Work Rate Classification

Naga Santhosh Kartheek Karnati

4/3/2020

R Markdown

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see http://rmarkdown.rstudio.com.

When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document. You can embed an R code chunk like this:

```
library(ggplot2)
library(dplyr)
```

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

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

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

```
library(tidyr)
library(stringr)
library(data.table)
```

```
##
## Attaching package: 'data.table'
```

```
## The following objects are masked from 'package:dplyr':
##
##     between, first, last
```

```
library(rio)
```

```
## Warning: package 'rio' was built under R version 3.6.2
```

```
library(modelr)
library(purrr)
```

```
##
## Attaching package: 'purrr'
```

```
## The following object is masked from 'package:data.table':
##
##     transpose
```

```
fifa20 <- fread('D:/NEU/Spring 2020/SML/Project/Datasets/players_20.csv')
class(fifa20)
```

```
## [1] "data.table" "data.frame"
```

```
#View(fifa20)
#fifa20 as a tibble
fifa20 <- as_tibble(fifa20)
```

```
updated_fifa20 <- fifa20 %>% select(-player_url, -long_name, -dob, -real_face, -player_tags,
                                    -loaned_from, -joined, -player_positions, -contract_valid_until,
                                    -nation_position, -nation_jersey_number, -player_traits, -gk_diving
                                    -gk_handling, -gk_kicking, -gk_reflexes, -gk_speed, -gk_positioning
                                    -goalkeeping_diving, -goalkeeping_handling, -goalkeeping_kicking,
                                    -goalkeeping_positioning, -goalkeeping_reflexes,
                                    -ls, -st, -rs, -lw, -lf, -cf, -rf, -rw, -lam, -cam, -ram,
                                    -lm, -lcm, -cm, -rcm, -rm, -lwb, -ldm, -cdm, -rdm, -rwb,
                                    -lb, -lcb, -cb, -rcb, -rb)


# remove oberservations that have missing values (NOT processing Goalkeepers)
clean_fifa20 <- na.omit(updated_fifa20)
View(clean_fifa20)
```

Work Rate Classification

```
#Number of classes in work_rate column:
unique(clean_fifa20$work_rate)
```

```
## [1] "Medium/Low"    "High/Low"      "High/Medium"   "High/High"
## [5] "Medium/Medium" "Medium/High"   "Low/High"      "Low/Medium"
## [9] "Low/Low"
```

```
#Dimensions of df:
dim(clean_fifa20)
```

```
## [1] 15077    55
```

```
#15077 rows and 55 columns

df <- clean_fifa20 %>% select(-sofifa_id, -short_name, -nationality, -club, -body_type, -team_jersey_nu
ddff <- df
#View(df)
```

#Logistic Regression to classify work rate:

```
library(caret)
```

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

```
##
## Attaching package: 'caret'
```

```
## The following object is masked from 'package:purrr':
##
##     lift
```

```
library(nnet)
```

```
## Warning: package 'nnet' was built under R version 3.6.2
```

```
set.seed(1)
#which(is.na(df$work_rate))

training.samples <- df$work_rate %>% createDataPartition(p = 0.8, list = FALSE)
train.data  <- df[training.samples, ]
test.data <- df[-training.samples, ]
dim(train.data)
```

```
## [1] 12065    48
```

```
dim(test.data)
```

```
## [1] 3012   48
```

