Using Machine Learning to Predict Future Points in the NHL

Takehiro Matsuzawa

Department of Statistics & Computer Science
Harvard University

This dissertation is submitted for the degree of
Joint Concentration of Statistics & *Computer Science*

Harvard College — March 2017

Declaration

I hereby declare that except where specific reference is made to the work of others, the contents of this dissertation are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other university. This dissertation is my own work and contains nothing which is the outcome of work done in collaboration with others. Except where states otherwise by reference or acknowledgment, the work presented is entirely my own.

Takehiro Matsuzawa
March 2017

Acknowledgements

Firstly, I would like to express my sincere gratitude to my advisor Professor Kevin Rader for the continuous support of my thesis research. His guidance helped me in all the time of research and writing of this thesis.

Besides my advisor, I would like to thank Professor Alexander Rush and Professor Finale Doshi-Velez, for generously accepting my requests for thesis readers.

Lastly, I would like to express my deepest gratitude to my mother and friends for their support.

Abstract

Goal creation in ice hockey is complicated and difficult to understand. Unlike baseball, players and the puck are always moving and a combination of 5 players produces a goal. Recently, the NHL has published new data about each game. As new statistical analysis tools such as neural network, *k*-nearest neighborhood and random forest regression become available, it becomes possible to analyze goals and assists more holistically by using a wide range of statistical methods.

The aims of this study were to predict the average number of points of each player in the next 5 games by looking at the statistics of an individual player, his team and his opponents in the previous 10 games. Subsequent to this study, I was able to find important variables to predict the number of points. In this study, the random forest regression predicted the average number of points in the next 5 games by mean squared error of 0.0675. This is about a 70% improvement compared to mean squared error of a baseline model that predicts that all the players get the average number of points in the next 5 games.

This paper starts with an exploration of relevant sports statistics and their history. Then the paper shifts its focus on explaining relevant work by other people (Chapter 1). Next the paper explains different modern machine algorithms such as neural network regression, random forest regression and *k*-nearest neighborhood regression used in this research (Chapter 2). Then the paper seeks to minimize prediction errors and find significant features with these methods (Chapter 3). Finally the paper summarizes our main findings (Chapter 4).

Table of Contents

List of Figures

List of Tables

Chapter 1

Getting Started

1.1 Introduction to Statistics in Ice Hockey

1.1.1 Brief Explanation of Statistics in Ice Hockey

A point is awarded to a player for each goal scored or assist earned. Points are one of the most important and simple criteria to evaluate players' goal creation abilities. For instance, the Art Ross Trophy is awarded to the National Hockey League player who leads the league in scoring points at the end of the regular season. However, the use of statistics in the NHL has been very rare until recently and the advanced statistics became available just a few years ago. The Major League Baseball went through a statistical revolution about a decade ago and MLB teams have changed the scouting system as can be seen in the movie "Moneyball". However, unlike baseball, the development of advanced statistics in the NHL has been slow. One of the major reasons was that players and the puck are always moving and it is harder to predict the number of goals and assists than other sports.

1.1.2 Modern Statistics in the NHL

Recently, Nate Silver published an article about how to predict the career of a player in the NBA by examining the careers of similar players [11]. He used statistical techniques similar to *k*-nearest neighborhood and predicted the career trajectories of NHL players. This article was explained more thoroughly later in this chapter. Since points are one of the most important criteria, it is natural to predict points as a criteria for players' offense abilities. In ice hockey games, forwards usually consist of 4 lines and defensemen consist of 3 lines. Players usually play with linemates throughout a game. Therefore, in the NHL, the statistics of players at the same line are assumed to be highly correlated. Moreover, the results of previous games heavily matter when it comes to predicting the number of points by a player because the results of previous games even change line combinations. These features of hockey games make it difficult to predict points. However, advanced statistical and machine learning techniques such as random forest regression and deep learning (neural network) are able to deal with highly correlated data.

1.2 Metrics and Purpose of Thesis

A lot of papers have been written about how to predict the number of points in a whole season. However, little research has focused on predicting the number of points in the next several games. If we are able to predict the number of points in the next several games, it will help NHL teams to select which player to use in playoffs or important games. Moreover, such a model will be useful in selecting which player to call up to the NHL and move down to the AHL in a short term. Therefore, this paper set the response variable as the average number of points by a player in the next 5 games. This research uses the data of previous 10 games as predictors since all the game data are highly interdependent among players at the same line and the results of previous games even change the line combinations. If a player did not play more than previous 10 games out of previous 20 games of the team, this research removes these players because it makes sense to assume that they get scratched and their data are unreliable. The metrics (error) of this research is defined as the following.

$$Error: \quad MSE(\hat{Y}) = \frac{1}{N}\sum_{i=1}^{N}(\hat{Y}_i - Y_i)^2, \tag{1.1}$$

where:

$\hat{Y}$ is the predicted number of average points in the next 5 games.

Y is the actual number of average points in the next 5 games.

N is the total number of the data sets.

The ultimate purpose of the thesis is to decrease this error term by using the best model and find out which features of games account for goal creation in the near future.

1.3 Data Resource

The Hockey Reference Website has data from each game for teams and players. This paper analyzes data from the 1996 to 2015 seasons. Since almost all the players who play consistently in the NHL are drafted players, this paper does not consider undrafted players.

Moreover, by removing undrafted players, this paper is able to include draft ranks as a quantitative predictor for a response variable. This paper scraped all the information from the Hockey Reference Website.

For Example (The Hockey Reference Website):
Team Data:
For example: http://www.hockey-reference.com/teams/MTL/2017.html
Player Data:
For example: http://www.hockey-reference.com/players/g/galchal01.html

1.3.1 Variables to Use

There are three types of data (player data, team data and opponent data).
Player data
Player data have the following features.

- Draft Ranks
- Height
- Weight
- Age
- Dominant Hand
- Position
- Player Assists
- Player Goals
- Player Number of Shifts
- Player Time on Ice
- Player Plus Minus
- Player Penalty Minutes
- Number of Games Played in the Season
- Results of Games for Players
- *Player Shots For

Team Data and Opponent Data
Both opponent data and team data have the following values.

- Games Team Played
- Home Games
- Team Goals Scored for
- Team Goals Against at Even Handed
- Team Goals Against at Power Play
- Cumulative Wins for Teams
- Cumulative Losses for Team
- *Team Shots For
- Team Winning Streaks

- Penalty Minutes for Team
- Team Goals in Power Play
- Team Goals in Shorthanded
- Team Shots Against
- Results of Games for Teams
- Team Goals in Shorthand
- Team Shots Against
- Team Goals Against in Power Play
- Team Goals Against in Shorthand

*‘Player Shots for’ mean ’Number of Shots for a Team by a Player’.
*‘Team Shots for’ mean ’Number of Shots by a Team’.
*‘Team Shots Against’ mean ’Number of Shots by Opponent’.

Procedures to scrape data

This research scraped all of these data from the Hockey Reference Websites as follows.

1. Scrape data about features of players such as height, weight, draft ranks from draft 1989 $\sim$ 2016. (`player_feature_data.ipynb`)
2. Scrape data about 10 recent games of teams. (`data_scraper_10_games_team.ipynb`)
3. Scrape data about 10 recent games of players. (`data_scraper_10_games.ipynb`)
4. Clean up team and player data sets. (`final_cleanup_1.ipynb`)
5. Merge two data sets (`final_cleanup_2.ipynb`)

All the scraped data will be stored in `comp_merged_data_full.csv` after this scraping process. All the regression models can be run by using this csv file.

1.4 Representations and Data Cleansing

1.4.1 Data Cleansing and Removal of Unreliable Data

First, players data and team data were scraped from the Hockey Reference Website. Then player data and teams data were merged. Players who did not play more than 10 games out of 20 most recent games of the team were removed from consideration. It is reasonable to assume these players get injured or get scratched. Some players play only 10 games in one season and these players also provide less accurate data. The more consistently players play in the NHL, the more reliable their data is since there are more data available for the players. A potential problem of removing players who play fewer games is that the data has more data about players who play more consistently such as Sidney Crosby and Jonathan Toews and the data do not have much information about third-line players. Then the means of player data, team data and opponent data in the previous 10 games of a player and the mean of player's points in the subsequent 5 games were calculated. Furthermore, if a player did not play at least 5 subsequent games out of 10 games of the team, this player was removed from consideration because it is reasonable to assume these players were injured or scratched.

1.4.2 Classification of Variables

Variables were classified into continuous variables and binary variables. This process is necessary to normalize variables and convert binary variables to 0s or 1s. There are only two binary variables (dominant hand and position).

Standardization of Continuous Variables

Since neural networks and some linear regressions such as ridge regression require standardization of the data sets, all the continuous variables are standardized. The standardization is defined as follow.

$$\text{Standardization:} \quad Z = \frac{X - E[X]}{\sigma(X)}, \tag{1.2}$$

where:

X is a continuous variable.

