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`. We also import a few purpose-built functions to help wrangle the data and build our model.

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.

Feature Selection

We'll use statistics from the previous two seasons to try and predict games played. We will intelligently select features from both random forest and boosting models built on all available predictors and consider various importance statistics over a few iterations of the models. As we want our model to be as generalizable as possible, we will build the random forest model using only the previous season's data, while the boosting model will use the previous two seasons (as incomplete observations can still be fed to a boosting model).

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. We'll then build a naive random forest model based on all predictors and look for factors that are consistently important.

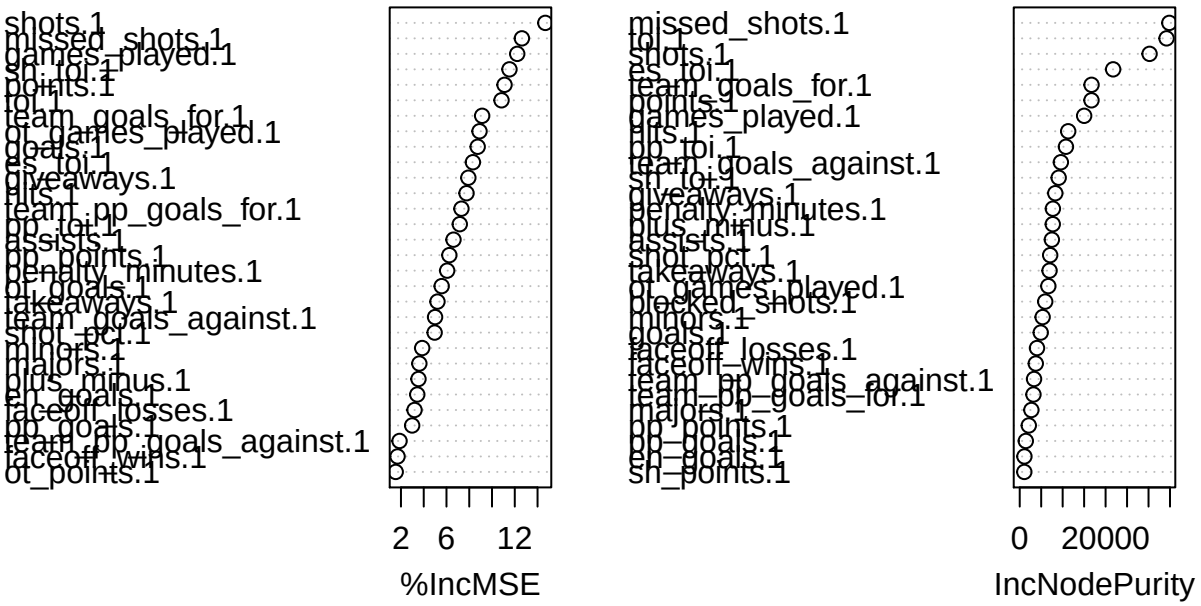
Note that based on the way the `nhlShape()` function shapes the data, the number after the predictor tells how many seasons prior to the predicted season the data describes (*e.g.*, `games_played.1` is the number of games played in the previous season).

```
skaterstats <- skaterstats[, -c(3:5, 38:41)]          # drop team/SO variables
gbmdata <- nhlShape(2012, 2013, outcome = 3)
rfdata <- nhlShape(2013, 2013, outcome = 3)
```

