Could an Independent Yorkshire Win the World Cup - LASSOs and Player Positions

Robert Hickman

2018-06-07

Recently, a Yorkshire national football team appeared in a league of national teams for stateless people. This got me wondering how the historic counties of the UK would do at the world cup. Could any of them compete with full international teams?

This is the complete script for an short article I wrote for CityMetric on the topic. It's split over 6 separate parts and is pretty hefty but contains pretty much everything you need to clone the article. Last time, we got the shapefiles for the historic counties of the UK and scraped the player data we'll use to build the teams

```r
library(dplyr)
library(magrittr)
library(rvest)
library(data.table)
library(ggplot2)
#we'll use glmnet to do LASSO regression to determine players positional ability
library(glmnet)
```

Work Out Player Position Ability

The data we've scraped only gives a player's overall 'ability' and their abilities on specific skills (e.g. strength, long shots, dribbling…). We want to use this to work out how good each player is at each position.

It's logical to assume that a player's overall ability is how good they are at their main position (the position listed first on their page on fifaindex.com). We can therefore use LASSO regression to work out which stats are contributing to their overall ability score. For instance, we would expect that a goalkeepers overall ability score is just a function of gk_positioning, gk_diving, gk_handling and so on… and doesn't care about (e.g.) dribbling.

This positional ability score is important as we can't just select the 11 best players for each team as we might end up playing a goalkeeper and 10 defenders (or etc.). We need to make sure we select the best palyers for each position on a realistic formation.

Unfortunately, there's going to be no real way to tell between a players ability to play on either side of the field. There's some correlation with their footedness ($foot), but it's not worth going too in the weeds about that. So first we want to make all position symmetrical (i.e. we do discriminate between left and right sided positions).

```r
all_players_data %<>% mutate(symmetric_position = gsub("L|R", "W", main_position))

unique(as.character(all_players_data$main_position))
```

```
##  [1] "CF"  "LW"  "ST"  "GK"  "CAM" "CM"  "CB"  "CDM" "RW"  "LB"  "RM"
## [12] "LM"  "RB"  "LWB" "RWB"
```

```r
unique(all_players_data$symmetric_position)
```

```
##  [1] "CF"  "WW"  "ST"  "GK"  "CAM" "CM"  "CB"  "CDM" "WB"  "WM"  "WWB"
```

First we need to convert the stats that we're going to use for this regression/prediction into matrix form

```
player_stats <- all_players_data %>%
  #select only the players stats for each skill
  select(c(12, 14:47)) %>%
  #transition data into matrix
  model.matrix(overall~., .)

#show the first 6 instances of the first 5 stats
head(player_stats[,c(1:5)])
```

```
##   (Intercept) ball_control dribbling marking slide_tackle
## 1           1           96        97      13           26
## 2           1           93        91      22           23
## 3           1           95        96      21           33
## 4           1           91        86      30           38
## 5           1           48        30      10           11
## 6           1           42        18      13           13
```

To show how this works I'm going to illustrate it using goalkeepers and predicting the ability of each outfield player to play in goal.

```
#filter out only the goalkeepers
goalkeepers <- all_players_data %>%
  filter(symmetric_position == "GK")

#select a percentage of these to use as training data
sample_percent <- 10
train_samples <- sample(1:nrow(goalkeepers), (nrow(goalkeepers)/100)*sample_percent)

gk_stats <- goalkeepers %>%
  slice(train_samples) %>%
  select(c(12, 14:47))

#get the stats per skill
train_matrix <- model.matrix(overall~., gk_stats)
#and the overall (gk) ability
train_ability <- gk_stats$overall

#perform the regression
cv_model <- cv.glmnet(train_matrix, train_ability)

#look at the weightings given to important variables using lambda value that gives minimum mean cross-v
coef(cv_model, s = "lambda.min")
```

```
## 36 x 1 sparse Matrix of class "dgCMatrix"
##                            1
## (Intercept)     1.1310857573
## (Intercept)     .
## ball_control    .
## dribbling       .
## marking         .
## slide_tackle   -0.0004103623
## stand_tackle    .
## aggression      .
## reactions       0.1099498678
## positioning     .
## interceptions   .
## vision          .
## composure       0.0006834373
## crossing        .
## short_pass      .
## long_pass       .
## acceleration    .
## stamina         .
## strength        .
## balance         .
## sprint_speed    .
## agility         .
## jumping         .
## heading         .
## shot_power      .
## finishing       .
## long_shots      .
## curve           .
## free_kicks      .
## penalties       0.0019077112
## volleys         .
## gk_positioning  0.2025813218
## gk_diving       0.2150702062
## gk_handling     0.2204928089
## gk_kicking      0.0486648298
## gk_reflexes     0.2003381582
```

