Logistic Regression

Caitlin Tuttle

```
-- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
v dplyr     1.1.4     v readr     2.1.5
v forcats   1.0.0     v stringr   1.5.1
v ggplot2   3.4.4     v tibble    3.2.1
v lubridate 1.9.3     v tidyr     1.3.0
v purrr     1.0.2
-- Conflicts ------------------------------------------ tidyverse_conflicts() --
x dplyr::filter() masks stats::filter()
x dplyr::lag()    masks stats::lag()
i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to becom
-- Attaching packages -------------------------------------- tidymodels 1.1.1 --

v broom        1.0.5     v rsample      1.2.0
v dials        1.2.0     v tune         1.1.2
v infer        1.0.6     v workflows    1.1.3
v modeldata    1.2.0     v workflowsets 1.0.1
v parsnip      1.1.1     v yardstick    1.2.0
v recipes      1.0.9

-- Conflicts ----------------------------------------- tidymodels_conflicts() --
x scales::discard() masks purrr::discard()
x dplyr::filter()   masks stats::filter()
x recipes::fixed()  masks stringr::fixed()
x dplyr::lag()      masks stats::lag()
x yardstick::spec() masks readr::spec()
x recipes::step()   masks stats::step()
* Dig deeper into tidy modeling with R at https://www.tmwr.org

Loading required package: lattice


Attaching package: 'caret'
```

```
The following objects are masked from 'package:yardstick':

    precision, recall, sensitivity, specificity


The following object is masked from 'package:purrr':

    lift


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


Attaching package: 'pROC'


The following objects are masked from 'package:stats':

    cov, smooth, var



Attaching package: 'kableExtra'


The following object is masked from 'package:dplyr':

    group_rows



Attaching package: 'gridExtra'


The following object is masked from 'package:dplyr':

    combine



Attaching package: 'stringdist'
```

```
The following object is masked from 'package:tidyr':

    extract


Loading required package: carData


Attaching package: 'car'


The following object is masked from 'package:dplyr':

    recode


The following object is masked from 'package:purrr':

    some
```

```r
setwd("~/Downloads")
eda <- read_csv("Wimbledon_featured_matches.csv", show_col_types = FALSE)
```

```r
eda$winner_shot_type[eda$p1_ace == 1] <- "Ace"
eda$winner_shot_type[eda$p2_ace == 1] <- "Ace"

eda$server[eda$server == 1] <- "Player 1"
eda$server[eda$server == 2] <- "Player 2"

eda$game_victor[eda$game_victor == 1] <- "Player 1"
eda$game_victor[eda$game_victor == 2] <- "Player 2"

eda$set_victor[eda$game_victor == 1] <- "Player 1"
eda$set_victor[eda$game_victor == 2] <- "Player 2"

eda$p1_net_pt[eda$p1_net_pt == 0] <- "Not at Net"
eda$p1_net_pt[eda$p1_net_pt == 1] <- "At Net"

eda$p2_net_pt[eda$p2_net_pt == 0] <- "Not at Net"
eda$p2_net_pt[eda$p2_net_pt == 1] <- "At Net"
```

```
eda$p1_net_pt_won[eda$p1_net_pt_won == 0] <- "Failed at net"
eda$p1_net_pt_won[eda$p1_net_pt_won == 1] <- "Scored At Net"

eda$p2_net_pt_won[eda$p2_net_pt_won == 0] <- "Failed at net"
eda$p2_net_pt_won[eda$p2_net_pt_won == 1] <- "Scored At Net"

eda$p1_unf_err[eda$p1_unf_err == 0] <- "No unforced Er"
