FIFA Player position classification and market value estimation

Giovanni Dal Mas

Player position classification and market value estimation

1. Abstract and motivation

FIFA is one of the most known videogames and the most famous sport title in the industry, in particular we considered FIFA 22 edition.

Each player covers a specific position on the field; what we want to do in the first part of the project is building some models to classify the position of the player, based on the values of its attributes. It's important to consider that some players may share some features with footballers playing in another position, and this may influence our task. For example, some attacking midfielders (CAM) have a good shot and pace, just like wingers (RW, LW). We will keep this into account and adjust our classification accordingly.

In the second part of our project we will perform a regression task, building some models to evaluate the market price of a football player, using Ridge and Lasso regression.

2. The dataset - Description & EDA

The original dataset has been extracted from https://sofifa.com/ and contains 19239 players described by 110 different features.

2.1 DataFrame inspection and rough slicing

```
## Loading required package: viridisLite
```

```
## Welcome! Want to learn more? See two factoextra-related books at https://goo.gl/ve3WBa
```

```
##
## 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
```

```
##
## Attaching package: 'MASS'
## The following object is masked from 'package:dplyr':
##
##     select

## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v forcats   1.0.0     v readr     2.1.4
## v lubridate 1.9.2     v tibble    3.2.1
## v purrr     1.0.1     v tidyr     1.3.0
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## x MASS::select()  masks dplyr::select()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become error
## corrplot 0.92 loaded
##
##
## Attaching package: 'gridExtra'
##
##
## The following object is masked from 'package:dplyr':
##
##     combine
##
##
##
## Attaching package: 'reshape'
##
##
## The following object is masked from 'package:lubridate':
##
##     stamp
##
##
## The following objects are masked from 'package:tidyr':
##
##     expand, smiths
##
##
## The following object is masked from 'package:class':
##
##     condense
##
##
## The following object is masked from 'package:dplyr':
##
##     rename
##
##
## The following objects are masked from 'package:reshape2':
##
##     colsplit, melt, recast
##
##
## Loading required package: lattice
```

```
##
##
## Attaching package: 'caret'
##
##
## The following object is masked from 'package:purrr':
##
##     lift
##
##
## randomForest 4.7-1.1
##
## Type rfNews() to see new features/changes/bug fixes.
##
##
## Attaching package: 'randomForest'
##
##
## The following object is masked from 'package:gridExtra':
##
##     combine
##
##
## The following object is masked from 'package:dplyr':
##
##     combine
##
##
## The following object is masked from 'package:ggplot2':
##
##     margin
##
##
##
## Attaching package: 'data.table'
##
##
## The following object is masked from 'package:reshape':
##
##     melt
##
##
## The following objects are masked from 'package:lubridate':
##
##     hour, isoweek, mday, minute, month, quarter, second, wday, week,
##     yday, year
##
##
## The following object is masked from 'package:purrr':
##
##     transpose
##
##
## The following objects are masked from 'package:dplyr':
```

```
##
##     between, first, last
##
##
## The following objects are masked from 'package:reshape2':
##
##     dcast, melt
##
##
## Loading required package: Matrix
##
##
## Attaching package: 'Matrix'
##
##
## The following object is masked from 'package:reshape':
##
##     expand
##
##
## The following objects are masked from 'package:tidyr':
##
##     expand, pack, unpack
##
##
## Loaded glmnet 4.1-7
```

We set the seed for reproducible experiments

```
set.seed(42)
```

First we load the dataset, and check the dimension.

```
players_full <- read.csv("players_22.csv") #full dataframe
dim(players_full) #full dataset
```

```
## [1] 19239   110
```

We have more or less 20k players with 110 attributes. Below we look at how those attributes are named.

```
colnames(players_full)
```

```
##   [1] "sofifa_id"                   "player_url"
##   [3] "short_name"                  "long_name"
##   [5] "player_positions"            "overall"
##   [7] "potential"                   "value_eur"
##   [9] "wage_eur"                    "age"
##  [11] "dob"                         "height_cm"
##  [13] "weight_kg"                   "club_team_id"
##  [15] "club_name"                   "league_name"
##  [17] "league_level"                "club_position"
##  [19] "club_jersey_number"          "club_loaned_from"
##  [21] "club_joined"                 "club_contract_valid_until"
##  [23] "nationality_id"              "nationality_name"
##  [25] "nation_team_id"              "nation_position"
##  [27] "nation_jersey_number"        "preferred_foot"
##  [29] "weak_foot"                   "skill_moves"
##  [31] "international_reputation"    "work_rate"
##  [33] "body_type"                   "real_face"
##  [35] "release_clause_eur"          "player_tags"
##  [37] "player_traits"               "pace"
##  [39] "shooting"                    "passing"
##  [41] "dribbling"                   "defending"
##  [43] "physic"                      "attacking_crossing"
##  [45] "attacking_finishing"         "attacking_heading_accuracy"
##  [47] "attacking_short_passing"     "attacking_volleys"
##  [49] "skill_dribbling"             "skill_curve"
##  [51] "skill_fk_accuracy"           "skill_long_passing"
##  [53] "skill_ball_control"          "movement_acceleration"
##  [55] "movement_sprint_speed"       "movement_agility"
##  [57] "movement_reactions"          "movement_balance"
##  [59] "power_shot_power"            "power_jumping"
##  [61] "power_stamina"               "power_strength"
##  [63] "power_long_shots"            "mentality_aggression"
##  [65] "mentality_interceptions"     "mentality_positioning"
##  [67] "mentality_vision"            "mentality_penalties"
##  [69] "mentality_composure"         "defending_marking_awareness"
##  [71] "defending_standing_tackle"   "defending_sliding_tackle"
##  [73] "goalkeeping_diving"          "goalkeeping_handling"
##  [75] "goalkeeping_kicking"         "goalkeeping_positioning"
##  [77] "goalkeeping_reflexes"        "goalkeeping_speed"
##  [79] "ls"                          "st"
##  [81] "rs"                          "lw"
##  [83] "lf"                          "cf"
##  [85] "rf"                          "rw"
##  [87] "lam"                         "cam"
##  [89] "ram"                         "lm"
##  [91] "lcm"                         "cm"
##  [93] "rcm"                         "rm"
##  [95] "lwb"                         "ldm"
##  [97] "cdm"                         "rdm"
##  [99] "rwb"                         "lb"
## [101] "lcb"                         "cb"
## [103] "rcb"                         "rb"
## [105] "gk"                          "player_face_url"
## [107] "club_logo_url"               "club_flag_url"
## [109] "nation_logo_url"             "nation_flag_url"
```

To get a better general idea, we also want to look at the type of data they provide

```r
head(players_full, 10)
```

```
##    sofifa_id                                                    player_url
## 1     158023              https://sofifa.com/player/158023/lionel-messi/220002
## 2     188545        https://sofifa.com/player/188545/robert-lewandowski/220002
## 3      20801 https://sofifa.com/player/20801/c-ronaldo-dos-santos-aveiro/220002
## 4     190871  https://sofifa.com/player/190871/neymar-da-silva-santos-jr/220002
## 5     192985            https://sofifa.com/player/192985/kevin-de-bruyne/220002
## 6     200389                  https://sofifa.com/player/200389/jan-oblak/220002
## 7     231747              https://sofifa.com/player/231747/kylian-mbappe/220002
## 8     167495               https://sofifa.com/player/167495/manuel-neuer/220002
## 9     192448      https://sofifa.com/player/192448/marc-andre-ter-stegen/220002
## 10    202126                 https://sofifa.com/player/202126/harry-kane/220002
##           short_name                           long_name player_positions
## 1           L. Messi      Lionel Andrés Messi Cuccittini       RW, ST, CF
## 2     R. Lewandowski                  Robert Lewandowski               ST
## 3  Cristiano Ronaldo Cristiano Ronaldo dos Santos Aveiro           ST, LW
## 4          Neymar Jr       Neymar da Silva Santos Júnior          LW, CAM
## 5       K. De Bruyne                     Kevin De Bruyne          CM, CAM
## 6           J. Oblak                           Jan Oblak               GK
## 7          K. Mbappé                 Kylian Mbappé Lottin           ST, LW
## 8           M. Neuer                  Manuel Peter Neuer               GK
## 9       M. ter Stegen               Marc-André ter Stegen               GK
## 10           H. Kane                          Harry Kane               ST
##    overall potential value_eur wage_eur age        dob height_cm weight_kg
## 1       93        93  78000000   320000  34 1987-06-24       170        72
## 2       92        92 119500000   270000  32 1988-08-21       185        81
## 3       91        91  45000000   270000  36 1985-02-05       187        83
## 4       91        91 129000000   270000  29 1992-02-05       175        68
## 5       91        91 125500000   350000  30 1991-06-28       181        70
## 6       91        93 112000000   130000  28 1993-01-07       188        87
## 7       91        95 194000000   230000  22 1998-12-20       182        73
## 8       90        90  13500000    86000  35 1986-03-27       193        93
## 9       90        92  99000000   250000  29 1992-04-30       187        85
## 10      90        90 129500000   240000  27 1993-07-28       188        89
##    club_team_id           club_name           league_name league_level
## 1            73 Paris Saint-Germain        French Ligue 1            1
## 2            21   FC Bayern München   German 1. Bundesliga            1
## 3            11   Manchester United English Premier League            1
## 4            73 Paris Saint-Germain        French Ligue 1            1
## 5            10     Manchester City English Premier League            1
## 6           240  Atlético de Madrid Spain Primera Division            1
## 7            73 Paris Saint-Germain        French Ligue 1            1
## 8            21   FC Bayern München   German 1. Bundesliga            1
## 9           241        FC Barcelona Spain Primera Division            1
## 10           18   Tottenham Hotspur English Premier League            1
##    club_position club_jersey_number club_loaned_from club_joined
## 1             RW                 30                   2021-08-10
## 2             ST                  9                   2014-07-01
## 3             ST                  7                   2021-08-27
## 4             LW                 10                   2017-08-03
## 5            RCM                 17                   2015-08-30
## 6             GK                 13                   2014-07-16
## 7             ST                  7                   2018-07-01
## 8             GK                  1                   2011-07-01
## 9             GK                  1                   2014-07-01
## 10            ST                 10                   2010-07-28
##    club_contract_valid_until nationality_id nationality_name nation_team_id
## 1                       2023             52        Argentina           1369
## 2                     2023             37          Poland           1353
## 3                     2023             38        Portugal           1354
## 4                     2025             54          Brazil             NA
## 5                     2025              7         Belgium           1325
## 6                     2023             44        Slovenia             NA
## 7                     2022             18          France           1335
## 8                     2023             21         Germany           1337
## 9                     2025             21         Germany             NA
## 10                    2024             14         England           1318
##    nation_position nation_jersey_number preferred_foot weak_foot skill_moves
## 1               RW                   10           Left         4           4
## 2               RS                    9          Right         4           4
## 3               ST                    7          Right         4           5
## 4                                    NA          Right         5           5
## 5              RCM                    7          Right         5           4
## 6                                    NA          Right         3           1
## 7               LW                   10          Right         4           5
## 8               GK                    1          Right         4           1
## 9                                    NA          Right         4           1
## 10              ST                    9          Right         5           3
##    international_reputation     work_rate body_type real_face
## 1                         5    Medium/Low    Unique       Yes
## 2                         5   High/Medium    Unique       Yes
## 3                         5      High/Low    Unique       Yes
## 4                         5   High/Medium    Unique       Yes
## 5                         4     High/High    Unique       Yes
## 6                         5 Medium/Medium    Unique       Yes
## 7                         4      High/Low    Unique       Yes
## 8                         5 Medium/Medium    Unique       Yes
## 9                         4 Medium/Medium    Unique       Yes
## 10                        4     High/High    Unique       Yes
##    release_clause_eur
## 1           144300000
## 2           197200000
## 3            83300000
## 4           238700000
## 5           232200000
## 6           238000000
## 7           373500000
## 8            22300000
## 9           210400000
## 10          246100000
##                                                                                         playe
## 1           #Dribbler, #Distance Shooter, #FK Specialist, #Acrobat, #Clinical Finisher, #Complete F
## 2                                    #Aerial Threat, #Distance Shooter, #Clinical Finisher, #Complete F
## 3  #Aerial Threat, #Dribbler, #Distance Shooter, #Crosser, #Acrobat, #Clinical Finisher, #Complete F
## 4                       #Speedster, #Dribbler, #Playmaker, #FK Specialist, #Acrobat, #Complete Midf
## 5                       #Dribbler, #Playmaker, #Engine, #Distance Shooter, #Crosser, #Complete Midf
## 6
## 7                                     #Speedster, #Dribbler, #Acrobat, #Clinical Finisher, #Complete F
## 8
## 9
## 10                                                                  #Distance Shooter, #Clinical Fi
##
```

```
## 1  Finesse Shot, Long Shot Taker (AI), Playmaker (AI), Outside Foot Shot, One Club Player, Chip Shot
## 2                                                                         Solid Player, Finesse Shot, Outs
## 3                                               Power Free-Kick, Flair, Long Shot Taker (AI), Speed Dri
## 4                                   Injury Prone, Flair, Speed Dribbler (AI), Playmaker (AI), Outside Foot
## 5              Injury Prone, Leadership, Early Crosser, Long Passer (AI), Long Shot Taker (AI), Play
## 6                                                                                                     GK
## 7                                                             Flair, Speed Dribbler (AI), Outside Foot
## 8                                                                   Leadership, GK Long Throw, Rushes O
## 9                                                                           Rushes Out Of Goal, Comes
## 10                                             Leadership, Long Passer (AI), Long Shot Taker (AI), Play
##    pace shooting passing dribbling defending physic attacking_crossing
## 1    85       92      91        95        34     65                 85
## 2    78       92      79        86        44     82                 71
## 3    87       94      80        88        34     75                 87
## 4    91       83      86        94        37     63                 85
## 5    76       86      93        88        64     78                 94
## 6    NA       NA      NA        NA        NA     NA                 13
## 7    97       88      80        92        36     77                 78
## 8    NA       NA      NA        NA        NA     NA                 15
## 9    NA       NA      NA        NA        NA     NA                 18
## 10   70       91      83        83        47     83                 80
##    attacking_finishing attacking_heading_accuracy attacking_short_passing
## 1                   95                         70                      91
## 2                   95                         90                      85
## 3                   95                         90                      80
## 4                   83                         63                      86
## 5                   82                         55                      94
## 6                   11                         15                      43
## 7                   93                         72                      85
## 8                   13                         25                      60
## 9                   14                         11                      61
## 10                  94                         86                      85
##    attacking_volleys skill_dribbling skill_curve skill_fk_accuracy
## 1                 88              96          93                94
## 2                 89              85          79                85
## 3                 86              88          81                84
## 4                 86              95          88                87
## 5                 82              88          85                83
## 6                 13              12          13                14
## 7                 83              93          80                69
## 8                 11              30          14                11
## 9                 14              21          18                12
## 10                88              83          83                65
##    skill_long_passing skill_ball_control movement_acceleration
## 1                  91                 96                    91
## 2                  70                 88                    77
## 3                  77                 88                    85
## 4                  81                 95                    93
## 5                  93                 91                    76
## 6                  40                 30                    43
## 7                  71                 91                    97
## 8                  68                 46                    54
## 9                  63                 30                    38
## 10                 86                 85                    65
##    movement_sprint_speed movement_agility movement_reactions movement_balance
## 1                     80               91                 94               95
## 2                     79               77                 93               82
## 3                     88               86                 94               74
## 4                     89               96                 89               84
## 5                     76               79                 91               78
## 6                     60               67                 88               49
## 7                     97               92                 93               83
## 8                     60               51                 87               35
## 9                     50               39                 86               43
## 10                    74               71                 92               70
##    power_shot_power power_jumping power_stamina power_strength power_long_shots
## 1                86            68            72             69               94
## 2                90            85            76             86               87
## 3                94            95            77             77               93
## 4                80            64            81             53               81
## 5                91            63            89             74               91
## 6                59            78            41             78               12
## 7                86            78            88             77               82
## 8                68            77            43             80               16
## 9                66            79            35             78               10
## 10               91            79            83             85               86
##    mentality_aggression mentality_interceptions mentality_positioning
## 1                    44                      40                    93
## 2                    81                      49                    95
## 3                    63                      29                    95
## 4                    63                      37                    86
## 5                    76                      66                    88
## 6                    34                      19                    11
## 7                    62                      38                    92
## 8                    29                      30                    12
## 9                    43                      22                    11
## 10                   80                      44                    94
##    mentality_vision mentality_penalties mentality_composure
## 1                95                  75                  96
## 2                81                  90                  88
## 3                76                  88                  95
## 4                90                  93                  93
## 5                94                  83                  89
## 6                65                  11                  68
## 7                82                  79                  88
## 8                70                  47                  70
## 9                70                  25                  70
## 10               87                  91                  91
##    defending_marking_awareness defending_standing_tackle
## 1                           20                        35
## 2                           35                        42
## 3                           24                        32
## 4                           35                        32
## 5                           68                        65
## 6                           27                        12
## 7                           26                        34
## 8                           17                        10
## 9                           25                        13
## 10                          50                         36
##    defending_sliding_tackle goalkeeping_diving goalkeeping_handling
## 1                        24                  6                   11
## 2                        19                 15                    6
## 3                        24                  7                   11
## 4                        29                  9                    9
## 5                        53                 15                   13
## 6                        18                 87                   92
## 7                        32                 13                    5
## 8                        11                 88                   88
## 9                        10                 88                   85
## 10                       38                  8                   10
##    goalkeeping_kicking goalkeeping_positioning goalkeeping_reflexes
## 1                   15                      14                    8
## 2                   12                       8                   10
## 3                   15                      14                   11
## 4                   15                      15                   11
## 5                    5                      10                   13
## 6                   78                      90                   90
## 7                    7                      11                    6
## 8                   91                      89                   88
## 9                   88                      88                   90
## 10                  11                      14                   11
##    goalkeeping_speed   ls   st   rs lw lf cf rf rw  lam  cam  ram   lm  lcm
## 1                 NA 89+3 89+3 89+3 92 93 93 93 92   93   93   93 91+2 87+3
## 2                 NA 90+2 90+2 90+2 85 88 88 88 85 86+3 86+3 86+3 84+3 80+3
## 3                 NA 90+1 90+1 90+1 88 89 89 89 88 86+3 86+3 86+3 86+3 78+3
## 4                 NA 83+3 83+3 83+3 90 88 88 88 90 89+2 89+2 89+2 89+2 82+3
## 5                 NA 83+3 83+3 83+3 88 87 87 87 88 89+2 89+2 89+2 89+2 89+2
## 6                 50 33+3 33+3 33+3 32 35 35 35 32 38+3 38+3 38+3 35+3 38+3
## 7                 NA 89+3 89+3 89+3 90 90 90 90 90 89+3 89+3 89+3 89+3 81+3
## 8                 56 40+3 40+3 40+3 40 43 43 43 40 47+3 47+3 47+3 44+3 50+3
## 9                 43 35+3 35+3 35+3 35 38 38 38 35 42+3 42+3 42+3 39+3 45+3
## 10                NA 88+2 88+2 88+2 84 86 86 86 84 85+3 85+3 85+3 84+3 82+3
##      cm  rcm   rm  lwb  ldm  cdm  rdm  rwb   lb  lcb   cb  rcb   rb   gk
## 1  87+3 87+3 91+2 66+3 64+3 64+3 64+3 66+3 61+3 50+3 50+3 50+3 61+3 19+3
## 2  80+3 80+3 84+3 64+3 66+3 66+3 66+3 64+3 61+3 60+3 60+3 60+3 61+3 19+3
## 3  78+3 78+3 86+3 63+3 59+3 59+3 59+3 63+3 60+3 53+3 53+3 53+3 60+3 20+3
## 4  82+3 82+3 89+2 67+3 63+3 63+3 63+3 67+3 62+3 50+3 50+3 50+3 62+3 20+3
## 5  89+2 89+2 89+2 79+3 80+3 80+3 80+3 79+3 75+3 69+3 69+3 69+3 75+3 21+3
## 6  38+3 38+3 35+3 32+3 36+3 36+3 36+3 32+3 32+3 33+3 33+3 33+3 32+3 89+3
## 7  81+3 81+3 89+3 67+3 63+3 63+3 63+3 67+3 63+3 54+3 54+3 54+3 63+3 18+3
## 8  50+3 50+3 44+3 37+3 43+3 43+3 43+3 37+3 35+3 34+3 34+3 34+3 35+3 88+2
## 9  45+3 45+3 39+3 33+3 41+3 41+3 41+3 33+3 31+3 33+3 33+3 33+3 31+3 88+3
## 10 82+3 82+3 84+3 67+3 68+3 68+3 68+3 67+3 64+3 61+3 61+3 61+3 64+3 20+3
##                                   player_face_url
## 1  https://cdn.sofifa.net/players/158/023/22_120.png
## 2  https://cdn.sofifa.net/players/188/545/22_120.png
## 3  https://cdn.sofifa.net/players/020/801/22_120.png
## 4  https://cdn.sofifa.net/players/190/871/22_120.png
## 5  https://cdn.sofifa.net/players/192/985/22_120.png
## 6  https://cdn.sofifa.net/players/200/389/22_120.png
## 7  https://cdn.sofifa.net/players/231/747/22_120.png
## 8  https://cdn.sofifa.net/players/167/495/22_120.png
## 9  https://cdn.sofifa.net/players/192/448/22_120.png
## 10 https://cdn.sofifa.net/players/202/126/22_120.png
##                              club_logo_url
## 1   https://cdn.sofifa.net/teams/73/60.png
## 2   https://cdn.sofifa.net/teams/21/60.png
## 3   https://cdn.sofifa.net/teams/11/60.png
## 4   https://cdn.sofifa.net/teams/73/60.png
## 5   https://cdn.sofifa.net/teams/10/60.png
## 6  https://cdn.sofifa.net/teams/240/60.png
## 7   https://cdn.sofifa.net/teams/73/60.png
## 8   https://cdn.sofifa.net/teams/21/60.png
## 9  https://cdn.sofifa.net/teams/241/60.png
## 10  https://cdn.sofifa.net/teams/18/60.png
##                              club_flag_url
## 1      https://cdn.sofifa.net/flags/fr.png
## 2      https://cdn.sofifa.net/flags/de.png
## 3  https://cdn.sofifa.net/flags/gb-eng.png
## 4      https://cdn.sofifa.net/flags/fr.png
## 5  https://cdn.sofifa.net/flags/gb-eng.png
## 6      https://cdn.sofifa.net/flags/es.png
## 7      https://cdn.sofifa.net/flags/fr.png
## 8      https://cdn.sofifa.net/flags/de.png
## 9      https://cdn.sofifa.net/flags/es.png
## 10 https://cdn.sofifa.net/flags/gb-eng.png
##                               nation_logo_url
## 1  https://cdn.sofifa.net/teams/1369/60.png
## 2  https://cdn.sofifa.net/teams/1353/60.png
## 3  https://cdn.sofifa.net/teams/1354/60.png
## 4
## 5  https://cdn.sofifa.net/teams/1325/60.png
## 6
## 7  https://cdn.sofifa.net/teams/1335/60.png
## 8  https://cdn.sofifa.net/teams/1337/60.png
## 9
## 10 https://cdn.sofifa.net/teams/1318/60.png
##                            nation_flag_url
## 1      https://cdn.sofifa.net/flags/ar.png
## 2      https://cdn.sofifa.net/flags/pl.png
## 3      https://cdn.sofifa.net/flags/pt.png
## 4      https://cdn.sofifa.net/flags/br.png
## 5      https://cdn.sofifa.net/flags/be.png
## 6      https://cdn.sofifa.net/flags/si.png
## 7      https://cdn.sofifa.net/flags/fr.png
## 8      https://cdn.sofifa.net/flags/de.png
## 9      https://cdn.sofifa.net/flags/de.png
## 10 https://cdn.sofifa.net/flags/gb-eng.png
```

We perform a rough removal of all the features that will obviously not be relevant to our classification, or some of the ones that are a obvious linear composition of other features. Moreover, our training will be performed on the league 1 players. Then, we check the dimensions again.

```
players_full <- players_full[players_full$league_level == 1,]

players_22 <- subset(players_full, select = c("short_name","player_positions","age","potential", "inter
dim(players_22)
```

```
## [1] 14918    45
```

Apparently we kept only 45 features. Good enough. We will remove more later by performing feature selection so stay tuned.

