Anthony Wilson

DSC 550 Data Mining

Professor Becky Deitenbeck

8/7/2020

Who Controls the Rink?

Introduction

In 2016 I went to my first hockey game and was hooked. I moved to St. Louis in 2017 and caught the hockey fever. During the 2019 Stanley cup championship the St. Louis Blues brought the cup home for the first time in franchise history. During the playoffs, it was rather stressful. When the Blues would be the first on the offensive, my stress level seemed to decrease, and it appeared they were able to control the game. I also noticed the odds of them winning the game increased as well, at least it seemed that way. This piqued my interest, is there something more to this. Teams, that are the first on the offensive, does this increase their odds of winning the game?

I am not a person to bet, but this could be beneficial to those that are. When it comes to predicting outcomes of a game many rely on stats to place their bets. With the way that sports gambling is changing by taking bets not only before the game but also during a game. Something like this could help in making someone's decisions. For me, I am interested in this purely for the sake of understanding the game. I want to dive deeper into the stats available for hockey, and this gave me the opportunity.

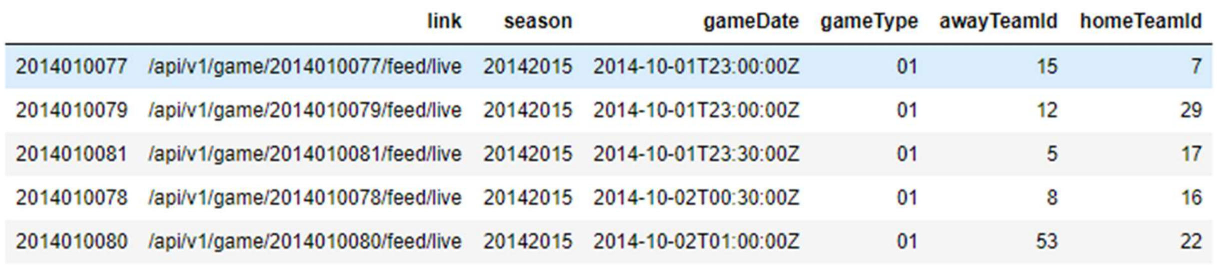
Related Work/Research

In researching how to approach a project like this I came across this post, “Predicting Events during a Hockey Game”, by Steven Ginzberg. (2016). Ginzberg for a final project in his grad program scraped the web for events in hockey games. He then took those events and tried to predict the likelihood of a goal. In it, he used the Naïve-Bayes method and logistical regression. The outcome for him wasn’t all that great. The likelihood of a goal was so low that it was essentially zero. He attempted logistical regression as well but ran out of memory. Some of the other research I found was focused on other classification models.

Based on the data that I collected, it was similar to Ginzberg’s data. I had most hockey events back to 2014. I was not trying to predict a goal, but I was trying to predict the outcome of a game, given all the events from the game. He used the Naïve-Bayes Method, and it seemed to make sense for mine as well. Since I planned to take the data and identify which team was the first at each event by using a Boolean flag for each event. I also used logistic regression like Ginzberg. He used it to get probabilities but ran out of memory. I used logistic regression as well to get the odds of winning. I have a laptop that can handle massive amounts of data. So, I knew that most models that I would run on my computer, it would be able to handle the processing. The only issue that I ran into was the time it would take to run a model. This didn’t happen with the logistic model but with the Support Vector Machines (SVM). So, I went ahead without it.

