

Exploring and Analyzing Hockey Scoring Sequences

DAR Assignment 2 - Hockey (Fall 2023)

Amy Enyenihi

2023-09-15

Assignment 2 Notebook Overview

This notebook is broken into two main parts:

- * Part 1 is a guide to pulling the Hockey Analysis repository from the RPI github
- * Part 2 is an introduction to the RPI Hockey motion capture data, including simple modelling
- * Part 3 ask you to do your own modeling
- * Parts 4 and 5 give you additional assignments.

This R Notebook and its related R scripts provide a very basic introduction to the RPI Hockey Motion Capture dataset. This data will be used by the **Hockey Analytics** group during DAR F23.

The RPI github repository for all the code required for this notebook may be found at:

- https://github.rpi.edu/DataINCITE/Hockey_Fall_2023

TBD: The `Hockey_Fall_2023` github may also contain topical notebooks which provide tutorials regarding a number of different research-worthy starting points. Feel free to examine these notebooks.

PART 1: CLONING A NOTEBOOK AND UPDATING THE REPOSITORY

In this assignment we're asking you to...

- Clone the `Hockey_Fall_2023` github repository...
- Create a personal branch using git, and...
- Copy this notebook (as instructed) and customize it; this includes adding code and answering questions.
- Add your new notebook to the project repo on the RPI github.

The instructions below explain how to accomplish all of this.

Cloning an RPI github repository

The recommended procedure for cloning and using this repository is as follows:

- Access the RPI network via VPN
 - See <https://itssc.rpi.edu/hc/en-us/articles/360008783172-VPN-Connection-and-Installation> for information
- Access RStudio Server on the IDEA Cluster at <http://lp01.idea.rpi.edu/rstudio-ose/>
 - You must be on the RPI VPN!!
- Access the Linux shell on the IDEA Cluster by clicking the **Terminal** tab of RStudio Server (lower left panel).
 - You now see the Linux shell on the IDEA Cluster
 - `cd` (change directory) to enter your home directory using: `cd ~`
 - Type `pwd` to confirm

- NOTE: Advanced users may use `ssh` to directly access the Linux shell from a macOS or Linux command line
- Type `git clone https://github.rpi.edu/DataINCITE/Hockey_Fall_2023` from within your home directory
 - This will create a new directory `Hockey_Fall_2023`
- In the Linux shell, `cd` to `Hockey_Fall_2023/StudentNotebooks/Assignment01`
 - Type `ls -al` to list the current contents
 - Don't be surprised if you see many files!
- In the Linux shell, type `git checkout -b dar-yourrrcs` where `yourrrcs` is your RCS id
 - For example, if your RCS is `erickj4`, your new branch should be `dar-erickj4`
 - It is *critical* that you include your RCS id in your branch id
- Now in the RStudio Server UI, navigate to the `Hockey_Fall_2023/StudentNotebooks/Assignment01` directory via the **Files** panel (lower right panel)
 - Under the **More** menu, set this to be your R working directory
 - Setting the correct working directory is essential for interactive R use!

