CARLETON UNIVERSITY

SCHOOL OF
MATHEMATICS AND STATISTICS

HONOURS PROJECT

TITLE: Analyzing Auston Matthews' Shot Data Using Classification

AUTHOR: Theresa Kennedy

SUPERVISOR: Dr. Shirley Mills

DATE: April 17, 2021

Table of Contents

Abstract

This project will explore the shooting data of Auston Matthews, a center on the Toronto Maple Leafs. It will attempt to build a classifier that is able to effectively model the data of his shots on net. The classifier models will use predictive variables combined with a binary response variable to provide valuable insights into his shooting patterns.

1. Introduction

1.1 Background

The use of data analytics in the National Hockey League has traditionally been a controversial subject. At the beginning of its usage, the management and head offices of the league were severe critics of it and were reluctant to modify the conventional techniques of analyzing the performance of players, teams, and scouting techniques. Many refused to incorporate proven data analytic techniques, claiming them to be pointless. [5] However, as data analytics started to emerge in the NHL, some teams were quick to utilize these strategies. In 2015, the Carolina Hurricanes hired Eric Tulsky, a prominent hockey data analyst. [6] Using data analytics in conjunction with older scouting techniques they were able to acquire strong defensive players which in turn helped them become a contender for the Stanley Cup in the 2018-2019 playoffs, reaching the conference finals. Now all teams in the NHL have a data analytics department. So why are data analytics important for improving the performance of hockey teams?

Data analysis plays a significant role in multiple areas in the sport of hockey. It allows coaches to analyze their team's overall performance. It can be used to see what went wrong in a game and how teams can improve from their mistakes. The data recorded also highlights the strengths and weaknesses of opponents which can then give teams knowledge on what to exploit to their advantage during a game. On a more granular basis, analytics can also be used to analyze the training and performance of individual athletes and to examine the effect of various combinations of athletes playing together.

This project focuses on how analytics can be used to examine the shooting performance of a single athlete. Auston Matthews is a center on the Toronto Maple Leafs. This project explores his overall shot pattern and the shooting variables that affect his goalscoring rate during the 2019-2020 season. Three different classification techniques are used to develop classifiers for his goal-scoring. The binary response that is modelled is whether Matthews scored a goal or not and the predictive variables used are shot location, side of ice, and shot style. A logistic regression model, a neural network, and a classification tree are built, and their performance is analyzed to see how accurately they predict his results. As well, the predictive variables are assessed to determine if they significantly impact whether he scores a goal or not.

1.2 Motivation

The main advanced statistics used for analyzing shot attempts in the NHL are Corsi, Fenwick, and xG models. Corsi and Fenwick are very similar except that Fenwick does not include blocked shots in its calculations. A Corsi percentage is the shot attempts for a team relative to the total shot attempts for and against that team. Corsi and Fenwick can both be applied to individual players as well as to the overall team. Expected goals models take into consideration factors that affect a shot such as shot type, location, and whether it occurred on the power play.

Not much research has been done in the analysis of shot patterns of individual players through using scoring a goal as a binary response variable. The Toronto Maple Leafs were the team I watched growing up and Auston Matthews is one of their strongest goal-scorers. This motivated me to focus my project on his shot pattern and how it relates to his goalscoring rate.

2. Literature Review of Related Work

As mentioned before, xG models have been used as one of the main advanced statistics analyzing shot data in the NHL. Since data analytics emerged in the NHL, multiple xG models have been developed. Older xG models such as Brian Macdonald's model [13] used Corsi and Fenwick statistics combined with other variables such as hits and faceoffs to predict future goals. These models did not consider other variables such as shot location, the type of shot that was taken, or what wing it was taken on. DTMAboutHeart and asmean developed a model [7] that takes into account multiple shooting variables including shot distance, shot angle, and shot type. In the past, the xG models had not considered shot quality but this model proved that these qualities have an impact on scoring a goal. In 2019, an xG model was developed by Alex Novet [16] that used pre-shot movement to predict goals. This model includes the passes that happen before a shot is taken as well as other pre-shot movements. The model showed that the variables of shot screens and odd man rushes had an impact on the shots as well as passes that travelled through the royal road on the ice.

3. Data Source and Manipulation

3.1 Data Source

The dataset used for this project was obtained from NHL Stats API Documentation on GitHub. [9] The dataset extracted consisted of data for the 70 regular season games Auston Matthews played during the 2019-2020 season. The data set included live data on all plays during each game in which he played. This project uses the coordinates of each goal and shot by Auston Matthews in the 70 regular season games.

3.2 Data Manipulation

From the original dataset, the (x, y) coordinates of his shots were extracted; 48 of the 297 shots extracted were goals and 249 were shots. The x and y coordinates were extracted into separate lists; the first held the x coordinates and the second held the y coordinates. Since Auston Matthews had only 290 shots during the season, some coordinates needed to be removed from the data extracted. One element of the goals was recorded in error as a goal and was removed. As well, all deflected shots were removed, since deflected shots were not to be counted as a shot on goal. The coordinates of shots and goals were then combined resulting in 47 goals with coordinates and 243 non-goals with coordinates.

The (x, y) coordinates of shots were plotted on a hockey rink that was obtained from GitHub. [3] Different colours were then used to indicate whether a shot resulted in a goal and whether it was a power play goal.

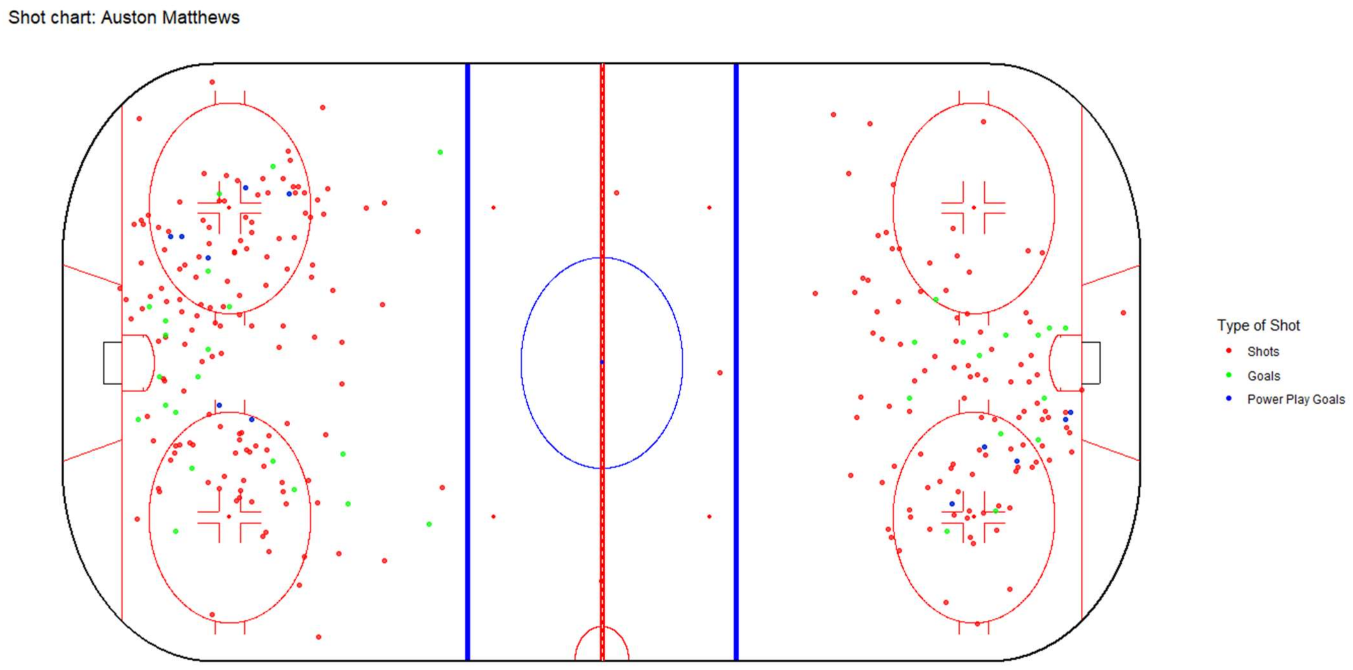


Figure 1: Shot Chart for Auston Matthews

Figure 1 shows that Auston Matthews' shots not resulting in a goal are in red, his non power play goals are in green, and his power play goals are in blue. Near the right net, his goals are mostly in the middle of the faceoff circles. Near the left net, there is a line of goals coming up the left side into the bottom face-off circle. As well, with the power-play goals, Fig. 1 shows that, for the power play, he tends to score when he is playing on his right side (which is his weak side). Fig. 1 is evidence that where he shoots from on the ice affects his scoring.

To analyze his shot pattern further, we model the response variable goal/no goal using his shooting location as well as side of ice and type of shot as predictive variables. The type of shot data is extracted from the original dataset for all shots. A list for the side of ice that each shot was taken on was created. As well a list for the response variable, goal or no goal was created. Then, using the (x, y) coordinates of his shots, the shot angle and distance from the net of each were calculated. A data frame was then produced with five (5) variables for each of the shots. The 5 variables in the data frame are:

Type of Shot: the type of shot that was taken; wrist shot, backhand, tip-in, snap shot, slap shot

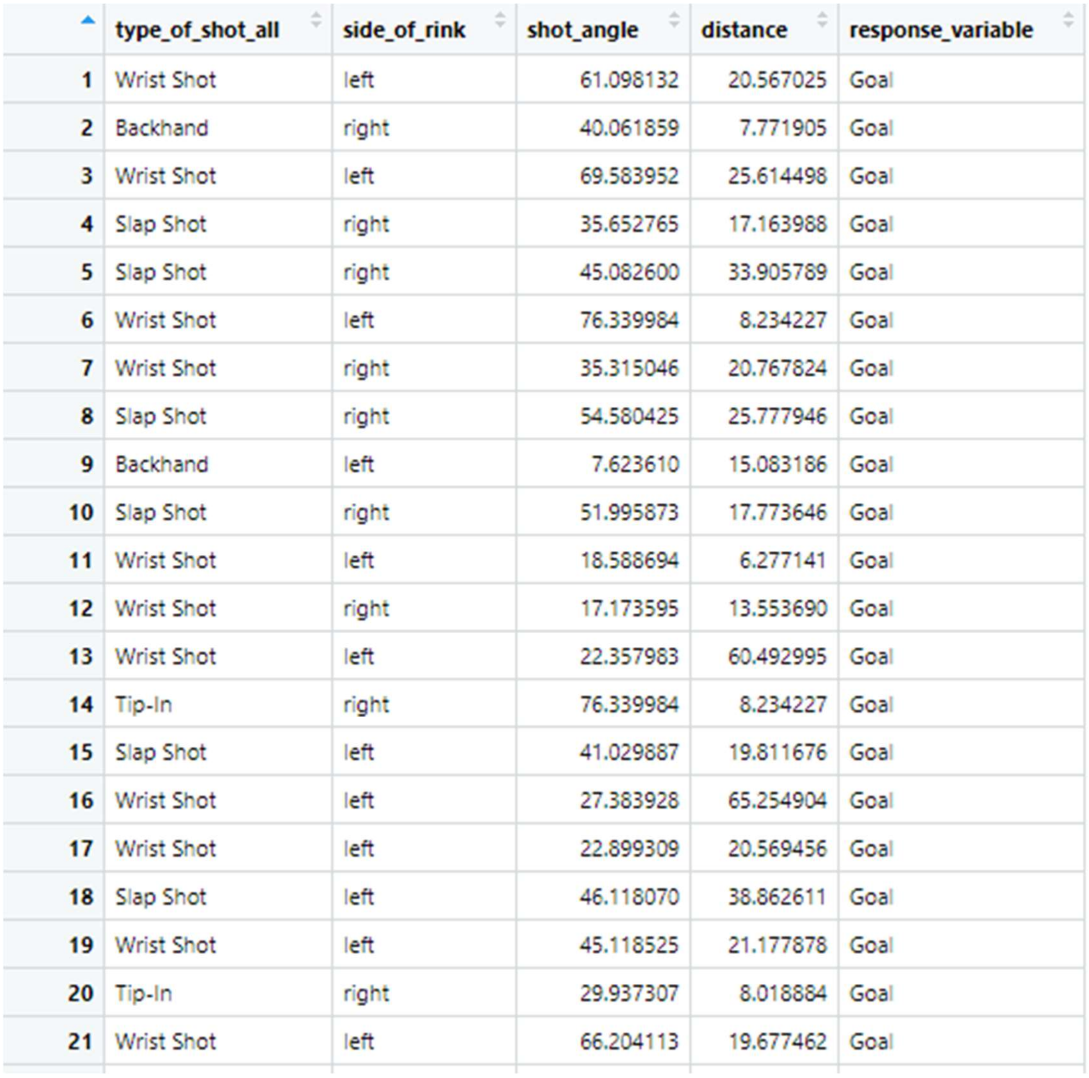
Side of Rink: the side of the rink the shot was on; right, middle, left