Data and Experimentation

The data I used was from the undocumented NHL stats API. (NHL 2020) this website provides a substantial amount of data. It has data related to the franchise, teams, players, and events within a game that can be pulled based on schedules or a specific date range. The main address I pulled from was the following API, "http://statsapi.web.nhl.com/api/v1/schedule?&startDate=YYYY-MM-DD&endDate=YYYY-MM-DD" (Hynes 2020) where YYYY-MM-DD represent dates. (Hynes 2020) This data pull will return all games between the start date provided and the end date. The data is returned in JSON format. It provides links to each game's live feed, team information as well as total wins and losses within the season. After I transformed the data, this is a snapshot of what it looked like.



	link	season	gameDate	gameType	awayTeamId	homeTeamId
2014010077	/api/v1/game/2014010077/feed/live	20142015	2014-10-01T23:00:00Z	01	15	7
2014010079	/api/v1/game/2014010079/feed/live	20142015	2014-10-01T23:00:00Z	01	12	29
2014010081	/api/v1/game/2014010081/feed/live	20142015	2014-10-01T23:30:00Z	01	5	17
2014010078	/api/v1/game/2014010078/feed/live	20142015	2014-10-02T00:30:00Z	01	8	16
2014010080	/api/v1/game/2014010080/feed/live	20142015	2014-10-02T01:00:00Z	01	53	22

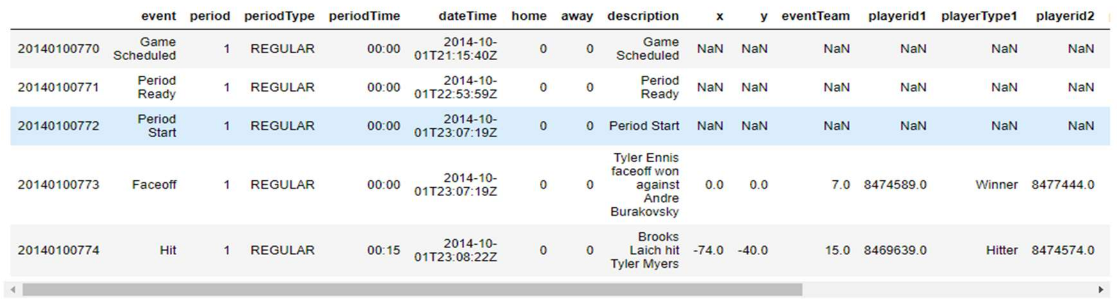
Table 1: Game data from undocumented API

The far-left column is the data frame index, which is the actual game key provided by the NHL API. Then we have the link to the game feed, season, the date of the game, game type (pre-season, regular season, playoffs, or all-star game) then each of the team's unique id from the game. I looped through this data frame and pulled each game's feed.

The game link returns box scores, line scores, team rosters as well as links to each API. The most important information returned from the feed is the events. The events are all actions that take place in a game. Table 2 shows all the events related to this analysis. We have blocked shots, goals, faceoffs, etc… the main events in a game. Other events not used in this table are the game start, period end, etc… anything that doesn't relate to an action a player makes on the ice. The data was setup tall instead of wide. See table 3 for tall data.

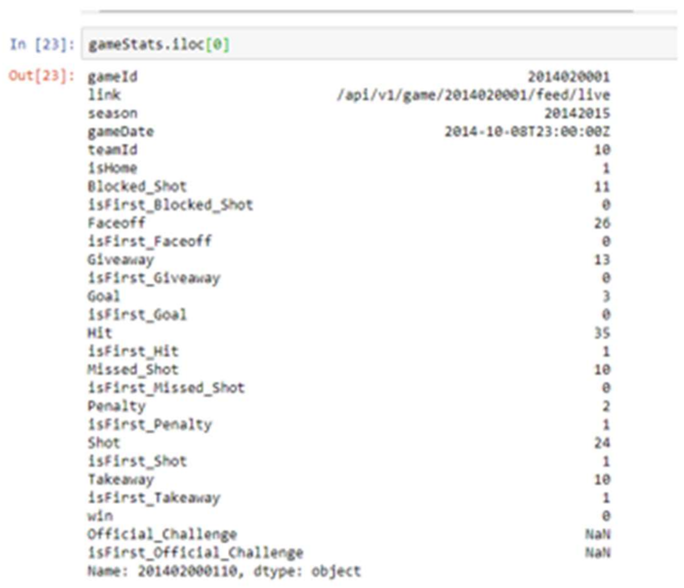
Event	Description
Blocked_Shot	A shot blocked by a skater from the opposite team with their stick or body.
Faceoff	Puck is brought into play by referee dropping the puck in front of two players, the player who gets the puck is rewarded with the faceoff win.
Giveaway	Puck is turned over to the other team in an unforced error.
Goal	Puck goes into the net and the team scores one point.
Hit	Body check to the other team's puck carrier.
Missed_Shot	A shot that is missed and not blocked by the other team and doesn't require a goalie save.
Penalty	Foul committed by a player and they are put in the penalty box.
Shot	Shot by team or player on the goal. Blocked shots and missed shots are not included for this stat.
Takeaway	The player takes the puck from another player on the opposing team.
Official_Challenge	Team challenging a play for review from the referee.

