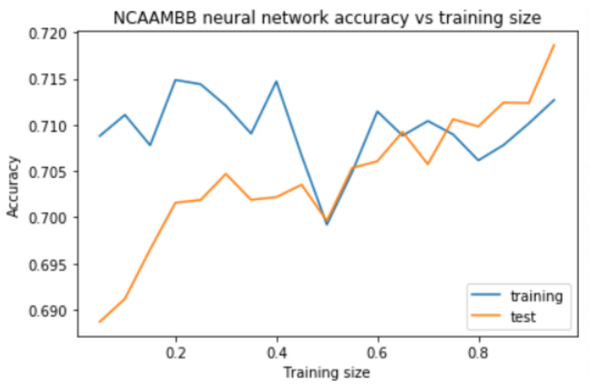
We should check how each model performs as a function of how much data is used to train the model. The reason this is important is because certain models may perform better after more data is collected. For this experiment, 100% of the data was used and the data was split into training and testing data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

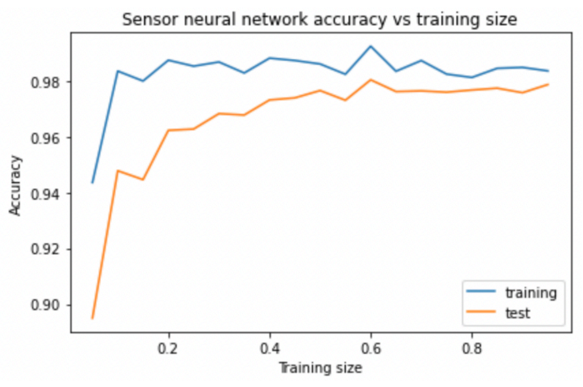
For NCAMBB, a performance plateau is reached very quickly - the number of training examples does not have much of an impact on the performance. There is a small bump at the end, but that is only for a 5% sized test set and so may have high variance. For the SENSOR data, there is a clear upward trend in performance with increasing size of the data and it is approaching the performance of the best model, the Neural Network.



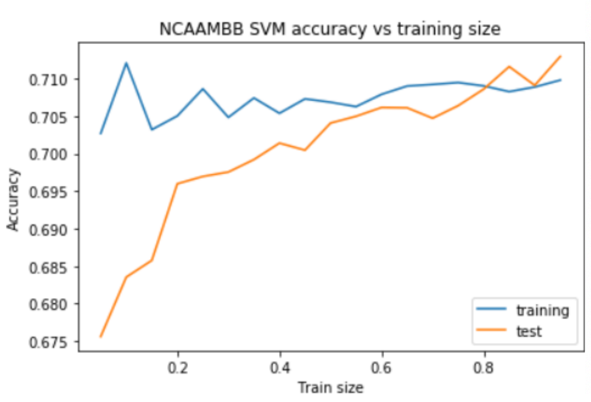


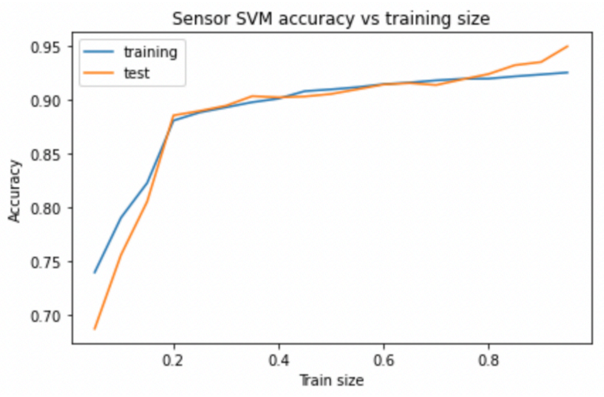
Both datasets shows an increasing trend on the test data as more data is used to train the Neural Network model. The SENSOR data reaches a plateau earlier. This is interesting because the Boosting model was chosen for the NCAAMBB dataset, but if more data can be obtained, then we may see the Neural Network outperform boosting.