Shot Angle: the angle of the shot

Shot Distance: the distance of the shot from the net

Response Variable: (binary variable) goal or no goal

A snapshot of this data frame can be seen in Figure 2.



	type_of_shot_all	side_of_rink	shot_angle	distance	response_variable
1	Wrist Shot	left	61.098132	20.567025	Goal
2	Backhand	right	40.061859	7.771905	Goal
3	Wrist Shot	left	69.583952	25.614498	Goal
4	Slap Shot	right	35.652765	17.163988	Goal
5	Slap Shot	right	45.082600	33.905789	Goal
6	Wrist Shot	left	76.339984	8.234227	Goal
7	Wrist Shot	right	35.315046	20.767824	Goal
8	Slap Shot	right	54.580425	25.777946	Goal
9	Backhand	left	7.623610	15.083186	Goal
10	Slap Shot	right	51.995873	17.773646	Goal
11	Wrist Shot	left	18.588694	6.277141	Goal
12	Wrist Shot	right	17.173595	13.553690	Goal
13	Wrist Shot	left	22.357983	60.492995	Goal
14	Tip-In	right	76.339984	8.234227	Goal
15	Slap Shot	left	41.029887	19.811676	Goal
16	Wrist Shot	left	27.383928	65.254904	Goal
17	Wrist Shot	left	22.899309	20.569456	Goal
18	Slap Shot	left	46.118070	38.862611	Goal
19	Wrist Shot	left	45.118525	21.177878	Goal
20	Tip-In	right	29.937307	8.018884	Goal
21	Wrist Shot	left	66.204113	19.677462	Goal

Figure 2: Snapshot of Variables of Data Frame

Using this data frame, three classification techniques are applied to create a classifier to analyze how the predictive variables affect goal scoring.

4. Methodology

Classification is a predictive modelling technique that groups data into classes. It is a branch of supervised learning since it builds a classifier from data and can compare its predicted results to actual observed results. Binary classification is a form of classification in which the result predicted has one of two possible outcomes. There are multiple methods of binary classification, but this project focuses on logistic regression, neural networks, and classification trees.

4.1 Logistic Regression

Logistic regression is a classification method in which explanatory variables are used to predict a binary categorical variable. Logistic regression differs from well-known linear regression because it has a binary response variable. If there is one explanatory variable it is considered simple logistic regression whereas if there is more than one it is called multiple logistic regression. The focus of this project will be on multiple logistic regression, but an explanation of simple logistic regression is needed before we introduce it.

The dependent categorical variable Y follows a binomial distribution where,

$$Y = \begin{cases} 0, & not\ a\ goal \\ 1, & goal \end{cases}$$

and the probability of $P(Y = 1) = p$ and $P(Y = 0) = 1 - p$. In simple logistic regression there is only one explanatory variable which we will assign as x. The probability of "success", or in our case that it is a goal i.e., $p = P(Y = 1)$ is assumed to follow a sigmoidal model:

$$p = \frac{e^{\beta 0 + \beta 1 x}}{1 + e^{\beta 0 + \beta 1 x}}$$

The logit transformation gives

$$logit(p) = ln\left[\frac{p}{1-p}\right] = \beta_0 + \beta_1 x$$

Whereas Y is binary, p ranges between 0 and 1 and using the logit transformation of p, we have a response variable that is linear in the betas, continuous, and can range from -∞ to ∞. Now expanding this to multiple logistic regression is intuitive. Here there are multiple explanatory variables which we will assign to $x = (x_1, x_2, \dots, x_p)$. Now, our logistic regression model becomes

$$p = \frac{e^{\beta_0+\beta_1x_1+\beta_2x_2+\dots+\beta_px_p}}{1 + e^{\beta_0+\beta_1x_1+\beta_2x_2+\dots+\beta_px_p}}$$

and the logit transformation is

$$logit(p) = ln\left[\frac{p}{1-p}\right] == \beta_0 + \beta_1x_1 + \beta_2x_2 + \cdots + \beta_px_p$$

4.1.1 Estimation of Parameters

To be able to estimate the chance of success, p, the estimates of $\beta's$ must be calculated. The method used to estimate the unknown parameters is called the maximum likelihood method (ML).

The goal of the maximum likelihood method is finding the estimators of the $\beta's$ that maximize the likelihood function, $L(\beta)$. Let $\beta = (\beta_0, \beta_1, \dots, \beta_p)$, x_j be explanatory variable, j where $j = 1, \dots, p$, y_i be the binary response variable that has values of either 0 and 1 and let p_i be the probability of $y_i = 1$. To construct the likelihood function we consider the contribution for $(x_1, \dots, x_p, y_i)$. For $y_i = 1$, the contribution is p_i and if y_i = 0 then the contribution is $1 - p_i$. Since the n observations are assumed to be independent, we write the likelihood function as a product of the probabilities,

$$L(\boldsymbol{\beta}) = \prod_{i=1}^{n} p_i(x_i)^{y_i}(1 - p_i(x_i))^{1-y_i}$$

We then take the log of this function since maximizing $log\ (L(\beta))$ is equivalent to maximizing $L(\beta)$ because log is a monotonic function. Thus,

$$log\big(L(\boldsymbol{\beta})\big) = \sum_{i}^{n}\{y_i\, log\, p(x_i) + (1 - y_i) log\ (1 - p(x_i))\}$$

$$= \sum_{i=1}^{n}(1 - y_i) log\ [1 - p(x_i)] + y_i log p(x_i)$$

$$= \sum_{i=1}^{n} y_i log \frac{p(x_i)}{1-p(x_i)} + \log[1 - p(x_i)]$$

$$= \sum_{i=1}^{n} y_i x_i \beta - \log\big(1 + e^{x_i\beta}\big)$$

We then take the partial derivatives of the log likelihood function w.r.t. $\boldsymbol{\beta}$ and set equal to zero to solve for

$$\frac{\partial log L(\beta)}{\partial \beta_0} = \sum_{i=1}^{n} y_i - p(x_i), \qquad \frac{\partial log L(\beta)}{\partial \beta_1} = \sum_{i=1}^{n} x_i(y_i - p(x_i))$$

4.1.2. Evaluation of Model

After the logistic regression model is created, the next step is to evaluate the model. The likelihood ratio test and the pseudo R^2 test are two ways that are used to assess the logistic model.

The likelihood ratio test compares the log likelihood of the data between two different logistic regression models. In logistic regression, a chi-square likelihood ratio test is used to calculate the ratio. The second model created has one or more variables removed from it. We will let L_0 represent the log likelihood for the second model and let L_1 represent the log likelihood for the model with all the predictive variables. The formula of the likelihood ratio statistic as per [13] is

$$Likelihood\ Ratio = 2(\ log(L_0(\boldsymbol{\beta}) - L_1(\boldsymbol{\beta})).$$

The likelihood ratio test statistic is chi-squared with degrees of freedom equal to the number of predictive variables. To determine which model is better we will set up a hypothesis test.

$$H_0 = L_0\ fits\ the\ model\ better$$

$$H_1 = L_0\ does\ not\ fit\ the\ model\ better$$

After the likelihood ratio is calculated, the p-value can be used to determine whether to reject the hypothesis. If the p-value is less than 0.05 this indicates that the hypothesis should be rejected and the full model fits the model better.

The second test to evaluate the model is the pseudo R^2 test. Pseudo R^2 values are used in logistic regression models to replace the R^2 value from linear regression. The pseudo R^2 that will be used in this project is the McFadden's R^2. The formula as per [2] for this statistic is,

$$McFadden's\ R^2 = 1 - [\log (L_1(\boldsymbol{\beta})/\log (L_0(\boldsymbol{\beta})].$$

McFadden's R^2 ranges from 0 to 1; the closer it is to 1 the better the model.

4.2 Neural Networks

Artificial neural networks simulate the neural networks in our brains. Let us visualize a simple neural network as per [11] in Figure 3.

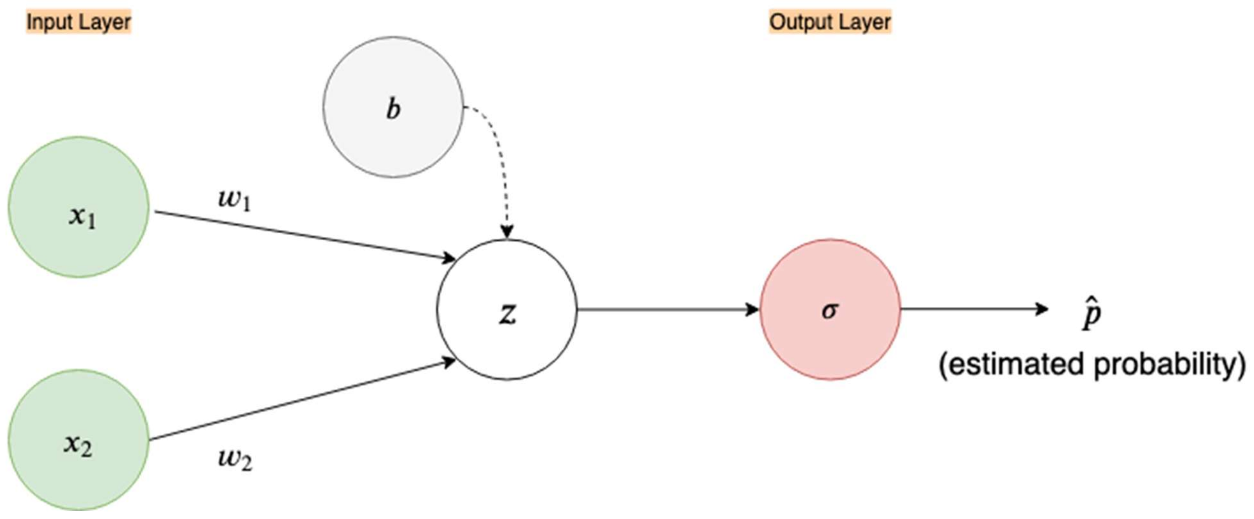


Figure 3: Simple neural network

In the above figure the input layer has two input nodes, x_1 and x_2, which represent two features of the data. $w_1$and w_2 represent weights that affect the calculation that enters the next node. The z node represents a linear function that is created from the inputs and the weights. For this neural network, the linear function would be,

$$z = x_1 w_1 + x_2 w_2.$$

The b node is the bias which is put into the z equation and influences the output. The sigmoid node, σ, takes this linear function z and puts it into the sigmoid function. The sigmoid function is an activation function that essentially transforms z into a value in the range $0 < \sigma < 1$. The second part of the output layer corresponds to an estimated probability, $\hat{p}$, that the inputted case corresponds to Y-1. Assuming we choose a threshold of $\hat{p} = 0.5$ then we would predict,

$$Y = \begin{cases} 1, \hat{p} > 0.5 \\ 0, \hat{p} < 0.5 \end{cases}$$

as per [11].

The next part of a binary classification neural network is a binary cross-entropy loss function. This function will help fix the estimated weights in the neural network by showing the difference between the predicted output and the output that is wanted. The formula for this function as per [11] is,

$$L(y,\hat{p}) = -ylog(\hat{p}) - (1-y)\log(1-\hat{p})$$
$$= \begin{cases} -\log(\hat{p}) & ,if\ y=1 \\ -\log(1-\hat{p}) & ,if\ y=0 \end{cases}$$

where y is the class and $\hat{p}$ is the estimated probability.

The next step is to look at the derivative, also called the gradient, of the Binary Cross-Entropy Loss function. The derivative of the function shown here as per [11],

$$\frac{\partial L(y,\hat{p})}{\partial \hat{p}} = \frac{\partial}{\partial \hat{p}}(-ylog(\hat{p}) - (1-y)\log(1-\hat{p}))$$
$$= \frac{-y}{\hat{p}} + \frac{1-y}{1-\hat{p}}$$

Substituting in $y = 0$ and $y = 1$ we get,

$$\frac{\partial L(y,\hat{p})}{\partial \hat{p}} = \begin{cases} \frac{-1}{\hat{p}} & if\ y=1 \\ \frac{1}{1-\hat{p}} & if\ y=0 \end{cases}$$

as per [11].

