fifa20 attack work rate 2 class

Naga Santhosh Kartheek Karnati

4/8/2020

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
```

```
library(caret)
```

```
## Loading required package: lattice

##
## Attaching package: 'caret'

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

```
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)


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>
```

```r
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
```

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

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

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

```r
df%>% count(attack_workrate)
```

```
## # A tibble: 3 x 2
##   attack_workrate     n
##   <chr>           <int>
## 1 High             4516
## 2 Low               844
## 3 Medium           9717
```

```r
df1 <- df
df2 <- df
```

#Classification:

```r
df1$attack_workrate[df1$attack_workrate == "Low"] <- "High"
df1$attack_workrate[df1$attack_workrate == "Medium"] <- "Low"

df1%>% count(attack_workrate)
```

```
## # A tibble: 2 x 2
##   attack_workrate     n
##   <chr>           <int>
## 1 High             5360
## 2 Low              9717
```

```r
#split dataset into test and train sets:
set.seed(1)
training.samples <- df1$attack_workrate %>% createDataPartition(p = 0.75, list = FALSE)
train.data  <- df1[training.samples, ]
test.data <- df1[-training.samples, ]
dim(train.data)
```

```
## [1] 11308    47
```

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

```
## [1] 3769   47
```

#LDA model:

```r
library(caret)
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
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] 0.6975325
```

```r
#69.7% accuracy

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

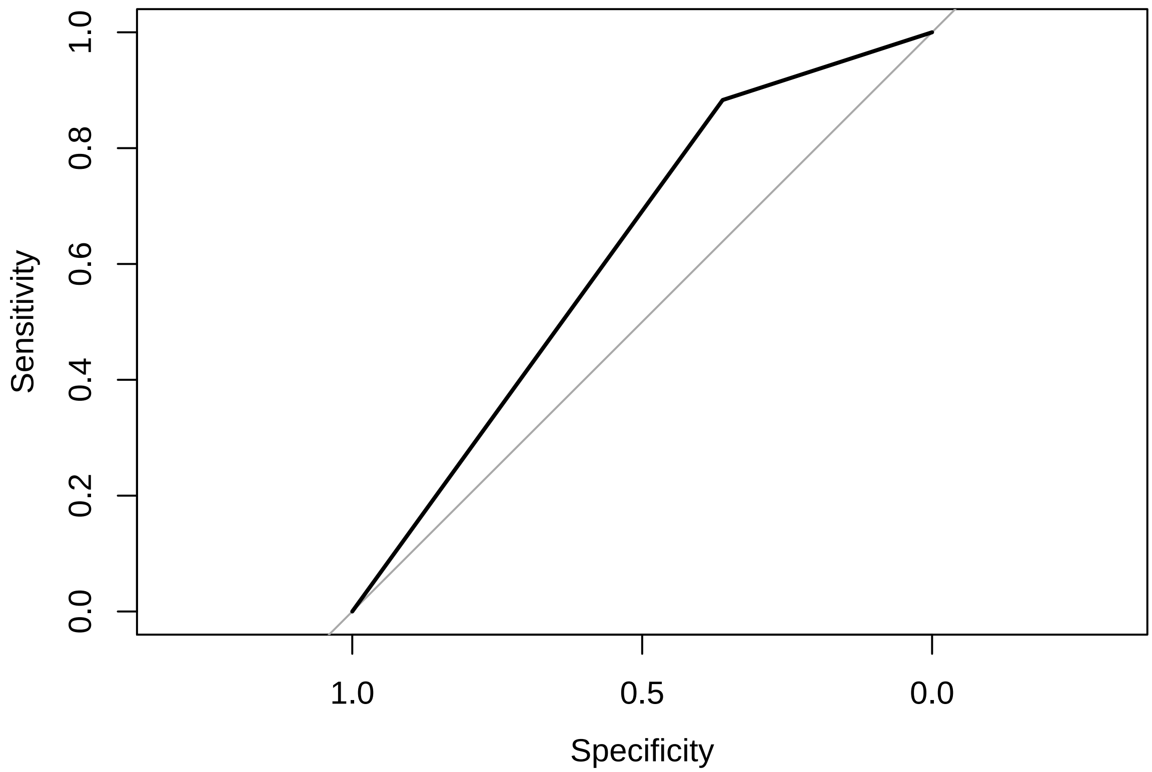
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction High  Low
##       High  484  856
##       Low   284 2145
##
##                Accuracy : 0.6975
##                  95% CI : (0.6826, 0.7122)
##     No Information Rate : 0.7962
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.2701
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.6302
##             Specificity : 0.7148
##          Pos Pred Value : 0.3612
##          Neg Pred Value : 0.8831
##              Prevalence : 0.2038
##          Detection Rate : 0.1284
##    Detection Prevalence : 0.3555
##       Balanced Accuracy : 0.6725
##
##        'Positive' Class : High
##
```

```r
lda_pred_wrate1 <- predict(lda_fit_wrate, test.data%>%select(-attack_workrate), probability=
                              TRUE)
###ROC curve:
roc_lda = roc(as.factor(test.data$attack_workrate), factor(lda_pred_wrate, ordered = TRUE),
    plot = TRUE)
```

