fifa20 attack work rate classification

Naga Santhosh Kartheek Karnati

4/4/2020

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
```

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

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

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

```r
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)


clean_fifa20 <- na.omit(updated_fifa20)
clean_fifa20
```

```
## # A tibble: 15,077 x 55
##    sofifa_id short_name   age height_cm weight_kg nationality club  overall
##        <int> <chr>      <int>     <int>     <int> <chr>       <chr>   <int>
##  1    158023 L. Messi      32       170        72 Argentina   FC B~      94
##  2     20801 Cristiano~    34       187        83 Portugal    Juve~      93
##  3    190871 Neymar Jr     27       175        68 Brazil      Pari~      92
##  4    183277 E. Hazard     28       175        74 Belgium     Real~      91
##  5    192985 K. De Bru~    28       181        70 Belgium     Manc~      91
##  6    203376 V. van Di~    27       193        92 Netherlands Live~      90
##  7    177003 L. ModriÄ‡    33       172        66 Croatia     Real~      90
##  8    209331 M. Salah      27       175        71 Egypt       Live~      90
##  9    231747 K. MbappÃ©    20       178        73 France      Pari~      89
## 10    201024 K. Koulib~    28       187        89 Senegal     Napo~      89
## # ... with 15,067 more rows, and 47 more variables: potential <int>,
## #   value_eur <int>, wage_eur <int>, preferred_foot <chr>,
## #   international_reputation <int>, weak_foot <int>, skill_moves <int>,
## #   work_rate <chr>, body_type <chr>, release_clause_eur <int>,
## #   team_position <chr>, team_jersey_number <int>, pace <int>, shooting <int>,
## #   passing <int>, dribbling <int>, defending <int>, physic <int>,
## #   attacking_crossing <int>, attacking_finishing <int>,
## #   attacking_heading_accuracy <int>, attacking_short_passing <int>,
## #   attacking_volleys <int>, skill_dribbling <int>, skill_curve <int>,
## #   skill_fk_accuracy <int>, skill_long_passing <int>,
## #   skill_ball_control <int>, movement_acceleration <int>,
## #   movement_sprint_speed <int>, movement_agility <int>,
```

```
## #   movement_reactions <int>, movement_balance <int>, power_shot_power <int>,
## #   power_jumping <int>, power_stamina <int>, power_strength <int>,
## #   power_long_shots <int>, mentality_aggression <int>,
## #   mentality_interceptions <int>, mentality_positioning <int>,
## #   mentality_vision <int>, mentality_penalties <int>,
## #   mentality_composure <int>, defending_marking <int>,
## #   defending_standing_tackle <int>, defending_sliding_tackle <int>
```

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


#split work rate into attack work rate and defense work rate:
df <- separate(df, work_rate, into = c("attack_workrate","defence_workrate"),
         sep = "/")
df <- df%>% select(-defence_workrate)
dim(df)
```

```
## [1] 15077    47
```

```
#number of classes in attack_workrate:
unique(df$attack_workrate)
```

```
## [1] "Medium" "High"   "Low"
```

```
#3 classes: Medium, High and Low
```

#Various classification models to classify attack work rate:

```
library(caret)
```

```
## Loading required package: lattice

##
## Attaching package: 'caret'

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

```
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
```

```r
#splitting the data into train and test sets:
set.seed(1)
training.samples <- df$attack_workrate %>% createDataPartition(p = 0.8, list = FALSE)
train.data  <- df[training.samples, ]
test.data <- df[-training.samples, ]
dim(train.data)
```

```
## [1] 12063    47
```

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

```
## [1] 3014   47
```

#Multinomial logistic regression:

```r
set.seed(2)
# Fit the model
model <- nnet::multinom(attack_workrate ~., data = train.data)
```

```
## # weights:  144 (94 variable)
## initial  value 13252.560038
## iter  10 value 9546.000264
## iter  20 value 8957.649488
## iter  30 value 8937.036257
## iter  40 value 8929.095677
## iter  50 value 8925.391382
## iter  60 value 8921.931993
## iter  70 value 8866.748738
## iter  80 value 8026.905516
## iter  90 value 7998.770733
## iter 100 value 7995.137262
## final  value 7995.137262
## stopped after 100 iterations
```

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