So how does the derivative help us determine how to make our model better? If the derivative is positive this means that our weights should be decreased, while if the derivative is negative the weights should be increased.

The derivative can now be used to perform backpropagation, which shows how much each portion of the network contributed to the loss and how we can change our network to make our model better. The basic idea of backpropagation is to work backwards and multiply upstream gradients with local gradients that are then passed on to the next node in the chain. We will use the simple neural network in Figure 3 to illustrate this process. The first upstream gradient is calculated by dividing the derivative of the loss by itself which equals 1. The local gradient is the derivative of the Loss function as shown above. Then the upstream gradient and local gradient are multiplied to equal,

$$\frac{\partial L}{\partial \hat{p}} = 1 * \frac{-y}{\hat{p}} + \frac{1-y}{1-\hat{p}}$$

$$= \frac{-y}{\hat{p}} + \frac{1-y}{1-\hat{p}}$$

as per [11]. The next local gradient is the derivative of the sigmoid function as per [11],

$$\hat{p} = \sigma(z) = \frac{1}{1+e^{-z}}$$

$$\frac{\partial p}{\partial z} = \hat{p}(1-\hat{p}).$$

Multiplying the derivative of the sigmoid function by the upstream gradient we get,

$$\frac{\partial L}{\partial \hat{p}} = \frac{\partial L}{\partial \hat{p}} * \frac{\partial \hat{p}}{\partial \hat{z}}$$

$$= \left(\frac{-y}{\hat{p}} + \frac{1-y}{1-\hat{p}}\right) * \hat{p}(1-\hat{p})$$

as per [11]. Since the next node would be the input nodes we stop here. We only want to change the weights that correspond with the input nodes, not the nodes themselves.

The next step is to change the weights and the bias using gradient descent. Gradient descent updates the parameters, weight, and bias, by transitioning away from the gradient. The equation for gradient descent as per [11] can be seen here,

$$w = w - \propto \frac{\partial L}{\partial w}$$

where $\propto$ is a learning rate and $w = (w_1, w_2, \cdots, w_i)$.. The learning rate regulates the transition away from the gradient and is set manually. Using backward propagation and gradient descent, the weights and bias can be changed to create a better classifier model.

4.3 Classification Trees

Classification trees represent a series of questions asked about values of the predictive variables until a decision can be made about the class of the binary response variable. These questions and the answers that follow are displayed in a tree-like structure that is made up of nodes and branches. There are three different nodes: root, internal, and leaf. The root node is at the top of the tree and then splits into internal nodes. The internal nodes then split into internal or leaf nodes. Leaf nodes are at the bottom of tree and have a class label. They represent the last step in the tree method.

There are three different measures used for choosing the binary splits at the nodes. Classification error rate at node t is the fraction of observations in the training set at a certain node t that are not part of the highest occurring class i.The error or impurity can be found by this equation as per [13],

$$Classification\ Error(t) = 1 - max[p(i|t)]$$

where $p(i|t)$ is the fraction of the training data at node t in class i. A second method to calculate error in the tree is called the Gini Index. This measures the amount of a single class i in a specific node t to see how homogeneous it is. The Gini Index is represented by this equation as per [13],

$$G(t) = 1 - \sum_i [p(i|t)]^2.$$

The smaller the Gini Index, the purer the node is. Another method that can be used to determine the best binary split is called Cross-Entropy. The formula for Cross-Entropy as per [13] is

$$Entropy(t) = -\sum_i p(i|t) \log_2 p(i|t).$$

Using these three methods, the best binary split can be determined at each node. In this project we create a classification tree using 'rpart' in R to build the tree since this routine can handle missing variables.

5. Results

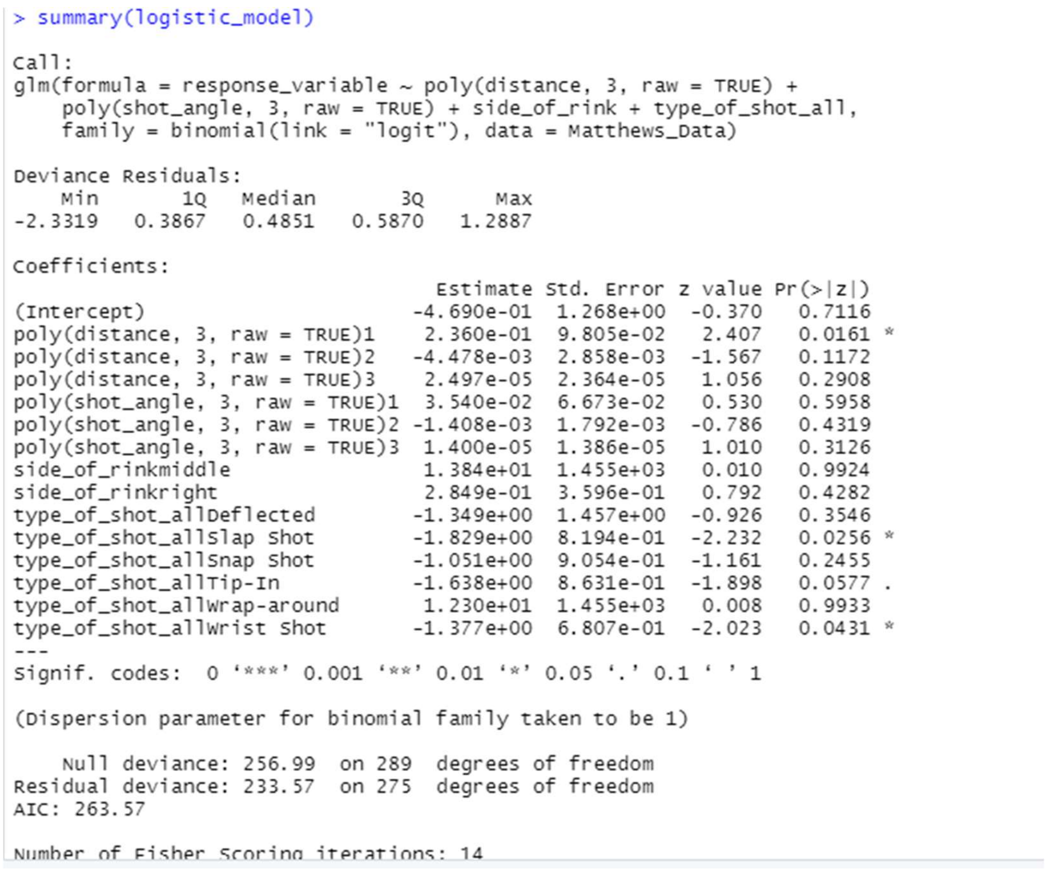
5.1 Logistic Regression

Using the methods described in Section 4 and the data frame described in Section 3, the logistic regression model to represent Auston Matthews' shooting data can be built. The model is built in R using the *glm* function with family set to binomial (link=logit), so the program knows it is a logistic regression model. The model created in R using,

```
logistic_model <- glm(as.factor(response_variable) ~ poly(distance,3, raw=TRUE) +
poly(shot_angle,3,raw=TRUE) + side_of_rink + type_of_shot_all,data=Matthews_Data, family =
binomial(link='logit'))
```

The binary response variable is whether Auston Matthews scored a goal or not, with values 1 or 0, respectively. The dependent variables are shot distance, shot angle, side of rink, and type of shot. The shot distance and shot angle variables are entered into the model as polynomials of degree three each.

The next step after running the logistic model is to determine which if any predictor variables are significant. This can be determined by running the summary of the model and looking at the coefficients of the model. The summary can be seen in Figure 4 below. The predictor variables that are significant are the ones with the symbol '*' and '.'beside it. From this summary, the first factor level of shot distance, and Tip-In, Wrist Shot, and Slap Shot are significant predictor variables. This indicates that these four variables affect the chance of him scoring.



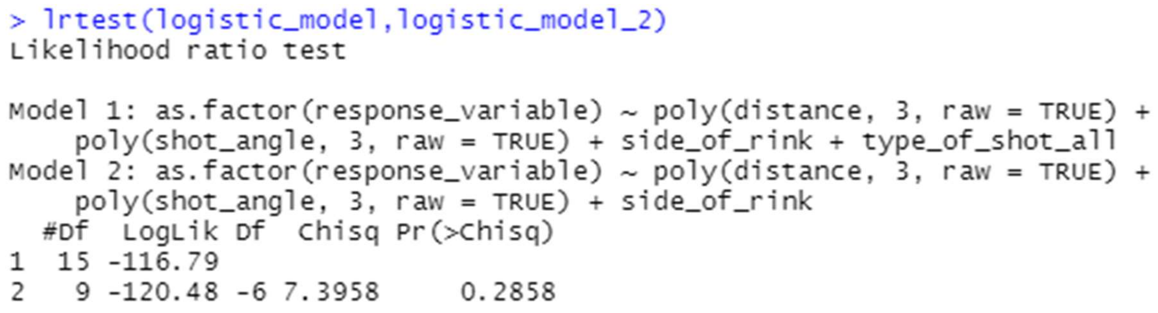
```
> summary(logistic_model)

Call:
glm(formula = response_variable ~ poly(distance, 3, raw = TRUE) +
    poly(shot_angle, 3, raw = TRUE) + side_of_rink + type_of_shot_all,
    family = binomial(link = "logit"), data = Matthews_Data)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.3319   0.3867   0.4851   0.5870   1.2887

Coefficients:
                                  Estimate Std. Error z value Pr(>|z|)
(Intercept)                     -4.690e-01  1.268e+00  -0.370   0.7116
poly(distance, 3, raw = TRUE)1   2.360e-01  9.805e-02   2.407   0.0161 *
poly(distance, 3, raw = TRUE)2  -4.478e-03  2.858e-03  -1.567   0.1172
poly(distance, 3, raw = TRUE)3   2.497e-05  2.364e-05   1.056   0.2908
poly(shot_angle, 3, raw = TRUE)1  3.540e-02  6.673e-02   0.530   0.5958
poly(shot_angle, 3, raw = TRUE)2 -1.408e-03  1.792e-03  -0.786   0.4319
poly(shot_angle, 3, raw = TRUE)3  1.400e-05  1.386e-05   1.010   0.3126
side_of_rinkmiddle               1.384e+01  1.455e+03   0.010   0.9924
side_of_rinkright                2.849e-01  3.596e-01   0.792   0.4282
type_of_shot_allDeflected       -1.349e+00  1.457e+00  -0.926   0.3546
type_of_shot_allSlap Shot       -1.829e+00  8.194e-01  -2.232   0.0256 *
type_of_shot_allSnap Shot       -1.051e+00  9.054e-01  -1.161   0.2455
type_of_shot_allTip-In          -1.638e+00  8.631e-01  -1.898   0.0577 .
type_of_shot_allWrap-around      1.230e+01  1.455e+03   0.008   0.9933
type_of_shot_allWrist Shot      -1.377e+00  6.807e-01  -2.023   0.0431 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 256.99  on 289  degrees of freedom
Residual deviance: 233.57  on 275  degrees of freedom
AIC: 263.57

Number of Fisher Scoring iterations: 14
```

Figure 4: Summary of Logistic Model

Even though the model shows that some of the variables are significant, the model first needs to be examined to determine if it actually fits the data.