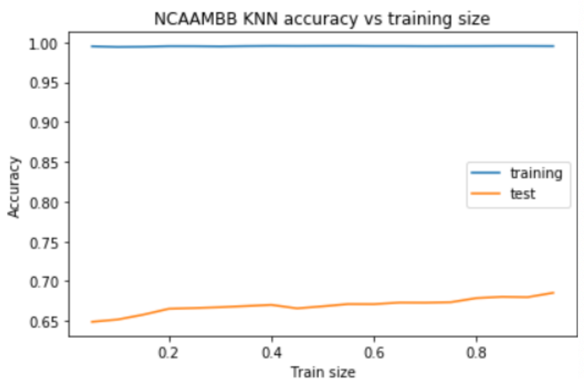


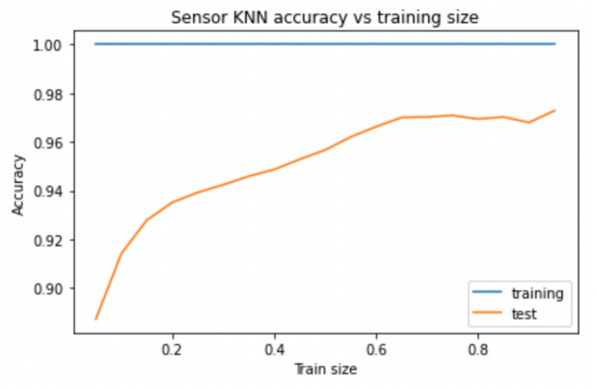
Both datasets show a positive trend in the SVM model as more data is added to the training set. We should expect SVM to perform better in the future when more data is available. However, the performance on the SENSOR data shows signs of leveling off. This can be due to the SVM model not being complex enough relative to the Neural Network.



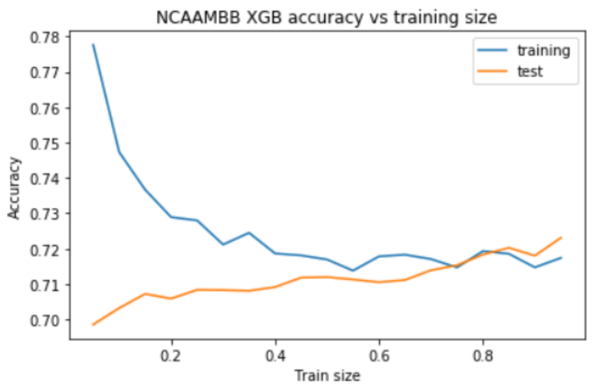


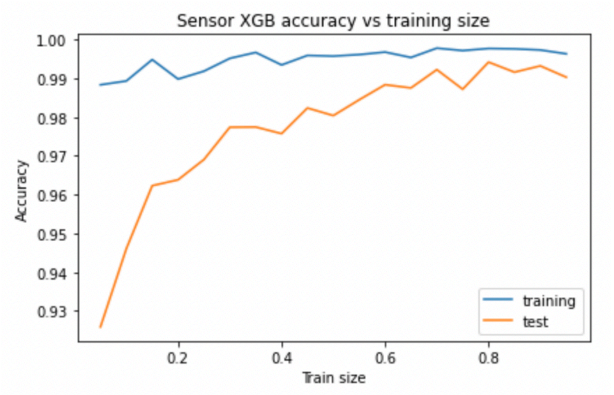
The KNN training score is not really relevant here as KNN training is not comparable to the other models. The test data score has a steady increase as more data is added to the training data, although the rate of the increase is much more prevalent for the much less noisy SENSOR data. As long as the data we are adding is noise-free, then we would expect performance to increase steadily for KNN because we would increase the likelihood of having more similar neighbors to the query points.





The Boosting 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.





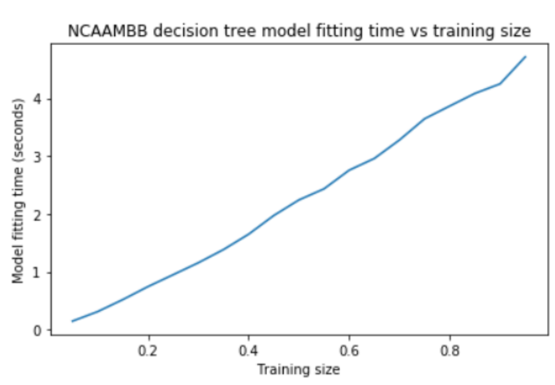
Training and Scoring Times by Training Size