```
## Call:
## nnet::multinom(formula = attack_workrate ~ ., data = train.data)
##
## Coefficients:
##        (Intercept)          age    height_cm     weight_kg    overall
## Low    0.003981837 -0.017907514 -0.002126633 -0.0006627199 0.21097172
## Medium 0.082424044 -0.008303212  0.031180989  0.0030697230 0.04210615
##            potential     value_eur      wage_eur international_reputation
## Low    -0.0278730411 -1.461295e-07 -2.515132e-05               0.13845878
## Medium  0.0001072018 -8.870279e-08 -3.853260e-06              -0.02435387
##          weak_foot skill_moves release_clause_eur       pace    shooting
## Low    -0.28963325  0.06205536       4.518079e-08 0.01895589 -0.05072763
## Medium -0.09393245 -0.10122893       3.195102e-08 0.06917058 -0.11031189
##            passing   dribbling  defending       physic attacking_crossing
```

```
## Low     0.007642033 -0.10033085 -0.1679242 -0.015693291        -0.02210820
## Medium -0.249447079  0.05076711 -0.1040380 -0.004913591         0.03500333
##        attacking_finishing attacking_heading_accuracy attacking_short_passing
## Low           -0.001861378               0.0173799611              0.01416779
## Medium         0.037360000              -0.0003399062              0.11148176
##        attacking_volleys skill_dribbling  skill_curve skill_fk_accuracy
## Low          0.008213568     0.007195515 0.0005865959       -0.00264244
## Medium       0.006989999    -0.046751479 0.0007939614        0.01538107
##        skill_long_passing skill_ball_control movement_acceleration
## Low           0.01933752        0.016874508           -0.03785939
## Medium        0.04578607       -0.003413917           -0.05676417
##        movement_sprint_speed movement_agility movement_reactions
## Low              -0.04726862      0.009042739        0.016204885
## Medium           -0.05846040     -0.003050052        0.007118117
##        movement_balance power_shot_power power_jumping power_stamina
## Low          0.01308259    -8.012735e-05   0.001245107   -0.02284748
## Medium       0.01180052     1.037617e-02   0.005751873   -0.02673055
##        power_strength power_long_shots mentality_aggression
## Low       0.007909334       0.02696863         -0.005644743
## Medium    0.009746365       0.03461797         -0.005034601
##        mentality_interceptions mentality_positioning mentality_vision
## Low                 0.05012099           -0.05572042    -0.0001587364
## Medium              0.02612448           -0.03819763     0.0556619394
##        mentality_penalties mentality_composure defending_marking
## Low            0.014909126        -0.007959287        0.04579932
## Medium         0.008215098        -0.008835074        0.02980943
##        defending_standing_tackle defending_sliding_tackle
## Low                   0.06979270             -0.012802757
## Medium                0.03318373              0.001558635
##
## Std. Errors:
##         (Intercept)          age    height_cm    weight_kg      overall
## Low    1.089287e-10 3.332025e-09 2.006904e-08 8.598999e-09 7.541549e-09
## Medium 5.806027e-11 1.748570e-09 1.083517e-08 4.693260e-09 4.039237e-09
##           potential    value_eur     wage_eur international_reputation
## Low    7.647936e-09 8.247397e-08 4.777866e-06             1.548409e-10
## Medium 4.152022e-09 3.554297e-08 1.965930e-06             6.722237e-11
##           weak_foot  skill_moves release_clause_eur         pace     shooting
## Low    3.099510e-10 2.433337e-10       4.084479e-08 5.812621e-09 5.064393e-09
## Medium 1.577596e-10 1.155944e-10       1.810715e-08 2.621407e-09 2.299978e-09
##             passing    dribbling    defending       physic attacking_crossing
## Low    6.139698e-09 6.229630e-09 7.159334e-09 7.565843e-09       5.638557e-09
## Medium 3.025355e-09 2.967238e-09 4.080075e-09 4.107939e-09       2.513614e-09
##        attacking_finishing attacking_heading_accuracy attacking_short_passing
## Low           4.463678e-09               7.317521e-09            6.924074e-09
## Medium        1.941759e-09               4.112724e-09            3.590262e-09
##        attacking_volleys skill_dribbling  skill_curve skill_fk_accuracy
## Low         4.570700e-09    5.893968e-09 5.213476e-09      4.712939e-09
## Medium      2.027528e-09    2.709528e-09 2.293019e-09      2.196389e-09
##        skill_long_passing skill_ball_control movement_acceleration
## Low          6.410608e-09       6.665779e-09          5.768821e-09
## Medium       3.381287e-09       3.309721e-09          2.596740e-09
##        movement_sprint_speed movement_agility movement_reactions
## Low             5.843432e-09     6.116081e-09       7.122078e-09
```

