An Application of Modern Statistical Learning Techniques to National Hockey League Data

Michael Feron [*1], Harrison Magee [†1], and Justin Nichols [‡1]

[1]Colorado School of Mines
Department of Applied Mathematics and Statistics

Abstract

In this paper, we explore the overlap between hockey statistics and player contract data by using statistical learning methods, including regression and regularization techniques, to predict the salary of a player. In addition, we employ classification methods to predict the position of players. We present rigorous selection methods based on many criteria and cross-validation for both model types. Several different models are constructed using a variety of methods, each with their own strengths and weaknesses. We then address how models of salary and position might be used together to evaluate and select players with respect to their potential value on a team as well as their value in the NHL marketplace.

[*]mjferon@mymail.mines.edu
[†]hmagee@mymail.mines.edu
[‡]jnichols2@mymail.mines.edu

Contents

List of Figures

List of Tables

1 Introduction

Professional sports often provide very interesting big data sets and even more interesting questions associated with them. When information about salary is introduced, the problem becomes an even more interesting, self-contained system modeling an economics problem. We can think of the players like resources and the teams like companies in competition with one another. Typically, a company will do everything it can to purchase and invest in only the combination of resources it needs at the cheapest available market price. Companies that do this better than others are at a market advantage and outperform their competition. Our group has chosen to evaluate datasets taken from both Thomas and "NHL Team Contracts & Payrolls" pertaining to the National Hockey League (NHL). These data have features like goals, assists, points, time on ice, faceoffs won, +/-, etc. and can be found thoroughly explained in the Appendix.

1.1 Research Question

We are interested in the information that can be gleaned from evaluating contract data with respect to in-game statistics of the NHL, and more specifically how teams could more accurately value a player, and thus, take advantage of their over or under valuation. Additionally, we are interested in learning how classification can help us predict the position of a player based on his stats. Most importantly, we concern ourselves with how this information can be of use to a National Hockey League team that chooses to evaluate it.

2 Data

The first action we took with our data was the cleaning. We removed any null values and adjusted errors in overlapping datasets – we collected data on both salary and statistics, and the overlap in the two consisted of mainly two problems: retired/demoted players and misspellings. Misspellings was an error that required careful combing of both datasets to correct, but correcting issues on retired/demoted players would provide more of a challenge. We had data on the 2018-2019 season, but 2019-2020 contract data, so anyone retired or now in the AHL did not appear in our salary dataset. This problem could have been solved by purchasing a proprietary dataset, but we had plenty of data left over (around 300 players) to continue our analysis. Additionally, we felt that a players 2018-2019 performance was the cause of their salary values the following year, i.e. current salary data, so we believe this salary data being one year ahead of the statistical data accurately captures the regressor/response relationship we seek.

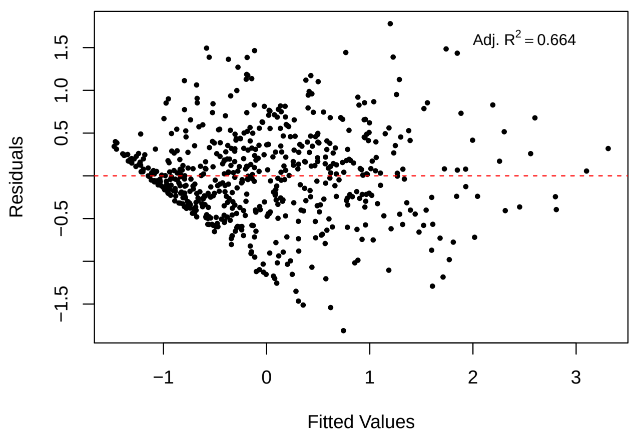
We then constructed the dataframe with a simple left join, then fit a simple linear model to remove linear dependencies. When linear dependencies are present, regression does not work properly since the matrix algebra required to perform regression handles low rank poorly. To do this, we found the first instance of a dependency, removed that regressor, and re-fit the model. We repeated this process until we achieved a model that contained no such dependencies. m Many such dependencies were found since many of our statistics are linear combinations of the others. For example, a linear model predicting goals, given all statistics including points and assists, would say simply take a players points and subtract the assists and set all other predictors to be zero. These dependencies were crucial to eliminate if we want a reliable model of any value.

We will predict salary from the remaining regressors, and to do this, we will use various statistical techniques to do so.

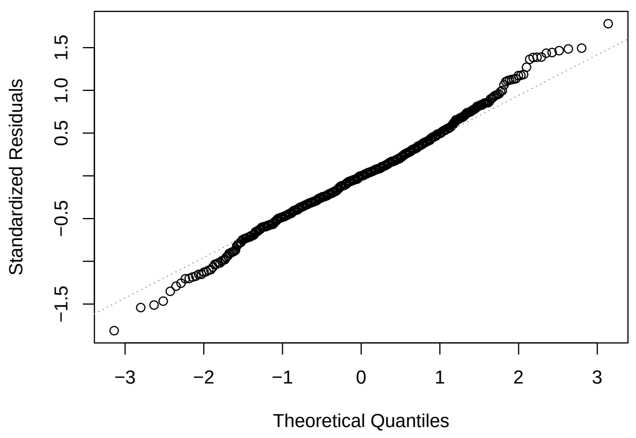
3 Salary Models

3.1 Linear Regression

We begin our assessment of how well salary is predicted by isolating our quantitative data, to be used later in classification. After removal, we are left with our response variable `Salary` and 89 other measurable variables. In order to get a starting glance out how this data will predict salary, we generate a linear model. By implementing multiple regression, we assume that the errors are normally distributed and the variance is constant, therefore there is no multicollinearity and the data is homoscedastic. This approach provides us with an idea of how well the data will predict salary, and alert of us if any of our assumptions are being violated. At this point, we anticipate a generally well fit model, due to a remaining excess of variables. The following indicates a residual plot for the resulting linear model, as well as an associated Q-Q plot. For reference to the code see Appendix B.1



(a) Residual plot for salary predictions based on all quantitative variables.



(b) Normal Q-Q Plot

Figure 1: Simple Linear Model

We see in Figure 1a that we may have non-constant variance, which would require a variance stabilizing transform, addressed in the Appendix; however, since the data consists of a large number of variables and observations, we expect the model to over-fit and have problems at this stage. This effect stems from low bias caused by the large number of variables in addition to high variance. Additionally, we see a downward sloping pattern toward the beginning potions of our fitted values. It looks to be indicating that the salaries in the lower portion of the league are harder to predict than those in the medium range onward. This would most likely come from the less predictable evolution of a player who has been in the league for fewer years. More specifically, this could be due to a rookie player initially making less with the chance of performing well and flourishing or losing momentum. However, the other half of the salaries in the residual plot seems to have perfect scattering hinting that our model would do well in this region.

Unfortunately, our Normal Q-Q plot in Figure 1b isn't quite what we would hope for, moreover we see in Figure 1b that numerous points are lying off the ideal line. This is somewhat worrisome, since we assume that our errors are normally distributed in order to proficiently use multiple regression. However, this plot implies that the errors we expect to be normally distributed may not be. Thus, we intend to mitigate this issue by implementing subset selections and shrinkage methods, hoping that this will eliminate numerous irrelevant variables that could be distracting in understanding the pertinent influences in our data, as well as in making our salary predictions.

3.2 Stepwise Selection

The first method we employ to reduce our model size is forward selection. We choose this over best subsets because the data set is much too large to exhaustively perform best subsets. More importantly, we wanted to capture only useful regressors that would leave us with better accuracy or bias. We start by using the goodness-of-fit statistics Mallows' C_p and the Bayesian Information Criterion (BIC) as indicators for how well forward selection performed on each model.

This was done by calculating both of the statistics for each model. As we know, larger values of R^2 indicate better fitting models, but for these other two, the best model is selected by the model in the collection with the minimum value of the statistic. As a result, from using the C_p values, we get the following variables and their respective coefficient values. This code is viewable in Appendix B.2

Variable Name	(Intercept)	P/GP	PPG	PPP	S%	TOI/GP	FOW
Coefficient Value	≈ 0	0.58	0.14	-0.33	-0.10	-0.57	-1.04
Variable Name	EV FOW	OZ FOW	NZ FO	On-Ice PP GF	SH TOI/GP	On-Ice PP GA	EV TOI/GP
Coefficient Value	1.43	0.28	-0.65	-0.11	0.26	0.17	0.56
Variable Name	On-Ice EV GF%	TkA/60	MsS Wide	MsS Post	SHA1	SHA1/60	SH iSAT/60
Coefficient Value	-0.03	0.07	-0.08	0.08	-0.10	0.07	0.07
Variable Name	SH S/60	PPA1	PP Shots	PPA1/60	PP TOI%	Ht	Wt
Coefficient Value	-0.12	0.22	0.18	-0.05	0.38	0.05	0.04

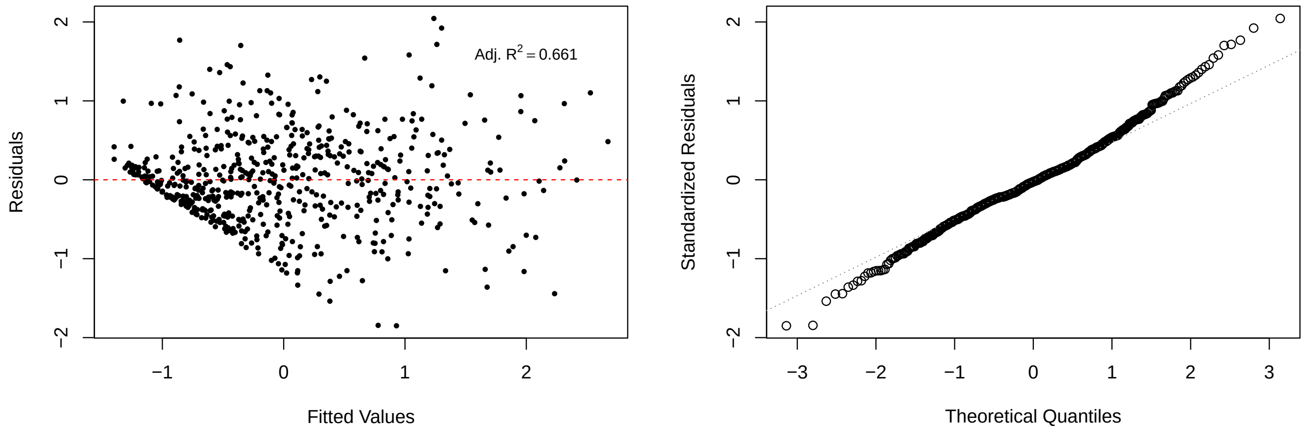
Table 1: Model selection based on C_p values

On the other hand, the model chosen by the BIC values is:

Variable Name	(Intercept)	P/GP	TOI/GP	FOW	EV FOW	OZ FOW	NZ FO
Coefficient Value	≈ 0	0.43	0.38	-1.04	1.42	0.26	-0.60
Variable Name	On-Ice PP GA	EV TOI/GP	MsS Post	SH S/60	PP Shots	Wt	
Coefficient Value	0.14	-0.14	0.07	-0.06	0.14	0.07	

Table 2: Model selection based on BIC values

As we should expect from standardizing the data, the intercept value coming back from both models is zero. Since we set out to minimize the number of variables our prediction model uses, of these two, we selected BIC. This statistical value tends to select the smaller models, therefore satisfying our objective. In order to get a better assessment of improvements, if any, of the model, we once again create residual and Normal Q-Q plots. However rather than using the entire 89 variables, we do so with the 12 variables listed in Table 2 that BIC chose for this model. The resulting plots follow:



(a) Residual plot for salary predictions based on variables in Table 2

(b) Normal Q-Q Plot

Figure 2: Forward selection model determined by BIC

We see once again with residual plot in Figure 2a that the peculiar sloping shape we saw previously in the lower regions of salary, is persisting. Consequently, we cannot conclude that this method will eliminate the concerns we have about the model being heteroscedastic; in other words, our assumption that the variance is constant may be broken. We see that this model struggles on the extremes having fat tails, or where we see the points dipping and rising heavily.