Evaluating the Model

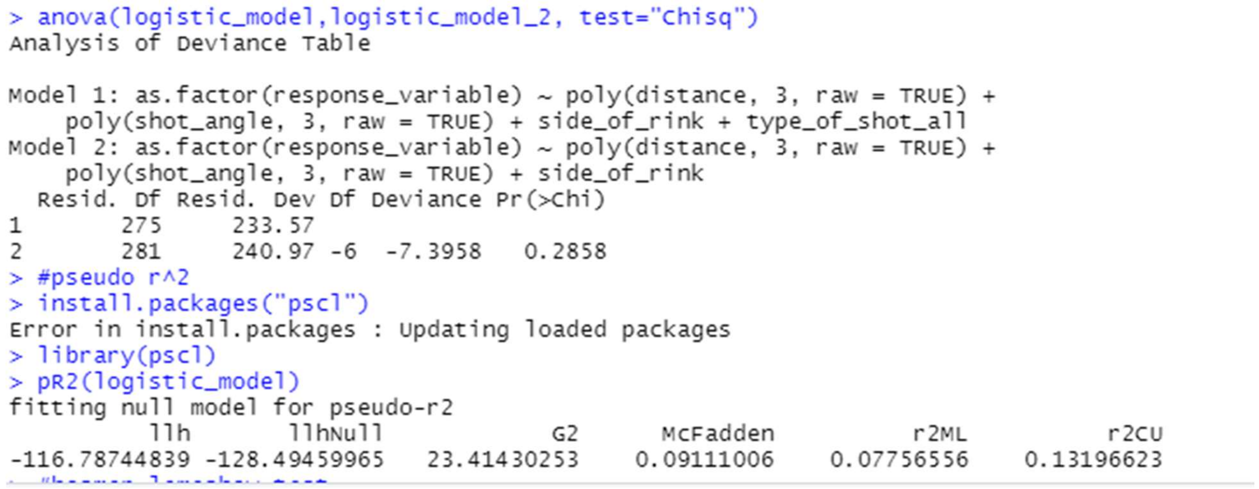
To evaluate the models, the two methods described in Section 4 will be used. The first method is the likelihood ratio test. The results of the likelihood ratio test can be seen below in Figure 5.



```
> lrtest(logistic_model,logistic_model_2)
Likelihood ratio test

Model 1: as.factor(response_variable) ~ poly(distance, 3, raw = TRUE) +
    poly(shot_angle, 3, raw = TRUE) + side_of_rink + type_of_shot_all
Model 2: as.factor(response_variable) ~ poly(distance, 3, raw = TRUE) +
    poly(shot_angle, 3, raw = TRUE) + side_of_rink
  #Df  LogLik Df  Chisq Pr(>Chisq)
1  15 -116.79
2   9 -120.48 -6 7.3958     0.2858
```

Figure 5: Likelihood Ratio Test

The p-value obtained from this test is 0.2858. This test shows that the logistic model that was built is not a good fit to the data. To verify the results of the likelihood ratio test, we perform one more test, the pseudo R^2 test. The results of the pseudo R^2 test are shown below in Figure 6.



```
> anova(logistic_model,logistic_model_2, test="Chisq")
Analysis of Deviance Table

Model 1: as.factor(response_variable) ~ poly(distance, 3, raw = TRUE) +
    poly(shot_angle, 3, raw = TRUE) + side_of_rink + type_of_shot_all
Model 2: as.factor(response_variable) ~ poly(distance, 3, raw = TRUE) +
    poly(shot_angle, 3, raw = TRUE) + side_of_rink
  Resid. Df Resid. Dev Df Deviance Pr(>Chi)
1       275     233.57
2       281     240.97 -6  -7.3958   0.2858
> #pseudo r^2
> install.packages("pscl")
Error in install.packages : Updating loaded packages
> library(pscl)
> pR2(logistic_model)
fitting null model for pseudo-r2
          llh       llhNull          G2     McFadden         r2ML         r2CU
-116.78744839 -128.49459965   23.41430253   0.09111006   0.07756556   0.13196623
```

Figure 6: Pseudo R^2 Test

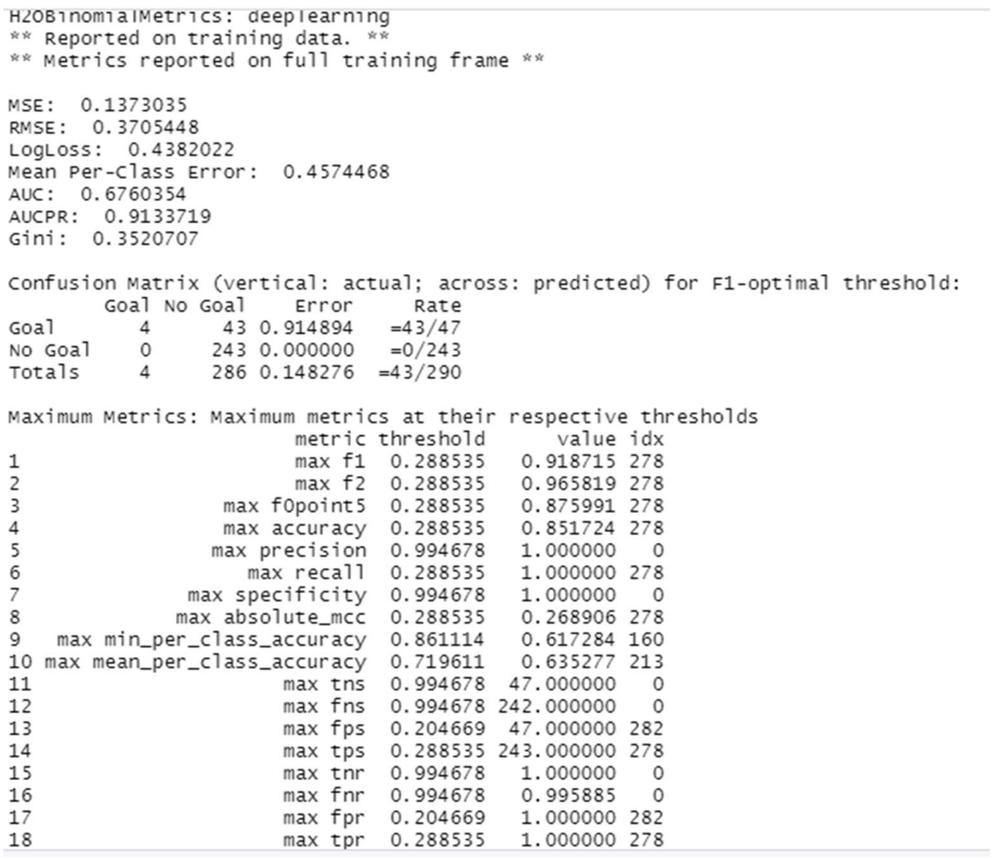
In this test, instead of looking at the p-value, we obtain what is called McFadden's R^2. The McFadden's R^2 is 0.09111. We know that McFadden's R^2 ranges from 0 to 1 and numbers closer to 0 show that the model has little predictive ability. Therefore, we can conclude from the results of the likelihood ratio test and the pseudo R^2 test that the logistic model is not a good fit for the data. Since logistic regression did not model the data very well, another classifier is constructed in an attempt to find a good classifier for this data.

5.2 Neural Networks

A neural network is the second classification method applied to the dataset. The neural network is built using the package 'h2o' in R. The dataset first must be converted to an *h2o* frame before the model can be built. The model is then built by using the function *h2o.deeplearning()* function and is shown here:

h2o_NN <- h2o.deeplearning(x=1:4, y=5, training_frame=train_set_h2o, nfolds=5, standardize = TRUE, activation="Rectifier", hidden=c(200,200), seed=40, variable_importances = T).

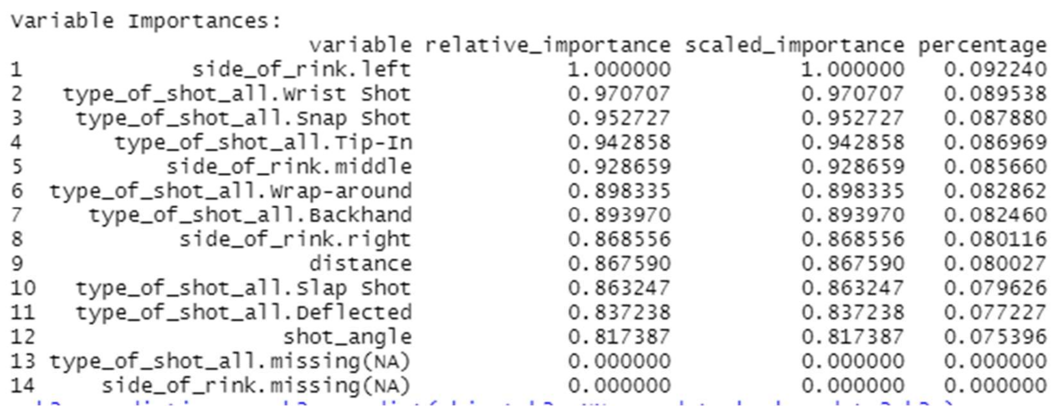
To evaluate the performance of the model, we utilize the function *h2o.performance()*. This outputs a summary of the model's metrics and tells us how well the classifier performs. The key metrics are the classification error shown in the confusion matrix, log loss, MSE, and AUC. The confusion matrix in Figure 7 tells us that 4 goals that were goals were predicted correctly and 243 shots that were not goals were predicted correctly. The classification error for 'Goal' was about 91% and the error for 'No Goal' was 0%. The classification error for 'Goal' is very high, indicating the model is not a good classifier for the goals. On the other hand, the log loss, MSE, and AUC indicated the model is good. The log loss is low at 0.438, and a lower log loss indicates a better model. The MSE, which is the average of the prediction errors squared, is 0.137 which is close to zero; it indicates a good classifier since it is close to zero. Lastly the AUC is 0.676 which indicates the model is okay as it is closer to 1.



```
H2OBinomialMetrics: deeplearning
** Reported on training data. **
** Metrics reported on full training frame **

MSE:  0.1373035
RMSE:  0.3705448
LogLoss:  0.4382022
Mean Per-Class Error:  0.4574468
AUC:  0.6760354
AUCPR:  0.9133719
Gini:  0.3520707

Confusion Matrix (vertical: actual; across: predicted) for F1-optimal threshold:
        Goal No Goal    Error     Rate
Goal       4      43 0.914894   =43/47
No Goal    0     243 0.000000   =0/243
Totals     4     286 0.148276  =43/290

Maximum Metrics: Maximum metrics at their respective thresholds
                        metric threshold      value idx
1                       max f1  0.288535   0.918715 278
2                       max f2  0.288535   0.965819 278
3                 max f0point5  0.288535   0.875991 278
4                 max accuracy  0.288535   0.851724 278
5                max precision  0.994678   1.000000   0
6                   max recall  0.288535   1.000000 278
7              max specificity  0.994678   1.000000   0
8             max absolute_mcc  0.288535   0.268906 278
9   max min_per_class_accuracy  0.861114   0.617284 160
10 max mean_per_class_accuracy  0.719611   0.635277 213
11                     max tns  0.994678  47.000000   0
12                     max fns  0.994678 242.000000   0
13                     max fps  0.204669  47.000000 282
14                     max tps  0.288535 243.000000 278
15                     max tnr  0.994678   1.000000   0
16                     max fnr  0.994678   0.995885   0
17                     max fpr  0.204669   1.000000 282
18                     max tpr  0.288535   1.000000 278
```

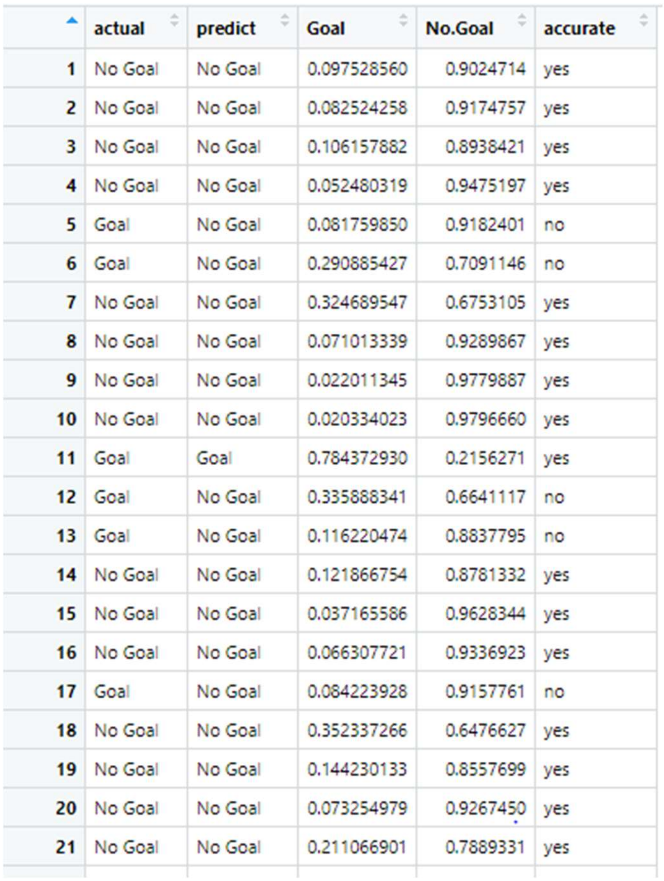
Figure 7: Summary of Model Performance

The next step is determining if any of the predictor variables have a significant effect on the response variable. Shown below in Figure 8 are the predictor variables ranked in order from most important to least important. The predictor variables with a relative importance higher than 90% are the left side of the rink, wrist shot, snap shot, tip-in, and the middle of the rink. Since there is only one observation in the middle of the rink and it did not occur with a goal this indicates that the variable importances are not accurate. It is typical for neural networks not to have accurate variable importances. When comparing with the significant variables in the logistic regression model, there are some variables that overlap. Wrist shots and tip-ins were significant in both the logistic regression model and the neural network model. This could indicate that these two variables do significantly impact the chance of Auston Matthews scoring.



