Predicting Games Played

Kyle Joecken

Injuries: The bane of any promising fantasy season.

Question

Can we use games played data from the previous few National Hockey League (NHL) seasons to predict how many games a particular skater (*i.e.*, non-goalie) will play in an upcoming season?

In particular, the hope is that games played for various players in their more recently played seasons will help to predict how many games they'll play in future seasons. This is quite difficult, as the data are exceedingly noisy due to the largely random nature of injuries in a fast-paced, full-contact sport. Surely there is some small signal for which we can search.

Data

First, we load the data from our local (scraped) database of season-wide NHL data. It is stored in a data frame called `skaterstats`.

The data was scraped from the NHL.com statistics pages by a Python script using the Scrapy framework; the relevant script and a `CODEBOOK.md` file can be found at this link.

Features

We'll use statistics from the previous two seasons to try and predict games played. To train the model, we will take all our data from 2012 and 2013 and try to find the most applicable variables to modeling games played in 2014.

First, we need to finish wrangling the data into a form we can put into various models. We need to drop all data pertaining to seasons before 2012, drop incomplete or irrelevant variables, then reshape the data so that there is only one observation per player.

```
skaterstats <- subset(skaterstats, season > 2011)     # shed old data
skaterstats <- skaterstats[, -c(3:5, 38:41)]          # drop team/SO variables
widestats <- reshape(skaterstats, timevar = "season",
                     idvar = "nhl_num", direction = "wide")
widestats <- widestats[, -(81:118)]               # drop non-GP 2014 vars
```

We'd also like to add a simplified age to the computation (`2014 - birthyear`).

```
library(lubridate)
skaters <- skaters[, c(3,5)]                    # grab nhl_num, birthday
skaters$age <- 2014 - year(skaters$birthday)    # compute "age"
skaters <- skaters[, -2]                        # drop birthday
data <- merge(skaters, widestats)[, -1]    # merge along/drop nhl_num
```

Finally, we reduce our collection of 1175 NHL skaters (for now) down to the set that played in all three of the last few seasons.

```
data <- data[!is.na(data$games_played.2012) &
                       !is.na(data$games_played.2013) &
                       !is.na(data$games_played.2014), ]
```

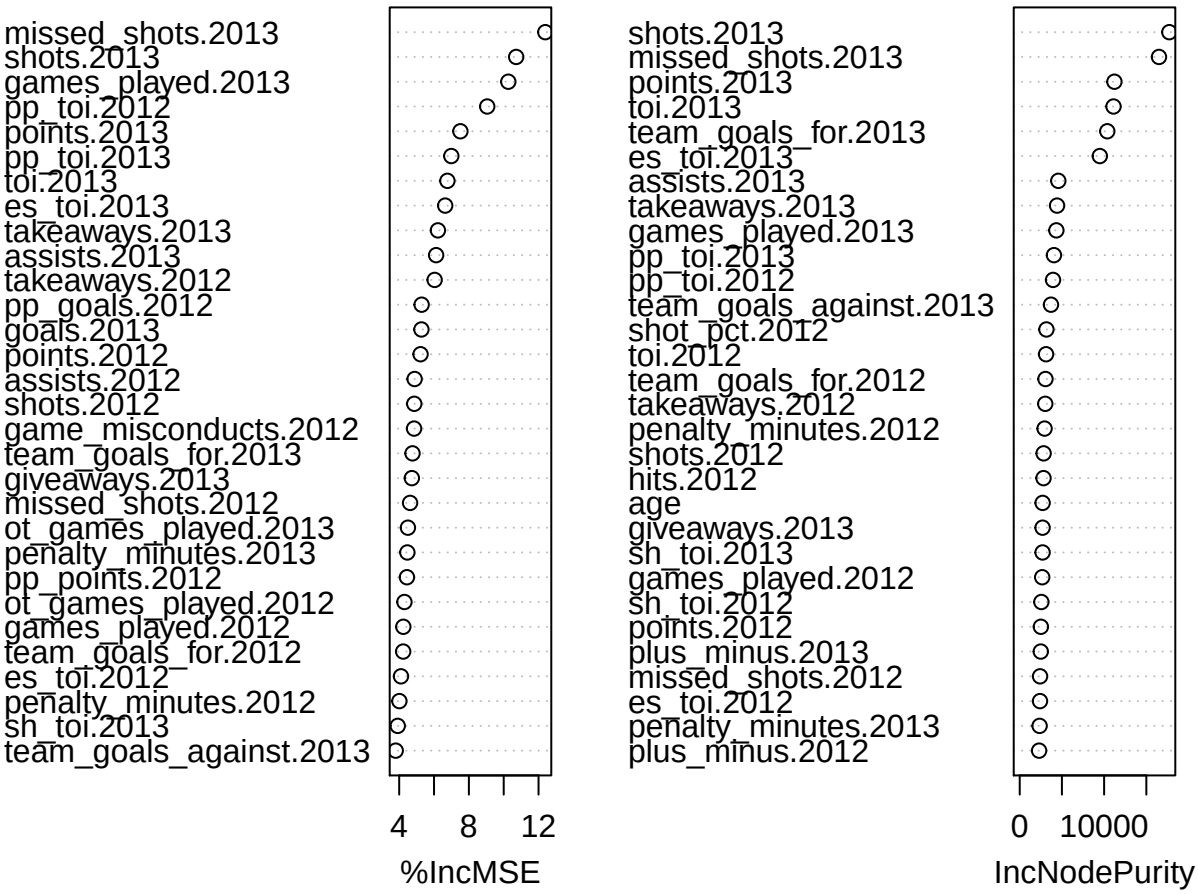
We first build a naive random forest model built on all variables, then ask it which variables it found important. We will try and pare down predictors intelligently from the output.