$E[X]$ is a sample mean of X.

$\sigma(X)$ is the standard deviation of X.

The standardization is very important when dealing with parameters of different units and scales. This is especially important when using k-nearest neighborhood or neural networks.

Conversion of Categorical Variables

DH(dominant hand) and POS(position) are both binary categorical variables and were converted to binary variables as follow.

$$DH = \begin{cases} 1, & \text{if dominant hand is right} \\ 0, & \text{otherwise} \end{cases} \tag{1.3}$$

$$POS = \begin{cases} 1, & \text{if position is forward} \\ 0, & \text{otherwise} \end{cases} \tag{1.4}$$

After data cleaning (1.4.1 to 1.4.4), all the variables in the data sets are now normalized and quantitative.

1.5 Related Work

Recently, Nate Silver published an article about how to predict the career of each player in the NBA by examining the careers of similar players [11]. He used statistical techniques similar to *k*-nearest neighborhood and predicted the careers of NBA players. Silver first defined players' skills as players' physical features (e.g. height, weight and draft rank) and player's game data (e.g. points, free-throw frequencies, three-point frequencies). He was able to figure out similar players based on the players' skills. Silver's analysis is very similar to *k*-nearest neighborhood. This research uses similar data sets (physical feature and game data) to Silver's NBA model.

Chapter 2

Method

2.1 Computational Platform & Model Selections

Performing full-data regressions on the entire dataset of about 400,000 rows is very computationally-intensive. Due to this computational constraint, this research had to be selective on which regressions to perform and each regression was computed on the Odyssey supercomputing cluster.

2.1.1 The Odyssey Cluster

This project was deployed on the Odyssey supercomputing cluster, hosted by Harvard University's Faculty of Arts and Sciences Research Computing Team. All regression models were run on the Harvard Odyssey cluster. The potentially highly-correlated large data with many quantitative variables suggests that linear regressions, *k*-nearest neighborhood regressions, neural network regressions, random forest regressions, gradient boosting regressions, and generalized linear models may be some of the most powerful models to predict the number of points in the next 5 games. This paper fit these models to the data on the Odyssey Cluster and computed the metrics, MSE(Mean Squared Error), defined as Equation 1.1 in Chapter 1. In this prediction model, this paper used the ordinary linear regression (OLR) with only intercept as a baseline evaluation. The MSE of OLR with only intercept is 0.2241. All the results from this chapter other than the baseline are in Chapter 3 (Results & Interpretation).

2.2 Linear Regressions

Since this prediction model is supervised learning, this paper used ordinary linear regression, ridge regression, lasso regression and elastic regression to predict the average number of points in the next 5 games.

2.2.1 Ordinary Linear Regression

This is the simplest linear regression model. This model might not be the best model since the variables in the model are highly correlated.

Mathematical Notation of OLR

$$J(\beta) = \sum_i (\beta^T x_i - y_i)^2$$

where

β is coefficients.

x is prediction variables.

y is response variables.

The purpose of OLR is to find the best estimate of β. The following formula derives the best estimate of β.

$$\frac{\partial J(\beta)}{\partial \beta} = \sum_i (\hat{\beta}^T x_i - y_i)x = 0$$

$$XX^T\hat{\beta} = Xy$$

$$\hat{\beta} = (XX^T)^{-1}Xy \tag{2.1}$$

1. Normality Assumption

Ordinary linear regression assumes that the response variable will follow the normal distribution. It is necessary to check the normality assumption of ordinary linear regression.

2. Independence Assumption

Since ordinary linear regression assumes the independence among variables, it is necessary to evaluate the assumption of independence among predictors. Variance inflation factors (VIF) measure how much the variance of the estimated regression coefficients are inflated as compared to when the predictor variables are not linearly related. Faraway explains Variance inflation factor in his book [5].

Variance Inflation Factor

$$\text{vif}_j = \frac{1}{1 - R_j{}^2} \tag{2.2}$$

where

$R_j{}^2$ is the R^2 for the least-square regression of x_j on the other predictors.
If $R_j{}^2$is near 1, "vif" becomes very large and x_j is involved in collinearity. Variables that are not removed based on VIF are defined as **Variables after VIF**.

3. Influential Observations

Cook's distance measures how much $\hat{\beta}$ changes due to the inclusion of the *i*th observation. An influential point is one whose removal from the dataset would cause a large change in the fit. Faraway explains Cook's distance in his book [5]. Cook's distance finds a potential outlier for the data set. Cook's distance of *i*th value is defined below.

$$D_i = \frac{(\hat{\beta}_{-i} - \beta)^T (var(\hat{\beta}_i))^{-1} (\hat{\beta}_i - \beta)}{J + 1}$$

where $\hat{\beta}_{-i}$ is the estimate of β from an analysis that omits the *i*th observation. Values of D_i near 1.0 or greater are conventionally indicative of being overly influential.

4. Feature Selection (Backward Stepwise)

Since there are many variables in this ordinary linear regression and too many variables cause over-fitting, it is reasonable to use a selection algorithm to choose important variables out of all variables. This research used **Variables after VIF** as the initial set of predictors. Then it selected important features from **Variables after VIF** by using backward stepwise feature selection. Backward stepwise variable selection method is one of the most widely used selection methods and defined as follows. α is set as 0.05.

1. Start with all the predictors (**Variables after VIF**) in the model
2. Remove a predictor with highest p-value greater than α
3. Refit the model and go to Step 2
4. Stop when all p-values are less than α

2.2.2 Ridge Regression

This research uses ridge regression mainly in order to avoid over-fitting training data. The detailed explanations of ridge regression are provided below [2, 6].

Properties of Ridge Regression

By shrinking the coefficients of predictors with L_2 penalties, ridge regression regulates the coefficients and prevents the model from overfitting the training data. Ultimately, ridge regression increases the bias and decreases variance. Moreover, the inclusion of λ makes problem non-singular and matrix invertible. One of the problems of the linear regression is that sometimes the matrix X^TX in equation (3.1) is nearly singular and not invertible. If it is singular, we find multiple solutions to the equation. Ridge regression makes the matrix invertible by adding λ and find a unique solution. Even though ridge regression is good for multicollinearity and grouped selection, it is not good for variable selection. Ridge regression shrinks the coefficients of unimportant variables close to 0, but not all the way to 0. Therefore, if two predictors are highly correlated among themselves, the estimated coefficients become similar for them and ridge regression shrinks coefficients of these variables. Ridge regression is thus suitable for group selection.

Mathematical Notation of Ridge Regression

$$J(\beta) = \lambda \sum_j \beta_j^2 + \sum_i (\beta^T x_i - y_i)^2$$

where

β is coefficients.
x is prediction variables.
y is response variable.
λ is a regularization term.

The purpose of ridge regression is to find the best estimate of β. The following formula derives the best estimate of β.

$$\frac{\partial J(\beta)}{\partial \beta} = \lambda \hat{\beta} + \sum_i (\hat{\beta}^T x_i - y_i)x = 0$$

$$(XX^T + \lambda I)\hat{\beta} = Xy$$

$$\hat{\beta} = (XX^T + \lambda I)^{-1} Xy$$

2.2.3 Lasso Regression

Properties of Lasso Regression

Lasso regression adds L_1 penalties to the linear regression and conducts variable selections. Lasso regression automatically selects important variables and reduces the coefficients of unimportant variables to 0 [6]. Therefore lasso regression is good for eliminating trivial variables but not good for selecting significant variables when variables are heavily correlated. Especially since the data of ice hockey is potentially highly correlated, lasso regression tends to pick one of correlated variables and shrinks the other variables' coefficients to 0. Therefore it makes sense to try a linear regression other than lasso regression such as elastic net regression, which combines the strength of lasso regression and ridge regression. Bishop provides mathematical notation for lasso regression [2] as below.

Mathematical Notation of Lasso Regression

$$\frac{1}{2}\sum_{n=1}^{N}(t_n-\beta^T x_n)^2+\frac{\lambda}{2}\sum_{j=1}^{M}|\beta_j|$$

2.2.4 Elastic Net Regression

Elastic net regression combines strength of ridge regression and lasso regression by encouraging grouping effect in the presence of highly correlated variables. Since some variables (such as goals and assists) are correlated, it is reasonable to use elastic net regression. Friedman provides a mathematical notation and the general idea behind elastic net regression in his paper [6].

Mathematical Notation of Elastic Net Regression

$$\min_{\beta_0,\beta\in R^{p+1}}\sum_{i=1}^{N}(y_i-\beta_0-\beta^T x_i)^2+\lambda P_\alpha(\beta)$$

where

$P_\alpha(\beta)=|\frac{1}{2}(1-\alpha)\beta_j^2+\alpha|\beta_j||$

The elastic net with $\alpha = 1-\varepsilon$ for some small $\varepsilon > 0$ performs much like the lasso, but removes any degeneracies caused by extreme correlations. More generally, P_α creates a bridge relationship between ridge and lasso.