#Multinomial logistic regression:

```
# Fit the model
model <- nnet::multinom(work_rate ~., data = train.data)
```

```
## # weights:  441 (384 variable)
## initial  value 26509.514526
## iter  10 value 25284.586736
## iter  20 value 25099.939279
## iter  30 value 19716.333875
## iter  40 value 17094.005476
## iter  50 value 16747.668821
## iter  60 value 16684.510832
## iter  70 value 16650.176822
## iter  80 value 16632.790617
## iter  90 value 16627.570024
## iter  90 value 16627.570024
## iter 100 value 16102.888036
## final  value 16102.888036
## stopped after 100 iterations
```

```r
# Summarize the model
summary(model)
```

```
## Call:
## nnet::multinom(formula = work_rate ~ ., data = train.data)
##
## Coefficients:
##                 (Intercept)          age   height_cm   weight_kg     overall
## High/Low      -0.0001402089  0.043211150 -0.01051337 -0.02830579  0.16374485
## High/Medium   -0.0025233032  0.033100807  0.02036206 -0.02563006 -0.05583033
## Low/High      -0.0021557277 -0.004560441 -0.01564582 -0.02073468  0.24580280
## Low/Low       -0.0015431888 -0.077286861 -0.02611835 -0.01930425  0.26047357
## Low/Medium     0.0008154242  0.033539629  0.03866870 -0.04209379  0.13430600
## Medium/High    0.0010215275  0.073806126  0.01996770 -0.02247515 -0.07774452
## Medium/Low    -0.0026665091 -0.018819776  0.02877101  0.00903318  0.21324531
## Medium/Medium  0.0115592511 -0.005826290  0.04340167 -0.01584540  0.05288054
##                  potential     value_eur      wage_eur preferred_footRight
## High/Low       0.04493965  2.015198e-07 -6.815239e-05        -0.049131707
## High/Medium    0.04003357  1.501158e-07 -4.394613e-05        -0.127785831
## Low/High       0.01248751 -4.153537e-07  3.015372e-05         0.045218106
## Low/Low       -0.08342467  1.245141e-07 -9.964797e-05         0.006748660
## Low/Medium     0.02351907  4.839949e-07 -9.939249e-05         0.025217669
## Medium/High    0.07890292  2.667771e-07 -6.269813e-05         0.008397495
## Medium/Low    -0.02945541  7.765282e-08 -5.340677e-06        -0.010906047
## Medium/Medium  0.03534673  1.920303e-07 -5.455867e-05         0.018326855
##                international_reputation   weak_foot  skill_moves
## High/Low                     0.022880333 -0.06009576  0.020279258
## High/Medium                  0.014960883  0.07006000  0.055760947
## Low/High                     0.001298393 -0.10098826  0.071013402
## Low/Low                     -0.009533299 -0.01509292  0.014139444
## Low/Medium                  -0.005135545 -0.10832186  0.048192950
## Medium/High                  0.050861404  0.05982293 -0.150096709
## Medium/Low                  -0.017785828 -0.10448934  0.009194582
## Medium/Medium               -0.058881178  0.01999343 -0.020780833
##                release_clause_eur         pace     shooting      passing
## High/Low           -6.745638e-08  0.023703172  0.016966778  0.006242356
## High/Medium         5.617183e-09 -0.055614794  0.007501359 -0.004481687
## Low/High           -2.947227e-08 -0.017046440 -0.021308004 -0.008342107
## Low/Low            -3.105108e-07  0.001207594 -0.003325573 -0.009896339
## Low/Medium         -1.752153e-07  0.008505189  0.003380696  0.016120943
## Medium/High        -2.462872e-08 -0.016212815 -0.025743727 -0.026278474
## Medium/Low         -6.707129e-08  0.019966638 -0.026720145  0.037536511
## Medium/Medium      -2.313176e-08  0.027545364 -0.026464914 -0.007019354
##                   dribbling    defending       physic attacking_crossing
## High/Low       0.013367419 -0.021793095 -0.030097258       -0.011350008
## High/Medium    0.023072674  0.019778113 -0.018491777        0.042752185
## Low/High      -0.024785547 -0.004171846  0.005130914       -0.026424686
## Low/Low       -0.015049198 -0.005640227 -0.004686565       -0.013575375
## Low/Medium    -0.015480459 -0.040134278 -0.011827299       -0.011276703
## Medium/High   -0.008760938 -0.025898396  0.014956306        0.001749031
## Medium/Low     0.018826725  0.017941552 -0.014031302       -0.014076210
## Medium/Medium  0.039157190 -0.024192493  0.001996320        0.001097342
##                attacking_finishing attacking_heading_accuracy
```

```
## High/Low            -0.0235919313                0.030792452
## High/Medium         -0.0048250991                0.016059308
## Low/High            -0.0301269954               -0.001981247
## Low/Low             -0.0123147274                0.006529656
## Low/Medium          -0.0341506587                0.007051338
## Medium/High         -0.0008386307               -0.005550834
## Medium/Low          -0.0097520026                0.010422682
## Medium/Medium       -0.0024102399               -0.001907209
##                attacking_short_passing attacking_volleys skill_dribbling
## High/Low                   -0.032277137      -0.026254385    -0.014761860
## High/Medium                -0.002055580      -0.035943955    -0.026816342
## Low/High                    0.012049454      -0.013464637    -0.048562180
## Low/Low                     0.014725693      -0.009785479    -0.032312804
## Low/Medium                  0.005635017      -0.014075704    -0.052344088
## Medium/High                 0.030375417      -0.011993180    -0.008051849
## Medium/Low                 -0.024824912      -0.004990908    -0.006993418
## Medium/Medium               0.030313496      -0.018532513    -0.048860373
##                 skill_curve skill_fk_accuracy skill_long_passing
## High/Low       0.011445842       0.009768773        0.018083511
## High/Medium    0.021460164       0.008633608       -0.013655289
## Low/High       0.007498616       0.003495574        0.018846380
## Low/Low        0.009965079       0.008030529        0.012208900
## Low/Medium     0.003405782       0.006860138        0.017325659
## Medium/High   -0.006411046       0.007185535        0.016366390
## Medium/Low     0.026930228       0.009357238        0.006139485
## Medium/Medium -0.004989447       0.011507192        0.003595852
##                skill_ball_control movement_acceleration movement_sprint_speed
## High/Low              0.048324912          -0.013263379           0.007541276
## High/Medium           0.034158961           0.041769190           0.036182302
## Low/High              0.022687106           0.014219070          -0.031695487
## Low/Low               0.018001140           0.012059866          -0.001756952
## Low/Medium            0.040449492          -0.023521083          -0.021186856
## Medium/High           0.055734142           0.003186822          -0.008769378
## Medium/Low            0.006774868          -0.048315781          -0.005366412
## Medium/Medium         0.040607316          -0.031619347          -0.023915522
##                movement_agility movement_reactions movement_balance
## High/Low            0.016316484        0.006461997     -0.024772910
## High/Medium         0.005431419        0.010586556     -0.007835031
## Low/High            0.022813662        0.012327003     -0.003534824
## Low/Low             0.017613300        0.032670441     -0.013062211
## Low/Medium          0.010073053        0.005409751      0.004267711
## Medium/High         0.018501426        0.009996714      0.006524879
## Medium/Low          0.012054060        0.011344217     -0.006954999
## Medium/Medium       0.007928704        0.026947709      0.003152736
##                power_shot_power power_jumping power_stamina power_strength
## High/Low         -0.0008784101  1.213455e-03   -0.07822690     0.02944381
## High/Medium       0.0064120153 -5.924244e-03   -0.04561305     0.03075789
## Low/High         -0.0005339260  1.879885e-02   -0.04942552     0.02743667
## Low/Low           0.0043135720  7.436038e-03   -0.05778268     0.03812561
## Low/Medium       -0.0029427283  1.174794e-03   -0.07372380     0.02762514
## Medium/High      -0.0023281601  1.442506e-02   -0.05613869     0.02772360
## Medium/Low       -0.0079354587 -5.300877e-05   -0.11488173     0.01675409
## Medium/Medium     0.0009500273  6.602905e-03   -0.08395882     0.02090644
##                power_long_shots mentality_aggression mentality_interceptions
```

```
## High/Low            0.01189720         -0.044941912            -0.013327391
## High/Medium         0.01776691         -0.024857198            -0.018267677
## Low/High            0.02557901         -0.017540035             0.006750197
## Low/Low             0.02174627         -0.019063985            -0.014041449
## Low/Medium          0.02100620         -0.035319087             0.016919782
## Medium/High         0.02927062         -0.007316217             0.002808735
## Medium/Low          0.01273735         -0.049530855            -0.022665871
## Medium/Medium       0.02433291         -0.040560570             0.002522708
##               mentality_positioning mentality_vision mentality_penalties
## High/Low                -0.04931319      -0.03120501         0.009557597
## High/Medium              0.01630847      -0.03455700        -0.009032187
## Low/High                -0.06445305      -0.03638379         0.007323003
## Low/Low                 -0.04892339      -0.01838259        -0.008216561
## Low/Medium              -0.07099998      -0.03692163         0.003321492
## Medium/High             -0.06656996      -0.02340861         0.001451828
## Medium/Low              -0.05704769      -0.01883735        -0.002944608
## Medium/Medium           -0.05165924      -0.02287533         0.003120851
##               mentality_composure defending_marking defending_standing_tackle
## High/Low             -0.001936813     -0.0155273431              -0.061550404
## High/Medium          -0.023136045     -0.0188853100              -0.037858597
## Low/High             -0.014537877     -0.0122131734              -0.002300339
## Low/Low               0.008685724     -0.0058866093              -0.032502250
## Low/Medium           -0.006422966     -0.0009787778              -0.015650521
## Medium/High          -0.009587054      0.0025303146              -0.018143062
## Medium/Low            0.005124669     -0.0221094748              -0.066761523
## Medium/Medium        -0.029569284     -0.0060406446              -0.020282387
##               defending_sliding_tackle
## High/Low                    0.03774838
## High/Medium                 0.03019895
## Low/High                   -0.01668514
## Low/Low                     0.03105122
## Low/Medium                  0.01170830
## Medium/High                 0.02224317
## Medium/Low                  0.02638801
## Medium/Medium               0.02633706
##
## Std. Errors:
##                (Intercept)          age    height_cm    weight_kg      overall
## High/Low      2.481190e-10 6.985323e-09 4.481635e-08 1.861367e-08 1.683357e-08
## High/Medium   6.497308e-11 1.800928e-09 1.173268e-08 4.842678e-09 4.319730e-09
## Low/High      9.789848e-11 2.497273e-09 1.847629e-08 7.999552e-09 6.486758e-09
## Low/Low       1.047346e-11 3.189022e-10 1.898401e-09 7.980493e-10 6.868551e-10
## Low/Medium    7.279196e-10 2.100351e-08 1.342956e-07 5.698822e-08 4.819152e-08
## Medium/High   1.070688e-10 3.101168e-09 1.960804e-08 8.312804e-09 7.233616e-09
## Medium/Low    1.083798e-10 2.960880e-09 1.974907e-08 8.326844e-09 7.454722e-09
## Medium/Medium 3.193104e-10 8.691691e-09 5.922619e-08 2.502571e-08 2.055862e-08
##                  potential    value_eur     wage_eur preferred_footRight
## High/Low      1.739502e-08 9.924908e-08 7.364387e-06        1.952945e-10
## High/Medium   4.477526e-09 5.984941e-08 3.685099e-06        4.919805e-11
## Low/High      7.045504e-09 2.443006e-07 8.378052e-06        8.272491e-11
## Low/Low       6.875421e-10 6.962933e-07 1.372176e-07        7.921087e-12
## Low/Medium    4.990679e-08 1.513844e-07 1.136697e-05        5.551296e-10
## Medium/High   7.460569e-09 7.001693e-08 4.570675e-06        8.258830e-11
## Medium/Low    7.798497e-09 1.054359e-07 5.145233e-06        8.297813e-11
```

```
## Medium/Medium 2.186243e-08 6.102552e-08 3.612280e-06       2.428881e-10
##               international_reputation    weak_foot  skill_moves
## High/Low                  2.988306e-10 7.699492e-10 6.947309e-10
## High/Medium               9.219433e-11 1.985133e-10 1.727952e-10
## Low/High                  1.069368e-10 2.505027e-10 1.673378e-10
## Low/Low                   1.137259e-11 3.095806e-11 2.467133e-11
## Low/Medium                8.213735e-10 2.044517e-09 1.547108e-09
## Medium/High               1.388521e-10 3.104814e-10 2.309057e-10
## Medium/Low                1.152339e-10 3.320989e-10 2.971320e-10
## Medium/Medium             3.031698e-10 8.723819e-10 6.282671e-10
##               release_clause_eur         pace     shooting      passing
## High/Low            5.131993e-08 1.742280e-08 1.541160e-08 1.444428e-08
## High/Medium         3.077393e-08 4.539064e-09 3.670027e-09 3.832225e-09
## Low/High            1.292793e-07 5.167977e-09 2.595259e-09 3.791443e-09
## Low/Low             4.105716e-07 6.627812e-10 5.321871e-10 5.865272e-10
## Low/Medium          7.963254e-08 3.954910e-08 3.025491e-08 3.740397e-08
## Medium/High         3.559370e-08 6.342772e-09 4.746839e-09 5.921643e-09
## Medium/Low          5.517071e-08 7.053822e-09 6.315480e-09 6.234264e-09
## Medium/Medium       3.135180e-08 1.652776e-08 1.171714e-08 1.506839e-08
##                  dribbling    defending       physic attacking_crossing
## High/Low      1.658649e-08 9.345797e-09 1.555592e-08       1.399750e-08
## High/Medium   4.201595e-09 3.297571e-09 4.223777e-09       3.910106e-09
## Low/High      3.885530e-09 6.482378e-09 7.286521e-09       2.810548e-09
## Low/Low       6.196220e-10 5.842299e-10 7.213756e-10       5.654031e-10
## Low/Medium    3.842538e-08 4.538922e-08 4.997301e-08       3.320221e-08
## Medium/High   6.167177e-09 6.933424e-09 7.548697e-09       5.440114e-09
## Medium/Low    7.001429e-09 4.724201e-09 6.794049e-09       5.714145e-09
## Medium/Medium 1.548215e-08 2.014078e-08 2.169341e-08       1.259865e-08
##               attacking_finishing attacking_heading_accuracy
## High/Low                1.534087e-08                1.502120e-08
## High/Medium             3.522140e-09                3.790999e-09
## Low/High                2.105523e-09                6.667269e-09
## Low/Low                 4.930341e-10                6.305343e-10
## Low/Medium              2.641628e-08                4.632261e-08
## Medium/High             4.107042e-09                6.787212e-09
## Medium/Low              6.163991e-09                6.657889e-09
## Medium/Medium           1.010259e-08                2.014298e-08
##               attacking_short_passing attacking_volleys skill_dribbling
## High/Low                  1.548950e-08      1.413882e-08    1.647602e-08
## High/Medium               4.063864e-09      3.307632e-09    4.151008e-09
## Low/High                  5.149304e-09      2.135964e-09    3.110183e-09
## Low/Low                   6.357542e-10      4.828910e-10    5.929831e-10
## Low/Medium                4.315567e-08      2.689723e-08    3.534805e-08
## Medium/High               6.710352e-09      4.268621e-09    5.838945e-09
## Medium/Low                6.840704e-09      5.782823e-09    6.924678e-09
## Medium/Medium             1.804003e-08      1.025115e-08    1.382812e-08
##                skill_curve skill_fk_accuracy skill_long_passing
## High/Low      1.411789e-08      1.246369e-08       1.324704e-08
## High/Medium   3.713373e-09      3.262040e-09       3.608170e-09
## Low/High      2.567869e-09      2.486845e-09       4.697763e-09
## Low/Low       5.389447e-10      4.931308e-10       5.778446e-10
## Low/Medium    3.082318e-08      2.877352e-08       3.947994e-08
## Medium/High   4.922362e-09      4.553555e-09       6.211962e-09
## Medium/Low    5.986199e-09      5.359580e-09       5.927581e-09
```

```
## Medium/Medium 1.160370e-08      1.114448e-08       1.638943e-08
##               skill_ball_control movement_acceleration movement_sprint_speed
## High/Low            1.663500e-08          1.734489e-08          1.748890e-08
## High/Medium         4.212053e-09          4.521409e-09          4.554401e-09
## Low/High            4.580898e-09          5.104001e-09          5.218686e-09
## Low/Low             6.335004e-10          6.589105e-10          6.658961e-10
## Low/Medium          4.147298e-08          3.911726e-08          3.988667e-08
## Medium/High         6.482038e-09          6.304005e-09          6.368919e-09
## Medium/Low          7.136308e-09          6.990916e-09          7.100094e-09
## Medium/Medium       1.707633e-08          1.630185e-08          1.670696e-08
##               movement_agility movement_reactions movement_balance
## High/Low          1.734468e-08       1.577676e-08     1.646484e-08
## High/Medium       4.441833e-09       4.054650e-09     4.326145e-09
## Low/High          4.589793e-09       5.739153e-09     4.698859e-09
## Low/Low           6.696545e-10       6.539145e-10     6.624595e-10
## Low/Medium        3.989649e-08       4.470847e-08     4.116952e-08
## Medium/High       6.397025e-09       6.775695e-09     6.479098e-09
## Medium/Low        7.064831e-09       6.979813e-09     6.827113e-09
## Medium/Medium     1.634898e-08       1.883726e-08     1.725909e-08
##               power_shot_power power_jumping power_stamina power_strength
## High/Low          1.647952e-08  1.646752e-08  1.561295e-08   1.635192e-08
## High/Medium       4.126703e-09  4.293690e-09  4.349966e-09   4.305759e-09
## Low/High          4.105973e-09  6.922792e-09  6.679370e-09   7.976395e-09
## Low/Low           6.254710e-10  7.235677e-10  7.065453e-10   7.408383e-10
## Low/Medium        3.939233e-08  4.870820e-08  4.518240e-08   5.307848e-08
## Medium/High       6.115236e-09  7.619466e-09  7.090235e-09   7.774992e-09
## Medium/Low        7.026986e-09  6.873214e-09  6.285124e-09   7.377338e-09
## Medium/Medium     1.592187e-08  2.083371e-08  1.926755e-08   2.340077e-08
##               power_long_shots mentality_aggression mentality_interceptions
## High/Low          1.471110e-08         1.324814e-08            8.798858e-09
## High/Medium       3.618191e-09         3.885011e-09            3.252384e-09
## Low/High          2.443400e-09         6.542001e-09            6.299242e-09
## Low/Low           5.315509e-10         6.892654e-10            5.781961e-10
## Low/Medium        2.975250e-08         4.840237e-08            4.475751e-08
## Medium/High       4.852384e-09         7.541849e-09            6.892113e-09
## Medium/Low        6.114569e-09         6.013354e-09            4.456170e-09
## Medium/Medium     1.122840e-08         2.066507e-08            1.977481e-08
##               mentality_positioning mentality_vision mentality_penalties
## High/Low              1.585423e-08     1.449438e-08        1.514716e-08
## High/Medium           3.911430e-09     3.701742e-09        3.583414e-09
## Low/High              2.275512e-09     3.021999e-09        3.218307e-09
## Low/Low               5.495844e-10     5.614078e-10        5.423497e-10
## Low/Medium            2.978644e-08     3.367464e-08        3.357060e-08
## Medium/High           4.784800e-09     5.406117e-09        5.074692e-09
## Medium/Low            6.343808e-09     6.208349e-09        6.236126e-09
## Medium/Medium         1.123194e-08     1.322671e-08        1.350638e-08
##               mentality_composure defending_marking defending_standing_tackle
## High/Low            1.546903e-08      8.938623e-09              8.617044e-09
## High/Medium         3.938047e-09      3.197477e-09              3.335820e-09
## Low/High            5.467035e-09      6.469823e-09              6.577947e-09
## Low/Low             6.439901e-10      5.726219e-10              5.887396e-10
## Low/Medium          4.379514e-08      4.488274e-08              4.613966e-08
## Medium/High         6.648831e-09      6.833520e-09              7.099146e-09
## Medium/Low          6.947544e-09      4.665330e-09              4.500840e-09
```

```
## Medium/Medium        1.819508e-08      2.002209e-08              2.055140e-08
##               defending_sliding_tackle
## High/Low                  8.043138e-09
## High/Medium               3.222121e-09
## Low/High                  6.274371e-09
## Low/Low                   5.681274e-10
## Low/Medium                4.462863e-08
## Medium/High               6.910970e-09
## Medium/Low                4.339825e-09
## Medium/Medium             1.987899e-08
##
## Residual Deviance: 32205.78
## AIC: 32973.78
```

```
# Make predictions
predicted.classes <- model %>% predict(test.data)
head(predicted.classes)
```

```
## [1] High/High   High/High   Medium/High High/High   Medium/High High/High
## 9 Levels: High/High High/Low High/Medium Low/High Low/Low ... Medium/Medium
```

```
# Model accuracy
mean(predicted.classes == test.data$work_rate)
```

```
## [1] 0.5126162
```

```
#Accuracy of 51%
```

