Sauce Validation Final Paper

Introduction

Background

Hockey is a sport played on ice where there are five skaters and a goalie on each team, with the objective being to score more goals than the other team. The National Hockey League (NHL) consists of 32 teams and is regarded as the top hockey league in the world. The skill level and speed of the NHL has led to certain in-game events (shots, passes, hits, etc) happening within milliseconds of each other, making it difficult to analyze what leads to team success. This led us to question which on-ice events are important to an NHL team winning a game and if we could use these variables to predict which team won.

Data Collection

We used Peter Tanner's Moneypuck, a hockey analytics website, to gather NHL data for the project. The "all teams" dataset contains over 100 variables for every single NHL game ranging from 2008 until present day. All of the non adjusted or expected statistics are scraped directly from the NHL's official league API.

Data Explanation & Cleaning

Before we started building our models, we decided that box score or "hard" statistics should be the only variables we used in our models to predict the home team winning a hockey game. We wanted to implement statistics that anyone watching a hockey game could observe and understand. Examples of box score statistics include the number of shots on goal for a team or the number of faceoff wins by the opposing team. Moneypuck had many variables in their dataset that they created based on their formulas using the box score statistics. Our group recognized that variables that contained the word "adjusted", "expected", "danger", or "credit" were Moneypuck variables, and therefore we removed them from our model building process. Furthermore, we did not want to use any "goalsFor" or "goalsAgainst" variables because that relationship determines the outcome of hockey games.

After removing all of the mentioned variables, we went from over 100 variables in our dataset to 38 variables. We used all of the remaining variables as predictors in an initial LDA model and discovered there were some collinearity concerns in our dataset. As a result, we ran a multicollinearity diagnostic test on the remaining variables through the "imcdiag" function in the "mctest" package. The test identified 17 variables as problematic that we then removed before building our models. We ultimately had 21 predictors of Home and Away team statistics with over 14,000 observations to predict whether the home team won a hockey game.

Model Building and Analysis

LDA and QDA

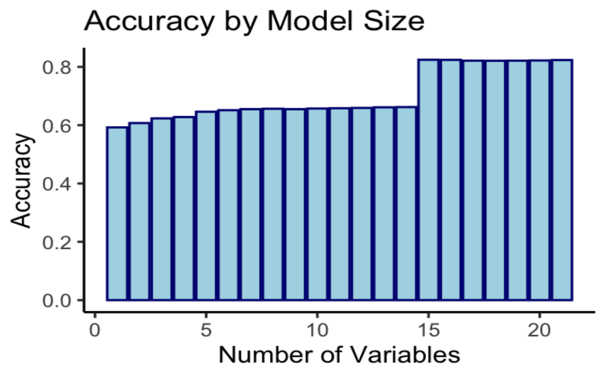
We built and evaluated the LDA and QDA models using code from HW7. The models were very similar, with an 80/20 training/test split. The only difference between the models was that we scaled the LDA data and did not scale the QDA data, as LDA involves each predictor assuming the same variance. We then created a confusion matrix to evaluate the accuracy, sensitivity, specificity, and area under the curve of each model. From the confusion matrix, we found that our QDA model performed better overall, with an accuracy of .82, a sensitivity of .85, a specificity of .79, and an AUC of .90. Our LDA model resulted in rates of .69 for accuracy, .86 for sensitivity, .48 for specificity, and .76 for AUC.