eda$p1_unf_err[eda$p1_unf_err == 1] <- "Unforced Er"

eda$p2_unf_err[eda$p2_unf_err == 0] <- "No unforced Er"
eda$p2_unf_err[eda$p2_unf_err == 1] <- "Unforced Er"

eda$serve_no[eda$serve_no == 1] <- "1"
eda$serve_no[eda$serve_no == 2] <- "2"

eda$p1_sets[eda$p1_sets == 0] <- "0 won"
eda$p1_sets[eda$p1_sets == 1] <- "1 won"
eda$p1_sets[eda$p1_sets == 2] <- "2 won"

eda$p2_sets[eda$p2_sets == 0] <- "0 won"
eda$p2_sets[eda$p2_sets == 1] <- "1 won"
eda$p2_sets[eda$p2_sets == 2] <- "2 won"
```

```
eda <- eda %>%
  mutate(ace = ifelse(p1_ace > p2_ace, "ace", ifelse(p2_ace > p1_ace, "ace", 0))) %>%
  mutate(winning_shot = ifelse(p1_winner > p2_winner, "P1 untouchable winner", ifelse(p2_w
    mutate(break_shot = ifelse(p1_break_pt > p2_break_pt, "P1 break P2 serve", ifelse(p2_b
  mutate(break_winning_shot = ifelse(p1_break_pt_won > p2_break_pt_won, "P1 break win", if
  mutate(break_missed_shot = ifelse(p1_break_pt_missed > p2_break_pt_missed, "P1 break mis
  mutate(double_fault = ifelse(p1_double_fault > p2_double_fault, "player 1 fault", ifelse
  mutate(Performance_Difference = p1_points_won - p2_points_won)
eda
```

```
# A tibble: 7,284 x 53
   match_id player1 player2 elapsed_time set_no game_no point_no p1_sets p2_sets
   <chr>    <chr>   <chr>   <time>        <dbl>   <dbl>    <dbl> <chr>   <chr>
 1 2023-wi~ Carlos~ Nicola~ 00'00"            1       1        1 0 won   0 won
 2 2023-wi~ Carlos~ Nicola~ 00'38"            1       1        2 0 won   0 won
 3 2023-wi~ Carlos~ Nicola~ 01'01"            1       1        3 0 won   0 won
 4 2023-wi~ Carlos~ Nicola~ 01'31"            1       1        4 0 won   0 won
 5 2023-wi~ Carlos~ Nicola~ 02'21"            1       1        5 0 won   0 won
 6 2023-wi~ Carlos~ Nicola~ 02'50"            1       1        6 0 won   0 won
 7 2023-wi~ Carlos~ Nicola~ 03'33"            1       1        7 0 won   0 won
 8 2023-wi~ Carlos~ Nicola~ 04'01"            1       1        8 0 won   0 won
 9 2023-wi~ Carlos~ Nicola~ 04'48"            1       1        9 0 won   0 won
10 2023-wi~ Carlos~ Nicola~ 05'32"            1       1       10 0 won   0 won
# i 7,274 more rows
# i 44 more variables: p1_games <dbl>, p2_games <dbl>, p1_score <chr>,
#   p2_score <chr>, server <chr>, serve_no <chr>, point_victor <dbl>,
#   p1_points_won <dbl>, p2_points_won <dbl>, game_victor <chr>,
#   set_victor <chr>, p1_ace <dbl>, p2_ace <dbl>, p1_winner <dbl>,
#   p2_winner <dbl>, winner_shot_type <chr>, p1_double_fault <dbl>,
#   p2_double_fault <dbl>, p1_unf_err <chr>, p2_unf_err <chr>, ...
```

```r
eda_split     <- initial_split(eda, prop = .75)
eda_train_log <- training(eda_split)
eda_test_log  <- testing(eda_split)
```

```r
set.seed(1152)
eda_train_log$point_victor <- as.factor(eda_train_log$point_victor)

# Specify the logistic regression model
eda_fit_logistic1 <- glm(point_victor ~ server + lag(ace) + lag(speed_mph), data = eda_tra
                        family = binomial)

# Extract tidy output from the logistic regression model
eda_tidy_logistic1 <- tidy(eda_fit_logistic1)

# Display tidy output as a table
kable(eda_tidy_logistic1, digits = 3)
```