#LDA Classification:

```
trCtrl <- trainControl(method = "cv", number = 5)
fit_wrate <- train(work_rate~., data=train.data, method="lda",
              trControl = trCtrl, metric = "Accuracy")


pred_wrate <- predict(fit_wrate, test.data%>%select(-work_rate))
comparison <- data.frame(original = test.data$work_rate, pred = pred_wrate)

#accuarcy of cross validated LDA model:
mean(comparison$pred == test.data$work_rate)
```

```
## [1] 0.499004
```

```
#50% accuracy

#confusion matrix:
confusionMatrix(as.factor(test.data$work_rate), comparison$pred)
```

```
## Confusion Matrix and Statistics
##
```

```
##                Reference
## Prediction      High/High High/Low High/Medium Low/High Low/Low Low/Medium
##   High/High            48        4          44        0       0          0
##   High/Low              2       17          32        0       0          0
##   High/Medium          19       26         201        2       0          2
##   Low/High              1        0           0       13       0          5
##   Low/Low               0        0           0        0       0          0
##   Low/Medium            0        0           0       11       0         19
##   Medium/High           8        1          14       18       0         10
##   Medium/Low            1       17          39        1       0          1
##   Medium/Medium        18       23         131       34       1         39
##                Reference
## Prediction      Medium/High Medium/Low Medium/Medium
##   High/High              13          3            74
##   High/Low                0          6            64
##   High/Medium             9         12           324
##   Low/High               12          0            46
##   Low/Low                 0          2             4
##   Low/Medium              7          0            48
##   Medium/High            66          0           206
##   Medium/Low              0         10            92
##   Medium/Medium          56         27          1129
##
## Overall Statistics
##
##                Accuracy : 0.499
##                  95% CI : (0.481, 0.517)
##     No Information Rate : 0.6597
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.216
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: High/High Class: High/Low Class: High/Medium
## Sensitivity                   0.49485        0.193182            0.43601
## Specificity                   0.95266        0.964432            0.84555
## Pos Pred Value                0.25806        0.140496            0.33782
## Neg Pred Value                0.98266        0.975441            0.89243
## Prevalence                    0.03220        0.029216            0.15305
## Detection Rate                0.01594        0.005644            0.06673
## Detection Prevalence          0.06175        0.040173            0.19754
## Balanced Accuracy             0.72375        0.578807            0.64078
##                      Class: Low/High Class: Low/Low Class: Low/Medium
## Sensitivity                 0.164557       0.000000          0.250000
## Specificity                 0.978179       0.998007          0.977520
## Pos Pred Value              0.168831       0.000000          0.223529
## Neg Pred Value              0.977513       0.999667          0.980526
## Prevalence                  0.026228       0.000332          0.025232
## Detection Rate              0.004316       0.000000          0.006308
## Detection Prevalence        0.025564       0.001992          0.028220
## Balanced Accuracy           0.571368       0.499004          0.613760
```

```
##                      Class: Medium/High Class: Medium/Low Class: Medium/Medium
## Sensitivity                     0.40491          0.16667               0.5682
## Specificity                     0.90979          0.94885               0.6790
## Pos Pred Value                  0.20433          0.06211               0.7743
## Neg Pred Value                  0.96393          0.98246               0.4479
## Prevalence                      0.05412          0.01992               0.6597
## Detection Rate                  0.02191          0.00332               0.3748
## Detection Prevalence            0.10724          0.05345               0.4841
## Balanced Accuracy               0.65735          0.55776               0.6236
```

```
#number of records per class: is there too much class imbalance?
df %>% count(work_rate)
```

```
## # A tibble: 9 x 2
##   work_rate         n
##   <chr>         <int>
## 1 High/High       932
## 2 High/Low        609
## 3 High/Medium    2975
## 4 Low/High        386
## 5 Low/Low          30
## 6 Low/Medium      428
## 7 Medium/High    1618
## 8 Medium/Low      807
## 9 Medium/Medium  7292
```

```
#Yes, there is a significant class imbalance

#grouping together a few classes into one class:
df$work_rate[df$work_rate == "Low/Low"] <- "Low/Medium"
df$work_rate[df$work_rate == "Low/High"] <- "Low/Medium"
#number of records per class
df %>% count(work_rate)
```

```
## # A tibble: 7 x 2
##   work_rate         n
##   <chr>         <int>
## 1 High/High       932
## 2 High/Low        609
## 3 High/Medium    2975
## 4 Low/Medium      844
## 5 Medium/High    1618
## 6 Medium/Low      807
## 7 Medium/Medium  7292
```

```
library(pROC)
```

```
## Warning: package 'pROC' was built under R version 3.6.2
```

```
## Type 'citation("pROC")' for a citation.
```

```
##
## Attaching package: 'pROC'
```

```
## The following objects are masked from 'package:stats':
##
##     cov, smooth, var
```

```
#LDA classification on new df:
set.seed(2)
#which(is.na(df$work_rate))
training.samples <- df$work_rate %>% createDataPartition(p = 0.8, list = FALSE)
train.data  <- df[training.samples, ]
test.data <- df[-training.samples, ]
dim(train.data)
```

```
## [1] 12065    48
```

```
dim(test.data)
```

```
## [1] 3012   48
```

```
trCtrl <- trainControl(method = "cv", number = 5)
fit_wrate <- train(work_rate~., data=train.data, method="lda",
              trControl = trCtrl, metric = "Accuracy")


pred_wrate <- predict(fit_wrate, test.data%>%select(-work_rate))
comparison <- data.frame(original = test.data$work_rate, pred = pred_wrate)

#accuarcy of cross validated LDA model:
mean(comparison$pred == test.data$work_rate)
```

```
## [1] 0.5136122
```

```
#51% accuracy

#confusion matrix:
confusionMatrix(as.factor(test.data$work_rate), comparison$pred)
```

```
## Confusion Matrix and Statistics
##
##                Reference
## Prediction      High/High High/Low High/Medium Low/Medium Medium/High
##   High/High            43        0          57          1           5
##   High/Low              3       16          37          1           0
##   High/Medium          21       12         213          0           2
##   Low/Medium            1        0           0         62          14
##   Medium/High          13        0          15         60          50
##   Medium/Low            0        6          41          0           0
##   Medium/Medium        20       15         133         87          25
##                Reference
## Prediction      Medium/Low Medium/Medium
##   High/High              1            79
##   High/Low               9            55
```

```
##   High/Medium           13           334
##   Low/Medium             1            90
##   Medium/High            0           185
##   Medium/Low            14           100
##   Medium/Medium         29          1149
##
## Overall Statistics
##
##                Accuracy : 0.5136
##                  95% CI : (0.4956, 0.5316)
##     No Information Rate : 0.6614
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.2354
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: High/High Class: High/Low Class: High/Medium
## Sensitivity                   0.42574        0.326531            0.42944
## Specificity                   0.95088        0.964563            0.84817
## Pos Pred Value                0.23118        0.132231            0.35798
## Neg Pred Value                0.97948        0.988585            0.88291
## Prevalence                    0.03353        0.016268            0.16467
## Detection Rate                0.01428        0.005312            0.07072
## Detection Prevalence          0.06175        0.040173            0.19754
## Balanced Accuracy             0.68831        0.645547            0.63880
##                      Class: Low/Medium Class: Medium/High Class: Medium/Low
## Sensitivity                    0.29384            0.52083          0.208955
## Specificity                    0.96216            0.90638          0.950085
## Pos Pred Value                 0.36905            0.15480          0.086957
## Neg Pred Value                 0.94761            0.98289          0.981410
## Prevalence                     0.07005            0.03187          0.022244
## Detection Rate                 0.02058            0.01660          0.004648
## Detection Prevalence           0.05578            0.10724          0.053453
## Balanced Accuracy              0.62800            0.71361          0.579520
##                      Class: Medium/Medium
## Sensitivity                        0.5768
## Specificity                        0.6971
## Pos Pred Value                     0.7881
## Neg Pred Value                     0.4575
## Prevalence                         0.6614
## Detection Rate                     0.3815
## Detection Prevalence               0.4841
## Balanced Accuracy                  0.6369
```

```
pred_wrate1 <- predict(fit_wrate, test.data%>%select(-work_rate), type="prob")
###ROC curve:
multiclass.roc(test.data$work_rate, pred_wrate1)
```

```
##
## Call:
## multiclass.roc.default(response = test.data$work_rate, predictor = pred_wrate1)
```

```
##
## Data: multivariate predictor pred_wrate1 with 7 levels of test.data$work_rate: High/High, High/Low, H
## Multi-class area under the curve: 0.8104
```

```
#Multi-class area under the curve: 0.8104
```

