Predicting NHL Players Points Using Regression

University of Denver Data Science Capstone

Adam Brenner

Introduction:

The National Hockey League was formed 103 years ago in Montreal, Quebec, Canada. Today the league is comprised of 32 teams, 25 of which are in the United States, and 7 reside in Canada. The rise of sports analytics had largely passed the NHL by, even though it has kept statistics since its inception, deeming team personnel decisions a skill that only a few had mastered and without the need for analytical intervention. However, in recent years, more teams have adopted on-ice and off-ice analytics for decision making. The goal of this research paper is to try and predict NHL players total points for a whole season using machine learning techniques, such as Decision Trees and Random Forests, juxtaposed with linear and ridge regression. With the implication of improving the NHL’s standing among other professional sports, within the team decisions and fantasy sports landscapes.

With the recent law changes regarding professional sports gambling, fantasy sports betting has seen a gradual increase, with many experts claiming it will only continue to rise. In 2019 the fantasy sports sector generated $7.2 billion USD in the United States alone, and with the availability of smart phones and general improvements in infrastructure, the population base should only continue to grow. Thus, the accurate predictions of player stats or teams wins and losses could hold immense value. It essentially already does, with sports betting giants FanDuel and DraftKings each having more than $1 billion in revenue in 2020, as well as ESPN and IBM teaming up for a fantasy football app in 2020. The key here is that hockey seems to be behind all the other major pro sports in many analytical categories.

Total points are the aggregation of goals and assists over a season per player. Goal and assist accumulation, or creation, are sometimes hard to understand for the uninitiated. Unlike other pro-sports, such as baseball, the puck and players are always in motion. Goals are awarded when the puck completely crosses the line between the two goal posts and under the crossbar, either from a player shooting the puck with their stick, or from a deflection off any part of the player, with a few conditions. An assist is awarded to two scoring team players, who touched the puck in any way on its journey to a player that eventually scores a goal, if the puck wasn’t controlled by the opposing team before the goal is scored. For example, with three different players, if player 3 passes the puck to player 2 who passes the puck to player 1 who scores before any opposing team players touch the puck, players 3 and 2 are awarded assists, and player 1 is awarded a goal. Predicting these goals and assists is a difficult task, not only because of the speed of the game, players and pucks routinely reach speeds of 20 – 25 mph and 90 – 95 mph respectively, but also due to “luck”. Out of all the major professional sports, luck is seen as contributing the most to hockey. This is due to the design and pace of the game simply introducing more randomness into the results, as well as the NHL having more league parity, with many skilled players not being confined to a few teams but spread throughout the league.

Data Gathering:

The first step in any data science project is to gather relevant data. This was done through the National Hockey Leagues Application Programming Interface or API. The NHL stats themselves are numerous and detailed, even documenting single games and their events and stats. Unfortunately, this is a terribly documented API by the NHL itself, luckily a few have taken it upon themselves to open and update a gitlab located in the sources below, that documents multiple endpoints and their options. Thus, the first step in this process was to familiarize oneself with the API and what the requests to each endpoint returned. After such, came acquiring stats that seemed relevant to the project and how to store them.

Since the project is about players total points, it only seemed natural to gather as many players from as many teams with their respective stats. This was done by gathering active NHL teams between 2010 and 2018, which was an arbitrary range, thus a team stats list was created that included team name, id, and season. The next step was gathering players unique ids that were on each teams’ respective rosters for each of those seasons above. After, the main data gathering step could begin. Using the unique player ids and the people and stats endpoints through the API, python dictionaries were created to hold a player’s, name, position, country, birth date, id, height, weight, and shoots, which simply means if a player uses a left-handed or right-handed stick. Then for each specific player, adding their respective stats per season. This included, goals, assists, shots, games, time on ice, penalty minutes, hits, shot percentage, blocked shots, plus minus, shifts, total points, and team.