```
## Medium          2.637860e-09     2.821574e-09       3.784154e-09
##        movement_balance power_shot_power power_jumping power_stamina
## Low        6.218671e-09     6.409675e-09  7.484873e-09  6.691101e-09
## Medium     2.930354e-09     3.122294e-09  3.949098e-09  3.362306e-09
##        power_strength power_long_shots mentality_aggression
## Low      7.959859e-09     5.079459e-09         7.663445e-09
## Medium   4.457861e-09     2.320290e-09         4.187031e-09
##        mentality_interceptions mentality_positioning mentality_vision
## Low               7.094414e-09           5.04068e-09     5.630870e-09
## Medium            4.004551e-09           2.11846e-09     2.682872e-09
##        mentality_penalties mentality_composure defending_marking
## Low           5.449891e-09        6.969840e-09       7.04310e-09
## Medium        2.600651e-09        3.713774e-09       4.07305e-09
##        defending_standing_tackle defending_sliding_tackle
## Low                 7.278688e-09             7.067933e-09
## Medium              4.132863e-09             4.035806e-09
##
## Residual Deviance: 15990.27
## AIC: 16178.27
```

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

```
## [1] High High High High High High
## Levels: High Low Medium
```

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

```
## [1] 0.6864632
```

```r
#Accuracy of 68.6%
```

#LDA Classification:

```r
set.seed(3)
trCtrl <- trainControl(method = "cv", number = 5)
lda_fit_wrate <- train(attack_workrate~., data=train.data, method="lda",
              trControl = trCtrl, metric = "Accuracy")


lda_pred_wrate <- predict(lda_fit_wrate, test.data%>%select(-attack_workrate))
lda_comparison <- data.frame(original = test.data$attack_workrate, pred = lda_pred_wrate)

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

```
## [1] 0.6731918
```

```r
#67.3% accuracy

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

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction High  Low Medium
##     High    377    5    521
##     Low       0   47    121
##     Medium  240   98   1605
##
## Overall Statistics
##
##                Accuracy : 0.6732
##                  95% CI : (0.6561, 0.6899)
##     No Information Rate : 0.7455
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.2822
##
##  Mcnemar's Test P-Value : <2e-16
##
## Statistics by Class:
##
##                      Class: High Class: Low Class: Medium
## Sensitivity               0.6110    0.31333        0.7143
## Specificity               0.7806    0.95775        0.5593
## Pos Pred Value            0.4175    0.27976        0.8260
## Neg Pred Value            0.8863    0.96381        0.4006
## Prevalence                0.2047    0.04977        0.7455
## Detection Rate            0.1251    0.01559        0.5325
## Detection Prevalence      0.2996    0.05574        0.6447
## Balanced Accuracy         0.6958    0.63554        0.6368
```

```r
lda_pred_wrate1 <- predict(lda_fit_wrate, test.data%>%select(-attack_workrate), type="prob")
###ROC curve:
multiclass.roc(test.data$attack_workrate, lda_pred_wrate1)
```

```
##
## Call:
## multiclass.roc.default(response = test.data$attack_workrate,     predictor = lda_pred_wrate1)
##
## Data: multivariate predictor lda_pred_wrate1 with 3 levels of test.data$attack_workrate: High, Low, 
## Multi-class area under the curve: 0.796
```

```r
#Multi-class area under the curve: 0.796
```