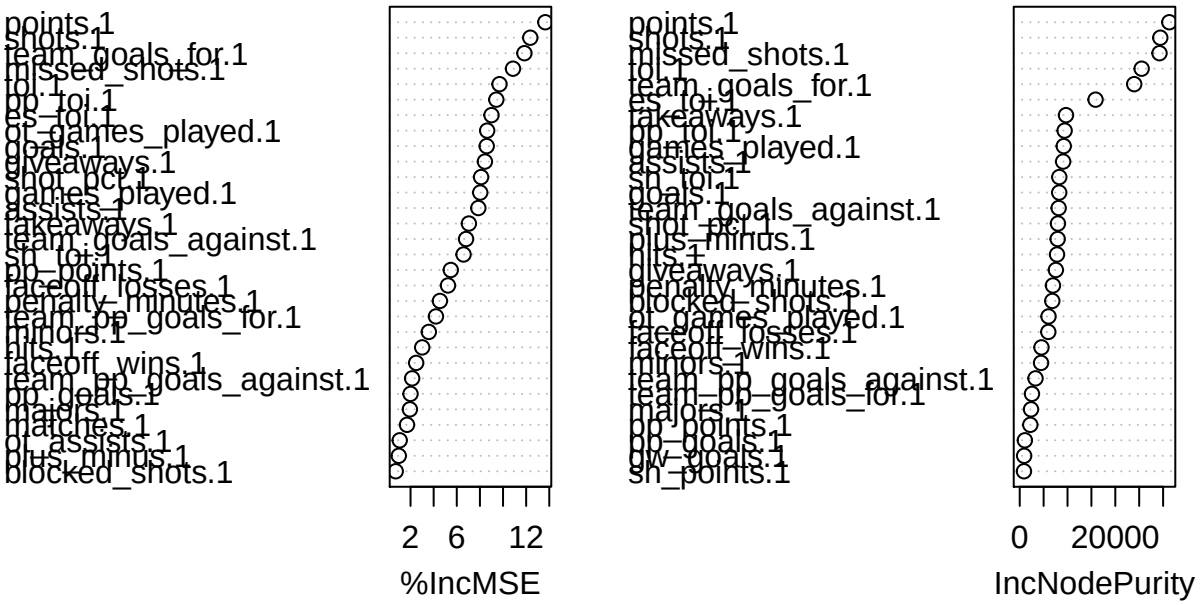
First, we build random forest models from several seeds.

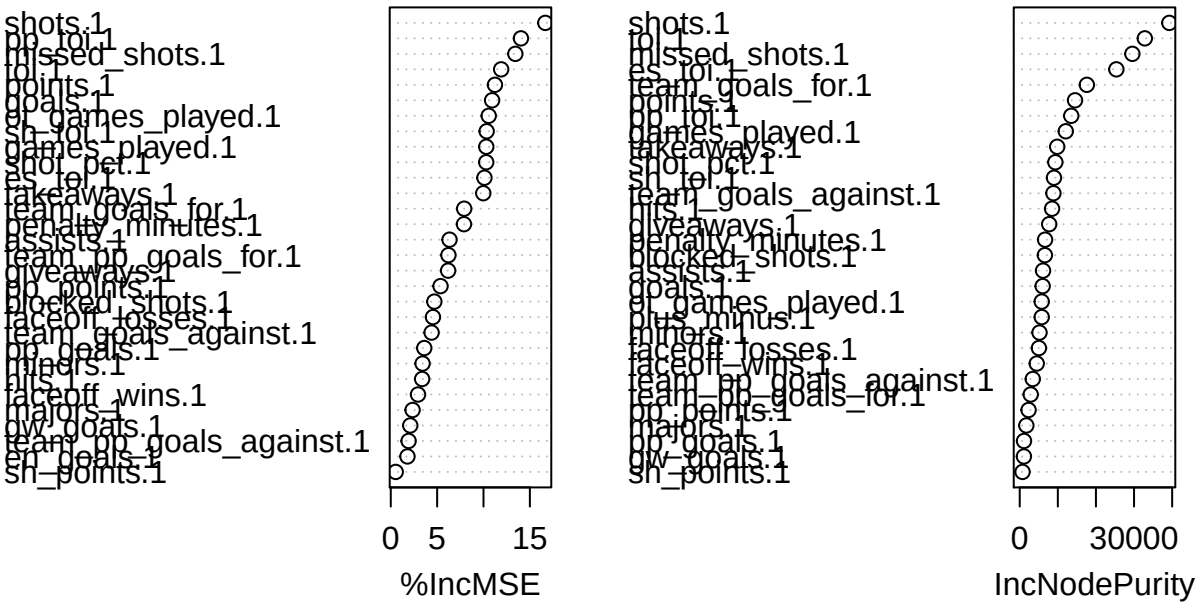
```
gprfmod1 <- nhlBuild(data = rfdata, perc = 0.7, seed = 9112)
gprfmod2 <- nhlBuild(data = rfdata, perc = 0.7, seed = 2857)
gprfmod3 <- nhlBuild(data = rfdata, perc = 0.7, seed = 31415)
gprfmod4 <- nhlBuild(data = rfdata, perc = 0.7, seed = 28182)
```

