Random Forest, Visuals

Andrew Henderson

2025-03-31

```
## Loading required package: randomForest
## randomForest 4.7-1.2
## Type rfNews() to see new features/changes/bug fixes.
## Loading required package: caret
## Loading required package: ggplot2
##
## Attaching package: 'ggplot2'
## The following object is masked from 'package:randomForest':
##
##     margin
## Loading required package: lattice
```

```
data <- read.csv("transfer_dataset.csv", stringsAsFactors = T)
```

```
str(data)
```

```
## 'data.frame':    4124 obs. of  32 variables:
##  $ transfer                   : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ player                     : Factor w/ 2326 levels "Aapo Halme","Aaron Leya Iseka",..: 658 1874 
##  $ age                        : int  24 26 26 30 21 25 21 22 22 31 ...
##  $ season                     : Factor w/ 4 levels "2017-2018","2018-2019",..: 1 1 1 1 1 1 1 1 1 1
##  $ minutes                    : num  3060 1006 965 2415 3060 ...
##  $ matches_played             : int  34 26 14 29 34 16 24 22 38 33 ...
##  $ play_proportion            : num  1 0.867 0.5 1 1 ...
##  $ raw_goals                  : int  7 2 0 8 3 0 2 2 8 1 ...
##  $ raw_assists                : int  3 3 1 1 0 2 1 3 0 6 ...
##  $ raw_nonpenaltykick_goal    : int  3 2 0 4 3 0 2 2 8 1 ...
##  $ total_pass_attempts        : int  2003 568 744 1016 1247 678 1042 211 652 2024 ...
##  $ Total_Cmp.                 : num  85.6 70.8 90.3 55.7 79.3 78.5 84.1 66.4 65 82.5 ...
##  $ Total_Passes_Leading_to_Shot: int  36 16 2 44 5 8 11 8 29 55 ...
##  $ Cmp_Passes_18_yard_box     : int  28 13 0 37 4 10 7 4 9 30 ...
##  $ prog_dist_per_pass         : num  37 31.8 44.4 39.1 82.4 ...
##  $ Shot_Creating_Actions      : int  92 47 12 105 23 27 46 16 69 101 ...
##  $ def_action_to_shot         : int  5 0 0 2 0 0 0 1 2 0 ...
##  $ goal_creating_action       : int  18 9 2 7 0 3 4 4 3 13 ...
##  $ takeon_to_goal             : int  4 1 0 0 0 0 1 0 1 0 ...
##  $ prog_dist_per_carry        : num  59 31.6 110 24.8 203.6 ...
##  $ Progressive_Passes_Received : int  57 127 6 316 1 30 51 82 179 76 ...
##  $ TakeOn_Attempts            : int  51 55 4 143 15 54 75 25 35 29 ...
##  $ TakeOn_Success_Percentage  : num  72.5 52.7 75 55.2 73.3 61.1 73.3 20 62.9 72.4 ...
```

```
##  $ tackle_ratio               : num  0.561 0.774 0.462 0.571 0.741 ...
##  $ Shot_Blocks                : int  13 0 1 2 25 4 2 0 1 12 ...
##  $ Pass_Blocks                : int  35 17 8 37 13 11 11 13 7 40 ...
##  $ Clearances                 : int  62 2 10 30 128 33 31 5 5 54 ...
##  $ Aerial_Win_Percentage      : num  67.6 29 50 20.6 61.3 38.5 66.7 35.7 38.9 51 ...
##  $ aerials_total              : int  139 31 6 63 93 13 33 42 18 49 ...
##  $ FW                         : int  0 1 0 1 0 0 0 1 1 0 ...
##  $ MF                         : int  1 1 1 1 0 0 1 0 1 1 ...
##  $ DF                         : int  0 0 0 0 1 1 1 0 0 0 ...
```

```
summary(data)
```

```
##     transfer                    player          age              season
##  Min.   :0.00000   Abdoulaye Bamba :   4   Min.   :16.00   2017-2018: 379
##  1st Qu.:0.00000   Abdoulaye Touré :   4   1st Qu.:22.00   2018-2019:1208
##  Median :0.00000   Adrien Hunou    :   4   Median :25.00   2019-2020:1244
##  Mean   :0.06159   Adrien Thomasson:   4   Mean   :25.54   2020-2021:1293
##  3rd Qu.:0.00000   Alexander Djiku :   4   3rd Qu.:28.00
##  Max.   :1.00000   Andrei Girotto  :   4   Max.   :42.00
##                    (Other)         :4100
##     minutes     matches_played  play_proportion    raw_goals
##  Min.   : 400   Min.   : 5.00   Min.   :0.1667   Min.   : 0.000
##  1st Qu.:1000   1st Qu.:18.00   1st Qu.:0.7273   1st Qu.: 0.000
##  Median :1640   Median :25.00   Median :0.8750   Median : 1.000
##  Mean   :1751   Mean   :25.15   Mean   :0.8259   Mean   : 2.479
##  3rd Qu.:2389   3rd Qu.:32.00   3rd Qu.:0.9688   3rd Qu.: 3.000
##  Max.   :4140   Max.   :46.00   Max.   :1.0000   Max.   :33.000
##
##   raw_assists     raw_nonpenaltykick_goal total_pass_attempts   Total_Cmp.
##  Min.   : 0.000   Min.   : 0.000          Min.   :  59.0      Min.   :35.80
##  1st Qu.: 0.000   1st Qu.: 0.000          1st Qu.: 402.0      1st Qu.:68.90
##  Median : 1.000   Median : 1.000          Median : 738.5      Median :74.85
##  Mean   : 1.689   Mean   : 2.239          Mean   : 851.6      Mean   :74.26
##  3rd Qu.: 2.000   3rd Qu.: 3.000          3rd Qu.:1197.2      3rd Qu.:80.40
##  Max.   :17.000   Max.   :32.000          Max.   :3456.0      Max.   :96.30
##
##  Total_Passes_Leading_to_Shot Cmp_Passes_18_yard_box prog_dist_per_pass
##  Min.   :  0.00               Min.   :  0.00         Min.   :  0.00
##  1st Qu.:  6.00               1st Qu.:  4.00         1st Qu.: 29.24
##  Median : 13.00               Median : 10.00         Median : 41.73
##  Mean   : 17.89               Mean   : 13.78         Mean   : 55.17
##  3rd Qu.: 24.00               3rd Qu.: 19.00         3rd Qu.: 70.46
##  Max.   :122.00               Max.   :137.00         Max.   :380.50
##
##  Shot_Creating_Actions def_action_to_shot goal_creating_action takeon_to_goal
##  Min.   :  0.0         Min.   :0.0000     Min.   : 0.00        Min.   :0.0000
##  1st Qu.: 17.0         1st Qu.:0.0000     1st Qu.: 1.00        1st Qu.:0.0000
##  Median : 33.0         Median :0.0000     Median : 3.00        Median :0.0000
##  Mean   : 42.2         Mean   :0.7085     Mean   : 4.21        Mean   :0.2789
##  3rd Qu.: 57.0         3rd Qu.:1.0000     3rd Qu.: 6.00        3rd Qu.:0.0000
##  Max.   :238.0         Max.   :7.0000     Max.   :36.00        Max.   :7.0000
##
##  prog_dist_per_carry Progressive_Passes_Received TakeOn_Attempts
##  Min.   :   0.00     Min.   :  0.00              Min.   :  1.00
##  1st Qu.:  30.50     1st Qu.: 17.00              1st Qu.:  8.00
```

```
##  Median :  41.74     Median : 52.00              Median : 20.00
##  Mean   :  80.31     Mean   : 72.52              Mean   : 28.71
##  3rd Qu.:  78.80     3rd Qu.:107.00              3rd Qu.: 39.00
##  Max.   :2533.00     Max.   :464.00              Max.   :252.00
##
##  TakeOn_Success_Percentage  tackle_ratio     Shot_Blocks      Pass_Blocks
##  Min.   :  0.00             Min.   :0.0000   Min.   : 0.000   Min.   : 0.00
##  1st Qu.: 49.48             1st Qu.:0.5500   1st Qu.: 1.000   1st Qu.: 6.00
##  Median : 58.80             Median :0.6250   Median : 3.000   Median :11.00
##  Mean   : 59.89             Mean   :0.6221   Mean   : 5.996   Mean   :13.63
##  3rd Qu.: 70.40             3rd Qu.:0.7000   3rd Qu.: 8.000   3rd Qu.:18.00
##  Max.   :100.00             Max.   :1.0000   Max.   :63.000   Max.   :65.00
##
##    Clearances     Aerial_Win_Percentage aerials_total          FW
##  Min.   :  0.00   Min.   :  0.00        Min.   :  1.00   Min.   :0.0000
##  1st Qu.: 10.00   1st Qu.: 35.20        1st Qu.: 29.00   1st Qu.:0.0000
##  Median : 24.00   Median : 46.75        Median : 52.00   Median :0.0000
##  Mean   : 40.21   Mean   : 45.65        Mean   : 74.16   Mean   :0.3654
##  3rd Qu.: 53.00   3rd Qu.: 56.83        3rd Qu.: 93.00   3rd Qu.:1.0000
##  Max.   :293.00   Max.   :100.00        Max.   :977.00   Max.   :1.0000
##
##        MF               DF
##  Min.   :0.0000   Min.   :0.0000
##  1st Qu.:0.0000   1st Qu.:0.0000
##  Median :0.0000   Median :0.0000
##  Mean   :0.4968   Mean   :0.4178
##  3rd Qu.:1.0000   3rd Qu.:1.0000
##  Max.   :1.0000   Max.   :1.0000
##
```

```
data$transfer <- as.factor(data$transfer)
data$MF <- as.factor(data$MF)
data$DF <- as.factor(data$DF)
data$FW <- as.factor(data$FW)

data <- na.omit(data)
```

```
set.seed(82)
trainIndex <- createDataPartition(data$transfer, p = 0.7, list = FALSE)
trainData <- data[trainIndex, ]
testData <- data[-trainIndex, ]
```

```
rf_model <- randomForest(transfer ~ . - season - player,
                         data = trainData,
                         ntree = 500,
                         mtry = floor(sqrt(ncol(trainData) - 1)),
                         importance = TRUE)
```

```
print(rf_model)
```

```
##
## Call:
##  randomForest(formula = transfer ~ . - season - player, data = trainData,      ntree = 500, mtry = fl
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 5
```

