NHL Performance Predictor

Pin Fan, Zach Gruwell, Noah Lahr, Mark Weinhold

Abstract - In sports there is a market for gambling and predicting statistics such as a winner. This paper will be comparing the implementation of four different machine learning algorithms towards predicting the winner of and NHL hockey game. The three different algorithms we are comparing are Linear Regression, Logistic Regression, Neural Networks using Tensorflow, and an implementation of Neural Networks from scratch. We trained and evaluated our data with different training sets for each algorithm trying to maximize the prediction rate and efficiency of each model. With past research outputting a 60% prediction rate, we are trying to improve that by testing various different models using different training sets and predictors. We have come to conclusion that Linear Regression will not be a appropriate machine learning approach to predicting a winner as the regressor is binary, after finding this out we decided to try and use Logistic Regression instead. Concluding on Neural Networks we found that is the most appealing when utilizing machine learning techniques to predict sports, this is due to the benefits of technology.

1. Introduction

Within sports, predictability is often an attractive affinity, as many participants in sports gambling or fantasy sports would find such predictability quite enticing. Ice hockey compared to other sports, has not received as much attention by sports analysts and staticians. This is surprising given that the NHL is the 4th most watched sport on National Television in the United States. With the NHL being founded over 101 years ago, there is a lot of data surrounding them and since this data is freely available, this project seemed feasible. “In hockey, five players and a goalie per team are on an ice surface and play for a total of 60 minutes. The goal of the game is to put a rubber puck into the opposing team’s net using a 1.5 to 2m long stick made of wood or a composite material. The team who scores the most goals in a game is the winner. In the regular season, if a game is tied after 60 minutes, the teams play an extra 5 minutes of sudden death overtime and after that the game is decided by a shootout. In the playoffs, after 60 minutes, additional 20 minute overtime periods are played until a team scores.” (International Journal of Computer Applications (0975 –8887).

The goal of this project is to be able to predict whether the home or away team is going to win. Sports outcome predictions have been popular throughout history, with certain sport such as baseball and horse racing gaining popularity in the data mining field for the ability to predict outcomes. A similar audience has not been seen trying to predict hockey outcome, yet we believe that this is possible.

For our project we have decided to utilize two different machine learning approaches to predict different outcomes in the National

Hockey League. The two different approaches we have decided to take are linear regression and neural networks. We have decided on these two because they take completely different approaches on predicting our outcomes- one being from a statistics oriented background, and the other emerging as a newer method from the machine learning side of prediction.

The initial data set we found included 9 .csv files covering several areas of NHL statistics, including data on individual games, players, teams, and plays. After preprocessing the data, our final data set contains 27 columns of 7441 data points each. 2 of these columns are used for team and season identification, and 24 columns are usable in data mining, excluding the column for winning results, giving a grand total of 186,025 total data points to process.

With the amount of predictors and the depth of our predictors our model uses, we are suspecting our model to be a much more accurate prediction than the already developed models. In Section 2 we will be talking about work related to ours, in Section 3 we will discuss how we accomplished our predictions, Section 4 will cover verifying our hypothesis while explaining the experiments we performed. Lastly, Section 5 cover our conclusions on each approach, and recommended future work.

2. Related Work

An independent researcher who goes by the name Gianni Pischedda from Uxbridge, London has tried predicting outcomes before in the National Hockey League but for a single team; the Ottawa Senators, in this study Pischedda claims his model output a 60% prediction rate. Pischedda used 16 predictors in his study compared to our 24. Pischeddas predictors consisted of the date and location of the game along with the opponent the Senators were playing, his dataset also used the total Goals For, total Goals against, the goal differentiation between the teams, the power play percentage along with the penalty kill percentage; the shot per game percentage and the save per game percentage; the current win streak of the teams along with their conference standing; and finally the Fenwick close %, the PDO, the 5-5F/A, and if it was a win or loss for the Senators. With our project consisting of more predictors as well as more in-depth predictors, we are trying to surpass Pischeddas prediction rate.