The process above was simply a means of gathering as many players as possible that played in the NHL between 2010 and 2018. However, the purpose of the project is to predict total points, so it didn’t make much sense to have players that had only played one season or who had simply since retired before 2018. Thus, the stats were reduced to players that had played at least three seasons and one hundred and twenty games. If a season wasn’t present for a player simply because they had started after 2010 but were still active, that season was added with all zeros in the stats category simply for the ease of creating a pandas data frame after. Each season was contained within a dictionary for a specific player. Thus, these dictionaries were essentially flattened and added as new columns to the data frame, resulting in an initial dataset of 5,349 players and 112 columns, shown below.

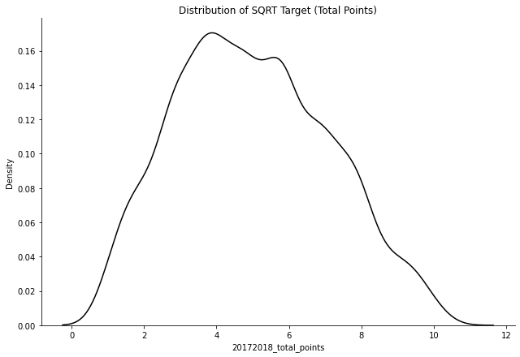
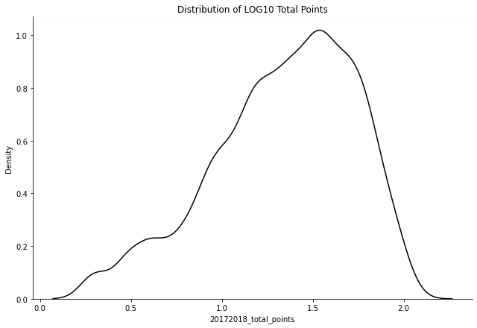
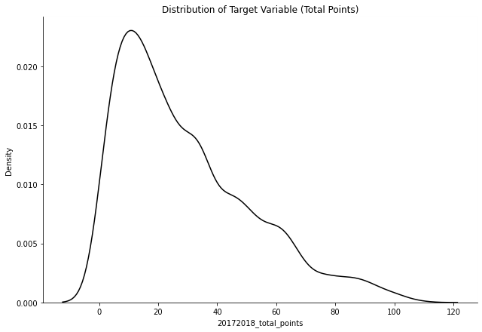


Data Preprocessing:

After gathering the data, it wasn’t quite ready to be thrown into a model, as there were categorical and numerical columns, as well as data that had to be transformed or simply dropped. The data cleaning steps consisted of splitting the time on ice column, stored as total minutes and seconds together. This was done by splitting the two components and creating their own specific columns, one for time on ice in minutes, and one for time on ice in seconds. Next came creating a function that would take the height of a player and return only the inches, thus 6’ 1’’ would simply be 73’’. Next came dropping the columns that were deemed not relevant, this consisted of team, id, name, and birth date. In future endeavors, the team and name columns will be used for assigning predictions back to that specific player as well as grouping players by their teams to identify which has the best outlook for the season, but I digress. Seasons before 2014, and their attached stats were also dropped, the reasoning for this was the number of zeros that were contained in those years, indicating that many players in the dataset had started after the 2010 season. Using the sklearn LableEncoder, the position, country, and shoots columns were encoded, turning categorical data into numerical. Finally, any players that hadn’t scored a point in the 2015-2016, 2016-2017, or 2017-2018 seasons were dropped, as it was more than likely that those players had retired. After such thorough data cleaning we were left with a data frame of 3,374 players and 58 stat columns, discussed at the end of the paper.