##QDA Classification:

```r
set.seed(4)
trCtrl <- trainControl(method = "cv", number = 5)
qda_fit_wrate <- train(attack_workrate~., data=train.data, method="qda",
              trControl = trCtrl, metric = "Accuracy")


qda_pred_wrate <- predict(qda_fit_wrate, test.data%>%select(-attack_workrate))
qda_comparison <- data.frame(original = test.data$attack_workrate, pred = qda_pred_wrate)

#accuarcy of cross validated QDA model:
mean(qda_comparison$pred == test.data$attack_workrate)
```

```
## [1] 0.5633709
```

```r
#56.3% accuracy
#lesser accuracy than LDA model, maybe features share the same covraince matrix.

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

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction High  Low Medium
##     High    443   31    429
##     Low       2  112     54
##     Medium  414  386   1143
##
## Overall Statistics
##
##                Accuracy : 0.5634
##                  95% CI : (0.5454, 0.5812)
##     No Information Rate : 0.5395
##     P-Value [Acc > NIR] : 0.004447
##
##                   Kappa : 0.2162
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: High Class: Low Class: Medium
## Sensitivity               0.5157    0.21172        0.7030
## Specificity               0.7865    0.97746        0.4236
## Pos Pred Value            0.4906    0.66667        0.5883
## Neg Pred Value            0.8029    0.85348        0.5490
## Prevalence                0.2850    0.17551        0.5395
## Detection Rate            0.1470    0.03716        0.3792
## Detection Prevalence      0.2996    0.05574        0.6447
## Balanced Accuracy         0.6511    0.59459        0.5633
```

```
qda_pred_wrate1 <- predict(qda_fit_wrate, test.data%>%select(-attack_workrate), type="prob")
###ROC curve:
multiclass.roc(test.data$attack_workrate, qda_pred_wrate1)
```

```
##
## Call:
## multiclass.roc.default(response = test.data$attack_workrate,     predictor = qda_pred_wrate1)
##
## Data: multivariate predictor qda_pred_wrate1 with 3 levels of test.data$attack_workrate: High, Low, M
## Multi-class area under the curve: 0.7866
```

```
#Multi-class area under the curve: 0.7866
```

#Decision tree classification:

```
set.seed(5)
trCtrl <- trainControl(method = "cv", number = 5)
dt_fit_wrate <- train(attack_workrate~., data=train.data, method="rpart",
              trControl = trCtrl, metric = "Accuracy")


dt_pred_wrate <- predict(dt_fit_wrate, test.data%>%select(-attack_workrate))
dt_comparison <- data.frame(original = test.data$attack_workrate, pred = dt_pred_wrate)

#accuarcy of cross validated decision tree model:
mean(dt_comparison$pred == test.data$attack_workrate)
```

```
## [1] 0.6768414
```

```
#67.6% accuracy
#more accauracy than QDA model

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

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction High  Low Medium
##     High    306    0    597
##     Low       2    0    166
##     Medium  209    0   1734
##
## Overall Statistics
##
##                Accuracy : 0.6768
##                  95% CI : (0.6598, 0.6935)
##     No Information Rate : 0.8285
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.2204
```

```
##
##  Mcnemar's Test P-Value : <2e-16
##
## Statistics by Class:
##
##                      Class: High Class: Low Class: Medium
## Sensitivity               0.5919         NA        0.6944
## Specificity               0.7609    0.94426        0.5957
## Pos Pred Value            0.3389         NA        0.8924
## Neg Pred Value            0.9000         NA        0.2876
## Prevalence                0.1715    0.00000        0.8285
## Detection Rate            0.1015    0.00000        0.5753
## Detection Prevalence      0.2996    0.05574        0.6447
## Balanced Accuracy         0.6764         NA        0.6451
```

```r
dt_pred_wrate1 <- predict(dt_fit_wrate, test.data%>%select(-attack_workrate), type="prob")
###ROC curve:
multiclass.roc(test.data$attack_workrate, dt_pred_wrate1)
```

```
##
## Call:
## multiclass.roc.default(response = test.data$attack_workrate,     predictor = dt_pred_wrate1)
##
## Data: multivariate predictor dt_pred_wrate1 with 3 levels of test.data$attack_workrate: High, Low, Me
## Multi-class area under the curve: 0.6467
```

```r
#Multi-class area under the curve: 0.6467

#Less accuarcy of the above models might be dute to having too many features in the
#model. We can perform dimensionality reduction to imcrease the accuracy.
```