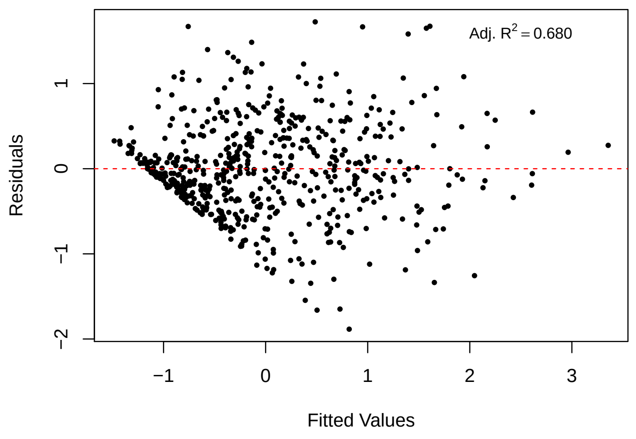
Ultimately, this approach provides us with a balance between a good fit and less complexity; however, since our goal is to make predictions, we must also maintain a predictive accuracy. This is best done by cross-validation as it will mimic the idea of receiving new information in order to make an accurate prediction. More specifically, this method will provide us with the a model from the collection that minimizes the mean squared error of prediction (MSEP).

Once again, we do forward selection on the data, but rather than using one of the goodness-of-fit statistics, we will use k-fold cross-validation. This process will divide the data randomly into k roughly equal parts. Next, a fold is held out for use in imitating a test set, and a model is chosen from the remaining folds of data. For each model, predictions are made based on the training set and then compared to the actual values. This gives us the ability to calculate the MSEP for each model, and determine the best fit based on the minimum of this value. We opted to use this method rather than leave-one-out, because it is computationally heavy as more variables are included. Since we have a large number of observations, we will set our folds to $k = 10$. The code for cross-validation will be found in Appendix B.2 From the collection of models, 10-fold cross-validation determined a model of

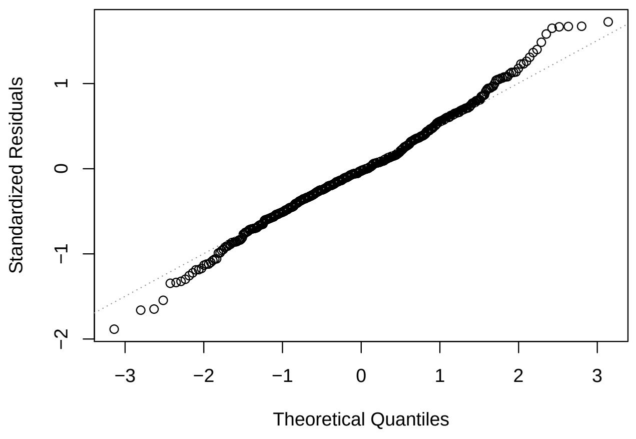
Variable Name	`(Intercept)`	`GP`	`P/GP`	`PPG`	`PPP`	`S%`	`TOI/GP`	`FOW`	`EV FOW`	`OZ FOW`
Coefficient Value	≈ 0	0.04	0.60	0.15	-0.32	-0.12	-0.69	-0.99	1.41	0.27
Variable Name	`NZ FO`	`On-Ice PP GF`	`SH TOI/GP`	`On-Ice PP GA`	`EV TOI/GP`	`On-Ice EV GF%`	`TkA/60`	`MsS Wide`	`MsS Post`	`SHA1`
Coefficient Value	-0.66	-0.13	0.51	0.15	0.65	-0.04	0.07	-0.10	0.07	-0.10
Variable Name	`SHA1/60`	`SH iSAT/60`	`SH S/60`	`SH TOI%`	`PPA1`	`PP Shots`	`PPA1/60`	`PP TOI%`	`Ht`	`Wt`
Coefficient Value	0.08	0.07	-0.12	-0.22	0.22	0.18	-0.06	0.42	0.05	0.04

Table 3: Model selection based on 10-fold cross-validation

Now, in comparison to the previous two selected models, the 27 variable C_p and the 12 variable BIC, this one retains more variables. This model also includes all of the variables from the one that the C_p has, but a few extra. The values of the coefficients are also very similar. We now look at the residual and Q-Q plots to get a better visual comparison;



(a) Residual plot for salary predictions based on variables in Table 3



(b) Normal Q-Q Plot

Figure 3: Forward selection model determined by 10-fold cross-validation

One noticeable change we see with this model is the improved value of the Adjusted R^2. This tells us that not only was cross-validation able to eliminate variables, it was able to efficiently do so in a way that resulted in an improvement in the fit. Once again, we see the same residual plot as the previous two models, but this time the Normal Q-Q plot looks to be a bit more tightly bound to the line. We haven't been able to eliminate our issue with the variance issues with low salaries, but we have been able to lessen the variables, improve the fit, and produce residuals that follow a better normal distribution.

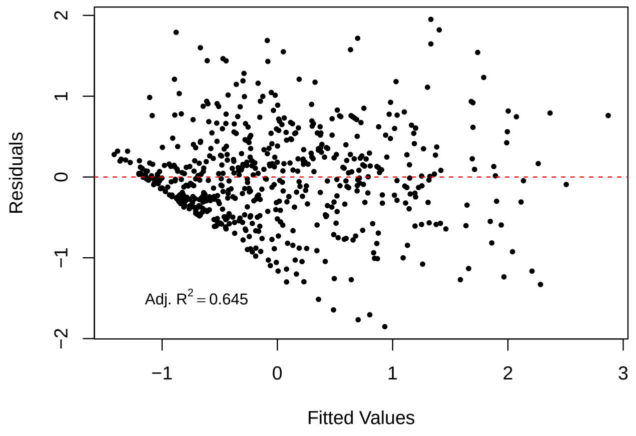
3.3 Least Absolute Shrinkage and Selection Operator (LASSO)

Following from our main goal being to eliminate variables, we decided to use the LASSO method initially, because of the way it pushes some variables towards zero, while forcing others to be identically zero. It's these variables that we can eliminate, thus supplying us with coefficients with less magnitude as well as minimising the amount of variables in the model. By using this method, we will increase the bias with the intention of gaining a reduction in the MSEP. Using the LASSO method and choosing the tuning parameter with cross-validation, which is further detailed in Appendix B.3, we are provided the following model:

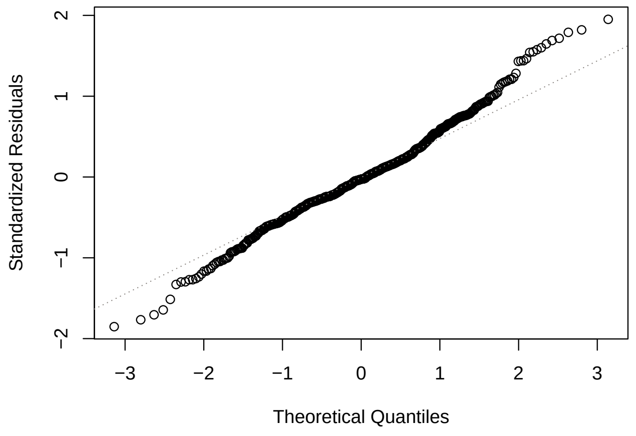
Variable Name	`(Intercept)`	`P/GP`	`OTG`	`TOI/GP`	`OZ FOW`	`On-Ice PP GA`	`On-Ice EV GA`
Coefficient Value	-1.35×10^{-16}	3.72×10^{-1}	3.66×10^{-3}	2.42×10^{-1}	5.17×10^{-2}	8.11×10^{-2}	3.60×10^{-5}
Variable Name	`On-Ice EV GF%`	`TkA`	`ENA`	`MsS Over`	`MsS Post`	`Match`	`SHA1/60`
Coefficient Value	-8.22×10^{-3}	3.43×10^{-2}	8.78×10^{-3}	2.55×10^{-2}	3.27×10^{-2}	5.98×10^{-3}	1.00×10^{-3}
Variable Name	`SH S/60`	`PPA1`	`PP Shots`	`PP S/60`	`PP TOI%`	`Ht`	`Wt`
Coefficient Value	-3.98×10^{-2}	2.27×10^{-2}	1.05×10^{-1}	4.92×10^{-3}	5.10×10^{-2}	4.12×10^{-3}	4.82×10^{-2}

Table 4: Model selection based on the LASSO

As expected, the amount of variables has significantly decreased, while the coefficient values have gone down by many degrees of magnitude. From cross-validation, we get a tuning parameter of $\lambda = 0.02535364$. Once again, to better visualize the differences in this method, we observe the residual and Q-Q plots:



(a) Residual plot for salary predictions based on variables in Table 4



(b) Normal Q-Q Plot

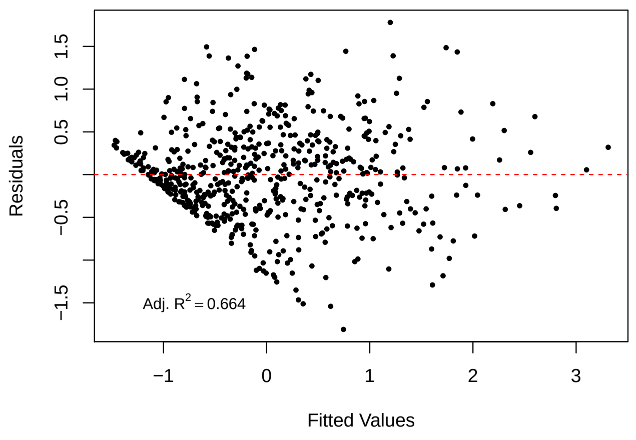
Figure 4: The model produced by the LASSO method

Even though this method has successively achieved a model with less variables, it still doesn't provide any improvements to the models seen previously. The Adjusted R^2 has gone down from our original linear model, and the fat tails are only getting more plump.

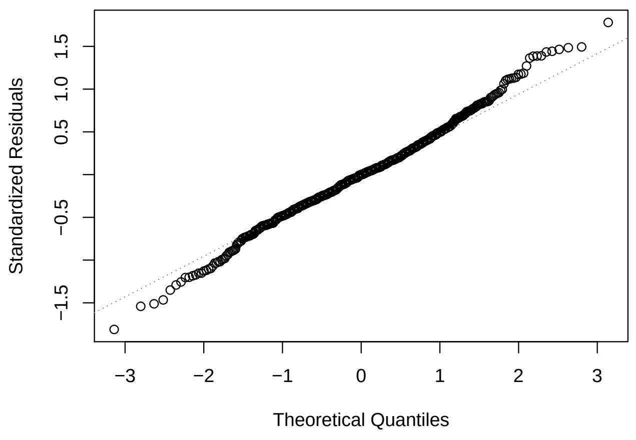
3.4 Ridge Regression

Next, at an attempt to remove the bias in our variables so that we can improve its prediction capability on new data, we implement ridge regression. This will also alleviate the possibility of our predictor variables being collinear. Contrary to the LASSO method, ridge regression will push the values of the coefficients towards zero, but it won't eliminate any of them. This will ultimately reduce the complexity of our model, but not entirely rid it of what seems to be all relevant varaibles. Due to ridge regression only reducing the magnitude of the coefficient values, they can be found in Appendix B.4

After a model was determined by ridge regression, we produced our comparison plots that follow:



(a) Residual plot for salary predictions based on variables in



(b) Normal Q-Q Plot

Figure 5: The model produced by Ridge Regression

This residual plot and the Normal Q-Q plot found in Figure 5 look to be very similar to that of our linear model we started with. The Adjusted R^2 value produced here matches the linear model, and the same patter is occuring in the tail of salaries and on the extremes of the Normal Q-Q plot. This indicates that this method was also incapable of providing us a solution to the breaking of our assumed constant variance.