REQUIRED FOR ASSIGNMENT 2

1. In RStudio, make a **copy** of `darf23-assignment2-template.Rmd` file using a *new, original, descriptive* filename that **includes your RCS ID!**
 - Open `darf23-assignment2-template.Rmd`
 - **Save As...** using a new filename that includes your RCS ID
 - Example filename for user `erickj4`: `erickj4-assignment1-f23.Rmd`
 - POINTS OFF IF:
 - You don't create a new filename!
 - You don't include your RCS ID!
 - You include `template` in your new filename!
2. Edit your new notebook using RStudio and save
 - Change the `title:` and `subtitle:` headers (at the top of the file)
 - Change the `author:`
 - Don't bother changing the `date:`; it should update automatically...
 - **Save** your changes
3. Use the RStudio **Knit** command to create an HTML file; repeat as necessary
 - Use the down arrow next to the word **Knit** and select **Knit to HTML**
 - You may also knit to PDF...
4. In the Linux terminal, use `git add` to add each new file you want to add to the repository
 - Type: `git add yourfilename.Rmd`
 - Type: `git add yourfilename.html` (created when you knitted)
 - Add your PDF if you also created one...
5. Continue making changes to your personal notebook
 - Add code where specified
 - Answer questions were indicated.
6. When you're ready, in Linux commit your changes:
 - Type: `git commit -m "some comment"` where "some comment" is a useful comment describing your changes
 - This commits your changes to your local repo, and sets the stage for your next operation.
7. Finally, push your commits to the RPI github repo
 - Type: `git push origin dar-yourrrcs` (where `dar-yourrrcs` is the branch you've been working in)
 - Your changes are now safely on the RPI github.
8. **REQUIRED:** On the RPI github, submit a pull request.
 - In a web browser, navigate to `https://github.rpi.edu/DataINCITE/Hockey_Fall_2023`
 - In the branch selector drop-down (by default says **master**), select your branch
 - **Submit a pull request for your branch**

- One of the DAR instructors will merge your branch, and your new files will be added to the master branch of the repo.

Please also see these handy github “cheatsheets”:

- <https://education.github.com/git-cheat-sheet-education.pdf>

PART 2: HOCKEY DATA PREPARATION AND MODELLING

NOTE: The code chunk `setup_1` should be run interactively (i.e. not by “knitting”) before attempting to knit this notebook!

Setup: Define functions

The major functions of this notebook are contained in the file `AnalysisCodeFunc.R`. This code chunk *sources* (imports) a helper script that defines various functions used through this notebook for data processing and analysis.

```
# All user-defined functions are contained in the following helper script file.
source("../..//AnalysisCodeFunc.R")
```

Setting program parameters

This section sets the dimensions of the data structures used in this notebook, based on the captured video.

```
# Size of rink image and of all plots
xsize <- 2000
ysize <- 850

# FPS of the video
fps <- 29.97

# Coordinates to the goal pipes
pipes_x <- 1890
lpipe_y <- 395
rpipe_y <- 455
```

Data preparation for predictive modelling

Based on our settings, this section reads in the captured image data.

The code is highly dependent upon the following directory structure:

- The file path determined by the `filepath` variable contains folders named with the game number, followed by ‘p’, followed by the period number
- Each `period` folder contains a folder named `Sequences`.
- Each `Sequences` folder contains `sequence_folders` that contain all the relevant sequence data.

```
# Set filepaths and read the rink
# Results suppressed because this can be messy...

# This file path should contain the hockey rink images and all the sequences
filepath <- '../..//FinalGoalShots/'

# See above for explanation of file path syntax
games <- c(24, 27, 33, 34)
# Only take the first and third periods. These are when the opposing team shoots on our goal. Our shots
periods <- map(games, ~ str_c(., 'p', c(1, 3))) %>% unlist
```

```

# Get the 'Sequences' folder for every period
period_folders <- map(periods, ~ {
  str_c(filepath, ., '/Sequences')
})

# Get every folder inside each 'Sequences' folder
sequence_folders <- period_folders %>%
  map(~ str_c(., '/', list.files(.))) %>%
  unlist

# Read the rink images and format them to a raster used for graphing
rink_raster <- makeRaster(filepath, 'Rink_Template.jpeg')
half_rink_raster <- makeRaster(filepath, 'Half_Rink_Template.jpeg')

# As every folder is run through the `combinePasses` function, the info.csv file in each sequence folder
info <- matrix(0, nrow = 0, ncol = 4) %>%
  data.frame %>%
  set_names(c('possessionFrame', 'shotFrame', 'outcome', 'rightHanded'))

# Read in all the sequences
# NOTE: This step takes a long time (minutes)
sequences = sequence_folders %>% map(combinePasses)

# Change outcomes to more verbose names
info$outcome %<>% fct_recode(Goal = 'G', Save = 'GB', 'Defender Block' = 'DB', Miss = 'M')