Our research differs from Pischeddas by being more robust, our model can predict for any team in the NHL rather than just the Ottawa Senators. We also differ from Pischedda by using Linear Regression rather than decision trees, with the amount of input we have Linear Regression will suit our model more. Pischedda used a 65/35 for his train/test data partition sizes, we have decided to use 66.6/33.3 for ours. Our data was collected over a period of six years, Pischeddas data derived from only one year, which was 2012-2013, the year before he conducted his research. We are similar to Pischedda and his research by trying to implement Multi-layer Artificial Neural Networks as well.

We haven’t yet seen someone trying to predict the NHL by Linear Regression because of it being solely numeric form of prediction, where as most regression models are based off of a single variable. For example, they would use total points scored in the game because their data is sufficient, our model will focus on many more variable, all being weighted, giving us a more accurate model. Compared to Pischedda, we will be implementing Neural Networks in 2 different ways rather than one, we will be implementing neural networks through Tensorflow with keras, along with by hand. As Keras is quite a high level API, much of the mathematics associated with neural networks is simplified to a great degree. This will allow us to compare both neural networks and determine which one has a more accurate prediction rate and why.

3. Proposed Work

3.1 Linear Regression

1) First, after collecting Data from 9 .csv files we started to preprocess it. During this process we determined which predictors are required to implement our predictions.
2) After this we took the data and put it in a single excel file- thus making it easier to upload into R studio.
3) After preprocessing the data set, we started hypothesising many different models for each of our outcomes. We have decided to use best model selection to determine which predictors are most important based on R^2_{adj} values. The model selection is both forward and backward selection based off of both AIC and F-statistics.
4) Once we were done with this process we took the strongest model- the one with the highest R^2_{adj} value) and begin testing and error analysis. We developed many models in the search for the best model for a prediction of the number of points scored in an NHL game.

   During the experimental phase we each hypothesized which predictors were going to be the most important. Noah Hypothesized that the goals would overrule most models and that they would be the only important predictor. Zach (having a background in hockey) hypothesized that predictors such as hits and shots for either team would be more important.

3.2 Neural Networks

The implementation of neural networks required a thorough interpretation of the dataset which had been chosen. Through the process it was realized that a few features within the dataset was negligible; the dataset provided statistics such as the season as well as a game identification number which often does not seem to be resoundingly influential to the games recorded. Regardless, much of the dataset was utilized to provide a comparison between a regression approach and a neural network approach.

While we were in the process of preprocessing the dataset, our project was forced to downsize due to the difficulty of implementing such ambitious objectives due to the constraint of time. As such, our implementation utilized the downloaded dataset to predict whether the home team would have a higher probability of being victorious in a given hockey game. Thus it was determined that the implementation of the neural network would be one that which would not account for the game identification numbers and season provided in the dataset, as it would not be easily integrated into the dataset.

The approach to neural network was decided to be an implementation of a regression model in Keras that would train from a portion of the dataset, and subsequently from a test dataset predict accurately without much error. The task at hand was handled through an utilization of Google's Tensorflow API as well as an attempt at a custom-made neural network. During the design stages of the project, much effort was concentrated in selecting the most optimal method in training and testing a model, and curiosity in the performance and efficiency of a homemade neural network was sufficient that work then commenced on it.

Much of the implementation in both the Tensorflow and native implementations followed the idea that home wins is a crucial factor in the determination of a victor. Thus, over 23 features of various statistics were used to predict the outcome, and as a result we derived a hypothesis from which a project began to take shape.

Throughout the process of implementing the neural network, however, we found that much of the underlying mathematics regarding machine learning and neural networks was blanketed due to the higher level nature of Tensorflow with Keras. Due to such, we developed the idea of a homemade neural network as an alternative approach. This homemade neural network would handle the same question posed during the commencement of the project, while also allowing us to have better access to lower level algorithms, such as the back propagation method, error calculation, and loss function. However, with this ambitious endeavor arrived many difficulties, as the implementation of such a custom network proved rather challenging. Ultimately, the attempts at measuring value loss and accuracy appeared to be a simpler task utilizing Tensorflow when compared to a custom-made neural network.