After all of these models were generated, one so called best must be chosen. Since we set out to minimize the variables used in our model, while maintaining the same prediction accuracy as using the entire data set, we selected our cross-validation model. This model was not only able to cut the predictor variables down to a third of their original amount, it was about to produce a better Adjusted R^2, and wound itself closer to the ideal line in the Normal Q-Q plot. We can't ignore the fact that there is an issue in our variance amongst low level salaries. As mentioned, this most likely comes from that area being the minimum.

4 Position Models

To establish a relationship between player position and the covariates in our dataset, we turn to classification. Classification is, for lack of a better epithet, simply regression for a qualitative response. It amounts to constructing a method to accurately categorize observations from a dataset into classes based on the values of their predictors. In this case, the class of each observation in the data is known beforehand.

Hockey players can be divided into four distinct positions: center-forwards, left-forwards, right-forwards, and defensemen. We will frequently refer to these positions as center, left, right, and defense, respectively. There are also goaltenders, but that is beyond the scope of this report. Defensemen are responsible for preventing their opponent from scoring, and forwards are responsible for scoring for their own team. An accurate classification model for player position could allow teams, for instance, to more accurately assess the type of qualities to seek out in potential players of a particular position.

In the analysis to follow, we will make frequent use of cross-validation. In general, it can be difficult to assess how a classification algorithm might perform when applied to data on which it was not trained. Cross-validation is a critical tool in model evaluation, and allows for an approximation of a model's efficacy by simulation of its performance on "new data."

The dataset we use for the remainder of this paper is an expanded version of the one used for regression. It includes 10 years of data: data from the 2009-2010 season through the 2018-2019 season. As such, each observation represents one player's statistics in one particular year, or a *player-year*. Obviously, this means that many players will be represented by multiple observations. For this reason, we must make an important, yet potentially unrealistic, assumption. In order to maintain validity in the models to follow, we will assume that every observation is independent of one another. This is certainly untrue, as an individual hockey player's performance is not likely to change substantially from year to year. Nonetheless, we hold to this assumption for the remainder of the paper.

Below we present several different models for player position, each utilizing a different classification method.

4.1 Linear Discriminant Analysis

Linear discriminant analysis, or LDA, is a simple, powerful implementation of the Bayes classifier. It combines the observed distribution of a dataset's covariates with assumed prior probabilities – often empirically estimated – to classify observations. The resulting classifier constructs linear decision boundaries to separate groups. Here, we make the assumption that each group – each position – is distributed multivariate normal with common variance-covariance. This is the key assumption in LDA.

LDA is performed on our dataset using empirical prior probabilities: 29% centers, 35% defensemen, 19% left-forwards, and 17% right-forwards. To assess model performance, we run 10-fold cross-validation to estimate the misclassification rate. The code to implement this can be found in Appendix C.1. Figure 5 shows the resultant confusion matrix.

	Prediction			
Truth	Center	Defense	Left	Right
Center	1460	1	259	141
Defense	0	2262	7	9
Left	104	5	668	424
Right	81	4	550	475

Table 5: LDA confusion matrix

This is a fairly well-performing classification method, with an estimated misclassification rate of 0.25. Looking at in-group misclassification rates can provide more insight into this model's performance. Below is a more detailed summary of the results of cross-validation. Table 6 contains in-group misclassification rates as determine by cross-validation.

Group	Center	Defense	Left	Right	**Overall Rate**
Rate	0.11	0.004	0.55	0.55	0.25

Table 6: LDA misclassification rates

This model performs exceptionally well in its classification of defensive players, with a misclassification rate less than half of a percent. Its classification of center-forwards is also outstanding. Unfortunately, the model struggles in its classification of left- and right-forwards, indicating that features for these two positions are likely very similar. Later in this paper, we will group these two positions together to get a better sense of our ability to distinguish between defensemen, centers, and other forwards.

4.2 Quadratic Discriminant Analysis

Quadratic discriminant analysis, or QDA, is very similar to linear discriminant analysis. We again assume that each group is multivariate normal, but we allow each group to have its own variance-covariance, which amounts to a much more realistic assumption. The result is a classification function that is quadratic, rather than linear, in the input variables. While LDA produces linear decision boundaries, QDA produces parabolic decision boundaries. In theory, this gives QDA the freedom to construct boundaries with more complex structure, which ideally would produce better results. This is not always the case, as we will show, as LDA is frequently the better option.

We again use empirical probabilities to perform quadratic discriminant analysis, and the model is assessed using 10-fold cross validation. The code for this implementation can be found in Appendix C.1. Table 7 depicts the confusion matrix.

	Prediction			
Truth	Center	Defense	Left	Right
Center	1467	5	78	311
Defense	6	2237	13	22
Left	210	17	187	787
Right	175	21	111	803

Table 7: QDA confusion matrix

In this case, the estimated misclassification rate is 0.27. This model is similar in performance to LDA, with only a moderately elevated misclassification rate. Below are the in-group misclassification rates.

Group	Center	Defense	Left	Right	**Overall Rate**
Rate	0.21	0.02	0.52	0.58	0.27

Table 8: QDA misclassification rates

Again, the in-group results are about the same as in LDA. We see a slight reduction in the misclassification rate for left-forwards. Otherwise, each in-group misclassification rate is higher, with the rate for center-forwards noticeably so.

Although QDA allows each group to co-vary independently, it does not yield very impressive results. This model offers nothing in the way of improvement over the previous model. It is fair to conclude that linear discriminant analysis produces a simpler, more accurate, more effective model. In addition, the linear decision boundaries of LDA are likely to have less variance in their predictions than QDA. This may be the very reason LDA performs better under the scrutiny of cross-validation.

4.3 Logistic Regression

Our next model is constructed using logistic regression. Logistic regression is a very common classification method. It is particularly useful in its ability to model probabilities. Logistic regression is a numerical method – with no closed-form solution – that involves the numerical approximation of a maximum likelihood estimator.

In performing logistic regression, each observation is assigned a probability of belonging to a particular class; in this case, one of center, defense, left, or forward. We assign each observation to the class with maximum predicted probability. To accurately assess the performance of this model, we again turn to 10-fold cross-validation. The relevant code can be found in Appendix C.2. Figure 9 depicts the confusion matrix.

	Prediction			
Truth	Center	Defense	Left	Right
Center	1502	25	161	173
Defense	4	2242	20	12
Left	188	50	503	460
Right	155	43	434	478

Table 9: Logistic Regression confusion matrix

Logistic regression results in an estimated misclassification rate of 0.27. Table 10 contains in-group misclassification rates.

Group	Center	Defense	Left	Right	**Overall Rate**
Rate	0.19	0.05	0.55	0.57	0.27

Table 10: Logistic Regression misclassification rates

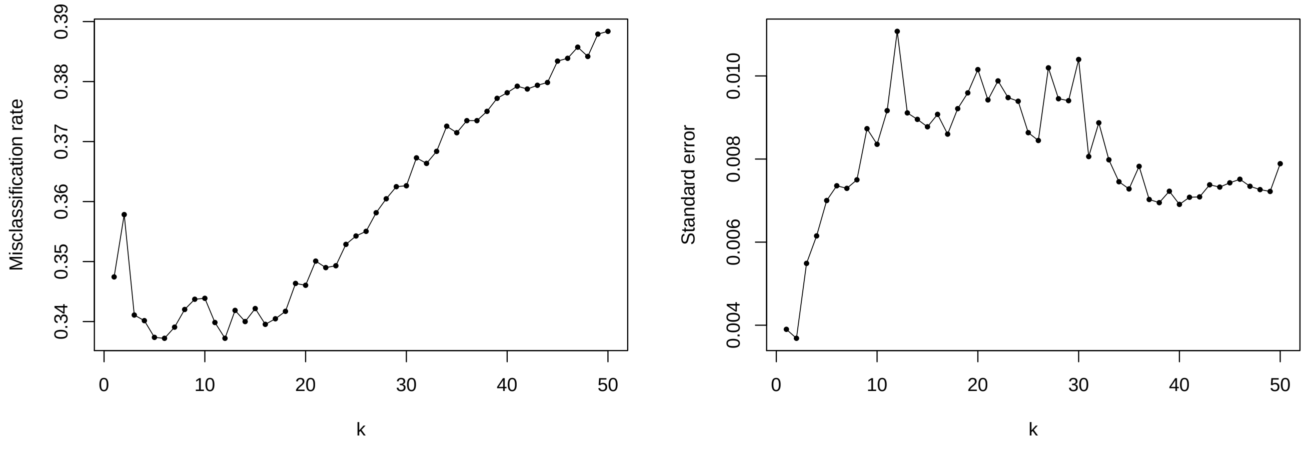
The results are again very similar to both of the previous models, particularly quadratic discriminant analysis. Classification of defensemen and centers is accurate, while classification of left- and right-forwards is inconclusive. There is no compelling reason to select this model over LDA.

4.4 *k*-Nearest Neighbors

This model for classification of player position uses one of the most common, intuitive, and frequently effective classification methods, k-nearest neighbors. This method is markedly different than either discriminant analyses or logistic regression. It is a non-parametric method which uses a "distance" metric to classify observations based on the assumption that observations in a given group are likely to be close to other members of the same group.

The value of k serves as a tuning parameter, and represents the number of nearest observations to use for classification. Given a particular observation, the k observations closest in distance – generally Euclidean distance – are used to determine to which group to assign the observation of interest. The observation is assigned to the group with majority representation among these k-nearest neighbors. Generally, as $k \to n$, where n is the number of observations, *all* data tends towards assignment to the most common group within the training set. As such, careful selection of this tuning parameter is necessary.

We use 10-fold cross-validation to estimate the misclassification rate for each value of k, for $k = 1, \ldots, 50$. Figure 6 depicts this estimated misclassification rate for each value of k. Alongside this plot is a plot of each measurement's associated standard error. The relevant code for k-nearest neighbors with 10-fold cross-validation can be found in Appendix C.4.



(a) Cross-validation misclassification rate for each k

(b) Associated standard error

Figure 6: k-Nearest Neighbors cross-validation results

The estimated misclassification rate is minimized at $k = 6$. However, values for k ranging from 3 to about 16 perform similarly. With a standard error of 0.007, any of these values for k are within about one standard deviation of the minimized misclassification rate, and will likely produce models of similar efficacy.

We choose to let $k = 6$, and re-perform cross-validation to produce our k-nearest neighbors model of choice. The relevant code can be found in Appendix C.3. Below is the confusion matrix.

	Prediction			
Truth	Center	Defense	Left	Right
Center	1460	77	161	163
Defense	10	2014	142	112
Left	146	265	387	403
Right	126	207	381	396

Table 11: k-Nearest Neighbors confusion matrix

This misclassification rate is 0.34. Table 12 contains in-group misclassification rates.

