# NHL Stats KNN

*Sasank Vishnubhatla*

*Friday, December 21, 2018*

## Inspiration

In the NHL, there are superstars, elite players, middle of the pack fillers, and waiver-wire warriors. Much like in the MLB, teams dip deep into their farm system and sometimes replace players that are not performing. At the end of the year, the best players usually end up with the most points scored, while average players usually sit in the middle of the pack, while enforcers and developing players sit near the bottom. Most analysts and fans agree that ranking players by their point production is one of the best ways to determine how good the player is. Using machine learning, I'm going to test that hypothesis and see if it is possible to accurately predict a player's rank given solely their core stats.

The package we will be using is the caret package. Caret has over 100 machine learning algorithms, one of them being the K Nearest Neighbors algorithm.

```r
library(caret)
```

```
## Loading required package: lattice
```

```
## Loading required package: ggplot2
```

```r
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

Instead of eyeballing the data and determining tiers like that, the K Nearest Neighbors (KNN) algorithm looks at each point and determines it's class based on it's neighbors. By using this algorithm, we can see boundaries in the data and see which players are similar to other. From this, we will be able to see artificial tiers that are prevalent statistically. However, if the boundaries are too small or there are too many classes, then there more likely to be no tiers.

## Data

The data used to train will be from the 2016-2017 season. All the data was taken from Hockey Reference.

```r
rawData = as.data.frame(read.csv("complete.csv", header = TRUE))
```

Our data is not in a nice form and contains some information that we don't need. So, let's reconfigure our dataset. Here's a list of fields we'll need to get:

- Names
- Goals
- Assists
- Plus/Minus

- Penalty Minutes
- Game Winning Goals
- Shots
- Blocks
- Hits
- Faceoff Win Percentage

We'll be looking at players that played at least 20 games. So, we need to get all the players that have played at least 20 games.

```
data16 = rawData[rawData$Season == 2016,]
totalData = data.frame("name" = data16$Player,
                       "gp" = data16$GP,
                       "rank" = data16$Rk,
                  "goals" = data16$G,
                  "assists" = data16$A,
                  "pm" = data16$plusminus,
                  "pim" = data16$PIM,
                  "gwg" = data16$GW,
                  "shots" = data16$S,
                  "blocks" = data16$BLK,
                  "hits" = data16$HIT,
                  "fowp" = data16$FO_percent)
```

Now that we have the data, we can recreate the dataframe for the KNN model. I'll include the rank so that I can match the data to the name later on after we've trained the model. However, the rank will not go into the model.

```
trainData = totalData[totalData$gp > 19,]
trainData = subset(trainData, select = c(-1, -2))
```

Now with the training data, we can create our KNN model. For the model, we'll want the output to be rank, as the higher the rank, the better the player is. First we have to pre-process our data, then we can train 2 models: a control model and the KNN model.

```
trainRank = trainData[, names(trainData) != "rank"]
preprocess = preProcess(x = trainRank, method = c("center", "scale"))
```

Now that we've pre-processed our data, let's create the two models.

```
controlModel = trainControl(method = "repeatedcv", repeats = 5)
knnModel = train(rank ~ ., data = trainData, method = "knn", trControl = controlModel, preProcess = c("
```

Now, let's see what the `knnModel` returns:

```
knnModel
```

```
## k-Nearest Neighbors
##
## 675 samples
##   9 predictor
##
## Pre-processing: centered (9), scaled (9)
## Resampling: Cross-Validated (10 fold, repeated 5 times)
## Summary of sample sizes: 607, 607, 607, 607, 607, 609, ...
## Resampling results across tuning parameters:
##
##   k   RMSE      Rsquared   MAE
```

```
##    5  51.00981  0.9413463  39.23104
##    7  50.67044  0.9426570  39.24440
##    9  51.70372  0.9413297  39.79791
##   11  51.79845  0.9420744  39.96153
##   13  52.34627  0.9413844  40.26115
##   15  52.94797  0.9405629  40.76706
##   17  53.86297  0.9391027  41.78728
##   19  54.48101  0.9386773  42.37737
##   21  55.55940  0.9369643  43.44543
##   23  56.62866  0.9351037  44.46464
##   25  57.39896  0.9338392  45.17971
##   27  58.08002  0.9327417  45.75639
##   29  58.58471  0.9322308  46.25654
##   31  59.22346  0.9313957  46.88243
##   33  59.79890  0.9307187  47.38913
##   35  60.39969  0.9298700  48.00933
##   37  61.13679  0.9286065  48.76266
##   39  61.76470  0.9276020  49.44057
##   41  62.35067  0.9268367  49.98451
##   43  63.03385  0.9256502  50.62675
##   45  63.78928  0.9243577  51.35336
##   47  64.41850  0.9233677  52.04771
##   49  65.07894  0.9221734  52.65819
##   51  65.69786  0.9209359  53.21863
##   53  66.31903  0.9197767  53.71696
##
## RMSE was used to select the optimal model using the smallest value.
## The final value used for the model was k = 7.
```

## Testing Players from Last Year (2017-2018)

Using our KNN model, let's test a few players from last year. Here is the list of players I'm selecting:

- Sidney Crosby (rank 2)
- Alexander Ovechkin (rank 21)
- Eric Staal (rank 29)
- Anze Kopitar (rank 89)
- John Klingberg (rank 107)
- Brayden Point (rank 161)
- Tom Wilson (rank 356)
- Austin Watson (rank 390)
- Ryan Reaves (rank 449)

So let's make our testing data frame.