Table 2: Events from the live game feed.



	event	period	periodType	periodTime	dateTime	home	away	description	x	y	eventTeam	playerid1	playerType1	playerid2
20140100770	Game Scheduled	1	REGULAR	00:00	2014-10-01T21:15:40Z	0	0	Game Scheduled	NaN	NaN	NaN	NaN	NaN	NaN
20140100771	Period Ready	1	REGULAR	00:00	2014-10-01T22:53:59Z	0	0	Period Ready	NaN	NaN	NaN	NaN	NaN	NaN
20140100772	Period Start	1	REGULAR	00:00	2014-10-01T23:07:19Z	0	0	Period Start	NaN	NaN	NaN	NaN	NaN	NaN
20140100773	Faceoff	1	REGULAR	00:00	2014-10-01T23:07:19Z	0	0	Tyler Ennis faceoff won against Andre Burakovsky	0.0	0.0	7.0	8474589.0	Winner	8477444.0
20140100774	Hit	1	REGULAR	00:15	2014-10-01T23:08:22Z	0	0	Brooks Laich hit Tyler Myers	-74.0	-40.0	15.0	8469639.0	Hitter	8474574.0

Table 3: Raw data from the live feed



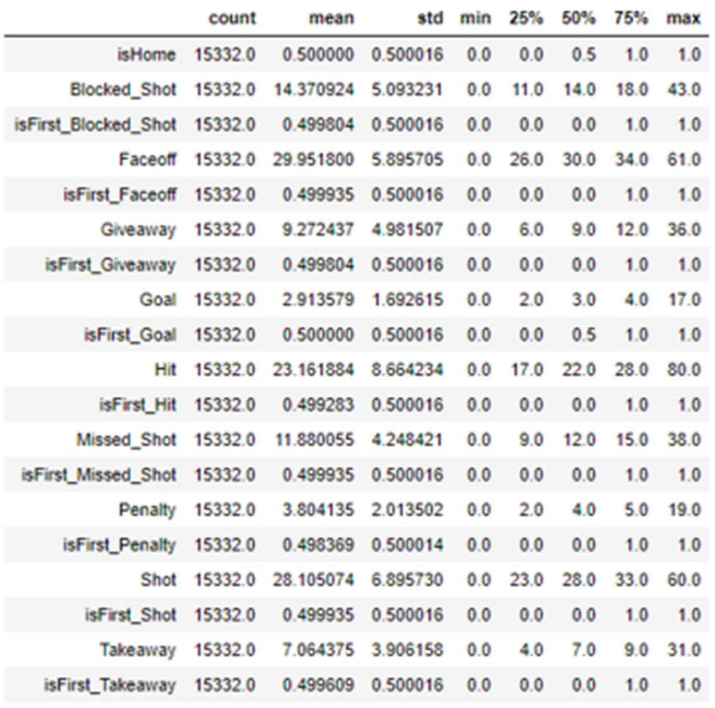
```
In [23]: gameStats.iloc[0]

Out[23]: gameId                                          2014020001
         link                   /api/v1/game/2014020001/feed/live
         season                                            20142015
         gameDate                              2014-10-08T23:00:00Z
         teamId                                                  10
         isHome                                                   1
         Blocked_Shot                                            11
         isFirst_Blocked_Shot                                     0
         Faceoff                                                 26
         isFirst_Faceoff                                          0
         Giveaway                                                13
         isFirst_Giveaway                                         0
         Goal                                                     3
         isFirst_Goal                                             0
         Hit                                                     35
         isFirst_Hit                                              1
         Missed_Shot                                             10
         isFirst_Missed_Shot                                      0
         Penalty                                                  2
         isFirst_Penalty                                          1
         Shot                                                    24
         isFirst_Shot                                             1
         Takeaway                                                10
         isFirst_Takeaway                                         1
         win                                                      0
         Official_Challenge                                     NaN
         isFirst_Official_Challenge                             NaN
         Name: 201402000110, dtype: object
```