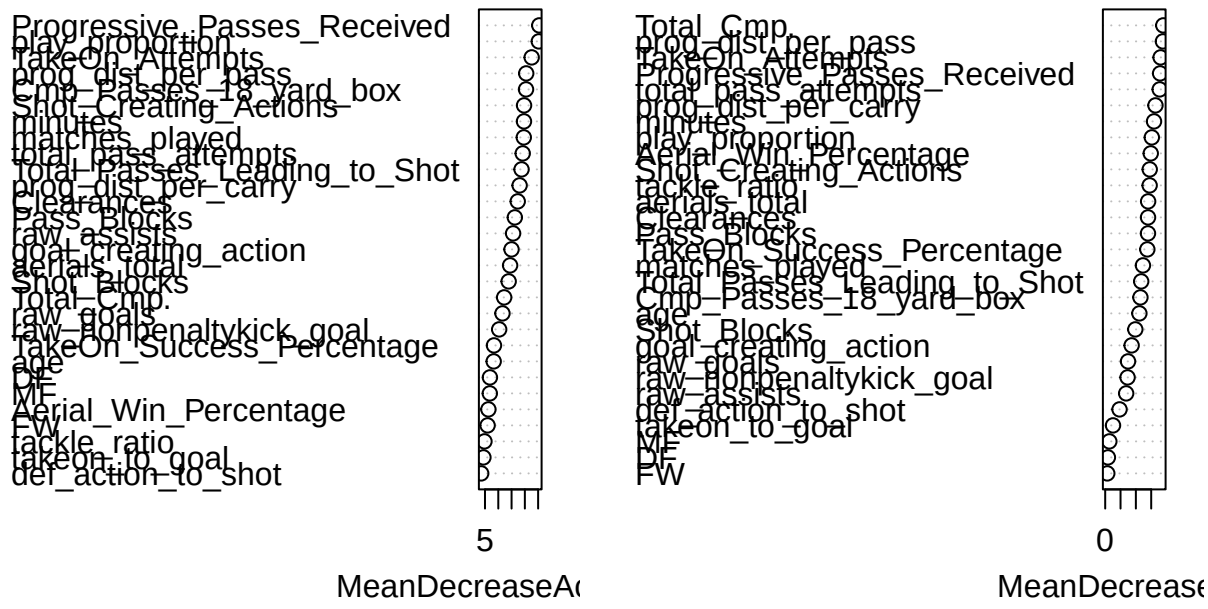
```
##
##         OOB estimate of  error rate: 6.03%
## Confusion matrix:
##      0 1 class.error
## 0 2709 0   0.0000000
## 1  174 4   0.9775281
```

```
predictions <- predict(rf_model, testData)
```

```
confMat <- confusionMatrix(predictions, testData$transfer)
print(confMat)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1161   72
##          1    0    4
##
##                Accuracy : 0.9418
##                  95% CI : (0.9273, 0.9542)
##     No Information Rate : 0.9386
##     P-Value [Acc > NIR] : 0.3446
##
##                   Kappa : 0.0944
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 1.00000
##             Specificity : 0.05263
##          Pos Pred Value : 0.94161
##          Neg Pred Value : 1.00000
##              Prevalence : 0.93856
##          Detection Rate : 0.93856
##    Detection Prevalence : 0.99677
##       Balanced Accuracy : 0.52632
##
##        'Positive' Class : 0
##
```

```
varImpPlot(rf_model)
```