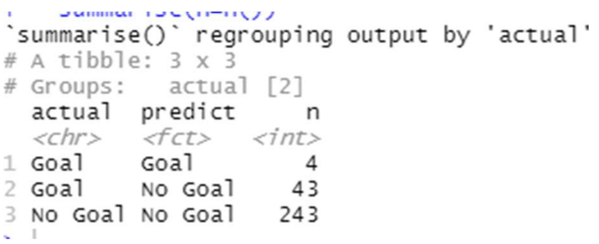
```
Variable Importances:
                         variable relative_importance scaled_importance percentage
1               side_of_rink.left            1.000000          1.000000   0.092240
2     type_of_shot_all.Wrist Shot            0.970707          0.970707   0.089538
3      type_of_shot_all.Snap Shot            0.952727          0.952727   0.087880
4         type_of_shot_all.Tip-In            0.942858          0.942858   0.086969
5             side_of_rink.middle            0.928659          0.928659   0.085660
6    type_of_shot_all.Wrap-around            0.898335          0.898335   0.082862
7       type_of_shot_all.Backhand            0.893970          0.893970   0.082460
8              side_of_rink.right            0.868556          0.868556   0.080116
9                        distance            0.867590          0.867590   0.080027
10     type_of_shot_all.Slap Shot            0.863247          0.863247   0.079626
11     type_of_shot_all.Deflected            0.837238          0.837238   0.077227
12                     shot_angle            0.817387          0.817387   0.075396
13 type_of_shot_all.missing(NA)            0.000000          0.000000   0.000000
14     side_of_rink.missing(NA)            0.000000          0.000000   0.000000
```

Figure 8: Variable Importance



	actual	predict	Goal	No.Goal	accurate
1	No Goal	No Goal	0.097528560	0.9024714	yes
2	No Goal	No Goal	0.082524258	0.9174757	yes
3	No Goal	No Goal	0.106157882	0.8938421	yes
4	No Goal	No Goal	0.052480319	0.9475197	yes
5	Goal	No Goal	0.081759850	0.9182401	no
6	Goal	No Goal	0.290885427	0.7091146	no
7	No Goal	No Goal	0.324689547	0.6753105	yes
8	No Goal	No Goal	0.071013339	0.9289867	yes
9	No Goal	No Goal	0.022011345	0.9779887	yes
10	No Goal	No Goal	0.020334023	0.9796660	yes
11	Goal	Goal	0.784372930	0.2156271	yes
12	Goal	No Goal	0.335888341	0.6641117	no
13	Goal	No Goal	0.116220474	0.8837795	no
14	No Goal	No Goal	0.121866754	0.8781332	yes
15	No Goal	No Goal	0.037165586	0.9628344	yes
16	No Goal	No Goal	0.066307721	0.9336923	yes
17	Goal	No Goal	0.084223928	0.9157761	no
18	No Goal	No Goal	0.352337266	0.6476627	yes
19	No Goal	No Goal	0.144230133	0.8557699	yes
20	No Goal	No Goal	0.073254979	0.9267450	yes
21	No Goal	No Goal	0.211066901	0.7889331	yes

Figure 9: Snapshot of Prediction Probabilities



```
+    summarise(n=n())
`summarise()` regrouping output by 'actual'
# A tibble: 3 x 3
# Groups:   actual [2]
  actual  predict     n
  <chr>   <fct>   <int>
1 Goal    Goal        4
2 Goal    No Goal    43
3 No Goal No Goal   243
```

Figure 10: Summary of Predictions

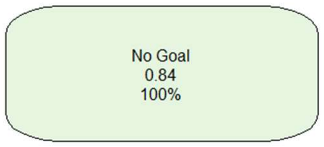
The next step is using the neural network model to predict the classes of the shots in the dataset. Using the function *h2o.predict(),* we obtain the prediction probabilities shown in Figure 9 along with whether the prediction was accurate or not. This is only a snapshot of the prediction probabilities but a summary of them can be seen in Figure 10. This summary shows that only 4 goals were accurately predicted and 243 non goals were accurately predicted. 43 actual goals were incorrectly classified as non-goals. Again, this indicates that the neural network model is not a good classifier of the model.

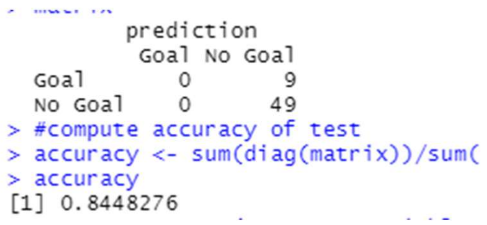
Classification Trees

Classification trees is the last classification method that will be applied to the dataset. The first step is to create a training and test set. The training set will be used to predict the classes of the test set. The training set was 80% of the dataset and the test set was the held-out 20% of the dataset. To see the importance of the predictor variables, the tree will be built one variable at a time. The first tree's model is,

tree_1 <- rpart(response_variable ~ side_of_rink,train_set,method='class')

and the plot is shown in Figure 11. For the first tree we used the *side of rink* variable and the accuracy of the test was 84.4%. 0 goals were correctly classified and, as there is just the root node in the plot, this indicates that the side of rink is not a significant variable. In the second tree the variable *type of shot* was added into the model formula. This produced the same plot and prediction accuracy as the first tree. This indicates that the type of shot is not significant either.





```
          prediction
           Goal No Goal
  Goal        0       9
  No Goal     0      49
> #compute accuracy of test
> accuracy <- sum(diag(matrix))/sum(
> accuracy
[1] 0.8448276
```

Figure 11: Tree 1 Plot & Prediction Accuracy

The third tree added in the variable *shot distance* and the model can be seen here,

tree_3 <- rpart(response_variable ~ side_of_rink + type_of_shot_all + distance,train_set,method='class').

In the plot of the third tree, the classification tree has child nodes now and we can see how the variable *distance* affects the classes of the response variable.

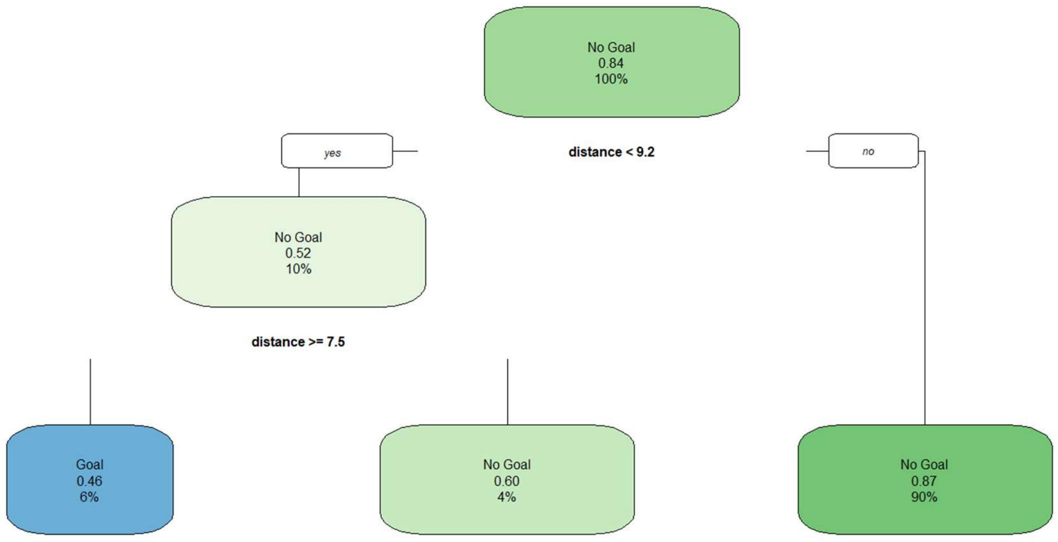
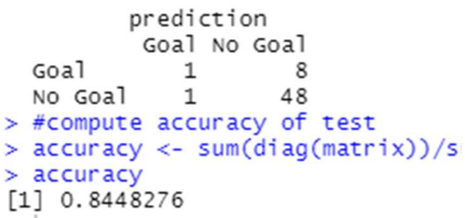


Figure 12: Tree 3

The prediction accuracy remained the same, but 1 goal was predicted correctly this time. The variable distance has more of an impact on predicting scoring a goal than the two previous variables.



```
         prediction
          Goal No Goal
  Goal       1       8
  No Goal    1      48
> #compute accuracy of test
> accuracy <- sum(diag(matrix))/s
> accuracy
[1] 0.8448276
```

Figure 13: Prediction Accuracy of Tree 3

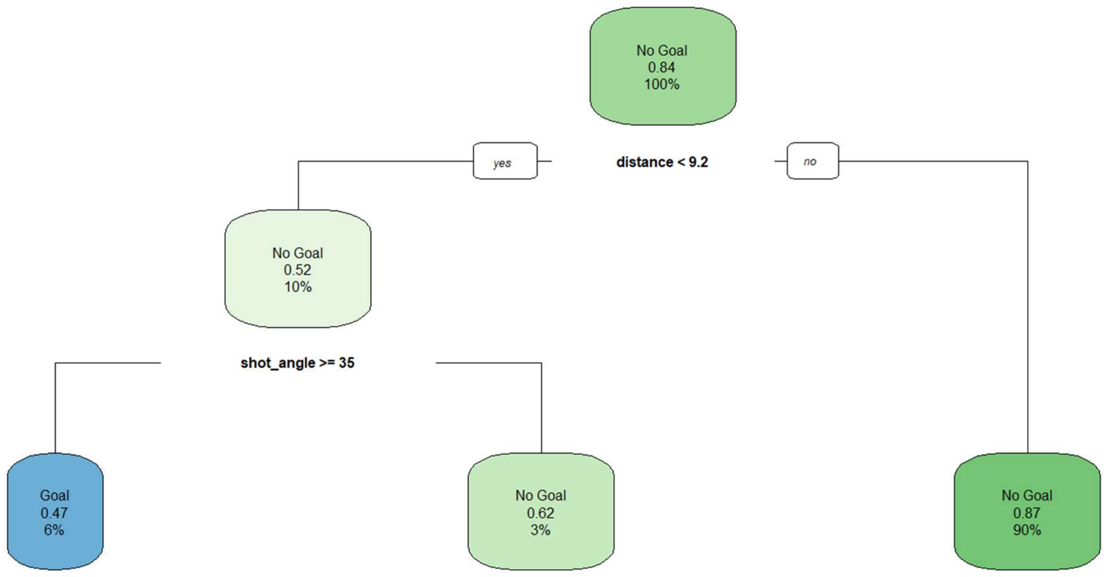
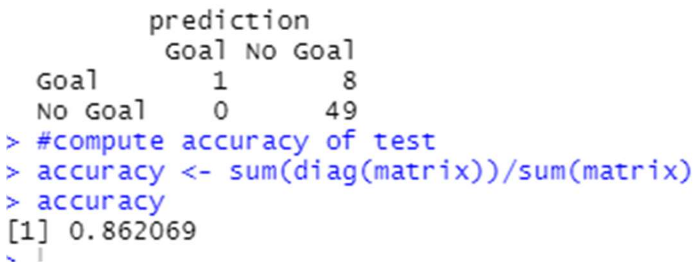


Figure 14: Tree 4

The model is then built using 'rpart' and it appears in the code as,

tree <- rpart(response_variable ~ side_of_rink + type_of_shot_all+ distance+ shot_angle,train_set,method='class')

The classification tree is plotted and is shown in Figure 14. In Fig.14 the top node represents the overall probability of 'No Goal' in the training set. 84% of the training set is classed as 'No Goal'. To go down to the next node, you can see that you can either go left or right. If distance > 9.2, we can proceed to the right node. Here we can see that for *shot distance* greater than 9.2 the prediction is no goal with probability 87%. For *shot distance* less than 9.2, we proceed to the left child node and then the next question is whether the *shot angle* is greater or less than 35. If less than 35, then the classifier predicts a goal with probability of 47%. If the shot angle is greater than or equal to 35, then the probability of it not being a goal is 62%. Next, using the tree built and the training set, we predict results for the test set. The confusion matrix for the predictions can be seen in Figure 15 along with the accuracy of the model.



```
          prediction
           Goal No Goal
  Goal        1       8
  No Goal     0      49
> #compute accuracy of test
> accuracy <- sum(diag(matrix))/sum(matrix)
> accuracy
[1] 0.862069
```

Figure 15: Prediction of the Test Set & Accuracy of Tree 4

The model predicted 1 goal out of 9 correctly and 49 non goals correctly. It misclassified 8 goals as non-goals; none of the non-goals were misclassified. The accuracy of the model is calculated from the confusion matrix as 86.2%. This is a higher accuracy percentage than the previous trees and seems to indicate that shot angle and distance have the most impact on predicting scoring a goal or not.