term	estimate	std.error	statistic	p.value
(Intercept)	-0.431	0.274	-1.572	0.116
serverPlayer 2	1.454	0.061	23.846	0.000
lag(ace)ace	0.277	0.107	2.584	0.010
lag(speed_mph)	-0.003	0.002	-1.262	0.207

```r
eda_train_log$point_victor <- as.factor(eda_train_log$point_victor)

# Specify the logistic regression model
eda_fit_logistic2 <- glm(point_victor ~ lag(point_victor) + server + lag(ace) +  lag(speed
      , data = eda_train_log,
                        family = binomial)

# Extract tidy output from the logistic regression model
eda_tidy_logistic2 <- tidy(eda_fit_logistic2)

# Display tidy output as a table
kable(eda_tidy_logistic2, digits = 3)
```

term	estimate	std.error	statistic	p.value
(Intercept)	-0.460	0.304	-1.515	0.130
lag(point_victor)2	0.079	0.062	1.279	0.201
serverPlayer 2	1.487	0.064	23.392	0.000
lag(ace)ace	0.288	0.115	2.514	0.012
lag(speed_mph)	-0.004	0.003	-1.525	0.127
lag(break_shot)P1 break P2 serve	0.010	0.161	0.064	0.949
lag(break_shot)P2 break P1 serve	0.182	0.185	0.988	0.323
lag(serve_width)BC	0.179	0.122	1.463	0.143
lag(serve_width)BW	0.097	0.117	0.829	0.407
lag(serve_width)C	0.114	0.114	0.993	0.321
lag(serve_width)W	0.104	0.114	0.908	0.364
p1_sets1 won	0.018	0.073	0.240	0.810
p1_sets2 won	-0.094	0.079	-1.177	0.239
p2_sets1 won	-0.050	0.069	-0.721	0.471
p2_sets2 won	-0.031	0.089	-0.349	0.727
p1_games	-0.042	0.026	-1.635	0.102
p2_games	0.056	0.026	2.154	0.031
serve_no2	-0.107	0.064	-1.668	0.095
lag(p1_distance_run)	0.006	0.006	0.947	0.343
lag(p2_distance_run)	-0.005	0.006	-0.784	0.433

```
vif_values <- car::vif(eda_fit_logistic2)
vif_values
```

```
                         GVIF Df GVIF^(1/(2*Df))
lag(point_victor)    1.023860  1        1.011860
server               1.079927  1        1.039195
lag(ace)             1.183717  1        1.087988
lag(speed_mph)       1.223775  1        1.106243
lag(break_shot)      1.031852  2        1.007870
lag(serve_width)     1.259180  4        1.029227
p1_sets              1.095409  2        1.023044
```

```
p2_sets              1.076369  2        1.018569
p1_games             2.398462  1        1.548697
p2_games             2.418323  1        1.555096
serve_no             1.005723  1        1.002857
lag(p1_distance_run) 6.804891  1        2.608619
lag(p2_distance_run) 6.813092  1        2.610190
```

```
# Define the format_estimate function
format_estimate <- function(x, p_value) {
  ifelse(p_value < 0.1, paste0(format(x, nsmall = 2), "**"), format(x, nsmall = 2))
}

# Assuming `eda_tidy` contains the relevant information
eda_tidy_logistic2 <- eda_tidy_logistic2 %>%
  mutate(estimate_formatted = format_estimate(estimate, p.value))

# Create a table with kable
eda_tidy_logistic2 %>%
  kable(caption = "Fitted Model (** if significant)") %>%
  kable_styling(
    full_width = FALSE,
    font_size = 8,
    position = "center",
    latex_options = c("striped", "hold_position")
  ) %>%
  column_spec(3, border_right = TRUE) %>%
  row_spec(0, bold = TRUE)
```

```
eda_train_log$point_victor <- as.factor(eda_train_log$point_victor)

# Specify the logistic regression model
eda_fit_logistic3 <- glm(point_victor ~ lag(point_victor) + server +  lag(speed_mph) + p1_
      , data = eda_train_log,
                        family = binomial)

# Extract tidy output from the logistic regression model
eda_tidy_logistic3 <- tidy(eda_fit_logistic3)

# Display tidy output as a table
kable(eda_tidy_logistic3, digits = 3)
```

Table 1: Fitted Model (** if significant)