```r
data17 = rawData[rawData$Season == 2017,]
testData = data.frame("name" = data17$Player,
                       "gp" = data17$GP,
                       "rank" = data17$Rk,
                  "goals" = data17$G,
                  "assists" = data17$A,
                  "pm" = data17$plusminus,
                  "pim" = data17$PIM,
                  "gwg" = data17$GW,
```

```r
                  "shots" = data17$S,
                  "blocks" = data17$BLK,
                  "hits" = data17$HIT,
                  "fowp" = data17$FO_percent)
selectRanks = c(2, 21, 29, 89, 107, 161, 356, 390, 449)
testData = testData[testData$rank %in% selectRanks,]
testingData = subset(testData, select = c(-1, -2))
```

Now, let's use the predict function to predict for each of these players.

```r
knnPrediction = predict.train(knnModel, newdata = testingData)
```

So, our predictions are:

```r
knnPrediction
```

```
## [1]  13.42857  29.14286  30.28571 101.85714 123.00000 163.57143 382.14286
## [8] 373.00000 512.14286
```

So, let's match the prediction with the player now:

```r
comparison = subset(testData, select = c(1, 3))
comparison$prediction = knnPrediction
comparison
```

```
##                           name rank prediction
## 2    Sidney Crosby\\crosbsi01    2   13.42857
## 21   Alex Ovechkin\\ovechal01   21   29.14286
## 29      Eric Staal\\staaler01   29   30.28571
## 89     Anze Kopitar\\kopitan01   89  101.85714
## 107 John Klingberg\\klingjo01  107  123.00000
## 161  Brayden Point\\pointbr01  161  163.57143
## 356     Tom Wilson\\wilsoto01  356  382.14286
## 390  Austin Watson\\watsoau01  390  373.00000
## 449    Ryan Reaves\\reavery01  449  512.14286
```

With just one year of data, we see that our model under predicts most players in their rank. So, let's re-train our model but with multiple years of data.

### Re-training the Model With 5 Years Worth of Data

Let's use data from 2011-2012 to 2016-2017 instead of just one year of data.

```r
years = c(2012, 2013, 2014, 2015, 2016)
data1116 = rawData[rawData$Season %in% years,]
totalData = data.frame("name" = data1116$Player,
                       "gp" = data1116$GP,
                       "rank" = data1116$Rk,
                  "goals" = data1116$G,
                  "assists" = data1116$A,
                  "pm" = data1116$plusminus,
                  "pim" = data1116$PIM,
                  "gwg" = data1116$GW,
                  "shots" = data1116$S,
                  "blocks" = data1116$BLK,
                  "hits" = data1116$HIT,
                  "fowp" = data1116$FO_percent)
```

```r
trainData = totalData[totalData$gp > 19,]
trainData = subset(trainData, select = c(-1, -2))

controlModel = trainControl(method = "repeatedcv", repeats = 5)
knnModel = train(rank ~ ., data = trainData, method = "knn", trControl = controlModel, preProcess = c("
```

So, our new model looks like this:

```r
knnModel
```

```
## k-Nearest Neighbors
##
## 3297 samples
##    9 predictor
##
## Pre-processing: centered (9), scaled (9)
## Resampling: Cross-Validated (10 fold, repeated 5 times)
## Summary of sample sizes: 2968, 2968, 2967, 2966, 2967, 2968, ...
## Resampling results across tuning parameters:
##
##   k   RMSE      Rsquared   MAE
##    5  66.61172  0.8903800  49.66203
##    7  65.29050  0.8953526  48.51905
##    9  64.88180  0.8972419  48.07861
##   11  64.78164  0.8979291  47.80102
##   13  64.57614  0.8989377  47.68960
##   15  64.44363  0.8997463  47.62924
##   17  64.63827  0.8994884  47.77185
##   19  64.79318  0.8994257  47.90355
##   21  65.10804  0.8987841  48.08787
##   23  65.30219  0.8984461  48.20189
##   25  65.68613  0.8974133  48.44599
##   27  65.97473  0.8966945  48.65017
##   29  66.36925  0.8956176  48.91066
##   31  66.66951  0.8948468  49.15850
##   33  66.89451  0.8943612  49.34222
##   35  67.13160  0.8938247  49.54323
##   37  67.33772  0.8933782  49.68456
##   39  67.60309  0.8926944  49.89918
##   41  67.93215  0.8917603  50.13167
##   43  68.15525  0.8911929  50.31148
##   45  68.39234  0.8905936  50.50002
##   47  68.62302  0.8900040  50.69770
##   49  68.89119  0.8893020  50.89284
##   51  69.16732  0.8885429  51.08548
##   53  69.39341  0.8879752  51.25251
##
## RMSE was used to select the optimal model using the smallest value.
## The final value used for the model was k = 15.
```

We can now re-test it:

```r
knnPrediction = predict.train(knnModel, newdata = testingData)
```

So, let's match the prediction with the player now:

```r
newComparison = subset(testData, select = c(1, 3))
newComparison$prediction = knnPrediction
newComparison
```

```
##                           name rank prediction
## 2    Sidney Crosby\\crosbsi01    2   14.80000
## 21   Alex Ovechkin\\ovechal01   21   36.66667
## 29      Eric Staal\\staaler01   29   41.26667
## 89    Anze Kopitar\\kopitan01   89   97.46667
## 107 John Klingberg\\klingjo01  107  159.60000
## 161  Brayden Point\\pointbr01  161  139.33333
## 356     Tom Wilson\\wilsoto01  356  398.60000
## 390  Austin Watson\\watsoau01  390  373.13333
## 449    Ryan Reaves\\reavery01  449  471.13333
```

So, we can see with the new data, our model still isn't able to pinpoint the ranks as well as we'd expect. This just shows how variable scoring is year to year, and how difficult it is to be consistent in the NHL.