Random Forest Models / Position

```r
positions <- c("MF", "FW", "DF")

models <- list()
conf_matrices <- list()

for (pos in positions) {

  # Subset the data for players that play the current position.
  # Assumes that a value of 1 indicates the player plays that position.
  pos_data <- subset(data, data[[pos]] == 1)

  # Remove unwanted columns: season, player, and all position indicator columns
  pos_data <- pos_data[, !(names(pos_data) %in% c("season", "player", "MD", "FW", "DF"))]

  # Ensure the response variable is a factor (should be already, but just in case)
  pos_data$transfer <- as.factor(pos_data$transfer)

  # Check that there is enough data for this subset; adjust the threshold as needed.
  if(nrow(pos_data) < 10) {
    cat("Not enough data for position", pos, "\n")
    next
  }
   set.seed(42)  # For reproducibility
  trainIndex <- createDataPartition(pos_data$transfer, p = 0.7, list = FALSE)
  trainData <- pos_data[trainIndex, ]
```

```r
  testData <- pos_data[-trainIndex, ]

  # Build the Random Forest model
  rf_model <- randomForest(transfer ~ .,
                           data = trainData,
                           ntree = 500,
                           mtry = floor(sqrt(ncol(trainData) - 1)),
                           importance = TRUE)

  # Store the model in the list
  models[[pos]] <- rf_model

  # Make predictions on the test set
  predictions <- predict(rf_model, testData)

  # Evaluate the model performance using a confusion matrix
  conf_mat <- confusionMatrix(predictions, testData$transfer)
  conf_matrices[[pos]] <- conf_mat

  cat("Confusion Matrix for", pos, "players:\n")
  print(conf_mat)
  cat("\n----------------------------------\n")
}
```

```
## Confusion Matrix for MF players:
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 579  33
##          1   0   2
##
##                Accuracy : 0.9463
##                  95% CI : (0.9253, 0.9627)
##     No Information Rate : 0.943
##     P-Value [Acc > NIR] : 0.4064
##
##                   Kappa : 0.1026
##
##  Mcnemar's Test P-Value : 2.54e-08
##
##             Sensitivity : 1.00000
##             Specificity : 0.05714
##          Pos Pred Value : 0.94608
##          Neg Pred Value : 1.00000
##              Prevalence : 0.94300
##          Detection Rate : 0.94300
##    Detection Prevalence : 0.99674
##       Balanced Accuracy : 0.52857
##
##        'Positive' Class : 0
##
##
## ----------------------------------
```

```
## Confusion Matrix for FW players:
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 421  27
##          1   1   3
##
##                Accuracy : 0.9381
##                  95% CI : (0.9117, 0.9584)
##     No Information Rate : 0.9336
##     P-Value [Acc > NIR] : 0.3982
##
##                   Kappa : 0.1634
##
##  Mcnemar's Test P-Value : 2.306e-06
##
##             Sensitivity : 0.9976
##             Specificity : 0.1000
##          Pos Pred Value : 0.9397
##          Neg Pred Value : 0.7500
##              Prevalence : 0.9336
##          Detection Rate : 0.9314
##    Detection Prevalence : 0.9912
##       Balanced Accuracy : 0.5488
##
##        'Positive' Class : 0
##
##
## --------------------------------
## Confusion Matrix for DF players:
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 485  30
##          1   0   1
##
##                Accuracy : 0.9419
##                  95% CI : (0.918, 0.9604)
##     No Information Rate : 0.9399
##     P-Value [Acc > NIR] : 0.4739
##
##                   Kappa : 0.059
##
##  Mcnemar's Test P-Value : 1.192e-07
##
##             Sensitivity : 1.00000
##             Specificity : 0.03226
##          Pos Pred Value : 0.94175
##          Neg Pred Value : 1.00000
##              Prevalence : 0.93992
##          Detection Rate : 0.93992
##    Detection Prevalence : 0.99806
```

```
##       Balanced Accuracy : 0.51613
##
##        'Positive' Class : 0
##
##
## ---------------------------------
```