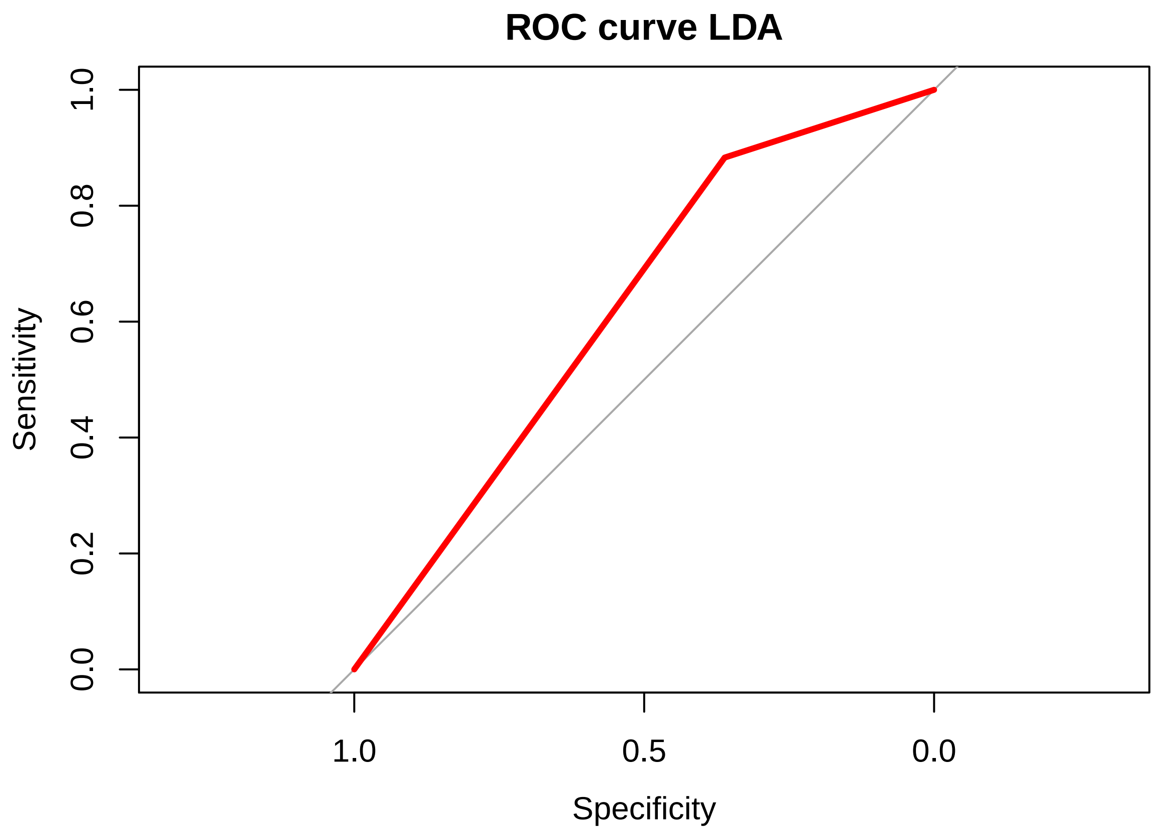
```
## Setting levels: control = High, case = Low
```

```
## Warning in value[[3L]](cond): Ordered predictor converted to numeric vector.
## Threshold values will not correspond to values in predictor.
```

```
## Setting direction: controls < cases
```




```
plot(roc_lda, col="red", lwd=3, main="ROC curve LDA")
```




```r
auc(roc_lda)
```

```
## Area under the curve: 0.6221
```

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

#Random Forest model:

```r
library(randomForest)
```

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

```
## randomForest 4.6-14
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:dplyr':
##
##     combine
```

```
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```r
#convert attack_workrate to factor:
train.data$attack_workrate <- as.factor(train.data$attack_workrate)
rf_model <- randomForest(attack_workrate ~ ., data = train.data, importance = TRUE)
rf_model
```

```
##
## Call:
##  randomForest(formula = attack_workrate ~ ., data = train.data,      importance = TRUE)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 6
##
##         OOB estimate of  error rate: 30.03%
## Confusion matrix:
##      High  Low class.error
## High 1423 2597   0.6460199
## Low   799 6489   0.1096323
```

```r
# Predicting on test set
pred_rf <- predict(rf_model, test.data, type = "class")
# Checking classification accuracy
mean(pred_rf == test.data$attack_workrate)   #70.68% accuracy
```

```
## [1] 0.7060228
```