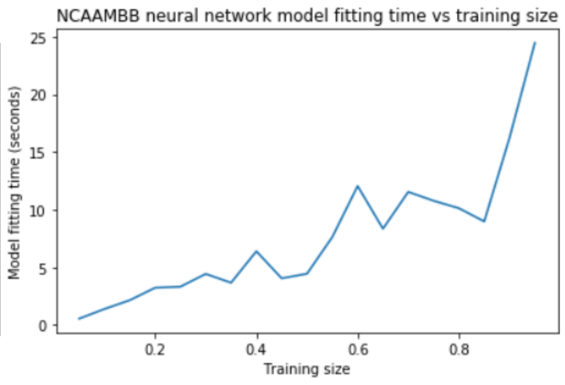
Considering just how a model performs on new data, while a significant factor, is not the only consideration for building a Machine Learning model. Modelers are also interested in how much time is required to train a model and how much time is required to score the model on new data. For simplicity,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

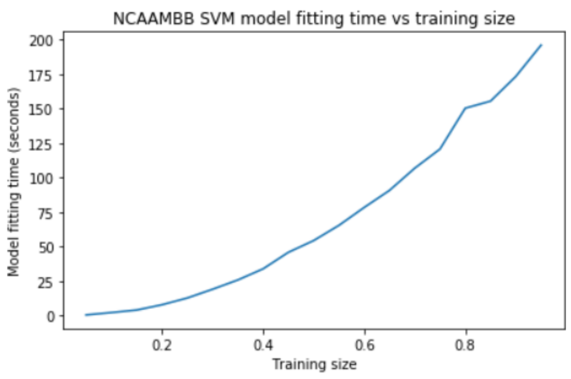
Training Times

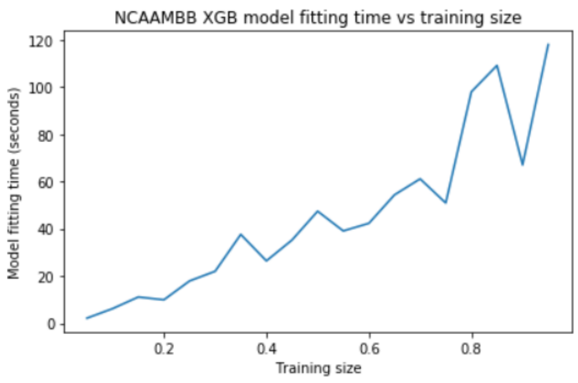
It is important to be aware of the different requirements around training times so that a modeler can set expectations for modeling iterations for hyperparameter tuning, or for retraining an existing model on new data. The plots below show the training times for each type of model, as a function of the 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 previously mentioned. 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.