Table 4: Example row after transformation

From the tall data, I pivoted it, summed up the total number of events by teams, identified who was the first at each one, and calculated the score to identify the winner of the game. In Table 4, I have one row taken from the transformed tall data set. Notice that it is one team per row. When exploring the data 103 games didn't return event information, so we would be missing 206 rows because each game is split up into two rows for each team. There was a total of 7,666 games that returned event information and 15,332 rows were created in the final table.

In some of the exploratory data analysis, we found that there were, at most, excluding Official_Challenge, 50 null values in a column in the final data set. Most columns within the data had 2-6 that was null. Official_Challenge had 13,556 that had null values. I decided to drop Official_Challenge. For the other columns, with null values, I filled them with zeros, and it shouldn't affect the data too much.

Table 5 shows the descriptive statistics for each column that was created. A couple of things to note, the mean goal is 2.91 with a 50% quartile at 3, 75% at 4, and the max at 80. It seems like the data might be a little skewed, we see this same pattern with most of the data that is not Boolean. I included the Boolean data, anything with "is..." in front of the variable name. Including these values in the descriptive stats was good so we could make sure they were all in populated and were just flagged with zeros and ones. If we were to look at the distribution plots of each event in figure 1, we could see the skewness in the data. Overall, though, the data seems to be close to a normal distribution. Blocked_Shot, giveaway, goal, Hit, missed_shot, penalty, and takeaway all seem to be slightly skewed to the right. The below figure was used to show what is happening with each event and the skewness.



	count	mean	std	min	25%	50%	75%	max
isHome	15332.0	0.500000	0.500016	0.0	0.0	0.5	1.0	1.0
Blocked_Shot	15332.0	14.370924	5.093231	0.0	11.0	14.0	18.0	43.0
isFirst_Blocked_Shot	15332.0	0.499804	0.500016	0.0	0.0	0.0	1.0	1.0
Faceoff	15332.0	29.951800	5.895705	0.0	26.0	30.0	34.0	61.0
isFirst_Faceoff	15332.0	0.499935	0.500016	0.0	0.0	0.0	1.0	1.0
Giveaway	15332.0	9.272437	4.981507	0.0	6.0	9.0	12.0	36.0
isFirst_Giveaway	15332.0	0.499804	0.500016	0.0	0.0	0.0	1.0	1.0
Goal	15332.0	2.913579	1.692615	0.0	2.0	3.0	4.0	17.0
isFirst_Goal	15332.0	0.500000	0.500016	0.0	0.0	0.5	1.0	1.0
Hit	15332.0	23.161884	8.664234	0.0	17.0	22.0	28.0	80.0
isFirst_Hit	15332.0	0.499283	0.500016	0.0	0.0	0.0	1.0	1.0
Missed_Shot	15332.0	11.880055	4.248421	0.0	9.0	12.0	15.0	38.0
isFirst_Missed_Shot	15332.0	0.499935	0.500016	0.0	0.0	0.0	1.0	1.0
Penalty	15332.0	3.804135	2.013502	0.0	2.0	4.0	5.0	19.0
isFirst_Penalty	15332.0	0.498369	0.500014	0.0	0.0	0.0	1.0	1.0
Shot	15332.0	28.105074	6.895730	0.0	23.0	28.0	33.0	60.0
isFirst_Shot	15332.0	0.499935	0.500016	0.0	0.0	0.0	1.0	1.0
Takeaway	15332.0	7.064375	3.906158	0.0	4.0	7.0	9.0	31.0
isFirst_Takeaway	15332.0	0.499609	0.500016	0.0	0.0	0.0	1.0	1.0