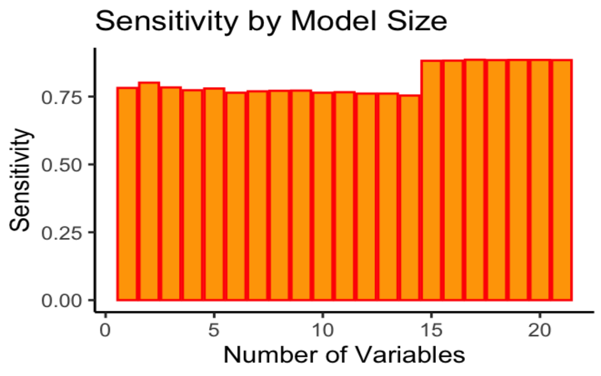
Logistic Regression with Cross Validated Model Selection

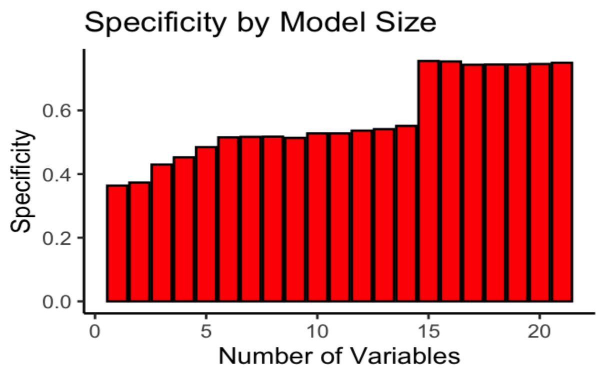
With collinear variables already controlled for in our data, the first step for our logistic regression was to decide an approach to finding the best logistic model we could create. The group chose to do a bottom-up model selection process for this. This form of model selection was chosen as a balance between randomly creating various models and comparing their strengths and weaknesses (easiest and least formal approach) and creating a complete set of all possible models to compare (most computationally taxing). While it is possible bottom-up selection may not give us the best possible model in its subset of "best" models, it does give us the ability to see where and when models jump in performance and what variables interact to best predict wins.

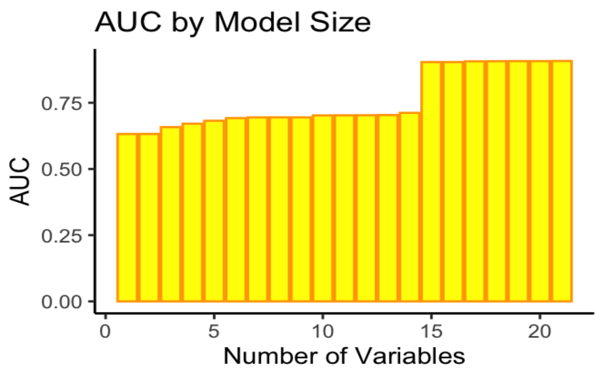
The model selection was done using a function custom-built for our dataset and purpose, and within it all results were cross validated using a random 80-20 split of train and test data. Starting with single predictor models, each iteration of the process tested all models with all previously selected predictors and one additional predictor added, and the best model according to cross validated accuracy was kept for each iteration. After model selection, we were left with 21 models with a range of 1-21 predictors in each; we can thus refer to our logistic models based on the number of predictors.