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

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction High  Low
##       High  479  861
##       Low   247 2182
##
##                Accuracy : 0.706
##                  95% CI : (0.6912, 0.7205)
##     No Information Rate : 0.8074
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.2851
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.6598
##             Specificity : 0.7171
##          Pos Pred Value : 0.3575
##          Neg Pred Value : 0.8983
```

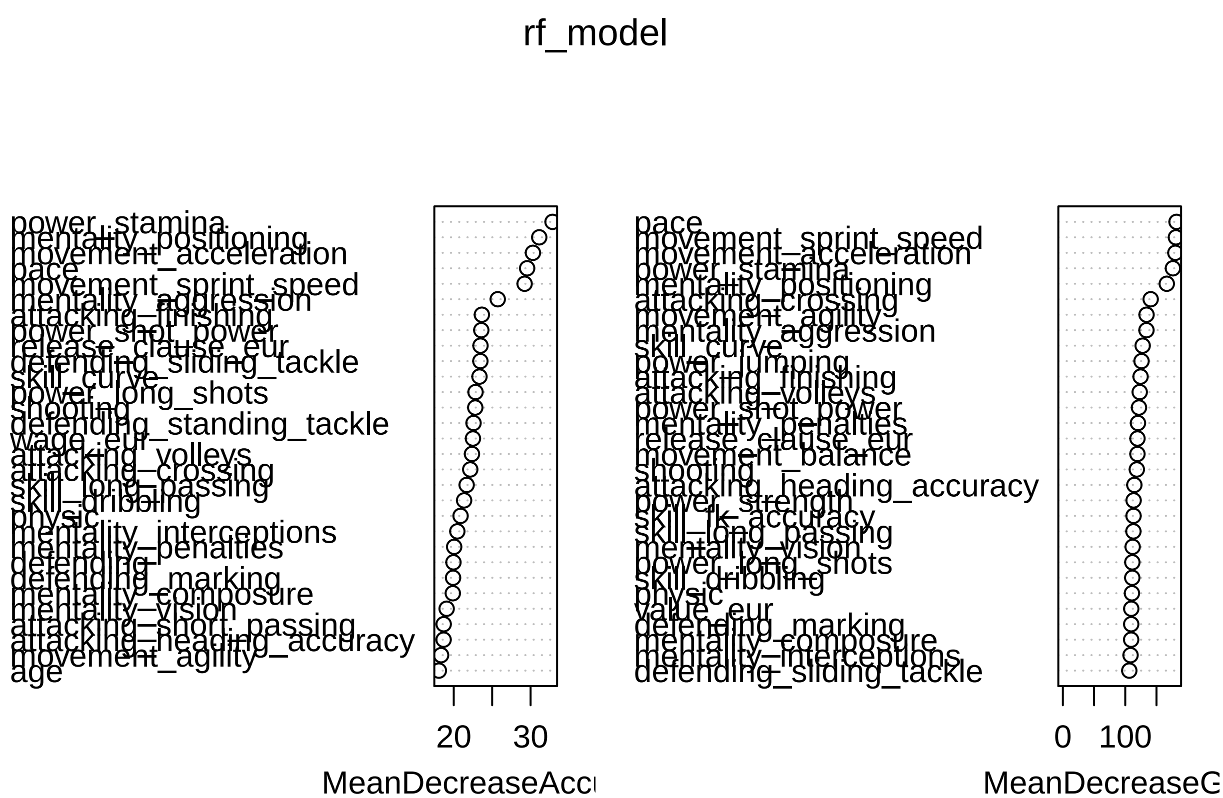
```
##               Prevalence : 0.1926
##           Detection Rate : 0.1271
##     Detection Prevalence : 0.3555
##        Balanced Accuracy : 0.6884
##
##         'Positive' Class : High
##
```

```r
#Better accuracy than LDA model

#Important variables:
importance(rf_model)
```

```
##                                   High        Low MeanDecreaseAccuracy
## age                         -2.7343931 17.7635629            18.062853
## height_cm                    1.0296011 12.3152434            13.623757
## weight_kg                   -1.0841528 12.2394354            10.926842
## overall                     -1.6714823 14.6252268            16.754806
## potential                    2.0940777 11.5528915            13.862215
## value_eur                   -0.7521512 15.8523142            17.900537
## wage_eur                    -8.6972861 25.2343340            22.497322
## international_reputation    -0.9822589  4.9472034             4.540644
## weak_foot                    4.4797098  0.9440742             3.452859
## skill_moves                  3.0798032  7.4657526            10.126939
## release_clause_eur          -0.3067862 20.0933396            23.479362
## pace                        14.5820262 19.7503377            29.551869
## shooting                    -8.1702363 21.5097383            22.800160
## passing                     -4.2151589 16.6986447            17.085175
## dribbling                   -2.6575787 15.5857196            17.716112
## defending                   -9.4851818 19.6695280            19.959929
## physic                      -6.4802849 21.4571043            20.885638
## attacking_crossing          10.3339233 14.0689254            22.146439
## attacking_finishing         -4.8988556 22.2720456            23.654199
## attacking_heading_accuracy  -9.8326595 21.4162549            18.681502
## attacking_short_passing     -4.2116491 18.5815650            18.689540
## attacking_volleys           -9.1667490 22.5014095            22.369031
## skill_dribbling              0.6066932 17.4275524            21.346848
## skill_curve                  1.5971511 20.3978350            23.346803
## skill_fk_accuracy           -7.2423610 18.6026808            15.208324
## skill_long_passing         -12.2176083 25.2878574            21.678403
## skill_ball_control          -6.4053672 16.9028746            16.818462
## movement_acceleration        9.3189690 22.6822021            30.300859
## movement_sprint_speed       21.8179461 13.3883622            29.222127
## movement_agility             6.6601375 12.8797716            18.355006
## movement_reactions          -4.0079157 19.3183673            17.682697
## movement_balance            -2.8124667 17.9622340            16.584248
## power_shot_power            -2.2272462 21.9470098            23.581819
## power_jumping               -1.6884409  8.7621512             6.999846
## power_stamina               18.2238633 23.5142281            32.856017
## power_strength              -4.1334937 17.5191119            16.681128
## power_long_shots           -10.5604234 24.2899216            22.833244
## mentality_aggression        -3.7341733 26.4361780            25.725297
## mentality_interceptions     -9.9830673 21.9127762            20.464719
## mentality_positioning        5.9562175 24.6310053            31.122002
```