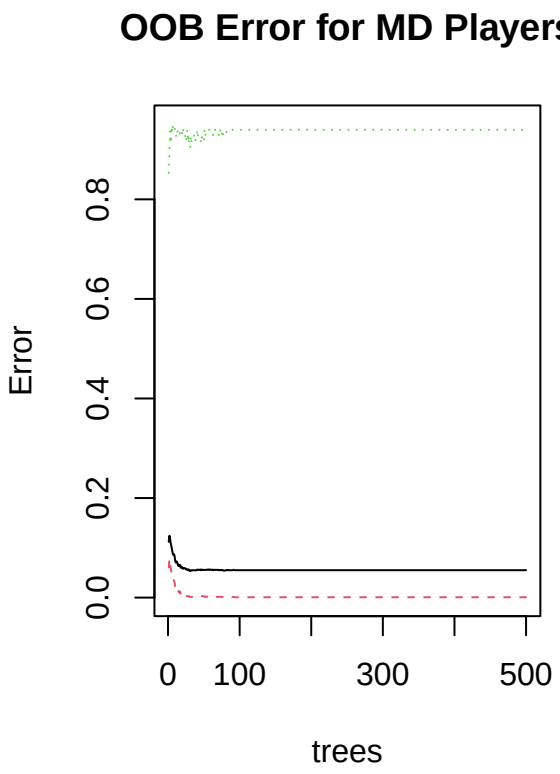
```
# For MF players:
cat("Model summary for MF players:\n")
```