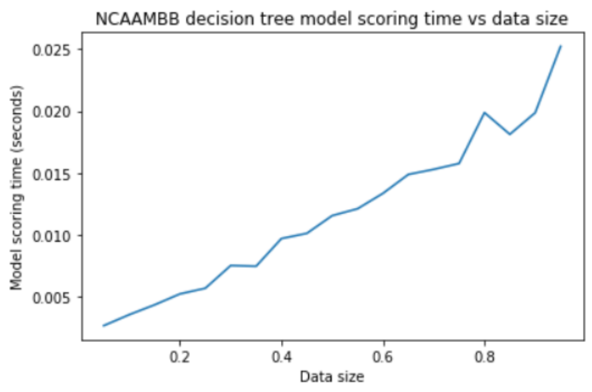


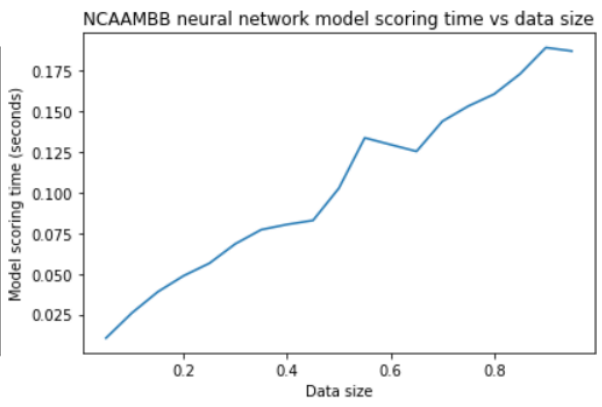


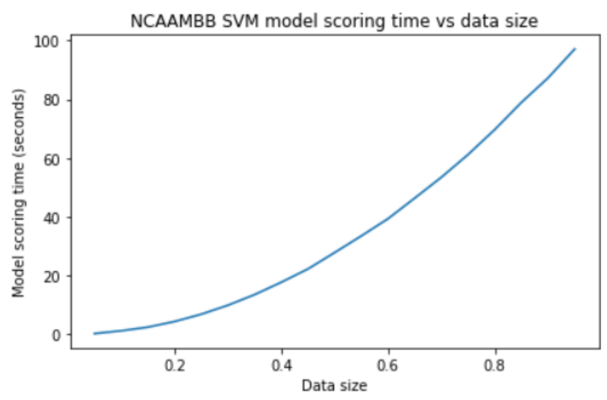
Scoring Times

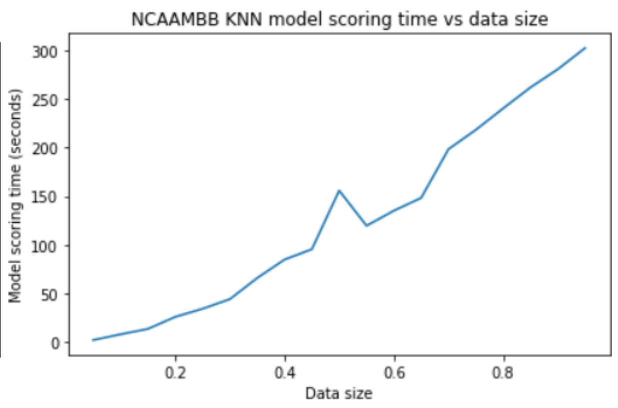
The amount of time required to score the model is perhaps even more important. While training the model has some endgame to it and eventually stops, scoring the model on a production system can continue in perpetuity. Depending on the application, the model may need to score thousands of new observations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

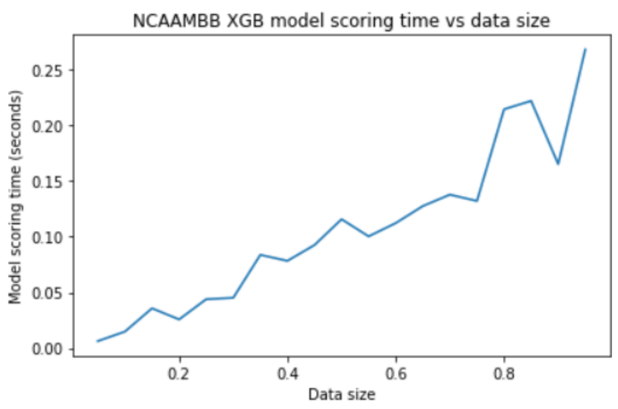
The plots below show the scoring time for all five models as a function of size. Technically, the scoring times can be compared after just scoring the models on a single observation. I have chosen to use the same format as above for the training time so that we can see the trend of scoring on a batch 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 NCAAMBB dataset to predict which school, or team, would win the game. The best model was the Boosted trees. 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. The similar procedure was used on the SENSOR data to predict if the person was laying down, sitting, standing, walking, walking downstairs, or walking upstairs.

From the comparisons of the two datasets, it is clear that the human element in the game of basketball makes that problem difficult to solve from a Machine Learning perspective. The outcomes are non-deterministic, and there are some factors that cannot be mapped to data even with advanced computer vision technology tracking player movement. For example, the mental state of the team's best player simply cannot be captured in data. Also, the input 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. The SENSOR data is describing physical properties about gravity and the way the human body moves. The predictability is clear when using effective Machine Learning models like the Neural Network or boosted Decision Trees. Even a simple algorithm like KNN performs reasonable well, and would improve over time as new training data is collected and more similar neighbors can be used in the query.

References

1. https://www.hawkeyeinnovations.com/sports/baseball

2. https://apnews.com/press-release/pr-businesswire/e3b90b0946e549e5952159a82c14e69a

3. https://www.marketwatch.com/story/firms-say-sports-betting-market-to-reach-8-billion-by-2025-2019-11-04

4. https://www.sports-reference.com/

5. https://www.kaggle.com/