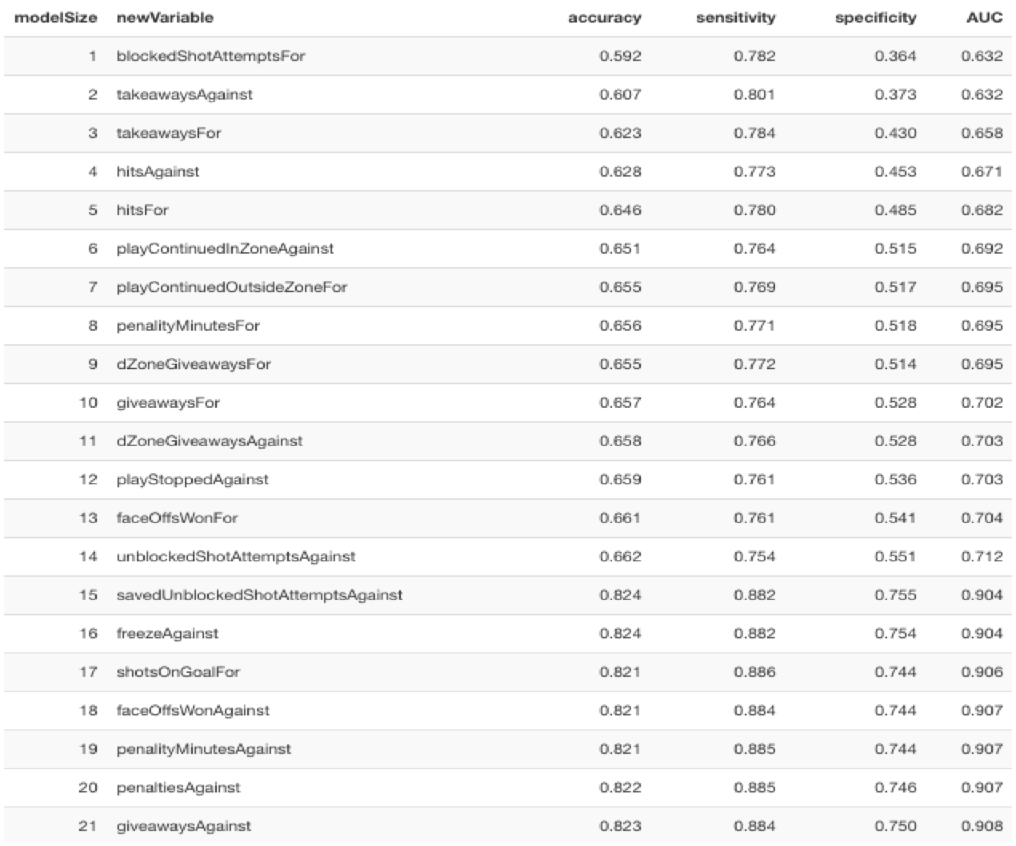
To assess these models, we used some of the standard measures of success for cross-validated models: accuracy, sensitivity, specificity, and Area Under the Curve (AUC). In each of the plots below, you can see the summary statistics for the best model of each model size based on the bottom-up model selection. Notably there is a large jump in model accuracy in the 15 variable model- this is due to the inclusion of *unblockedShotAttemptsAgainst* and *savedUnblockedShotAttemptsAgainst*, which together will give the model a linear combination to find a rough estimate of *goalsAgainst*, which we do not include in the set of predictors we allow because of its strong ability of predict the results of games on its own or especially when combined with *goalsFor*. The most accurate single predictor model included the predictor *blockedShotAttemptsFor*. A full list of the models and their successively added predictors can be found below the plots.