2.3 K-Nearest Neighborhood Regressions

2.3.1 Ordinary K-Nearest Neighborhood Regression

Ordinary *k*-nearest neighborhood regression compares data and predict based on similar data. This method is similar to Nate Silver's method [11]. Silver's method was explained as Related Work in Chapter 1. *K*-nearest neighborhood algorithm is defined as follows.

1. Given a query instance x_q to be predicted.
   Let $x_1, x_2, x_3, x_4, ..., x_K$ denote K instances from training examples that are nearest to x_q in the feature space. These neighbors are chosen from a set of training points whose response variable is known.
2. Return the average of the predicted response variables of the K nearest instances.

2.3.2 Attribute Weighted K-Nearest Neighborhood

Curse of Dimensionality
Distance in ordinary *k*-nearest neighborhood regression usually relates to all the attributes and assumes all of them have the same effects on distance. The similarity metrics do not consider the relation of attributes which result in inaccurate distance and potentially increase MSE. Increase in mean squared error of regression due to the presence of many irrelevant attributes is often termed as the curse of dimensionality [4]. One way to avoid the curse of dimensionality is to use backward elimination algorithm and decrease the number of variables to use.
Backward Elimination Algorithm

1. Delete one variable.
2. For each training example x_i in the training data set
   (a) Find the *k* nearest neighbors based on the Euclidean distance
   (b) Calculate the mean of the response variables of the *k* nearest neighbors
   (c) Calculate MSE defined in Equation (1.1)
3. If MSE decreases after a variable is removed, the variable is not used in the next iteration.

Repeat backward elimination algorithm and keep on removing variables as long as MSE decreases when refitting the weighted K-nearest neighborhood regression.

2.4 Random Forest Regression and Boosting Algorithm

2.4.1 Random Forest Regression

Breiman proposed random forests, which add further randomness to bagging [3]. In bagging, each node is split using the best split among all variables. In a random forest, each node is split using the best among a subset of predictors randomly chosen at each mode. This random feature selection process is robust against overfitting. Random forest regression works by making a number of trees and taking the mean of response variables of observation falling in that node. Random forest is potentially useful in this hockey data set because there are many variables and random forest is able to identify important variables. Moreover, random forest is robust against high correlation. The main idea behind random forest regression is that a group of weak learners can work together to form a strong learner. Liaw explains algorithm of random forest regression in detail [8]:

Algorithm 1 Random Forest Regression

```
For Input: data set $D = (x_1, y_1), ..., (x_n, y_n)$
Let $f$ is the number of features

for $t = 1$ to $M_{RF}$ do where $M_{RF}$ is the number of trees
    $D_t \subseteq D$
    $h_t$ = Cart(K, M) where $K \leq f$ randomly chosen features at each split and $M$ is the minimum number of nodes
end for

$T(\cdot) = \frac{1}{M_{RF}} \sum_{t=1}^{M_{RF}} h_t(\cdot)$
return $T(\cdot)$
```

First, draw n_{tree} samples at random with replacement from the original data where n_{tree} is the number of trees. Second, at each node, randomly sample m_{try} of the predictors and choose the best split from among those m_{try} variables. This paper uses $m_{try} = n/3$ since changing m_{try} did not decrease MSE. Finally, predict new data by taking average of the n_{tree} trees. This research paper tried several possible number of trees, minimum samples for a terminal node to minimize the MSE.

Variable Importance

Breiman proposed a method to measure variable importance in random forest [3]. Suppose there are M input variables. After each tree is constructed, the values of the mth variable in examples are randomly permuted. The predicted value given for each x_n is saved. This is repeated for $m = 1,2,...,M$. At the end of the run, the plurality of predicted value for x_n with the mth variable noised up is compared with the true value of x_n to give a mean squared error rate. The output is the percent increase in error rate.

2.4.2 Gradient Boosted Regression Tree Algorithm

Gradient boosted regression tree algorithm works by putting more weight on difficult instances to predict and less on those already handled well. New weak learners are added sequentially and focus their training on the more difficult patterns. This means that samples that are difficult to predict receive increasing larger weights. Trees are constructed greedily, choosing the best split points based on purity scores to minimize the squared loss. Mohan explains gradient boosted regression tree algorithm [10].

Algorithm 2 Gradient Boosted Regression Tree Algorithm

```
For Input: data set $D = (x_1,y_1),...,(x_n,y_n)$

$F$ = RandomForests($D, K_{RF}, M_{RF}$ ) where $M_{RF}$ is the number of trees in the forest and
$K_{RF}$ is the number of features used for split in each node

for $i = 1$ to $n_1$ do
    Initialization: $r_i = y_i - F(x_i)$
end for

for $t = 1$ to $M_B$ do where $M_B$ is the number of iterations

    $T_t$ = Cart($(x_1,y_1),...,(x_n,y_n), f, d$) where $d$ is depth and $f$ is number of features

    for $i = 1$ to $n$ do
        $r_i = r_i - \alpha T_t(x_i)$
    end for

end for

$T(\cdot) = F(\cdot) + \alpha \sum_{t=1}^{M_B} T_t$
return $T(\cdot)$
```

Let $T(x_i)$ denote the current prediction of sample x_i. The objective of this algorithm is to minimize the square loss $L = \frac{1}{2}\sum_{i=1}^{n}(T(x_i) - y_i)^2$. Gradient boosted regression tree algorithm performs gradient descent in the instance space $x_1,...,x_n$. During each iteration the current prediction $T(x_i)$ is updated with a gradient step

$$T(x_i) \leftarrow T(x_i) - \alpha \frac{L}{T(x_i)}$$

where
α is a learning rate.

This research paper explored the number of trees, minimum samples for a terminal node and learning rate to minimize MSE.

2.5 Neural Network

Neural networks are conventionally good at dealing with correlated data. Since it is natural to assume that hockey variables are highly correlated, it is reasonable to use neural networks for this data set. Even though neural networks tend to have high accuracy, there is a tradeoff between cost and benefits such as accuracy and speed [9]. There are many variables and it is computationally too expensive to calculate the optimal weights of neural network by using all the variables. According to the results of random forest which was the best model so far in this paper, the following attributes are the most important variables to predict the number of average points in the next 5 games.

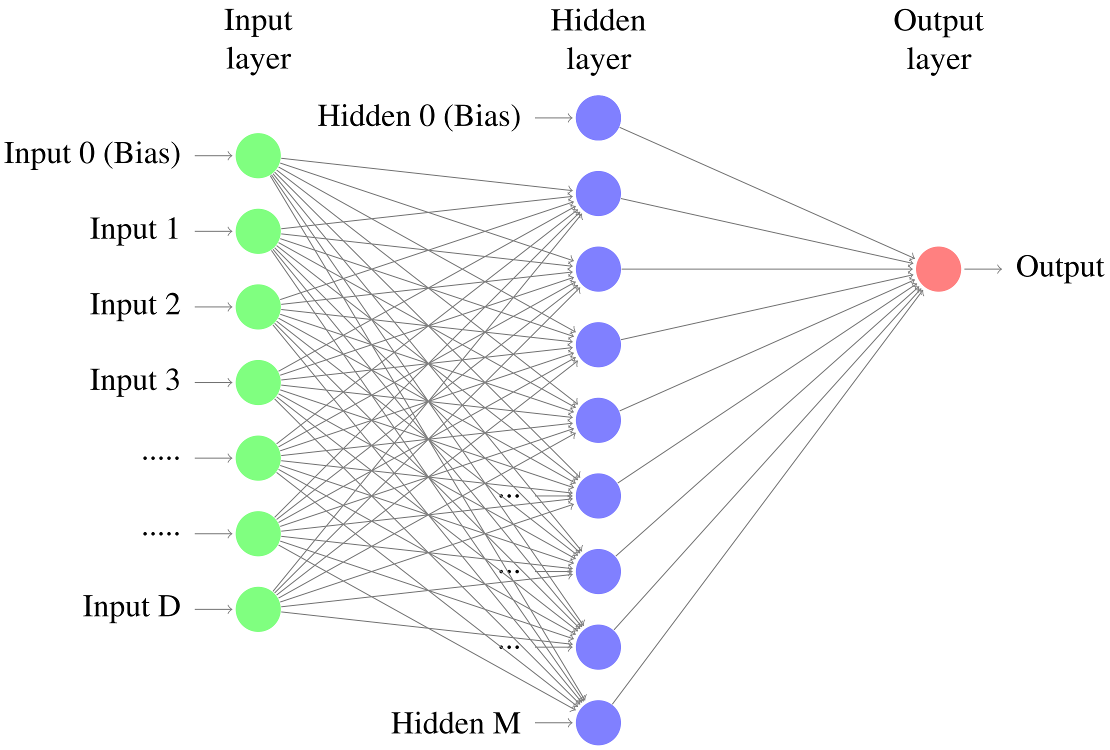
- Player Assists
- Time on Ice
- Goals at Power Play
- Position (Forward or Defenseman)
- Goals at even handed
- Number of Shifts
- Height
- Weight
- Drafts

As stated above, the variable selection by random forest narrows down variables to use. However, over-fitting is a serious problem in neural networks and large networks are also slow to implement. Therefore it is reasonable to remove insignificant variables [12]. One of the ways to avoid over-fitting is to use principal component analysis (PCA) and reduce the dimensionality of data. This paper created two models of the neural networks with different sets of variables. The first model is to use variables at original scale (Model 1). The second model is to use variables transformed by PCA (Model 2).