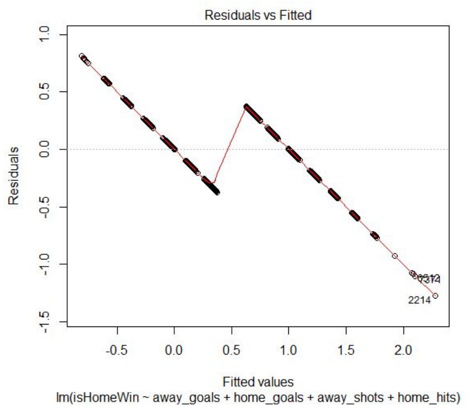
The table below summarizes the approaches utilized within this project:

ML Approach	**APIs**
Regression	RStudio
Neural Network	Tensorflow + Keras, from scratch

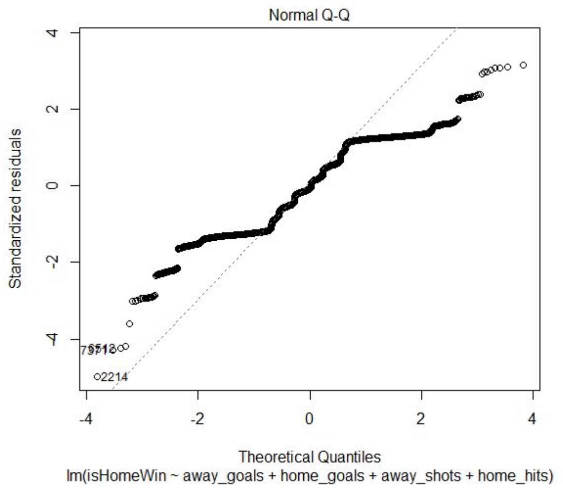
4. Experimental Evaluation

4.1 Experimentation With Linear Regression Models

In terms of experimentation, the data had our hands tied with what we were able to do. The variable that the predictors were regressed on was binary which limits the amount we are able to manipulate the model. Typically the residuals are key in finding how well your model fits the data set and adjustments can be made to the regressor in order to randomize the residuals. With our regressor being binary, if we were to take the log of it or square it, it wouldn't change it at all. Another transformation we were unable to implement was y= 1/y. With our y's being only 0's and 1's that would become undefinable. The only real change we were able to make is to change how many, and which predictors we were to use which after we find our best model using model selection, would be taking a step backwards by changing the best model. The residual plot for the best model is below:



The residual plot is not how it should look, but the reasoning behind it is that the regressor is binary, so there can only be two values. The residuals are hovering around .5 because of the regressor as well. Along with a residual plot, a Normal QQ plot is also an important way of measuring the normality of a model compared to data. The Normal QQ plot is as follows:



This follows a large part of the normal line, yet levels off at the top and bottom showing that a large part of the data is not normal. The model is able to stretch around a large part of the data, yet the data is not distributed normally enough for the model to have a high accuracy rate. The accuracy of this model in terms of R^2_{adj} is .7334. This means that 73.34% of the data can be explained by the best fit model. R^2_{adj} also factors in the numbers of predictors used in the model.

Our initial intent was to also predict the number of points scored in a game, but the data showed high multicorrellation which lead to many predictors being used on the model. Once the model was made the R^2_{adj} value was under .05. We concluded that we would be unable to predict number of points scored with this data set.

4.2 Experimentation With Logistic Regression Models

While time was limited with our implementation of Logistic regression, we felt that it summarized the data better than a multiple linear regression model. The experimenting that we were able to complete within the given time was to create a best model using the same predictors as in the linear regression model, and using it for model validation. We divided our data into thirds- using two thirds to train the model and one third to access the error.

To do this we first had to find the optimal cutoff that would differ the prediction between 0 and 1. We calculated our cutoff to be .01 meaning that anything below would call under the 0 category and anything above would be considered a one.

Using this cutoff value, we ran a misclassification test which would show the percent of the data that was misclassified. Our percent was 0 which is a red flag we did not have time to access.