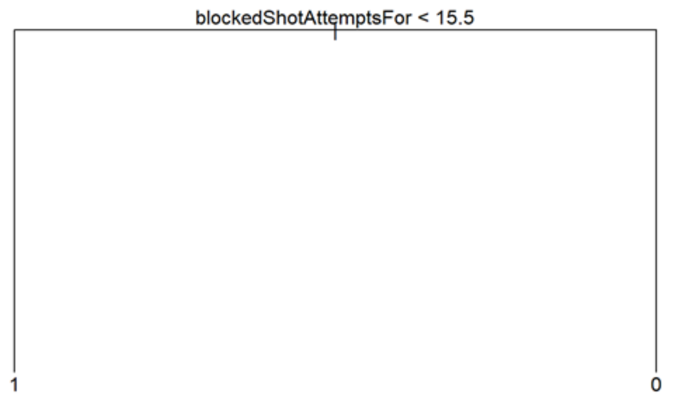




modelSize	newVariable	accuracy	sensitivity	specificity	AUC
1	blockedShotAttemptsFor	0.592	0.782	0.364	0.632
2	takeawaysAgainst	0.607	0.801	0.373	0.632
3	takeawaysFor	0.623	0.784	0.430	0.658
4	hitsAgainst	0.628	0.773	0.453	0.671
5	hitsFor	0.646	0.780	0.485	0.682
6	playContinuedInZoneAgainst	0.651	0.764	0.515	0.692
7	playContinuedOutsideZoneFor	0.655	0.769	0.517	0.695
8	penalityMinutesFor	0.656	0.771	0.518	0.695
9	dZoneGiveawaysFor	0.655	0.772	0.514	0.695
10	giveawaysFor	0.657	0.764	0.528	0.702
11	dZoneGiveawaysAgainst	0.658	0.766	0.528	0.703
12	playStoppedAgainst	0.659	0.761	0.536	0.703
13	faceOffsWonFor	0.661	0.761	0.541	0.704
14	unblockedShotAttemptsAgainst	0.662	0.754	0.551	0.712
15	savedUnblockedShotAttemptsAgainst	0.824	0.882	0.755	0.904
16	freezeAgainst	0.824	0.882	0.754	0.904
17	shotsOnGoalFor	0.821	0.886	0.744	0.906
18	faceOffsWonAgainst	0.821	0.884	0.744	0.907
19	penalityMinutesAgainst	0.821	0.885	0.744	0.907
20	penaltiesAgainst	0.822	0.885	0.746	0.907
21	giveawaysAgainst	0.823	0.884	0.750	0.908

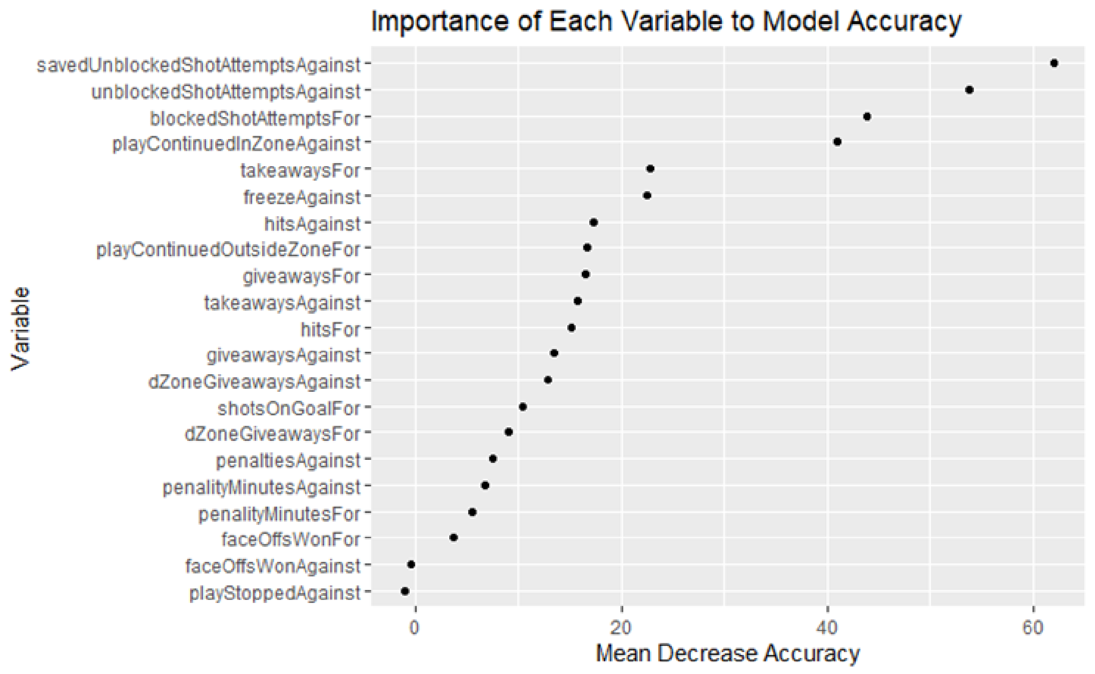
Decision Trees and Random Forest

To make our decision trees, we used the same 21 variables as in our other models, without scaling or transformations. As with the other variables, we used an 80/20 training and test set split on our data. Our resulting decision tree contained only a single variable *blockedShotAttemptsFor*. Perhaps surprisingly, no other variables were deemed significant enough to be included in the decision tree, emphasizing the importance of shot attempt variables demonstrated by our other models.