#QDA Classification:

```
fit_wrate_qda <- train(work_rate~., data=train.data, method="qda",
              trControl = trCtrl, metric = "Accuracy")


pred_wrate_qda <- predict(fit_wrate_qda, test.data%>%select(-work_rate))
comparison_qda <- data.frame(original = test.data$work_rate, pred_qda = pred_wrate_qda)

#accuarcy of cross validated LDA model:
mean(comparison_qda$pred_qda == test.data$work_rate)
```

```
## [1] 0.4243028
```

```
#42% accuracy

#confusion matrix:
confusionMatrix(as.factor(test.data$work_rate), comparison_qda$pred_qda)
```

```
## Confusion Matrix and Statistics
##
##                Reference
## Prediction      High/High High/Low High/Medium Low/Medium Medium/High
##   High/High            52        5          27          5          17
##   High/Low              3       21          20          1           0
##   High/Medium          47       29         138          9          27
##   Low/Medium            2        0           0         94          29
##   Medium/High          27        1           9         89          91
##   Medium/Low            2       15          14          1           1
##   Medium/Medium        55       36         101        228         101
##                Reference
## Prediction      Medium/Low Medium/Medium
##   High/High             12            68
##   High/Low              31            45
##   High/Medium           74           271
##   Low/Medium             4            39
##   Medium/High            6           100
##   Medium/Low            52            76
##   Medium/Medium        107           830
##
## Overall Statistics
##
##                Accuracy : 0.4243
##                  95% CI : (0.4066, 0.4422)
##     No Information Rate : 0.4744
##     P-Value [Acc > NIR] : 1
```

```
##
##                   Kappa : 0.203
##
##  Mcnemar's Test P-Value : <2e-16
##
## Statistics by Class:
##
##                      Class: High/High Class: High/Low Class: High/Medium
## Sensitivity                   0.27660        0.196262            0.44660
## Specificity                   0.95255        0.965577            0.83093
## Pos Pred Value                0.27957        0.173554            0.23193
## Neg Pred Value                0.95188        0.970253            0.92925
## Prevalence                    0.06242        0.035525            0.10259
## Detection Rate                0.01726        0.006972            0.04582
## Detection Prevalence          0.06175        0.040173            0.19754
## Balanced Accuracy             0.61457        0.580919            0.63877
##                      Class: Low/Medium Class: Medium/High Class: Medium/Low
## Sensitivity                    0.22014            0.34211           0.18182
## Specificity                    0.97137            0.91551           0.96001
## Pos Pred Value                 0.55952            0.28173           0.32298
## Neg Pred Value                 0.88291            0.93492           0.91792
## Prevalence                     0.14177            0.08831           0.09495
## Detection Rate                 0.03121            0.03021           0.01726
## Detection Prevalence           0.05578            0.10724           0.05345
## Balanced Accuracy              0.59576            0.62881           0.57092
##                      Class: Medium/Medium
## Sensitivity                        0.5808
## Specificity                        0.6033
## Pos Pred Value                     0.5693
## Neg Pred Value                     0.6145
## Prevalence                         0.4744
## Detection Rate                     0.2756
## Detection Prevalence               0.4841
## Balanced Accuracy                  0.5921
```

```
pred_wrate_qda1 <- predict(fit_wrate_qda, test.data%>%select(-work_rate), type="prob")
###ROC curve:
multiclass.roc(test.data$work_rate, pred_wrate_qda1)
```

```
##
## Call:
## multiclass.roc.default(response = test.data$work_rate, predictor = pred_wrate_qda1)
##
## Data: multivariate predictor pred_wrate_qda1 with 7 levels of test.data$work_rate: High/High, High/L
## Multi-class area under the curve: 0.7692
```

```
#Multi-class area under the curve: 0.7692
```