4.3 Neural Networks

In our development of our Tensorflow neural network, we utilized various supplemental libraries such as pandas, sklearn, as well as matplotlib. These libraries were utilized mainly in processing the dataset into a dataframe in order to fit a model with the appropriate data. Matplotlib specifically was used, however, to plot the accuracy measure of the training and test models. Keras, as well, being one of the most utilized APIs included within the Tensorflow API, was crucial to the progression of our research.

The dataset was first normalized before it would be ready for processing by the constructed model. This was added to the neural network preprocessing due to the improved results our team found upon normalizing the data. This was not done to the data for the regression models, due to regression models handling larger numbers better, and the lack of benefit observed when we used the normalized data. We utilized min-max scaling in order to normalize our dataset, as its wide range in data values required a reliable method to produce values within the range of [-1, 1]. The equation of the normalization is shown below:

$$z = \frac{x - \min(x)}{\max(x) - \min(x)}$$

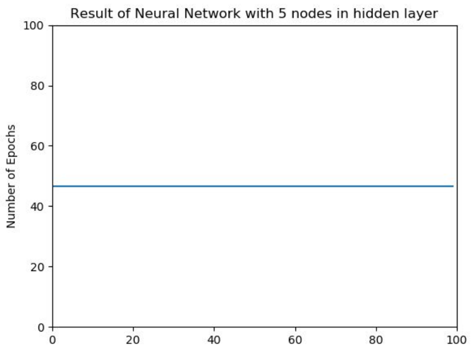
where variable $x = (x_1, \ldots, x_n)$, max and min representing the maximum and minimum respectively. The dataset was finally then split into features and a label isHomeWin, which is the aforementioned outcome variable we aim to build a model to predict.

The Keras sequential model allows one to build multiple levels of hidden layers within a given neural network, and as such a preliminary model was first developed which featured a total of 4 layers: an input layer of 23 nodes, coinciding with the amount of input variables, 2 hidden layers of 64 nodes, and a binary output layer with a single node. The activation of these layers each utilized the rectified linear unit (reLU) activation function, which we determined a good fit for the network as it all values had been normalized to a range of [0, 1], minimizing the compute cost when compared to other activation functions as the equation is linear for all positive values.

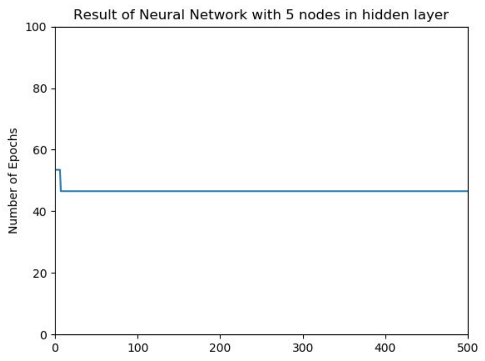
To optimize the model, the Adam optimizer was utilized as we found it the most commonplace in prior research conducted. A final action done before the model began training was that we split the data into a training set and a test set. The training set comprised of 80% of the dataset set while the test set consisted of 20%. However, some difficulty surfaced when attempting to train the model, as some technical problems with the model produced an enormous number of outliers in accuracy in the model.

An additional hidden layer was subsequently appended to the neural network during debugging and the value loss decreased incrementally as intended. An initial evaluation of the model utilizing the test data averaged an accuracy of 0.5482, or 54.82%, which was lower than initially anticipated. This was done utilizing Kera's native evaluate method, which feeds the model the test data in numerous batches. Subsequently we determined that the model was perhaps overfitting and decided to downsize the structure of the neural network. We eventually decided to reduce the number of nodes in the hidden layers to 10 and removing the additional layer. After averaging 10 model evaluations we achieved an accuracy of 61.38%, but we are unsure of the reliability of the model as we had trouble implementing k-fold cross validation.