Table 5: Descriptive statistics of each of the columns

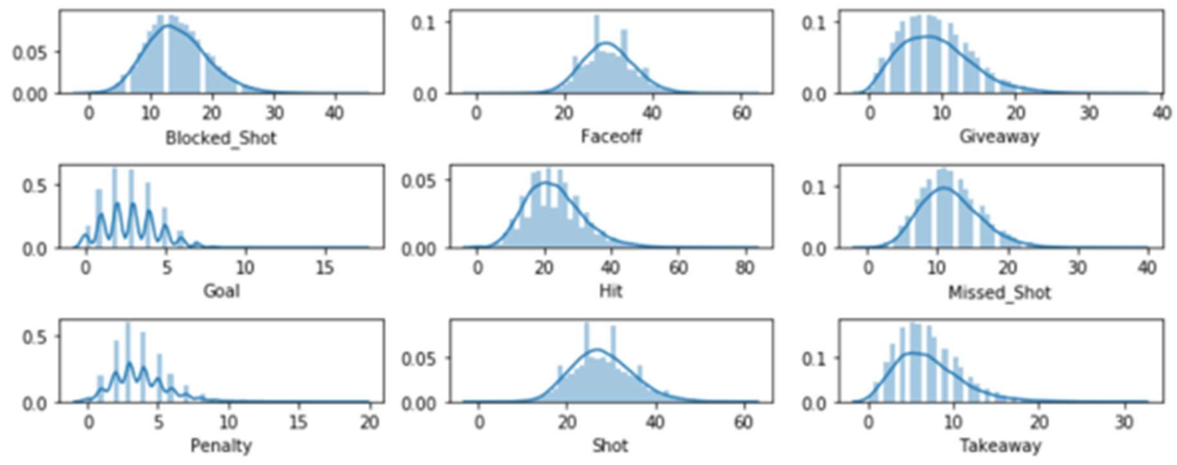
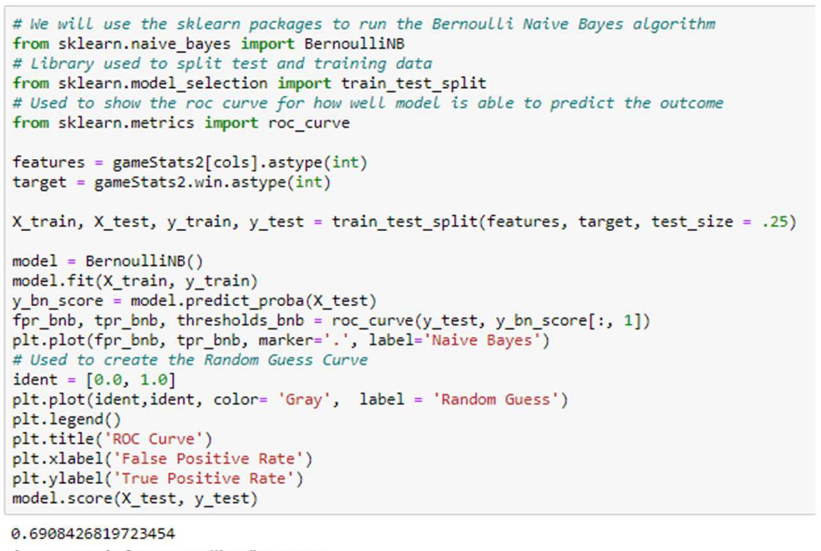