2.5.1 Mathematical Explanation of Neural Network Regression

A case of a 2-layer neural network regression model is explained in the below. It is not hard to make a generalized layer neural network regression model from a 2-layer neural network model. Bishop explains Mathematical Notation of Neural Network Regression [2].

Fig. 2.1 Neural Network Regression



First we construct M linear combinations of the input variables $x_1, ..., x_D$ in the form

$$a_j = \sum_{i=1}^{D} w_{ij}^{(1)} x_i + w_{j0}^{(1)} \tag{2.3}$$

where $j = 1, 2, ..., M$ and we refer $w_{ij}^{(1)}$ as *weights* and parameters $w_{j0}^{(1)}$ as *biases*. In Fig 2.1, Input 0 is $w_{j0}^{(1)}$ (bias). The quantities a_j are called 'activations'.

$$z_j = h(a_j) \tag{2.4}$$

where

$h(x)$ is an activation function.

This paper uses the identity function as an activation function because this neural network is a regression, not a classification. The following transformation corresponds to the second layer of the network.

$$z_j = h(a_j) = a_j \tag{2.5}$$

$$a_k = \sum_{j=1}^{M} w_{kj}^{(2)} z_j + w_{k0}^{(2)} \tag{2.6}$$

$$y_k = \sigma(a_k) = a_k \tag{2.7}$$

where

k is $k = 1,...,K$, and K is the total number of outputs.

$w_{kj}^{(2)}$ is *weights*

$w_{k0}^{(2)}$ is *biases*

y_k is a set of network outputs

σ is activation function and it is identity here From (3.3) to (3.7),

$$y_k = \sum_{j=1}^{M} w_{ji}^{(2)} (\sum_{i=1}^{D} w_{ij}^{(1)} x_i + w_{j0}^{(1)}) + w_{k0}^{(2)} \tag{2.8}$$

where the set of all weight and bias parameters have been grouped together into a vector w. Thus neural network regression model is simply a nonlinear function from a set of input variables x_i to a set of output variables y_k controlled by a vector w of adjustable parameters. If a model has more layers, the model just needs more linear regressions between each layer.

2.5.2 Training Neural Network Regression (Backpropagation)

1. Gradient Descent Optimization

This research paper uses gradient descent optimization in order to minimize the overall error. Gradient descent optimization finds the local optimal weight values for neural network regression.

$$E(w) = \sum_{n=1}^{N} E_n(w) \tag{2.9}$$

$$w^{(i+1)} = w^{(i)} - \eta \nabla E_n(w^{(i)}) = w^{(i)} - \eta \frac{\partial E}{\partial w_{ji}} \tag{2.10}$$

where

E_n is the error of input x_n.

$w^{(i)}$ is the weight at ith iteration.

2. Back Error Propagation

Back error propagation is a way to find an efficient technique for evaluating the gradient of an error function $E(w)$ for a feed-forward neural network. Back error propagation is used for neural networks for both models (Normal Model and PCA Model).

$$y_k = \sum_i w_{ki} x_i \tag{2.11}$$

$$E_n = \frac{1}{2} \sum_k (y_{nk} - t_{nk})^2 \tag{2.12}$$

$$\frac{\partial E_n}{\partial w_{ji}} = (y_{nj} - t_{nj}) x_{ni} \tag{2.13}$$

Now consider the evaluation of the derivative of E_n with respect to a weight w_{ji}. The outputs of the various units will depend on the particular input pattern n. Since this is a neural network regression and this uses identity as an activation function,

$$\frac{\partial E_n}{\partial w_{ji}} = \frac{\partial E_n}{\partial a_j} \frac{\partial a_j}{\partial w_{ji}} = x_i \tag{2.14}$$

The derivative of the total error E can then be obtained by repeating the above steps for each pattern in the training set and then summing over all patterns.

$$\frac{\partial E}{\partial w_{ji}} = \sum_n \frac{\partial E_n}{\partial w_{ji}} \tag{2.15}$$

2.5.3 PCA-Transformed Variables

1. Transform Variables with PCA

As Abdi explains [1], principal component analysis (PCA) is a multivariate technique that analyzes a data table in which observations are described by several inter-correlated quantitative dependent variables. Its goal is to extract the important information from the table, to represent it as a set of new orthogonal variables called principal components.

2. Mathematical Definition of PCA

Kolenikov explains a mathematical definition of PCA [7] as follows. If x is a random vector of dimension p with finite $p \times p$ variance-covariance matrix $Var[x] = \Sigma$, then principal component analysis (PCA) solves the problem of finding the directions of the greatest variance of the linear combinations of x's. In other words, it finds the orthonormal set of coefficient vectors $a_1, a_2, ..., a_k$ such that

$$a_1 = \underset{a:|a|=1}{\text{arg max}} \, Var[a'x]$$

$$\cdots$$

$$a_k = \underset{a:|a|=1}{\text{arg max}} \, Var[a'x]$$

where

$a \perp a_1, a_2, ..., a_{k-1}$

The linear combination $a'_k x$ is referred to as the k-th principal component (PC). The first principal component has the greatest variance and extract the largest information from the data. The second component will be orthogonal to the first one and has the greatest variance orthogonal to the first component, and acquire the greatest information in that subspace. PCA repeats this process.

2. Proportion of Variance Explained

We are able to calculate the proportion of variance explained by each component. Then we can reduce the data to smaller dimensionality by keeping dimensions that explain the variance well. According to Kolenikov [7], A popular measure of fit by principal components is referred to as proportion of explained variance:

Proportion of Variance Explained by dimension k is

$$R_k^{ind} = \frac{\lambda_k}{\lambda_1 + \lambda_2 + ...\lambda_P} \tag{2.16}$$

where p is the number of variables.

Cumulative Proportion of Variance Explained by dimension up to k is

$$R_k^{cum} = \frac{\lambda_1 + \lambda_2 + ...\lambda_k}{\lambda_1 + \lambda_2 + ...\lambda_P} \tag{2.17}$$

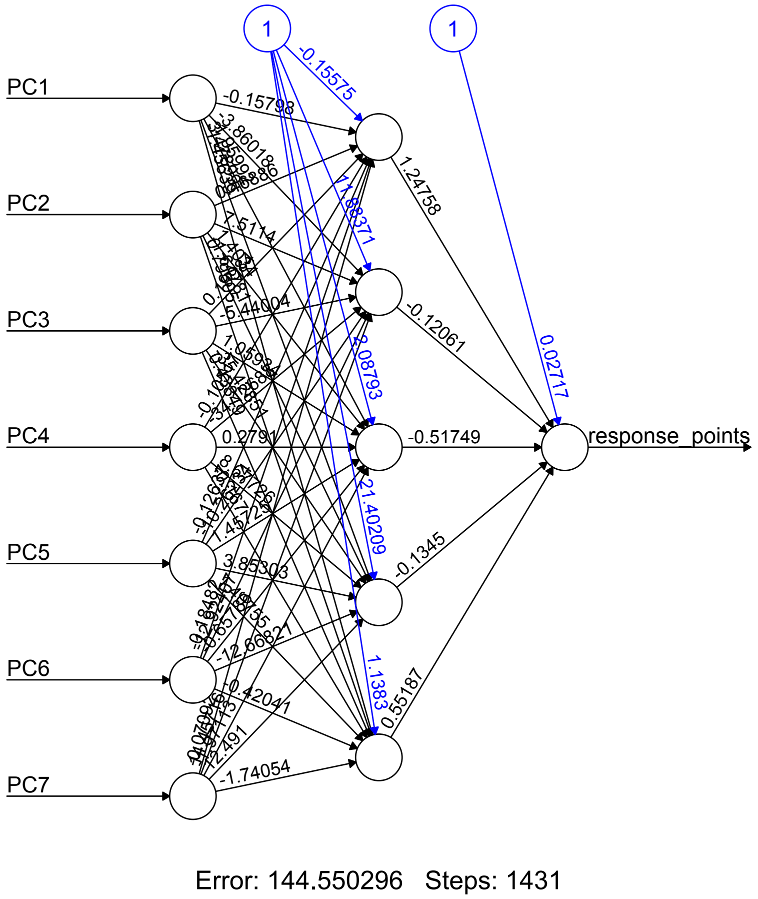
where p is the number of variables.