We define a training set on 60% of the data, with a two-tier training set on which to cross-validate.

```
library(caret)
library(randomForest)
set.seed(485)
inTest <- createDataPartition(data$games_played.2014, p = .2, list = FALSE)
testing  <- data[inTest, ]
building <- data[-inTest, ]
inTrain <- createDataPartition(building$games_played.2014, p = .75, list = FALSE)
training <- building[inTrain, ]
checking <- building[-inTrain, ]
testMod <- randomForest(games_played.2014 ~ ., data = training, importance = TRUE)
print(testMod)
```

```
##
## Call:
##  randomForest(formula = games_played.2014 ~ ., data = training,      importance = TRUE)
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 26
##
##           Mean of squared residuals: 341.4
##                     % Var explained: 37.21
```

```
varImpPlot(testMod)
```



Through several iterations of modelling these data, shots and missed shots are at the top of every list. Various iterations of situational time on ice, giveaways, takeaways, and points also frequently appear. As such, we'll tentatively select these predictors and throw in our pet variable age for another look. We will also restrict ourselves to 2013 to increase generalizability of our model.

We reframe the data to pare down the predictors further and see where we are.

```
skaterstats <- subset(skaterstats, season > 2012)
skaterstats <- skaterstats[, c(1:3, 6, 15, 30:32, 38:40)] # pare down predictors
widestats <- reshape(skaterstats, timevar = "season",
                     idvar = "nhl_num", direction = "wide")
widestats <- widestats[, -(12:19)]
data <- merge(skaters, widestats)[, -1]
data <- data[!is.na(data$games_played.2013) & !is.na(data$games_played.2014), ]
```