The regression selects only variables which have a strong relationship with the outcome (overall ability in the gk position in this case). As expected, it selects only the gk_… skillset and also a players reactions, which makes sense if goalkeepers have to make point blank saves.

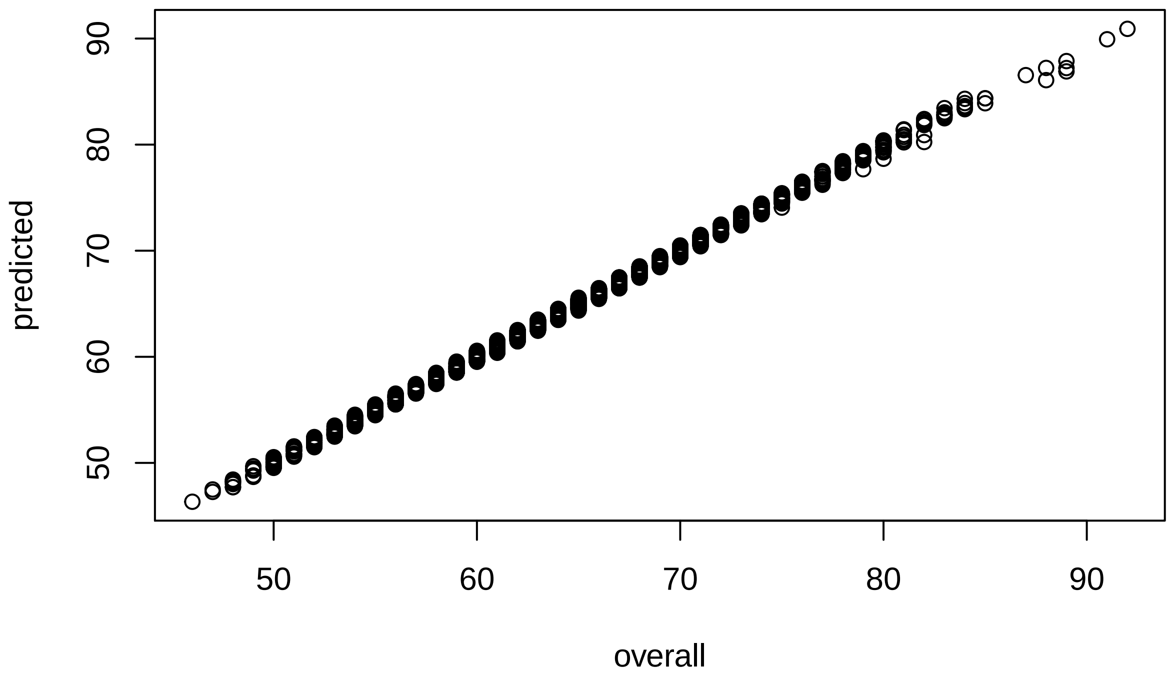
We can validate this by predicting the overall ability of the goalkeepers that aren't in the training set fairly simply

```r
#find non training examples of goalkeepers
test_matrix <- goalkeepers %>%
  slice(-train_samples) %>%
  select(c(12, 14:47)) %>%
  model.matrix(overall~., .)

#get the overall ability for these players
gk_abilities <- goalkeepers %>%
  slice(-train_samples) %>%
  select(overall)

#predict their overall ability based on their stats
gk_abilities$predicted <- as.vector(predict(cv_model, newx = test_matrix, s = "lambda.min", type="respo

#plot these
plot(data = gk_abilities, predicted~overall)
```



Which gives a very good fit. This is expected for the dataset we're using as the overall ability hasbeen directly calclulated from the complete set of skills using some hidden algorithm. In the real world, the actual vs. predicted results would most likely have more noise.