```
## mentality_vision            -7.5225776 19.8930910            19.077058
## mentality_penalties        -11.0633199 23.7735056            20.053884
## mentality_composure          4.0286782 15.8232276            19.878508
## defending_marking           -6.3354564 20.7844620            19.910802
## defending_standing_tackle  -12.2636049 23.2625765            22.573629
## defending_sliding_tackle    -8.4271424 23.7212703            23.466594
##                            MeanDecreaseGini
## age                              103.135606
## height_cm                        104.259358
## weight_kg                        105.921614
## overall                           80.050972
## potential                         92.953142
## value_eur                        109.594600
## wage_eur                          83.088032
## international_reputation           7.012178
## weak_foot                         41.218017
## skill_moves                       24.758981
## release_clause_eur               119.553199
## pace                             182.173005
## shooting                         118.498843
## passing                           93.729968
## dribbling                        106.134019
## defending                         92.290454
## physic                           110.659087
## attacking_crossing               140.710952
## attacking_finishing              124.527367
## attacking_heading_accuracy       114.536078
## attacking_short_passing          101.241271
## attacking_volleys                123.149882
## skill_dribbling                  111.181710
## skill_curve                      127.904060
## skill_fk_accuracy                113.224553
## skill_long_passing               113.027266
## skill_ball_control                99.287011
## movement_acceleration            180.274669
## movement_sprint_speed            181.116849
## movement_agility                 134.068818
## movement_reactions               103.503391
## movement_balance                 119.506868
## power_shot_power                 121.941339
## power_jumping                    126.035003
## power_stamina                    176.317157
## power_strength                   113.374464
## power_long_shots                 111.308032
## mentality_aggression             133.956290
## mentality_interceptions          108.464125
## mentality_positioning            166.499741
## mentality_vision                 111.804334
## mentality_penalties              120.322873
## mentality_composure              109.408188
## defending_marking                109.527027
## defending_standing_tackle        102.337670
## defending_sliding_tackle         106.414135
```

```r
varImpPlot(rf_model)
```



#PCA

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

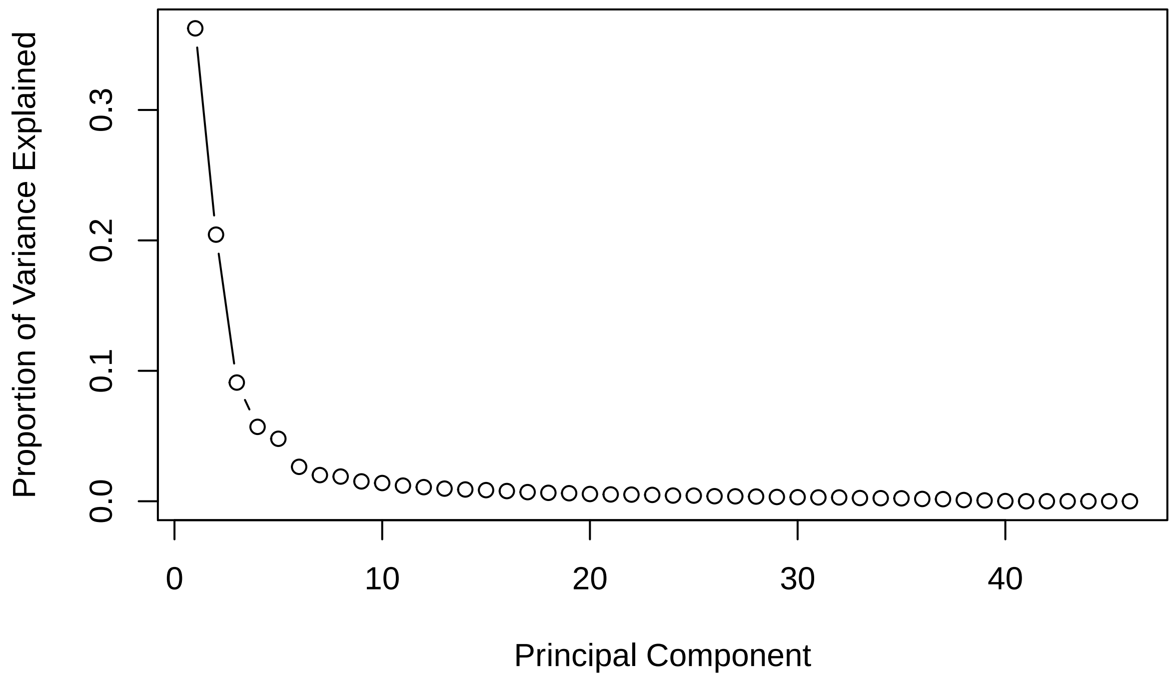
```
## Classes 'tbl_df', 'tbl' and 'data.frame':    11308 obs. of  46 variables:
##  $ age                       : int  34 28 28 20 28 28 34 30 25 26 ...
##  $ height_cm                 : int  187 175 181 178 187 168 187 189 177 191 ...
##  $ weight_kg                 : int  83 74 70 73 89 72 85 76 75 84 ...
##  $ overall                   : int  93 91 91 89 89 89 89 89 88 88 ...
##  $ potential                 : int  93 91 91 95 91 90 89 89 92 91 ...
##  $ value_eur                 : int  58500000 90000000 90000000 93500000 67500000 66000000 24500000 5
##  $ wage_eur                  : int  405000 470000 370000 155000 150000 235000 215000 300000 215000 2
##  $ international_reputation  : int  5 4 4 3 3 3 4 4 3 4 ...
##  $ weak_foot                 : int  4 4 5 4 3 3 3 3 3 4 ...
##  $ skill_moves               : int  5 4 4 5 2 2 2 3 4 5 ...
##  $ release_clause_eur        : int  96500000 184500000 166500000 191700000 119800000 130400000 40400
##  $ pace                      : int  90 91 76 96 71 78 68 42 83 74 ...
##  $ shooting                  : int  93 83 86 84 28 65 46 62 82 81 ...
##  $ passing                   : int  82 86 92 78 54 77 58 80 84 86 ...
##  $ dribbling                 : int  89 94 86 90 67 81 60 80 90 85 ...
```