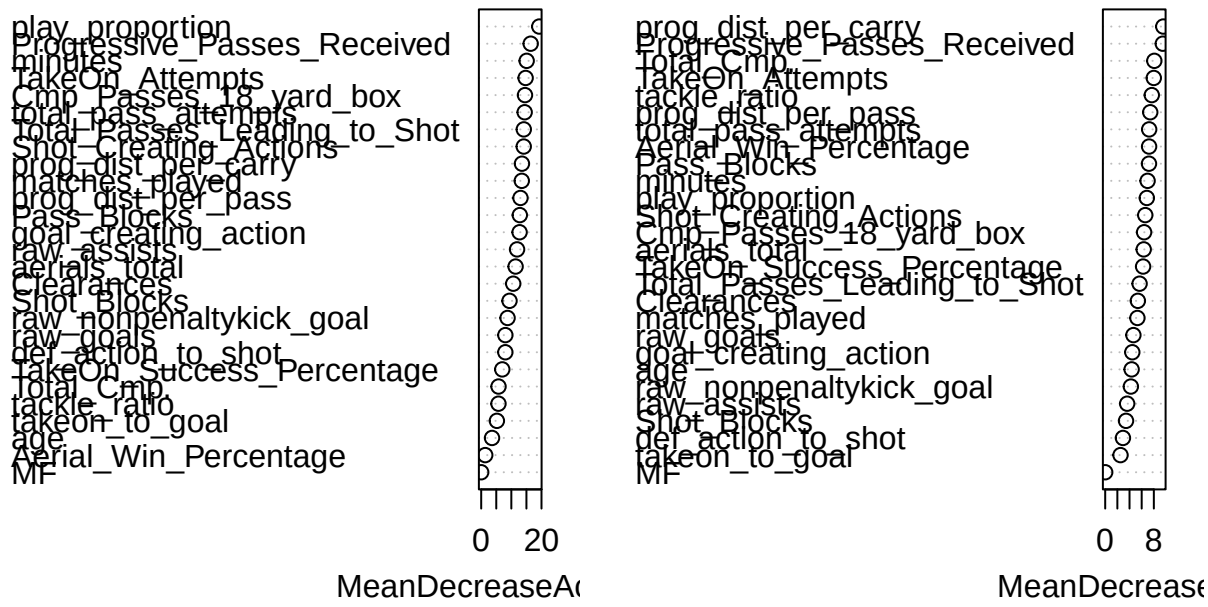
```
## Model summary for MF players:
```

```
print(models[["MF"]])
```

```
##
## Call:
##  randomForest(formula = transfer ~ ., data = trainData, ntree = 500,      mtry = floor(sqrt(ncol(tra
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 5
##
##         OOB estimate of  error rate: 5.51%
## Confusion matrix:
##      0 1 class.error
## 0 1351 1 0.000739645
## 1   78 5 0.939759036
```

```
par(mfrow = c(1, 2))  # Arrange plots side by side
plot(models[["MF"]], main = "OOB Error for MD Players")
varImpPlot(models[["MF"]], main = "Variable Importance for MD Players")
```


OOB Error for MD Players
Error
0.0
0.2
0.4
0.6
0.8
0
100
300
500
trees



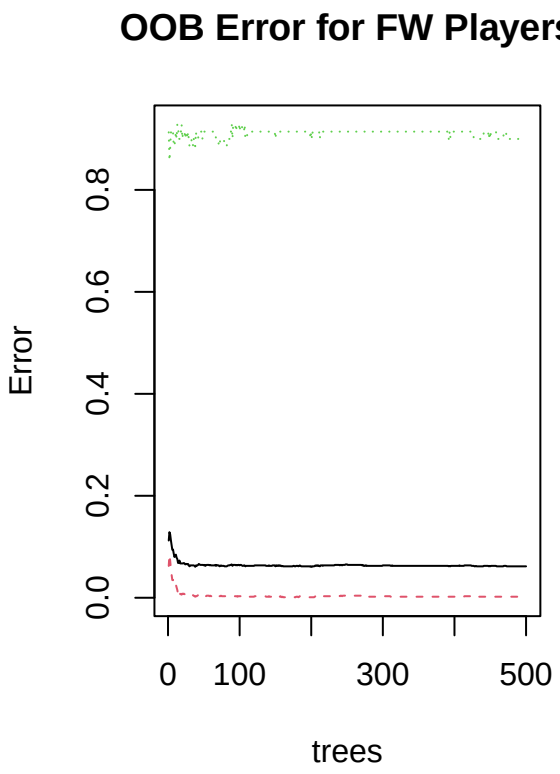

```r
par(mfrow = c(1, 1))  # Reset plotting layout

# For FW players:
cat("Model summary for FW players:\n")
```