term	estimate	std.error	statistic	p.value	estimate__formatted
(Intercept)	-0.4603904	0.3038209	-1.5153348	0.1296876	-0.460390433
lag(point__victor)2	0.0791400	0.0619000	1.2785129	0.2010687	0.079139981
serverPlayer 2	1.4872448	0.0635795	23.3918922	0.0000000	1.487244805**
lag(ace)ace	0.2882138	0.1146213	2.5144866	0.0119206	0.288213756**
lag(speed__mph)	-0.0040083	0.0026281	-1.5251557	0.1272203	-0.004008314
lag(break__shot)P1 break P2 serve	0.0103954	0.1614109	0.0644032	0.9486492	0.010395384
lag(break__shot)P2 break P1 serve	0.1824233	0.1846424	0.9879817	0.3231616	0.182423295
lag(serve__width)BC	0.1789053	0.1222796	1.4630837	0.1434445	0.178905336
lag(serve__width)BW	0.0972541	0.1172527	0.8294398	0.4068556	0.097254063
lag(serve__width)C	0.1135028	0.1142624	0.9933520	0.3205384	0.113502768
lag(serve__width)W	0.1035304	0.1140679	0.9076211	0.3640784	0.103530398
p1__sets1 won	0.0175427	0.0731220	0.2399100	0.8104001	0.017542696
p1__sets2 won	-0.0935139	0.0794773	-1.1766110	0.2393508	-0.093513902
p2__sets1 won	-0.0497474	0.0690454	-0.7205034	0.4712151	-0.049747420
p2__sets2 won	-0.0311195	0.0892500	-0.3486780	0.7273311	-0.031119500
p1__games	-0.0418155	0.0255704	-1.6353104	0.1019840	-0.041815517
p2__games	0.0563985	0.0261820	2.1540920	0.0312330	0.056398505**
serve__no2	-0.1068004	0.0640343	-1.6678621	0.0953431	-0.106800450**
lag(p1__distance__run)	0.0055504	0.0058593	0.9472809	0.3434956	0.005550392
lag(p2__distance__run)	-0.0045968	0.0058646	-0.7838175	0.4331472	-0.004596759

term	estimate	std.error	statistic	p.value
(Intercept)	-0.502	0.290	-1.731	0.084
lag(point__victor)2	0.076	0.061	1.235	0.217
serverPlayer 2	1.489	0.063	23.491	0.000
lag(speed__mph)	-0.002	0.002	-0.889	0.374
p1__sets1 won	0.013	0.073	0.184	0.854
p1__sets2 won	-0.097	0.079	-1.226	0.220
p2__sets1 won	-0.039	0.069	-0.572	0.568
p2__sets2 won	-0.015	0.089	-0.170	0.865
p1__games	-0.044	0.025	-1.740	0.082
p2__games	0.056	0.026	2.154	0.031
serve__no2	-0.105	0.064	-1.639	0.101
lag(p1__distance__run)	0.004	0.006	0.730	0.465
lag(p2__distance__run)	-0.005	0.006	-0.874	0.382

```
vif_values <- car::vif(eda_fit_logistic3)
vif_values
```

```
                         GVIF Df GVIF^(1/(2*Df))
lag(point_victor)    1.013398  1        1.006677
server               1.079050  1        1.038773
lag(speed_mph)       1.032873  1        1.016303
```

```
p1_sets             1.092515  2        1.022367
p2_sets             1.071344  2        1.017378
p1_games            2.397403  1        1.548355
p2_games            2.416564  1        1.554530
serve_no            1.003535  1        1.001766
lag(p1_distance_run) 6.717923  1        2.591896
lag(p2_distance_run) 6.739064  1        2.595971
```

1555

```
library(tidymodels)
library(pROC)
library(knitr)


# Assuming you have three different logistic regression models (eda_fit_logistic1, eda_fit
# Replace these with your actual logistic regression models
# Replace "edaflow_test_lin" with your actual test dataset

# Ensure that point_victor is treated as a factor (categorical) in the test dataset
eda_test_log$point_victor <- as.factor(eda_test_log$point_victor)

# Make predictions on the test dataset for each model
predictions1 <- predict(eda_fit_logistic1, newdata = eda_test_log, type = "response")
predictions2 <- predict(eda_fit_logistic2, newdata = eda_test_log, type = "response")
predictions3 <- predict(eda_fit_logistic3, newdata = eda_test_log, type = "response")

# Evaluate model performance
roc_curve1 <- roc(eda_test_log$point_victor, predictions1)
```

```
Setting levels: control = 1, case = 2
```

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

```
roc_auc1 <- auc(roc_curve1)
accuracy1 <- mean((predictions1 > 0.5) == as.integer(eda_test_log$point_victor))
rsquared1 <- summary(eda_fit_logistic1)$r.squared

roc_curve2 <- roc(eda_test_log$point_victor, predictions2)
```

```
Setting levels: control = 1, case = 2
Setting direction: controls < cases
```