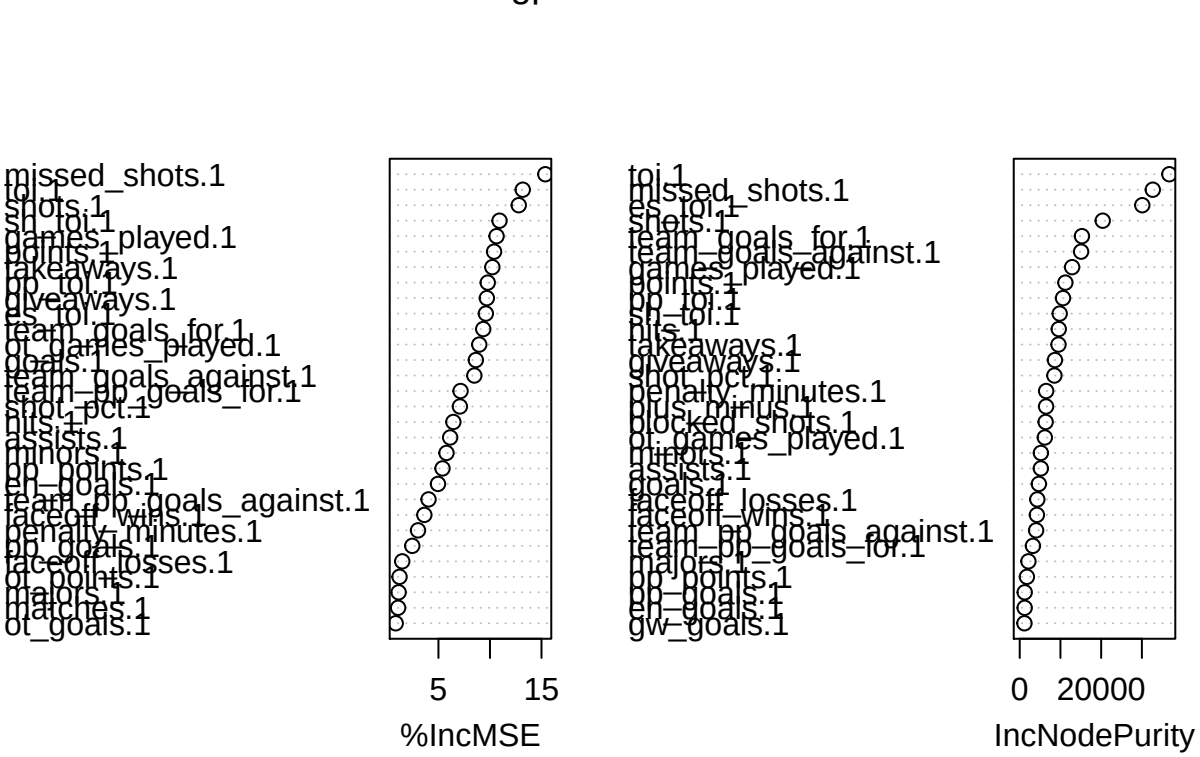
gprfmod1



gprfmod2







Several predictors make repeat appearances; these can be broken down into a few distinct categories:

1. **Offense created:** `shots`, `missed_shots`, `shot_pct`, `team_goals_for`, `points`
2. **Coaching trust:** `*toi`, `ot_games_played`

3. **Possession:** `giveaways`, `takeaways`
4. **Physicality:** `hits`, `penalty_minutes`, `majors`, `minors`