We can use this model now to predict how well outfield players would fare in goal, given that they have (low) ratings for all of these skills

```
#find all outfield players and convert stats the matrix
outfield_players <- all_players_data %>%
  filter(symmetric_position != "GK") %>%
  select(c(12, 14:47)) %>%
  model.matrix(overall~., .)

#get the names of each outfield palyer
outfield_goalkeepers <- all_players_data %>%
  filter(symmetric_position != "GK") %>%
  select(name)

#predict how well each outfield player would do in goal
outfield_goalkeepers$predicted_ability <- as.vector(predict(cv_model, newx = outfield_players, s = "lam

head(outfield_goalkeepers)
```

```
##                  name predicted_ability
## 1        Lionel Messi          20.65968
## 2   Cristiano Ronaldo          21.60466
## 3              Neymar          20.90301
## 4        Luis Suárez          38.48660
## 5    Kevin De Bruyne          22.17731
## 6 Robert Lewandowski          20.11308
```

It is of course, extremely ironic that Luis Suarez scores relatively highly as an outfield player in goal given his history.

We can use this technique to predict how each player would play in each position using the following function

```
#function to predict how each player would play in each position
get_position_weights <- function(position, sample_percent) {
  #filter data
  position_df <- all_players_data %>%
    filter(symmetric_position == position)

  #get training data
  train_samples <- sample(1:nrow(position_df), (nrow(position_df)/100)*sample_percent)

  train_stats <- position_df %>%
    .[train_samples,] %>%
    select(c(12, 14:47))

  train_matrix <- model.matrix(overall~., train_stats)

  train_ability <- train_stats$overall

  #use LASSO regression to find weighting of significant covariates
  cv_model <- cv.glmnet(train_matrix, train_ability)

  #predict players ability in that position
  position_ability <- predict(cv_model, newx = player_stats, s = "lambda.min", type="response")
}

#run through every mirrored position
#using a high percentage of palyers in training set (50%) as in theory should be perfect regression
position_abilities <- lapply(unique(all_players_data$symmetric_position),
                             get_position_weights, sample_percent = 50) %>%
  do.call(cbind, .) %>%
  data.frame()

#name each position
names(position_abilities) <- unique(all_players_data$symmetric_position)

#bind this to the data we have on all the players
all_players_data <- cbind(all_players_data, position_abilities) %>%
  #convert all non-natural goalkeepers goalkeeping ability to zero
  #want to make sure no non-goalkeepers are chosen in goal
  mutate(GK = ifelse(main_position == "GK", overall, 0))

#show the first 6 rows
head(select(all_players_data, c(1, 49:60)))
```

```
##                 name symmetric_position       CF       WW       ST GK
## 1       Lionel Messi                 CF 92.63857 92.16985 89.68254  0
## 2 Cristiano Ronaldo                 WW 91.78822 90.94465 92.61850  0
## 3             Neymar                 WW 88.81305 89.22014 85.14340  0
## 4       Luis Suárez                 ST 88.80038 87.29764 88.89122  0
## 5      Manuel Neuer                 GK 44.94065 39.48991 36.62275 92
## 6            De Gea                 GK 43.30221 36.69296 34.57000 91
##        CAM       CM       CB      CDM       WB       WM      WWB
## 1 93.38891 85.16995 46.29257 60.37721 58.40458 91.19466 62.58392
## 2 90.37737 83.20217 53.81265 62.79760 62.32275 90.33120 66.85699
## 3 89.09586 80.73827 47.46581 60.21325 60.28660 88.30355 64.62051
## 4 87.61615 81.24147 59.99181 66.79817 65.27095 86.42500 67.58171
## 5 47.25503 49.57625 34.38189 42.75352 35.17281 44.16759 29.43039
## 6 43.77373 46.38808 34.74957 42.21545 36.29784 40.85131 29.83339
```