```

Shot statistics retrieval

This section constructs the dataframe used to predict if shots are successful.

```

# Get stats for the shot in every sequence
shots_stats.df <- seq_along(sequences) %>%
  map_dfr(goalShotStats) %>%
  # Some models can't use logical data
  mutate_if(is.logical, as.factor)

```

We first combine the shots data with the outcomes vector;

```

# Split data into training and validation sets
outcomes.goal <- (info$outcome == 'Goal') %>% as.numeric %>% as.factor

# Append to shots_stats.df
shots_stats_goal.df <- cbind(shots_stats.df, outcomes.goal)

# Save this dataframe on the file system in case we want to simply load it later (to save time)
saveRDS(shots_stats_goal.df, "shots_stats_goal.df.Rds")

```

Predictive modelling

Prepare 80/20 Train/Test Sets

Now let's do some basic classification analysis on the `shots_stats_goal.df` dataset!

We'll use `tidymodels`, part of the **tidyverse**, to split the data into an 80% train/20% test split.

```

#Create training set
set.seed(100)

# Type ?initial_split , ?training , or ?testing in the R console to see how these work!
hockey_split <- initial_split(shots_stats_goal.df, prop = 0.8)
hockeyTrain <- training(hockey_split)
hockeyTest <- testing(hockey_split)

# Check how many observations for each split we have
nrow(hockeyTrain)

## [1] 84

nrow(hockeyTest)

## [1] 21

# How many features are there
ncol(hockeyTrain)

## [1] 12

```

Model One: Logistic regression and Balanced Accuracy (full dataset)

We would like to create a model to predict the `outcomes.goal` variable which indicates whether a play resulted in a goal or not.

Our first model is trained using *logistic regression*. We show the most important coefficients in the following table.

```

# Create the model
LR <- glm(outcomes.goal ~ . , family = "binomial", data=hockeyTrain)

## Warning: glm.fit: algorithm did not converge
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

# Review our model
summary(LR)

##
## Call:
## glm(formula = outcomes.goal ~ . , family = "binomial", data = hockeyTrain)
##
## Deviance Residuals:
##      Min       1Q   Median       3Q      Max
## -1.053e-04 -2.100e-08 -2.100e-08 -2.100e-08  1.170e-04
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)   -1.752e+02  8.339e+04  -0.002   0.998
## puckDist      -8.133e-01  2.606e+02  -0.003   0.998
## puckAngle      7.875e-01  5.190e+02   0.002   0.999
## puckSpeed      2.191e+00  1.506e+03   0.001   0.999
## shooterSpeed   4.660e+00  2.822e+03   0.002   0.999
## goalieDist    -8.823e-02  6.834e+02   0.000   1.000
## goalieAngle    1.035e+00  4.787e+02   0.002   0.998
## posTime       -4.383e+00  1.620e+03  -0.003   0.998

```

```
## sameDefenders      5.918e+01  7.826e+04  0.001  0.999
## oppDefenders       9.886e+01  4.682e+04  0.002  0.998
## goalieScreenedTRUE -1.537e+02  6.191e+04 -0.002  0.998
## rightHanded        -1.418e+02  4.923e+04 -0.003  0.998
##
## (Dispersion parameter for binomial family taken to be 1)
##
## Null deviance: 4.3230e+01  on 83  degrees of freedom
## Residual deviance: 3.3388e-08  on 72  degrees of freedom
## AIC: 24
##
## Number of Fisher Scoring iterations: 25

# Get the p-values of the coefficients and show the ones that are less than 0.05 (if any!)
coefs <- coef(summary(LR))

kable(coefs[coefs[,4] < 0.05,c(1,4)])
```

Estimate	Pr(> z)
----------	----------

Computing *Balanced Accuracy* on the test set

Now we generate the confusion matrix and calculate BA from its cells based on e.g.

<https://statisticaloddsandends.wordpress.com/2020/01/23/what-is-balanced-accuracy/>

where...