We have a short look at the numerical summary of all the features we selected. On a first glance they look like they need some normalization. But before that, we would love to make some visual presentations.

```
summary(players_22)
```

```
##   short_name        player_positions        age          potential    
##  Length:14918       Length:14918       Min.   :16.00   Min.   :49.00  
##  Class :character   Class :character   1st Qu.:21.00   1st Qu.:67.00  
##  Mode  :character   Mode  :character   Median :25.00   Median :71.00  
##                                        Mean   :25.34   Mean   :71.36  
##                                        3rd Qu.:29.00   3rd Qu.:76.00  
##                                        Max.   :54.00   Max.   :95.00  
##                                        NA's   :61      NA's   :61     
##  international_reputation   value_eur          height_cm     weight_kg     
##  Min.   :1.000           Min.   :     9000   Min.   :155   Min.   : 49.00  
##  1st Qu.:1.000           1st Qu.:   475000   1st Qu.:176   1st Qu.: 70.00  
##  Median :1.000           Median :  1000000   Median :181   Median : 75.00  
##  Mean   :1.116           Mean   :  3341258   Mean   :181   Mean   : 74.84  
##  3rd Qu.:1.000           3rd Qu.:  2400000   3rd Qu.:186   3rd Qu.: 80.00  
##  Max.   :5.000           Max.   :194000000   Max.   :203   Max.   :107.00  
##  NA's   :61              NA's   :71          NA's   :61    NA's   :61      
##       pace          shooting       passing      preferred_foot    
##  Min.   :28.00   Min.   :18.0   Min.   :25.00   Length:14918      
##  1st Qu.:62.00   1st Qu.:42.0   1st Qu.:51.00   Class :character  
##  Median :69.00   Median :55.0   Median :58.00   Mode  :character  
##  Mean   :68.33   Mean   :52.8   Mean   :57.88                     
##  3rd Qu.:76.00   3rd Qu.:64.0   3rd Qu.:65.00                     
##  Max.   :97.00   Max.   :94.0   Max.   :93.00                     
##  NA's   :1725    NA's   :1725   NA's   :1725                      
##    weak_foot       dribbling       defending         physic     
##  Min.   :1.000   Min.   :27.00   Min.   :15.00   Min.   :29.00  
##  1st Qu.:3.000   1st Qu.:57.00   1st Qu.:38.00   1st Qu.:59.00  
##  Median :3.000   Median :64.00   Median :56.00   Median :66.00  
##  Mean   :2.948   Mean   :62.99   Mean   :52.03   Mean   :64.89  
##  3rd Qu.:3.000   3rd Qu.:70.00   3rd Qu.:65.00   3rd Qu.:72.00  
##  Max.   :5.000   Max.   :95.00   Max.   :91.00   Max.   :90.00  
##  NA's   :61      NA's   :1725    NA's   :1725    NA's   :1725   
##  attacking_crossing attacking_finishing attacking_heading_accuracy
##  Min.   : 6         Min.   : 2.0        Min.   : 5.00             
##  1st Qu.:39         1st Qu.:31.0        1st Qu.:44.00             
##  Median :54         Median :50.0        Median :55.00             
##  Mean   :50         Mean   :46.2        Mean   :51.95             
##  3rd Qu.:64         3rd Qu.:62.0        3rd Qu.:64.00             
##  Max.   :94         Max.   :95.0        Max.   :93.00
##  NA's   :61         NA's   :61          NA's   :61
##  attacking_short_passing attacking_volleys skill_dribbling  skill_curve
##  Min.   : 7.00           Min.   : 3.00     Min.   : 4      Min.   : 6.00
##  1st Qu.:55.00           1st Qu.:30.00     1st Qu.:50      1st Qu.:35.00
##  Median :63.00           Median :44.00     Median :62      Median :49.00
##  Mean   :59.33           Mean   :42.89     Mean   :56      Mean   :47.73
##  3rd Qu.:69.00           3rd Qu.:57.00     3rd Qu.:69      3rd Qu.:62.00
##  Max.   :94.00           Max.   :90.00     Max.   :96      Max.   :94.00
##  NA's   :61              NA's   :61        NA's   :61      NA's   :61
##  skill_fk_accuracy skill_long_passing skill_ball_control movement_acceleration
##  Min.   : 4.00     Min.   : 9.00      Min.   : 8.00      Min.   :14.0
##  1st Qu.:31.00     1st Qu.:45.00      1st Qu.:55.00      1st Qu.:58.0
##  Median :41.00     Median :57.00      Median :63.00      Median :68.0
##  Mean   :42.65     Mean   :53.63      Mean   :58.88      Mean   :64.7
##  3rd Qu.:56.00     3rd Qu.:65.00      3rd Qu.:70.00      3rd Qu.:75.0
##  Max.   :94.00     Max.   :93.00      Max.   :96.00      Max.   :97.0
##  NA's   :61        NA's   :61         NA's   :61         NA's   :61
##  movement_sprint_speed movement_agility movement_reactions movement_balance
##  Min.   :15.00         Min.   :18.00    Min.   :25.00      Min.   :19.0
##  1st Qu.:58.00         1st Qu.:55.00    1st Qu.:56.00      1st Qu.:56.0
##  Median :68.00         Median :66.00    Median :62.00      Median :66.0
##  Mean   :64.77         Mean   :63.55    Mean   :61.91      Mean   :64.1
##  3rd Qu.:75.00         3rd Qu.:74.00    3rd Qu.:68.00      3rd Qu.:74.0
##  Max.   :97.00         Max.   :96.00    Max.   :94.00      Max.   :96.0
##  NA's   :61            NA's   :61       NA's   :61         NA's   :61
##  power_shot_power power_jumping   power_stamina   power_strength
##  Min.   :20.00    Min.   :24.00   Min.   :12.00   Min.   :19.00
##  1st Qu.:48.00    1st Qu.:57.00   1st Qu.:56.00   1st Qu.:57.00
##  Median :59.00    Median :65.00   Median :67.00   Median :66.00
##  Mean   :58.19    Mean   :64.75   Mean   :63.15   Mean   :64.97
##  3rd Qu.:68.00    3rd Qu.:73.00   3rd Qu.:74.00   3rd Qu.:74.00
##  Max.   :95.00    Max.   :95.00   Max.   :97.00   Max.   :96.00
##  NA's   :61       NA's   :61      NA's   :61      NA's   :61
##  power_long_shots mentality_aggression mentality_interceptions
##  Min.   : 4.00    Min.   :10.00        Min.   : 4.00
##  1st Qu.:32.00    1st Qu.:45.00        1st Qu.:26.00
##  Median :51.00    Median :59.00        Median :53.00
##  Mean   :47.08    Mean   :55.85        Mean   :46.95
##  3rd Qu.:63.00    3rd Qu.:69.00        3rd Qu.:64.00
##  Max.   :94.00    Max.   :95.00        Max.   :91.00
##  NA's   :61       NA's   :61           NA's   :61
##  mentality_positioning mentality_vision mentality_penalties mentality_composure
##  Min.   : 2.00         Min.   :10.00    Min.   : 7.00       Min.   :12.0
##  1st Qu.:40.00         1st Qu.:45.00    1st Qu.:38.00       1st Qu.:50.0
##  Median :56.00         Median :56.00    Median :49.00       Median :59.0
##  Mean   :50.76         Mean   :54.49    Mean   :48.11       Mean   :58.4
##  3rd Qu.:65.00         3rd Qu.:65.00    3rd Qu.:60.00       3rd Qu.:67.0
##  Max.   :96.00         Max.   :95.00    Max.   :93.00       Max.   :96.0
##  NA's   :61            NA's   :61       NA's   :61          NA's   :61
##  defending_marking_awareness defending_standing_tackle defending_sliding_tackle
##  Min.   : 4.00               Min.   : 5.00             Min.   : 5.00
##  1st Qu.:29.00               1st Qu.:28.00             1st Qu.:26.00
##  Median :52.00               Median :55.00             Median :53.00
##  Mean   :46.86              Mean   :48.28            Mean   :46.12
##  3rd Qu.:64.00              3rd Qu.:66.00            3rd Qu.:64.00
##  Max.   :93.00              Max.   :93.00            Max.   :92.00
##  NA's   :61                 NA's   :61               NA's   :61
```

2.2 Managing empty entries

We look at how many NAs we have on each attribute, in order to decide if we prefer removing them or filling them.

```
which(apply(X = players_22, MARGIN = 2, FUN = anyNA) == TRUE) # check for NA
```

```
##                  short_name            player_positions
##                           1                           2
##                         age                   potential
##                           3                           4
##    international_reputation                   value_eur
##                           5                           6
##                   height_cm                   weight_kg
##                           7                           8
##                        pace                    shooting
##                           9                          10
##                     passing              preferred_foot
##                          11                          12
##                   weak_foot                   dribbling
##                          13                          14
##                   defending                      physic
##                          15                          16
##          attacking_crossing          attacking_finishing
##                          17                          18
##  attacking_heading_accuracy     attacking_short_passing
##                          19                          20
##           attacking_volleys              skill_dribbling
##                          21                          22
##                 skill_curve            skill_fk_accuracy
##                          23                          24
##          skill_long_passing           skill_ball_control
##                          25                          26
##       movement_acceleration        movement_sprint_speed
##                          27                          28
##            movement_agility           movement_reactions
##                          29                          30
##            movement_balance             power_shot_power
##                          31                          32
##               power_jumping                power_stamina
##                          33                          34
##              power_strength             power_long_shots
##                          35                          36
##        mentality_aggression     mentality_interceptions
##                          37                          38
##       mentality_positioning             mentality_vision
##                          39                          40
##         mentality_penalties          mentality_composure
##                            41                            42
## defending_marking_awareness   defending_standing_tackle
##                            43                            44
##    defending_sliding_tackle
##                            45
```

We decide that we have a statistically dispensable number of NAs so we remove them.

```
players <- na.omit(players_22) # delete NA
dim(players)
```

```
## [1] 13183    45
```

We still have a good chunk of the dataset left. Since goalkeepers have special stats, we also would like to take them out. Given the fact that they are not movement players they may be missing one or more attributes typical of a footballer that plays outside, such shot precision, dribbling or finishing, for this reason they may have already been discarded in the previous phase. Just to be sure, let's verify.

```
goalkeepers <- str_count(players$player_positions, "GK")
sum(goalkeepers)
```

```
## [1] 0
```

Indeed, there are no goalkeeper left. Thus, while they are indisposable on the field, we could not say the same about their data, as it would reduce the accuracy of the classification of the other main positions.

2.3 Subset creation

We create two subsets: one with the market value in euro of the player, that we will use for the regression on the market price (we will call this subset 'players_regress') and a second subset that does not contain such attribute, since it is not useful for the position classification that we will perform with that.

```
#subset WITH market value (for later)

players_copy <- as.data.frame(copy(players))

players_regress <-subset(players_copy, player_positions!="GK")

#subset WITHOUT market value
players_class <-subset(players_copy, player_positions!="GK", select = -value_eur)
```

2.4 Labelling

Some players play in multiple positions, but we only want to identify their main one, so we only keep that one. Moreover, we turn the binary "preferred_foot" feature into a numerical type.

```
#Keep only the main preferred position
players_class$player_positions<- word(players_class$player_positions, 1, sep = fixed(","))
unique(players_class$player_positions)
```

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

```
# Left foot is -1 and Right foot is 1. Basically one-hot encoding but we only have 2 categories so its

players_class$preferred_foot[players_class[,"preferred_foot"]== "Left"] <- as.numeric(-1)
players_class$preferred_foot[players_class[,"preferred_foot"]== "Right"] <- as.numeric(1)
players_class$preferred_foot <- as.numeric(players_class$preferred_foot)
```

We can see some of the positions available on the game. Goalkeeper excluded, there are 26 positions, namely:

1. LWB = Left Wing Back
2. LB = Left Back
3. LCB = Left Center Back
4. CB = Center Back
5. RCB = Right Center Back
6. RB = Right Back
7. RWB = Right Wing Back
8. LDM = Left Defensive Midfield
9. CDM = Center Defensive Midfield
10. RDM = Right Defensive Midfield
11. RCM = Right Center Midfield
12. CM = Center Midfield
13. LCM = Left Center Midfield
14. RAM = Right Attacking Midfield
15. CAM = Center Attacking Midfield
16. LAM = Left Attacking Midfield
17. LM = Left Midfield
18. RM = Right Midfield
19. LW = Left Winger
20. RW = Right Winger
21. LF = Left Forward
22. CF = Center Forward
23. RF = Right Striker
24. LS = Left Striker
25. ST = Striker
26. RS = Right Striker

Since 26 labels positions are clearly too many, we cluster them into nine classes of positions based on the area of action on the field.

Note: Here we apply our "domain knowledge".

Below a picture where we can visually see how we are going to group the various positions to obtain the nine classes desired.

```
#central back
players_class$player_positions[players_class[,"player_positions"]== "LCB"|players_class[,"player_positi

#left back
players_class$player_positions[players_class[,"player_positions"]== "LWB"|players_class[,"player_positi

#right back
players_class$player_positions[players_class[,"player_positions"]== "RWB"|players_class[,"player_positi
```

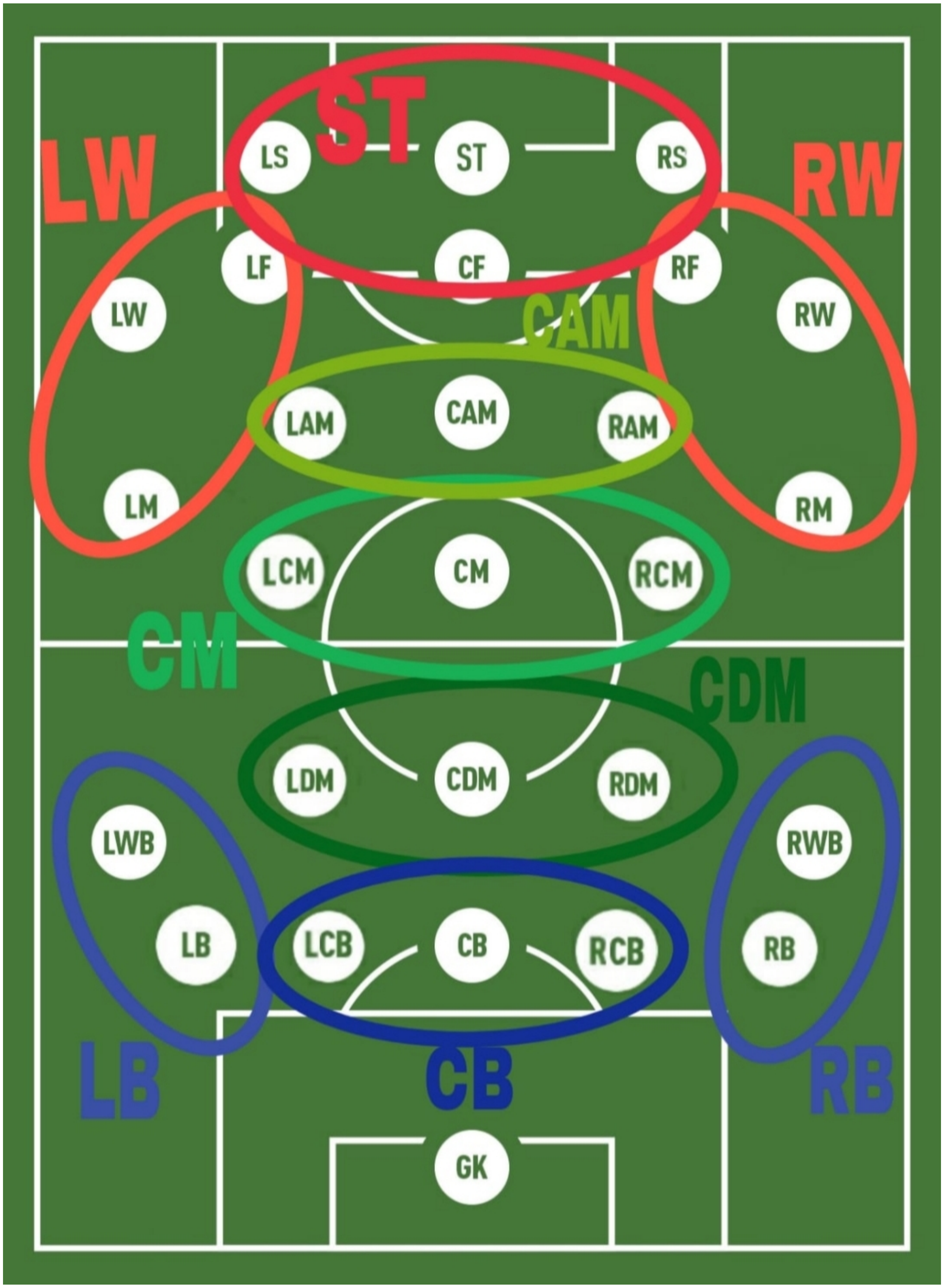
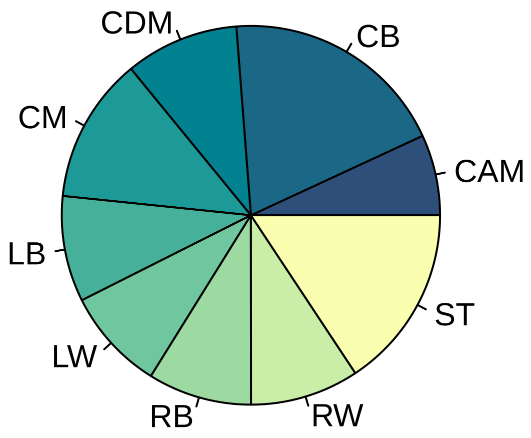


Figure 1: field positions

```
#central deffensive midfielder
players_class$player_positions[players_class[,"player_positions"]== "LDM"|players_class[,"player_positi

#central midfielder
players_class$player_positions[players_class[,"player_positions"]== "LCM"|players_class[,"player_positi

#central attacking midfielder
players_class$player_positions[players_class[,"player_positions"]== "LAM"|players_class[,"player_positi

#left winger
players_class$player_positions[players_class[,"player_positions"]== "LM"|players_class[,"player_positio

#right winger
players_class$player_positions[players_class[,"player_positions"]== "RM"|players_class[,"player_positio

#striker
players_class$player_positions[players_class[,"player_positions"]== "LS"|players_class[,"player_positio
```

Lets take a look at the distribution of our labels.

```
cat<- table(factor(players_class$player_positions))
pie(cat,
    col = hcl.colors(length(cat), "BluYl"))
```



Time to normalize the numerical values, as promised. For that, we implement a simple re-scaling function, and we apply it on the whole dataframe.