```
##  $ defending                 : int  35 35 61 39 89 87 90 85 43 66 ...
##  $ physic                    : int  78 66 78 75 87 83 82 80 64 86 ...
##  $ attacking_crossing        : int  84 81 93 78 30 68 54 62 82 80 ...
##  $ attacking_finishing       : int  94 84 82 89 22 65 33 67 80 75 ...
##  $ attacking_heading_accuracy: int  89 61 55 77 83 54 83 68 64 75 ...
##  $ attacking_short_passing   : int  83 89 92 82 71 86 65 89 87 86 ...
##  $ attacking_volleys         : int  87 83 82 79 14 56 45 44 88 84 ...
##  $ skill_dribbling           : int  89 95 86 91 69 79 59 80 90 87 ...
##  $ skill_curve               : int  81 83 85 79 28 49 60 66 88 85 ...
##  $ skill_fk_accuracy         : int  76 79 83 63 28 49 31 68 88 82 ...
##  $ skill_long_passing        : int  77 83 91 70 63 81 65 82 75 90 ...
##  $ skill_ball_control        : int  92 94 91 90 71 80 61 88 93 90 ...
##  $ movement_acceleration     : int  89 94 77 96 69 79 61 40 86 67 ...
##  $ movement_sprint_speed     : int  91 88 76 96 73 77 73 43 81 79 ...
##  $ movement_agility          : int  87 95 78 92 52 82 57 67 91 75 ...
##  $ movement_reactions        : int  96 90 91 89 86 93 82 87 84 82 ...
##  $ movement_balance          : int  71 94 76 83 41 92 57 49 85 66 ...
##  $ power_shot_power          : int  95 82 91 83 55 71 78 61 80 90 ...
##  $ power_jumping             : int  95 56 63 76 81 77 89 66 75 82 ...
##  $ power_stamina             : int  85 84 89 84 73 97 59 86 79 87 ...
##  $ power_strength            : int  78 63 74 76 95 73 89 77 61 89 ...
##  $ power_long_shots          : int  93 80 90 79 15 63 49 54 86 82 ...
##  $ mentality_aggression      : int  63 54 76 62 87 90 91 85 48 78 ...
##  $ mentality_interceptions   : int  29 41 61 38 88 92 88 89 42 64 ...
##  $ mentality_positioning     : int  95 87 88 89 35 72 28 77 80 83 ...
##  $ mentality_vision          : int  82 89 94 80 52 79 50 86 87 88 ...
##  $ mentality_penalties       : int  85 88 79 70 33 54 50 60 86 83 ...
##  $ mentality_composure       : int  95 91 91 84 82 85 84 93 84 87 ...
##  $ defending_marking         : int  28 34 68 34 91 90 94 90 32 63 ...
##  $ defending_standing_tackle : int  32 27 58 34 90 91 91 86 48 67 ...
##  $ defending_sliding_tackle  : int  24 22 51 32 87 85 89 80 40 65 ...
```

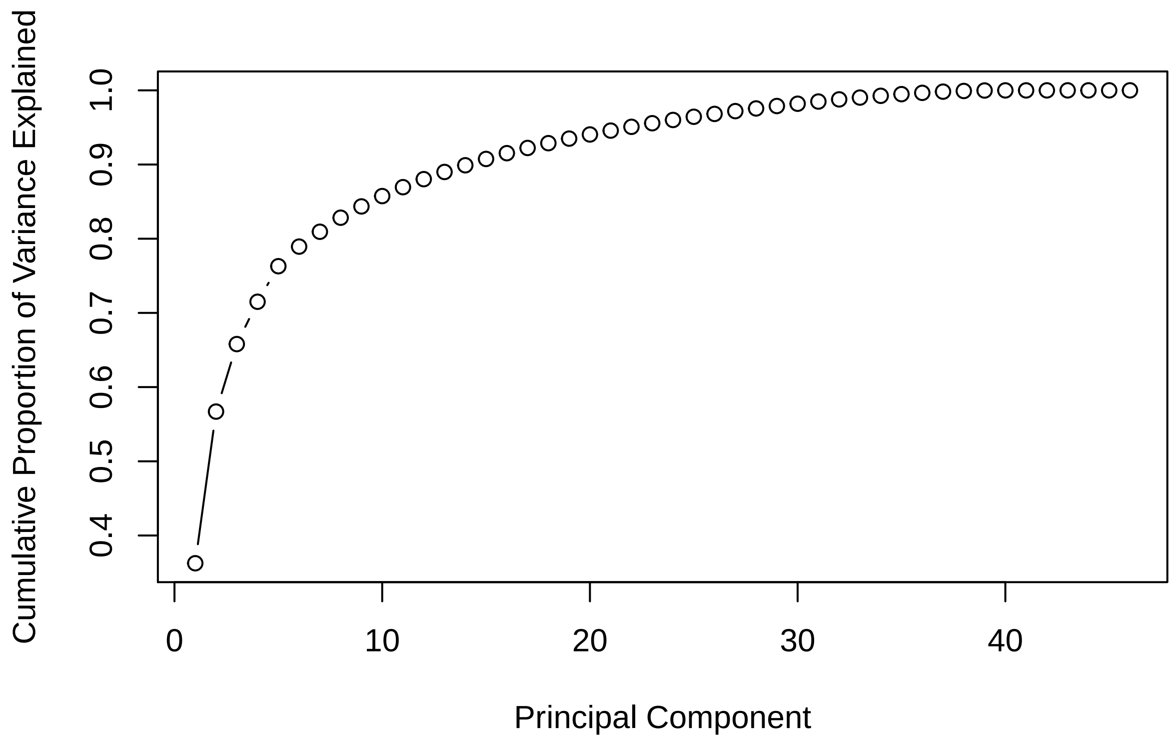
```r
dim(pca_trainset)
```