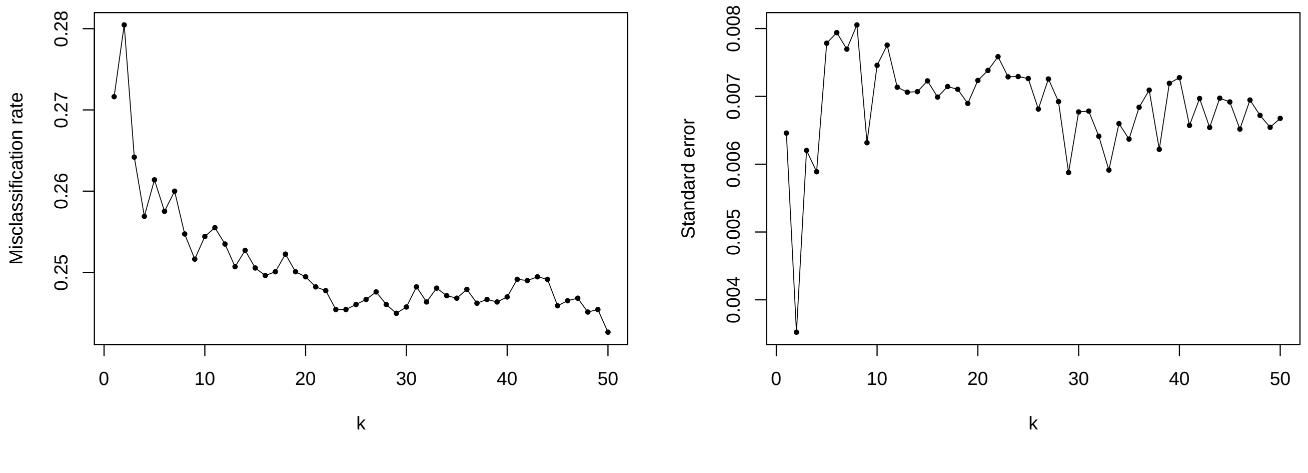
Group	Center	Defense	Left	Right	**Overall Rate**
Rate	0.16	0.21	0.64	0.63	0.34

Table 12: k-Nearest Neighbors misclassification rates

This model performs notably worse than any of the previous models, and, with such a high misclassification rate, is not especially useful. However, there is one important aspect of this model to carefully consider. Since Euclidean distance is used as a measure of distance, this model is highly sensitive to scale − highly sensitive to the relative magnitudes of the covariates. Variables whose magnitudes are large in value or which have large variance will tend to dominate measures of distance. In the next section, we construct a model which seeks to mitigate these issues.

4.5 Scaled *k*-Nearest Neighbors

As discussed in the previous section, the relative magnitude of covariates in a dataset can have an adverse effect on the performance of k-nearest neighbors. For this reason, we re-construct the model in the previous section after standardizing our covariates. Again, 10-fold cross-validation is used to estimate the misclassification rate for each value of k for $k = 1, \ldots, 50$. Figure 7 depicts the misclassification rate for each value of k, alongside its associated standard error. The relevant code can be found in Appendix C.4.



(a) Scaled: Cross-validation misclassification rate for each k

(b) Scaled: Associated standard error

Figure 7: Scaled: k-Nearest Neighbors cross-validation results

The estimated misclassification rate is minimized at $k = 50$, with a standard error of 0.007. On the other hand, most of the values for $k \geq 23$ have misclassification rates within one standard deviation of the minimized misclassification rate. However, given that there is a noticeable drop in the misclassification rate between $k = 49$ and $k = 50$, it may be suitable to choose $k = 50$ as the optimal classification parameter. We do this, and we re-perform cross-validation. The relevant code can be found in Appendix C.3. Below is the confusion matrix.

	Prediction			
Truth	Center	Defense	Left	Right
Center	1521	34	192	114
Defense	6	2262	7	3
Left	190	50	666	295
Right	138	33	506	433

Table 13: Scaled: k-Nearest Neighbors confusion matrix

The estimated misclassification rate is 0.24. This rate is on-par with linear discriminant analysis, relative to which k-nearest neighbors also performs better in classification of left- and right-forwards. Table 14 contains in-group misclassification rates.

Group	Center	Defense	Left	Right	**Overall Rate**
Rate	0.18	0.05	0.51	0.49	0.24

Table 14: Scaled: k-Nearest Neighbors misclassification rates

The decision to standardize the data prior to input into the k-nearest neighbors algorithm dramatically improved results. Here is a comparison – between the original data and the standardized data – of the results of cross-validation for estimating the misclassification rate for each k. The figure on the left shows the results using un-standardized data; the figure on the right shows the standardized results.

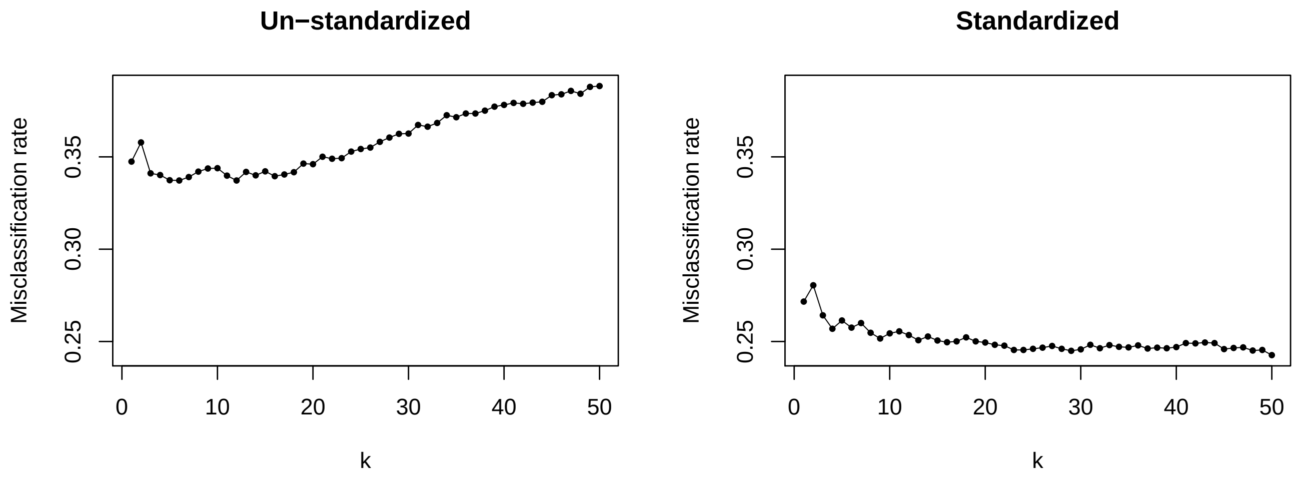


Figure 8: k-Nearest Neighbors cross-validation comparison

Overall, this is a very effective model, and performs much better with standardized data than without.

4.6 Adjusted Position Model

While most of the models constructed above have relatively low misclassification rates, their performance is slightly skewed by their practical inability to distinguish between left- and right-forwards. In-group misclassification rates for these groups in particular are generally greater than 50%. In that light, an interesting question to ask is, "How accurately can we classify players' positions if we don't make a distinction between left- and right-forwards?"

Each of the models from sections 4.2 through 4.5 is now re-constructed with left- and right-forwards considered as a single group. The results of these analyses can be seen in the Appendix. The focus of this section is on the results of linear discriminant analysis – hitherto our (arguably) best-performing model, as we saw in section 4.1. Below are the results.

	Prediction		
Truth	Center	Defense	Forward
Center	1404	1	456
Defense	0	2260	18
Forward	147	8	2156

Table 15: Adjusted: LDA confusion matrix

The results are excellent. The estimated misclassification rate is now 0.09. In particular, classification of defensemen is very nearly perfect. Below are the in-group misclassification rates.

Group	Center	Defense	Forward	**Overall Rate**
Rate	0.09	0.004	0.18	0.09

Table 16: Adjusted: LDA misclassification rates

So, although it is difficult to distinguish between left- and right-forwards, this particular model can classify centers, defensemen, and other forwards with remarkable accuracy.

Given the success of the adjusted LDA model, another interesting question to ask is, "Are all the features of the data necessary to accurately predict adjusted player position, or is there some smaller subset of variables capable of producing similar results?" As smaller models are generally preferred, this is an important question. It may also yield insight into *which* variables are most influential in the determination of a player's position.

To answer this question, we perform a forward-selection algorithm for selecting a subset of k variables, for each k from $k = 2, \ldots p$, where p is the dimension of the feature space. First, each of the $\binom{p}{2}$ subsets of size $k = 2$ is considered, ultimately being selected based on its estimated misclassification rate, as computed through cross-validation using linear discriminant analysis. From there, the best model of size $k = 3$ is chosen by considering each of the $p - 2$ possible models obtained by appending an additional variable. This continues for every possible subset size. The code for this algorithm can be found in Appendix C.5.

Below are the results of this subset selection procedure. Figure 9 shows the estimated misclassification rate as a function of the number of variables in the model.

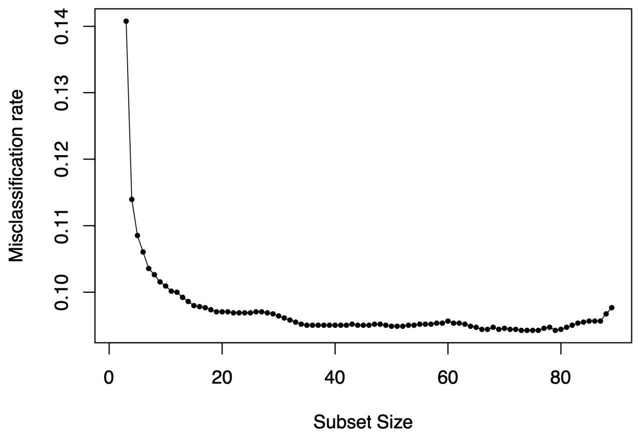


Figure 9: LDA Subset Selection

The misclassification rate is minimized with 73 variables. However, subset sizes beyond 20 perform roughly equivalently, with the addition of variables seeing only moderate gains. In fact, the model with only two variables still performs reasonably well, with an estimated misclassification rate of about 14%. These two variables are `EV FO` and `BkS/60`, or even-strength faceoffs and blocked shots per 60 minutes, respectively.

The indication that these two variables are strong predictors of player position presents an opportunity to visualize the decision boundaries constructed by linear discriminant analysis. In a 2-dimensional feature space, these boundaries can be plotted with relative ease. Figure 10 depicts the results of linear discriminant analysis for the previously described 2-variable model.