Proportion of Variance Explained by each dimension is useful in determining which dimension to use in a neural network regression.

3. Example of Neural Network Regression

Fig. 2.2 is an example of Neural Network with 7 input nodes and 5 hidden nodes.

Fig. 2.2 Neural Networks (Example)



2.6 Generalized Linear Models

Generalized linear models (GLMs) extend linear models to accommodate both non-normal response distributions and transformations to linearity.

Assumptions of Generalized Linear Model

A generalized linear model makes the following assumptions [13].

1. There is a response variable y observed independently at fixed values of predictor variables $x_l, ..., x_p$
2. The predictor variables $(x_l, ..., x_p)$ may only influence the distribution of y through a linear predictor
3. The distribution of y has an exponential family distribution.

4. The mean μ is a smooth invertible function of the linear predictor. $\eta = g(\mu)$ where g is a link function.

In this chapter, we predict the average number of points in the next 5 games by using the two different link functions (poisson distribution and zero-inflated poisson distribution). This paper explains how these two methods meet the assumptions of generalized linear models.

2.6.1 GLM with Poisson Distribution as Link Function

This research assumes the average number of points in the next 5 games will follow the poisson distribution for the reasons described below.

Characters of Poisson Distribution & Points

1. The number of points in each interval can range from zero to infinity.
2. Points are rare events. The poisson distribution is the distribution of infrequent (rare) events
3. Each point is independent of the other points as the poisson distribution assumes independent events
4. Expected number of points are assumed to be constant as poisson distribution assumes the constant expected number of events.

Mathematical Explanation of GLM with Poisson Distribution

$$Y_i \sim \mathrm{Po}(\mu_i) \tag{2.18}$$

$$P(Y_i = y_i) = (1 - \sigma_i) * \frac{e^{-\mu_i}\mu_i^{y_i}}{y_i!} \tag{2.19}$$

$$\eta_i = x_i\beta \tag{2.20}$$

$$g(\mu_i) = \log(\mu_i) = \eta_i \tag{2.21}$$

where

Y is the number of average points in the next 5 games.
x is the predictors.
β is the coefficients for linear model to fit link-transformed μ.
λ is the expected poisson count.
μ is the expected number of points in the next 5 games.

2.6.2 GLM with Zero-Inflated Poisson Distribution as Link Function

It is reasonable to assume that the distribution follows zero-inflated poisson distribution because there are many players who do not get any points in the next 5 games. Zero-inflated poisson regression categorized 0s into *Always-0 group* and *Not Always-0 group*. Define the probability that the ith observation is not in *Always-0 group* is $1-\sigma_i$ and the probability that the ith observation is in *Always-0 group* is σ_i.

Case 1: Always-0 group

The probability that ith observation is in *Always-0 group* can be predicted by the characteristic of ith observation, so that I can write as

$$\sigma_i = F(x_i\gamma) \tag{2.22}$$

where

F is logit function or probit function.

x is ith predictors.

γ is the coefficients for logit or probit function F.

σ is the probability that ith observation is in *Always-0 group*

Case 2: Not Always-0 group

For *Not Always-0 group*, the positive outcome is predicted by the standard poisson distribution, so the distribution can be written as

$$P(Y_i = y_i) = \frac{e^{-\mu_i}\mu_i^{y_i}}{y_i!} \tag{2.23}$$

where

μ_i is the conditional mean.

$$Y_i \sim \mathrm{Po}(\mu_i) \tag{2.24}$$

The probabilities for zero-inflated regressions are expressed as follows.
For Always-0 group,

$$P(Y_i = 0) = \sigma_i \tag{2.25}$$

For Not Always-0 group,

$$P(Y_i = y_i) = (1-\sigma_i) * \frac{e^{-\mu_i}\mu_i^{y_i}}{y_i!} \tag{2.26}$$

where

Y is the number of average points in the next 5 games.

μ is the mean of poisson distribution.

Overall,

$$P(Y_i = y_i) = \begin{cases} \sigma_i + (1-\sigma_i)e^{-\lambda}, & \text{if } y_i = 0 \\ (1-\sigma_i)\frac{e^{-\lambda}\lambda^{y_i}}{y_i!}, & \text{otherwise} \end{cases} \tag{2.27}$$

$$\eta_i = x_i\beta \tag{2.28}$$

$$g(\mu_i) = \log(\mu_i) = \eta_i \tag{2.29}$$

where

Y is the number of average points in the next 5 games.

x is the predictors.

β is the coefficients for linear model to fit link-transformed μ.

λ is the expected poisson count.

σ is the probability of *Always-0 group*.

Chapter 3

Results & Interpretation

3.1 Linear Regressions

3.1.1 Calculation of Mean Squared Error

Define the following method as a general way to calculate the mean squared error (MSE) of a model. Let this method be ‘Method A’.

Method A

Divide the data set into a training set (80%) and test set (20%). Calculate the mean squared error (MSE) of test set with a model made from training set. This was repeated 10 times and the mean of 10 MSEs was used as MSE of the model.

3.1.2 Ordinary Linear Regressions

1. Normality Assumption

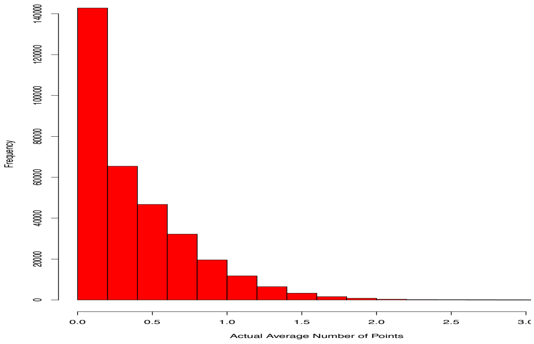


Fig. 3.1 Average Points

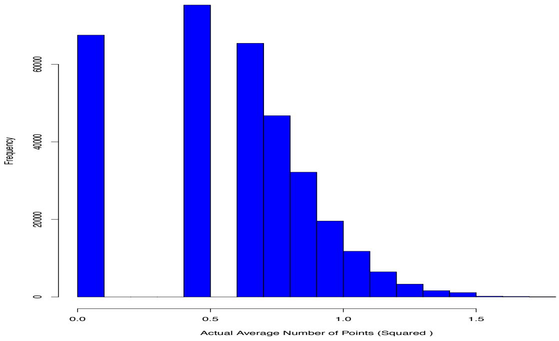
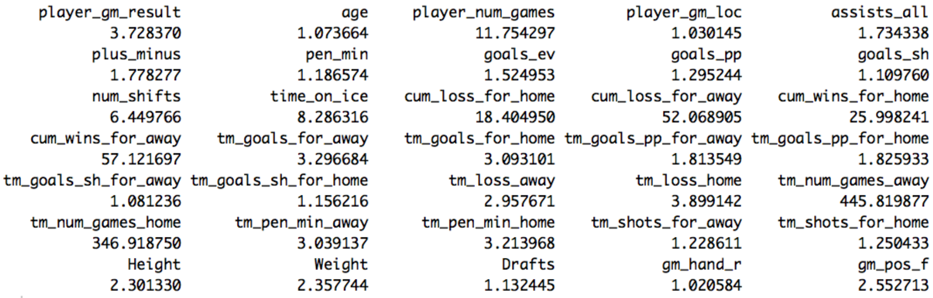


Fig. 3.2 Squared Average Points

Ordinary linear regression assumes that the response variable will follow the normal distribution. It is necessary to check this assumption of ordinary linear regression. Fig. 3.1 is the distribution of actual average points in the next 5 games. This distribution is very right skewed and contains many values of 0. Therefore it makes sense to square transform the response variables since square transformation can keep values of 0. In contrast, log transformation transforms values of 0 to $-\infty$. As Fig. 3.2 shows, square transformation makes the data more normally distributed. Then this research transformed the data back to the original scale and calculated the mean squared error. MSE is listed later in this subsection.