- True positive: `cm[1,1]`
- True negative: `cm[2,2]`
- False negative: `cm[1,2]`
- False positive: `cm[2,1]`

```
testRes <- predict(LR,hockeyTest, type='response')

# Our Confusion Matrix
cm <- as.matrix(table(Actual = hockeyTest$outcomes.goal, Predicted = testRes>0.5))

# Display the table (pretty!)
kable(cm)
```

	FALSE	TRUE
0	18	0
1	1	2

```
# Balanced Accuracy
balancedAccuracyTest1<-(cm[1,1]/(cm[1,1]+cm[1,2]) + cm[2,2]/(cm[2,1]+cm[2,2]))/2

balancedAccuracyTest1

## [1] 0.8333333
```

Model Two: Logistic regression and Balanced Accuracy (dropping a variable)

Now we repeat the analysis dropping the `goalieScreened` variable which indicates whether the goalie's line of sight was blocked. How does the accuracy of the prediction change?

```
# Model 2

LR <- glm(outcomes.goal ~ .- `goalieScreened`, family = "binomial", data=hockeyTrain)

## Warning: glm.fit: algorithm did not converge
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

# Review our model
summary(LR)

##
## Call:
## glm(formula = outcomes.goal ~ . - goalieScreened, family = "binomial",
##      data = hockeyTrain)
##
## Deviance Residuals:
##      Min       1Q   Median       3Q      Max
## -1.204e-04 -2.100e-08 -2.100e-08 -2.100e-08  1.365e-04
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -2.375e+01  1.118e+05  0.000   1.000
## puckDist    -8.022e-01  2.225e+02 -0.004   0.997
## puckAngle    7.879e-01  6.251e+02  0.001   0.999
## puckSpeed    2.159e+00  1.105e+03  0.002   0.998
## shooterSpeed  4.648e+00  3.361e+03  0.001   0.999
## goalieDist   -7.816e-02  6.348e+02  0.000   1.000
## goalieAngle  1.011e+00  6.366e+02  0.002   0.999
## posTime     -4.328e+00  1.324e+03 -0.003   0.997
## sameDefenders 5.885e+01  6.769e+04  0.001   0.999
## oppDefenders -5.332e+01  3.885e+04 -0.001   0.999
## rightHanded  -1.406e+02  6.056e+04 -0.002   0.998
##
## (Dispersion parameter for binomial family taken to be 1)
##
##      Null deviance: 4.3230e+01  on 83  degrees of freedom
## Residual deviance: 4.3943e-08  on 73  degrees of freedom
## AIC: 22
##
## Number of Fisher Scoring iterations: 25

# Get the p-values of the coefficients and show the ones that are less than 0.05 (if any!)
coefs<-coef(summary(LR))
kable(coefs[coefs[,4]<0.05,c(1,4)])
```

Estimate	Pr(> z)
----------	----------

Computing *Balanced Accuracy* on the test set

Can we still do a good job without the `goalieScreened` variable?

```
testRes<-predict(LR,hockeyTest, type='response')

cm <- as.matrix(table(Actual = hockeyTest$outcomes.goal, Predicted = testRes>0.5))

# Display the table (pretty!)
kable(cm)
```

	FALSE	TRUE
0	17	1
1	1	2

```
# Balanced Accuracy
balancedAccuracyTest2<-(cm[1,1]/(cm[1,1]+cm[1,2]) + cm[2,2]/(cm[2,1]+cm[2,2]))/2

balancedAccuracyTest2

## [1] 0.8055556
```

How much did we change our balanced accuracy?