Data Distributions:

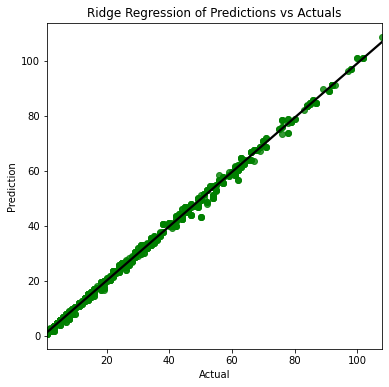
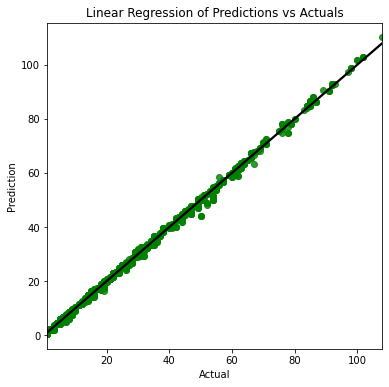
With cleaned data, it was time to investigate the distributions of the feature and target variables, identify if any scaling needed to be performed, and check for outliers. Plotting the target variable of 2017-2018 season total points, it was clearly heavily right skewed. After applying a logarithmic transformation, it simply switched the data to a left skew. Thus, a square root transformation was applied, and while it wasn’t perfect, it looked much more normal than the original or logarithmic. This process is displayed below.



This distribution is important as linear regression was going to be applied, and normally distributed data is one of the many assumptions it makes. Therefore, checking the rest of the variable’s distributions revealed much of the same right skewed data. Luckily, the height and weight columns were normally distributed, which reinforced that we had good data, as these have been shown to follow a bell shape curve when enough samples are applied. Running off our target variable assumption, that a square root transformation works best, it made sense to apply the same to the goals and assists columns per season, simply due to total points being the aggregation of goals and assists. Much of the same followed for the variables, shots, penalty minutes, hits, shot percentage, and blocked shots. The time on ice variable was left as is as it seemed to follow a normal distribution. Checking for linear relationships between the feature and target variables revealed that some certainly had one, and others did not, but more variables were in the former category. As far as outliers were concerned, descriptive statistics revealed none, only the normal standard deviations in the height and weight columns. With all of that completed it was time to apply the data to some models and interpret results.

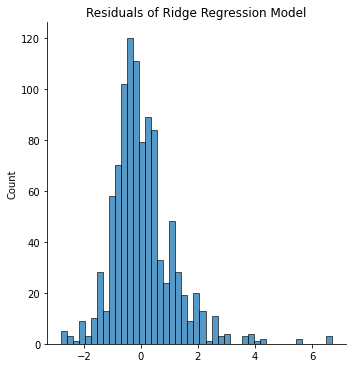
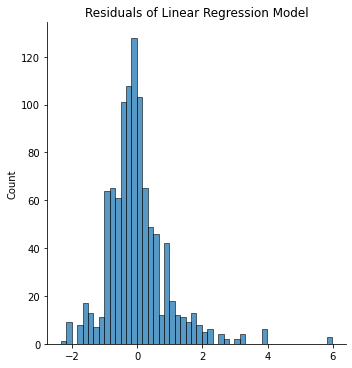
Model Building and Evaluation:

Before applying any models, a feature and target set was created. The feature set consists of all variables except the 2017-2018 total points, which comprised the target set. As explained above, a square root transformation was applied to many of the feature set variables as well as the target variable. A MinMaxScaler was also applied to both. The MinMaxScaler, rescales all variables between the range of 0 and 1. This was used due to the large discrepancy in values between the different features, and the fact that linear regression uses a gradient descent optimization method impacted by large variable inequalities. For example, the weight and time on ice in minutes columns each had values in the three-digit range while goals and assist rarely reached the fifty mark. After scaling, the data was split into training and testing sets. A linear regression model was fit to the training set and predictions were made on the testing set. Once these predictions were made, they had to be rescaled, this involved inverse transforming the MinMaxScaler, and then squaring the predictions. The same steps were taken for the y-testing set as this would be used to measure the performance of the model, which involved the root mean squared error (RMSE) as well as the mean absolute error (MAE). These will be discussed at length later. The linear regression model returned an RMSE of .931 and an MAE of .636. Essentially meaning that every prediction was within, on average, .636 player total points from the actual. Which while impressive, certainly could be showing some overfitting taking place.