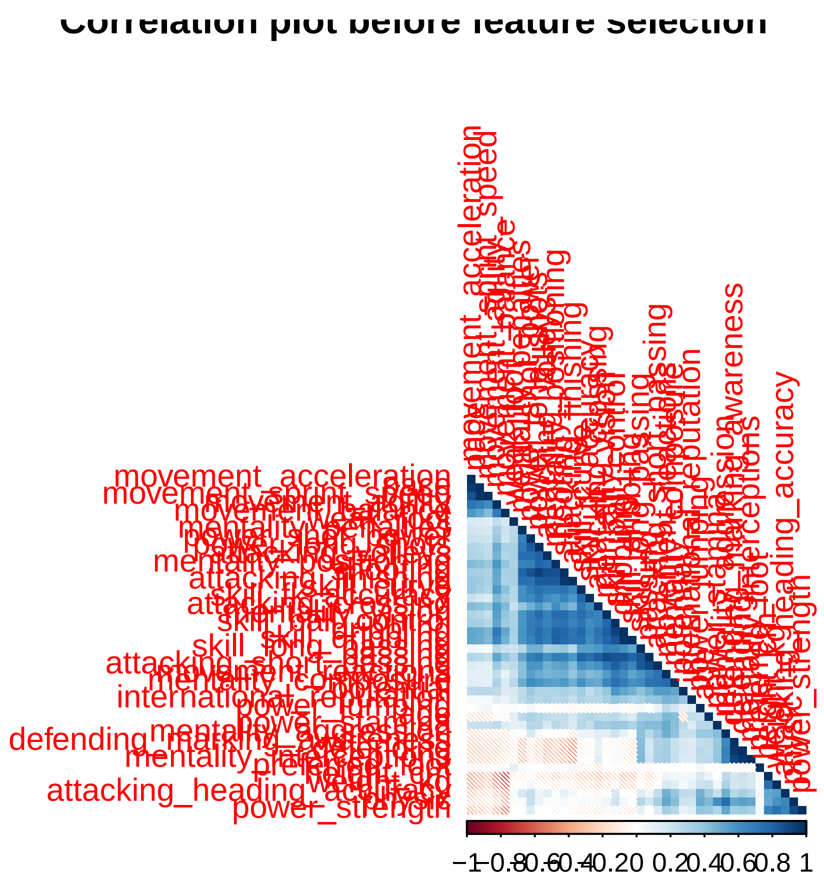
```
# normalization function
normalize <-function(x) { (x -min(x))/(max(x)-min(x))   }

# normalize
players_norm <- as.data.frame(lapply(players_class[, c(3:42)], normalize))
head(players_norm,5)
```

```
##         age potential international_reputation height_cm weight_kg      pace
## 1 0.7826087 0.9565217                     1.00 0.3125000 0.4423077 0.8260870
## 2 0.6956522 0.9347826                     1.00 0.6250000 0.6153846 0.7246377
## 3 0.8695652 0.9130435                     1.00 0.6666667 0.6538462 0.8550725
## 4 0.5652174 0.9130435                     1.00 0.4166667 0.3653846 0.9130435
## 5 0.6086957 0.9130435                     0.75 0.5416667 0.4038462 0.6956522
##    shooting   passing preferred_foot weak_foot dribbling defending    physic
## 1 0.9736842 0.9705882              0      0.75 1.0000000 0.2500000 0.5901639
## 2 0.9736842 0.7941176              1      0.75 0.8676471 0.3815789 0.8688525
## 3 1.0000000 0.8088235              1      0.75 0.8970588 0.2500000 0.7540984
## 4 0.8552632 0.8970588              1      1.00 0.9852941 0.2894737 0.5573770
## 5 0.8947368 1.0000000              1      1.00 0.8970588 0.6447368 0.8032787
##   attacking_crossing attacking_finishing attacking_heading_accuracy
## 1          0.8860759           1.0000000                  0.6973684
## 2          0.7088608           1.0000000                  0.9605263
## 3          0.9113924           1.0000000                  0.9605263
## 4          0.8860759           0.8588235                  0.6052632
## 5          1.0000000           0.8470588                  0.5000000
##   attacking_short_passing attacking_volleys skill_dribbling skill_curve
## 1               0.9577465            0.9750       1.0000000   0.9878049
## 2               0.8732394            0.9875       0.8589744   0.8170732
## 3               0.8028169            0.9500       0.8974359   0.8414634
## 4               0.8873239            0.9500       0.9871795   0.9268293
## 5               1.0000000            0.9000       0.8974359   0.8902439
##   skill_fk_accuracy skill_long_passing skill_ball_control movement_acceleration
## 1         1.0000000          0.9726027          1.0000000             0.9142857
## 2         0.8928571          0.6849315          0.8888889             0.7142857
## 3         0.8809524          0.7808219          0.8888889             0.8285714
## 4         0.9166667          0.8356164          0.9861111             0.9428571
## 5         0.8690476          1.0000000          0.9305556             0.7000000
##   movement_sprint_speed movement_agility movement_reactions movement_balance
## 1             0.7571429        0.9275362          1.0000000        0.9857143
## 2             0.7428571        0.7246377          0.9846154        0.8000000
## 3             0.8714286        0.8550725          1.0000000        0.6857143
## 4             0.8857143        1.0000000          0.9230769        0.8285714
## 5             0.7000000        0.7536232          0.9538462        0.7428571
##   power_shot_power power_jumping power_stamina power_strength power_long_shots
## 1        0.8800000     0.5909091     0.6478873      0.6493506        1.0000000
## 2        0.9333333     0.8484848     0.7042254      0.8701299        0.9156627
## 3        0.9866667     1.0000000     0.7183099      0.7532468        0.9879518
## 4        0.8000000     0.5303030     0.7746479      0.4415584        0.8433735
## 5        0.9466667     0.5151515     0.8873239      0.7142857        0.9638554
##   mentality_aggression mentality_interceptions mentality_positioning
## 1            0.3200000               0.3703704             0.9642857
## 2            0.8133333               0.4814815             0.9880952
## 3           0.5733333              0.2345679             0.9880952
## 4           0.5733333              0.3333333             0.8809524
## 5           0.7466667              0.6913580             0.9047619
##   mentality_vision mentality_penalties mentality_composure
## 1        1.0000000              0.7750           1.0000000
## 2        0.8292683              0.9625           0.8787879
## 3        0.7682927              0.9375           0.9848485
## 4        0.9390244              1.0000           0.9545455
## 5        0.9878049              0.8750           0.8939394
##   defending_marking_awareness
## 1                   0.1204819
## 2                   0.3012048
## 3                   0.1686747
## 4                   0.3012048
## 5                   0.6987952
```

2.5 Correlation matrix and feature selection

We create a correlation matrix. It is big and maybe a bit hard to read, but R gives us the visually appealing option to group plotted features into highly correlated clusters.

```
cormatrix <- cor(players_norm)
corrplot(cor(players_norm), method = 'shade', sig.level = 0.10, type = 'lower', order = 'hclust', title
```



Now, in order to reduce the number of features, we take away the ones that provide the data with the highest overall correlation.

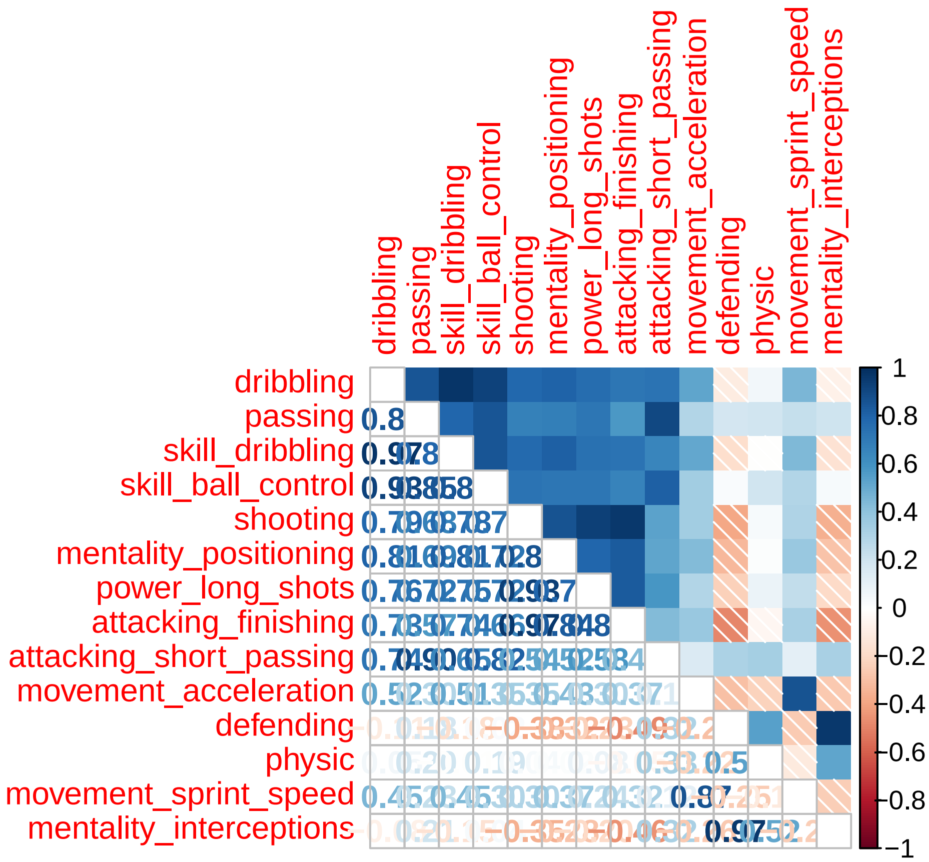
```r
highcorr <- findCorrelation(cormatrix, cutoff=0.8)
highcorr
```

```
##  [1] 11  8 19 23  7 36 33 15 17 24 12 13 25 35
```

```r
col2<-colnames(players_norm)
col2
```

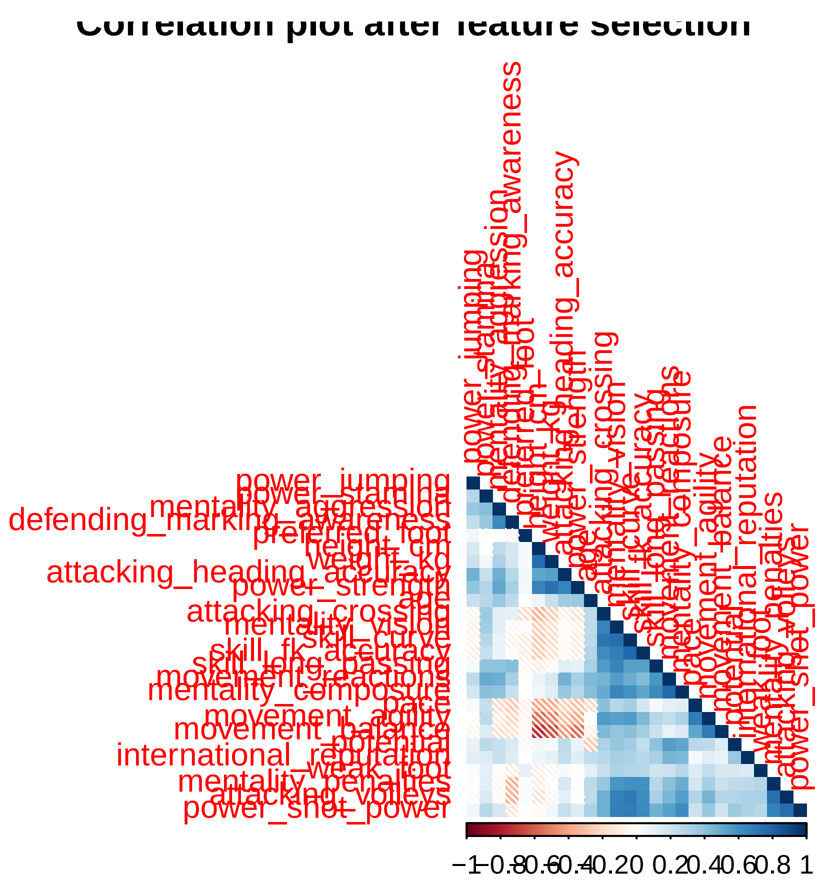
```
##  [1] "age"                         "potential"
##  [3] "international_reputation"    "height_cm"
##  [5] "weight_kg"                   "pace"
##  [7] "shooting"                    "passing"
##  [9] "preferred_foot"              "weak_foot"
## [11] "dribbling"                   "defending"
## [13] "physic"                      "attacking_crossing"
## [15] "attacking_finishing"         "attacking_heading_accuracy"
## [17] "attacking_short_passing"     "attacking_volleys"
## [19] "skill_dribbling"             "skill_curve"
## [21] "skill_fk_accuracy"           "skill_long_passing"
## [23] "skill_ball_control"          "movement_acceleration"
## [25] "movement_sprint_speed"       "movement_agility"
## [27] "movement_reactions"          "movement_balance"
## [29] "power_shot_power"            "power_jumping"
## [31] "power_stamina"               "power_strength"
## [33] "power_long_shots"            "mentality_aggression"
## [35] "mentality_interceptions"     "mentality_positioning"
## [37] "mentality_vision"            "mentality_penalties"
## [39] "mentality_composure"         "defending_marking_awareness"
```

```r
col2<-col2[-highcorr]
corrplot.mixed(cor(players_norm[highcorr]), lower = "number", upper="shade", tl.pos = 'lt')
```



We take a look if we eliminated some of the dark spots from our correlation matrix.

```
corrplot(cor(players_norm[col2]), type = 'lower',method = 'shade', order = 'hclust', title = "Correlati
```



```
players_f.selected <- players_norm[col2]  #features left
players_model <- subset(players_f.selected)

#we can add the positions back
players_model$player_positions <- c(players_class$player_positions)
```

```
colnames(players_model)
```

```
##  [1] "age"                          "potential"
##  [3] "international_reputation"     "height_cm"
##  [5] "weight_kg"                    "pace"
##  [7] "preferred_foot"               "weak_foot"
##  [9] "attacking_crossing"           "attacking_heading_accuracy"
## [11] "attacking_volleys"            "skill_curve"
## [13] "skill_fk_accuracy"            "skill_long_passing"
## [15] "movement_agility"             "movement_reactions"
## [17] "movement_balance"             "power_shot_power"
## [19] "power_jumping"                "power_stamina"
## [21] "power_strength"               "mentality_aggression"
## [23] "mentality_vision"             "mentality_penalties"
## [25] "mentality_composure"          "defending_marking_awareness"
## [27] "player_positions"
```

We did. Looks much better and ready for further investigation.

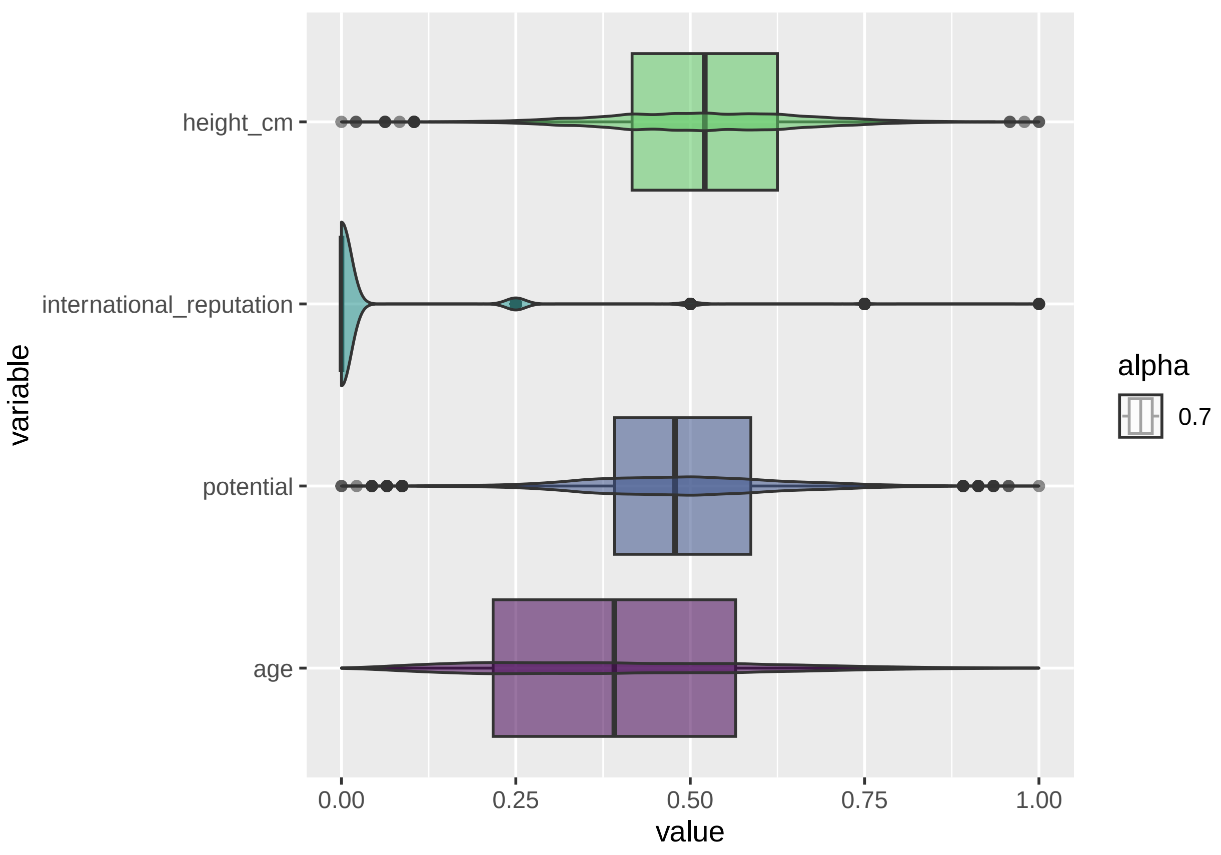
2.6 Individual feature investigation

We want to look at the individual distributions of each of the features left. We fit violin plots, and put boxplots on top of them.

```
# here we do the cool violin plots to check distributions

# set up the plotting layout as a 4x2 grid
par(mfrow = c(4, 2))

# create a combined boxplot and violin plot using ggplot2
ggplot(data = melt(players_f.selected[, 1:4]),  # convert data to long format (tabular form)
       aes(y = variable, x = value, fill = variable, alpha = 0.7)) +  # specify aesthetics
  geom_boxplot() +  # add boxplot layer
  geom_violin() +  # add violin plot layer
  scale_fill_manual(values = viridis(5)) +  # set fill colors using Viridis palette
  guides(fill = "none")  # remove legend for fill colors
```

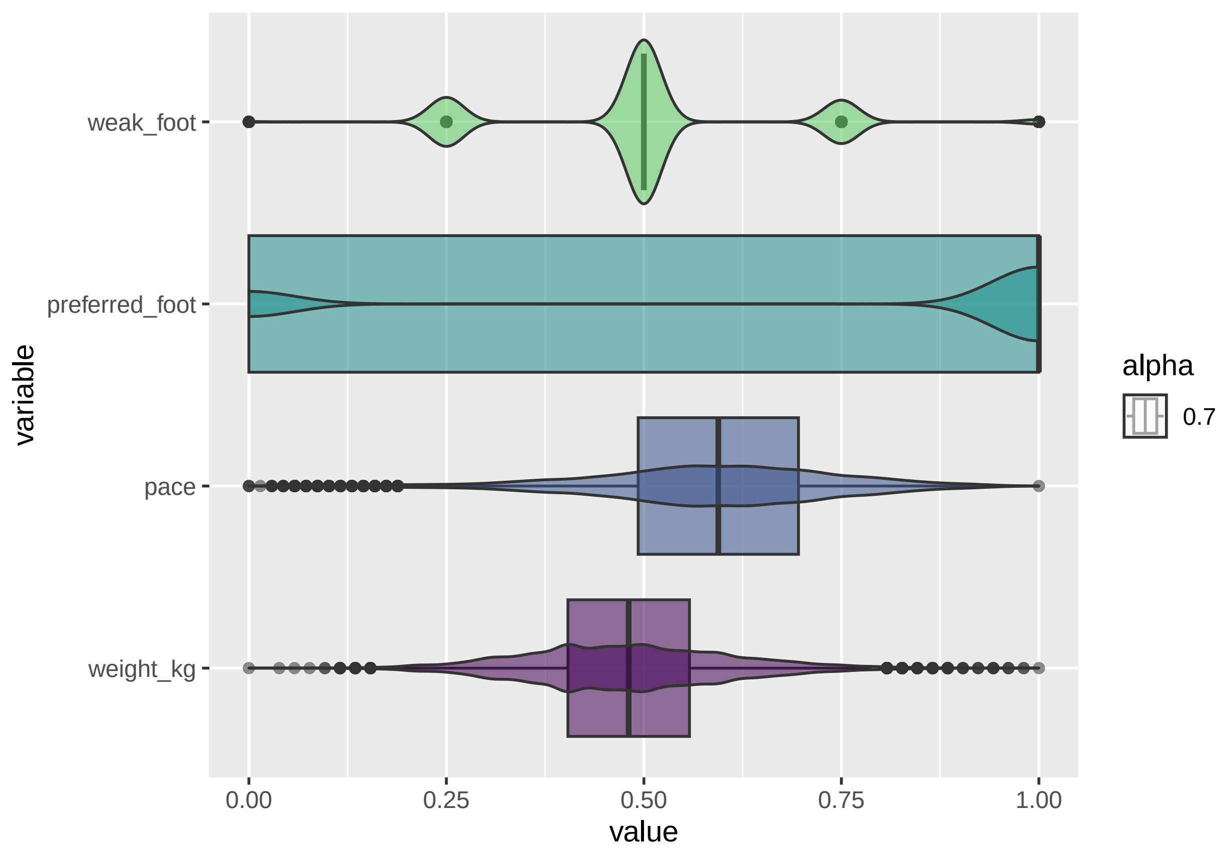
```
## Warning in melt(players_f.selected[, 1:4]): The melt generic in data.table has
## been passed a data.frame and will attempt to redirect to the relevant reshape2
## method; please note that reshape2 is deprecated, and this redirection is now
## deprecated as well. To continue using melt methods from reshape2 while both
## libraries are attached, e.g. melt.list, you can prepend the namespace like
## reshape2::melt(players_f.selected[, 1:4]). In the next version, this warning
## will become an error.
```

```
## No id variables; using all as measure variables
```



```
ggplot(data = melt(players_f.selected[,5:8]), aes(y = variable, x = value, fill = variable, alpha = 0.7
```