2. Independence Assumption

Fig. 3.3 VIF: Before removing variables



```
    player_gm_result                  age     player_num_games       player_gm_loc         assists_all
            3.728370             1.073664            11.754297            1.030145            1.734338
          plus_minus              pen_min             goals_ev            goals_pp            goals_sh
            1.778277             1.186574             1.524953            1.295244            1.109760
          num_shifts          time_on_ice    cum_loss_for_home   cum_loss_for_away   cum_wins_for_home
            6.449766             8.286316            18.404950           52.068905           25.998241
   cum_wins_for_away    tm_goals_for_away    tm_goals_for_home tm_goals_pp_for_away tm_goals_pp_for_home
           57.121697             3.296684             3.093101            1.813549            1.825933
tm_goals_sh_for_away tm_goals_sh_for_home         tm_loss_away        tm_loss_home   tm_num_games_away
            1.081236             1.156216             2.957671            3.899142          445.819877
   tm_num_games_home      tm_pen_min_away      tm_pen_min_home   tm_shots_for_away   tm_shots_for_home
          346.918750             3.039137             3.213968            1.228611            1.250433
              Height               Weight               Drafts           gm_hand_r            gm_pos_f
            2.301330             2.357744             1.132445            1.020584            2.552713
```

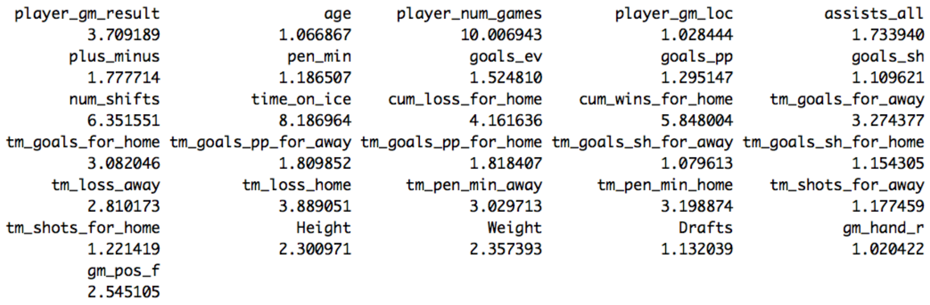
Fig. 3.3 shows that the following variables are highly correlated based on VIF because its variance inflation factor is large(> 10) and they are removed from the consideration for variables for linear regressions.

List of Highly Correlated Variables

- Number of Home Games
- Number of Away Games
- Number of Cumulative Wins for Teams
- Number of Cumulative Losses for Teams

Refit the linear regression model after removing the variables and calculated the variance inflation. The following variables are defined as "**Variables after VIF**". Fig. 3.4 shows that all the VIF values are less than 10 this time. Therefore there is no statistically significant evidence of multicollinearity among the variables in Fig. 3.4 . Now refit an ordinary linear regression to the data set by using "**Variables after VIF**".

Fig. 3.4 VIF: After removing variables

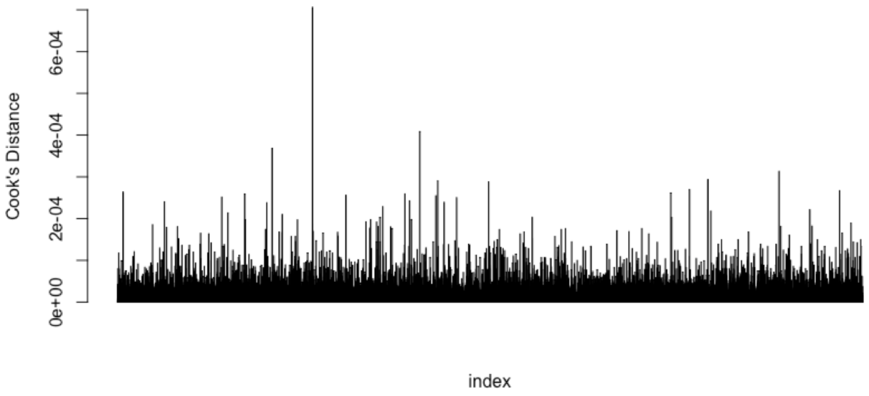


```
   player_gm_result                age   player_num_games      player_gm_loc        assists_all
           3.709189           1.066867          10.006943           1.028444           1.733940
         plus_minus            pen_min           goals_ev           goals_pp           goals_sh
           1.777714           1.186507           1.524810           1.295147           1.109621
         num_shifts        time_on_ice  cum_loss_for_home  cum_wins_for_home  tm_goals_for_away
           6.351551           8.186964           4.161636           5.848004           3.274377
 tm_goals_for_home tm_goals_pp_for_away tm_goals_pp_for_home tm_goals_sh_for_away tm_goals_sh_for_home
           3.082046           1.809852           1.818407           1.079613           1.154305
       tm_loss_away       tm_loss_home    tm_pen_min_away    tm_pen_min_home  tm_shots_for_away
           2.810173           3.889051           3.029713           3.198874           1.177459
  tm_shots_for_home             Height             Weight             Drafts           gm_hand_r
           1.221419           2.300971           2.357393           1.132039           1.020422
           gm_pos_f
           2.545105
```

3. Influential Observations

Cook’s distance finds a potential outlier for the data set. Fig 3.5 shows the maximum value of Cook’s distance is 0.00071. Conventionally, values of more than 1 are considered to be outliers. Therefore there are no obvious outliers based on Cook’s distance.

Fig. 3.5 Cook’s Distance



4. Feature Selection (Backward Stepwise)

Stepwise regression only removed "penalty minute of each player". Refit the OLR to data by using **Variables after VIF** except "penalty minute of each player".

Table 3.1 Results of Ordinary Linear Regressions

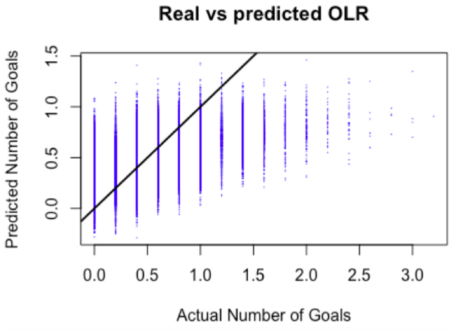
lambda values	MSE
OLR	0.1124
OLR (Squared Transformed)	0.1123
OLR (After VIF)	0.1119
OLR (After Backward Stepwise)	0.1117

5. Results of Linear Regressions

The MSEs (Mean Squared Errors) of ordinary linear regressions are summarized in Table 3.1. MSEs are calculated with ‘Method A’. Table 3.1 shows that the MSE gets a little smaller as variables are removed. However, there is no significant improvement of errors even though the variables are transformed and correlated variables are removed.

6. Predicted vs Actual Number of Points

Since the distribution is right skewed, as the actual number of goals increase, the error gets bigger as the right figure shows.



7. Coefficients of Variables

Fig. 3.6 Regression Coefficients

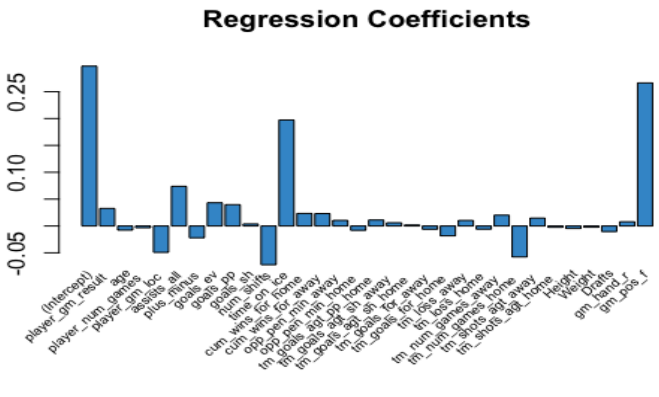


Fig. 3.7 Coefficients of Ordinary Linear Regressions

Variable	Coefficient
Intercept	0.298
Assists	0.0738
Goals at EV	0.0432
Goals at PP	0.0397
Time on Ice	0.197
Position Forward	0.2663
Player Game Location Home	-0.04916
Number of Shifts	-0.0722

Variables with large coefficients after Backward Stepwise1

Fig. 3.6 and Fig. 3.7 show variables with large coefficients.

1. Variables with Positive Coefficients

'Assists', 'Goals at Even Handed', 'Goals at Power Play', 'Time on Ice' and 'Position Forward' are predictors with large positive coefficients. These results mean that if a player scores more goals in the previous games, he is more likely to score in the next games and if a player is a forward, he is more likely to score goals than a defenseman. This makes sense.

2. Variables with Negative Coefficients

Holding other variables the same, 'Number of Shifts' and 'Number of Home Games for a player' are predictors with large negative coefficients. This is an interesting analysis because when holding all the other variables the same, the more home games and shifts a player has, the fewer points the player gets according to a linear regression.

3.1.3 Ridge Regression

1. Results of Ridge Regressions

Table 3.2 Results of Ridge Regressions

λ	MSE
0.1	0.1151
0.05	0.1138
0.01	0.1124
0.005	0.1122
0.001	0.1122
0.0005	0.1123
0.0001	0.1122
0.00005	0.1125
0.00001	0.1121

MSEs are calculated with 'Method A'. Table 3.2 shows that the optimized mean squared error is 0.1121 when $\lambda = 0.00001$. This does not change much from the result of ordinary linear regression. When $\lambda = 0$, lasso regression becomes a OLR.

3.1.4 Lasso Regression

1. Results of Lasso Regressions

MSEs are calculated with 'Method A'. Table 3.3 shows that the optimized mean squared error is 0.1120. We are not able to see an improvement from ordinary linear regression to ridge regression. The lasso regression did not remove any variables from an ordinary linear regression at the end. Therefore it makes sense that the results of lasso regressions are very similar to the results of ordinary linear regressions.