Figure 1: Distribution Plots of Event Data

Solution/Method

For the Naïve-Bayes Method, we used Bernoulli distribution. The Bernoulli was used because we have binary variables that describe the events in the game. We are just looking for the presence of the team being the first at an event. We took the data and split it into training and test data sets. From python, we used the package Scikit Learn to split the data into two separate data sets, test and training. We also used the Scikit Learn package to run the Bernoulli Naïve-Bayes Method along with creating a receiver operating characteristic or ROC curve. A ROC curve is used to measure how accurate a binary classifier is compared to the actual data. It measures the true positive rate (sensitivity) against the false positive rate (specificity). The closer to the top left of the graph, a curve is, the more accurate the results.

For the analysis we are only going to be looking at the columns "is" in front of them for the independent variables. The variable win is used for the dependent variable or what we are trying to predict. We create four data sets total. We have one for our independent training variables and one for our dependent training variables. The two test data sets are set up the same as the two training data sets. Figure 2 shows the code for the Bernoulli Naïve-Bayes Method, along with its model score.



```
# We will use the sklearn packages to run the Bernoulli Naive Bayes algorithm
from sklearn.naive_bayes import BernoulliNB
# Library used to split test and training data
from sklearn.model_selection import train_test_split
# Used to show the roc curve for how well model is able to predict the outcome
from sklearn.metrics import roc_curve

features = gameStats2[cols].astype(int)
target = gameStats2.win.astype(int)

X_train, X_test, y_train, y_test = train_test_split(features, target, test_size = .25)

model = BernoulliNB()
model.fit(X_train, y_train)
y_bn_score = model.predict_proba(X_test)
fpr_bnb, tpr_bnb, thresholds_bnb = roc_curve(y_test, y_bn_score[:, 1])
plt.plot(fpr_bnb, tpr_bnb, marker='.', label='Naive Bayes')
# Used to create the Random Guess Curve
ident = [0.0, 1.0]
plt.plot(ident,ident, color= 'Gray',  label = 'Random Guess')
plt.legend()
plt.title('ROC Curve')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
model.score(X_test, y_test)
```

```
0.6908426819723454
```

Figure 2: Code for Bernoulli Naïve-Bayes

The model score measures the same thing as

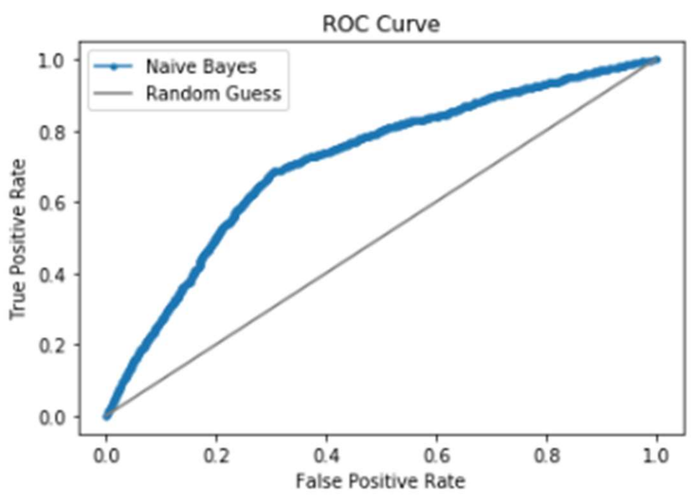
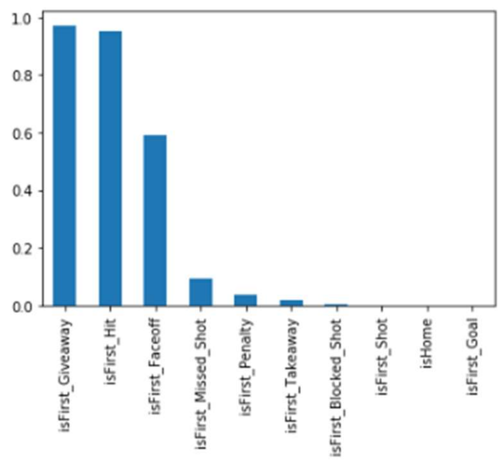