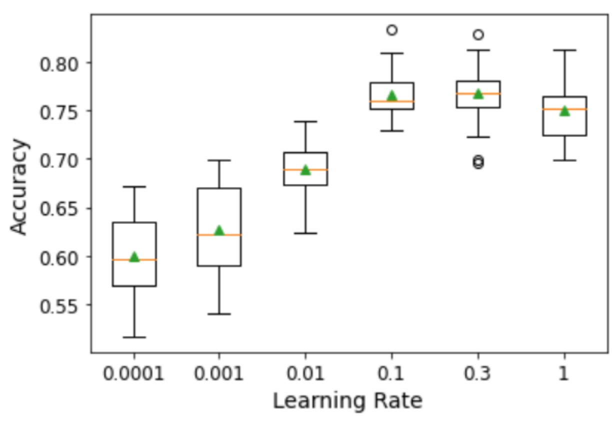
This model, while only containing one variable, was still able to achieve an accuracy of 0.597 and an AUC of 0.586, with a sensitivity of 0.692 and a specificity of 0.480. However, these figures remained well below those of our other models, and we believed including more variables could improve the predictions. This led us to explore a random forest model. Using out of bag error estimation, we determined the best random forest models to come with twenty variables per tree. Building 500 trees gave us a much more accurate model, as our random forest achieved an accuracy of 0.805, sensitivity of 0.840, specificity of 0.763, and an AUC of 0.801. This affirmed our belief that adding more variables would produce better predictions, as it did with many of our other models.

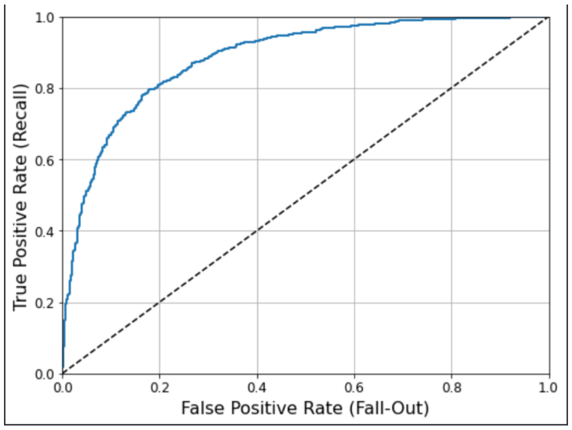
Again, shot attempt variables were the most important predictors for our random forest model. In fact, four variables, *savedUnblockedShotAttemptsAgainst, unblockedShotAttemptsAgainst, blockedShotAttemptsFor,* and *playContinuedInZoneAgainst,* drove much of the prediction accuracy of our random forest model, as demonstrated by the variable importance plot. Three of these variables are related to shot attempts. Meanwhile, variables such as *playStoppedAgainst* and those having to do with penalty minutes and faceoffs were deemed less important by our random forest model.



Gradient Boosting

We built a gradient boosting model using the "XGBoost" package, an open source software that does an extremely optimized version of gradient boosting. After experimenting with several different learning rates and running 10-fold cross validation, we found that a learning rate of 0.3 led to an accuracy of around 76% and an AUC of 0.88. For reference, the Florida Panthers were the top team in the NHL this season and won roughly 70% of their games this season.