#Decision tree classification:

```
fit_wrate_dtree = train(work_rate ~ .,
                  data=train.data,
                  method="rpart",
```

```
                  trControl = trCtrl,
                  metric = "Accuracy")

pred_wrate_dtree <- predict(fit_wrate_dtree, test.data%>%select(-work_rate))
comparison_dtree <- data.frame(original = test.data$work_rate, pred_dtree = pred_wrate_dtree)

#accuarcy of cross validated LDA model:
mean(comparison_dtree$pred_dtree == test.data$work_rate)
```

```
## [1] 0.5063081
```

```
#50% accuracy

#confusion matrix:
confusionMatrix(as.factor(test.data$work_rate), comparison_dtree$pred_dtree)
```

```
## Confusion Matrix and Statistics
##
##                 Reference
## Prediction      High/High High/Low High/Medium Low/Medium Medium/High
##   High/High            12        0          52          0           0
##   High/Low              0        0          26          0           0
##   High/Medium          12        0         158          0           0
##   Low/Medium            0        0           0          0           0
##   Medium/High           3        0          12          0           0
##   Medium/Low            0        0          21          0           0
##   Medium/Medium         4        0          99          0           0
##                 Reference
## Prediction      Medium/Low Medium/Medium
##   High/High               0           122
##   High/Low                0            95
##   High/Medium             0           425
##   Low/Medium              0           168
##   Medium/High             0           308
##   Medium/Low              0           140
##   Medium/Medium           0          1355
##
## Overall Statistics
##
##                 Accuracy : 0.5063
##                   95% CI : (0.4883, 0.5243)
##     No Information Rate : 0.8675
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.1109
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: High/High Class: High/Low Class: High/Medium
## Sensitivity                  0.387097              NA            0.42935
```

```
## Specificity                    0.941630         0.95983            0.83472
## Pos Pred Value                 0.064516              NA            0.26555
## Neg Pred Value                 0.993277              NA            0.91312
## Prevalence                     0.010292         0.00000            0.12218
## Detection Rate                 0.003984         0.00000            0.05246
## Detection Prevalence           0.061753         0.04017            0.19754
## Balanced Accuracy              0.664364              NA            0.63203
##                       Class: Low/Medium Class: Medium/High Class: Medium/Low
## Sensitivity                          NA                 NA                NA
## Specificity                     0.94422             0.8928           0.94655
## Pos Pred Value                       NA                 NA                NA
## Neg Pred Value                       NA                 NA                NA
## Prevalence                      0.00000             0.0000           0.00000
## Detection Rate                  0.00000             0.0000           0.00000
## Detection Prevalence            0.05578             0.1072           0.05345
## Balanced Accuracy                    NA                 NA                NA
##                       Class: Medium/Medium
## Sensitivity                         0.5186
## Specificity                         0.7419
## Pos Pred Value                      0.9294
## Neg Pred Value                      0.1905
## Prevalence                          0.8675
## Detection Rate                      0.4499
## Detection Prevalence                0.4841
## Balanced Accuracy                   0.6302
```

```
#We are seeing very low accuracies in our model.
#This maybe because we are including all the features in the dataset to predict
#work rate.
#We need to perform feature selection or dimensionality reduction to get better
#accuracies in our models.
```

####PCA:

```
dim(df)
```

```
## [1] 15077    48
```

```
str(df)
```

```
## Classes 'tbl_df', 'tbl' and 'data.frame':    15077 obs. of  48 variables:
##  $ age                     : int  32 34 27 28 28 27 33 27 20 28 ...
##  $ height_cm               : int  170 187 175 175 181 193 172 175 178 187 ...
##  $ weight_kg               : int  72 83 68 74 70 92 66 71 73 89 ...
##  $ overall                 : int  94 93 92 91 91 90 90 90 89 89 ...
##  $ potential               : int  94 93 92 91 91 91 90 90 95 91 ...
##  $ value_eur               : int  95500000 58500000 105500000 90000000 90000000 78000000 45000000 8
##  $ wage_eur                : int  565000 405000 290000 470000 370000 200000 340000 240000 155000 15
##  $ preferred_foot          : chr  "Left" "Right" "Right" "Right" ...
##  $ international_reputation  : int  5 5 5 4 4 3 4 3 3 3 ...
##  $ weak_foot               : int  4 4 5 4 5 3 4 3 4 3 ...
##  $ skill_moves             : int  4 5 5 4 4 2 4 4 5 2 ...
##  $ work_rate               : chr  "Medium/Low" "High/Low" "High/Medium" "High/Medium" ...
```

```
##  $ release_clause_eur        : int  195800000 96500000 195200000 184500000 166500000 150200000 92300
##  $ pace                      : int  87 90 91 91 76 77 74 93 96 71 ...
##  $ shooting                  : int  92 93 85 83 86 60 76 86 84 28 ...
##  $ passing                   : int  92 82 87 86 92 70 89 81 78 54 ...
##  $ dribbling                 : int  96 89 95 94 86 71 89 89 90 67 ...
##  $ defending                 : int  39 35 32 35 61 90 72 45 39 89 ...
##  $ physic                    : int  66 78 58 66 78 86 66 74 75 87 ...
##  $ attacking_crossing        : int  88 84 87 81 93 53 86 79 78 30 ...
##  $ attacking_finishing       : int  95 94 87 84 82 52 72 90 89 22 ...
##  $ attacking_heading_accuracy: int  70 89 62 61 55 86 55 59 77 83 ...
##  $ attacking_short_passing   : int  92 83 87 89 92 78 92 84 82 71 ...
##  $ attacking_volleys         : int  88 87 87 83 82 45 76 79 79 14 ...
##  $ skill_dribbling           : int  97 89 96 95 86 70 87 89 91 69 ...
##  $ skill_curve               : int  93 81 88 83 85 60 85 83 79 28 ...
##  $ skill_fk_accuracy         : int  94 76 87 79 83 70 78 69 63 28 ...
##  $ skill_long_passing        : int  92 77 81 83 91 81 88 75 70 63 ...
##  $ skill_ball_control        : int  96 92 95 94 91 76 92 89 90 71 ...
##  $ movement_acceleration     : int  91 89 94 94 77 74 77 94 96 69 ...
##  $ movement_sprint_speed     : int  84 91 89 88 76 79 71 92 96 73 ...
##  $ movement_agility          : int  93 87 96 95 78 61 92 91 92 52 ...
##  $ movement_reactions        : int  95 96 92 90 91 88 89 92 89 86 ...
##  $ movement_balance          : int  95 71 84 94 76 53 93 88 83 41 ...
##  $ power_shot_power          : int  86 95 80 82 91 81 79 80 83 55 ...
##  $ power_jumping             : int  68 95 61 56 63 90 68 69 76 81 ...
##  $ power_stamina             : int  75 85 81 84 89 75 85 85 84 73 ...
##  $ power_strength            : int  68 78 49 63 74 92 58 73 76 95 ...
##  $ power_long_shots          : int  94 93 84 80 90 64 82 84 79 15 ...
##  $ mentality_aggression      : int  48 63 51 54 76 82 62 63 62 87 ...
##  $ mentality_interceptions   : int  40 29 36 41 61 89 82 55 38 88 ...
##  $ mentality_positioning     : int  94 95 87 87 88 47 79 92 89 35 ...
##  $ mentality_vision          : int  94 82 90 89 94 65 91 84 80 52 ...
##  $ mentality_penalties       : int  75 85 90 88 79 62 82 77 70 33 ...
##  $ mentality_composure       : int  96 95 94 91 91 89 92 91 84 82 ...
##  $ defending_marking         : int  33 28 27 34 68 91 68 38 34 91 ...
##  $ defending_standing_tackle : int  37 32 26 27 58 92 76 43 34 90 ...
##  $ defending_sliding_tackle  : int  26 24 29 22 51 85 71 41 32 87 ...
##  - attr(*, ".internal.selfref")=<externalptr>
##  - attr(*, "na.action")= 'omit' Named int  4 7 14 15 26 29 31 32 33 54 ...
##   ..- attr(*, "names")= chr  "4" "7" "14" "15" ...
```