Conclusion

The goal of this project was to attempt to find a good classifier model for the shooting data of Auston Matthews. Three classification techniques were used to build a model to classify the binary response variable and to test the significance of the predictor variables. The three methods were logistic regression, neural networks, and classification trees.

The logistic regression model that was built was evaluated using the likelihood ratio test and the pseudo R^2 test. These tests showed that the model was not a good classifier for the data.

The second classification method was a neural network. The neural network model was evaluated by the classification error, log loss, MSE, and AUC. Though the log loss, MSE, and AUC showed that the model was good, the high classification error told a different story.

The last method used on the dataset was a classification tree using 'rpart'. The classification tree was the best classifier out of the three methods. It had a high prediction accuracy of about 86% and showed that the most significant predictor variables were shot angle and distance.

In all cases, the classifiers had difficulty due to the imbalance in the dataset. The dataset had several more non-goals than goals and in this case the classifier had trouble correctly classifying the smaller class: goals.

7. References

[1] A. (2015, August 17). Evaluating Logistic Regression Models [Web log post]. Retrieved February 14, 2021, from https://www.r-bloggers.com/2015/08/evaluating-logistic-regression-models/

[2] Allison, P. (2019, November 27). What's the Best R-squared for Logistic Regression? Retrieved April 17, 2021, from https://statisticalhorizons.com/r2logistic

[3] Allison, P. D. (2012). *Logistic Regression Using SAS: Theory and Application* (2nd ed.). Cary, NC: SAS Institute. Retrieved April 3, 2021, from https://learning.oreilly.com/library/view/logistic-regression-using/9781607649953/.

[4] A., M. (2019, June 17). Retrieved April 12, 2021, from https://github.com/mtthwastn/statswithmatt/blob/master/hockey-with-r/hockey-rink.R.

[5] Barlowe, M. (2017, September 16). NHL Expected Goals Model. Retrieved January 21, 2021, from https://rstudio-pubs-static.s3.amazonaws.com/311470_f6e88d4842da46e9941cc6547405a051.html.

[6] Basen, N. (2019, January 23). How Analytics Will Help Win the Next Stanley Cup. *The Walrus*. Retrieved March 27, 2021, from https://thewalrus.ca/how-analytics-will-help-win-the-next-stanley-cup/

[7] Berkshire, A. (2019, May 01). Why this year's Carolina Hurricanes were finally able to break through. Retrieved March 28, 2021, from https://www.sportsnet.ca/hockey/nhl/years-carolina-hurricanes-finally-able-break/

[8] Dtmaboutheart. (2015, October 1). Expected Goals are a better predictor of future scoring than Corsi, Goals. Retrieved April 05, 2021, from https://rpubs.com/evolvingwild/395136/.

[9] Elsinghorst, S. (2017, February 27). Building deep neural nets with h2o and rsparkling that predict arrhythmia of the heart. Retrieved April 2, 2021, from https://shiring.github.io/machine_learning/2017/02/27/h2o

[10] Hynes, D. (2020). NHL API Documentation. Retrieved November 23, 2020, from https://gitlab.com/dword4/nhlapi

[11] Khan, R. (2020, December 08). Nothing but Numpy: Understanding & Creating Binary Classification Neural Networks with Computational Graphs from Scratch. Retrieved April 3, 2021, from https://towardsdatascience.com/nothing-but-numpy-understanding-creating-binary-classification-neural-networks-with-e746423c8d5c#:~:text=In%20binary%20classification%20tasks%2C%20it,probability%20is%20below%20the%20threshold.

[12] Kleinbaum, D. G., & Klein, M. (2010). *Logistic regression: A self-learning text* (3rd ed.). New York: Springer. Retrieved April 1, 2021, from https://books-scholarsportal-info.proxy.library.carleton.ca/uri/ebooks/ebooks2/springer/2011-04-28/2/9781441917423.

[13] Le, J. (2018, June 19). Decision Trees in R. Retrieved April 16, 2021, from https://www.datacamp.com/community/tutorials/decision-trees-R

[14] Logistic Regression. (2015). Retrieved April 16, 2021, from https://projects.upei.ca/mer/files/2015/04/pages_from_mer16_121119first10.pdf.

[15] Macdonald, Brian. (2012). *An Expected Goals Model for Evaluating NHL Teams and Players.* http://www.hockeyanalytics.com/Research_files/NHL-Expected-Goals-Brian-Macdonald.pdf

[16] Mills, Shirley. (2021) STAT5703: Data Mining I, Section 8 Classification Methods. Retrieved April 1, 2021, from https://culearn.carleton.ca/moodle/pluginfile.php/4382810/mod_resource/content/1/Section%2008-18.pdf

[17] *Neural networks in R Tutorial (H2O tutorial in R: how to create an Ann for binary classification)* [Video file]. (2021, March 21). Retrieved March 29, 2021, from https://www.youtube.com/watch?v=Ck10_VtN_88&t=3s

[18] Novet, A. (2019, August 16). Expected goals model with Pre-shot MOVEMENT, Part 1: The model. Retrieved April 05, 2021, from https://hockey-graphs.com/2019/08/12/expected-goals-model-with-pre-shot-movement-part-1-the-model/.

[19] O'Connor, C. (2017, November 25). An advanced stat primer: Understanding basic hockey metrics. Retrieved March 29, 2021, from https://theathletic.com/121980/2017/10/09/an-advanced-stat-primer-understanding-basic-hockey-metrics/

[20] Pastor, K. (2020, April 24). NHL Analytics With Python. Retrieved April 05, 2021, from https://towardsdatascience.com/nhl-analytics-with-python-6390c5d3206

[21] Rosario, S. (2017, June 7). Appreciating The Importance of Sports Analytics in Hockey [Web log post]. Retrieved March 18, 2021, from https://www.workinsports.com/blog/appreciating-the-importance-of-sports-analytics-in-hockey/

[22] Tan, P., Steinbach, M., Karpatne, A., & Kumar, V. (2020). Classification: Basic Concepts, Decision Trees, and Model Evaluation. In *Introduction to Data Mining* (2nd ed., pp. 146-205). New

York, NY: Pearson. Retrieved April 3, 2021, from https://www-users.cs.umn.edu/~kumar001/dmbook/ch4.pdf