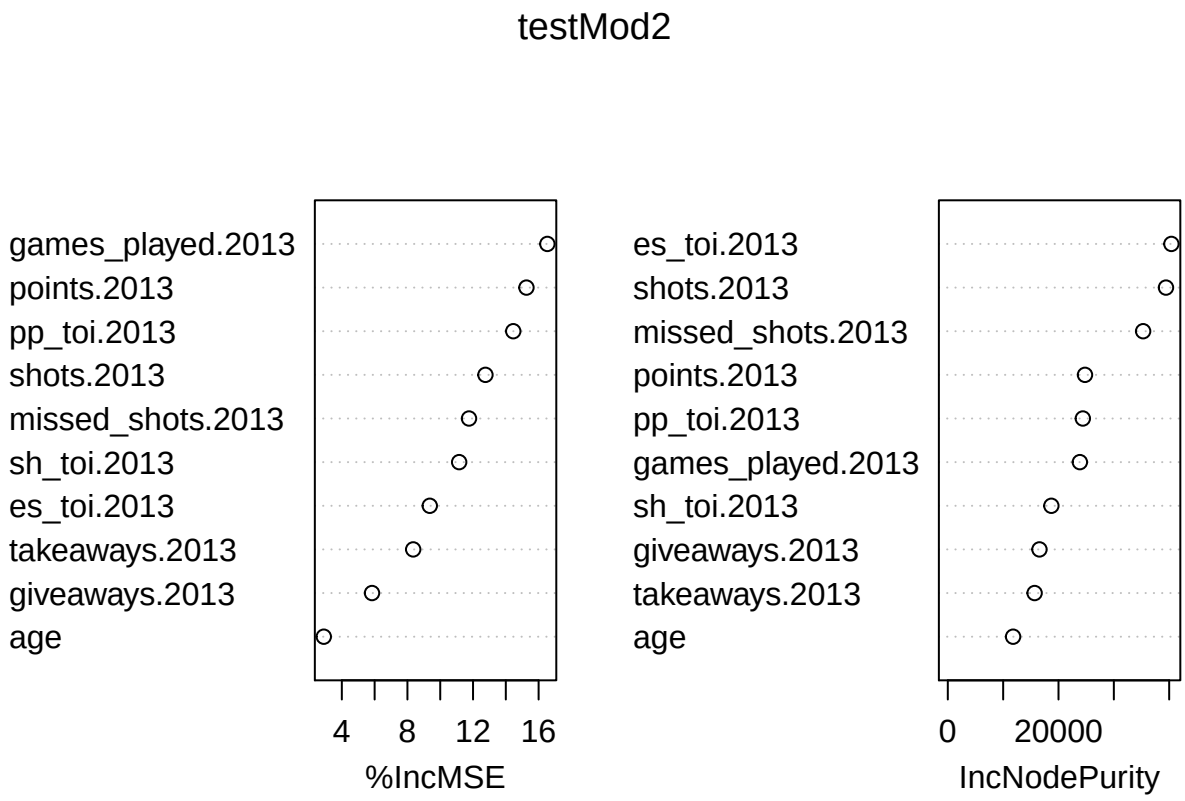
We now build a random forest model as before and have a look at importance.

```
set.seed(2857)
inTest <- createDataPartition(data$games_played.2014, p = .2, list = FALSE)
testing  <- data[inTest, ]
```

```
building <- data[-inTest, ]
inTrain <- createDataPartition(building$games_played.2014, p = .75, list = FALSE)
training <- building[inTrain, ]
checking <- building[-inTrain, ]
testMod2 <- randomForest(games_played.2014 ~ ., data = training, importance = TRUE)
print(testMod2)
```

```
##
## Call:
##  randomForest(formula = games_played.2014 ~ ., data = training,      importance = TRUE)
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 3
##
##           Mean of squared residuals: 389.8
##                     % Var explained: 38.08
```

```
varImpPlot(testMod2)
```



And now one more time without age, to see if it is worth keeping.

```
data2 <- widestats[!is.na(widestats$games_played.2013) &
                         !is.na(widestats$games_played.2014), -1]
inTest2 <- createDataPartition(data2$games_played.2014, p = .2, list = FALSE)
testing2  <- data2[inTest2, ]
building2 <- data2[-inTest2, ]
inTrain2 <- createDataPartition(building2$games_played.2014, p = .75, list = FALSE)
```

```
training2 <- building2[inTrain2, ]
checking2 <- building2[-inTrain2, ]
testMod3 <- randomForest(games_played.2014 ~ ., data = training2, importance = TRUE)
print(testMod3)
```

```
##
## Call:
##  randomForest(formula = games_played.2014 ~ ., data = training2,      importance = TRUE)
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 3
##
##           Mean of squared residuals: 410.1
##                     % Var explained: 33
```

```
varImpPlot(testMod3)
```