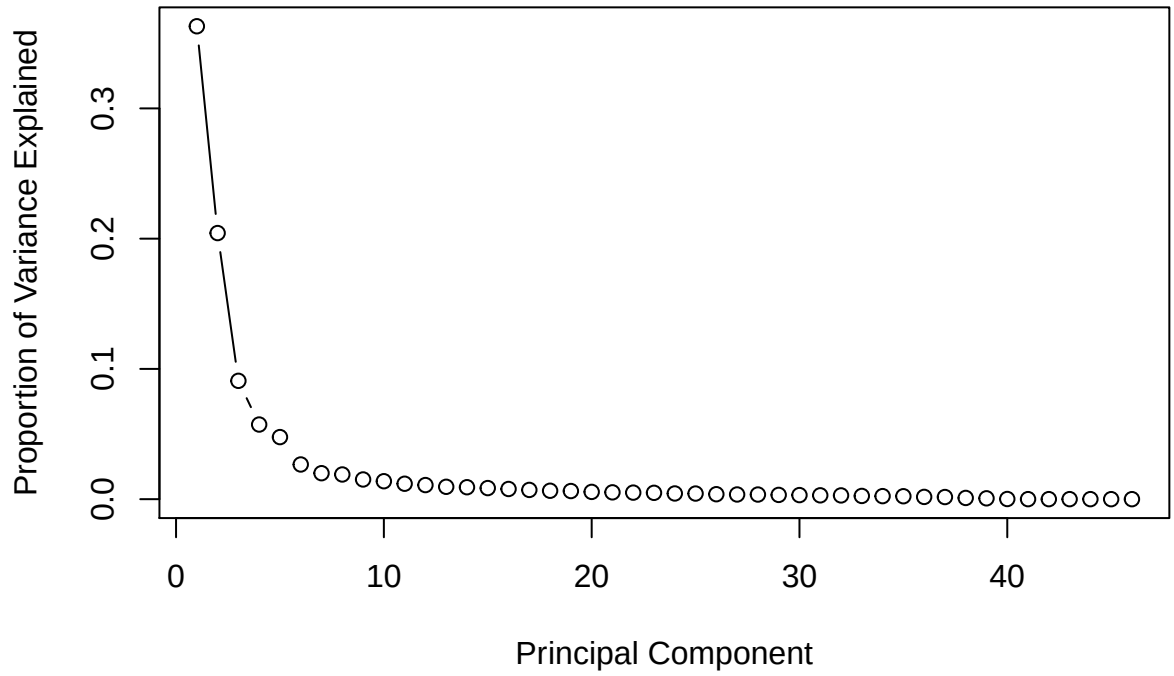
```
df <- df %>% select(-preferred_foot)

#split df into training and testing sets:
set.seed(4321)
smp_size2 <- floor(0.80 * nrow(df))
train_ind <- sample(seq_len(nrow(df)), size = smp_size2)

trainset <- df[train_ind, ]
testset <- df[-train_ind, ]

#PCA train and test sets:
pca_trainset <- trainset %>% select( -work_rate )
pca_testset <- testset
str(pca_trainset)
```

```
## Classes 'tbl_df', 'tbl' and 'data.frame':    12061 obs. of  46 variables:
##  $ age                        : int  18 23 25 21 25 30 29 26 22 20 ...
##  $ height_cm                  : int  180 178 184 170 177 180 178 175 183 171 ...
##  $ weight_kg                  : int  75 70 72 71 71 71 72 67 80 65 ...
##  $ overall                    : int  68 63 63 72 77 88 72 65 59 59 ...
##  $ potential                  : int  83 71 68 78 82 88 72 67 66 73 ...
##  $ value_eur                  : int  1800000 475000 425000 4300000 10500000 56000000 2600000 600000 19
##  $ wage_eur                   : int  11000 2000 1000 8000 47000 170000 20000 2000 1000 2000 ...
##  $ international_reputation   : int  1 1 1 1 2 4 2 1 1 1 ...
##  $ weak_foot                  : int  2 3 2 3 4 4 3 3 2 3 ...
##  $ skill_moves                : int  2 2 2 3 4 4 3 3 2 2 ...
##  $ release_clause_eur         : int  4700000 855000 563000 9000000 18600000 92400000 4100000 915000 27
##  $ pace                       : int  85 73 74 71 80 85 73 65 72 63 ...
##  $ shooting                   : int  62 27 25 61 62 88 46 55 29 47 ...
##  $ passing                    : int  59 50 51 71 75 84 67 62 49 64 ...
##  $ dribbling                  : int  72 59 60 77 84 87 73 66 57 60 ...
##  $ defending                  : int  25 60 58 38 72 45 68 63 56 41 ...
##  $ physic                     : int  55 61 69 57 68 66 69 62 63 46 ...
##  $ attacking_crossing         : int  59 56 59 72 78 82 72 55 56 54 ...
##  $ attacking_finishing        : int  62 21 24 59 63 87 36 48 32 44 ...
##  $ attacking_heading_accuracy: int  59 54 35 39 65 49 60 57 60 48 ...
##  $ attacking_short_passing    : int  64 62 61 73 77 86 71 72 52 68 ...
##  $ attacking_volleys          : int  61 49 20 59 59 90 49 46 29 48 ...
##  $ skill_dribbling            : int  73 56 58 80 85 87 71 64 54 60 ...
##  $ skill_curve                : int  62 33 33 74 69 89 60 44 28 60 ...
##  $ skill_fk_accuracy          : int  49 32 29 68 52 86 47 59 25 55 ...
##  $ skill_long_passing         : int  45 30 44 69 68 77 69 66 55 71 ...
##  $ skill_ball_control         : int  73 60 60 75 83 88 74 68 61 59 ...
##  $ movement_acceleration      : int  86 75 75 74 85 85 74 67 74 55 ...
##  $ movement_sprint_speed      : int  84 72 74 68 76 85 73 64 70 69 ...
##  $ movement_agility           : int  76 69 65 74 84 86 77 68 61 56 ...
##  $ movement_reactions         : int  51 63 54 67 78 87 66 61 45 52 ...
##  $ movement_balance           : int  69 62 67 83 78 84 76 69 63 81 ...
##  $ power_shot_power           : int  65 28 26 63 58 88 66 66 24 63 ...
##  $ power_jumping              : int  47 61 63 47 67 71 67 63 64 57 ...
##  $ power_stamina              : int  66 68 77 67 77 76 81 64 64 51 ...
##  $ power_strength             : int  55 59 68 53 64 67 61 64 66 43 ...
##  $ power_long_shots           : int  59 27 20 64 61 87 43 61 23 37 ...
##  $ mentality_aggression       : int  42 59 62 56 69 51 74 55 56 42 ...
##  $ mentality_interceptions    : int  18 60 63 38 75 48 68 63 52 25 ...
##  $ mentality_positioning      : int  62 54 50 61 74 89 59 52 36 44 ...
##  $ mentality_vision           : int  61 45 42 70 79 86 59 52 44 63 ...
##  $ mentality_penalties        : int  60 36 33 65 58 90 60 54 31 52 ...
##  $ mentality_composure        : int  48 67 45 72 80 85 68 58 46 58 ...
##  $ defending_marking          : int  22 57 58 32 71 49 66 65 47 38 ...
##  $ defending_standing_tackle : int  22 65 61 42 73 37 73 64 63 46 ...
##  $ defending_sliding_tackle   : int  24 60 58 39 72 45 71 63 62 59 ...
##  - attr(*, ".internal.selfref")=<externalptr>
##  - attr(*, "na.action")= 'omit' Named int  4 7 14 15 26 29 31 32 33 54 ...
##   ..- attr(*, "names")= chr  "4" "7" "14" "15" ...
```

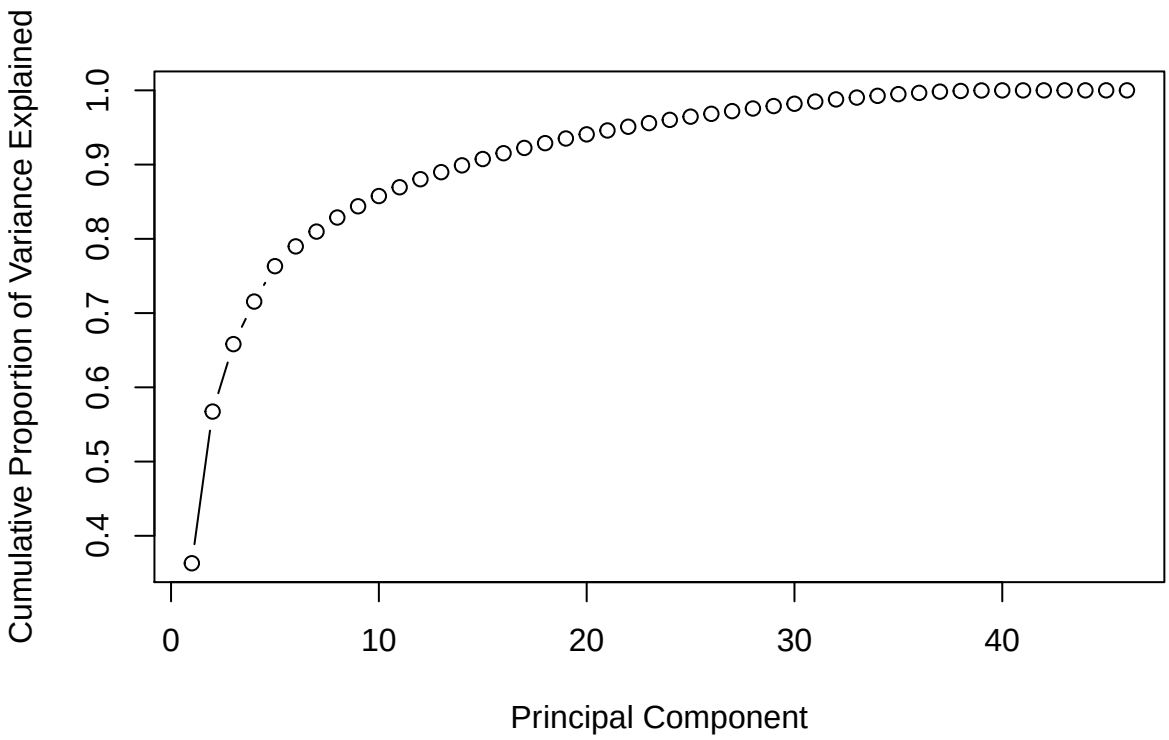
```
dim(pca_trainset)
```

```
## [1] 12061    46
```

```r
#PCA on the train set:
pca <- prcomp( pca_trainset, scale = T )

# variance
pr_var <- ( pca$sdev )^2

# % of variance
prop_varex <- pr_var / sum( pr_var )

#plot of proportion of variance explained by components:
plot( prop_varex, xlab = "Principal Component",
                  ylab = "Proportion of Variance Explained", type = "b" )
```




```r
#Scree plot, prop of cumulative variance explained by components:
plot( cumsum( prop_varex ), xlab = "Principal Component",
                            ylab = "Cumulative Proportion of Variance Explained", type = "b" )
```




```
#we see that about 97% of the variance explained is done by 36 of the 46 features.
#Therefore we can model with these first 36 PCs.
```

#PCA Continuation:

```
# Creating a new dataset
actrain = data.frame( class = trainset$work_rate, pca$x )
t = as.data.frame( predict( pca, newdata = pca_testset ) )

new_trainset = actrain[, 1:37]
new_testset =  t[, 1:36]
```