#Dimensionality reduction: PCA

```r
#PCA train and test sets:
pca_trainset <- train.data %>% select( -attack_workrate)
pca_testset <- test.data
str(pca_trainset)
```

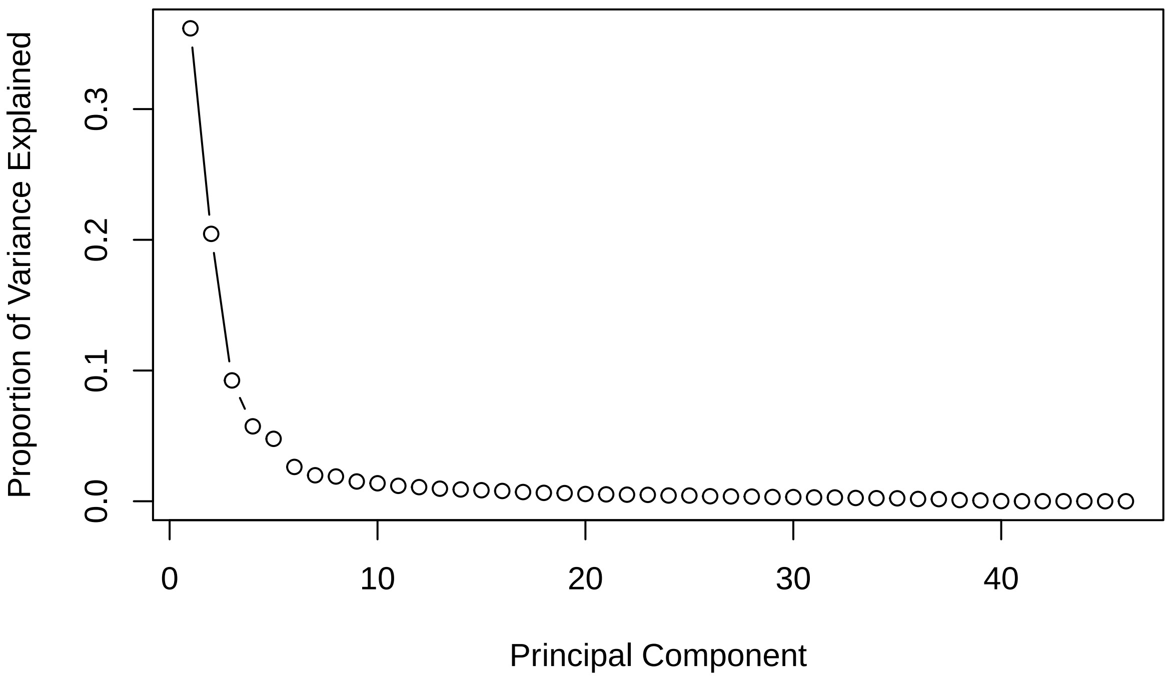
```
## Classes 'tbl_df', 'tbl' and 'data.frame':    12063 obs. of  46 variables:
##  $ age                      : int  34 28 28 33 27 20 28 28 34 31 ...
##  $ height_cm                : int  187 175 181 172 175 178 187 168 187 173 ...
##  $ weight_kg                : int  83 74 70 66 71 73 89 72 85 70 ...
##  $ overall                  : int  93 91 91 90 90 89 89 89 89 89 ...
##  $ potential                : int  93 91 91 90 90 95 91 90 89 89 ...
##  $ value_eur                : int  58500000 90000000 90000000 45000000 80500000 93500000 67500000 60
##  $ wage_eur                 : int  405000 470000 370000 340000 240000 155000 150000 235000 215000 30
##  $ international_reputation : int  5 4 4 4 3 3 3 3 4 4 ...
##  $ weak_foot                : int  4 4 5 4 3 4 3 3 3 4 ...
##  $ skill_moves              : int  5 4 4 4 4 5 2 2 2 4 ...
##  $ release_clause_eur       : int  96500000 184500000 166500000 92300000 148900000 191700000 119800C
##  $ pace                     : int  90 91 76 74 93 96 71 78 68 80 ...
##  $ shooting                 : int  93 83 86 76 86 84 28 65 46 90 ...
```

```
##  $ passing                   : int  82 86 92 89 81 78 54 77 58 77 ...
##  $ dribbling                 : int  89 94 86 89 89 90 67 81 60 88 ...
##  $ defending                 : int  35 35 61 72 45 39 89 87 90 33 ...
##  $ physic                    : int  78 66 78 66 74 75 87 83 82 74 ...
##  $ attacking_crossing        : int  84 81 93 86 79 78 30 68 54 70 ...
##  $ attacking_finishing       : int  94 84 82 72 90 89 22 65 33 93 ...
##  $ attacking_heading_accuracy: int  89 61 55 55 59 77 83 54 83 78 ...
##  $ attacking_short_passing   : int  83 89 92 92 84 82 71 86 65 83 ...