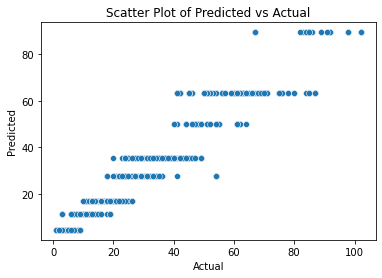
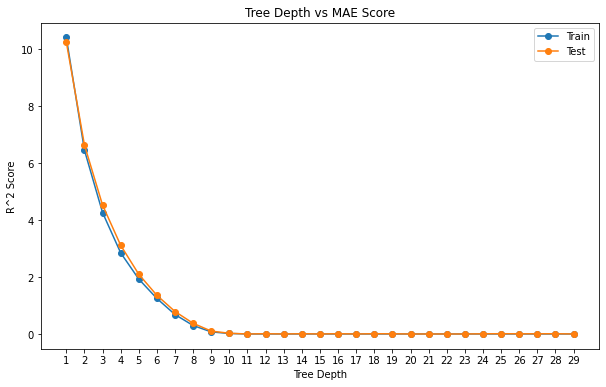
The mean absolute error of a model represents the distance from predicted points to actual points. . For instance, on the graph above, if you were to draw a line from any point straight up or down to the regression line, that equates to the error distance for that point. You add up all those error points and that is your MAE. This is probably the easiest error score to understand as it simply means how far away, on average, are the predicted values from the actual values. The root mean squared error is also commonly used. Square rooting the mean squared error forces the errors back into the units of the target variable. However, due to the initial squaring these errors put more emphasis on bad predictions or outliers. Therefore, it is useful to use when large errors would be detrimental, such as if predicting when a nuclear core could meltdown. In the scope of this project both the RMSE and MAE will be used for evaluating models.

A ridge regression model was trained and fitted next. Ridge regression is typically used when multi-collinearity could be an issue. While there wasn’t much evidence of multi-collinearity among these variables, it seemed a useful practice just in case. The model returned an MAE of .77 and an RMSE of 1.097, further showing that multi-collinearity wasn’t an issue, as there is a miniscule difference between the error scores returned from the linear regression. The residual plots of each model displayed normality of the residuals, which when combined with the other evidence, made the validity of the models seem appropriate.

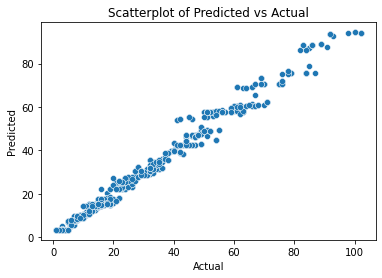
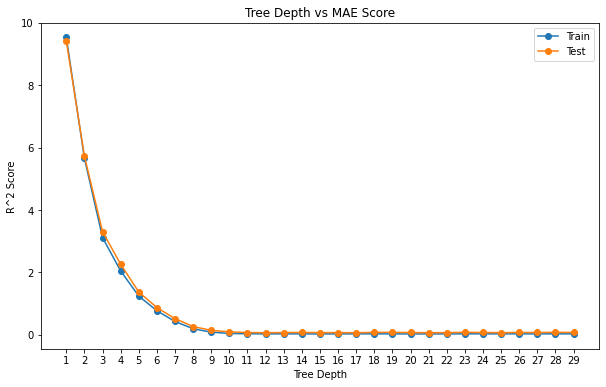


Tree based models of decision trees and random forests were chosen due to their ease of being explained and understood. Essentially a decision tree uses a series of if-then rules to make predictions. It determines features to split on, and then determines when to stop splitting. For features to split on, it determines which feature will create the best resulting dataset. For instance, it might split on the 2014-2015 season games feature, which would split the dataset into players that have high total points and players that have low total points. Random forests create an ensemble of decision trees simultaneously from sampled data. However, random forests randomly split on features while making predictions, then all these predictions are combined, like an average, hence the ensemble methodology. Decision trees are typically good for complex, non-linear data, while random forests excel at handling high dimensionality data.