Figure 3: ROC curve for Bernoulli Naïve-Bayes Method

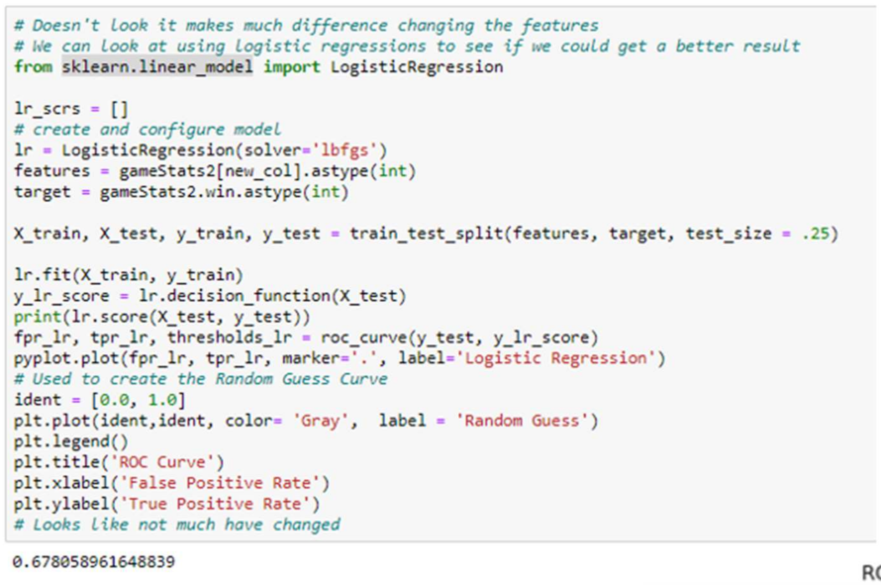
the ROC curve. It measures the model's accuracy. In figure 3, we can see the ROC curve for this model. Notice that we are on the left side of the random guess. The line "Random Guess" represents on avg if someone was to guess win/lose for a game without any data, they are bound to get it right 50% of the time. In this case, the model is right 69% of the time, based on the data that was pulled.

For the Logistic regression, I also used the Scikit Learn package. Before diving straight into the regression, the Chi-Square test for independence was used to reduce the number of variables within the model. I wanted to reduce the dependency of feature variables upon each other. This issue is more prevalent in a regression model compared to the Naïve-Bayes where it can handle variables that might have some dependencies. Figure 4 shows a bar chart with the p-values from the Chi-Square test. The goal is to have a small p-value, less than .01. The variables we will use based on a small p-value are isFirst_Blocked_Shot, isFirst_Shot, isHome, and isFirst_Goal. Figure 5 is the code for the Logistic Regression. It is set up like the Bernoulli code from figure 2. We split the data the same, create a ROC curve, and calculate the model score. In figure 6 we show the ROC curve for logistic regression. The model score for the logistic regression was 68% accurate based on the data that was pulled. Which is still better than a "Random Guess".



```
isFirst_Giveaway        0.97266
isFirst_Hit             0.95443
isFirst_Faceoff         0.59138
isFirst_Missed_Shot     0.09314
isFirst_Penalty         0.03431
isFirst_Takeaway        0.01747
isFirst_Blocked_Shot    0.00445
isFirst_Shot            0.00022
isHome                  0.00000
isFirst_Goal            0.00000
dtype: object
```