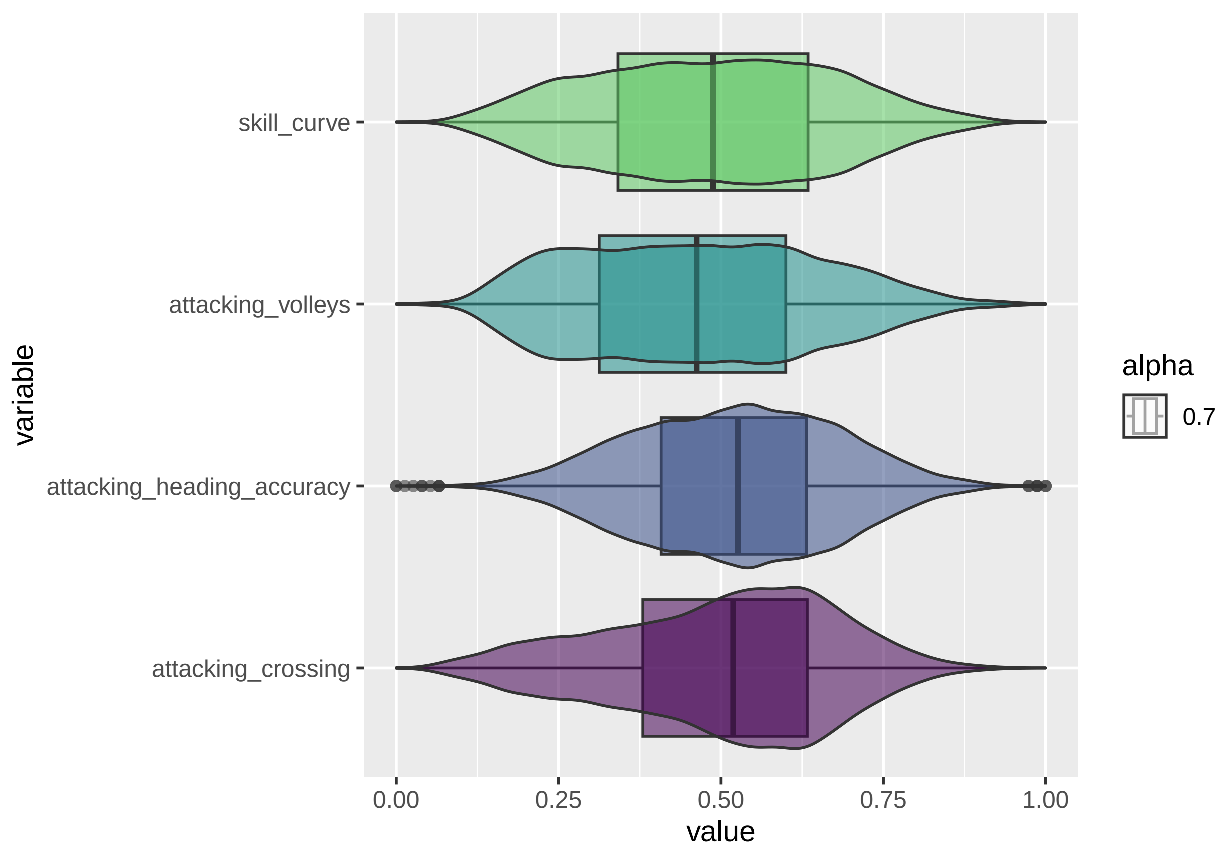
```
## Warning in melt(players_f.selected[, 5:8]): The melt generic in data.table has
## been passed a data.frame and will attempt to redirect to the relevant reshape2
## method; please note that reshape2 is deprecated, and this redirection is now
## deprecated as well. To continue using melt methods from reshape2 while both
## libraries are attached, e.g. melt.list, you can prepend the namespace like
## reshape2::melt(players_f.selected[, 5:8]). In the next version, this warning
## will become an error.

## No id variables; using all as measure variables
```



Weak foot is a discrete RV with values in 1-5. Preferred foot is +/-1, as discussed above. Still, as in real life, a significantly larger proportion of right-footed people.

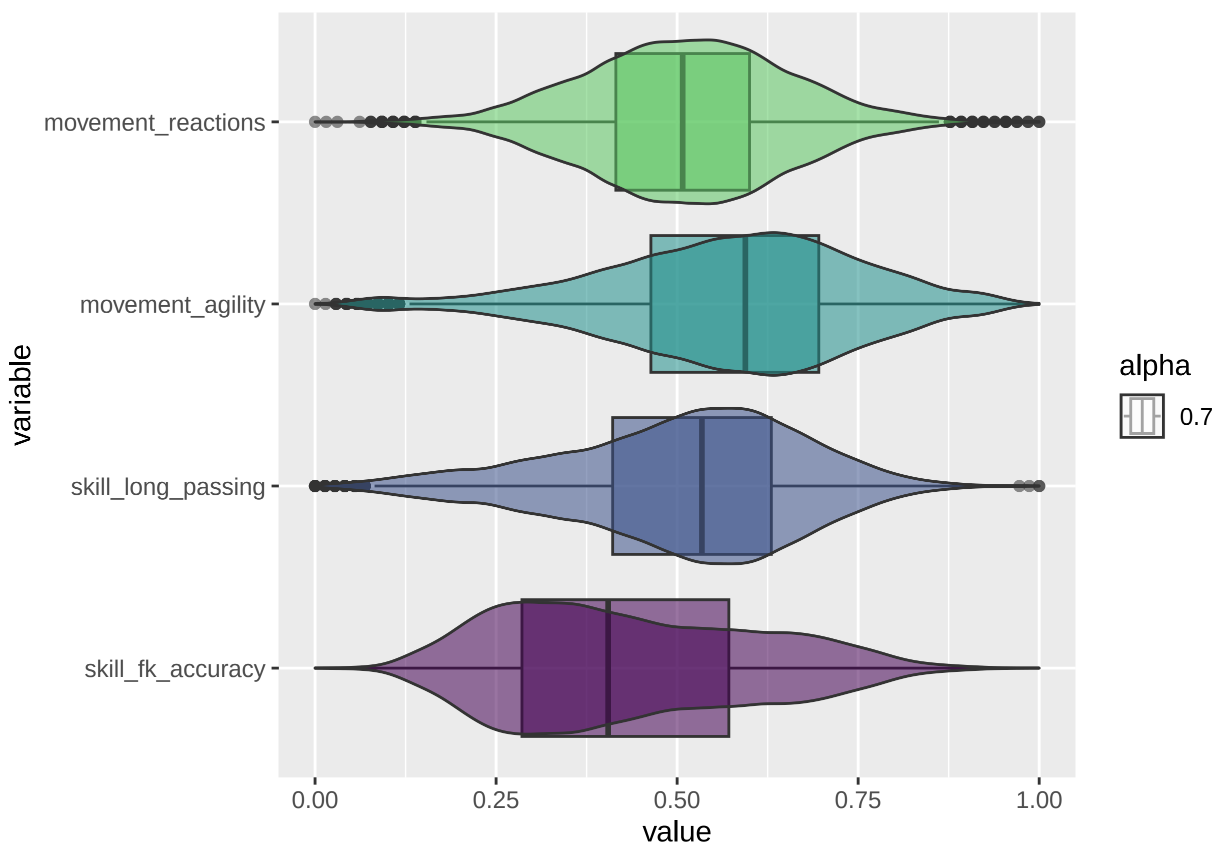
```
ggplot(data = melt(players_f.selected[,9:12]), aes(y = variable, x = value, fill = variable, alpha = 0.
```

```
## Warning in melt(players_f.selected[, 9:12]): The melt generic in data.table has
## been passed a data.frame and will attempt to redirect to the relevant reshape2
## method; please note that reshape2 is deprecated, and this redirection is now
## deprecated as well. To continue using melt methods from reshape2 while both
## libraries are attached, e.g. melt.list, you can prepend the namespace like
## reshape2::melt(players_f.selected[, 9:12]). In the next version, this warning
## will become an error.

## No id variables; using all as measure variables
```



```
ggplot(data = melt(players_f.selected[,13:16]), aes(y = variable, x = value, fill = variable, alpha = 0
```

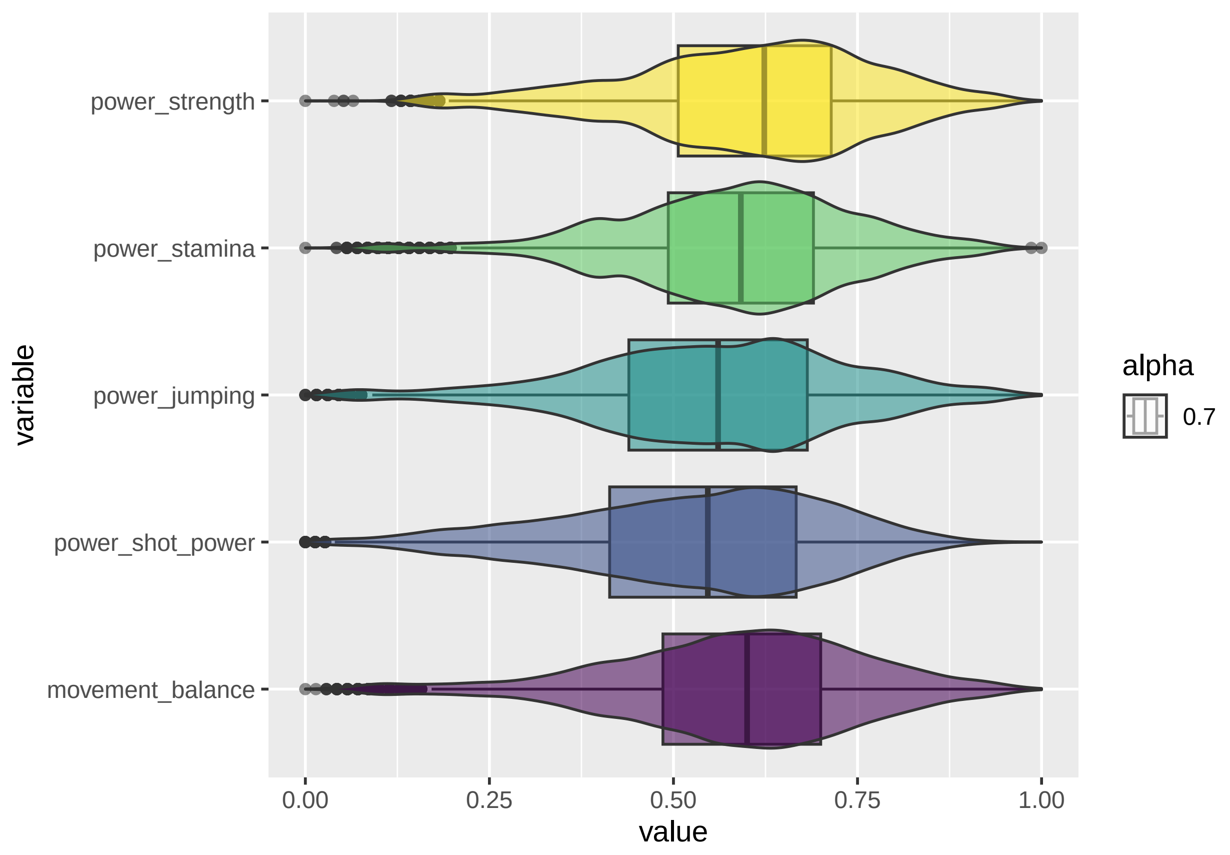
```
## Warning in melt(players_f.selected[, 13:16]): The melt generic in data.table
## has been passed a data.frame and will attempt to redirect to the relevant
## reshape2 method; please note that reshape2 is deprecated, and this redirection
## is now deprecated as well. To continue using melt methods from reshape2 while
## both libraries are attached, e.g. melt.list, you can prepend the namespace like
## reshape2::melt(players_f.selected[, 13:16]). In the next version, this warning
## will become an error.

## No id variables; using all as measure variables
```



```
ggplot(data = melt(players_f.selected[,17:21]), aes(y = variable, x = value, fill = variable, alpha = 0
```

```
## Warning in melt(players_f.selected[, 17:21]): The melt generic in data.table
## has been passed a data.frame and will attempt to redirect to the relevant
## reshape2 method; please note that reshape2 is deprecated, and this redirection
## is now deprecated as well. To continue using melt methods from reshape2 while
## both libraries are attached, e.g. melt.list, you can prepend the namespace like
## reshape2::melt(players_f.selected[, 17:21]). In the next version, this warning
## will become an error.
```

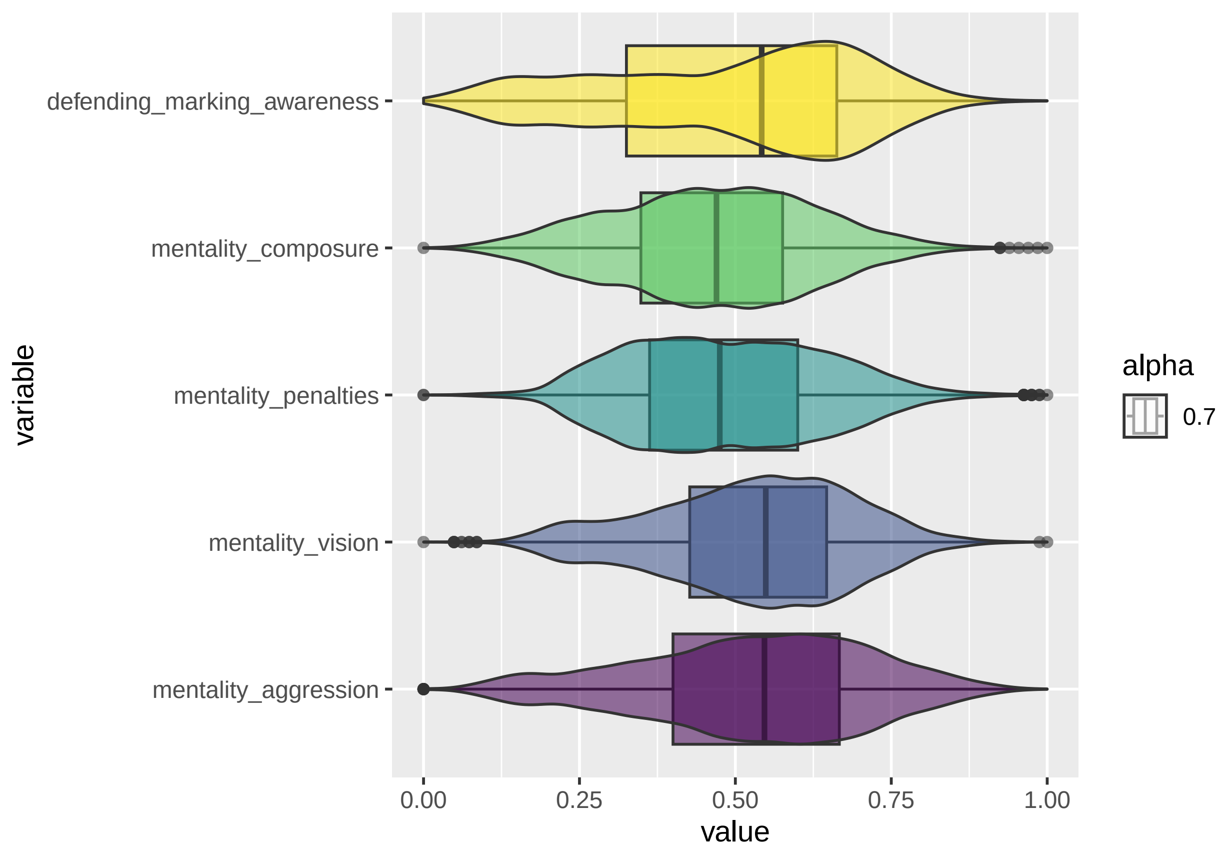
```
## No id variables; using all as measure variables
```



```
ggplot(data = melt(players_f.selected[,22:26]), aes(y = variable, x = value, fill = variable, alpha = 0
```

```
## Warning in melt(players_f.selected[, 22:26]): The melt generic in data.table
## has been passed a data.frame and will attempt to redirect to the relevant
## reshape2 method; please note that reshape2 is deprecated, and this redirection
## is now deprecated as well. To continue using melt methods from reshape2 while
## both libraries are attached, e.g. melt.list, you can prepend the namespace like
## reshape2::melt(players_f.selected[, 22:26]). In the next version, this warning
## will become an error.
```

```
## No id variables; using all as measure variables
```



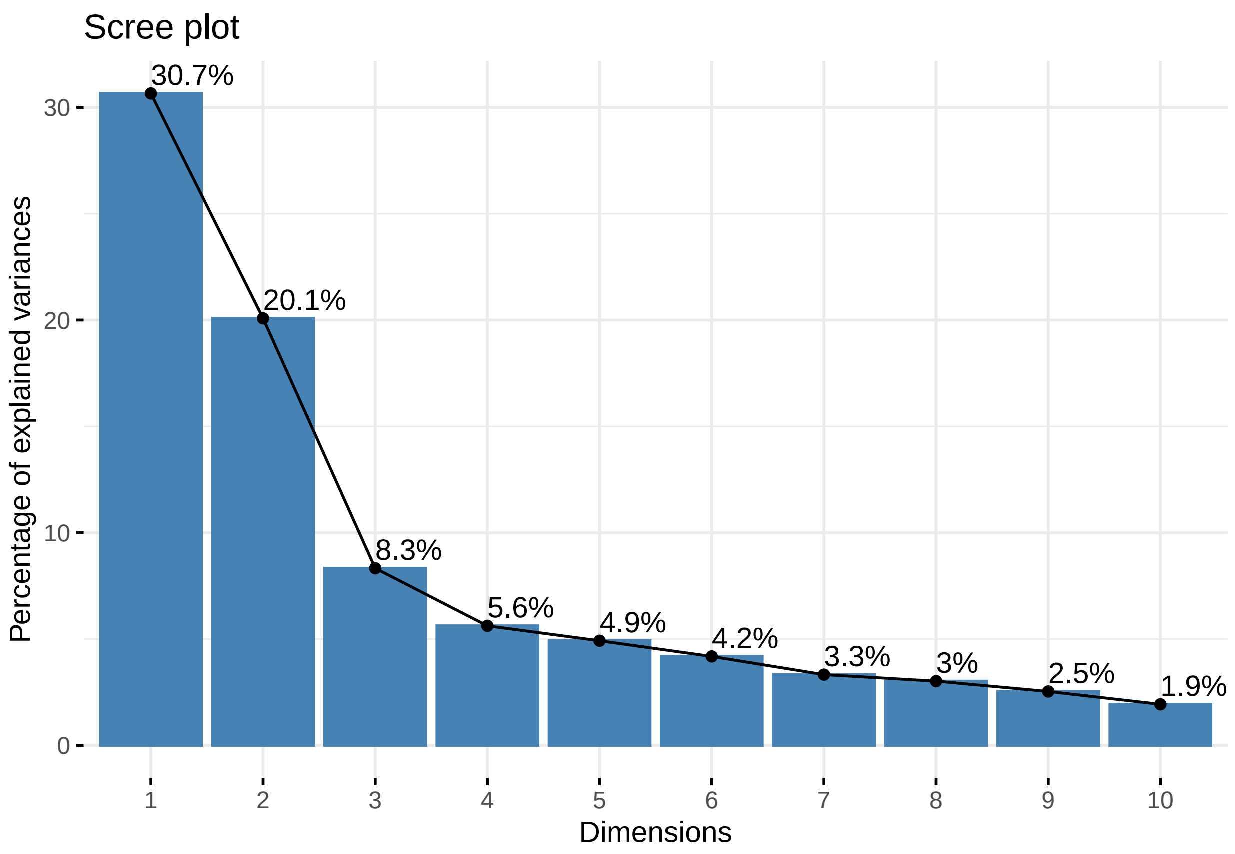
### 2.7 Principal Component Analysis

```
players.pca<-prcomp(players_f.selected,center=TRUE, scale.=TRUE)
summary(players.pca)
```

```
## Importance of components:
##                           PC1    PC2     PC3     PC4     PC5     PC6     PC7
## Standard deviation     2.8232 2.2849 1.47152 1.20892 1.13086 1.04265 0.92988
## Proportion of Variance 0.3066 0.2008 0.08328 0.05621 0.04919 0.04181 0.03326
## Cumulative Proportion  0.3066 0.5074 0.59064 0.64685 0.69604 0.73785 0.77111
##                           PC8     PC9   PC10    PC11    PC12    PC13    PC14
## Standard deviation     0.8861 0.81169 0.7084 0.67291 0.63723 0.61597 0.56246
## Proportion of Variance 0.0302 0.02534 0.0193 0.01742 0.01562 0.01459 0.01217
## Cumulative Proportion  0.8013 0.82665 0.8459 0.86336 0.87898 0.89358 0.90574
##                           PC15    PC16    PC17    PC18   PC19    PC20    PC21
## Standard deviation     0.52867 0.51801 0.50465 0.48131 0.4700 0.45150 0.43178
## Proportion of Variance 0.01075 0.01032 0.00979 0.00891 0.0085 0.00784 0.00717
## Cumulative Proportion  0.91649 0.92681 0.93661 0.94552 0.9540 0.96186 0.96903
##                          PC22    PC23    PC24    PC25    PC26
## Standard deviation     0.4265 0.41564 0.40683 0.39030 0.36437
## Proportion of Variance 0.0070 0.00664 0.00637 0.00586 0.00511
## Cumulative Proportion  0.9760 0.98267 0.98903 0.99489 1.00000
```