These 16 predictors consistently show as valuable analytically and pass a basic smell test; we will look to confirm or confound their importance by constructing a few boosting models and looking for similar patterns.

```
gpgbmmod1 <- nhlBuild(data = gbmdata, type = "gbm", perc = 0.7, seed = 9112)
gpgbmmod2 <- nhlBuild(data = gbmdata, type = "gbm", perc = 0.7, seed = 2857)
gpgbmmod3 <- nhlBuild(data = gbmdata, type = "gbm", perc = 0.7, seed = 31415)
gpgbmmod4 <- nhlBuild(data = gbmdata, type = "gbm", perc = 0.7, seed = 28182)
```

```
##                     var var.inf.tot
## 1                toi.1   48.553942
## 2              shots.1   46.107139
## 3        missed_shots.1   42.395097
## 4      team_goals_for.1   26.626308
## 5              points.1   22.579940
## 6        games_played.1   14.833408
## 7           takeaways.1   10.512577
## 8              es_toi.1    9.863885
## 9              pp_toi.2    9.279737
## 10          takeaways.2    9.072721
## 11             sh_toi.1    9.052247
## 12               hits.2    9.042317
## 13                toi.2    8.255706
## 14             pp_toi.1    7.797689
## 15               hits.1    7.240044
## 16             points.2    6.910308
## 17    penalty_minutes.1    6.901909
## 18      blocked_shots.2    6.551916
## 19           shot_pct.1    6.186289
## 20             es_toi.2    6.135321
## 21 team_goals_against.1    5.756593
## 22      blocked_shots.1    5.551244
## 23             sh_toi.2    4.978970
## 24            assists.2    4.953884
## 25            assists.1    4.477017
## 26 team_goals_against.2    4.368863
## 27              goals.1    4.181640
## 28             minors.2    4.037176
```

This aggregated summary shows that the same collection of predictors are present in the gbm model's list, with the exceptions of majors and giveaways. We also see a few new predictors appear, namely blocked shots, assists, and team goals against. We will consider those predictors that do not appear toward the top of both models to be of a lower tier of importance.

Algorithm

To build our model, we will first build random forest models on various subsets of the predictors, throwing in one of each of the lesser importance predictors and using only the previous season's data. We will do the same with boosting models but with the previous two seasons' data. Finally, we will build a model using the output of the other models as predictors to arrive at a consensus-type boosting model.

First, we prepare the data sets. Recall that our predictors are:

1. **Tier 1:** `shots`, `missed_shots`, `shot_pct`, `team_goals_for`, `points`, `*toi*`, `ot_games_played`, `takeaways`, `hits`, `penalty_minutes`, `minors`
2. **Tier 2:** `giveaways`, `majors`, `blocked_shots`, `assists`, `team_goals_against`

We first proceed to set up the various data sets.

```
col1 <- c(1:3, 6, 8, 15:16, 19, 24, 28, 30, 32, 35, 38:41) # Tier 1 predictors
col2 <- c(5, 20, 26, 29, 31)                               # Tier 2 predictors
gpDataRF1 <- nhlShape(2013, 2013, cols = c(col1, col2[1]), outcome = 3, rm.nhlnum = F)
gpDataRF2 <- nhlShape(2013, 2013, cols = c(col1, col2[2]), outcome = 3, rm.nhlnum = F)
gpDataRF3 <- nhlShape(2013, 2013, cols = c(col1, col2[3]), outcome = 3, rm.nhlnum = F)
gpDataRF4 <- nhlShape(2013, 2013, cols = c(col1, col2[4]), outcome = 3, rm.nhlnum = F)
gpDataRF5 <- nhlShape(2013, 2013, cols = c(col1, col2[5]), outcome = 3, rm.nhlnum = F)
gpDataRF6 <- nhlShape(2013, 2013, cols = c(col1, col2), outcome = 3, rm.nhlnum = F)
gpDataGBM1 <- nhlShape(2012, 2013, cols = c(col1, col2[1]), outcome = 3,
                       rm.nhlnum = F, rm.NA = FALSE)
gpDataGBM1 <- subset(gpDataGBM1, !is.na(games_played.1))
gpDataGBM2 <- nhlShape(2012, 2013, cols = c(col1, col2[2]), outcome = 3,
                       rm.nhlnum = F, rm.NA = FALSE)
gpDataGBM2 <- subset(gpDataGBM2, !is.na(games_played.1))
gpDataGBM3 <- nhlShape(2012, 2013, cols = c(col1, col2[3]), outcome = 3,
                       rm.nhlnum = F, rm.NA = FALSE)
gpDataGBM3 <- subset(gpDataGBM3, !is.na(games_played.1))
gpDataGBM4 <- nhlShape(2012, 2013, cols = c(col1, col2[4]), outcome = 3,
                       rm.nhlnum = F, rm.NA = FALSE)
gpDataGBM4 <- subset(gpDataGBM4, !is.na(games_played.1))
gpDataGBM5 <- nhlShape(2012, 2013, cols = c(col1, col2[5]), outcome = 3,
                       rm.nhlnum = F, rm.NA = FALSE)
gpDataGBM5 <- subset(gpDataGBM5, !is.na(games_played.1))
gpDataGBM6 <- nhlShape(2012, 2013, cols = c(col1, col2), outcome = 3,
                       rm.nhlnum = F, rm.NA = FALSE)
gpDataGBM6 <- subset(gpDataGBM6, !is.na(games_played.1))
```