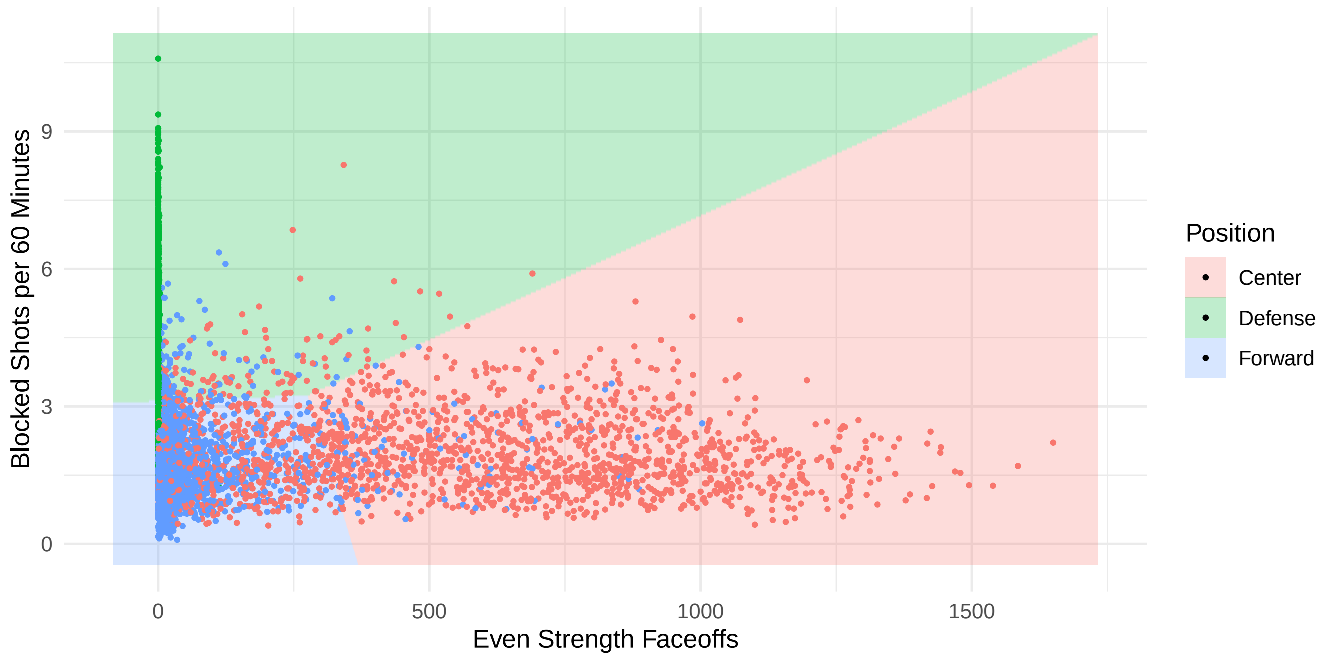


Figure 10: LDA decision boundaries for 2-variable model

In this plot, color represents player position. The background colors represent predicted decision regions. Points falling into a particular colored region will be classified in the corresponding group. The overlaid points are the actual values – the actual groups from the training data.

It becomes immediately clear from this plot exactly why linear discriminant analysis performs so well, and why these two particular variables are so powerful. There are distinct groupings in the training data, modeled very accurately by this classification method. Defensemen are so easy to classify because they block the most shots and do not participate in faceoffs. Centers perform the majority of faceoffs, and forwards, in general, do not block many shots.

Given the struggles that each of our models demonstrated in distinguishing between left- and right-forwards, the decision to join these groups together is likely a wise one. The model presented just above – using linear discriminant analysis – is capable of predicting player position with very high accuracy. As to just how many input features are necessary to do so, it seems that only a small subset of the original data is required – 20 variables are sufficient. So, we select, as our most effective model for predicting player position, that which uses linear discriminant analysis with 20 input features. A list of these features can be found in Appendix C.9.

5 Conclusions

In the previous sections we have manipulated collected data to create a refined dataset, used Ordinary Least Squares to create a model predicting salary, and refined that model through various shrinkage methods, all while paying attention to the assumptions we make to generate these results. We have tested our model with these criteria as well as cross validation techniques, and evaluated its reliability to generate, what we feel, is the most effective model for salary prediction.

We have also employed many classification tools to help us build another predictive model to classify players by position, using LDA, QDA, and k-nearest neighbors. We again use cross validation to refine our model and ultimately generate one that we feel performs exceptionally well. Now, what should one do with these models?

With the information we have provided, we feel the best course of action for a team would be to evaluate their needs first. Does a team score frequently but give up goals consistently? Is the attack solid in the middle of the zone but weak on the edges? Does a team need more scoring from a defenseman? To answer these questions, a team should consult the classification model. Let's say, for instance, a team gets very little scoring contributions from its defensemen but is locked into long term deals with their top attack line. It would be prudent to go pick up a scoring defenseman, without sacrificing too much defensive prowess. Our classification model would suggest picking up a defenseman that lies close to, or even inside of the forward/center classification zone to generate this kind of offense.

Once a cluster of players in a team's region of interest in our position model is selected, the team should then evaluate contract options with our salary model. They should put all of these players in the model and see what players they could pick up at a market value they are comfortable with. We feel that these models provided create a rigorous, algorithmic method for handling personnel, and can give teams a cutting edge in their analytic department.

Appendix

A Variables

Variable Name	Description
Player	Player Name
Season	Season
Team	Teams Played For
S/C	Skater Shoots/Goalie Catches
Pos	Player Position
GP	Games Played
G	Goals
A	Assists
P	Points
+/-	Plus-Minus
PIM	Penalty Minutes
P/GP	Points Per Game Played
EVG	Even Strength Goals
EVP	Even Strength Points
PPG	Power Play Goals
PPP	Power Play Points
SHG	Shorthanded Goals
SHP	Shorthanded Points
OTG	Overtime Goals
GWG	Game Winning Goals
S	Shots
S%	Shooting Percentage
TOI/GP	Time On Ice per Game Played
FOW%	Faceoff Win Percentage
FO	Total Face-offs
FOW	Faceoffs Won
FOL	Faceoffs Lost
EV FO	Even Strength Faceoffs
EV FOW	Even Strength Faceoffs Won
EV FOL	Even Strength Faceoffs Lost
PP FO	Power Play Faceoffs
PP FOW	Power Play Faceoffs Won

Variable Name	Description
SH iSAT/60	Shorthanded Individual SAT For per 60 minutes
SH S/60	Shorthanded Shots per 60 minutes
PP GA/60	Power Play Goals Against per 60 minutes
SH TOI	Shorthanded Time On Ice
SH TOI%	Player's % of Team Shorthanded Time, per Game
PPA	Power Play Assists
PPA1	Power Play Primary Assists
PPA2	Power Play Secondary Assists
PP iSAT	Power Play Individual SAT For
PP Shots	Power Play Shots
PP S%	Power Play Shooting Pct
PPG/60	Power Play Goals per 60 minutes
PPA1/60	Power Play Primary Assists per 60 minutes
PPA2/60	Power Play Secondary Assists per 60 minutes
PPP/60	Power Play Points per 60 minutes
PP iSAT/60	Power Play Individual SAT For per 60 minutes
PP S/60	Power Play Shots per 60 minutes
PP GF/60	Power Play Goals For per 60 minutes
PP TOI	Power Play Time on Ice
PP TOI%	Player's % of Team Power Play Time, per Game
DOB	Date of Birth
Birth City	Player Birth City
S/P	Player Birth State or Province
Ctry	Player Birth Country
Ntnlty	Player Nationality
Ht	Player Height
Wt	Player Weight
Draft Yr	Player Draft Year
Round	Player Draft Round
Overall	Overall Player Draft Number
1st Season	First Season For Game Type
HOF	In Hall of Fame

Variable Name	Description
PP FOL	Power Play Faceoffs Lost
SH FO	Shorthanded Faceoffs
SH FOW	Shorthanded Faceoffs Won
SH FOL	Shorthanded Faceoffs Lost
OZ FO	Offensive Zone Faceoffs
OZ FOW	Offensive Zone Faceoffs Won
OZ FOL	Offensive Zone Faceoffs Lost
NZ FO	Neutral Zone Faceoffs
NZ FOW	Neutral Zone Faceoffs Won
NZ FOL	Neutral Zone Faceoffs Lost
DZ FO	Defensive Zone Faceoffs
DZ FOW	Defensive Zone Faceoffs Won
DZ FOL	Defensive Zone Faceoffs Lost
PP TOI/GP	Power Play Time on Ice per Game
On-Ice PP GF	On-Ice Power Play Goals For
On-Ice SH GA	On-Ice Shorthanded Play Goals Against
SH TOI/GP	Shorthanded Time On Ice Per Game
On-Ice SH GF	On-Ice Short Handed Goals For
On-Ice PP GA	On-Ice Power Play Goals Against
EV TOI/GP	Even Strength Time On Ice Per Game
On-Ice EV GF	On-Ice Even Strength Goals For
On-Ice EV GA	On-Ice Even Strength Goals Against
On-Ice EV GD	On-Ice Even Strength Goal Differential
On-Ice EV GF%	On-Ice Even Strength Goals For Percent
Hits	Hits
Hits/60	Hits per 60 minutes
BkS	Blocked Shots
BkS/60	Blocked Shots per 60 minutes
GvA	Giveaways
GvA/60	Giveaways per 60 minutes
TkA	Takeaways
TkA/60	Takeaways per 60 minutes

Variable Name	Description
1g	Times Scored First Goal of Game
ENG	Empty Net Goals
ENA	Empty Net Assists
ENP	Empty Net Points
MsS	Missed Shots
MsS Wide	Missed Shots Wide of Net
MsS Over	Missed Shots Over Net
MsS Post	Missed Shots Hit Post
MsS Cross	Missed Shots Hit Crossbar
PIM/GP	Penalty Minutes per Game
PIM/TOI%	Penalty Minutes per Time on Ice
Pen Drawn	Penalties Drawn
Pen Taken	Penalties Taken
Net Pen	Net Penalties
Pen Drawn/60	Penalties Drawn per 60 minutes
Pen Taken/60	Penalties Taken per 60 minutes
Net Pen/60	Net Penalties per 60 minutes
Minor	Minor Penalties
Major	Major Penalties
Match	Match Penalties
Msct	Misconduct Penalties
G Msct	Game Misconduct Penalties
SHA	Shorthanded Assists
SHA1	Shorthanded Primary Assists
SHA2	Shorthanded Secondary Assists
SH iSAT	Shorthanded Individual SAT For
SH Shots	Shorthanded Shots
SH S%	Shorthanded Shooting Pct
SHG/60	Shorthanded Goals per 60 minutes
SHA1/60	Shorthanded Primary Assists per 60 minutes
SHA2/60	Shorthanded Secondary Assists per 60 minutes
SHP/60	Shorthanded Points per 60 minutes

B Regression

B.1 Starting Model

Linear Model

```
rm( list = ls() )

# Make sure you're in the 'src' directory.
# REGRESSION ANALYSIS

library(tidyverse)
library(lubridate)
library(leaps)
library(latex2exp)

# Load `data`
load('../data/newData.Rdata')

# X.ind is all the quantitative data and standardized
X.ind <- as_tibble(data %>% select( -c(1, 3:5) ) %>% scale())

# start by fitting a simple linear model
simple.model <- lm(Salary ~ ., data = X.ind)

# plot of residuals vs. fitted
pdf(file="../plots/regression/resid_simp_linear.pdf", bg="transparent", width=6, height=4.8)
plot(simple.model$fitted.values, simple.model$residuals, pch=20, cex=0.75,
     xlab=TeX("Fitted Values"), ylab = TeX("Residuals"))
text(2.5,1.6, TeX(sprintf("Adj. $R^2 = %.3f", summary(simple.model)$adj.r.squared)), cex=0.85)
abline( h = 0, col = 'red', lwd = 1, lty=2 )
dev.off()

# plot of actual vs. fitted
pdf(file="../plots/regression/actVSfit_simp_linear.pdf", bg="transparent", width=6, height=4.8)
plot(simple.model$fitted.values, X.ind$Salary, pch=20, cex=0.75,
     xlab=TeX("Fitted Values"), ylab = TeX("Actual Values"))
text(2.5,-0.5, TeX(sprintf("Adj. $R^2 = %.3f", summary(simple.model)$adj.r.squared)), cex=0.85)
abline( a=0,b=1, col = 'red', lwd = 1, lty=2 )
dev.off()

# Q-Q plot
pdf(file="../plots/regression/qqplot_simp_linear.pdf", bg="transparent", width=6, height=4.8)
qqnorm(simple.model$residuals, xlab=TeX("Theoretical Quantiles"), ylab=TeX("Standardized Residuals"), main="")
qqline(simple.model$residuals, lty=3, col='seashell4')
dev.off()
```