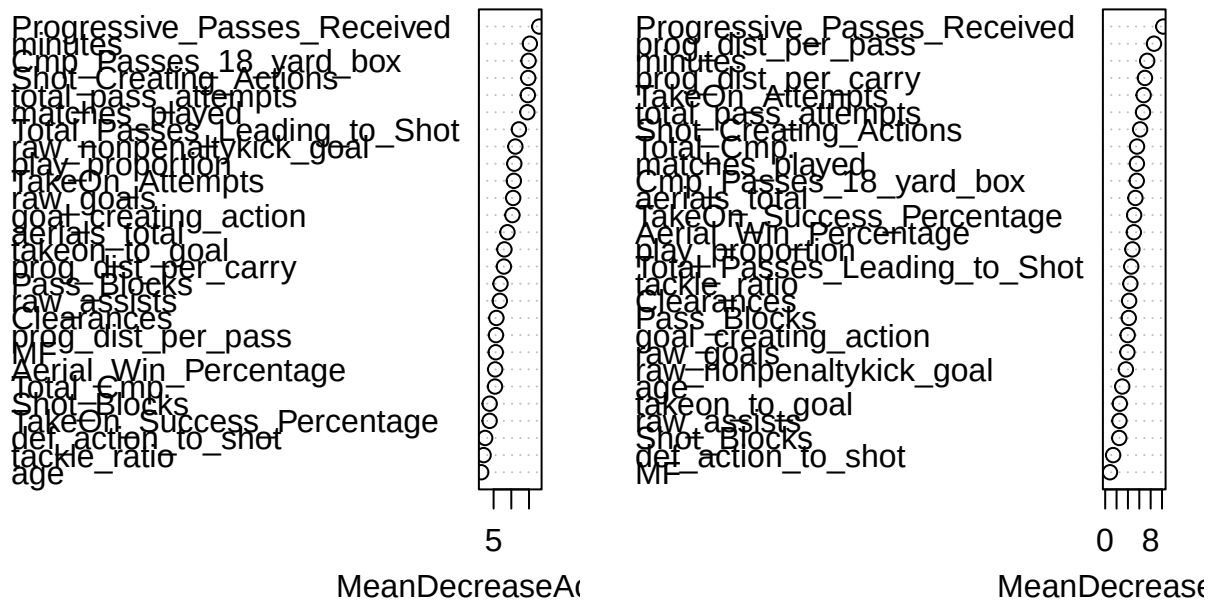
```
## Model summary for FW players:
```

```r
print(models[["FW"]])
```

```
##
## Call:
##  randomForest(formula = transfer ~ ., data = trainData, ntree = 500,      mtry = floor(sqrt(ncol(tra
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 5
##
##         OOB estimate of  error rate: 6.16%
## Confusion matrix:
##     0 1 class.error
## 0 983 2 0.002030457
## 1  63 7 0.900000000
```

```r
par(mfrow = c(1, 2))
plot(models[["FW"]], main = "OOB Error for FW Players")
varImpPlot(models[["FW"]], main = "Variable Importance for FW Players")
```


OOB Error for FW Players
Error
0.0
0.2
0.4
0.6
0.8
0
100
300
500
trees



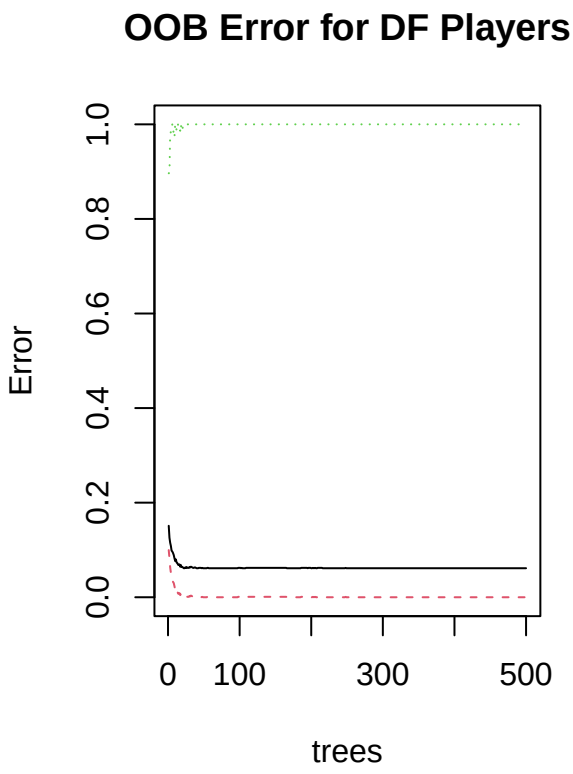

```
par(mfrow = c(1, 1))

# For DF players:
cat("Model summary for DF players:\n")
```