To train and fit the decision tree and random forest models, another set of features and target were created, however, this time the data used was from the original dataset, meaning no scaling or transformations had taken place. As discussed above, most of the variables were right skewed, which both models are equipped to handle. A training and testing set were created and then trained and fit on the decision tree model. Predictions were made, and it was immediately clear that overfitting had occurred as the MAE was nearly zero. This wasn’t a surprise however, as many tree-based models will overfit the data if they are allowed to grow to a full tree. Thus, some hyperparameter tuning was required, a for loop was used and different models were created at different tree max depths, while MAE and RMSE scores were recorded for each. This resulted in the following graph for MAE. It is clearly evident that as the tree grows the MAE gets better and better, eventually reaching near zero. However, an MAE of near zero on the training set simply means the model will underperform when presented with unseen data. Another decision tree model was trained and fit, however, this model had a maximum depth of 3. The resulting model had an MAE of 4.528 and an RMSE of 6.365 and produced the scatterplot below, but the important part is displayed little to no overfitting.



Following the same procedure as above, a random forest model was fit and trained on the unscaled split data. Again, the model displayed overfitting, but not as heavily as the initial decision tree model, simply due to how random forest models are constructed. The same steps were followed in plotting different model depths and their resulting MAE and RMSE scores. Eventually, a random forest model with a max depth of 4 was fit on the data. The resulting predictions produced an MAE of 2.270 and an RMSE of 3.299, which in the context of this project, would indicate that each prediction was, on average, 2.27 points from the actual value. With both models tuned to avoid overfitting and the resulting MAE scores, it was clear that either could be used as viable models. However, with larger datasets and more variables the hyperparameter tuning could become computationally expensive, especially if other model parameters were used in conjunction with max depth.



Insights:

Each of the four models used showed promising results for the prediction of total points for an NHL player over one season. The linear regression model did especially well, however, the data had to be transformed via square root and then scaled with a min-max scaler. Then, unscaled and squared to make predictions. Also, data distributions had to be investigated to know which variables would react well to the square root transformation. Antagonistic to that, the decision tree and random forest models were trained on unscaled data with some preprocessing. However, these models had to be tuned to avoid overfitting, which, with the hyperparameters chosen in this project, didn’t take much time, but that doesn’t mean it couldn’t. This process could aid in the building of NHL team rosters but will never fully replace human intuition. The route ahead would be using the NHL API to bring data in by game instead of season and use the last ten games or so to make predictions on player points over the next five games. That would allow people building fantasy teams the ability of choosing the best players to start and the best players to sit, but as discussed above, the luck of the game could obscure or enhance such results.

Sources:

NHL API Documentation:

<https://gitlab.com/dword4/nhlapi/-/blob/master/stats-api.md#teams>

Géron, Aurelien. *Hands-on Machine Learning with Scikit-Learn and Tensorflow: Concepts, Tools, and Techniques to Build Intelligent Systems*. O'Reilly Media, 2019.

*Learning Python*. O'Reilly Media, 2009.

Dataset Description:

Position: describes which position a player plays. Possible values include, left wing, right wing, centerman, or defenseman

Country: describes where a player was born

Height: player height in inches

Weight: player weight in pounds

Shoots: left or right handed shooting style

2014-2018 Goals: how many goals per season the player scored

2014-2018 Assists: how many goals the player aided in

2014-2018 Shots: how many shots per season the player had

2014-2018 Games: how many games per season the player played in

2014-2018 Penalty Min: how many penalty minutes the player accrued over the season

2014-2018 Hits: the amount of checks the player had per season

2014-2018 Shot Pct: how many shots the player took that either scored or the goalie stopped

2014-2018 Blocked: how many shots the player blocked per season

2014-2018 Plus Minus: if the player was on the ice when his team goal was scored, it equates to plus one. If the player was on the ice when the opposing team scored, it equates to minus one

2014-2018 Shifts: how many shifts the player had per season

2014-2018 Total Points: the total goals and assists the player had per season