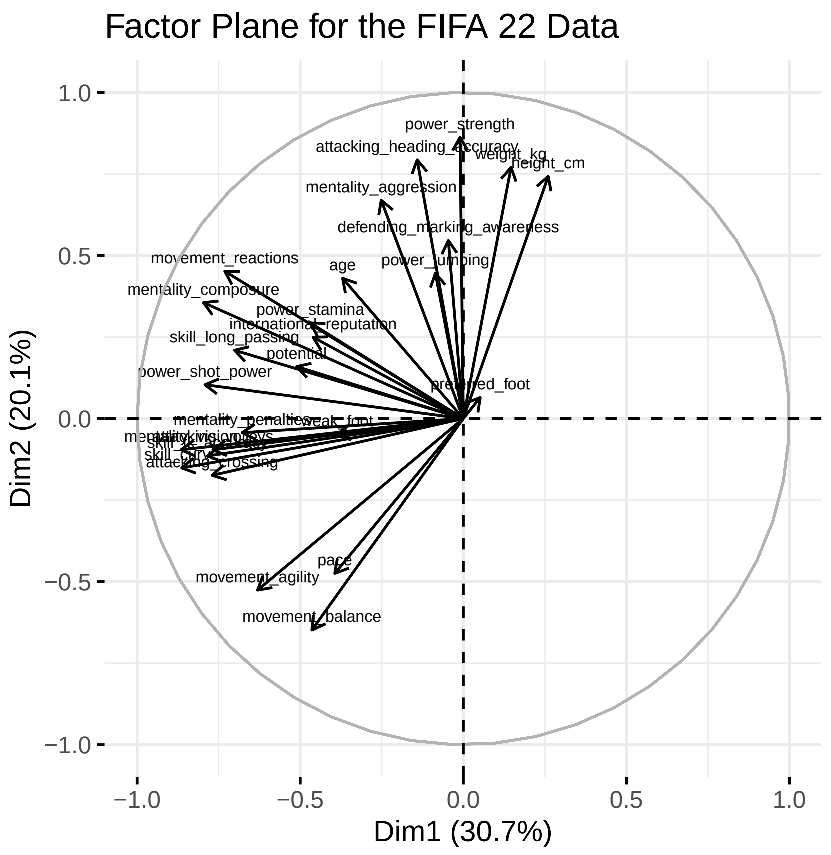
We obtain 26 components. We want to visualize them.

```
fviz_eig(players.pca, addlabels = TRUE)
```



The first 5 components account for 69.6% of the explained variance, while the first 2 for 50.8% of it. Now we want to see how our features project into the main 2D factor plane.

```
fviz_pca_var(players.pca, labelsize = 2, alpha.var = 1.0, title = "Factor Plane for the FIFA 22 Data")
```



From the factor plane we can appreciate the relationship among particular attributes. Pace is positively correlated with agility and balance, which makes absolute sense since agility and a good balance are the key elements of the fastest players on the field. Another group of highly correlated features is the one that comprehends the physical attributes and aerial game: the taller and heaviest guys are usually the strongest ones and given their physical structure they are more likely to be also good headers. All these characteristics are the more desirable for a central defender (CB) and this is the reason why also aggression and marking awareness belong to the group.

3. Modelling - Multiclass classification

Now its finally time to dive into the actual modelling process. We experiment and compare different classification algorithms.

3.1 Train-validation split

Classical split for training and testing models. We keep the classical 70%-30% approach.

```
## 70% of the sample size
smp_size <- floor(0.7 * nrow(players_model))

train_ind <- sample(seq_len(nrow(players_model)), size = smp_size)

train_with_label <- players_model[train_ind, ]
test_with_label <- players_model[-train_ind, ]
```

```r
print('Train set size:')
```

```
## [1] "Train set size:"
```

```r
print(dim(train_with_label))
```

```
## [1] 9228   27
```

```r
print('Validation set size:')
```

```
## [1] "Validation set size:"
```

```r
print(dim(test_with_label))
```

```
## [1] 3955   27
```

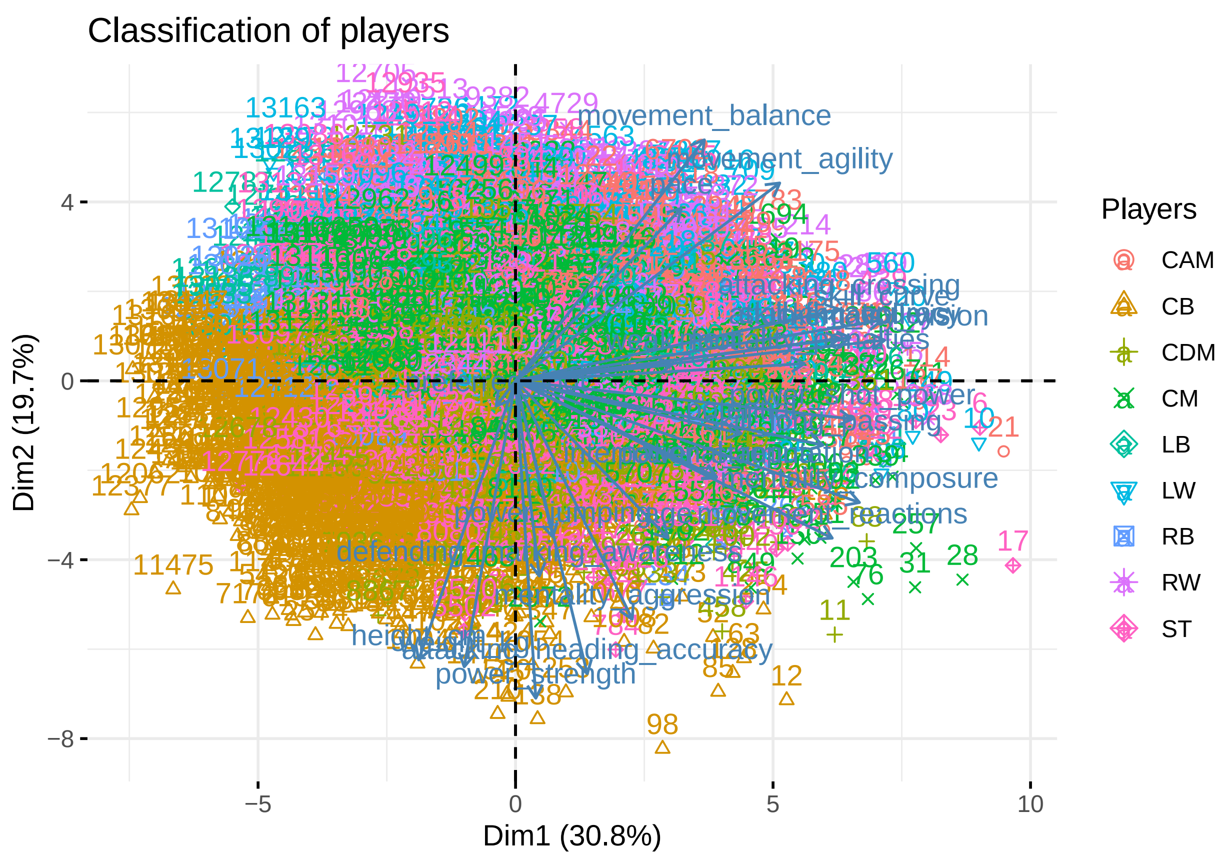
We factorize the labels, so we can use them in our models.

```r
# factorize labels
train_y <- as.factor(train_with_label[,27])
test_y <- as.factor(test_with_label[,27])
# remove labels from sets
train <- train_with_label[1:(length(train_with_label)-1)]
test <- test_with_label[1:(length(test_with_label)-1)]
head(test)
```

```
##          age potential international_reputation height_cm weight_kg      pace
## 6  0.2608696 1.0000000                     0.75 0.5625000 0.4615385 1.0000000
## 10 0.5217391 0.8695652                     0.75 0.5833333 0.5576923 0.8695652
## 11 0.5652174 0.8695652                     0.50 0.6250000 0.6730769 0.5362319
## 12 0.5652174 0.8695652                     0.75 0.7916667 0.8269231 0.7246377
## 17 0.7826087 0.8478261                     1.00 0.5625000 0.6538462 0.6376812
## 20 0.4347826 0.8695652                     0.75 0.3125000 0.3846154 0.9130435
##    preferred_foot weak_foot attacking_crossing attacking_heading_accuracy
## 6               1      0.75          0.7974684                  0.7236842
## 10              1      1.00          0.8607595                  0.6710526
## 11              1      0.50          0.5443038                  0.8157895
## 12              1      0.50          0.4810127                  0.9210526
## 17              1      0.75          0.8227848                  0.8815789
## 20              1      0.50          0.8101266                  0.3815789
##    attacking_volleys skill_curve skill_fk_accuracy skill_long_passing
## 6             0.9125   0.8292683         0.7023810          0.6986301
## 10            0.8500   0.8902439         0.7619048          0.7397260
## 11            0.6500   0.6097561         0.7619048          0.8767123
## 12            0.4375   0.5853659         0.7142857          0.9041096
## 17            1.0000   0.9024390         0.8571429          0.7808219
## 20            0.7125   0.8170732         0.7380952          0.6712329
##    movement_agility movement_reactions movement_balance power_shot_power
## 6         0.9420290          0.9846154        0.8142857        0.8800000
## 10        0.8550725          0.9538462        0.7428571        0.9066667
## 11        0.4927536          0.8923077        0.5714286        0.9066667
## 12        0.4927536          0.8923077        0.3857143        0.8133333
## 17        0.6956522          0.9692308        0.7428571        0.9200000
## 20        0.9710145          0.9384615        0.9571429        0.7733333
##    power_jumping power_stamina power_strength mentality_aggression
## 6      0.7424242     0.8732394      0.7532468            0.5600000
## 10     0.4696970     0.8732394      0.5844156            0.5600000
## 11     0.8787879     0.8873239      0.9220779            0.9466667
## 12     0.9242424     0.6056338      0.9480519            0.8400000
## 17     0.6060606     0.7323944      0.8571429            0.8933333
## 20     0.4242424     0.7464789      0.5844156            0.5200000
##    mentality_vision mentality_penalties mentality_composure
## 6         0.8414634              0.8250           0.8787879
## 10        0.8536585              0.7500           0.8939394
## 11        0.8170732              0.6625           0.8181818
## 12        0.6341463              0.6125           0.9090909
## 17        0.8658537              0.8750           0.8636364
## 20        0.8414634              0.6750           0.7272727
##    defending_marking_awareness
## 6                    0.1927711
## 10                   0.4819277
## 11                   0.9036145
## 12                   0.9879518
## 17                   0.3855422
## 20                   0.4457831
```

Just to take a sneak peek, this is how the validation labels are roughly distributed on the factor plane.We notice that the factor plane separates some types of labels quite good (e.g. CB), some not.

```r
test.pca<-prcomp(test,center=TRUE, scale.=TRUE)
fviz_pca_biplot(test.pca,
                label = "all",
                col.ind = test_y,
                legend.title = "Players",
                title = "Classification of players")
```



3.2 Useful functions

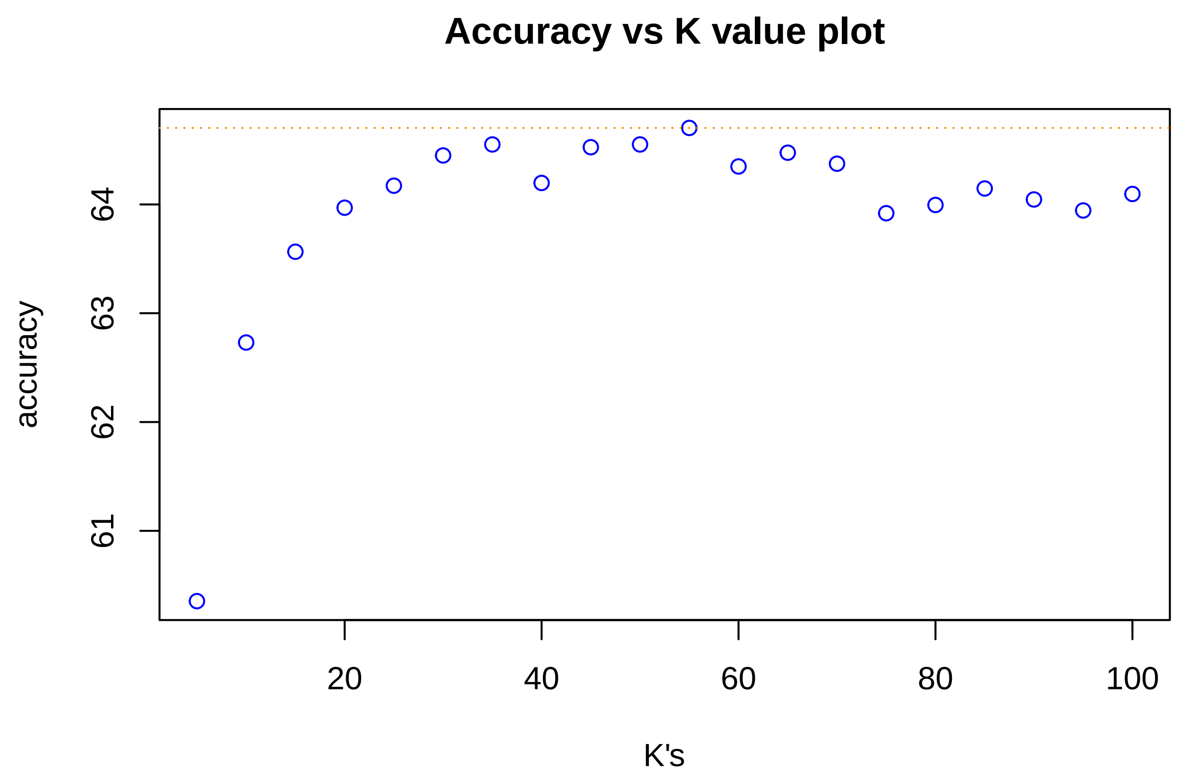
Before we train any model, we want to create a function that computes accuracy, and one that selects the missclassified data so we can visualize it later on the factor plane.

```
accuracy <- function(x){sum(diag(x)/(sum(rowSums(x)))) * 100}

missclassified <- function(pred, label){
  l<- pred
  l[c(pred)==c(label)]<- 0
  return (as.factor(l))
}

#cross validation
ctrl <-  trainControl(method = "repeatedcv", number = 100, repeats = 10)
```

3.3 Knn

```
##run knn function
class <- factor(c(train_y))

train <- train[1:(length(train)-1)]
test <- test[1:(length(test)-1)]
```

```
accuracy_vect <- c()
ks<- c()

for(k1 in seq(5,100,5)) {
    test_pred <-knn(train = train, test = test, cl = class, k = k1)
    accuracy_vect <- append(accuracy_vect,accuracy(table(test_y,test_pred)))
    ks <- append(ks, k1)
}

plot(ks, accuracy_vect, type = "p", col="blue", xlab="K's", ylab="accuracy", main="Accuracy vs K value p
abline(h = max(accuracy_vect), col = "darkorange", lty = 3)
```



We get the best k and its accuracy, represented by the orange line, namely k = 55, with 64.7% accuracy.

```
print('The best K in our case is:')
```

```
## [1] "The best K in our case is:"
```

```
print(ks[which.max(accuracy_vect)])
```

```
## [1] 55
```

```
print('And it gives us an accuracy of:' )
```

```
## [1] "And it gives us an accuracy of:"
```

```
print(accuracy_vect[which.max(accuracy_vect)])
```

```
## [1] 64.70291
```

Using the best k (k=55) we generate a confusion matrix to check misslabeled data

```
set.seed(42)
pred_knn <-knn(train = train, test = test, cl = class, k = 55)
```

```
conf_mat_knn <- confusionMatrix(data=pred_knn, reference = test_y)
conf_mat_knn
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction CAM  CB CDM  CM  LB  LW  RB  RW  ST
##        CAM  80   0   1  38   1  22   0  25   4
##        CB    0 655  46   2   8   0  18   0   8
##        CDM   1  34 165  55   1   1  16   3   3
##        CM  101   3 149 332  14  33  29  54  23
##        LB   13  31  21  33 317  38   4  16  21
##        LW   37   0   0   4   4 114   3  81  19
##        RB    1  23   6   6  24  10 260  23   2
##        RW   40   0   0  14   6  82   4 121  23
##        ST   12   6   2   3   0  45   9  41 511
##
## Overall Statistics
##
##                Accuracy : 0.646
##                  95% CI : (0.6309, 0.6609)
##     No Information Rate : 0.1901
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.5953
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: CAM Class: CB Class: CDM Class: CM Class: LB
## Sensitivity             0.28070    0.8710    0.42308   0.68172   0.84533
## Specificity             0.97520    0.9744    0.96802   0.88293   0.95056
## Pos Pred Value          0.46784    0.8887    0.59140   0.44986   0.64170
## Neg Pred Value          0.94582    0.9699    0.93879   0.95182   0.98324
## Prevalence              0.07206    0.1901    0.09861   0.12314   0.09482
## Detection Rate          0.02023    0.1656    0.04172   0.08394   0.08015
## Detection Prevalence    0.04324    0.1863    0.07054   0.18660   0.12491
## Balanced Accuracy       0.62795    0.9227    0.69555   0.78233   0.89795
##                      Class: LW Class: RB Class: RW Class: ST
## Sensitivity            0.33043   0.75802   0.33242    0.8322
## Specificity            0.95900   0.97370   0.95294    0.9647
## Pos Pred Value         0.43511   0.73239   0.41724    0.8124
## Neg Pred Value         0.93745   0.97694   0.93370    0.9690
## Prevalence             0.08723   0.08673   0.09204    0.1552
## Detection Rate         0.02882   0.06574   0.03059    0.1292
## Detection Prevalence   0.06625   0.08976   0.07332    0.1590
## Balanced Accuracy      0.64472   0.86586   0.64268    0.8985
```

3.4 Logistic regression

```
set.seed(42)
modellr <- nnet::multinom( train_y~., data = train,trControl=crtl)
```

```
## # weights:  243 (208 variable)
## initial  value 20275.988400
## iter  10 value 11090.535066
## iter  20 value 8282.291651
## iter  30 value 7202.533704
## iter  40 value 6919.539741
## iter  50 value 6824.388976
## iter  60 value 6810.745073
## iter  70 value 6808.940543
## iter  80 value 6808.805159
## iter  90 value 6808.776719
## final  value 6808.774198
## converged
```

```
set.seed(42)
pred_lr <- predict(modellr, test, type = "class")
accuracy(table(test_y, pred_lr))
```

```
## [1] 69.50695
```

The logistic regression performs better, with 69.5% accuracy. Let's have a look at the confusion matrix.