Table 3.3 Results of Lasso Regressions

λ	MSE
0.1	0.1451
0.05	0.1248
0.01	0.1143
0.005	0.1135
0.001	0.1124
0.0005	0.1126
0.0001	0.1121
0.00005	0.1121
0.00001	0.1120

Table 3.4 Results of Elastic Net Regression

λ	α	MSE
0.001	0.01	0.1114
0.001	0.05	0.1112
0.001	0.1	0.1113
0.005	0.01	0.1111
0.005	0.05	0.1113
0.005	0.1	0.1108
0.01	0.01	0.1113
0.01	0.05	0.1115
0.01	0.1	0.1116

3.1.5 Elastic Net Regression

MSEs are calculated with ‘Method A’. Table 3.4 shows that the optimized mean squared error is 0.1108. This is slightly better than other linear regressions. Elastic net regression also removes unimportant variables. When $\lambda = 0.005$ and $\alpha = 0.1$, the following is a list of variables with coefficient 0.

1. Player Number of Games so far that season
2. Number of Team Home Games so far that season
3. Number of Team Away Games so far that season

Elastic net regression performed slightly better than other regularization linear models. They concluded that number of games so far that season are not significant to predict the number of average points in the next 5 games.

3.2 K-Nearest Neighborhood Regressions

3.2.1 Ordinary K-Nearest Neighborhood Regression

For each k value, MSE was calculated with ‘Method A’ . Table 3.5 is the summary of tables of MSEs for different k values. According to the table, When $k = 10$, the k-nearest neighborhood

Table 3.5 Results of Ordinary KNN

K values	MSE
1	0.1334
5	0.1198
10	0.1189
15	0.1203
20	0.1207
30	0.1212

regression has the least MSE and its values is 0.1189. This is not an improvement from linear regression models.

3.2.2 Attribute Weighted K-Nearest Neighborhood Regression

Table 3.6 is the summary table for mean squared error for each removed variable in 1st iteration. None means that none of the variables is removed and all the variables in that iteration were used unless they are removed before. First remove variables with * because they do not decrease the MSE of weighted *k*-nearest neighborhood regression. The error of the weighted *k*-nearest neighbor regression is smaller if these variables are removed. 10 percent of data were used when the error of the model without one variable was calculated because the data set is too large to use for each variable. This process was repeated 10 times and the mean error of 10 times was calculated in each iteration.

1st Iteration

Table 3.6 Results of Backward Elimination Algorithm (1st Iteration)

Removed Variable	Mean Squared Error
None	0.1253
player gm result	0.1257
age	0.1271
player num games	0.1271
player gm loc	0.1247 *
assists all	0.1310
plus minus	0.1251 *
pen min	0.1285
goals ev	0.1291
goals pp	0.1289
goals sh	0.1271
num shifts	0.1278
time on ice	0.1303
cum loss for home team	0.1256
cum loss for away team	0.1284
cum wins for home tam	0.1257
opp pen min away	0.1248 *
tm goals agt pp away	0.1263
tm goals agt pp home	0.1278
tm goals agt sh away	0.1248 *
tm goals agt sh home	0.1240 *
tm goals for away	0.1242 *
tm goals pp for away	0.1263
tm goals pp for home	0.1278
tm goals sh for away	0.1238 *
tm goals sh for home	0.1239 *
tm loss away	0.1248 *
tm loss home	0.1288
tm num games away	0.1261
tm num games home	0.1264
tm pen min away	0.1264
tm pen min home	0.1248 *
tm shots agt away	0.1240 *
tm shots agt home	0.1226 *
player gm loc	0.1247 *
tm shots for away	0.1257
tm shots for home	0.1274
Height	0.1261
Weight	0.1260
Drafts	0.1265
gm hand r	0.1259
gm pos f	0.1256

Now refit the weighted *k*-nearest neighborhood regression algorithm to the whole data set after removing variables in 1st iteration. MSEs are calculated with ‘Method A’ and the whole data set are used. The mean MSE is 0.079.

2nd Iteration

Backward Elimination Algorithm was applied to the data set after variables are removed in the 1st iteration.

Table 3.7 Results of Backward Elimination Algorithm (2nd Iteration)

Removed Variable	Mean Squared Error
None	0.1223
player gm result	0.1230
age	0.1240
player num games	0.1209
assists all	0.1306
pen min	0.1254
goals ev	0.1235
goals pp	0.1267
goals sh	0.1213*
num shifts	0.1201*
time on ice	0.1227
cum loss for home team	0.1217*
cum loss for away team	0.1202*
cum wins for home team	0.1213*
cum wins for away team	0.1218*
opp pen min home	0.1232
tm goals agt pp away	0.1250
tm goals agt pp home	0.1219
tm goals for home	0.1220*
tm goals pp for away	0.1226
tm goals pp for home	0.1257
tm loss home	0.12348
tm num games away	0.1219 *
tm num games home	0.1210 *
tm pen min away	0.1216 *
tm shots for away	0.1230
tm shots for home	0.1239
Height	0.1236
Weight	0.1216 *
Drafts	0.1236
gm hand r	0.1191 *
gm pos f	0.1225

Remove variables with * in Table 3.7 and then refit KNN to the whole data set (Not 10 %). MSEs are calculated with ‘Method A’. The MSE is 0.075. Keep on removing variables after 2nd iteration because MSE decreased after 2nd iteration.

3rd Iteration

Backward Elimination Algorithm was applied to the data set after variables are removed after the 2nd iteration.

Table 3.8 Results of Backward Elimination Algorithm (3rd Iteration)

Removed Variable	Mean Squared Error	Removed Variable	Mean Squared Error
None	0.1222	tm goals agt pp away	0.1202*
player gm result	0.1218	tm goals agt pp home	0.1211*
age	0.1216	tm goals pp for away	0.1227
player num games	0.1223	tm goals pp for home	0.1202
assists all	0.1250	tm loss home	0.1212*
pen min	0.1221*	tm shots for away	0.1224
goals ev	0.1244	tm shots for home	0.1234
goals pp	0.1245	Height	0.1221*
time on ice	0.1270	Drafts	0.1230
opp pen min home	0.1218*	gm pos f	0.1245

Table 3.8 shows a list of important variables after 3rd iteration. Most of the variables are indicators of offensive variables. Power play situation and even handed situations are especially significant in predicting the average number of points in the near future. Draft ranks, time on ice, age, and game result are also strong predictors for the average number of points.

List of Important Variables after 3rd Iteration

- player gm result
- age
- player games
- assists all
- goals at even handed
- goals at power play
- time on ice
- power play team goals at home
- power play team goals at away
- team shots at home
- team shots at away
- position (forward or defenseman)
- drafts

Remove variables with * in Table 3.8. Then refit KNN after 3rd iteration. MSEs are calculated with ‘Method A’. The MSE is 0.088. Stop removing variables because MSE increased after 3rd iteration.

3.3 Random Forest Regression and Boosting Algorithm

1. Results of Random Forest Algorithm

Table 3.9 Results of Random Forest Regressions

Minimum Nodes	Number of Trees	MSE
1%	25	0.1107
1%	50	0.1103
1%	100	0.1029
0.1%	25	0.0820
0.1%	50	0.0816
0.1%	100	0.0814
0.01%	25	0.0791
0.01%	50	0.0694
0.01%	100	0.0675
0.001%	25	0.0813
0.001%	50	0.0707

MSEs are calculated with ‘Method A’. Table 3.9 shows the mean squared error for each minimum number of node and each number of tree. As the number of trees grows up, MSE decreases. The best MSE is 0.0675. Random forest regression gets the best result when minimum nodes is 0.01%. It overfits the training data if the minimum node is smaller than 0.01%. It underfits the training data if the minimum node is bigger than 0.01%.

2. Variable Importance

Variable importance was calculated based on Breiman’s method when minimum node is 0.01% and the number of trees is 100. Fig.3.8 shows that assists, time on ice, goals at power play, goals at even-handed, position, number of shifts and draft ranks are the most important features to predict the number of points.

3.3.1 Gradient Boosted Regression Tree Algorithm

Results of Gradient Boosted Regression

Since it is computationally very expensive to use the whole data set, the research bagged 10% of data 10 times and calculated the prediction error.

Table 3.10 shows that GBM performs best when shrinkage is 0.1 and the portion of minimum nodes is 0.01%. Refit the GBM to the whole data set by using shrinkage = 0.1 and the portion of minimum nodes is 0.01%.

MSEs are calculated with ‘Method A’. The MSE is 0.1109 when the whole data set are used.

