Data Analyst position at Raków Częstochowa - Task

Dr. Artur Czeszumski

June 26, 2024

Contents

W tym dokumencie przedstawiam odpowiedzi na pytania zwiazane z aplikacja na pozycje analityka danych w zespole Raków Częstochowa. Ten dokument zawiera kod, wizualizacje i odpowiedzi na pytania. Komentarze do kodu są w języku angielskim.

```
# Load required libraries
library(jsonlite)
library(tidyverse)
library(knitr)
library(ggsoccer)

# Read JSON files.
#Please remember to keep this file in the same folder as the data shared for the assignment.
d_match_stats <- stream_in(file('player_match_stats_match_3891414.json'))
d_events <- stream_in(file('events_match_3891414.json'))
d_lineup <- stream_in(file('lineups_match_3891414.json'))
```

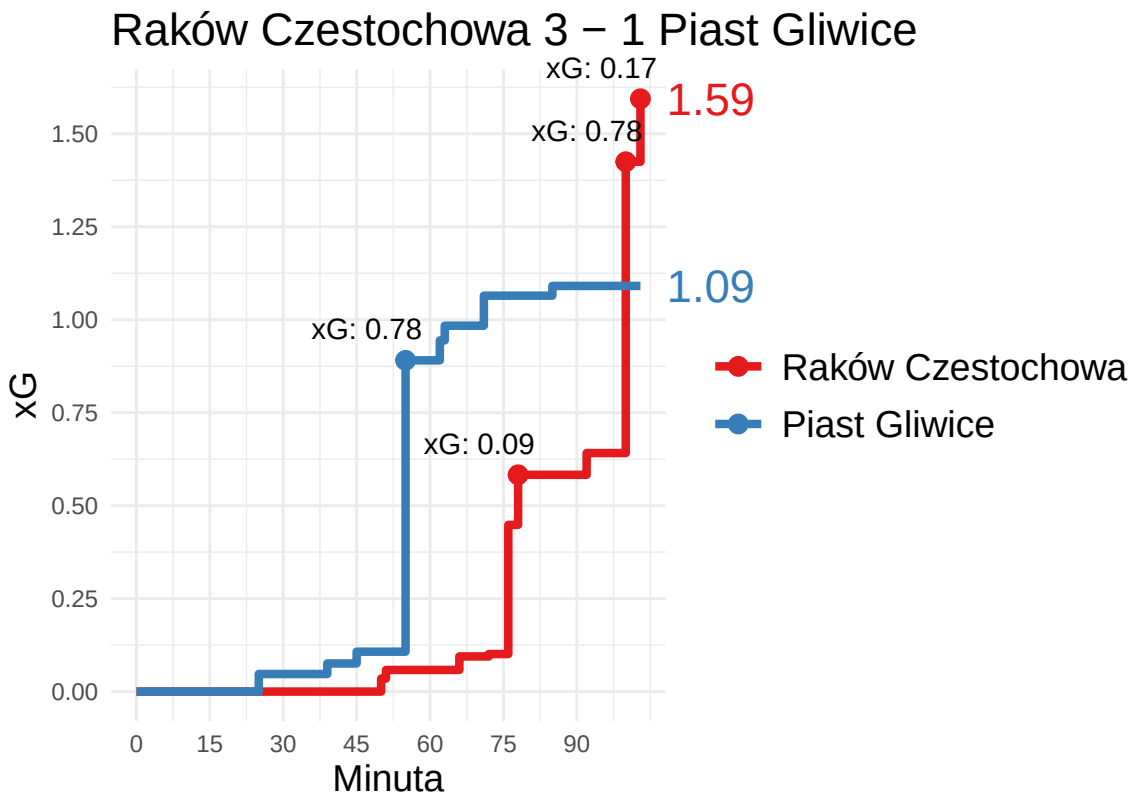
Pytanie 1:

Oblicz różnicę xG (expected goals) dla każdego stanu meczu (wygrana-przegrana-remis) dla obu drużyn.

```
# Filter events with xG values (shots)
shots_xg <- !is.na(d_events$shot$statsbomb_xg)

# Create a dataframe with xG values, outcomes, teams, and minutes of each shot
xG <- data.frame(d_events[shots_xg,]$shot$statsbomb_xg,
                 d_events[shots_xg,]$shot$outcome$name,
```

```
                d_events[shots_xg,]$possession_team$name,
                d_events[shots_xg,]$minute)
# change column names for more readable
colnames(xG) <- c('xG', 'Outcome', 'Team', 'Minute')
# create a variable that keeps both team names
teams <- unique(xG$Team)
# Add initial row with minute 0 and accumulated xG of 0 for each team
initial_rows <- data.frame(
  xG = 0,
  Outcome = NA,
  Team = teams,
  Minute = 0
)
# combine empty row with the data frame
xG <- rbind(initial_rows, xG)

# Convert Team to a factor and reverse the levels
xG$Team <- factor(xG$Team)
xG$Team <- factor(xG$Team, levels = rev(levels(xG$Team)))

# Calculate accumulated xG for each team over time
xG <- xG %>%
  arrange(Team, Minute) %>%
  group_by(Team) %>%
  mutate(accumulated_xG = cumsum(xG))

# Find the maximum minute in the match
max_minute <- max(xG$Minute)

# Add final row with maximum minute for each team
final_rows <- xG %>%
  group_by(Team) %>%
  summarize(Minute = max_minute, accumulated_xG = last(accumulated_xG), .groups = 'drop') %>%
  mutate(xG = 0, Outcome = NA)
xG <- rbind(xG, final_rows)

# Filter data for goals
goals <- xG %>%
  filter(Outcome == "Goal")

# Calculate the score for each team
score <- goals %>%
  group_by(Team) %>%
  summarise(Goals = n()) %>%
  ungroup()

# Ensure both teams are in the score data frame
score <- complete(score, Team, fill = list(Goals = 0))

# Create a label for the game score
game_score_label <- paste(score$Team[1], score$Goals[1], "-", score$Goals[2], score$Team[2])

# Plot accumulated xG over time for each team using step lines
```

```r
ggplot(xG, aes(x = Minute, y = accumulated_xG, color = Team, group = Team)) +
  geom_step(linewidth = 1.5) +  # Draw step lines to show the progression of accumulated xG over time
  geom_point(data = goals, aes(x = Minute, y = accumulated_xG, fill = Team), size = 3, shape = 21) +  #
  geom_text(data = goals, aes(x = Minute, y = accumulated_xG,
                              label = paste0("xG: ", sprintf("%.2f", xG))),
            vjust = -1, hjust = 0.85, size = 4, color = "black") +  # Annotate goals with their xG valu
  geom_text(data = final_rows, aes(x = Minute, y = accumulated_xG,
                                   label = sprintf("%.2f", accumulated_xG), color = Team),
            hjust = -0.3, size = 6, show.legend = FALSE) +  # Annotate the final accumulated xG values
  labs(title = game_score_label,  # Set the title of the plot
       x = "Minuta",  # Label for the x-axis
       y = "xG") +  # Label for the y-axis
  scale_y_continuous(breaks = seq(0, max(xG$accumulated_xG) * 1.2, by = 0.25)) +  # Set y-axis breaks e
  scale_x_continuous(breaks = seq(0, max(xG$Minute), by = 15)) +  # Set x-axis breaks every 15 minutes
  scale_color_brewer(palette = "Set1") +  # Apply a color palette from ColorBrewer to the lines and poi
  scale_fill_brewer(palette = "Set1") +   # Ensure the fill color for points matches the line color
  coord_cartesian(clip = 'off') +  # Ensure text annotations are not clipped by the plot area
  theme_minimal() +  # Use a minimal theme for a clean look
  theme(legend.title = element_blank(),  # Remove the legend title