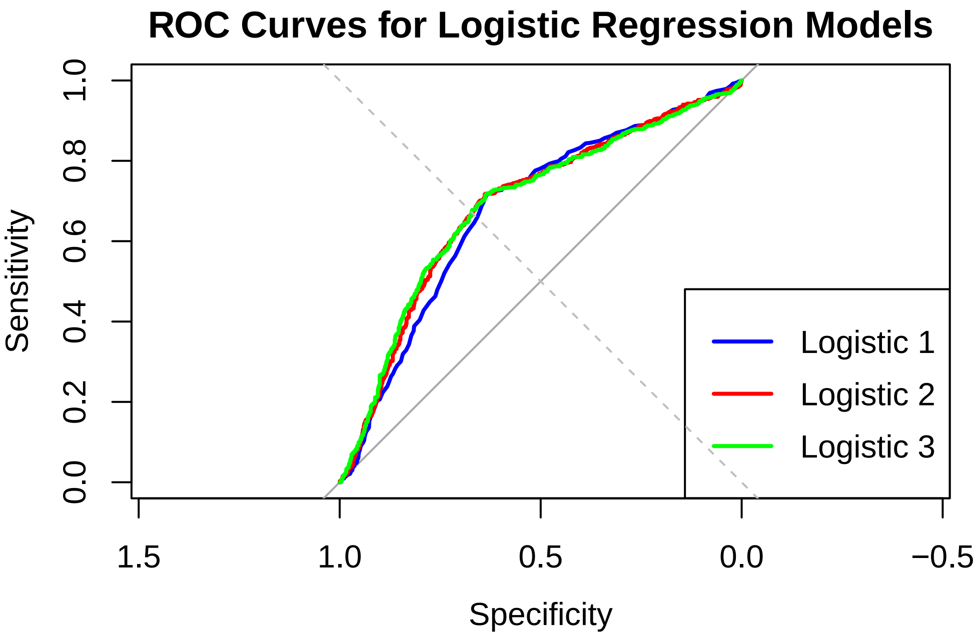
```r
roc_auc2 <- auc(roc_curve2)
accuracy2 <- mean((predictions2 > 0.5) == as.integer(eda_test_log$point_victor))
rsquared2 <- summary(eda_fit_logistic2)$r.squared

roc_curve3 <- roc(eda_test_log$point_victor, predictions3)
```

```
Setting levels: control = 1, case = 2
Setting direction: controls < cases
```

```r
roc_auc3 <- auc(roc_curve3)
accuracy3 <- mean((predictions3 > 0.5) == as.integer(eda_test_log$point_victor))
rsquared3 <- summary(eda_fit_logistic3)$r.squared

# Calculate AIC and BIC
aic1 <- AIC(eda_fit_logistic1)
aic2 <- AIC(eda_fit_logistic2)
aic3 <- AIC(eda_fit_logistic3)

bic1 <- BIC(eda_fit_logistic1)
bic2 <- BIC(eda_fit_logistic2)
bic3 <- BIC(eda_fit_logistic3)

# Create a summary table
summary_table <- data.frame(
  Model = c("eda_fit_logistic1", "eda_fit_logistic2", "eda_fit_logistic3"),
  AUC_ROC = c(roc_auc1, roc_auc2, roc_auc3),
  AIC = c(aic1, aic2, aic3),
  BIC = c(bic1, bic2, bic3)
)

# Display the summary table
kable(summary_table, digits = 3)
```

Model	AUC_ROC	AIC	BIC
eda_fit_logistic1	0.677	6198.761	6224.757
eda_fit_logistic2	0.688	6190.896	6320.799
eda_fit_logistic3	0.689	6209.238	6293.725

```r
plot(roc_curve1, col = "blue", main = "ROC Curves for Logistic Regression Models", lwd = 2
lines(roc_curve2, col = "red", lwd = 2)
lines(roc_curve3, col = "green", lwd = 2)
legend("bottomright", legend = c("Logistic 1", "Logistic 2", "Logistic 3"), col = c("blue"
abline(a = 0, b = 1, col = "gray", lty = 2)  # Diagonal line for reference
```


ROC Curves for Logistic Regression Models
Sensitivity
Specificity
Logistic 1
Logistic 2
Logistic 3