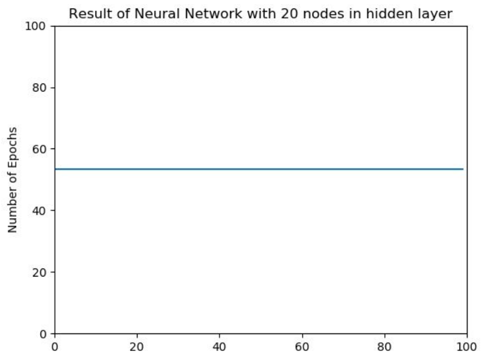
The custom-made neural network utilizes a similar method in processing the data, but was faced with a much more difficult development. Though much of the data processing was similar to that of the Tensorflow approach, the construction of the model differed fundamentally. The network makes use of the sigmoid activation function, as it is a reliable alternative to the rectified linear unit activation function. Problems arose due to the complexity of the development of a custom-made neural network, and a accuracy measure was not reliably recorded for the network. However, below displays several experimentations of the model:



Due to problems with the back propagation and error calculation implementations, much of the plots generated simply present a horizontal linear line. The plot above measures the accuracy of a network with a 5 node hidden layer through 100 epochs. The plot below displays the result with the same network design through 500 epochs:



While this result has a slight deviation at the beginning, the data moves down in accuracy from 53.47% to 46.53%, and remains at that rAnother test yielded similar results, albeit a network design of a 20 node hidden layer through 100 epochs:



As seen from the plots above, the accuracy of the plots yield a percentage of either 53.47% or 46.53% exclusively. Though there are various issues currently at hand with the model, it appears that the model mirrors the actual result of the games that was taken from the data. Analyzing our data, we found that the home team had a 53.47% chance, meaning that the neural network either always chose the home team, or always chose the away team.

The likely reason for this error comes from the back propagation and error calculation. Upon further inspection of the code, it was discovered that the error was being calculated incorrectly. Upon completing the forward propagation, when the system would calculate the error it would either be small enough to have little or no impact on the results, or large enough to over correct the data. This resulted in the data quickly becoming overly corrected, and eventually causing the error to become a constant 0. Despite the use of several different loss functions, error calculations, and back propagation methods, this problem persisted. Had this problem been resolved given more time, it is likely the custom-made neural network would perform close to or above the Tensorflow implementation.

5 Conclusions and Future Work

5.1 Conclusions on the Linear Regression Approach

We conclude that the implementation of a linear regression model is not appropriate for the prediction of NHL games. This method could possibly be used in the prediction of a football game or any other high scoring sport, but in the low scoring game of hockey, it doesn't allow much room for the predictors to have value. For future implementation of this project, we would like to focus more time into the implementation of a logistic regression model. We believe from what we have seen from the small amount of time we spent with the model that it would fix the issues that we encountered in our attempt to manipulate the linear model to make it fit the data. The data set in this project was mainly binary, which should have been a red flag for using a linear regression model upon our first implementation.

5.2 Conclusions on the Neural Network Approach

Though much debate could be had upon the debate of the ethics of gambling, it is difficult not to admit to its popularity among the general population. The implementation of a neural network to predict an outcome of a sports event is one of the most appealing utilizations within machine learning, as recent innovations such as the use of IBM Watson in predicting fantasy football probabilities. Aside from hockey, various other sports such as baseball, football, soccer, etc may benefit from the implementation of such a technology. From predicting batting percentages in baseball to likelihoods of goals from outside the box in soccer to the probability of a pass succeeding in football, many utilizations of neural networks are present. Though our current implementation of a neural network does need optimization and fine-tuning, the accuracy between the Tensorflow and custom-made implementations do fall into similar ranges.

References

Bunker, Rory P., and Fadi Thabtah. "A Machine Learning Framework for Sport Result Prediction." Applied Computing and Informatics, vol. 15, no. 1, 2019, pp. 27–33., doi:10.1016/j.aci.2017.09.005.

Montgomery, Douglas C. *Introduction to Linear Regression Analysis, Fifth Edition Set*, John Wiley & Sons, 2013. 978-1118780572

Pischedda, Gianni. "Predicting NHL Match Outcomes with ML Models." *International Journal of Computer Applications*, vol. 101, no. 9, 2014, pp. 15–22., doi:10.5120/17714-8249.

"Welcome to STAT 508! | STAT 508." *Comparing Two Quantitative Variables | STAT 800*, Feb. 2018, newonlinecourses.science.psu.edu/stat508/.