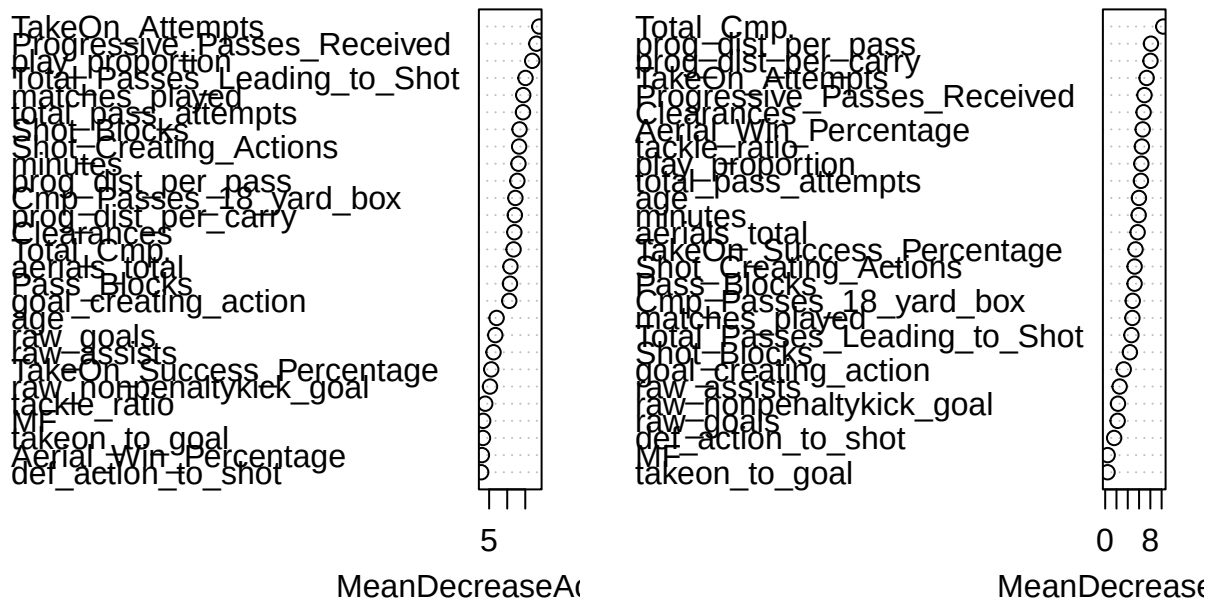
```
## Model summary for DF players:
```

```
print(models[["DF"]])
```

```
##
## Call:
##  randomForest(formula = transfer ~ ., data = trainData, ntree = 500,      mtry = floor(sqrt(ncol(tra
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 5
##
##         OOB estimate of  error rate: 6.13%
## Confusion matrix:
##      0 1 class.error
## 0 1133 0           0
## 1   74 0           1
```

```
par(mfrow = c(1, 2))
plot(models[["DF"]], main = "OOB Error for DF Players")
varImpPlot(models[["DF"]], main = "Variable Importance for DF Players")
```


OOB Error for DF Players
Error
0.0
0.2
0.4
0.6
0.8
1.0
0
100
300
500
trees




```r
par(mfrow = c(1, 1))
```

```r
library(knitr)

# For MD players:
md_conf <- as.data.frame(conf_matrices[["MD"]]$table)
kable(md_conf, caption = "Confusion Matrix for MD Players", align = "c")
```

Table: Confusion Matrix for MD Players

```r
# For FW players:
fw_conf <- as.data.frame(conf_matrices[["FW"]]$table)
kable(fw_conf, caption = "Confusion Matrix for FW Players", align = "c")
```

Table 1: Confusion Matrix for FW Players

Prediction	Reference	Freq
0	0	421
1	0	1
0	1	27
1	1	3

```r
# For DF players:
df_conf <- as.data.frame(conf_matrices[["DF"]]$table)
kable(df_conf, caption = "Confusion Matrix for DF Players", align = "c")
```

Table 2: Confusion Matrix for DF Players

Prediction	Reference	Freq
0	0	485
1	0	0
0	1	30
1	1	1