Here are the lower level models.

```
gprfmod1 <- nhlBuild(gpDataRF1[, -1], perc = 1, seed = 1512)
gprfmod2 <- nhlBuild(gpDataRF2[, -1], perc = 1)
gprfmod3 <- nhlBuild(gpDataRF3[, -1], perc = 1)
gprfmod4 <- nhlBuild(gpDataRF4[, -1], perc = 1)
gprfmod5 <- nhlBuild(gpDataRF5[, -1], perc = 1)
gprfmod6 <- nhlBuild(gpDataRF6[, -1], perc = 1)
gpgbmmod1 <- nhlBuild(gpDataGBM1[, -1], type = "gbm", perc = 1)
gpgbmmod2 <- nhlBuild(gpDataGBM2[, -1], type = "gbm", perc = 1)
gpgbmmod3 <- nhlBuild(gpDataGBM3[, -1], type = "gbm", perc = 1)
gpgbmmod4 <- nhlBuild(gpDataGBM4[, -1], type = "gbm", perc = 1)
gpgbmmod5 <- nhlBuild(gpDataGBM5[, -1], type = "gbm", perc = 1)
gpgbmmod6 <- nhlBuild(gpDataGBM6[, -1], type = "gbm", perc = 1)
```

We now build a data set for the cumulative model.

```
gprf1 <- predict(gprfmod1, gpDataRF1)
gprf2 <- predict(gprfmod2, gpDataRF2)
gprf3 <- predict(gprfmod3, gpDataRF3)
```

```
gprf4 <- predict(gprfmod4, gpDataRF4)
gprf5 <- predict(gprfmod5, gpDataRF5)
gprf6 <- predict(gprfmod6, gpDataRF6)
gprf <- as.data.frame(cbind(gpDataRF1[, 1], gprf1, gprf2, gprf3, gprf4, gprf5, gprf6))
names(gprf) <- c("nhl_num", "gprf1", "gprf2", "gprf3", "gprf4", "gprf5", "gprf6")
gpgbm1 <- predict(gpgbmmod1, gpDataGBM1)
```

```
## Using 8139 trees...
```

```
gpgbm2 <- predict(gpgbmmod2, gpDataGBM2)
```

```
## Using 3338 trees...
```

```
gpgbm3 <- predict(gpgbmmod3, gpDataGBM3)
```

```
## Using 6380 trees...
```

```
gpgbm4 <- predict(gpgbmmod4, gpDataGBM4)
```

```
## Using 4965 trees...
```

```
gpgbm5 <- predict(gpgbmmod5, gpDataGBM5)
```

```
## Using 6346 trees...
```

```
gpgbm6 <- predict(gpgbmmod6, gpDataGBM6)
```

```
## Using 5388 trees...
```

```
gpgbm <- as.data.frame(cbind(gpDataGBM1[, 1], gpgbm1, gpgbm2, gpgbm3, gpgbm4, gpgbm5, gpgbm6))
names(gpgbm) <- c("nhl_num", "gpgbm1", "gpgbm2", "gpgbm3", "gpgbm4", "gpgbm5", "gpgbm6")
gpData <- merge(gprf, gpgbm, all = TRUE)
gpData <- merge(gpData, gpDataGBM1[, c(1, 34)])
```

Now we finally build a boosting model on the given data set.

```
gpModel <- nhlBuild(gpData[, -1], type = "gbm", perc = 0.7, seed = 61616)
```