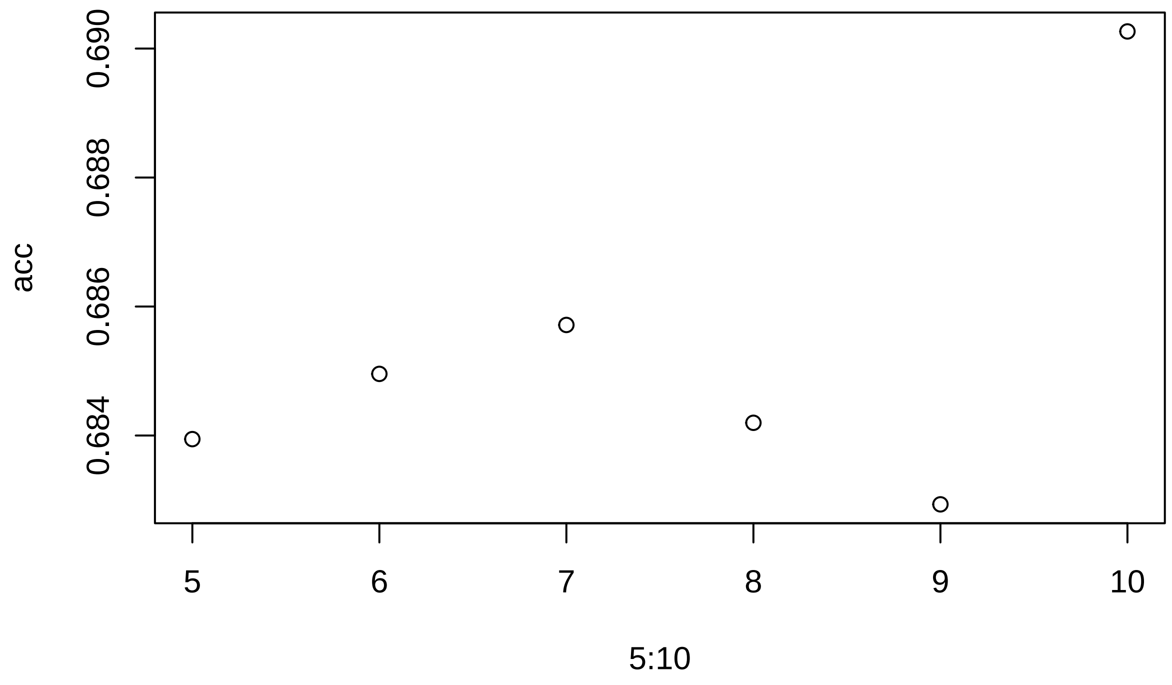
```
conf_mat <- confusionMatrix(data=pred_lr, reference = test_y)
conf_mat
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction CAM  CB CDM  CM  LB  LW  RB  RW  ST
##        CAM 119   0   0  52   0  32   0  28  13
##        CB    0 671  46   4   9   0  15   0   2
##        CDM   3  24 215  61   1   1   4   5   1
##        CM   76   4 109 354   8  10   3  21   7
##        LB    4  19   4   2 307  19   4   8   3
##          LW   26   0   0   2  13  77   0  81  17
##          RB    1  28  15   6  32   9 301  25   5
##          RW   27   0   0   6   5 143  14 163  24
##          ST   29   6   1   0   0  54   2  33 542
## 
## Overall Statistics
##                                           
##                Accuracy : 0.6951          
##                  95% CI : (0.6805, 0.7094)
##     No Information Rate : 0.1901          
##     P-Value [Acc > NIR] : < 2.2e-16       
##                                           
##                   Kappa : 0.6517          
##                                           
##  Mcnemar's Test P-Value : NA              
## 
## Statistics by Class:
## 
##                      Class: CAM Class: CB Class: CDM Class: CM Class: LB
## Sensitivity             0.41754    0.8923    0.55128   0.72690   0.81867
## Specificity             0.96594    0.9763    0.97195   0.93137   0.98240
## Pos Pred Value          0.48770    0.8983    0.68254   0.59797   0.82973
## Neg Pred Value          0.95527    0.9748    0.95192   0.96045   0.98103
## Prevalence              0.07206    0.1901    0.09861   0.12314   0.09482
## Detection Rate          0.03009    0.1697    0.05436   0.08951   0.07762
## Detection Prevalence    0.06169    0.1889    0.07965   0.14968   0.09355
## Balanced Accuracy       0.69174    0.9343    0.76162   0.82914   0.90053
##                      Class: LW Class: RB Class: RW Class: ST
## Sensitivity            0.22319   0.87755   0.44780    0.8827
## Specificity            0.96150   0.96650   0.93901    0.9626
## Pos Pred Value         0.35648   0.71327   0.42670    0.8126
## Neg Pred Value         0.92832   0.98811   0.94374    0.9781
## Prevalence             0.08723   0.08673   0.09204    0.1552
## Detection Rate         0.01947   0.07611   0.04121    0.1370
## Detection Prevalence   0.05461   0.10670   0.09659    0.1686
## Balanced Accuracy      0.59234   0.92203   0.69341    0.9227
```

3.5 Random Forrest

The hyperparameter we experiment with is 'mtry', that controls the number of variables randomly sampled as candidates for splitting at each tree node. Changing the number of trees does not do much, and from previous experimentation we realized that around 500 is the optimum value.

```
acc = c()  # empty vector to store accuracy values
i = 5

# Loop through values of i from 5 to 10
for (i in seq(5,10)) {
  # Train a random forest model with varying mtry values
  model_RF <- randomForest(train_y ~ ., data = train, ntree = 500, mtry = i, importance = TRUE)

  # Make predictions using the trained random forest model
  prediction_RF <- predict(model_RF, test, type = "class")
  # Calculate accuracy and store it in the 'acc' vector
  acc[i - 4] = mean(prediction_RF == test_y)  # Accuracy is calculated by comparing predicted and actua
}

# Plot accuracy values against the mtry values
plot(5:10, acc)
```



```
acc
```

```
## [1] 0.6839444 0.6849558 0.6857143 0.6841972 0.6829330 0.6902655
```

Randomly sampling 9 variables at each split gives the best accuracy with 68,4%.

```
model_rf <- randomForest(train_y ~ ., data = train, ntree = 500, mtry = 9, importance = TRUE)
pred_rf <- predict(model_rf, test, type = "class")
```

```
conf_mat <- confusionMatrix(data=pred_rf, reference = test_y)
conf_mat
```

```
## Confusion Matrix and Statistics
##
##            Reference
## Prediction CAM  CB CDM  CM  LB  LW  RB  RW  ST
##        CAM 103   0   2  43   0  21   0  23  10
##        CB    1 687  57   3  11   0  22   0   3
##        CDM   1  25 182  68   1   1   7   4   1
##        CM   88   3 130 339  13  14  10  32   9
##        LB    3  16   6   7 304  21   4  12   3
##        LW   39   0   0   7   9 114   4  96  17
##        RB    3  17  12   4  30  15 281  30   2
##        RW   24   0   0   8   6 109  10 133  26
##        ST   23   4   1   8   1  50   5  34 543
##
## Overall Statistics
##
##                Accuracy : 0.6791
##                  95% CI : (0.6643, 0.6937)
##     No Information Rate : 0.1901
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6329
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: CAM Class: CB Class: CDM Class: CM Class: LB
## Sensitivity             0.36140    0.9136    0.46667   0.69610   0.81067
## Specificity             0.97302    0.9697    0.96971   0.91378   0.97989
## Pos Pred Value          0.50990    0.8763    0.62759   0.53135   0.80851
## Neg Pred Value          0.95151    0.9795    0.94325   0.95538   0.98016
## Prevalence              0.07206    0.1901    0.09861   0.12314   0.09482
## Detection Rate          0.02604    0.1737    0.04602   0.08571   0.07686
## Detection Prevalence    0.05107    0.1982    0.07332   0.16131   0.09507
## Balanced Accuracy       0.66721    0.9416    0.71819   0.80494   0.89528
##                      Class: LW Class: RB Class: RW Class: ST
## Sensitivity            0.33043   0.81924   0.36538    0.8844
## Specificity            0.95235   0.96872   0.94904    0.9623
## Pos Pred Value         0.39860   0.71320   0.42089    0.8117
## Neg Pred Value         0.93704   0.98259   0.93652    0.9784
## Prevalence             0.08723   0.08673   0.09204    0.1552
## Detection Rate         0.02882   0.07105   0.03363    0.1373
## Detection Prevalence   0.07231   0.09962   0.07990    0.1692
## Balanced Accuracy      0.64139   0.89398   0.65721    0.9233
```

3.6 SVM

```
svm1 <- svm(formula= train_y~., data=train,
          type="C-classification", kernal="radial",
          gamma=0.1, cost=10)
```

```
pred_svm <- predict(svm1,test, type = "class")
accuracy(table(test_y, pred_svm))
```

```
## [1] 67.10493
```

We produce a summary of the model.

```r
summary(svm1)
```

```
##
## Call:
## svm(formula = train_y ~ ., data = train, type = "C-classification",
##     kernal = "radial", gamma = 0.1, cost = 10)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  10
##
## Number of Support Vectors:  7006
##
##  ( 793 875 566 837 611 639 829 1058 798 )
##
##
## Number of Classes:  9
##
## Levels:
##  CAM CB CDM CM LB LW RB RW ST
```

```r
conf_mat <- confusionMatrix(data=pred_svm, reference = test_y)
conf_mat
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction CAM  CB CDM  CM  LB  LW  RB  RW  ST
##        CAM 102   0  10  54   1  21   0  34  14
##        CB    1 676  43   5  12   0  23   0   1
##        CDM   5  25 207  58   0   1  12   3   2
##        CM   63   7  96 317   8  11   5  21   8
##        LB    2  20   7   4 296  19  15   7   5
##        LW   47   0   3  17  18 134   7 110  26
##        RB    0  20  17   9  25   9 263  17   6
##        RW   40   0   3  15  13 108  15 139  32
##        ST   25   4   4   8   2  42   3  33 520
##
## Overall Statistics
##
##                Accuracy : 0.671
##                  95% CI : (0.6562, 0.6857)
##     No Information Rate : 0.1901
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6245
##
##  Mcnemar's Test P-Value : NA
##
```

```
## Statistics by Class:
##
##                      Class: CAM Class: CB Class: CDM Class: CM Class: LB
## Sensitivity             0.35789    0.8989    0.53077   0.65092   0.78933
## Specificity             0.96349    0.9735    0.97027   0.93685   0.97793
## Pos Pred Value          0.43220    0.8883    0.66134   0.59142   0.78933
## Neg Pred Value          0.95079    0.9762    0.94975   0.95028   0.97793
## Prevalence              0.07206    0.1901    0.09861   0.12314   0.09482
## Detection Rate          0.02579    0.1709    0.05234   0.08015   0.07484
## Detection Prevalence    0.05967    0.1924    0.07914   0.13552   0.09482
## Balanced Accuracy       0.66069    0.9362    0.75052   0.79389   0.88363
##                      Class: LW Class: RB Class: RW Class: ST
## Sensitivity            0.38841   0.76676   0.38187    0.8469
## Specificity            0.93684   0.97148   0.93706    0.9638
## Pos Pred Value         0.37017   0.71858   0.38082    0.8112
## Neg Pred Value         0.94127   0.97771   0.93733    0.9716
## Prevalence             0.08723   0.08673   0.09204    0.1552
## Detection Rate         0.03388   0.06650   0.03515    0.1315
## Detection Prevalence   0.09153   0.09254   0.09229    0.1621
## Balanced Accuracy      0.66262   0.86912   0.65947    0.9053
```

3.7 Label Grouping

The accuracies obtained are decent but not great, and the confusion matrix clearly explains why. Positions like CB, ST, LB, RB get classified really well. On the opposite side, positions like CAM/CM and RW/LW get often confused.

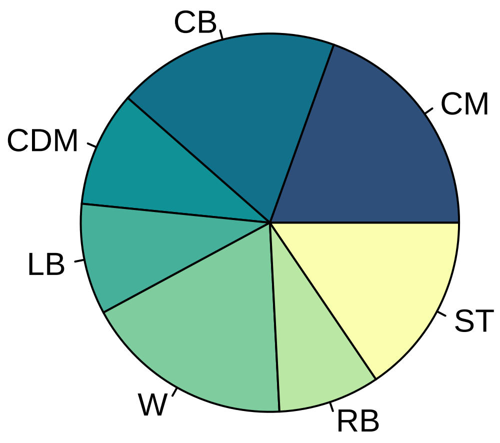
The first missclassification is explainable with basic attributes of the role. Centrer Attacking Midfielder (CAM) shares a lot of attacking characteristics with the Winger such as shooting and pace but also many with CM, like passing. This common attributes of course hindrance the task of the machine learning model.

It is a bit more tricky to detect why Left Winger and Right Winger get missclassified, when other positions that depends on the side (left or right), such Left Back and Right Back, are easily distinguished. For LB and RB the preferred foot plays a big role, since it's hard to find a righty who plays on the left and viceversa, because they cross and tackle mostly with their dominant foot. For RW and LW the distinction is less definable based on the preferred foot. On one hand, a lot of righty players like to play as Left Winger so they can converge to the center to shoot with their strong foot. Same is true for lefty on RW. On the other hand, many Wingers like to cross more, so they tend to do it with their preferred foot (LW with the left foot and RW with the right one). So for the model of course it's really not an easy job to detect these differences that pertain to the single player style of play; and this problem explains the drop in accuracy for these positions. In order to improve the accuracy of our classifiers, we group RW and LW together in a new position 'W = Winger' and we also group the CAM with CM in the class "CM = Center Midfielder".

```r
test_y2 <- test_y
levels(test_y2)[levels(test_y2) == "RW"| levels(test_y2) == "LW"] <- "W"
levels(test_y2)[levels(test_y2) == "CAM"| levels(test_y2) == "CM"] <- "CM"



train_y2 <- train_y
levels(train_y2)[levels(train_y2) == "RW"| levels(train_y2) == "LW"] <- "W"
levels(train_y2)[levels(train_y2) == "CAM"| levels(train_y2) == "CM"] <- "CM"

unique(test_y2)
```

```
## [1] ST  W   CDM CB  CM  RB  LB
## Levels: CM CB CDM LB W RB ST
```

```
#plot pie chart again
cat<- table(factor(test_y2))
pie(cat, col = hcl.colors(length(cat), "BluYl"))
```



This is the new distribution of labels. Now we reproduce the same experiments, expecting a hefty increase in accuracy, with the price of ablation.

3.7.1 Knn

```
pred_knn2 <-knn(train = train, test = test, cl = train_y2, k = 51)
confusionMatrix(data=pred_knn2, reference = test_y2)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  CM  CB CDM  LB   W  RB  ST
##        CM  573   5 165  18 131  37  37
##        CB    2 654  47   8   0  19   7
##        CDM  47  35 150   1   3  14   2
##        LB   25  32  17 308  40   4  16
##        W   112   0   3  20 473  18  68
##        RB    5  21   7  20  16 247   3
##        ST    8   5   1   0  46   4 481
##
## Overall Statistics
##
##                Accuracy : 0.7297
##                  95% CI : (0.7156, 0.7435)
##     No Information Rate : 0.1952
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6783
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: CM Class: CB Class: CDM Class: LB Class: W
## Sensitivity             0.7422    0.8697    0.38462   0.82133   0.6671
## Specificity             0.8765    0.9741    0.97139   0.96257   0.9319
## Pos Pred Value          0.5932    0.8874    0.59524   0.69683   0.6816
## Neg Pred Value          0.9334    0.9695    0.93519   0.98093   0.9276
## Prevalence              0.1952    0.1901    0.09861   0.09482   0.1793
## Detection Rate          0.1449    0.1654    0.03793   0.07788   0.1196
## Detection Prevalence    0.2442    0.1863    0.06372   0.11176   0.1755
## Balanced Accuracy       0.8094    0.9219    0.67800   0.89195   0.7995
##                      Class: RB Class: ST
## Sensitivity            0.72012    0.7834
## Specificity            0.98007    0.9808
## Pos Pred Value         0.77429    0.8826
## Neg Pred Value         0.97360    0.9610
## Prevalence             0.08673    0.1552
## Detection Rate         0.06245    0.1216
## Detection Prevalence   0.08066    0.1378
## Balanced Accuracy      0.85009    0.8821
```

```
accuracy(table(test_y2, pred_knn2))
```

```
## [1] 72.97092
```

3.7.2 Logistic Regression

```
modellr2 <- nnet::multinom( train_y2~., data = train,trControl=crtl)
```

```
## # weights:  189 (156 variable)
## initial  value 17956.858855
## iter  10 value 9362.483529
## iter  20 value 6550.464489
## iter  30 value 5339.982341
## iter  40 value 5060.370961
## iter  50 value 5046.775088
## iter  60 value 5046.323525
## iter  70 value 5046.204604
## iter  80 value 5046.183419
## final  value 5046.182036
## converged
```

```
pred_lr2 <- predict(modellr2, test, type = "class")
accuracy(table(test_y2, pred_lr2))
```

```
## [1] 78.71049
```

```
confusionMatrix(data=pred_lr2, reference = test_y2)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  CM  CB CDM  LB   W  RB  ST
##        CM  607   7 121  10  82   4  28
##        CB    3 671  47   8   0  16   2
##        CDM  60  22 203   1   5   4   0
##        LB    2  18   4 300  18   4   2
##        W    79   0   0  30 521  22  59
##        RB    4  29  15  26  24 292   4
##        ST   17   5   0   0  59   1 519
##
## Overall Statistics
##
##                Accuracy : 0.7871
##                  95% CI : (0.774, 0.7998)
##     No Information Rate : 0.1952
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.747
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: CM Class: CB Class: CDM Class: LB Class: W
## Sensitivity             0.7863    0.8923    0.52051   0.80000   0.7348
## Specificity             0.9208    0.9763    0.97419   0.98659   0.9415
## Pos Pred Value          0.7066    0.8983    0.68814   0.86207   0.7328
## Neg Pred Value          0.9467    0.9748    0.94891   0.97921   0.9420
## Prevalence              0.1952    0.1901    0.09861   0.09482   0.1793
## Detection Rate          0.1535    0.1697    0.05133   0.07585   0.1317
## Detection Prevalence    0.2172    0.1889    0.07459   0.08799   0.1798
## Balanced Accuracy       0.8535    0.9343    0.74735   0.89330   0.8382
##                      Class: RB Class: ST
## Sensitivity            0.85131    0.8453
## Specificity            0.97176    0.9755
## Pos Pred Value         0.74112    0.8636
## Neg Pred Value         0.98568    0.9717
## Prevalence             0.08673    0.1552
## Detection Rate         0.07383    0.1312
## Detection Prevalence   0.09962    0.1520
## Balanced Accuracy      0.91154    0.9104
```

3.6.2 Random Forrest

```
model_rf2 <- randomForest(train_y2 ~ ., data = train, ntree = 500, mtry = 8, importance = TRUE)
pred_rf2 <- predict(model_rf2, test, type = "class")
summary(model_rf2)
```

```
##                 Length Class  Mode
## call                6  -none- call
## type                1  -none- character
## predicted        9228  factor numeric
## err.rate         4000  -none- numeric
## confusion          56  -none- numeric
## votes           64596  matrix numeric
## oob.times        9228  -none- numeric
## classes             7  -none- character
## importance        225  -none- numeric
## importanceSD      200  -none- numeric
## localImportance     0  -none- NULL
## proximity           0  -none- NULL
## ntree               1  -none- numeric
## mtry                1  -none- numeric
## forest             14  -none- list
## y                9228  factor numeric
## test                0  -none- NULL
## inbag               0  -none- NULL
## terms               3  terms  call
```

```
accuracy(table(pred_rf2, test_y2))
```

```
## [1] 77.21871
```

```
confusionMatrix(data=pred_rf2, reference = test_y2)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  CM  CB CDM  LB   W  RB  ST
##        CM  588   5 148  17  90  14  22
##        CB    2 690  56  14   0  21   4
##        CDM  61  22 173   1   2   7   0
##        LB    3  15   3 287  17   4   3
##        W    96   0   0  30 520  22  58
##        RB    4  16  10  26  22 270   1
##        ST   18   4   0   0  58   5 526
##
## Overall Statistics
##
##                Accuracy : 0.7722
##                  95% CI : (0.7588, 0.7852)
##     No Information Rate : 0.1952
##     P-Value [Acc > NIR] : < 2.2e-16
```

```
##
##                   Kappa : 0.7286
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: CM Class: CB Class: CDM Class: LB Class: W
## Sensitivity             0.7617    0.9176    0.44359   0.76533   0.7334
## Specificity             0.9070    0.9697    0.97391   0.98743   0.9365
## Pos Pred Value          0.6652    0.8767    0.65038   0.86446   0.7163
## Neg Pred Value          0.9401    0.9804    0.94118   0.97571   0.9415
## Prevalence              0.1952    0.1901    0.09861   0.09482   0.1793
## Detection Rate          0.1487    0.1745    0.04374   0.07257   0.1315
## Detection Prevalence    0.2235    0.1990    0.06726   0.08394   0.1836
## Balanced Accuracy       0.8343    0.9436    0.70875   0.87638   0.8350
##                      Class: RB Class: ST
## Sensitivity            0.78717    0.8567
## Specificity            0.97813    0.9746
## Pos Pred Value         0.77364    0.8609
## Neg Pred Value         0.97976    0.9737
## Prevalence             0.08673    0.1552
## Detection Rate         0.06827    0.1330
## Detection Prevalence   0.08824    0.1545
## Balanced Accuracy      0.88265    0.9156
```

3.6.3 SVM

```
model_svm2 <- svm(formula= train_y2~., data=train,
          type="C-classification", kernal="radial",
          gamma=0.1, cost=10)
pred_svm2 <- predict(model_svm2, test, type = "class")
```

```
summary(model_svm2)
```

```
##
## Call:
## svm(formula = train_y2 ~ ., data = train, type = "C-classification",
##     kernal = "radial", gamma = 0.1, cost = 10)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  10
##
## Number of Support Vectors:  6424
##
##  ( 1325 866 567 1447 631 795 793 )
##
##
```

```
## Number of Classes:  7
##
## Levels:
##  CM CB CDM LB W RB ST
```

```
accuracy(table(test_y2, pred_svm2))
```

```
## [1] 75.32238
```

```
confusionMatrix(data=pred_svm2, reference = test_y2)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  CM  CB CDM  LB   W  RB  ST
##        CM  549   7 113   8  96   9  25
##        CB    5 676  44  12   0  23   1
##        CDM  59  25 202   0   3  11   2
##        LB    5  20   5 291  25  14   2
##        W   119   1   9  39 488  24  65
##        RB    9  20  13  23  23 259   5
##        ST   26   3   4   2  74   3 514
##
## Overall Statistics
##
##                Accuracy : 0.7532
##                  95% CI : (0.7395, 0.7666)
##     No Information Rate : 0.1952
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7067
##
##  Mcnemar's Test P-Value : 0.001686
##
## Statistics by Class:
##
##                      Class: CM Class: CB Class: CDM Class: LB Class: W
## Sensitivity             0.7111    0.8989    0.51795   0.77600   0.6883
## Specificity             0.9189    0.9735    0.97195   0.98017   0.9208
## Pos Pred Value          0.6803    0.8883    0.66887   0.80387   0.6550
## Neg Pred Value          0.9292    0.9762    0.94854   0.97662   0.9312
## Prevalence              0.1952    0.1901    0.09861   0.09482   0.1793
## Detection Rate          0.1388    0.1709    0.05107   0.07358   0.1234
## Detection Prevalence    0.2040    0.1924    0.07636   0.09153   0.1884
## Balanced Accuracy       0.8150    0.9362    0.74495   0.87808   0.8046
##                      Class: RB Class: ST
## Sensitivity            0.75510    0.8371
## Specificity            0.97425    0.9665
## Pos Pred Value         0.73580    0.8211
## Neg Pred Value         0.97669    0.9700
## Prevalence             0.08673    0.1552
## Detection Rate         0.06549    0.1300
## Detection Prevalence   0.08900    0.1583
## Balanced Accuracy      0.86468    0.9018
```

Overall we can appreciate a significant improvement in accuracy, on average of 8-9%, in all the models. The model that performed the best was the Logistic Regression with 78.7%, followed in order by Random Forest (77.2%), SVM (75.3%) and KNN (72.9%). In today's Machine Learning standards these might not seem staggering result, but we cannot neglect the challenging task we tried to address. In such a complex sport where roles and attributes intertwine and change during the season or even in the same match, classify with precision the position of a player is not an easy job, because often a player cannot be confined to a single area of the field.