By dropping the one feature, how did we change our model?

```
if (balancedAccuracyTest2 < balancedAccuracyTest1) {
  print(paste0("Balanced accuracy decreased: Model2 = ",balancedAccuracyTest2, " < Model1 = ", balancedAccuracyTest1))
} else if (balancedAccuracyTest2 > balancedAccuracyTest1) {
  print(paste0("Balanced accuracy increased: Model2 = ",balancedAccuracyTest2, " > Model1 = ", balancedAccuracyTest1))
} else {print("No change...")}

## [1] "Balanced accuracy decreased: Model2 = 0.805555555555556 < Model1 = 0.833333333333333"
```

PART 3: Create and evaluate your own model

Create your own classification model to predict goals using the training set, and evaluate the balanced accuracy on the test set. Use the classification method of your choice.

Introduce yourself and the coordinate with your teammates on the Webex chat for the class called DAR Hockey Analytics F23. Make sure that each teammate creates a unique model (you are welcome to run extras if you like).

Describe your modeling method.

```
install.packages("randomForest")

## Installing package into '/home/enyena/R/x86_64-pc-linux-gnu-library/4.1'
## (as 'lib' is unspecified)

## Warning in install.packages("randomForest"): installation of package
## 'randomForest' had non-zero exit status

install.packages("caret")

## Installing package into '/home/enyena/R/x86_64-pc-linux-gnu-library/4.1'
## (as 'lib' is unspecified)

## Warning in install.packages("caret"): installation of package 'caret' had
## non-zero exit status
```



```

library(randomForest)

## randomForest 4.7-1.1
## Type rfNews() to see new features/changes/bug fixes.
##
## Attaching package: 'randomForest'
## The following object is masked from 'package:dplyr':
##
##      combine
## The following object is masked from 'package:ggplot2':
##
##      margin
library(caret)

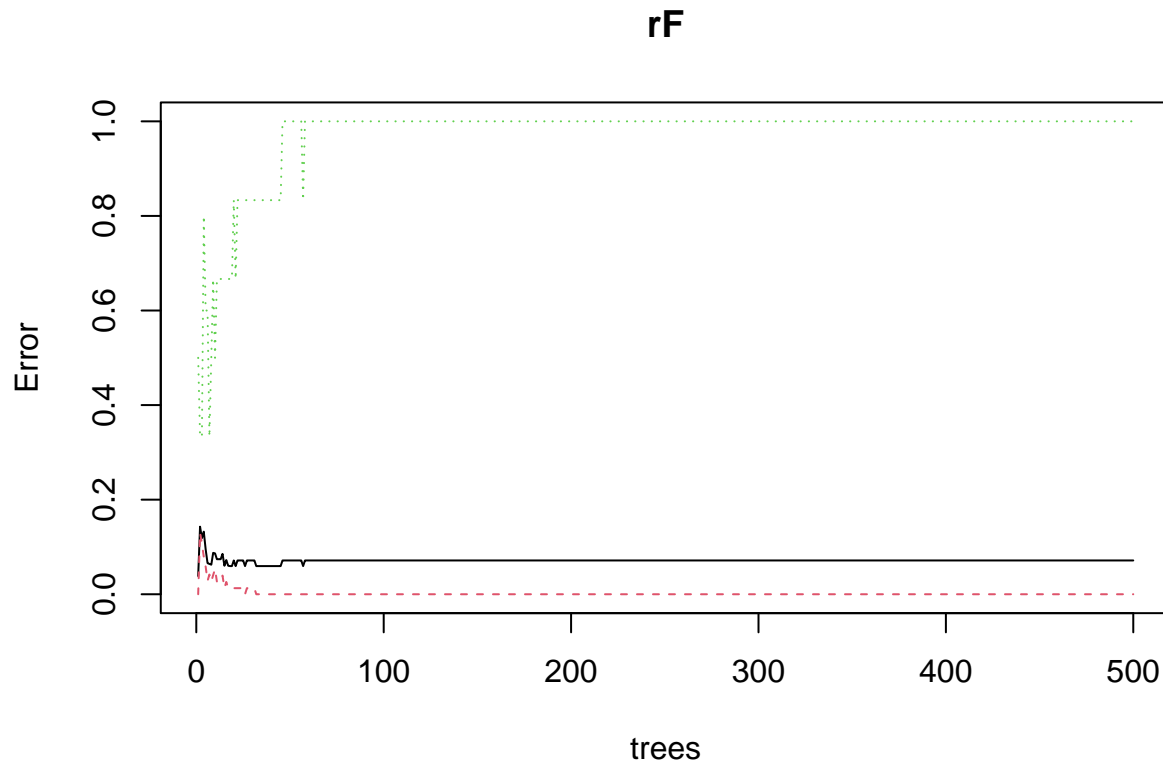
## Loading required package: lattice
##
## Attaching package: 'caret'
## The following objects are masked from 'package:yardstick':
##
##      precision, recall, sensitivity, specificity
## The following object is masked from 'package:purrr':
##
##      lift
#using random forest classification
rF <- randomForest(outcomes.goal ~ ., data = hockeyTrain)

print(rF) #results and confusion matrix of the random forest

##
## Call:
## randomForest(formula = outcomes.goal ~ ., data = hockeyTrain)
##              Type of random forest: classification
##              Number of trees: 500
## No. of variables tried at each split: 3
##
##              OOB estimate of  error rate: 7.14%
## Confusion matrix:
##      0 1 class.error
## 0 78 0           0
## 1  6 0           1

plot(rF) #visualizing the error rate. the error rate stabilizes with more trees