Appendix: Code

```
#install packages needed
install.packages("RJSONIO")
library("RJSONIO")
install.packages("tidyverse")
library(tidyverse)
install.packages("ggplot2")
library(ggplot2)
install.packages("jsonlite")
library(jsonlite)
install.packages("devtools")
library(devtools)
require(devtools)
install_version("jsonlite", version = "1.7.1", repos = "http://cran.us.r-project.org")
#access games Auston Matthews played
AustonMatthews <-
fromJSON("http://statsapi.web.nhl.com/api/v1/people/8479318/stats?stats=gameLog&season=2019
2020")

AustonMatthews$games <-
c(2019021070,2019021048,2019021041,2019021028,2019021003,2019020987,2019020970,20190
20951,2019020933,2019020921,2019020913,2019020901,2019020881,2019020867,2019020845,2
019020839,2019020825,2019020809,2019020795,2019020777,2019020770,

2019020747,2019020728,2019020715,2019020707,2019020675,2019020659,2019020646,201902
0633,2019020616,2019020593,2019020583,2019020569,2019020556,2019020550,2019020525,20
19020511,2019020494,

2019020481,2019020459,2019020434,2019020428,2019020405,2019020392,2019020379,201902
0357,2019020343,2019020329,2019020308,2019020294,2019020282,2019020268,2019020253,20
19020237,2019020228,2019020209,2019020180,

2019020163,2019020156,2019020133,2019020129,2019020115,2019020093,2019020085,201902
0066,2019020047,2019020034,2019020019,2019020015,2019020001)


#extract goals scored by Auston Matthews
x <- {}
y <- {}
for(i in 1:length(AustonMatthews$games)){
  game <-
fromJSON(paste("http://statsapi.web.nhl.com/api/v1/game/",AustonMatthews$games[i],"/feed/live
"))
  for (j in 1:length(game$liveData$plays$allPlays$result$event)){
    Goal <- game$liveData$plays$allPlays$result$event[j]%in%c("Goal")
    Player <- game$liveData$plays$allPlays$players[[j]]$player$fullName%in%"Auston Matthews"
    PlayerType <-
game$liveData$plays$allPlays$players[[j]]$playerType%in%c("Shooter","Scorer")
    if(Goal&&Player&&PlayerType){
      x<-append(x,game$liveData$plays$allPlays$coordinates$x[j]);
      y<-append(y,game$liveData$plays$allPlays$coordinates$y[j])
    }
  }
}

#remove element 19 - not a goal
x_goal <- x[-19]
y_goal <- y[-19]

#power play goals
power_play_goals_x <- c(-78,71,77,86,-73,-80,-71,87,-58,-65,-66,65)
power_play_goals_y <- c(18,-12,-14,-8,15,18,-6,-7,24,-8,25,-20)

#extract shots taken by Auston Matthews that were not goals
a <- {}
b <- {}
for(i in 1:length(AustonMatthews$games)){
  game <-
fromJSON(paste("http://statsapi.web.nhl.com/api/v1/game/",AustonMatthews$games[i],"/feed/live
"))
  for (j in 1:length(game$liveData$plays$allPlays$result$event)){
    Shot <- game$liveData$plays$allPlays$result$event[j]%in%c("Shot")
    Player <- game$liveData$plays$allPlays$players[[j]]$player$fullName%in%"Auston Matthews"
    PlayerType <-
game$liveData$plays$allPlays$players[[j]]$playerType%in%c("Shooter","Scorer")
    if(Shot&&Player&&PlayerType){
      a<-append(a,game$liveData$plays$allPlays$coordinates$x[j]);
      b<-append(b,game$liveData$plays$allPlays$coordinates$y[j])
    }
  }
}
#remove deflected shots
a<- a[-110]
a<- a[-130]
```

```
a<- a[-234]
b<- b[-110]
b<- b[-130]
b<- b[-234]


#add the goals and shots together for x and y coordinates
all_x_shots <- append(x_goal,a,after=length(x_goal))
all_y_shots <- append(y_goal,b,after=length(y_goal))

#extract the shot style for all shots that were not goals
type_of_shot <- {}
type_of_shot_goals <- {}
for(i in 1:length(AustonMatthews$games)){
  game <-
fromJSON(paste("http://statsapi.web.nhl.com/api/v1/game/",AustonMatthews$games[i],"/feed/live
"))
  for (j in 1:length(game$liveData$plays$allPlays$result$event)){
    Shot <- game$liveData$plays$allPlays$result$event[j]%in%c("Shot")
    Player <- game$liveData$plays$allPlays$players[[j]]$player$fullName%in%"Auston Matthews"
    PlayerType <-
game$liveData$plays$allPlays$players[[j]]$playerType%in%c("Shooter","Scorer")
    if(Shot&&Player&&PlayerType){
      type_of_shot<-append(type_of_shot,game$liveData$plays$allPlays$result$secondaryType[j])
    }
  }
}
#remove deflected shots
type_of_shot <- type_of_shot[-110]
type_of_shot <- type_of_shot[-130]
type_of_shot <- type_of_shot[-234]
#extract the shot style for all goals
for(i in 1:length(AustonMatthews$games)){
  game <-
fromJSON(paste("http://statsapi.web.nhl.com/api/v1/game/",AustonMatthews$games[i],"/feed/live
"))
  for (j in 1:length(game$liveData$plays$allPlays$result$event)){
    Goal <- game$liveData$plays$allPlays$result$event[j]%in%c("Goal")
    Player <- game$liveData$plays$allPlays$players[[j]]$player$fullName%in%"Auston Matthews"
    PlayerType <-
game$liveData$plays$allPlays$players[[j]]$playerType%in%c("Shooter","Scorer")
    if(Goal&&Player&&PlayerType){
      type_of_shot_goals<-
```

```
append(type_of_shot_goals,game$liveData$plays$allPlays$result$secondaryType[j])
    }
  }
}
#remove element 19 - not a goal
type_of_shot_goals_fixed <- type_of_shot_goals[-19]

#combine type of shot for shots and goals
type_of_shot_all <- append(type_of_shot_goals_fixed,type_of_shot)

#making the hockey rink
library(ggplot2)

gg_rink <- function(side = "right", specs = "nhl"){

  ### this function uses ggplot's annotate()
  # to draw the rink.
  # I recommend calling this function PRIOR to invoking
  # geoms for data so that the points are not covered
  # by the annotations

  ### inputs:
  #     1. side = which side to plot: "right" (default) or "left"
  #     2. specs = which rink size to use: "nhl" (default) or "iihf" for
  #        international

  # check inputs
  side <- tolower(side)
  specs <- tolower(specs)
  stopifnot(side %in% c("right", "left"))
  stopifnot(specs %in% c("nhl", "iihf"))

  side <- switch(side,
                 "right" = 1,
                 "left" = -1)

  nsteps <- 1001 # line resolution for drawing circles/segments
  circle <- seq(0, 2*pi, length = nsteps) # angles to draw a circle

  switch(specs,
         "nhl" = {
           # NHL specifications
           # all units in feet
```

```
### rink boundaries ###
## assumed to be standard 200x85ft dimensions
x_max <- 100
y_max <- 42.5
y_min <- -y_max
# blue line 75' from end boards
x_blue <- x_max - 75
# goal line 11' from end boards
x_goal <- x_max - 11

### parameter setup
## corners rounded in arc of circle with 28' radius
r_corner <- 28

## crease semi-circle
# 6' radius from center of goal line starting 4.5' out
crease_end <- 4.5
r_crease <- 6
# deepest point of net is 40"
net_depth <- 40/12
# crease is 8' long; goal posts 6' apart
goal_post_start <- 6/2
crease_start_y <- 8/2
# inner crease lines begin 4' from goal line
# extend 5" into crease
crease_small_start <- 4
crease_small_length <- 5/12

## face-off circle dots and lines
# dot locations: 20' from goal line, 22' in each y direction
x_dot_dist <- 20
y_faceoff_dot <- 22
# face-off circle radius 15'
r_faceoff <- 15
# hash marks 2' long, 5'7" apart
hash_length <- 2
hash_space <- 67/12
# circle inner lines:
# x-direction: lines 4' apart, so start 2' from dot
# y-direction: lines 18" apart, so start 9" from dot
inner_start_x <- 2
inner_start_y <- 1.5/2
```

```
  # lines parallel to side boards: 4' long
  par_side_length <- 4
  # lines parallel to end boards: 3' long
  par_end_length <- 3

  ## other parameters
  # neutral zone dots are 5' from blue line, 44' apart
  x_dot_neutral <- 5
  # ref circle 5m radius
  r_ref <- 5
  ## trapezoid (NHL only)
  # begins 8' from each goal post
  # bottom base is 28' long
  y_traps_start <- goal_post_start + 8
  y_traps_end <- 14
},
"iihf" = {
  # IIHF specifications
  # all units in meters

  ### rink boundaries ###
  ## assumed to be standard 60x30m dimensions
  x_max <- 30
  y_max <- 15
  y_min <- -y_max
  # blue line 22.86m from end boards, 30cm wide
  x_blue <- x_max - 22.86
  # goal line 4m from end boards
  x_goal <- x_max - 4

  ### parameter setup
  ## corners rounded in arc of circle with 8.5m radius
  r_corner <- 8.5

  ## crease semi-circle
  # 183cm radius from center of goal line starting 137cm out
  crease_end <- 1.37
  r_crease <- 1.83
  # deepest point of net is 1.12m
  net_depth <- 1.12
  # crease is 244cm long; goal posts 183.5cm apart
  goal_post_start <- 1.835/2
  crease_start_y <- 2.44/2
```

```
    # inner crease lines begin 122cm from goal line
    # extend 13m into crease
    crease_small_start <- 1.22
    crease_small_length <- 0.13

    ## face-off circle dots and lines
    # dot locations: 6m from goal line, 7m in each y direction
    x_dot_dist <- 6
    y_faceoff_dot <- 7
    # face-off circle radius 4.5m
    r_faceoff <- 4.5
    # hash marks 60cm long, 170cm apart
    hash_length <- 0.6
    hash_space <- 1.7
    # circle inner lines:
    # x-direction: lines 120cm apart, start 60cm from dot
    # y-direction: lines 45cm apart, so start 22.5cm from dot
    inner_start_x <- 0.6
    inner_start_y <- 0.225
    # lines parallel to side boards: 120cm long
    par_side_length <- 1.2
    # lines parallel to end boards: 90cm long
    par_end_length <- 0.9

    ## other parameters
    # neutral zone dots are 1.5m from blue line
    x_dot_neutral <- 1.5
    # ref circle 3m radius
    r_ref <- 3
  }
)

## corners
curve_angle <- seq(pi/2, 0, length = nsteps)
curve_angle_last <- curve_angle[nsteps]
# y coord at end of curve to connect ends
y_curve_end <- (y_max - r_corner) + r_corner*sin(curve_angle_last)
# for goal line, find y coord when x is at goal line
goal_angle <- acos(
  (x_goal - (x_max - r_corner))/r_corner
)
y_goal <- (y_max - r_corner) + r_corner*sin(goal_angle)
```

```
## crease
crease_angles <- seq(
  pi - acos(crease_end/r_crease),
  pi + acos(crease_end/r_crease),
  length = nsteps
)

## face-off circle
x_faceoff_dot <- x_goal - x_dot_dist
# find y coord on circle where hashes begin
y_hash <- r_faceoff*sin(
  acos((hash_space/2)/r_faceoff)
)

### create list of components to pass to ggplot
list(
  theme_minimal(),
  theme(panel.grid = element_blank()),
  ### blue line
  annotate(
    "segment",
    x = x_blue*side, y = y_max,
    xend = x_blue*side, yend = y_min,
    color = "blue", size = 2
  ),
  ### ref crease
  annotate(
    "path",
    x = r_ref*cos(seq(pi/2, 0, length = nsteps))*side,
    y = y_min + r_ref*sin(seq(pi/2, 0, length = nsteps)),
    color = "red"
  ),
  ### face-off circle, center ice
  annotate(
    "path",
    x = r_faceoff*cos(seq(pi/2, -pi/2, length = nsteps))*side,
    y = r_faceoff*sin(seq(pi/2, -pi/2, length = nsteps)),
    color = "blue"
  ),
  ### center line:
  annotate(
    "segment",
    x = 0, y = y_max,
  xend = 0, yend = y_min,
  color = "red", size = 2
),
switch(specs,
       "nhl" = annotate(
         # dashed white lines atop center line (NHL only)
         "segment",
         x = 0, y = y_max,
         xend = 0, yend = y_min,
         color = "white", size = 0.5, linetype = "dashed"
       ),
       "iihf" = annotate(
         # 50cm space between lines around center dot
         "segment",
         x = 0, y = 0.5,
         xend = 0, yend = -0.5,
         color = "white", size = 2.5
       )
),
### face-off dot, center ice
annotate(
  "point",
  x = 0,
  y = 0,
  color = "blue", size = 1
),
### neutral zone dots
annotate(
  "point",
  x = (x_blue - x_dot_neutral)*side,
  y = y_faceoff_dot*c(1, -1),
  color = "red", size = 1
),
### side boards
annotate(
  "segment",
  x = 0, y = c(y_min, y_max),
  # stop where corner curve begins
  xend = (x_max - r_corner)*side, yend = c(y_min, y_max),
  size = 1
),
### ends
# goal line
```

```
annotate(
  "segment",
  x = x_goal*side, y = y_goal,
  xend = x_goal*side, yend = -y_goal,
  color = "red"
),
# connect ends
annotate(
  "segment",
  x = x_max*side, y = y_curve_end,
  xend = x_max*side, yend = -y_curve_end,
  size = 1
),
# corners rounded in arc of circle
# starting point: (x_max, y_max) - r_circle from pi/2 to 0
annotate(
  "path",
  x = ((x_max - r_corner) + r_corner*cos(curve_angle))*side,
  y = (y_max - r_corner) + r_corner*sin(curve_angle),
  size = 1
),
annotate(
  "path",
  x = ((x_max - r_corner) + r_corner*cos(curve_angle))*side,
  y = -((y_max - r_corner) + r_corner*sin(curve_angle)),
  size = 1
),
### crease
annotate(
  "segment",
  x = x_goal*side,
  y = crease_start_y*c(-1, 1),
  xend = (x_goal - crease_end)*side,
  yend = crease_start_y*c(-1, 1),
  col = "red"
),
# crease lines
annotate(
  "segment",
  x = (x_goal - crease_small_start)*side,
  y = crease_start_y*c(-1, 1),
  xend = (x_goal - crease_small_start)*side,
  yend = (crease_start_y - crease_small_length)*c(-1, 1),
  col = "red"
),
# semi-circle starting 137cm out with 183cm radius from center of goal line
annotate(
  "path",
  x = (x_goal + r_crease*cos(crease_angles))*side,
  y = r_crease*sin(crease_angles),
  col = "red"
),
if (specs == "nhl") {
  ### restricted area (NHL only)
  annotate(
    "segment",
    x = x_goal*side, y = y_traps_start*c(-1, 1),
    xend = x_max*side, yend = y_traps_end*c(-1, 1),
    color = "red"
  )
},
### net
annotate(
  "segment",
  x = x_goal*side,
  y = goal_post_start*c(-1, 1),
  xend = (x_goal + net_depth)*side,
  yend = goal_post_start*c(-1, 1)
),
annotate(
  "segment",
  x = (x_goal + net_depth)*side,
  y = -goal_post_start,
  xend = (x_goal + net_depth)*side,
  yend = goal_post_start
),
### face-off circles
# dot
annotate(
  "point",
  x = x_faceoff_dot*side,