4. Regularization: Ridge regression and Lasso

In this part we explore two shrinkage methods, namely Ridge regression and Lasso, also known as penalized regression methods. Through these techniques we can fit a model where all the predictors are contained, but some coefficient estimates are shrinked towards zero.

First we take the players_market and remove the player_positions column that we will not consider for this task.

```
players_market <-subset(players_regress, select = -player_positions)
head(players_market)
```

```
##          short_name age potential international_reputation value_eur height_cm
## 1          L. Messi  34        93                        5  78000000       170
## 2    R. Lewandowski  32        92                        5 119500000       185
## 3 Cristiano Ronaldo  36        91                        5  45000000       187
## 4         Neymar Jr  29        91                        5 129000000       175
## 5      K. De Bruyne  30        91                        4 125500000       181
## 7         K. Mbappé  22        95                        4 194000000       182
##   weight_kg pace shooting passing preferred_foot weak_foot dribbling defending
## 1        72   85       92      91           Left         4        95        34
## 2        81   78       92      79          Right         4        86        44
## 3        83   87       94      80          Right         4        88        34
## 4        68   91       83      86          Right         5        94        37
## 5        70   76       86      93          Right         5        88        64
## 7        73   97       88      80          Right         4        92        36
##   physic attacking_crossing attacking_finishing attacking_heading_accuracy
## 1     65                 85                  95                         70
## 2     82                 71                  95                         90
## 3     75                 87                  95                         90
## 4     63                 85                  83                         63
## 5     78                 94                  82                         55
## 7     77                 78                  93                         72
##   attacking_short_passing attacking_volleys skill_dribbling skill_curve
## 1                      91                88              96          93
## 2                      85                89              85          79
## 3                      80                86              88          81
## 4                      86                86              95          88
## 5                      94                82              88          85
## 7                      85                83              93          80
##   skill_fk_accuracy skill_long_passing skill_ball_control movement_acceleration
## 1                94                 91                 96                    91
## 2                85                 70                 88                    77
## 3                84                 77                 88                    85
## 4                87                 81                 95                    93
## 5                83                 93                 91                     76
## 7                69                 71                 91                     97
##   movement_sprint_speed movement_agility movement_reactions movement_balance
## 1                    80               91                 94               95
## 2                    79               77                 93               82
## 3                    88               86                 94               74
## 4                    89               96                 89               84
## 5                    76               79                 91               78
## 7                    97               92                 93               83
##   power_shot_power power_jumping power_stamina power_strength power_long_shots
## 1               86            68            72             69               94
## 2               90            85            76             86               87
## 3               94            95            77             77               93
## 4               80            64            81             53               81
## 5               91            63            89             74               91
## 7               86            78            88             77               82
##   mentality_aggression mentality_interceptions mentality_positioning
## 1                   44                      40                    93
## 2                   81                      49                    95
## 3                   63                      29                    95
## 4                   63                      37                    86
## 5                   76                      66                    88
## 7                   62                      38                    92
##   mentality_vision mentality_penalties mentality_composure
## 1               95                  75                  96
## 2               81                  90                  88
## 3               76                  88                  95
## 4               90                  93                  93
## 5               94                  83                  89
## 7               82                  79                  88
##   defending_marking_awareness defending_standing_tackle
## 1                          20                        35
## 2                          35                        42
## 3                          24                        32
## 4                          35                        32
## 5                          68                        65
## 7                          26                        34
##   defending_sliding_tackle
## 1                       24
## 2                       19
## 3                       24
## 4                       29
## 5                       53
## 7                       32
```

```
players_market$value_eur <- as.integer(players_market$value_eur)
# Left foot is -1 and Right foot is 1. Basically one-hot encoding but we only have 2 categories so its
players_market$preferred_foot[players_market[,"preferred_foot"]== "Left"] <- as.numeric(-1)
players_market$preferred_foot[players_market[,"preferred_foot"]== "Right"] <- as.numeric(1)
players_market$preferred_foot <- as.numeric(players_market$preferred_foot)
head(players_market)
```

```
##          short_name age potential international_reputation value_eur height_cm
## 1          L. Messi  34        93                        5  78000000       170
## 2    R. Lewandowski  32        92                         5 119500000       185
## 3 Cristiano Ronaldo  36        91                         5  45000000       187
## 4         Neymar Jr  29        91                         5 129000000       175
## 5      K. De Bruyne  30        91                         4 125500000       181
## 7         K. Mbappé  22        95                         4 194000000       182
##   weight_kg pace shooting passing preferred_foot weak_foot dribbling defending
## 1        72   85       92      91             -1         4        95        34
## 2        81   78       92      79              1         4        86        44
## 3        83   87       94      80              1         4        88        34
## 4        68   91       83      86              1         5        94        37
## 5        70   76       86      93              1         5        88        64
## 7        73   97       88      80              1         4        92        36
##   physic attacking_crossing attacking_finishing attacking_heading_accuracy
## 1     65                 85                  95                         70
## 2     82                 71                  95                         90
## 3     75                 87                  95                         90
## 4     63                 85                  83                         63
## 5     78                 94                  82                         55
## 7     77                 78                  93                         72
##   attacking_short_passing attacking_volleys skill_dribbling skill_curve
## 1                      91                88              96          93
## 2                      85                89              85          79
## 3                      80                86              88          81
## 4                      86                86              95          88
## 5                      94                82              88          85
## 7                      85                83              93          80
##   skill_fk_accuracy skill_long_passing skill_ball_control movement_acceleration
## 1                94                 91                 96                    91
## 2                85                 70                 88                    77
## 3                84                 77                 88                    85
## 4                87                 81                 95                    93
## 5                83                 93                 91                    76
## 7                69                 71                 91                    97
##   movement_sprint_speed movement_agility movement_reactions movement_balance
## 1                    80               91                 94               95
## 2                    79               77                 93               82
## 3                    88               86                 94               74
## 4                    89               96                 89               84
## 5                    76               79                 91               78
## 7                    97               92                 93               83
##   power_shot_power power_jumping power_stamina power_strength power_long_shots
## 1               86            68            72             69               94
## 2               90            85            76             86               87
## 3               94            95            77             77               93
## 4               80            64            81             53               81
## 5               91            63            89             74               91
## 7               86            78            88             77               82
##   mentality_aggression mentality_interceptions mentality_positioning
## 1                   44                      40                    93
## 2                   81                      49                    95
## 3                   63                      29                    95
## 4                   63                      37                    86
## 5                   76                      66                    88
## 7                   62                      38                    92
##   mentality_vision mentality_penalties mentality_composure
## 1               95                  75                  96
## 2               81                  90                  88
## 3               76                  88                  95
## 4               90                  93                  93
## 5               94                  83                  89
## 7               82                  79                  88
##   defending_marking_awareness defending_standing_tackle
## 1                          20                        35
## 2                          35                        42
## 3                          24                        32
## 4                          35                        32
## 5                          68                        65
## 7                          26                        34
##   defending_sliding_tackle
## 1                       24
## 2                       19
## 3                       24
## 4                       29
## 5                       53
## 7                       32
```

```
#norm
# normalization function
normalize <-function(x) { (x -min(x))/(max(x)-min(x))   }
colnames(players_market)
```

```
##  [1] "short_name"                  "age"
##  [3] "potential"                   "international_reputation"
##  [5] "value_eur"                   "height_cm"
##  [7] "weight_kg"                   "pace"
##  [9] "shooting"                    "passing"
## [11] "preferred_foot"              "weak_foot"
## [13] "dribbling"                   "defending"
## [15] "physic"                      "attacking_crossing"
## [17] "attacking_finishing"         "attacking_heading_accuracy"
## [19] "attacking_short_passing"     "attacking_volleys"
## [21] "skill_dribbling"             "skill_curve"
## [23] "skill_fk_accuracy"           "skill_long_passing"
## [25] "skill_ball_control"          "movement_acceleration"
## [27] "movement_sprint_speed"       "movement_agility"
## [29] "movement_reactions"          "movement_balance"
## [31] "power_shot_power"            "power_jumping"
## [33] "power_stamina"               "power_strength"
## [35] "power_long_shots"            "mentality_aggression"
## [37] "mentality_interceptions"     "mentality_positioning"
## [39] "mentality_vision"            "mentality_penalties"
## [41] "mentality_composure"         "defending_marking_awareness"
## [43] "defending_standing_tackle"   "defending_sliding_tackle"
```

```
# normalize
players_market_norm <- as.data.frame(lapply(players_market[, c(2:44)], normalize))
head(players_market_norm,5)
```

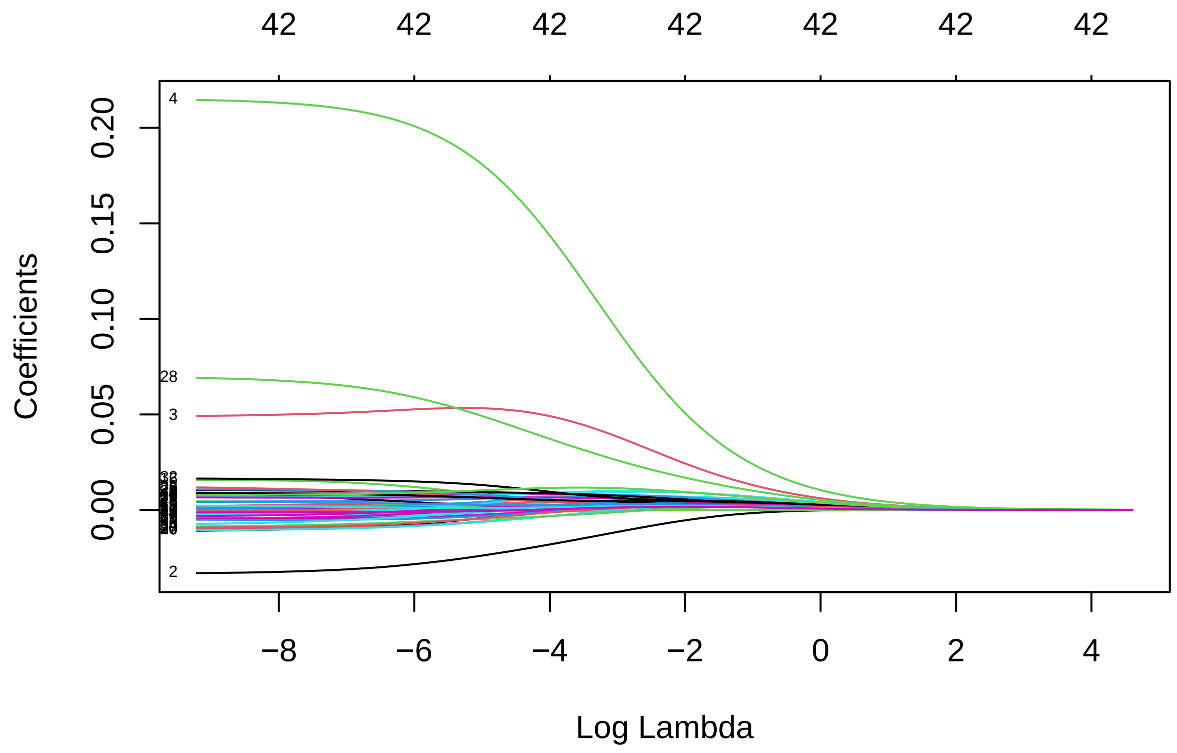
```
##         age potential international_reputation value_eur height_cm weight_kg
## 1 0.7826087 0.9565217                     1.00 0.4020156 0.3125000 0.4423077
## 2 0.6956522 0.9347826                     1.00 0.6159497 0.6250000 0.6153846
## 3 0.8695652 0.9130435                     1.00 0.2318994 0.6666667 0.6538462
## 4 0.5652174 0.9130435                     1.00 0.6649225 0.4166667 0.3653846
## 5 0.6086957 0.9130435                     0.75 0.6468799 0.5416667 0.4038462
##        pace  shooting   passing preferred_foot weak_foot dribbling defending
## 1 0.8260870 0.9736842 0.9705882              0      0.75 1.0000000 0.2500000
## 2 0.7246377 0.9736842 0.7941176              1      0.75 0.8676471 0.3815789
## 3 0.8550725 1.0000000 0.8088235              1      0.75 0.8970588 0.2500000
## 4 0.9130435 0.8552632 0.8970588              1      1.00 0.9852941 0.2894737
## 5 0.6956522 0.8947368 1.0000000              1      1.00 0.8970588 0.6447368
##      physic attacking_crossing attacking_finishing attacking_heading_accuracy
## 1 0.5901639          0.8860759           1.0000000                  0.6973684
## 2 0.8688525          0.7088608           1.0000000                  0.9605263
## 3 0.7540984          0.9113924           1.0000000                  0.9605263
## 4 0.5573770          0.8860759           0.8588235                  0.6052632
## 5 0.8032787          1.0000000           0.8470588                  0.5000000
##   attacking_short_passing attacking_volleys skill_dribbling skill_curve
## 1               0.9577465            0.9750       1.0000000   0.9878049
## 2               0.8732394            0.9875       0.8589744   0.8170732
## 3               0.8028169            0.9500       0.8974359   0.8414634
## 4               0.8873239            0.9500       0.9871795   0.9268293
## 5               1.0000000            0.9000       0.8974359   0.8902439
##   skill_fk_accuracy skill_long_passing skill_ball_control movement_acceleration
## 1         1.0000000          0.9726027          1.0000000             0.9142857
## 2         0.8928571          0.6849315          0.8888889             0.7142857
## 3         0.8809524          0.7808219          0.8888889             0.8285714
## 4         0.9166667          0.8356164          0.9861111             0.9428571
## 5         0.8690476          1.0000000          0.9305556             0.7000000
##   movement_sprint_speed movement_agility movement_reactions movement_balance
## 1             0.7571429        0.9275362          1.0000000        0.9857143
## 2             0.7428571        0.7246377          0.9846154        0.8000000
## 3             0.8714286        0.8550725          1.0000000        0.6857143
## 4             0.8857143        1.0000000          0.9230769        0.8285714
## 5             0.7000000        0.7536232          0.9538462        0.7428571
##   power_shot_power power_jumping power_stamina power_strength power_long_shots
## 1        0.8800000     0.5909091     0.6478873      0.6493506        1.0000000
## 2        0.9333333     0.8484848     0.7042254      0.8701299        0.9156627
## 3        0.9866667     1.0000000     0.7183099      0.7532468        0.9879518
## 4        0.8000000     0.5303030     0.7746479      0.4415584        0.8433735
## 5        0.9466667     0.5151515     0.8873239      0.7142857        0.9638554
##   mentality_aggression mentality_interceptions mentality_positioning
## 1            0.3200000               0.3703704             0.9642857
## 2            0.8133333               0.4814815             0.9880952
## 3            0.5733333               0.2345679             0.9880952
## 4            0.5733333               0.3333333             0.8809524
## 5            0.7466667               0.6913580             0.9047619
##   mentality_vision mentality_penalties mentality_composure
## 1        1.0000000              0.7750           1.0000000
## 2        0.8292683              0.9625           0.8787879
## 3        0.7682927              0.9375           0.9848485
## 4        0.9390244              1.0000           0.9545455
## 5        0.9878049              0.8750           0.8939394
##   defending_marking_awareness defending_standing_tackle
## 1                   0.1204819                 0.3012048
## 2                   0.3012048                 0.3855422
## 3                   0.1686747                 0.2650602
## 4                   0.3012048                 0.2650602
## 5                   0.6987952                 0.6626506
##   defending_sliding_tackle
## 1                0.1707317
## 2                0.1097561
## 3                0.1707317
## 4                0.2317073
## 5                0.5243902
```

```
attach(players_market_norm)
X <- model.matrix(value_eur ~.,players_market_norm)
y <- value_eur
```

4.1 Ridge Regression

Ridge uses quadratic shrinking.

```
#ridge regression
grid.ridge <-10^seq(-4,2,length=100)
ridge.mod<-glmnet(X,y,alpha = 0,lambda = grid.ridge)
plot(ridge.mod, xvar="lambda", label= TRUE)
```



The plot illustrates how much the coefficients are penalized for different values of lambda. Notice none of the coefficients are forced to be zero.