```



```
pred <- predict(rF, newdata = hockeyTest)
pred_matrix <- confusionMatrix(table(pred, hockeyTest$outcomes.goal)) #the prediction to compute the
pred_matrix
```

```
## Confusion Matrix and Statistics
##
##
## pred  0  1
##      0 18  3
##      1  0  0
##
##               Accuracy : 0.8571
##               95% CI   : (0.6366, 0.9695)
##      No Information Rate : 0.8571
##      P-Value [Acc > NIR] : 0.6477
##
##               Kappa : 0
##
##  Mcnemar's Test P-Value : 0.2482
##
##               Sensitivity : 1.0000
##               Specificity : 0.0000
##      Pos Pred Value : 0.8571
##      Neg Pred Value :      NaN
##      Prevalence : 0.8571
##      Detection Rate : 0.8571
##      Detection Prevalence : 1.0000
##      Balanced Accuracy : 0.5000
##
##      'Positive' Class : 0
```

```
##
```

Determine the balanced accuracy and compare with the above approaches.

```
rF_accuracy <- pred_matrix$overall[1]
```

```
#comparing balanced model 1 to the balanced accuracy of random forest
```

```
if (rF_accuracy < balancedAccuracyTest1) {  
  print(paste0("Balanced accuracy decreased: Random Forest = ", rF_accuracy, " < Model1 = ", balancedAccuracyTest1))  
} else if (rF_accuracy > balancedAccuracyTest1) {  
  print(paste0("Balanced accuracy increased: Random Forest = ", rF_accuracy, " > Model1 = ", balancedAccuracyTest1))  
} else {print("No change...")}
```

```
## [1] "Balanced accuracy increased: Random Forest = 0.857142857142857 > Model1 = 0.833333333333333"
```

```
#comparing balanced model 2 to the balanced accuracy of random forest
```

```
if (rF_accuracy < balancedAccuracyTest2) {  
  print(paste0("Balanced accuracy decreased: Random Forest = ", rF_accuracy, " < Model2 = ", balancedAccuracyTest2))  
} else if (rF_accuracy > balancedAccuracyTest2) {  
  print(paste0("Balanced accuracy increased: Random Forest = ", rF_accuracy, " > Model2 = ", balancedAccuracyTest2))  
} else {print("No change...")}
```

```
## [1] "Balanced accuracy increased: Random Forest = 0.857142857142857 > Model2 = 0.805555555555556"
```

Discuss how your results and how they compare with others.