Fig. 3.8 Important Variables Based on Random Forest

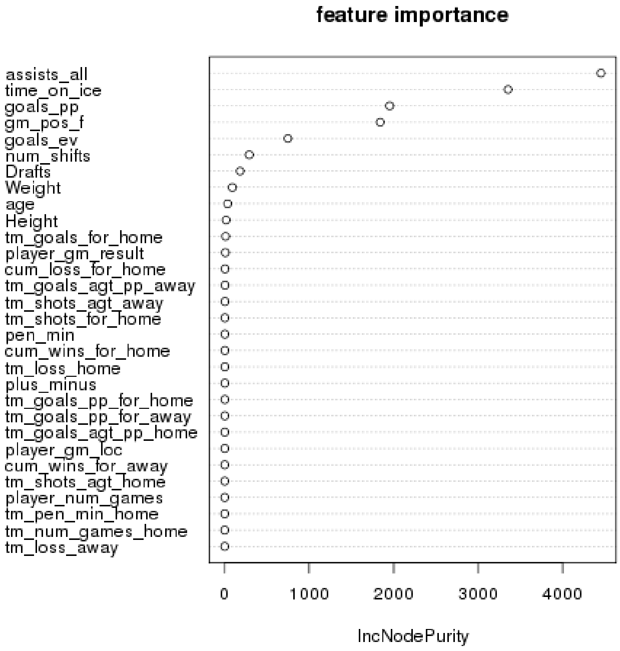


Table 3.10 Results of Gradient Boosted Regressions (Squared Error)

Shrinkage	Minimum Nodes	Number of Trees	Prediction
0.1	0.1%	3000	0.1133
0.1	0.01%	3000	0.1112
0.01	0.1%	3000	0.1137
0.01	0.01%	3000	0.1136

3.4 Neural Networks

3.4.1 All Variables

Results of Neural Networks

MSEs are calculated with ‘Method A’. Table 3.11 shows that random forest regression performs better than neural network regression. Since neural network regression tends to overfit data, it is reasonable to use PCA-transformed variables.

3.4.2 PCA-transformed Variables

Variance Explained by Each Dimension

Table 3.11 Results of Neural Networks (All Variables)

Number of Hidden Nodes	MSE
1	0.1132
2	0.1101
3	0.1091
4	0.1107
5	0.1096
6	0.1092
7	0.1092
8	0.1106
9	0.1091

Fig. 3.9 Variance Explained (Figure)

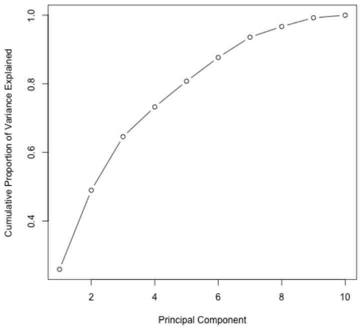


Fig. 3.10 Variance Explained (Table)

Dimensions	Cumulative Proportion	Individual Portion
1	0.2796	0.2796
2	0.5072	0.2276
3	0.6754	0.1682
4	0.7712	0.0958
5	0.8540	0.0828
6	0.9273	0.0733
7	0.9632	0.0359
8	0.9915	0.0283
9	1.000	0.00852

Fig.3.9 and Fig.3.10 show the proportion of variance explained by each dimension. This research uses 7 and 8 as dimensions because these two dimensions explain more than 95% of the cumulative portion of variance.

Results of Neural Networks (PCA)

MSEs are calculated with 'Method A'. Table 3.12 shows that there is no improvement from random forest regression models in each dimension and different number of hidden nodes even if PCA is used to avoid overfitting data.

3.5 Generalized Linear Models

Results of GLMs

MSEs are calculated with 'Method A'. Table 3.13 shows that zero-inflated poisson model

Table 3.12 Results of Neural Networks (PCA)

		Hidden Nodes								
		1	2	3	4	5	6	7	8	9
Input	7	0.1156	0.1144	0.1127	0.1132	0.1128	0.1158	0.1138	−−	−−
Nodes	8	0.1133	0.1129	0.1119	0.1115	0.1125	0.1116	0.1112	0.1112	−−
	9	0.1132	0.1101	0.1091	0.1107	0.1096	0.1092	0.1092	0.1106	0.1091

Table 3.13 Results of GLMs

Link Function	MSE
Poisson Distribution	0.2256
Zero-Inflated Poisson Distribution	0.1022

performed much better than regular poisson model. This makes sense because Fig 3.1 shows that the number of points have values of 0 in many cases. Therefore, zero-inflated poisson model is more appropriate than regular poisson model. However, it did not perform better than a random forest or KNN.

Chapter 4

Discussion

4.1 Conclusion

4.1.1 Overall Approach and Conclusion

This paper has devised several machine learning approaches to predict the average number of points in the next 5 games. This paper has examined player data, team data and opponent data. This paper has implemented several regressions such as random forest regressions, *k*-nearest neighborhoods and generalized linear models. Random forest regression performed best among regression models made in this research. Random forest regression calculated the average number of points in the next 5 games with mean squared error of 0.0675 with the optimized parameters. This is about a 70% improvement compared to mean squared error of a baseline model that predicts that all the players get the average number of points in the next 5 games.

4.1.2 Noticeable Findings

Random forest regressions and *k*-nearest neighborhood regressions are two of the best models. Based on these algorithms, assists, time on ice, goals at power play, goals at even-handed, number of team shots, position and draft ranks are significant features to predict the number of points for a player.
The following is a list of noticeable findings.

1. Team shots are more significant than team goals to predict the number of points for a player according to *k*-nearest neighborhood regressions. In ice hockey games, it is hard to predict points because a team sometimes scores 4 goals with 20 shots and a team sometimes scores no goals with 40 shots. However, the distribution of shots in

a game is more stable and predictable. Therefore it is reasonable that team shots are more significant predictors than team goals.

2. Assists are more significant features than goals to predict the average number of points. This makes sense because players usually provide assists more frequently than they score goals.
3. Weight and height are not very significant predictors. Most of the NHL scouts consider weight and height to be important predictors, but if other variables held the same, weight and height are not as significant as past points in predicting the number of points in the future.
4. According to random forest regressions and KNN, opponent's data is not as important as team's data.
5. According to important feature selections by random forest regression, team data is not as important as player data. Player data is much more significant to predict the number of points in the next 5 games.

4.1.3 Model Use

NHL teams would use this model to figure out which players they should use to increase the number of points in the next several games. This model would be especially useful in playoffs because goal production in the near future is very important. Moreover, NHL teams would be able to figure out which players could potentially increase the number of points in the short term during a season.

4.1.4 Further Research & Directions

There are several possible further studies to be done.

1. Time Series
   This research did not consider time series of data. Further research needs to be done in order to understand how goal production in ice hockey has changed over time. Especially the changes of rules and goalie equipment may have affected the production of goals.
2. Goal Production in Different Situations
   This research predicted the total number of points. However, it is reasonable to try to predict the number of points in even-handed, power play and short-handed situations separately.

3. Line Combination
   Even though this paper used the models that are robust against correlation, it did not explicitly consider the line combinations of players. A model that is able to consider line combinations of players may increase the accuracy.

Several studies have been done about prediction of points in ice hockey. However, most of the papers predicted the number of points in one season, not the next several games. Moreover, each study uses a different model and there is still significant room for advancement in research on points in the next few games. A more sophisticated model would help NHL teams to figure out which players they should acquire and sell.

References

[1] Abdi, H. and Williams, L. J. (2010). Principal component analysis. *Wiley interdisciplinary reviews: computational statistics*, 2(4):433–459.

[2] Bishop, C. M. (2006). Pattern recognition. *Machine Learning*, 128:1–58.

[3] Breiman, L. (2001). Random forests. *Machine learning*, 45(1):5–32.

[4] Deokar, S. (2009). Weighted k nearest neighbor.

[5] Faraway, J. (2002). Practical regression and anova using r: http://cran. r-project. org/doc/contrib. *Faraway-PRA. pdf*.

[6] Friedman, J., Hastie, T., and Tibshirani, R. (2010). Regularization paths for generalized linear models via coordinate descent. *Journal of statistical software*, 33(1):1.

[7] Kolenikov, S., Angeles, G., et al. (2004). The use of discrete data in pca: theory, simulations, and applications to socioeconomic indices. *Chapel Hill: Carolina Population Center, University of North Carolina*, pages 1–59.

[8] Liaw, A. and Wiener, M. (2002). Classification and regression by randomforest. *R news*, 2(3):18–22.

[9] Matan, O., Kiang, R., Stenard, C., Boser, B., Denker, J., Henderson, D., Howard, R., Hubbard, W., Jackel, L., and Le Cun, Y. (1990). Handwritten character recognition using neural network architectures. In *Proceedings of the 4th USPS advanced technology conference*, pages 1003–1011.

[10] Mohan, A., Chen, Z., and Weinberger, K. Q. (2011). Web-search ranking with initialized gradient boosted regression trees. In *Yahoo! Learning to Rank Challenge*, pages 77–89.

[11] Silver, N. (1905). We're predicting the career of every nba player. here's how.

[12] Srivastava, N., Hinton, G. E., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958.

[13] Venables, W. and Ripley, B. (1999). Generalized linear models. In *Modern Applied Statistics with S-PLUS*, pages 211–240. Springer.