Looking at the plot the features:

[2]age has a negative impact on th market value, which makes absolute sense since the older the player the less money he will be worth. On the other hand, [4] international_reputation, [26] movement acceleration and [3] potential have a big positive impact on the market price

```r
colnames(players_market)
```

```
##  [1] "short_name"                "age"
##  [3] "potential"                 "international_reputation"
##  [5] "value_eur"                 "height_cm"
##  [7] "weight_kg"                 "pace"
##  [9] "shooting"                  "passing"
## [11] "preferred_foot"            "weak_foot"
## [13] "dribbling"                 "defending"
## [15] "physic"                    "attacking_crossing"
## [17] "attacking_finishing"       "attacking_heading_accuracy"
## [19] "attacking_short_passing"   "attacking_volleys"
## [21] "skill_dribbling"           "skill_curve"
## [23] "skill_fk_accuracy"         "skill_long_passing"
## [25] "skill_ball_control"        "movement_acceleration"
## [27] "movement_sprint_speed"     "movement_agility"
## [29] "movement_reactions"        "movement_balance"
## [31] "power_shot_power"          "power_jumping"
## [33] "power_stamina"             "power_strength"
## [35] "power_long_shots"          "mentality_aggression"
## [37] "mentality_interceptions"   "mentality_positioning"
## [39] "mentality_vision"          "mentality_penalties"
## [41] "mentality_composure"       "defending_marking_awareness"
## [43] "defending_standing_tackle" "defending_sliding_tackle"
```

```r
# select n/2 observations for training set
train <- sample(1:nrow(X), nrow(X)/2)
test <- (-train)
y.test <- y[test]
```

```r
# fit ridge regression on the training set
ridge.mod <- glmnet(X[train, ], y[train], alpha = 0,
                    lambda = grid.ridge, thresh = 1e-12)
```

We estimate the test MSE for one lambda value, e.g lambda = 9

```r
#We estimate the test MSE for one lambda value, e.g lambda = 9
ridge.pred <- predict(ridge.mod, s = 9, newx = X[test, ], type="response")
mean((ridge.pred - y.test)^2)
```

```
## [1] 0.002040698
```

let’s see the coefficients for lambda = 9

```r
predict(ridge.mod, s = 9, exact = TRUE, type = "coefficients",
        x = X[train, ], y = y[train])[1:41, ]
```

```
##                (Intercept)                (Intercept) 
##               1.088777e-02               0.000000e+00 
##                        age                  potential 
##               3.844676e-05               8.079290e-04 
##   international_reputation                  height_cm 
##               1.268657e-03               8.481373e-05 
##                  weight_kg                       pace 
##               1.225221e-04               2.533876e-04 
##                   shooting                    passing 
##               3.305241e-04               5.376185e-04 
##             preferred_foot                  weak_foot 
##              -4.531360e-06               1.912625e-04 
##                  dribbling                  defending 
##               5.675027e-04               1.393295e-04 
##                     physic         attacking_crossing 
##               2.993649e-04               3.346249e-04 
##        attacking_finishing attacking_heading_accuracy 
##               2.824697e-04               3.197270e-04 
##    attacking_short_passing          attacking_volleys 
##               6.535299e-04               3.271983e-04 
##             skill_dribbling                skill_curve 
##               4.654953e-04               3.372552e-04 
##           skill_fk_accuracy         skill_long_passing 
##               2.843348e-04               3.959557e-04 
##          skill_ball_control      movement_acceleration 
##               6.532623e-04               2.302409e-04 
##       movement_sprint_speed           movement_agility 
##               2.486640e-04               2.238252e-04 
##          movement_reactions           movement_balance 
##               7.552708e-04               1.434842e-04 
##           power_shot_power              power_jumping 
##               3.627821e-04               1.447466e-04 
##              power_stamina             power_strength 
##               3.418250e-04               1.794867e-04 
##           power_long_shots       mentality_aggression 
##               2.972412e-04               2.392645e-04 
##    mentality_interceptions      mentality_positioning 
##               1.208521e-04               3.498357e-04 
##           mentality_vision        mentality_penalties 
##               4.497341e-04               2.975080e-04 
##        mentality_composure 
##               5.715892e-04
```

We use cross-validation to choose the value of lambda

```r
cv.out.ridge <- cv.glmnet(X[train, ], y[train], alpha = 0, nfold=10)
cv.out.ridge$lambda[1:10]
```

```
##  [1] 27.61409 25.16093 22.92570 20.88905 19.03332 17.34245 15.80180 14.39801
##  [9] 13.11893 11.95348
```

```
summary(cv.out.ridge$lambda)
```

```
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
##  0.002761  0.027636  0.276440  3.108108  2.763685 27.614090
```

```
# The mean cross-validated error
cv.out.ridge$cvm[1:10]
```

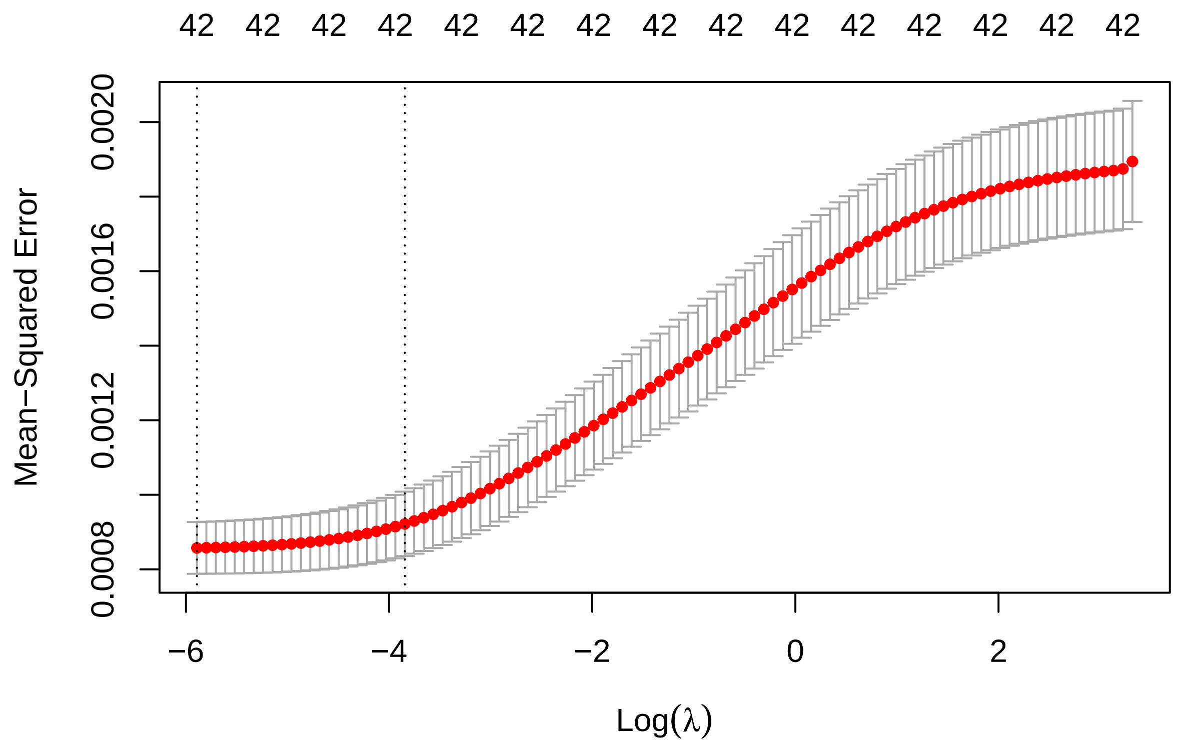
```
##  [1] 0.001894262 0.001874377 0.001869855 0.001867391 0.001864706 0.001861782
##  [7] 0.001858602 0.001855144 0.001851388 0.001847312
```

The following plot shows the cross-validation curve (red dotted line) with upper and lower standard deviation curves along the lambda sequence (error bars).

Two special values along the lambda sequence are indicated by the vertical dotted lines:

lambda.min is the value of lambda that gives minimum mean cross-validated error, while lambda.1se is the value of lambda that gives the most regularized model such that the cross-validated error is within one standard error of the minimum.

```
plot(cv.out.ridge)
```



```
# identify the best lambda value
i.bestlam <- which.min(cv.out.ridge$cvm)
i.bestlam
```

```
## [1] 100
```

```
bestlam <- cv.out.ridge$lambda[i.bestlam]
bestlam
```

```
## [1] 0.002761409
```

```
# mean cross-validated error for best lambda
cv.out.ridge$cvm[i.bestlam]
```

```
## [1] 0.0008573019
```

```
# estimate the test MSE
ridge.pred <- predict(ridge.mod, s = bestlam,
                      newx = X[test, ])
#MSE
mean((ridge.pred - y.test)^2)
```

```
## [1] 0.0009352046
```

```
# fit the coefficient with lambda=bestlam on all the data
out <- glmnet(X, y, alpha = 0)
predict(out, type = "coefficients", s = bestlam)
```

```
## 44 x 1 sparse Matrix of class "dgCMatrix"
##                                        s1
## (Intercept)                 -0.0723140503
## (Intercept)                  .
## age                         -0.0277912710
## potential                    0.0528695501
## international_reputation     0.1987834173
## height_cm                    0.0052479420
## weight_kg                   -0.0023766865
## pace                         0.0081741359
## shooting                     0.0037478398
## passing                      0.0032878494
## preferred_foot              -0.0003350786
## weak_foot                    0.0019683724
## dribbling                   -0.0012836592
## defending                    0.0009078396
## physic                       0.0077565378
## attacking_crossing           0.0003103480
## attacking_finishing          0.0115587699
## attacking_heading_accuracy   0.0026543530
## attacking_short_passing      0.0071171730
## attacking_volleys           -0.0015097796
## skill_dribbling             -0.0069628036
## skill_curve                 -0.0004080408
## skill_fk_accuracy            0.0026058103
## skill_long_passing          -0.0012107416
## skill_ball_control           0.0008634378
## movement_acceleration        0.0065536710
## movement_sprint_speed        0.0097110603
## movement_agility            -0.0078658353
## movement_reactions           0.0577623407
## movement_balance             0.0093242251
## power_shot_power            -0.0081411233
## power_jumping               -0.0035208811
## power_stamina                0.0147824128
## power_strength               0.0083613424
## power_long_shots            -0.0059305628
## mentality_aggression        -0.0043741063
## mentality_interceptions     -0.0030247251
## mentality_positioning       -0.0002608931
## mentality_vision             0.0073916237
## mentality_penalties         -0.0064979429
## mentality_composure          0.0092059060
## defending_marking_awareness  0.0031617404
## defending_standing_tackle    0.0011006923
## defending_sliding_tackle    -0.0009163138
```

```r
# residual sum of squares
ridge.rss <- sum((ridge.pred - y.test)^2)
ridge.rss
```

```
## [1] 6.164869
```

Let's compute the R squared. The R squared is a statistical measure of how well the regression predictions approximate the real data points. An R2 of 1 indicates that the regression predictions perfectly fit the data

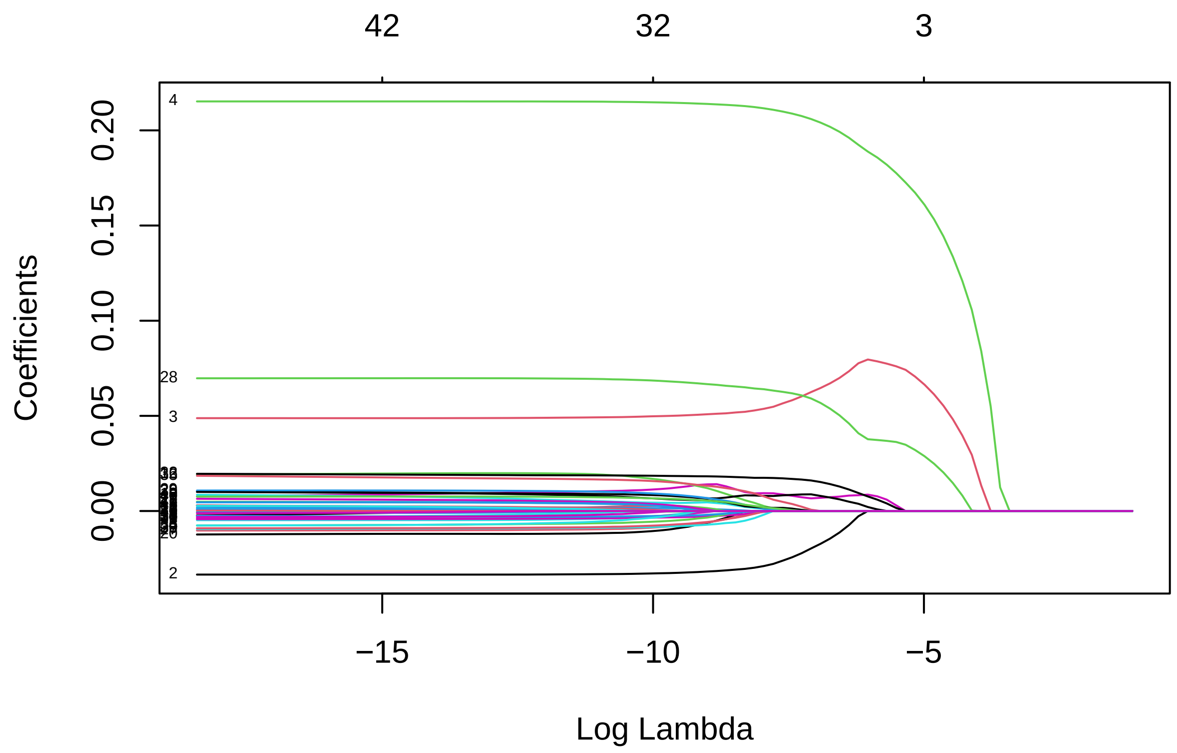
```r
ridge.tss <- sum((y.test - mean(y.test)) ^ 2)  ## total sum of squares
ridge.rsq <- 1 - ridge.rss/ridge.tss  # R squared
ridge.rsq
```

```
## [1] 0.5564568
```

4.2 Lasso Regularization

Lasso uses absolute-value shrinking.

```r
grid.lasso<-10^seq(-0.5,-8,length=100)
lasso.mod<-glmnet(X,y,alpha = 1,lambda = grid.lasso)
plot(lasso.mod, xvar="lambda", label = TRUE)
```



In the Lasso plot we can notice that some coefficients are forced to be zero. Moreover, it is clear once again the impact of international_reputation, acceleration, potential and age on the estimation of a player market price.

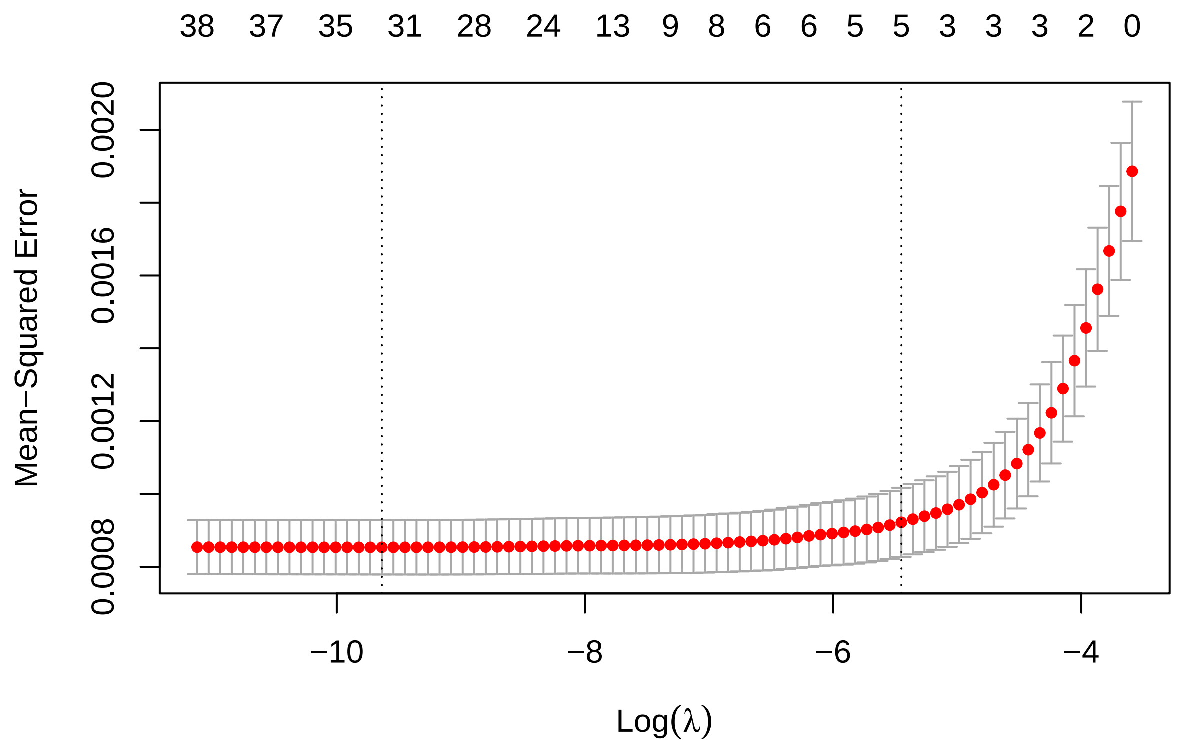
We use cross-validation to choose the value of lambda

```r
cv.out.lasso <- cv.glmnet(X[train, ], y[train], alpha = 1, nfold=10)
cv.out.lasso$lambda[1:10]
```

```
##  [1] 0.02761409 0.02516093 0.02292570 0.02088905 0.01903332 0.01734245
##  [7] 0.01580180 0.01439801 0.01311893 0.01195348
```

```r
# apply lasso to the training set
lasso.mod <- glmnet(X[train,], y[train], alpha=1, lambda=grid.lasso)
```

```r
# apply 10fold cross-validation to the training set
cv.out.lasso <- cv.glmnet(X[train,], y[train], alpha=1)
plot(cv.out.lasso)
```



```
# estimate test MSE
bestlam <- cv.out.lasso$lambda.min
lasso.pred <- predict(lasso.mod, s=bestlam, newx=X[test,])
mean((lasso.pred-y.test)^2)
```

```
## [1] 0.0009281257
```

```
# fit the model with best-lambda on all the data
lasso.coef <- predict(lasso.mod,type="coefficients",s=bestlam)[1:44,]
lasso.coef
```

```
##                (Intercept)                (Intercept)
##              -0.0700086157               0.0000000000
##                        age                  potential
##              -0.0327822931               0.0478762929
##   international_reputation                  height_cm
##               0.2076411021               0.0046156413
##                  weight_kg                       pace
##               0.0006093772               0.0030624471
##                   shooting                    passing
##               0.0000000000               0.0000000000
##             preferred_foot                  weak_foot
##              -0.0010156365               0.0000000000
##                  dribbling                  defending
##               0.0000000000               0.0000000000
##                     physic          attacking_crossing
##               0.0000000000                0.0009885228
##        attacking_finishing  attacking_heading_accuracy
##               0.0174554515                0.0055992081
##    attacking_short_passing           attacking_volleys
##               0.0076089278               -0.0010496705
##            skill_dribbling                 skill_curve
##              -0.0092430676               -0.0035565897
##          skill_fk_accuracy          skill_long_passing
##               0.0092326280               -0.0024819947
##         skill_ball_control       movement_acceleration
##               0.0000000000                0.0038861309
##      movement_sprint_speed            movement_agility
##               0.0171929792               -0.0083969806
##         movement_reactions            movement_balance
##               0.0655784093                0.0094006466
##           power_shot_power               power_jumping
##              -0.0051379458               -0.0039456886
##              power_stamina              power_strength
##               0.0199717551                0.0123359522
##           power_long_shots        mentality_aggression
##              -0.0074238244               -0.0005190495
##    mentality_interceptions       mentality_positioning
##              -0.0036212838               -0.0004165908
##           mentality_vision         mentality_penalties
##               0.0099833188               -0.0131316470
##        mentality_composure defending_marking_awareness
##               0.0046010020                0.0000000000
##   defending_standing_tackle    defending_sliding_tackle
##               0.0000000000                0.0000000000
```

We check which coefficients were forced to zero

```
lasso.coef[lasso.coef!=0]
```

```
##                (Intercept)                        age
##              -0.0700086157              -0.0327822931
##                  potential   international_reputation
##               0.0478762929               0.2076411021
##                  height_cm                  weight_kg
##               0.0046156413               0.0006093772
##                       pace             preferred_foot
##               0.0030624471              -0.0010156365
##         attacking_crossing        attacking_finishing
##               0.0009885228               0.0174554515
## attacking_heading_accuracy    attacking_short_passing
##               0.0055992081               0.0076089278
##          attacking_volleys            skill_dribbling
##              -0.0010496705              -0.0092430676
##                skill_curve          skill_fk_accuracy
##              -0.0035565897               0.0092326280
##         skill_long_passing      movement_acceleration
##              -0.0024819947               0.0038861309
##      movement_sprint_speed           movement_agility
##               0.0171929792              -0.0083969806
##         movement_reactions           movement_balance
##               0.0655784093               0.0094006466
##           power_shot_power              power_jumping
##              -0.0051379458              -0.0039456886
##              power_stamina             power_strength
##               0.0199717551               0.0123359522
##           power_long_shots       mentality_aggression
##              -0.0074238244              -0.0005190495
##    mentality_interceptions      mentality_positioning
##              -0.0036212838              -0.0004165908
##           mentality_vision        mentality_penalties
##               0.0099833188              -0.0131316470
##        mentality_composure
##               0.0046010020
```

```r
length(lasso.coef[lasso.coef!=0])
```

```
## [1] 33
```

```r
which(lasso.coef==0)
```

```
##                 (Intercept)                    shooting
##                           2                           9
##                     passing                   weak_foot
##                          10                          12
##                   dribbling                   defending
##                          13                          14
##                      physic           skill_ball_control
##                          15                          25
## defending_marking_awareness   defending_standing_tackle
##                          42                          43
##    defending_sliding_tackle
##                          44
```

Shooting, passing, dribbling, defending, physic, attacking_volleys, skill_ball_control and defending_sliding_tackle were forced to be zero.

```r
# residual sum of squares
lasso.rss <- sum((lasso.pred - y.test)^2)
lasso.rss
```

```
## [1] 6.118205
```

```r
lasso.tss <- sum((y.test - mean(y.test)) ^ 2)  ## total sum of squares
lasso.rsq <- 1 - lasso.rss/lasso.tss  # R squared
lasso.rsq
```

```
## [1] 0.5598142
```

```
#ridge.rss vs lasso.rss
ridge.rss
```

```
## [1] 6.164869
```

```
lasso.rss
```

```
## [1] 6.118205
```

```
#ridge.rsq vs lasso.rsq
ridge.rsq
```

```
## [1] 0.5564568
```

```
lasso.rsq
```

```
## [1] 0.5598142
```

Overall, we can say Lasso regression performs slightly better than Ridge Regression, given the higher R squared and lower Residual Sum of Squares. From both models is clear how age and agility are the features that mostly impact the market value (negatively and positively, respectively), together with the international reputation that has the biggest positive influence of them all. The results show that the models explain around 54% of the variance for the market value. R squared value between 0.5 and 0.7 is considered a moderate effect size.

5. Conclusion and further research

All in all, position classification is possible for some distinct areas of the football field, but for some specific ones is quite impossible, in the case of multiclass classification.

On one hand, football is a very heterogeneous sport and often the values of the attributes cannot explain as a whole the position of a player since his style of play heavily influence how the role is interpreted and consequently where exactly the player acts on the field. On the other hand, we would also like to believe that with sufficient data, even effective positioning of real players could be calculated.

As regard players' market price, an R squared of 0.54 is a decent value, but not remarkable. Of course age plays a big role, since young footballers have way larger margins of improvement; but it is also true that aspects like the contract expiry, which was not provided, have a significant effect in real life. When the date of expiry of a player approaches, the club is more inclined to agree even a lower price in order not to lose this athlete for free. The feature with the undisputed heavier positive impact is international_reputation and this come as no surprise. Football players that have a huge fan base (for example on social media) have a higher price since the club can profit from his visibility, that will result in more supporters and more revenue. Market value is the result of plenty of different information, some of which are very difficult to add among the features. For example, the current form of a player greatly influence his value: a striker scoring for the previous 9 consecutive matches will surely see his value skyrocket. With additional information like these, the models would definitely improve their performance.