Figure 4: Chi-Square Test for Independence P-values



```python
# Doesn't look it makes much difference changing the features
# We can look at using logistic regressions to see if we could get a better result
from sklearn.linear_model import LogisticRegression

lr_scrs = []
# create and configure model
lr = LogisticRegression(solver='lbfgs')
features = gameStats2[new_col].astype(int)
target = gameStats2.win.astype(int)

X_train, X_test, y_train, y_test = train_test_split(features, target, test_size = .25)

lr.fit(X_train, y_train)
y_lr_score = lr.decision_function(X_test)
print(lr.score(X_test, y_test))
fpr_lr, tpr_lr, thresholds_lr = roc_curve(y_test, y_lr_score)
pyplot.plot(fpr_lr, tpr_lr, marker='.', label='Logistic Regression')
# Used to create the Random Guess Curve
ident = [0.0, 1.0]
plt.plot(ident,ident, color= 'Gray',  label = 'Random Guess')
plt.legend()
plt.title('ROC Curve')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
# Looks like not much have changed
```

```
0.678058961648839
```

Figure 5: Code for Logistic Regression

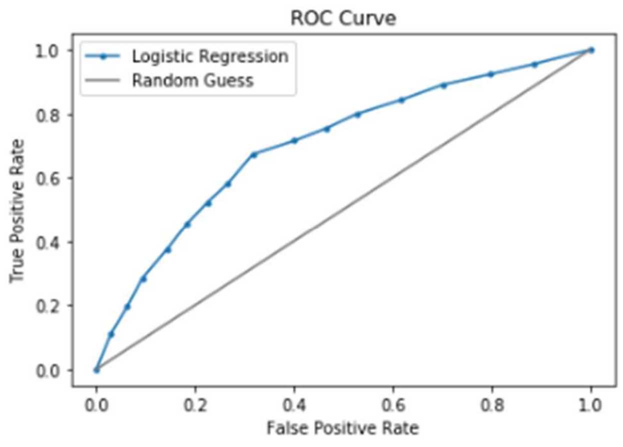


Figure 6: ROC curve for Logistic Regression

Evaluation and Results

Overall the results seem to be good, the models are about 20% better than a random guess. Both models produce a very similar result. The graphs seem to align as well. After running several experiments, the models don't change their results. We can say based on this model and given the data provided it is right about 69% of the time.

Some of the enhancements for this data could include a win/loss ratio for each team during a given season. The win/loss ratio would have to be based on all prior games within a season. This could potentially boost the accuracy by knowing how well the team performs relative to other teams. Looking at the average number of events the team has per season. It would need to be a rolling average by season. We could also take the data and add periods in there for each game. Each period would have a flag for who was the first at each event. We could even create a model to determine if the odds of a team winning at the time of each event for example the first goal is scored in a live game, what are the odds that team wins the game?

Some other feature enhancements outside of team stats could be based on the players. The models used in this analysis assume all teams are equal, but they are not. Teams are constantly trading players and players may age, get injured, or retire. Also, each team has its strengths and weaknesses based on their players. Individual stats based on players could help with the analysis. Such things like the player's event stats per game, their win percentage, how well they do against the other players on the other team, and how often they are injured.

For the accuracy of this model, it seems reasonable based on this data set, more analysis should be done to be more confident in the output. I think we need to look at the data more in-depth. Look at each season, dive deeper into the games, pull in the players, and look at each team's data at the time of an event during a game. Treat each game as if it is their first.

Sources

Ginzberg, S. (2016, July 2). Predicting Events During a Hockey Game. Retrieved July 31, 2020, from https://nycdatascience.com/blog/student-works/predicting-events-hockey-game/

Ginzberg, S. (2016, June 27). Nycdatasci / bootcamp005_project. Retrieved July 31, 2020, from https://github.com/nycdatasci/bootcamp005_project/tree/master/Project5-Capstone/Project%20NHL

Hynes, D. (2020, April 4). Stats-api.md · master · Drew Hynes / nhlapi. Retrieved June 21, 2020, from https://gitlab.com/dword4/nhlapi/-/blob/master/stats-api.md

NHL. (2020). Https://statsapi.web.nhl.com/api/v1. Retrieved June 21, 2020, from https://statsapi.web.nhl.com/api/v1/schedule