##  $ attacking_volleys         : int  87 83 82 76 79 79 14 56 45 85 ...
##  $ skill_dribbling           : int  89 95 86 87 89 91 69 79 59 88 ...
##  $ skill_curve               : int  81 83 85 85 83 79 28 49 60 83 ...
##  $ skill_fk_accuracy         : int  76 79 83 78 69 63 28 49 31 73 ...
##  $ skill_long_passing        : int  77 83 91 88 75 70 63 81 65 64 ...
##  $ skill_ball_control        : int  92 94 91 92 89 90 71 80 61 89 ...
##  $ movement_acceleration     : int  89 94 77 77 94 96 69 79 61 82 ...
##  $ movement_sprint_speed     : int  91 88 76 71 92 96 73 77 73 78 ...
##  $ movement_agility          : int  87 95 78 92 91 92 52 82 57 84 ...
##  $ movement_reactions        : int  96 90 91 89 92 89 86 93 82 92 ...
##  $ movement_balance          : int  71 94 76 93 88 83 41 92 57 91 ...
##  $ power_shot_power          : int  95 82 91 79 80 83 55 71 78 89 ...
##  $ power_jumping             : int  95 56 63 68 69 76 81 77 89 81 ...
##  $ power_stamina             : int  85 84 89 85 85 84 73 97 59 79 ...
##  $ power_strength            : int  78 63 74 58 73 76 95 73 89 74 ...
##  $ power_long_shots          : int  93 80 90 82 84 79 15 63 49 84 ...
##  $ mentality_aggression      : int  63 54 76 62 63 62 87 90 91 65 ...
##  $ mentality_interceptions   : int  29 41 61 82 55 38 88 92 88 24 ...
##  $ mentality_positioning     : int  95 87 88 79 92 89 35 72 28 93 ...
##  $ mentality_vision          : int  82 89 94 91 84 80 52 79 50 83 ...
##  $ mentality_penalties       : int  85 88 79 82 77 70 33 54 50 83 ...
##  $ mentality_composure       : int  95 91 91 92 91 84 82 85 84 90 ...
##  $ defending_marking         : int  28 34 68 68 38 34 91 90 94 30 ...
##  $ defending_standing_tackle : int  32 27 58 76 43 34 90 91 91 29 ...
##  $ defending_sliding_tackle  : int  24 22 51 71 41 32 87 85 89 24 ...
```