B.2 Subset Selection

Forward Selection Determined by Goodness-of-fit

```
num.features <- length( names(X.ind) ) - 1

# forward selection performed using goodness of fit
step.forward <- regsubsets( Salary ~ ., data = X.ind, method = "forward", nvmax = num.features )
step.forward.sum <- summary(step.forward)

# formulas for determining best model with many variables
adjusted.fits <- as.data.frame( cbind( Rsqr = step.forward.sum$rsq, adjRsqr = step.forward.sum$adjr2,
                                        bic = step.forward.sum$bic, cp = step.forward.sum$cp ) )

best.models <- as.data.frame(t(c(apply(adjusted.fits[1:2], 2, which.max), apply( adjusted.fits[3:4], 2, which.min ))))
names(best.models) <- c('Rsqr', 'adjRsqr', 'bic', 'cp')

# best model determined by bic, which usually produces a small model (penalty on more variables)
best.bic <- as.data.frame( coef( step.forward.sum$obj, best.models$bic ) )
names(best.bic)[1] <- "value"
row.names(best.bic) <- str_replace_all(row.names(best.bic),"`", "")

#best model determined by cp values
best.cp <- as.data.frame( coef( step.forward.sum$obj, best.models$cp ) )
names(best.cp)[1] <- "value"
row.names(best.cp) <- str_replace_all(row.names(best.cp),"`", "")

# gather data based only on the variables bic chose
X.bic <- cbind(X.ind$Salary, X.ind %>% select(row.names(best.bic)[-1]))
names(X.bic)[1] <- 'Salary'

fit.bic <- lm(Salary~., data=X.bic)
resid.bic <- fit.bic$residuals
adjr2.bic <- summary(fit.bic)$adj.r.squared
```

```r
Salary.bic <- fit.bic$fitted.values

# plot of residuals vs. fitted
pdf(file="../plots/regression/resid_forwd_bic.pdf", bg="transparent", width=6, height=4.8)
plot(Salary.bic, resid.bic, pch=20, cex=0.75,
     xlab=TeX("Fitted Values"), ylab = TeX("Residuals"))
text(2,1.6, TeX(sprintf("Adj. $R^2 = %.3f", adjr2.bic)), cex=0.85)
abline( h = 0, col = 'red', lwd = 1, lty=2 )
dev.off()

# plot of actual vs. fitted
pdf(file="../plots/regression/actVSfit_forwd_bic.pdf", bg="transparent", width=6, height=4.8)
plot(Salary.bic, X.ind$Salary, pch=20, cex=0.75,
     xlab=TeX("Fitted Values"), ylab = TeX("Actual Values"))
text(2,-0.5, TeX(sprintf("Adj. $R^2 = %.3f", adjr2.bic)), cex=0.85)
abline( a=0,b=1, col = 'red', lwd = 1, lty=2 )
dev.off()

# Q-Q plot
pdf(file="../plots/regression/qqplot_forwd_bic.pdf", bg="transparent", width=6, height=4.8)
qqnorm(resid.bic, xlab=TeX("Theoretical Quantiles"), ylab=TeX("Standardized Residuals"), main="")
qqline(resid.bic, lty=3, col='seashell4')
dev.off()
```

Forward Selection Determined by 10-fold Cross-validation

```r
# k fold cross validation with forward selection
set.seed(1)
k = 10
folds = sample( 1:k, nrow(X.ind), replace=TRUE )
cv.errors = matrix( NA, k, num.features, dimnames = list( NULL,c(1:num.features) ) )

for (j in 1:k){
  best.fit = regsubsets( Salary ~., data=X.ind[folds!=j,], nvmax=num.features, method="forward" )
  testmat = model.matrix( Salary ~., data = X.ind[folds==j,] )
  for (i in 1:num.features){
    coefi = coef( best.fit, id=i )
    xvars = names( coefi )
    pred = testmat[,xvars]%*%coefi
    cv.errors[j,i] = mean( (X.ind$Salary[folds==j]-pred)^2 )
  }
}
msep <- apply( cv.errors, 2, mean )
number.variables <- which.min(msep)
msep.min <- min(msep)

best.fit <- regsubsets( Salary ~., data = X.ind, nvmax = number.variables, method = "forward" )
best.kfold <- as.matrix(coef( best.fit,id=number.variables ))
names(best.kfold)[1] <- "value"

rownames(best.kfold) <- rownames(best.kfold) %>% str_replace_all("`", "")
X.kfold <- cbind(X.ind$Salary, X.ind %>% select(rownames(best.kfold)[-1]))
names(X.kfold)[1] <- 'Salary'

fit.kfold <- lm(Salary~., data=X.kfold)
resid.kfold <- fit.kfold$residuals
adjr2.kfold <- summary(fit.kfold)$adj.r.squared
Salary.kfold <- fit.kfold$fitted.values

# plot of residuals vs. fitted
pdf(file="../plots/regression/resid_forwd_kfold.pdf", bg="transparent", width=6, height=4.8)
plot(Salary.kfold, resid.kfold, pch=20, cex=0.75,
     xlab=TeX("Fitted Values"), ylab = TeX("Residuals"))
text(2.5,1.6, TeX(sprintf("Adj. $R^2 = %.3f", adjr2.kfold)), cex=0.85)
abline( h = 0, col = 'red', lwd = 1, lty=2 )
dev.off()

# plot of actual vs. fitted
pdf(file="../plots/regression/actVSfit_kfold.pdf", bg="transparent", width=6, height=4.8)
plot(Salary.kfold, X.ind$Salary, pch=20, cex=0.75,
     xlab=TeX("Fitted Values"), ylab = TeX("Actual Values"))
text(2,-0.5, TeX(sprintf("Adj. $R^2 = %.3f", adjr2.kfold)), cex=0.85)
abline( a=0,b=1, col = 'red', lwd = 1, lty=2 )
dev.off()

# Q-Q plot
pdf(file="../plots/regression/qqplot_forwd_kfold.pdf", bg="transparent", width=6, height=4.8)
qqnorm(resid.kfold, xlab=TeX("Theoretical Quantiles"), ylab=TeX("Standardized Residuals"), main="")
qqline(resid.kfold, lty=3, col='seashell4')
dev.off()
```

B.3 Shrinkage Methods

LASSO Method

```r
rm( list = ls() )

# Make sure you're in the 'src' directory.
# SHRINKAGE ANALYSIS

library(tidyverse)
library(lubridate)
library(leaps)
library(glmnet)

# Load `df` and `ds`.
load('../data/newData.Rdata')
# df -- this is the complete dataset, unstandardized.
# ds -- same as df, but standardized.

# X is all the quantitative data.
X.ind <- as_tibble(data %>% select( -c(1, 3:5) ) %>% scale())

num.features <- length( names(X.ind) ) - 1

# lasso method
set.seed(1)
k = 10
grid = 10^seq(-2, 8, length=100)
folds = sample(1:k, nrow(X.ind), replace=TRUE)
cv.errors = matrix(NA, k, 100, dimnames=list(NULL, c(1:100)))

x = model.matrix(Salary ~ ., data=X.ind)[,-1]
y = X.ind$Salary
for (j in 1:k) {
  lasso = glmnet(x[folds!=j,], y[folds!=j], alpha=1, lambda=grid)
  testmat = model.matrix(Salary ~ ., data=X.ind[folds==j,])
  for (i in 1:100) {
    coefi = coef(lasso)[,i]
    pred = testmat%*%coefi
    cv.errors[j, i] = mean((X.ind$Salary[folds==j]-pred)^2)
  }
}

msep <- apply(cv.errors, 2, mean)
min.index <- which.min(msep)
lambda.min <- grid[101-min.index]
msep.min <- min(msep)

lasso.full <- glmnet(x, y, alpha=1, lambda=grid)
lasso.coef <- predict(lasso.full, s=lambda.min, type="coefficients")

# extract the sparse matrix elinating the . entries
lass.coef.tibble<- tibble(name = lasso.coef@Dimnames[[1]][lasso.coef@i + 1], coefficient = lasso.coef@x)
lass.coef.tibble$name<- lass.coef.tibble$name %>% str_replace_all("`", "")

X.pred.lasso <- cbind(X.ind$Salary, X.ind %>% select(lass.coef.tibble$name[-1]))
names(X.pred.lasso)[1] <- 'Salary'

fit.lasso <- lm(Salary~., data=X.pred.lasso)
resid.lasso <- fit.lasso$residuals
adjr2.lasso <- summary(fit.lasso)$adj.r.squared
Salary.lasso <- fit.lasso$fitted.values

# plot of residuals vs. fitted
pdf(file="../plots/shrinkage/resid_lasso.pdf", bg="transparent", width=6, height=4.8)
plot(Salary.lasso, resid.lasso, pch=20, cex=0.75,
     xlab=TeX("Fitted Values"), ylab = TeX("Residuals"))
text(-0.7,-1.5, TeX(sprintf("Adj. $R^2 = %.3f", adjr2.lasso)), cex=0.85)
abline( h = 0, col = 'red', lwd = 1, lty=2 )
dev.off()

# plot of actual vs. fitted
pdf(file="../plots/shrinkage/actVSfit_lasso.pdf", bg="transparent", width=6, height=4.8)
plot(Salary.lasso, X.ind$Salary, pch=20, cex=0.75,
     xlab=TeX("Fitted Values"), ylab = TeX("Actual Values"))
text(-0.5,2.5, TeX(sprintf("Adj. $R^2 = %.3f", adjr2.lasso)), cex=0.85)
abline( a=0,b=1, col = 'red', lwd = 1, lty=2 )
dev.off()

#Q-Q plot
pdf(file="../plots/shrinkage/qqplot_lasso.pdf", bg="transparent", width=6, height=4.8)
qqnorm(resid.lasso, xlab=TeX("Theoretical Quantiles"), ylab=TeX("Standardized Residuals"), main="")
qqline(resid.lasso, lty=3, col='seashell4')
dev.off()
```

Ridge Regression

```r
# ridge regression
set.seed(1)
k = 10
grid = 10^seq(-2, 8, length=100)
folds = sample(1:k, nrow(X.ind), replace=TRUE)
cv.errors = matrix(NA, k, 100, dimnames=list(NULL, c(1:100)))
x = model.matrix(Salary ~ ., data=X.ind)[,-1]
y = X.ind$Salary

for (j in 1:k) {
  ridge = glmnet(x[folds!=j,], y[folds!=j], alpha=0, lambda=grid)
  testmat = model.matrix(Salary ~ ., data=X.ind[folds==j,])
  for (i in 1:100) {
    coefi = coef(ridge)[,i]
    pred = testmat%*%coefi
    cv.errors[j, i] = mean((X.ind$Salary[folds==j]-pred)^2)
  }
}

msep <- apply(cv.errors, 2, mean)
min.index <- which.min(msep)
lambda.min <- grid[101-min.index]
msep.min <- min(msep)

ridge.full <- glmnet(x, y, alpha=0, lambda=grid)
ridge.coef <- predict(ridge.full, s=lambda.min, type="coefficients")

ridge.coef.tibble<- tibble(name = ridge.coef@Dimnames[[1]][ridge.coef@i + 1], coefficient = ridge.coef@x)
ridge.coef.tibble$name<- ridge.coef.tibble$name %>% str_replace_all("`", "")

X.pred.ridge <- cbind(X.ind$Salary, X.ind %>% select(ridge.coef.tibble$name[-1]))
names(X.pred.ridge)[1] <- 'Salary'

fit.ridge <- lm(Salary~., data=X.pred.ridge)
resid.ridge <- fit.ridge$residuals
adjr2.ridge <- summary(fit.ridge)$adj.r.squared
Salary.ridge <- fit.ridge$fitted.values

# plot of residuals vs. fitted
pdf(file="../plots/shrinkage/resid_ridge.pdf", bg="transparent", width=6, height=4.8)
plot(Salary.ridge, resid.ridge, pch=20, cex=0.75,
     xlab=TeX("Fitted Values"), ylab = TeX("Residuals"))
text(-0.7,-1.5, TeX(sprintf("Adj. $R^2 = %.3f", adjr2.ridge)), cex=0.85)
abline( h = 0, col = 'red', lwd = 1, lty=2 )
dev.off()

# plot of actual vs. fitted
pdf(file="../plots/shrinkage/actVSfit_ridge.pdf", bg="transparent", width=6, height=4.8)
plot(Salary.ridge, X.ind$Salary, pch=20, cex=0.75,
     xlab=TeX("Fitted Values"), ylab = TeX("Actual Values"))
text(-0.5,2.5, TeX(sprintf("Adj. $R^2 = %.3f", adjr2.ridge)), cex=0.85)
abline( a=0,b=1, col = 'red', lwd = 1, lty=2 )
dev.off()

#Q-Q plot
pdf(file="../plots/shrinkage/qqplot_ridge.pdf", bg="transparent", width=6, height=4.8)
qqnorm(resid.ridge, xlab=TeX("Theoretical Quantiles"), ylab=TeX("Standardized Residuals"), main="")
qqline(resid.ridge, lty=3, col='seashell4')
dev.off()
```