```
## [1] 11308    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" )
```




```
#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 95% of the variance explained is done by 34 of the 46 features.
#Therefore we can model with these first 26 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[, 1:27]
new_testset =  t[, 1:26]
```

#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] 0.6972672
```

```r
#69.73% 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
##       High  471  869
##       Low   272 2157
##
##                Accuracy : 0.6973
##                  95% CI : (0.6823, 0.7119)
##     No Information Rate : 0.8029
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.2661
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.6339
##             Specificity : 0.7128
##          Pos Pred Value : 0.3515
##          Neg Pred Value : 0.8880
##              Prevalence : 0.1971
##          Detection Rate : 0.1250
##    Detection Prevalence : 0.3555
##       Balanced Accuracy : 0.6734
##
##        'Positive' Class : High
##
```

#Random Forest:

```r
#convert attack_workrate to factor:
new_trainset$class <- as.factor(new_trainset$class)
rf_model_pca <- randomForest(class ~ ., data = new_trainset, importance = TRUE)
rf_model_pca
```

```
##
## Call:
##  randomForest(formula = class ~ ., data = new_trainset, importance = TRUE)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 5
##
##         OOB estimate of  error rate: 30.83%
## Confusion matrix:
##      High  Low class.error
## High 1309 2711   0.6743781
## Low   775 6513   0.1063392
```

```r
# Predicting on test set
pred_rf_pca <- predict(rf_model_pca, new_testset, type = "class")
# Checking classification accuracy
mean(pred_rf_pca == pca_testset$attack_workrate)   #70.5% accuracy
```

```
## [1] 0.700451
```

```r
#confusion matrix:
confusionMatrix(as.factor(pca_testset$attack_workrate), pred_rf_pca)
```

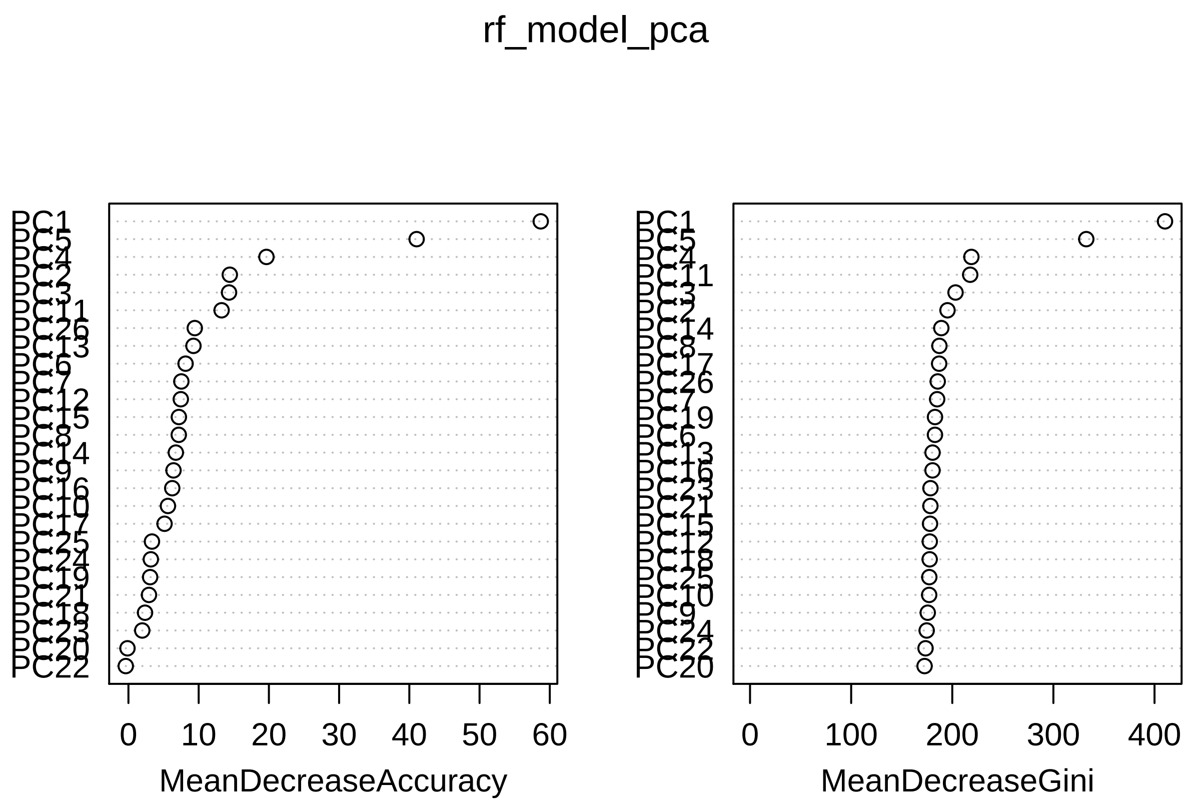
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction High  Low
##       High  446  894
##       Low   235 2194
##
##                Accuracy : 0.7005
##                  95% CI : (0.6855, 0.715)
##     No Information Rate : 0.8193
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.2653
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.6549
##             Specificity : 0.7105
##          Pos Pred Value : 0.3328
##          Neg Pred Value : 0.9033
##              Prevalence : 0.1807
##          Detection Rate : 0.1183
##    Detection Prevalence : 0.3555
##       Balanced Accuracy : 0.6827
##
##        'Positive' Class : High
##
```