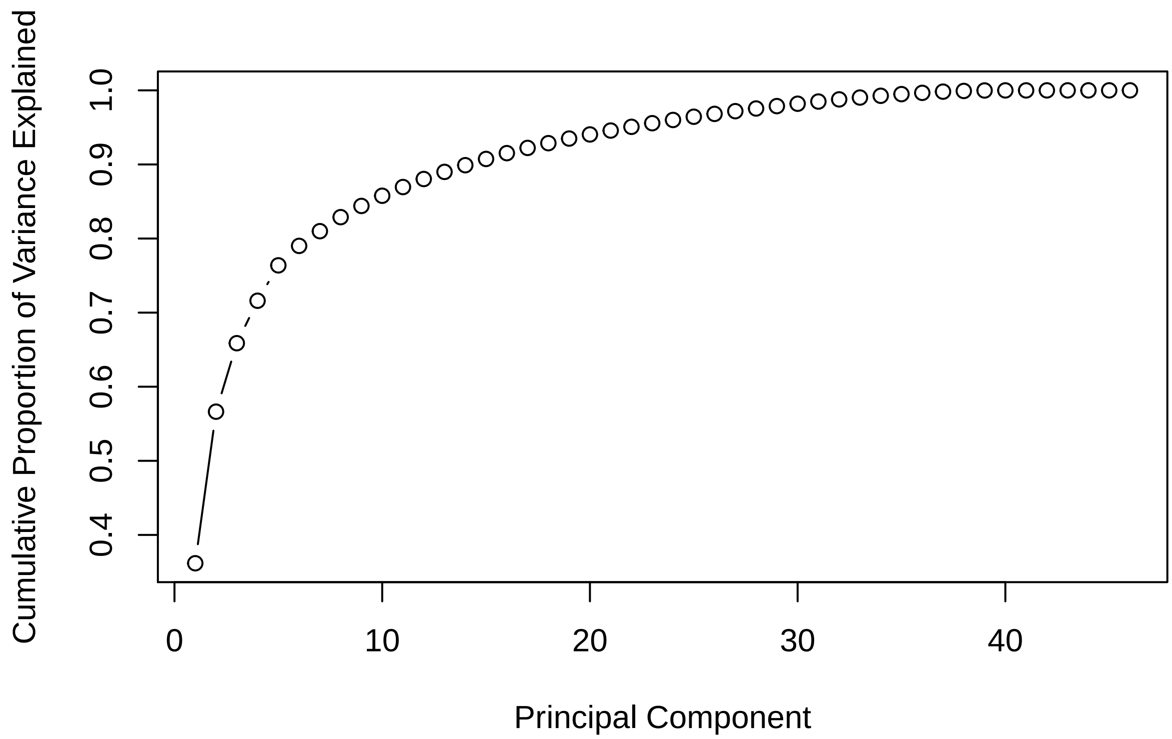
```r
dim(pca_trainset)
```

```
## [1] 12063    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 34 of the 46 features.
#Therefore we can model with these first 36 PCs.
```

#PCA Continuation:

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

new_trainset = train[, 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$attack_workrate)
```

```
## [1] 0.6745189
```

```r
#67.45% accuracy
#The accuracy didnt increase much even after performing PCA.
#confusion matrix:
confusionMatrix(as.factor(pca_testset$attack_workrate), tt)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction High  Low Medium
##     High    374    4    525
##     Low       0   47    121
##     Medium  239   92   1612
##
## Overall Statistics
##
##                Accuracy : 0.6745
##                  95% CI : (0.6575, 0.6912)
##     No Information Rate : 0.7492
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.2822
##
##  Mcnemar's Test P-Value : <2e-16
##
## Statistics by Class:
##
##                      Class: High Class: Low Class: Medium
## Sensitivity               0.6101    0.32867        0.7139
## Specificity               0.7797    0.95785        0.5622
## Pos Pred Value            0.4142    0.27976        0.8296
## Neg Pred Value            0.8868    0.96627        0.3968
## Prevalence                0.2034    0.04745        0.7492
## Detection Rate            0.1241    0.01559        0.5348
## Detection Prevalence      0.2996    0.05574        0.6447
## Balanced Accuracy         0.6949    0.64326        0.6380
```

#Decision tree model on the new dataset after PCA:

```r
fit_wrate_pca_dt <- train(class~., data=new_trainset, method="rpart",
              trControl = trCtrl, metric = "Accuracy")

tt <- predict( fit_wrate_pca_dt, new_testset)

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

```
## [1] 0.6698739
```

```r
#67% accuracy
#The accuracy didnt increase much even after performing PCA.
#confusion matrix:
confusionMatrix(as.factor(pca_testset$attack_workrate), tt)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction High  Low Medium
##     High    431    0    472
##     Low       8    0    160
##     Medium  355    0   1588
##
## Overall Statistics
##
##                Accuracy : 0.6699
##                  95% CI : (0.6528, 0.6867)
##     No Information Rate : 0.7366
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.2602
##
##  Mcnemar's Test P-Value : <2e-16
##
## Statistics by Class:
##
##                      Class: High Class: Low Class: Medium
## Sensitivity               0.5428         NA        0.7153
## Specificity               0.7874    0.94426        0.5529
## Pos Pred Value            0.4773         NA        0.8173
## Neg Pred Value            0.8280         NA        0.4099
## Prevalence                0.2634    0.00000        0.7366
## Detection Rate            0.1430    0.00000        0.5269
## Detection Prevalence      0.2996    0.05574        0.6447
## Balanced Accuracy         0.6651         NA        0.6341
```

#Correlation matrix to reduce number of features:

```
#correlaiton matrix:
dup_df <- df%>%select(-attack_workrate)
cor_mat <- cor(dup_df)
#summary of cor mat:
#print(cor_mat)
#attributes that are highly correlated:
highlyCorrelated <- findCorrelation(cor_mat, cutoff=0.75)
#indices of highly correlated attributes:
highlyCorrelated
```

```
##  [1] 15 23 14 27 13  4 40 41 37 19 21 24 31 16 28 11 12 32 45 39 44  7 17  3
```

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

```
## [1] 15077    24
```

```r
#append work_rate to dup_df:
dataset <- cbind(attack_workrate = df$attack_workrate, dup_df)
dim(dataset)
```

```
## [1] 15077    25
```

#Classification models on dataset:

```r
#split dataset into train and test sets:
set.seed(8)
smp_size <- floor(0.80 * nrow(dataset))
train_ind <- sample(seq_len(nrow(dataset)), size = smp_size)

ds_train <- dataset[train_ind, ]
ds_test <- dataset[-train_ind, ]
```

#LDA on dataset

```r
ds_fit_wrate <- train(attack_workrate~., data=ds_train, method="lda",
              trControl = trCtrl, metric = "Accuracy")


ds_pred_wrate <- predict(ds_fit_wrate, ds_test%>%select(-attack_workrate))
ds_comparison <- data.frame(original = ds_test$attack_workrate, pred = ds_pred_wrate)

#accuarcy of cross validated LDA model:
mean(ds_comparison$pred == ds_test$attack_workrate)
```

```
## [1] 0.6744032
```

```r
#67.4% accuracy

#confusion matrix:
confusionMatrix(as.factor(ds_test$attack_workrate), ds_comparison$pred)
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction High  Low Medium
##     High    354    1    522
##     Low       1   48    123
##     Medium  213  122   1632
##
## Overall Statistics
##
##                Accuracy : 0.6744
##                  95% CI : (0.6574, 0.6911)
##     No Information Rate : 0.755
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.2758
```

```
##
##  Mcnemar's Test P-Value : <2e-16
##
## Statistics by Class:
##
##                      Class: High Class: Low Class: Medium
## Sensitivity               0.6232    0.28070        0.7167
## Specificity               0.7864    0.95641        0.5467
## Pos Pred Value            0.4036    0.27907        0.8297
## Neg Pred Value            0.9000    0.95675        0.3851
## Prevalence                0.1883    0.05670        0.7550
## Detection Rate            0.1174    0.01592        0.5411
## Detection Prevalence      0.2908    0.05703        0.6522
## Balanced Accuracy         0.7048    0.61856        0.6317
```

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


##########work rates were probably not determined other features in the dataset.
#but were rather determined with a fair amount of bias involved.
```