B.4 Ridge Regression Coefficient Values

Variable Name	(Intercept)	GP	G	A	+/-	PIM
Coefficient Value	-1.62×10^{-16}	-2.10×10^{-2}	4.96×10^{-2}	4.33×10^{-2}	2.68×10^{-2}	-1.02×10^{-2}
Variable Name	P/GP	EVG	EVP	PPG	PPP	OTG
Coefficient Value	2.17×10^{-1}	3.59×10^{-2}	1.09×10^{-1}	7.68×10^{-2}	-8.38×10^{-2}	1.69×10^{-2}
Variable Name	GWG	S	S%	TOI/GP	FO	FOW
Coefficient Value	-3.97×10^{-3}	6.52×10^{-3}	-5.96×10^{-2}	1.02×10^{-1}	-5.19×10^{-2}	3.38×10^{-2}
Variable Name	EV FO	EV FOW	PP FO	PP FOW	OZ FO	OZ FOW
Coefficient Value	1.30×10^{-2}	1.01×10^{-1}	-4.22×10^{-2}	-3.86×10^{-3}	8.34×10^{-2}	1.52×10^{-1}
Variable Name	NZ FO	NZ FOW	PP TOI/GP	On-Ice PP GF	On-Ice SH GA	SH TOI/GP
Coefficient Value	-1.68×10^{-1}	-7.16×10^{-2}	6.91×10^{-2}	-8.05×10^{-2}	-7.87×10^{-3}	8.86×10^{-2}
Variable Name	On-Ice SH GF	On-Ice PP GA	EV TOI/GP	On-Ice EV GF	On-Ice EV GA	On-Ice EV GF%
Coefficient Value	-7.48×10^{-3}	1.12×10^{-1}	7.59×10^{-2}	4.80×10^{-3}	1.31×10^{-2}	-5.03×10^{-2}
Variable Name	Hits	Hits/60	BkS	BkS/60	GvA	GvA/60
Coefficient Value	3.45×10^{-3}	-2.77×10^{-2}	1.13×10^{-2}	-4.96×10^{-2}	-3.80×10^{-3}	9.08×10^{-3}
Variable Name	TkA	TkA/60	1g	ENG	ENA	MsS
Coefficient Value	2.01×10^{-2}	5.35×10^{-2}	-4.28×10^{-2}	-1.14×10^{-2}	2.68×10^{-2}	-1.69×10^{-3}
Variable Name	MsS Wide	MsS Over	MsS Post	PIM/GP	PIM/TOI%	Pen Drawn
Coefficient Value	-4.36×10^{-2}	2.42×10^{-2}	6.17×10^{-2}	1.49×10^{-2}	1.17×10^{-2}	-7.61×10^{-2}
Variable Name	Pen Taken	Pen Drawn/60	Pen Taken/60	Net Pen/60	Minor	Major
Coefficient Value	9.18×10^{-3}	2.37×10^{-2}	1.46×10^{-2}	1.04×10^{-2}	1.38×10^{-2}	-1.22×10^{-2}
Variable Name	Match	Msct	SHA1	SH iSAT	SH Shots	SH S%
Coefficient Value	3.33×10^{-2}	-1.60×10^{-2}	-7.76×10^{-2}	-3.93×10^{-2}	-8.94×10^{-3}	1.93×10^{-3}
Variable Name	SHG/60	SHA1/60	SHA2/60	SHP/60	SH iSAT/60	SH S/60
Coefficient Value	-4.50×10^{-3}	6.68×10^{-2}	-2.80×10^{-2}	1.86×10^{-2}	5.52×10^{-2}	-1.01×10^{-1}
Variable Name	PP GA/60	SH TOI	SH TOI%	PPA1	PP iSAT	PP Shots
Coefficient Value	-1.07×10^{-3}	2.53×10^{-2}	1.68×10^{-2}	1.40×10^{-1}	3.21×10^{-2}	1.20×10^{-1}
Variable Name	PP S%	PPG/60	PPA1/60	PPA2/60	PPP/60	PP iSAT/60
Coefficient Value	3.52×10^{-2}	-1.48×10^{-2}	-3.90×10^{-2}	2.73×10^{-2}	-1.39×10^{-2}	-6.54×10^{-3}
Variable Name	PP S/60	PP GF/60	PP TOI	PP TOI%	Ht	Wt
Coefficient Value	2.06×10^{-2}	5.13×10^{-3}	-1.32×10^{-1}	1.65×10^{-1}	2.57×10^{-2}	4.97×10^{-2}

Table 19: Model selection based on 10-fold cross-validation

C Classification

C.1 Discriminant Analysis

Discriminant Analysis

```
my.da <- function(data, var, type = 'lda', priors = NULL) {
  set.seed(1)
  k <- 10
  n <- nrow(data)
  p <- ncol(data)
  fold <- sample(k, n, replace = TRUE)

  g <- length(unique(data[[var]]))
  confuse <- matrix(rep(0, g^2), nrow = g)
  pred <- data[[1]]
  if (is.null(priors)) {
    priors <- tapply(data[[var]], data[[var]], function(x) length(x) / n) %>%
              as.numeric
  }

  for (i in 1:k) {
    if (type == 'lda') {
      zcv <- lda(data[fold != i, var]~., data[fold != i, 2:p],
                 prior = priors)
    } else if (type == 'qda') {
      zcv <- qda(data[fold != i, var]~., data[fold != i, 2:p],
                 prior = priors)
    } else {
      stop('Invalid type.')
    }
    ppcv <- predict(zcv, data[fold == i, 2:p])
    confuse <- confuse + table(data[fold == i, var], ppcv$class)
    pred[fold == i] <- ppcv$class
  }
  rate <- 1 - sum(diag(confuse)) / sum(confuse)
  rates <- 1 - diag(confuse) / apply(confuse, 2, sum)

  return(list(confuse = confuse, rate = rate, rates = rates, pred = pred))
}
```

C.2 Logistic Regression

Logistic Regression

```
my.log <- function(data, var) {
  set.seed(1)
  k <- 10
  n <- nrow(data)
  p <- ncol(data)
  fold <- sample(k, n, replace = TRUE)

  g <- length(unique(data[[var]]))
  confuse <- matrix(rep(0, g^2), nrow = g)
  pred <- as.character(rep(0, n))
  for (i in 1:k) {
    lr <- multinom(data[fold != i, var] ~ ., data[fold != i, 2:p],
                   trace = FALSE) %>% summary
    logodds <- (lr$coefficients[, 1] + lr$coefficients[, 2:p] %*%
      (data[fold == i, -1] %>% as.matrix %>% t)) %>%
      t %>% cbind(0, .)
    class <- apply(logodds, 1, which.max)
    confuse <- confuse + table(data[fold == i, var], class)
    pred[fold == i] <- class
  }
  rate <- 1 - sum(diag(confuse)) / sum(confuse)
  rates <- 1 - diag(confuse) / apply(confuse, 2, sum)

  return(list(confuse = confuse, rate = rate, rates = rates, pred = pred))
}
```

C.3 *k*-Nearest Neighbors

k-Nearest Neighbors

```
my.knn <- function(data, k) {
  set.seed(1)
  c <- 10
  n <- nrow(data)
  fold <- sample(c, n, replace = TRUE)
  g <- length(unique(data[[1]]))
  confuse <- matrix(rep(0, g^2), nrow = g)
  pred <- data[[1]]
  for (i in 1:c) {
    zcv <- knn(data[fold != i, -1], data[fold == i, -1], data[fold != i, 1], k)
    confuse <- confuse + table(data[fold == i, 1], zcv)
    pred[fold == i] <- zcv
  }
  rate <- 1 - sum(diag(confuse)) / sum(confuse)
  rates <- 1 - diag(confuse) / apply(confuse, 2, sum)

  return(list(confuse = confuse, rate = rate, rates = rates, pred = pred))
}
```

C.4 *k*-Nearest Neighbors Parameter Tuning Neighbors

k-Nearest Neighbors Parameter Tuning

```
cv.knn <- function(data) {
  print(names(data[1]))
  set.seed(1)
  nk <- 50
  c <- 10
  n <- nrow(data)
  fold <- sample(c, n, replace = TRUE)
  misclassrate <- rep(0, nk)
  se <- rep(0, nk)
  nfold <- rep(0, c)
  for (i in 1:c) {
    nfold[i] <- length(fold[fold == i])
  }
  for (k in 1:nk) {
    print(k)
    mcl <- rep(0, c)
    mclrate <- rep(0, c)
    for (i in 1:c) {
      pre <- knn(data[fold != i, -1], data[fold == i, -1], data[fold != i, 1], k)
      mcl[i] <- sum(pre != data[fold == i, 1])
      mclrate[i] <- sum(pre != data[fold == i, 1])/nfold[i]
    }
    se[k] <- sd(mclrate)/sqrt(c)
    misclassrate[k] <- sum(mcl)/n
  }

  return(list(rates = misclassrate, se = se))
}
```

C.5 LDA Subset Selection

LDA Subset Selection

```
set.seed(1)
k <- 10
n <- dim(data)[1]
p <- dim(data)[2]
fold <- sample(k, n, replace = TRUE)
g <- length(unique(data$Pos))
prior <- tapply(data$Pos, data$Pos, function(x) length(x) / n) %>%
  as.numeric

vars <- names(data)[-1]
set <- subsets(length(vars), 2, vars)
rates <- c()

for (j in 1:(dim(set)[1])) {
  print(j)
  confuse <- matrix(rep(0, g^2), nrow = g)
  temp <- data %>% dplyr::select(c('Pos', set[j, ]))
  for (i in 1:k) {
    zcv <- lda(formula = Pos~., data = temp[fold != i,],
               prior = prior)
    ppcv <- predict(zcv, temp[fold == i, 2:3])
    confuse <- confuse + table(temp[fold == i, 'Pos'], ppcv$class)
  }
  rate <- 1 - sum(diag(confuse)) / sum(confuse)
  rates <- c(rates, rate)
}

start_vars <- set[which.min(rates), ]

m <- ncol(data) - 1
all_rates <- c(1, 1)
cv_vars <- vector(mode = 'list', length = m)
cv_vars[[1]] <- NULL; cv_vars[[2]] <- NULL

for (j in 3:m) {
  print(j)
  next_vars <- vars[!(vars %in% start_vars)]
  rates <- c()
  for (var in next_vars) {
    v <- c(start_vars, var)
    confuse <- matrix(rep(0, g^2), nrow = g)
    temp <- data %>% dplyr::select(c('Pos', v))
    for (i in 1:k) {
      zcv <- lda(formula = Pos~., data = temp[fold != i,],
                 prior = prior)
      ppcv <- predict(zcv, temp[fold == i, 2:(dim(temp)[2])])
      confuse <- confuse + table(temp[fold == i, 'Pos'], ppcv$class)
    }
    rate <- 1 - sum(diag(confuse)) / sum(confuse)
    rates <- c(rates, rate)
  }

  all_rates <- c(all_rates, min(rates))
  start_vars <- c(start_vars, next_vars[which.min(rates)])
  cv_vars[[j]] <- start_vars
}

print(which.min(all_rates))
```

```r
print(min(all_rates))
print(cv_vars[[which.min(all_rates)]])

out_vars <- data.frame(vars = cv_vars[[which.min(all_rates)]])
out_data <- data.frame(rates = all_rates)

write.csv(out_vars, file = '../data/class-subset-vars.csv')
write.csv(out_data, file = '../data/class-subset-data.csv')
```