Additional Models

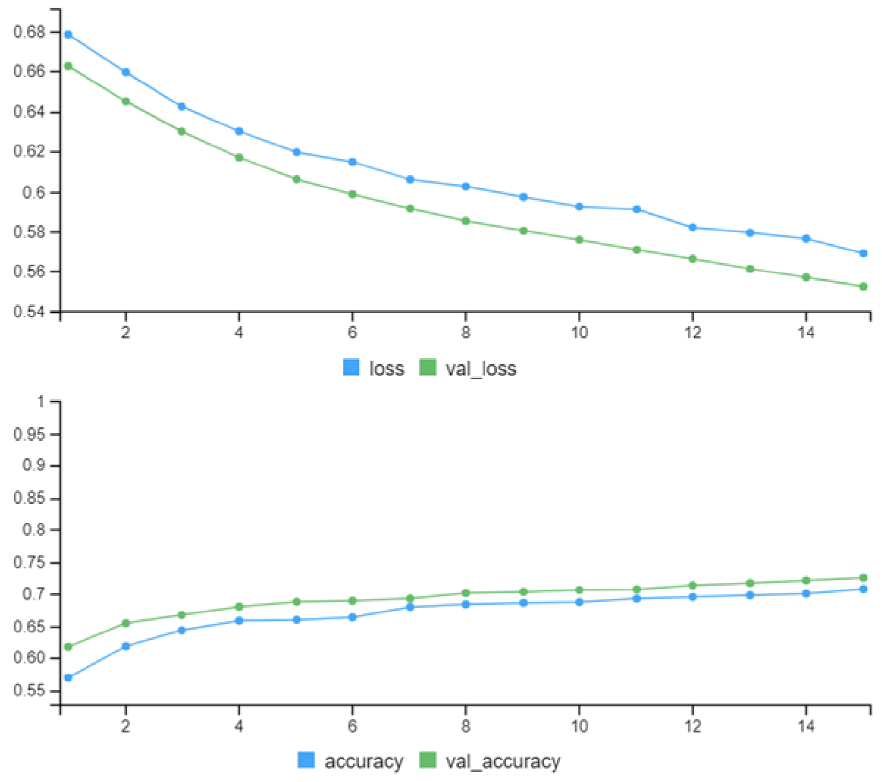
To try to further improve our accuracy, we built several other models. These attempts did not yield predictive accuracy that was comparable to our best models, however.

K Nearest Neighbors

As with our LDA model, we scaled our data before building our KNN model. We then tested various values for k, determining their accuracy using cross validation. None of the values for k produced a cross validation accuracy beyond 66%, and the best k value, 125, achieved an accuracy of 0.643 on the test set. Although the KNN model produced much worse prediction accuracy compared to many of our other models, the sensitivity, 0.813, was fairly high. It seemed KNN did a good job correctly predicting teams that won, while it struggled to predict teams that lost. This may be because there were slightly more teams that won (55%) than lost (45%) in our dataset, though this likely would not explain the entire gap between false positives and false negatives we saw in our KNN model.

Neural Networks

We also built a dense neural network for our dataset. The NN model included one hidden layer with 15 units, a dropout rate of 0.4, and a softmax activation function. After 15 epochs, the model accuracy remained 0.713, again significantly below our best models. Although we attempted other neural network structures, including more hidden layers or a different activation function, we ultimately could not improve accuracy beyond this point.



Predicting Future Games

We also tested the limits of our data by trying to use a team's statistics over the course of a season to predict how they would play in future games. If a team has a certain number of shot attempts, hits, penalties, and faceoffs won per game, we thought we could use that information to predict whether the team would win in the future. Unfortunately, none of the models we built achieved a prediction accuracy much better than about 55%, or the baseline probability the home team will win. The models we built are much better at predicting whether a team won a game, knowing their stats from that game. This can help us understand whether a team "deserved" to win a game based on how they played, or whether they earned a tough luck loss. When we try to extrapolate the insights from our models to predict the future, we have much less success.

Conclusions and Takeaways

Basic Takeaways

Given the size of our dataset and the fact that we only used 21 predictors to build our models, it comes as no surprise that models with more variables produced more accurate predictions. Any small amount of data we could get out of a hockey game generally improved our ability to predict who won. Thus, the best producing models included most or all of the variables, even if the weight applied to some of them was small.