  y = y_faceoff_dot*c(1, -1),
  col = "red",
  size = 1
),
# circles
```

```
annotate(
  # top
  "path",
  x = side*(x_faceoff_dot + r_faceoff*cos(circle)),
  y = y_faceoff_dot + r_faceoff*sin(circle),
  col = "red"
),
annotate(
  # bottom
  "path",
  x = side*(x_faceoff_dot + r_faceoff*cos(circle)),
  y = -(y_faceoff_dot + r_faceoff*sin(circle)),
  col = "red"
),
# hashes
annotate(
  "segment",
  x = side*(
    x_faceoff_dot + (hash_space/2)*rep(c(1, -1), each = 4)
  ),
  y = (y_faceoff_dot + y_hash*c(1, -1))*rep(c(1, 1, -1, -1), times = 2),
  xend = side*(
    x_faceoff_dot + (hash_space/2)*rep(c(1, -1), each = 4)
  ),
  yend = (y_faceoff_dot + (y_hash + hash_length)*c(1, -1))*
    rep(c(1, 1, -1, -1), times = 2),
  col = "red"
),
## inner lines
# parallel to side boards
annotate(
  # parallel to side boards
  "segment",
  x = side*(
    x_faceoff_dot + inner_start_x*rep(c(1, -1), each = 4)
  ),
  y = (y_faceoff_dot + inner_start_y*c(1, -1))*
    rep(c(1, 1, -1, -1), times = 2),
  xend = side*(
    x_faceoff_dot + (inner_start_x + par_side_length)*
      rep(c(1, -1), each = 4)
  ),
  yend = (y_faceoff_dot + inner_start_y*c(1, -1))*
```

```
        rep(c(1, 1, -1, -1), times = 2),
      col = "red"
    ),
    annotate(
      # parallel to end boards
      "segment",
      x = side*(
        x_faceoff_dot + inner_start_x*rep(c(1, -1), each = 4)
      ),
      y = (y_faceoff_dot + inner_start_y*c(1, -1))*
        rep(c(1, 1, -1, -1), times = 2),
      xend = side*(
        x_faceoff_dot + inner_start_x*rep(c(1, -1), each = 4)
      ),
      yend = (y_faceoff_dot + (inner_start_y + par_end_length)*c(1, -1))*
        rep(c(1, 1, -1, -1), times = 2),
      col = "red"
    )
  )
}
#create a dataframe to plot the data
data.frame(df,stringAsFactors=TRUE)

goals <- data.frame(x_goal,y_goal)
power_play_goals <- data.frame(power_play_goals_x,power_play_goals_y)
just_shots <- data.frame(a,b)
shots <- data.frame(all_x_shots, all_y_shots)
df <- data.frame(shots,type_of_shot_all)
df <- data.frame(all_x_shots,all_y_shots)
#Plotting the Rink
ggplot(just_shots, aes(x = a, y = b)) +
  gg_rink(side = "right") +
  gg_rink(side = "left")+
  geom_point(aes(color ='red'),
             position = "jitter", size = 1.5, alpha = 0.7)+
  geom_point(data=goals,aes(x=x_goal,y=y_goal,color='green'),size=1.5,alpha=0.7,show.legend =
TRUE)+

geom_point(data=power_play_goals,aes(x=power_play_goals_x,y=power_play_goals_y,color='blu
e'),size=1.5,alpha=0.7,show.legend=TRUE)+
  scale_color_identity(name = "Type of Shot",breaks = c("red","green","blue"),labels =
c("Shots","Goals","Power Play Goals"),guide="legend")+
  labs(title = "Shot chart: Auston Matthews",
      # subtitle = "NHL rink",
      x = NULL,
      y = NULL) +

  scale_x_continuous(breaks = seq(-100, 100, by = 10)) +
  scale_y_continuous(breaks = seq(-40, 40, by = 10))

side_of_rink <-
c("left","right","left","right","right","left","right","right","left","right","left","right","left","right","le
ft","left","left","left","left","right","left","right","left","left","left","left","right","right","left","left","
right","right","left","right","left","left","right","left","left","left","right","left","left","left","left","rig
ht","right","left","left","left","right","right","left","left","right","right","left","right","left","right","le
ft","left","left","left","right","left","left","left","left","right","right","left","left","right","left","right",
"middle","left","left","right","right","left","left","right","right","left","left","right","left","right","rig
ht","left","right","right","left","left","right","right","left","right","right","right","right","left","right",
"right","right","left","left","right","right","right","right","right","right","left","left","right","right","r
ight","left","left","left","left","right","right","left","right","left","left","left","left","right","right","ri
ght","left","left","left","right","right","right","left","left","left","left","left","left","right","right","left
","left","left","left","left","left","right","left","left","left","right","right","left","right","left","left","ri
ght","left","right","right","left","left","left","left","right","right","left","left","left","right","right","le
ft","left","left","right","left","left","right","right","left","right","right","left","left","left","left","left",
"left","left","right","left","right","left","left","left","left","right","left","left","left","right","left","left
","right","right","right","left","left","left","left","left","left","left","left","right","left","right","right",
"left","left","left","right","right","right","right","right","right","left","right","left","left","right","left
","left","right","right","left","left","left","left","right","left","left","right","left","left","right","right",
"left","left","left","right","left","left","right","right","left","right","right","right","left","right","left",
"left","right","right","right","left","left","right","left","left","left","right","left","left","right","right",
"right","left","left")
#create dataframe with shot coordinates, type of shot and side of rink
df <- data.frame(all_x_shots,all_y_shots)
df <- cbind(df,type_of_shot_all)
df <- cbind(df,side_of_rink)

#call dataframe Matthews_Data
Matthews_Data <- cbind(df,side_of_rink)
#create column response variable
response_variable <-
c("Goal","Goal","Goal","Goal","Goal","Goal","Goal","Goal","Goal","Goal","Goal","Goal","Goal",
"Goal","Goal","Goal","Goal","Goal","Goal","Goal","Goal","Goal","Goal","Goal",

"Goal","Goal","Goal","Goal","Goal","Goal","Goal","Goal","Goal","Goal","Goal","Goal","Goal","
Goal","Goal","Goal","Goal","Goal","Goal","Goal","Goal","Goal","Goal",
                    "No Goal","No Goal","No Goal","No Goal","No Goal","No Goal","No Goal","No
```

```
Goal","No Goal","No Goal","No Goal","No Goal","No Goal","No Goal","No Goal","No
Goal","No Goal",
                "No Goal","No Goal","No Goal","No Goal","No Goal","No Goal","No Goal","No
Goal","No Goal","No Goal","No Goal","No Goal","No Goal","No Goal","No Goal","No
Goal","No Goal","No Goal","No Goal","No Goal","No Goal",
                "No Goal","No Goal","No Goal","No Goal","No Goal","No Goal","No Goal","No
Goal","No Goal","No Goal","No Goal","No Goal","No Goal","No Goal","No Goal","No
Goal","No Goal","No Goal","No Goal","No Goal","No Goal","No Goal","No Goal","No Goal",
                "No Goal","No Goal","No Goal","No Goal","No Goal","No Goal","No Goal","No
Goal","No Goal","No Goal","No Goal","No Goal","No Goal","No Goal","No Goal","No
Goal","No Goal","No Goal","No Goal","No Goal","No Goal","No Goal","No Goal","No
Goal","No Goal",
                "No Goal","No Goal","No Goal","No Goal","No Goal","No Goal","No Goal","No
Goal","No Goal","No Goal","No Goal","No Goal","No Goal","No Goal","No Goal","No
Goal","No Goal","No Goal","No Goal","No Goal","No Goal","No Goal","No Goal","No
Goal","No Goal","No Goal",
                "No Goal","No Goal","No Goal","No Goal","No Goal","No Goal","No Goal","No
Goal","No Goal","No Goal","No Goal","No Goal","No Goal","No Goal","No Goal","No
Goal","No Goal","No Goal","No Goal","No Goal","No Goal","No Goal","No Goal","No
Goal","No Goal","No Goal",
                "No Goal","No Goal","No Goal","No Goal","No Goal","No Goal","No Goal","No
Goal","No Goal","No Goal","No Goal","No Goal","No Goal","No Goal","No Goal","No
Goal","No Goal","No Goal","No Goal","No Goal","No Goal","No Goal","No Goal","No
Goal","No Goal","No Goal","No Goal","No Goal","No Goal","No Goal",
                "No Goal","No Goal","No Goal","No Goal","No Goal","No Goal","No Goal","No
Goal","No Goal","No Goal","No Goal","No Goal","No Goal","No Goal","No Goal","No
Goal","No Goal","No Goal","No Goal","No Goal","No Goal","No Goal","No Goal","No Goal",
                "No Goal","No Goal","No Goal","No Goal","No Goal","No Goal","No Goal","No
Goal","No Goal","No Goal","No Goal","No Goal","No Goal","No Goal","No Goal","No
Goal","No Goal","No Goal","No Goal","No Goal","No Goal","No Goal","No Goal","No
Goal","No Goal",
                "No Goal","No Goal","No Goal","No Goal","No Goal","No Goal","No Goal","No
Goal","No Goal","No Goal","No Goal","No Goal","No Goal","No Goal","No Goal","No
Goal","No Goal","No Goal","No Goal","No Goal","No Goal","No Goal","No Goal","No
Goal","No Goal")

#converting coordinates into shot angle and distance
df$all_y_shots <- ifelse(Matthews_Data$all_x_shots < 0, -1 * Matthews_Data$all_y_shots,
Matthews_Data$all_y_shots)
df$all_x_shots <- abs(Matthews_Data$all_x_shots)
df$shot_angle <- (asin(abs(Matthews_Data$all_y_shots)/sqrt((87.95-
abs(Matthews_Data$all_x_shots))^2+Matthews_Data$all_y_shots^2))*180/3.14)
df$shot_angle <- ifelse(abs(Matthews_Data$all_x_shots)>88,90+(180-
(90+Matthews_Data$shot_angle)),Matthews_Data$shot_angle)
df$distance <- sqrt((87.95-abs(Matthews_Data$all_x_shots))^2+Matthews_Data$all_y_shots^2)
#add binary response column
Matthews_Data <- cbind(df,response_variable)
#take out columns for x and y coordinates
Matthews_Data <- subset(Matthews_Data,select=-c(all_x_shots,all_y_shots))

#relevel(Matthews_Data$response_variable,ref='No Goal')

#creating logistic model
logistic_model <- glm(as.factor(response_variable) ~ poly(distance,3, raw=TRUE) +
poly(shot_angle,3,raw=TRUE) + side_of_rink + type_of_shot_all,data=Matthews_Data, family =
binomial(link='logit'))

#summary and coefficients of model
summary(logistic_model)
coef(logistic_model)
#creating second model for liklihood ratio test
logistic_model_2 <- glm(as.factor(response_variable) ~ poly(distance,3, raw=TRUE) +
poly(shot_angle,3,raw=TRUE) + side_of_rink,data=Matthews_Data, family=binomial(link='logit'))

#evaluating the model
#likelihood ratio test
install.packages("lmtest")
library(lmtest)
lrtest(logistic_model,logistic_model_2)
anova(logistic_model,logistic_model_2, test="Chisq")

#pseudo r^2 test
install.packages("pscl")
library(pscl)
pR2(logistic_model)

#Creating a Neural Network with h2o
#install packages needed
install.packages("h2o")
library(h2o)
install.packages("mlbench")
library(h2o)

#factoring the dataset variables
Matthews_Data_h2o[,1] <- as.factor(Matthews_Data_h2o[,1])
Matthews_Data_h2o[,2] <- as.factor(Matthews_Data_h2o[,2])
```

```
Matthews_Data_h2o[,5] <- as.factor(Matthews_Data_h2o[,5])

#initiate h2o
h2o.init()
#convert to h2o frame
Matthews_Data_h2o <- as.h2o(Matthews_Data)
h2o.describe(Matthews_Data_h2o)
#factoring the dataset variables
Matthews_Data_h2o[,1] <- as.factor(Matthews_Data_h2o[,1])
Matthews_Data_h2o[,2] <- as.factor(Matthews_Data_h2o[,2])
Matthews_Data_h2o[,5] <- as.factor(Matthews_Data_h2o[,5])
#create neural network using h2o
h2o_NN <- h2o.deeplearning(x=1:4,
y=5,training_frame=Matthews_Data_h2o,nfolds=5,standardize =
TRUE,activation="Rectifier",hidden=c(200,200),seed=35,variable_importances = T)
#show performance of the neural network
h2o.performance(h2o_NN)
#show variable importances
h2o.varimp(h2o_NN)
#performance on test data
test_performance <- h2o.performance(h2o_NN,Matthews_Data_h2o)
test_performance
#using the model to predict the classes
h2o_predictions <- h2o.predict(object=h2o_NN, newdata=Matthews_Data_h2o)
h2o_predictions <- as.data.frame(h2o_predictions)
predictions <- data.frame(actual =
as.vector(Matthews_Data_h2o$response_variable),as.data.frame(h2o.predict(object=h2o_NN,newd
ata=Matthews_Data_h2o)))
predictions$accurate <- ifelse(predictions$actual == predictions$predict,"yes","no")
predictions %>%
  group_by(actual,predict) %>%
  summarise(n=n())
#shutting h2o down
h2o.shutdown()
#create classification tree
set.seed(50)
#shuffle index
shuffled_index <- sample(1:nrow(Matthews_Data))
Matthews_Data <- Matthews_Data[shuffled_index,]
#create training and test set
create_train_test <- function(data, size = 0.8,train=TRUE) {
  n_row = nrow(data)
  total_row = size*n_row
  train_sample <-1:total_row
  if(train == TRUE){
    return(data[train_sample,])
  }else{
    return(data[-train_sample,])
  }
}
train_set <- create_train_test(Matthews_Data,0.8,train=TRUE)
test_set <- create_train_test(Matthews_Data,0.2,train=FALSE)

#install rpart package
install.packages("rpart")
install.packages("rpart.plot")
library(rpart.plot)
library(rpart)
#create first tree and plot
tree_1 <- rpart(response_variable ~ side_of_rink,train_set,method='class',cp=-1)
rpart.plot(tree_1,extra=106)
#prediction for tree 1
prediction <- predict(tree_1,test_set,type='class')
#create second tree and plot
tree_2 <- rpart(response_variable ~ side_of_rink + type_of_shot_all,train_set,method='class')
rpart.plot(tree_2,extra=106)
#prediction for tree 2
prediction <- predict(tree_2,test_set,type='class')
#create third tree and plot
tree_3 <- rpart(response_variable ~ side_of_rink + type_of_shot_all +
distance,train_set,method='class')
rpart.plot(tree_3,extra=106)
prediction <- predict(tree_3,test_set,type='class')
#create fourth tree and plot
tree_4 <- rpart(response_variable ~ side_of_rink + type_of_shot_all + distance +
shot_angle,train_set,method='class')
rpart.plot(tree_4,extra=106)
prediction <- predict(tree_3,test_set,type='class')
#confusion matrix for predictions
matrix <- table(test_set$response_variable,prediction)
matrix
#compute accuracy of test
accuracy <- sum(diag(matrix))/sum(matrix)
accuracy
```