```r
#Better accuracy than LDA model

#Important variables:
importance(rf_model_pca)
```

```
##             High        Low MeanDecreaseAccuracy MeanDecreaseGini
## PC1  42.50816042 39.9215648           58.7128301         410.3769
## PC2   3.18850090 14.2616088           14.4359021         195.2918
## PC3   3.30036780 14.1098806           14.3237638         203.2130
## PC4  -1.37094397 22.5930852           19.6512261         218.8915
## PC5  39.91680885 18.1918963           41.0409352         332.5245
## PC6  -0.84251357 10.6710669            8.1341852         182.8683
## PC7   0.75406114  8.7045118            7.5326018         185.0517
## PC8  -1.80268799  9.4975405            7.1680162         187.2684
```

```
## PC9  -0.39342088  8.2296357           6.4132795        175.7457
## PC10 -1.97407809  8.4215125           5.6236022        177.1422
## PC11  5.61564437 12.4098225          13.2658847        217.7194
## PC12 -1.00879286  9.5846196           7.4555289        177.7323
## PC13 -2.40591430 12.4221931           9.2555754        180.5035
## PC14  7.85678494  2.0626067           6.7489943        189.1618
## PC15 -1.72045860  9.9233769           7.1806077        178.0028
## PC16 -1.79206827  9.4163755           6.2355375        180.4679
## PC17  4.15576256  3.3407497           5.1394788        187.0298
## PC18 -0.81073941  3.6766935           2.3598583        177.6493
## PC19 -1.19456554  4.7329864           3.0870479        182.9110
## PC20 -0.39778555  0.1352754          -0.1398694        172.6124
## PC21  2.09261851  2.0901244           2.9222853        178.3707
## PC22  0.05245234 -0.4817811          -0.3803513        173.5573
## PC23  2.58564471  0.4523847           1.9646101        178.4157
## PC24  0.18704222  3.8339245           3.1891537        174.7019
## PC25  0.57678110  3.6722752           3.3496288        177.2796
## PC26 -1.10015215 12.3352594           9.4447115        185.6051
```

```r
varImpPlot(rf_model_pca)
```



#Dealing with class imbalance:

```r
# Set up control function for training

ctrl <- trainControl(method = "repeatedcv",
```

```r
                     number = 10,
                     repeats = 3,
                     summaryFunction = twoClassSummary,
                     classProbs = TRUE)

# Build a standard classifier using a gradient boosted machine

set.seed(5627)

orig_fit <- train(attack_workrate ~ .,
                  data = train.data,
                  method = "lda",
                  verbose = FALSE,
                  metric = "ROC",
                  trControl = ctrl)

# Build custom AUC function to extract AUC
# from the caret model object

test_roc <- function(model, data) {

  roc(data$attack_workrate,
      predict(model, data, type = "prob")[, "High"])

}

orig_fit %>%
  test_roc(data = test.data) %>%
  auc()
```

```
## Setting levels: control = High, case = Low

## Setting direction: controls > cases

## Area under the curve: 0.6908
```

```r
library(DMwR) #for smote
```

```
## Warning: package 'DMwR' was built under R version 3.6.3

## Loading required package: grid

## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo

##
## Attaching package: 'DMwR'

## The following object is masked from 'package:modelr':
##
##     bootstrap
```

```r
#Upscaling, downscaling and SMOTE techniques:
# Use the same seed to ensure same cross-validation splits

ctrl$seeds <- orig_fit$control$seeds

# Build down-sampled model

ctrl$sampling <- "down"

down_fit <- train(attack_workrate ~ .,
                  data = train.data,
                  method = "lda",
                  verbose = FALSE,
                  metric = "ROC",
                  trControl = ctrl)

# Build up-sampled model

ctrl$sampling <- "up"

up_fit <- train(attack_workrate ~ .,
                data = train.data,
                method = "lda",
                verbose = FALSE,
                metric = "ROC",
                trControl = ctrl)

# Build smote model

ctrl$sampling <- "smote"

smote_fit <- train(attack_workrate ~ .,
                   data = train.data,
                   method = "lda",
                   verbose = FALSE,
                   metric = "ROC",
                   trControl = ctrl)
```