#LDA model on the new dataset after PCA:

```
fit_wrate_pca_lda <- train(class~., data=new_trainset, method="lda",
              trControl = trCtrl, metric = "Accuracy")

tt <- predict( fit_wrate_pca_lda, new_testset)

#accuarcy of cross validated PCA-LDA model:
mean(tt == pca_testset$work_rate)
```

```
## [1] 0.5242042
```

```r
#52.4% accuracy
#The accuracy didnt increase much even after performing PCA.
#confusion matrix:
confusionMatrix(as.factor(pca_testset$work_rate), tt)
```

```
## Confusion Matrix and Statistics
##
##                Reference
## Prediction      High/High High/Low High/Medium Low/Medium Medium/High
##   High/High            44        0          49          0           8
##   High/Low              1       14          34          1           0
##   High/Medium          28       14         215          4           6
##   Low/Medium            0        0           1         67          18
##   Medium/High          11        0          13         41          55
##   Medium/Low            1        8          28          1           0
##   Medium/Medium        17       16         115         70          36
##                Reference
## Prediction      Medium/Low Medium/Medium
##   High/High              2            82
##   High/Low               9            60
##   High/Medium           13           327
##   Low/Medium             2            96
##   Medium/High            0           202
##   Medium/Low            16            97
##   Medium/Medium         24          1170
##
## Overall Statistics
##
##                Accuracy : 0.5242
##                  95% CI : (0.5062, 0.5422)
##     No Information Rate : 0.6744
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.2494
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: High/High Class: High/Low Class: High/Medium
## Sensitivity                   0.43137        0.269231            0.47253
## Specificity                   0.95161        0.964575            0.84693
## Pos Pred Value                0.23784        0.117647            0.35420
## Neg Pred Value                0.97951        0.986883            0.90037
## Prevalence                    0.03382        0.017241            0.15086
## Detection Rate                0.01459        0.004642            0.07129
## Detection Prevalence          0.06134        0.039456            0.20126
## Balanced Accuracy             0.69149        0.616903            0.65973
##                      Class: Low/Medium Class: Medium/High Class: Medium/Low
## Sensitivity                    0.36413            0.44715          0.242424
## Specificity                    0.95869            0.90771          0.954237
## Pos Pred Value                 0.36413            0.17081          0.105960
## Neg Pred Value                 0.95869            0.97476          0.982548
```

```
## Prevalence                     0.06101           0.04078          0.021883
## Detection Rate                 0.02221           0.01824          0.005305
## Detection Prevalence           0.06101           0.10676          0.050066
## Balanced Accuracy              0.66141           0.67743          0.598331
##                       Class: Medium/Medium
## Sensitivity                         0.5752
## Specificity                         0.7169
## Pos Pred Value                      0.8080
## Neg Pred Value                      0.4490
## Prevalence                          0.6744
## Detection Rate                      0.3879
## Detection Prevalence                0.4801
## Balanced Accuracy                   0.6461
```

#Feature selection approach to reduce number of dimensions:

```
str(df)
```

```
## Classes 'tbl_df', 'tbl' and 'data.frame':    15077 obs. of  47 variables:
##  $ age                        : int  32 34 27 28 28 27 33 27 20 28 ...
##  $ height_cm                  : int  170 187 175 175 181 193 172 175 178 187 ...
##  $ weight_kg                  : int  72 83 68 74 70 92 66 71 73 89 ...
##  $ overall                    : int  94 93 92 91 91 90 90 90 89 89 ...
##  $ potential                  : int  94 93 92 91 91 91 90 90 95 91 ...
##  $ value_eur                  : int  95500000 58500000 105500000 90000000 90000000 78000000 45000000 8
##  $ wage_eur                   : int  565000 405000 290000 470000 370000 200000 340000 240000 155000 15
##  $ international_reputation   : int  5 5 5 4 4 3 4 3 3 3 ...
##  $ weak_foot                  : int  4 4 5 4 5 3 4 3 4 3 ...
##  $ skill_moves                : int  4 5 5 4 4 2 4 4 5 2 ...
##  $ work_rate                  : chr  "Medium/Low" "High/Low" "High/Medium" "High/Medium" ...
##  $ release_clause_eur         : int  195800000 96500000 195200000 184500000 166500000 150200000 92300
##  $ pace                       : int  87 90 91 91 76 77 74 93 96 71 ...
##  $ shooting                   : int  92 93 85 83 86 60 76 86 84 28 ...
##  $ passing                    : int  92 82 87 86 92 70 89 81 78 54 ...
##  $ dribbling                  : int  96 89 95 94 86 71 89 89 90 67 ...
##  $ defending                  : int  39 35 32 35 61 90 72 45 39 89 ...
##  $ physic                     : int  66 78 58 66 78 86 66 74 75 87 ...
##  $ attacking_crossing         : int  88 84 87 81 93 53 86 79 78 30 ...
##  $ attacking_finishing        : int  95 94 87 84 82 52 72 90 89 22 ...
##  $ attacking_heading_accuracy: int  70 89 62 61 55 86 55 59 77 83 ...
##  $ attacking_short_passing    : int  92 83 87 89 92 78 92 84 82 71 ...
##  $ attacking_volleys          : int  88 87 87 83 82 45 76 79 79 14 ...
##  $ skill_dribbling            : int  97 89 96 95 86 70 87 89 91 69 ...
##  $ skill_curve                : int  93 81 88 83 85 60 85 83 79 28 ...
##  $ skill_fk_accuracy          : int  94 76 87 79 83 70 78 69 63 28 ...
##  $ skill_long_passing         : int  92 77 81 83 91 81 88 75 70 63 ...
##  $ skill_ball_control         : int  96 92 95 94 91 76 92 89 90 71 ...
##  $ movement_acceleration      : int  91 89 94 94 77 74 77 94 96 69 ...
##  $ movement_sprint_speed      : int  84 91 89 88 76 79 71 92 96 73 ...
##  $ movement_agility           : int  93 87 96 95 78 61 92 91 92 52 ...
##  $ movement_reactions         : int  95 96 92 90 91 88 89 92 89 86 ...
##  $ movement_balance           : int  95 71 84 94 76 53 93 88 83 41 ...
##  $ power_shot_power           : int  86 95 80 82 91 81 79 80 83 55 ...
```

```
##  $ power_jumping             : int  68 95 61 56 63 90 68 69 76 81 ...
##  $ power_stamina             : int  75 85 81 84 89 75 85 85 84 73 ...
##  $ power_strength            : int  68 78 49 63 74 92 58 73 76 95 ...
##  $ power_long_shots          : int  94 93 84 80 90 64 82 84 79 15 ...
##  $ mentality_aggression      : int  48 63 51 54 76 82 62 63 62 87 ...
##  $ mentality_interceptions   : int  40 29 36 41 61 89 82 55 38 88 ...
##  $ mentality_positioning     : int  94 95 87 87 88 47 79 92 89 35 ...
##  $ mentality_vision          : int  94 82 90 89 94 65 91 84 80 52 ...
##  $ mentality_penalties       : int  75 85 90 88 79 62 82 77 70 33 ...
##  $ mentality_composure       : int  96 95 94 91 91 89 92 91 84 82 ...
##  $ defending_marking         : int  33 28 27 34 68 91 68 38 34 91 ...
##  $ defending_standing_tackle : int  37 32 26 27 58 92 76 43 34 90 ...
##  $ defending_sliding_tackle  : int  26 24 29 22 51 85 71 41 32 87 ...
##  - attr(*, ".internal.selfref")=<externalptr>
##  - attr(*, "na.action")= 'omit' Named int  4 7 14 15 26 29 31 32 33 54 ...
##   ..- attr(*, "names")= chr  "4" "7" "14" "15" ...
```

```
#correlaiton matrix:
dup_df <- df%>%select(-work_rate)
cor_mat <- cor(dup_df)
#summary of cor mat:
#print(cor_mat)
#attributes that are highly correlated:
highlyCorrelated <- findCorrelation(cor_mat, cutoff=0.7)
#indices of highly correlated attributes:
highlyCorrelated
```

```
##  [1] 15 23 14 27 13  4 40 41 37 19 21 24 22 31 30  6 16 28 12 32 45 39 44  7 17
## [26] 36  3
```

```
#we get 27 features that are highly correlated
#View(dup_df)
#selecting only relevant features from dup_df:
dup_df <- dup_df[,highlyCorrelated]
#append work_rate to dup_df:
dim(dup_df)
```

```
## [1] 15077    27
```

```
dim(ddff)
```

```
## [1] 15077    48
```

```
dataset3 <- cbind(work_rate = ddff$work_rate, dup_df)
dim(dataset3)
```

```
## [1] 15077    28
```

#Classification models on dataset3:

```r
#split dataset3 into train and test sets:
set.seed(43)
smp_size3 <- floor(0.80 * nrow(dataset3))
train_ind3 <- sample(seq_len(nrow(dataset3)), size = smp_size3)

ds3_train <- df[train_ind3, ]
ds3_test <- df[-train_ind3, ]
```