Part 4: Feature selection

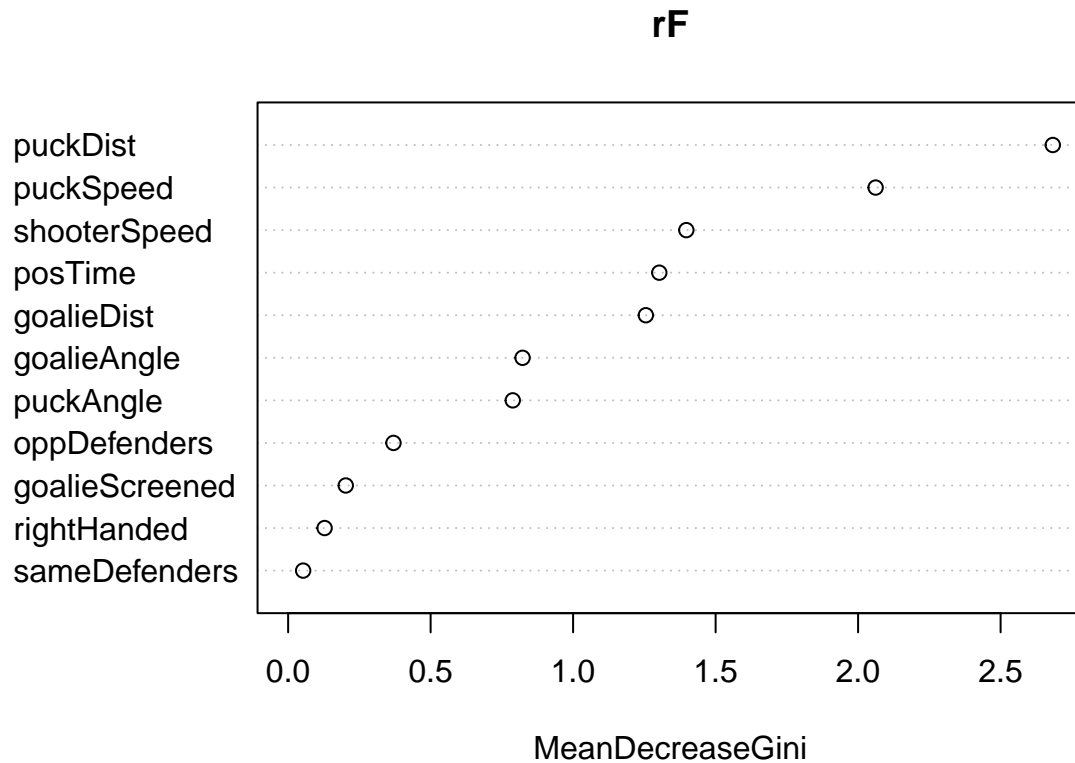
One of the aims of the hockey project is to determine which features are most predictive of successful goals. Perform a creative analysis of your choice to examine one or more features and how they may be related to successful or unsuccessful goals. Coordinate with your teammates to make sure you focus on different aspects of the project.

Describe your analysis.

```
importance(rF) #listing all the variables with their relative importance
```

```
##           MeanDecreaseGini  
## puckDist           2.68317149  
## puckAngle          0.78817912  
## puckSpeed          2.06149384  
## shooterSpeed       1.39731270  
## goalieDist         1.25520566  
## goalieAngle        0.82262281  
## posTime            1.30238119  
## sameDefenders      0.05267905  
## oppDefenders       0.36961250  
## goalieScreened     0.20223037  
## rightHanded        0.12799699
```

```
varImpPlot(rF) #plotting the list above
```



Discuss your results and what they mean.

#The graph produced plots the variables in increasing importance. The puck distance was deemed the most

Part 5: Prepare group presentation

Prepare a (at most) *three-slide* presentation of your classification results and creative analysis. Create a joint presentation with your teammates using the Google Slides template available here: <https://bit.ly/45twUP> (copy the template and customize with your content)

You should include a slide with a table that summarizes all of your and your teammates' balanced accuracy results.

Be prepared to present your results on 13 Sep 2023 in class!