```r
# Examine results for test set

model_list <- list(original = orig_fit,
                   down = down_fit,
                   up = up_fit,
                   SMOTE = smote_fit)

model_list_roc <- model_list %>%
  map(test_roc, data = test.data)
```

```
## Setting levels: control = High, case = Low

## Setting direction: controls > cases

## Setting levels: control = High, case = Low
```

```
## Setting direction: controls > cases

## Setting levels: control = High, case = Low

## Setting direction: controls > cases

## Setting levels: control = High, case = Low

## Setting direction: controls > cases
```

```
model_list_roc %>%
  map(auc)
```

```
## $original
## Area under the curve: 0.6908
##
## $down
## Area under the curve: 0.691
##
## $up
## Area under the curve: 0.69
##
## $SMOTE
## Area under the curve: 0.6914
```

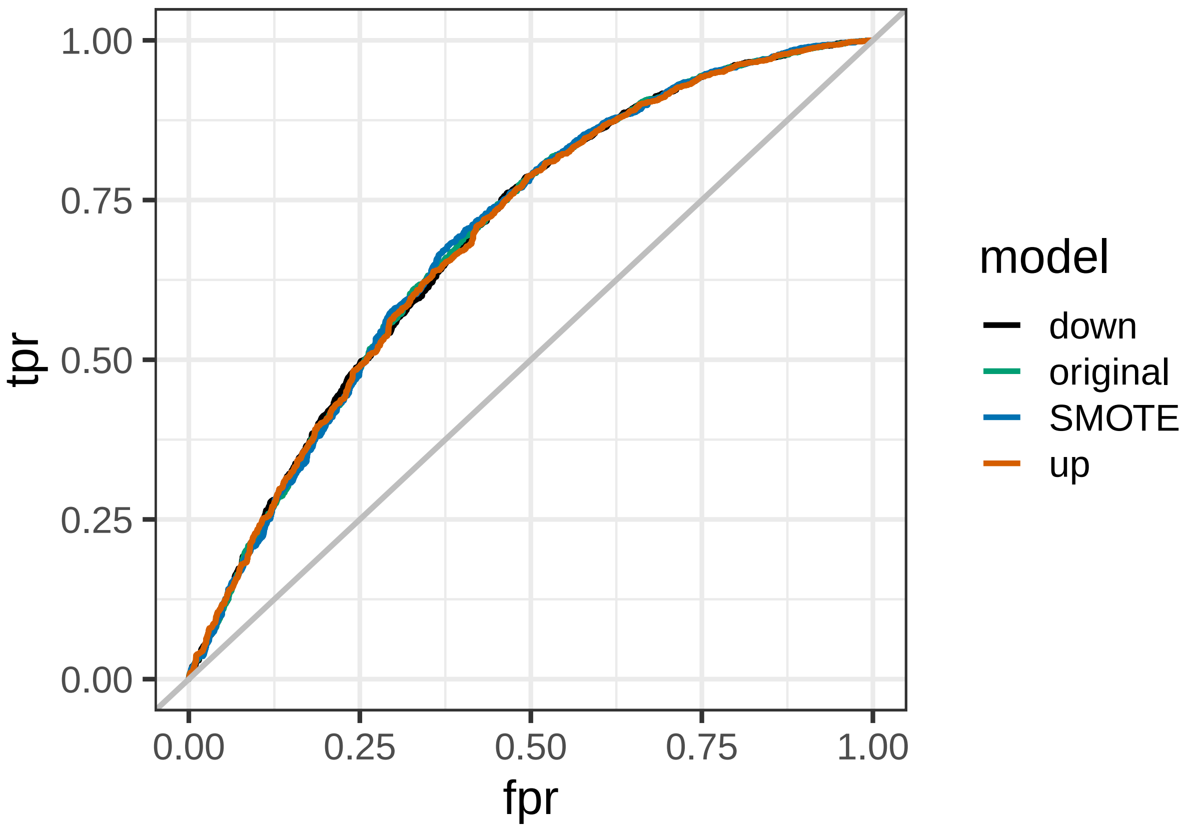
```
#smote method results are slightly better than the other 3 models.
```

```
results_list_roc <- list(NA)
num_mod <- 1

for(the_roc in model_list_roc){

  results_list_roc[[num_mod]] <-
    data_frame(tpr = the_roc$sensitivities,
               fpr = 1 - the_roc$specificities,
               model = names(model_list)[num_mod])

  num_mod <- num_mod + 1

}
```

```
## Warning: `data_frame()` is deprecated, use `tibble()`.
## This warning is displayed once per session.
```

```
results_df_roc <- bind_rows(results_list_roc)

# Plot ROC curve for all 4 models

custom_col <- c("#000000", "#009E73", "#0072B2", "#D55E00")

ggplot(aes(x = fpr,  y = tpr, group = model), data = results_df_roc) +
```

```r
  geom_line(aes(color = model), size = 1) +
  scale_color_manual(values = custom_col) +
  geom_abline(intercept = 0, slope = 1, color = "gray", size = 1) +
  theme_bw(base_size = 18)
```




```r
#predictions of the smote fit:
pred_smote <- predict(smote_fit, test.data)

#accuracy:
mean(pred_smote == test.data$attack_workrate)
```

```
## [1] 0.6800212
```

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

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction High  Low
##       High  683  549
##       Low   657 1880
##
##                Accuracy : 0.68
```

```
##                95% CI : (0.6649, 0.6949)
##     No Information Rate : 0.6445
##     P-Value [Acc > NIR] : 2.35e-06
##
##                   Kappa : 0.2889
##
##  Mcnemar's Test P-Value : 0.002062
##
##             Sensitivity : 0.5097
##             Specificity : 0.7740
##          Pos Pred Value : 0.5544
##          Neg Pred Value : 0.7410
##              Prevalence : 0.3555
##          Detection Rate : 0.1812
##    Detection Prevalence : 0.3269
##       Balanced Accuracy : 0.6418
##
##        'Positive' Class : High
##
```