#LDA

```r
ds3_fit_wrate <- train(work_rate~., data=ds3_train, method="lda",
              trControl = trCtrl, metric = "Accuracy")


ds3_pred_wrate <- predict(ds3_fit_wrate, ds3_test%>%select(-work_rate))
ds3_comparison <- data.frame(original = ds3_test$work_rate, pred = ds3_pred_wrate)

#accuarcy of cross validated LDA model:
mean(ds3_comparison$pred == ds3_test$work_rate)
```

```
## [1] 0.5016578
```

```r
#50% accuracy

#confusion matrix:
confusionMatrix(as.factor(ds3_test$work_rate), ds3_comparison$pred)
```

```
## Confusion Matrix and Statistics
##
##                Reference
## Prediction      High/High High/Low High/Medium Low/Medium Medium/High
##   High/High            39        1          55          0          10
##   High/Low              1       13          43          0           1
##   High/Medium          16       16         201          3           7
##   Low/Medium            3        0           1         64          18
##   Medium/High          21        0          11         44          57
##   Medium/Low            1        6          33          0           0
##   Medium/Medium        15       18         129         95          48
##                Reference
## Prediction      Medium/Low Medium/Medium
##   High/High              2            83
##   High/Low              11            63
##   High/Medium           14           328
##   Low/Medium             1            82
##   Medium/High            1           194
##   Medium/Low            16            95
##   Medium/Medium         33          1123
##
## Overall Statistics
##
##                Accuracy : 0.5017
##                  95% CI : (0.4837, 0.5197)
```

```
##     No Information Rate : 0.6525
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.2219
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: High/High Class: High/Low Class: High/Medium
## Sensitivity                   0.40625         0.24074            0.42495
## Specificity                   0.94829         0.95982            0.84900
## Pos Pred Value                0.20526         0.09848            0.34359
## Neg Pred Value                0.97983         0.98578            0.88811
## Prevalence                    0.03183         0.01790            0.15683
## Detection Rate                0.01293         0.00431            0.06664
## Detection Prevalence          0.06300         0.04377            0.19397
## Balanced Accuracy             0.67727         0.60028            0.63697
##                      Class: Low/Medium Class: Medium/High Class: Medium/Low
## Sensitivity                    0.31068            0.40426           0.205128
## Specificity                    0.96263            0.90574           0.954050
## Pos Pred Value                 0.37870            0.17378           0.105960
## Neg Pred Value                 0.95012            0.96875           0.978360
## Prevalence                     0.06830            0.04675           0.025862
## Detection Rate                 0.02122            0.01890           0.005305
## Detection Prevalence           0.05603            0.10875           0.050066
## Balanced Accuracy              0.63666            0.65500           0.579589
##                      Class: Medium/Medium
## Sensitivity                        0.5706
## Specificity                        0.6775
## Pos Pred Value                     0.7687
## Neg Pred Value                     0.4566
## Prevalence                         0.6525
## Detection Rate                     0.3723
## Detection Prevalence               0.4844
## Balanced Accuracy                  0.6241
```

```r
#The accuracy is still low even after using a subset of features from the original
#dataset.
```

#Using domain knowledge to select features:

```r
#View(dataset3)

dataset4 <- dataset3 %>% select(-skill_dribbling, -mentality_vision,
                                -attacking_short_passing, -skill_curve, -attacking_volleys,
                                -movement_reactions, -movement_agility, -movement_acceleration, -moveme


#split dataset4 into train and test sets:
set.seed(25)
smp_size4 <- floor(0.80 * nrow(dataset4))
train_ind4 <- sample(seq_len(nrow(dataset4)), size = smp_size4)
```

```
ds4_train <- df[train_ind4, ]
ds4_test <- df[-train_ind4, ]
```

#LDA

```
ds4_fit_wrate <- train(work_rate~., data=ds4_train, method="lda",
              trControl = trCtrl, metric = "Accuracy")


ds4_pred_wrate <- predict(ds4_fit_wrate, ds4_test%>%select(-work_rate))
ds4_comparison <- data.frame(original = ds4_test$work_rate, pred = ds4_pred_wrate)

#accuarcy of cross validated LDA model:
mean(ds4_comparison$pred == ds4_test$work_rate)
```

```
## [1] 0.5099469
```

```
#50% accuracy

#confusion matrix:
confusionMatrix(as.factor(ds4_test$work_rate), ds4_comparison$pred)
```

```
## Confusion Matrix and Statistics
##
##                Reference
## Prediction      High/High High/Low High/Medium Low/Medium Medium/High
##   High/High            56        2          43          1           8
##   High/Low              1       16          35          0           0
##   High/Medium          26       17         193          6           8
##   Low/Medium            1        0           0         68          20
##   Medium/High          14        1          12         34          49
##   Medium/Low            0       13          39          1           0
##   Medium/Medium        19       18         138         93          62
##                Reference
## Prediction      Medium/Low Medium/Medium
##   High/High              0            87
##   High/Low               8            50
##   High/Medium            7           309
##   Low/Medium             1            95
##   Medium/High            0           199
##   Medium/Low            18            90
##   Medium/Medium         20          1138
##
## Overall Statistics
##
##                Accuracy : 0.5099
##                  95% CI : (0.4919, 0.5279)
##     No Information Rate : 0.6525
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.2295
##
```

```
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: High/High Class: High/Low Class: High/Medium
## Sensitivity                   0.47863        0.238806            0.41957
## Specificity                   0.95136        0.968125            0.85407
## Pos Pred Value                0.28426        0.145455            0.34099
## Neg Pred Value                0.97836        0.982450            0.89102
## Prevalence                    0.03879        0.022215            0.15252
## Detection Rate                0.01857        0.005305            0.06399
## Detection Prevalence          0.06532        0.036472            0.18767
## Balanced Accuracy             0.71500        0.603465            0.63682
##                      Class: Low/Medium Class: Medium/High Class: Medium/Low
## Sensitivity                    0.33498             0.33333           0.333333
## Specificity                    0.95841             0.90938           0.951722
## Pos Pred Value                 0.36757             0.15858           0.111801
## Neg Pred Value                 0.95231             0.96380           0.987391
## Prevalence                     0.06731             0.04874           0.017905
## Detection Rate                 0.02255             0.01625           0.005968
## Detection Prevalence           0.06134             0.10245           0.053382
## Balanced Accuracy              0.64669             0.62135           0.642528
##                      Class: Medium/Medium
## Sensitivity                        0.5783
## Specificity                        0.6660
## Pos Pred Value                     0.7648
## Neg Pred Value                     0.4568
## Prevalence                         0.6525
## Detection Rate                     0.3773
## Detection Prevalence               0.4934
## Balanced Accuracy                  0.6221
```

```
#The accuracy is still low even after using a subset of features from the original
#dataset.
```

#Further narrowing down the features in the dataset3:

```
dataset5 <- dataset3 %>% select(shooting, attacking_finishing, wage_eur, defending,
                                pace, defending_standing_tackle, defending_marking)


#split dataset5 into train and test sets:
set.seed(25)
smp_size5 <- floor(0.80 * nrow(dataset5))
train_ind5 <- sample(seq_len(nrow(dataset5)), size = smp_size5)

ds5_train <- df[train_ind5, ]
ds5_test <- df[-train_ind5, ]
```

#LDA

```
set.seed(456)
ds5_fit_wrate <- train(work_rate~., data=ds5_train, method="lda",
```

```
              trControl = trCtrl, metric = "Accuracy")


ds5_pred_wrate <- predict(ds5_fit_wrate, ds5_test%>%select(-work_rate))
ds5_comparison <- data.frame(original = ds5_test$work_rate, pred = ds5_pred_wrate)

#accuarcy of cross validated LDA model:
mean(ds5_comparison$pred == ds5_test$work_rate)
```

```
## [1] 0.5099469
```

```
#50% accuracy

#confusion matrix:
confusionMatrix(as.factor(ds5_test$work_rate), ds5_comparison$pred)
```

```
## Confusion Matrix and Statistics
##
##                Reference
## Prediction      High/High High/Low High/Medium Low/Medium Medium/High
##   High/High            56        2          43          1           8
##   High/Low              1       16          35          0           0
##   High/Medium          26       17         193          6           8
##   Low/Medium            1        0           0         68          20
##   Medium/High          14        1          12         34          49
##   Medium/Low            0       13          39          1           0
##   Medium/Medium        19       18         138         93          62
##                Reference
## Prediction      Medium/Low Medium/Medium
##   High/High              0            87
##   High/Low               8            50
##   High/Medium            7           309
##   Low/Medium             1            95
##   Medium/High            0           199
##   Medium/Low            18            90
##   Medium/Medium         20          1138
##
## Overall Statistics
##
##                Accuracy : 0.5099
##                  95% CI : (0.4919, 0.5279)
##     No Information Rate : 0.6525
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.2295
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: High/High Class: High/Low Class: High/Medium
## Sensitivity                   0.47863        0.238806            0.41957
```

```
## Specificity                    0.95136        0.968125            0.85407
## Pos Pred Value                 0.28426        0.145455            0.34099
## Neg Pred Value                 0.97836        0.982450            0.89102
## Prevalence                     0.03879        0.022215            0.15252
## Detection Rate                 0.01857        0.005305            0.06399
## Detection Prevalence           0.06532        0.036472            0.18767
## Balanced Accuracy              0.71500        0.603465            0.63682
##                      Class: Low/Medium Class: Medium/High Class: Medium/Low
## Sensitivity                    0.33498            0.33333           0.333333
## Specificity                    0.95841            0.90938           0.951722
## Pos Pred Value                 0.36757            0.15858           0.111801
## Neg Pred Value                 0.95231            0.96380           0.987391
## Prevalence                     0.06731            0.04874           0.017905
## Detection Rate                 0.02255            0.01625           0.005968
## Detection Prevalence           0.06134            0.10245           0.053382
## Balanced Accuracy              0.64669            0.62135           0.642528
##                      Class: Medium/Medium
## Sensitivity                        0.5783
## Specificity                        0.6660
## Pos Pred Value                     0.7648
## Neg Pred Value                     0.4568
## Prevalence                         0.6525
## Detection Rate                     0.3773
## Detection Prevalence               0.4934
## Balanced Accuracy                  0.6221
```

```
#accuracy is still low after further narrowing down the features too.
```