As shown by our logistic and random forest models, the variables with the greatest impact on accuracy seemed to deal with how many shot attempts a team and their opponent had in the game. Since goals scored and goals allowed are the determining factors in who wins and loses a game, it is not surprising to see shot attempt variables rated as very important. The more shots a team attempts, the more likely they are to have won the game. In general, stats that were more correlated with goals contributed the most to our model accuracy. On the other hand, we were surprised to find the variables dealing with penalty minutes and faceoffs to have little importance in our models. These variables are discussed a lot on hockey broadcasts, as announcers will talk about the importance of winning faceoff battles and avoiding penalties. Instead, we found these stats correlate little with winning games.

Potential Sources of Error

Hockey is a game that is constantly evolving. Games from the 2008 season have differences from games today, and those differences will find their way into the data. Before the 2018 season, the NHL shrank goalie equipment, which led to the highest goal scoring season for teams since the 2005-2006 season. Referees also began to crack down on certain penalties such as cross checks and interference, leading to more power plays, which leads to more goals. Changes like these can lead to an overweighting of shots and other offensive metrics when scoring was not as common in the previous 10 seasons that the model used.

The sport of hockey itself is also very random and there is a lot of variability in how teams can win a game. In hockey, a team will average around 40-50 shot attempts and 5-7% of them will usually go in, whereas in basketball, teams shoot around 40-50%, so the skill level often outshines any luck in play. A team in hockey can outperform the opposing team in several metrics and dominate play, but unlucky shooting or great goaltending by the other team can lead to an "undeserving win" for the opposing team, or a false positive in the model. This phenomenon is seen plenty of times when looking at one sixty-minute game of hockey and not a whole season or a playoff series.

There were also errors within our data, both raw and cleaned, that should be mentioned as potential problems for our models and overall process. The original data from Moneypuck was missing 3 years of regular season home game data for the Arizona Coyotes (2011-2013 seasons), which was the only significant issue we noticed with the set. This may be due to an issue with the Coyotes ownership during those years, as they were briefly owned and controlled by the league and may have been intentionally penalized or unintentionally mishandled in statistics tracking because of it. While it is unfortunate, we could not find a way to include this data, we do not think it significantly hinders our models in any way as we had no year-to-year interactions within our models and these games had no bearing on predicting other games other than potentially swaying the training of models slightly if they held many significant outliers, which is unlikely.

We also must comment on the shortcomings of our process to limit certain variables making it into our models; while we were conservative in removing all Moneypuck-modeled variables and any collinearity in our predictors for all models, there was still a significant interaction between unblockedShotAttemptsAgainst and savedUnblockedShotAttemptsAgainst that slipped through and dominated the accuracy of many models. This can particularly be seen in the dramatic change in performance within our logistic models after these two variables, which individually were not very strong predictors, were combined later in the bottom-up selection. In our exploratory data analysis, we were able to see the clearly strong predicting abilities of goalsFor and goalsAgainst and purposely removed them as predictors in order to challenge ourselves to predict games based on less "luck-based" statistics, but despite these intentional efforts the interaction of the two shot attempt variables gave a rough estimate of goalsAgainst. In future research we think it would be prudent to explore removing one or both of these variables and the effects on model performance it would have.

Applications of the Models

The applications of these models are various across the professional sports and sports media industries. First and foremost, they can give insight to NHL stakeholders (owners, coaches, executives, players, etc) into the potential result of a given game in a season, but also more importantly lend insight towards a "formula" for winning to apply to talent acquisition, player training, and in-game strategy. For example, it is often noted how well a team performs at faceoffs and minimizing penalties (or drawing them), but our models show that other statistics like blocked shot attempts, takeaways, and hits give a model more significant information when predicting which team wins.

Outside of the NHL itself, our models can be used for the purpose of fan interest and entertainment by sports media companies like ESPN, or by fans themselves interested in fantasy sports or sports betting. One potential use of the models could be live betting games, which is a bet on the result of the game or score at some point during the game. Given a game is tied at the end of the second period, it may be insightful to apply the prorated game statistics to the models in order to see which team is playing more like a winning team and potentially has a predicted edge towards getting the win.