C.6 Adjusted Position Model Results

C.6.1 Quadratic Discriminant Analysis

	Prediction		
Truth	Center	Defense	Forward
Center	1525	7	329
Defense	9	2237	32
Forward	486	43	1782

Table 20: QDA confusion matrix

Group	Center	Defense	Forward	**Overall Rate**
Rate	0.25	0.02	0.17	0.14

Table 21: QDA misclassification rates

C.6.2 Logistic Regression

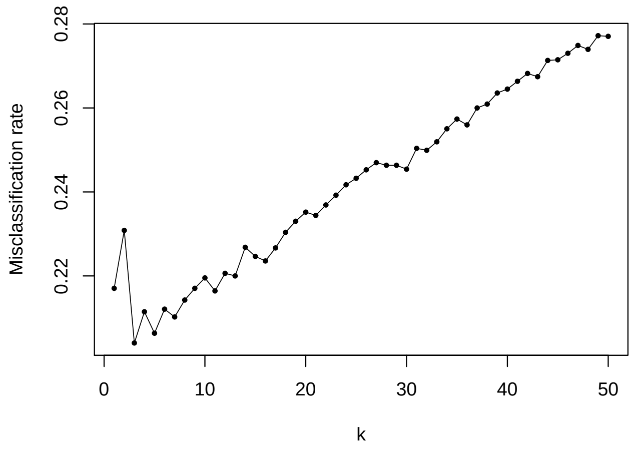
	Prediction		
Truth	Center	Defense	Forward
Center	1376	66	419
Defense	0	2240	38
Forward	248	44	2019

Table 22: Logistic Regression confusion matrix

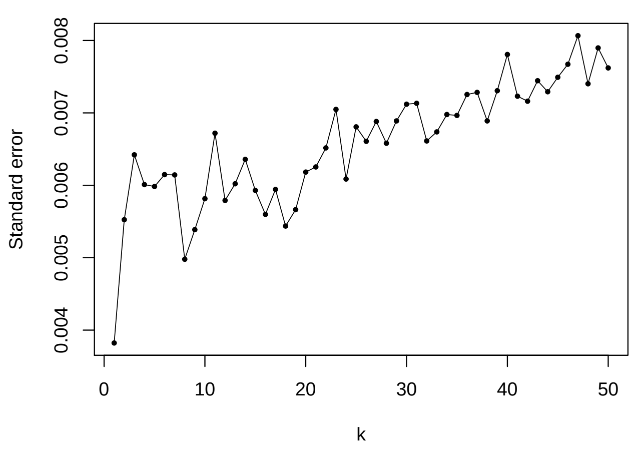
Group	Center	Defense	Forward	**Overall Rate**
Rate	0.15	0.05	0.18	0.13

Table 23: Logistic Regression misclassification rates

C.7 *k*-Nearest Neighbors



(a) Cross-validation misclassification rate for each *k*



(b) Associated standard error

Figure 11: *k*-Nearest Neighbors cross-validation results

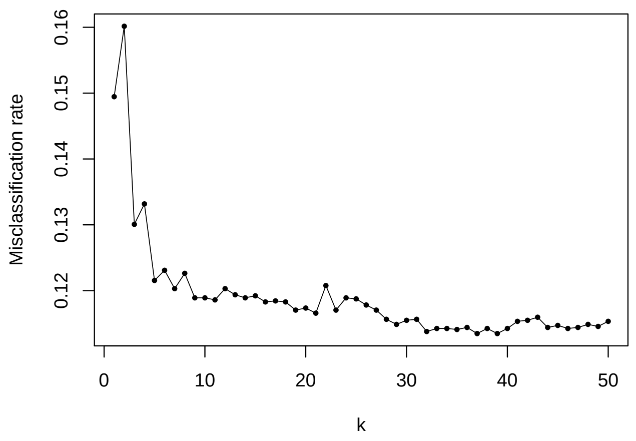
	Prediction		
Truth	Center	Defense	Forward
Center	1429	58	374
Defense	15	1958	305
Forward	257	302	1752

Table 24: *k*-Nearest Neighbors confusion matrix

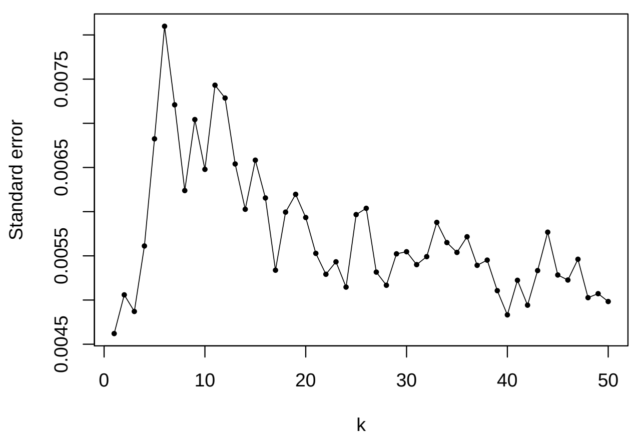
Group	Center	Defense	Forward	**Overall Rate**
Rate	0.16	0.16	0.28	0.20

Table 25: *k*-Nearest Neighbors misclassification rates

C.8 Scaled *k*-Nearest Neighbors



(a) Scaled: Cross-validation misclassification rate for each k



(b) Scaled: Associated standard error

Figure 12: Scaled: k-Nearest Neighbors cross-validation results

	Prediction		
Truth	Center	Defense	Forward
Center	1349	19	493
Defense	5	2245	28
Forward	158	36	2117

Table 26: Scaled: k-Nearest Neighbors confusion matrix

Group	Center	Defense	Forward	**Overall Rate**
Rate	0.11	0.02	0.20	0.11

Table 27: Scaled: k-Nearest Neighbors misclassification rates

C.9 Final Position Model Variables

EV FO	BkS/60	EV TOI/GP	EVG
TkA/60	Major	SH TOI%	TkA
Net Pen/60	On-Ice EV GF%	PP Shots	Hits/60
P/GP	GvA/60	Pen Drawn	SHA1/60
+/-	PPP	GWG	ENA

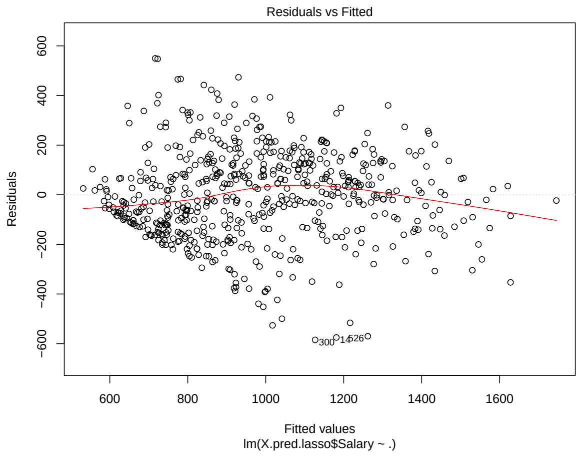
D Variance Stabilization

We will employ the Box-Cox Variance Stabilization Transform. When we run the Box-Cox function In R on the data that was previously selected by the LASSO method, we yield $\lambda = 0.4$, which corresponds to a tranformation of the model

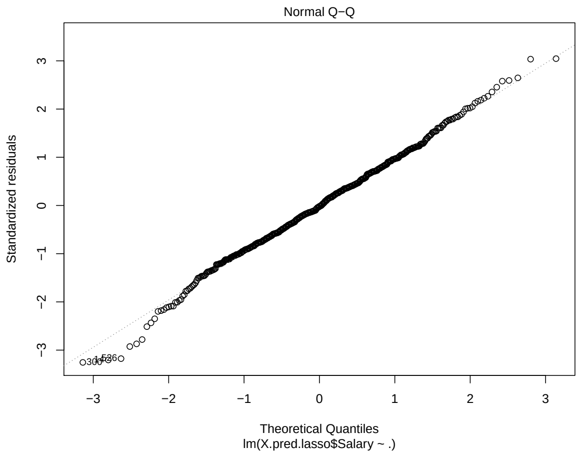
$$Y = \beta X + \varepsilon$$

$$Y^* = \beta X + \varepsilon,\ Y^* = \frac{Y^\lambda - 1}{\lambda}$$

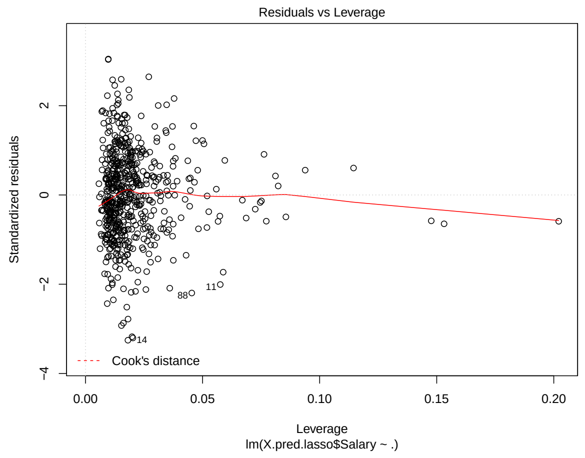
We re-fit the model, making the transformation to Salary, and yield the following results:



(a) Residual Plot



(b) Normal-QQ Plot



(c) Leverage and Influence Points

Figure 13: Model for predicting Salary

Though the normal-QQ plot looks excellent and we appear to have no leverage or influential points skewing our results, we still see the variance appears non-constant. The Box-Cox method has helped a bit, but not enough for us to fully trust our model results. We feel that predicting salaries for players making about league median may be the most difficult to do accurately. The model for doing so is presented below:

Intercept	P/GP	TOI/GP	OZ FOW
69.6287865	353.0108006	0.3269317	0.1661380
On-Ice PP GA	TkA	MsS Over	MsS Post
5.1536340	1.1135807	3.2988508	7.0256657
SH S/60	PP S/60	PP TOI%	Wt
-5.4576648	1.4264103	2.1545694	1.2168675

Table 28: Coefficient estimates for the Variance Stabilized Model

such that the coefficient estimates are β in the model

$$\frac{\text{Salary}^{0.4} - 1}{0.4} = \beta X + \varepsilon$$

References

[1] "NHL Team Contracts & Payrolls". In: *Spotrac* (). Retrieved April 10, 2020 from `https://www.spotrac.com/nhl/`. URL: `https://www.spotrac.com/nhl/`.

[2] A.C. Thomas. "War on Ice: NHL Webscraper". In: *GitHub* (). Retrieved April 10, 2020 from `https://github.com/war-on-ice/nhlscrapr`. URL: `https://github.com/war-on-ice/nhlscrapr`.