        legend.position = "right",  # Position the legend on the right side of the plot
        legend.text = element_text(size = 14),  # Increase the legend text size
        legend.key.size = unit(1.5, "lines"),   # Increase the size of the legend keys
        title = element_text(size = 14),  # Set the title text size
        plot.margin = unit(c(1, 2, 1, 1), "lines"))  # Adjust the plot margins
```




```r
### Calculate the difference between xG for each score change
```

```r
# Function to calculate xG difference at the time of each goal
calculate_xG_difference <- function(goals, xG) {
  results <- data.frame()

  for (i in 1:nrow(goals)) {
    goal_event <- goals[i, ]
    goal_minute <- goal_event$Minute
    scoring_team <- goal_event$Team

    # Get the state of the game at the goal minute for both teams
    game_state <- xG %>%
      filter(Minute <= goal_minute) %>%
      group_by(Team) %>%
      summarise(accumulated_xG = last(accumulated_xG)) %>%
      ungroup()

    # Count the goals up to the goal minute for both teams
    goal_state <- goals %>%
      filter(Minute <= goal_minute) %>%
      group_by(Team) %>%
      summarise(goals_scored = n()) %>%
      ungroup()

    # Ensure both teams are included in the game state
    if (nrow(game_state) == 2) {
      team1 <- game_state$Team[1]
      team2 <- game_state$Team[2]
      xg_team1 <- game_state$accumulated_xG[1]
      xg_team2 <- game_state$accumulated_xG[2]

      goals_team1 <- ifelse(team1 %in% goal_state$Team, goal_state$goals_scored[goal_state$Team == team
      goals_team2 <- ifelse(team2 %in% goal_state$Team, goal_state$goals_scored[goal_state$Team == team

      difference <- abs(xg_team1 - xg_team2)

      # Determine the state of the game for each team based on goals
      if (goals_team1 == goals_team2) {
        state_team1 <- "tie"
        state_team2 <- "tie"
      } else {
        state_team1 <- if (goals_team1 > goals_team2) "win" else "lose"
        state_team2 <- if (goals_team2 > goals_team1) "win" else "lose"
      }

      result_row <- data.frame(
        Minute = goal_minute,
        Team1 = team1,
        Team2 = team2,
        accumulated_xG_Team1 = xg_team1,
        accumulated_xG_Team2 = xg_team2,
        goals_Team1 = goals_team1,
        goals_Team2 = goals_team2,
        xG_difference = difference,
```

```
        state_Team1 = state_team1,
        state_Team2 = state_team2
      )

      results <- rbind(results, result_row)
    }
  }

  # Order the results by minute
  results <- results %>%
    arrange(Minute)

  return(results)
}

# Calculate the xG differences for goals
goal_differences <- calculate_xG_difference(goals, xG)

# remove unnessary columns
goal_differences <- goal_differences %>%
  select(Team1, Team2, xG_difference, state_Team1, state_Team2)

# Print the results
kable(goal_differences)
```

Team1	Team2	xG_difference	state_Team1	state_Team2
Raków Częstochowa	Piast Gliwice	0.8326087	lose	win
Raków Częstochowa	Piast Gliwice	0.4817014	tie	tie
Raków Częstochowa	Piast Gliwice	0.3339243	win	lose
Raków Częstochowa	Piast Gliwice	0.5032797	win	lose

Odpowiedź:

W odpowiedzi na to pytanie przygotowałem wizualizację pokazującą zmiany xG dla obu drużyn po każdym oddanym strzale. Wszystkie gole są oznaczone z opisem wartości xG, a zaakumulowany xG dla obu drużyn jest przedstawiony. Dodatkowo policzyłem różnicę między zdobytym i straconym xG dla każdego stanu meczu (wygrana-przegrana-remis) i przedstawiłem wszystkie wartości w tabeli.

Raków przegrana: -0.83
Raków remis: -0.48
Raków wygrana: 0.33
Raków wygrana: 0.5

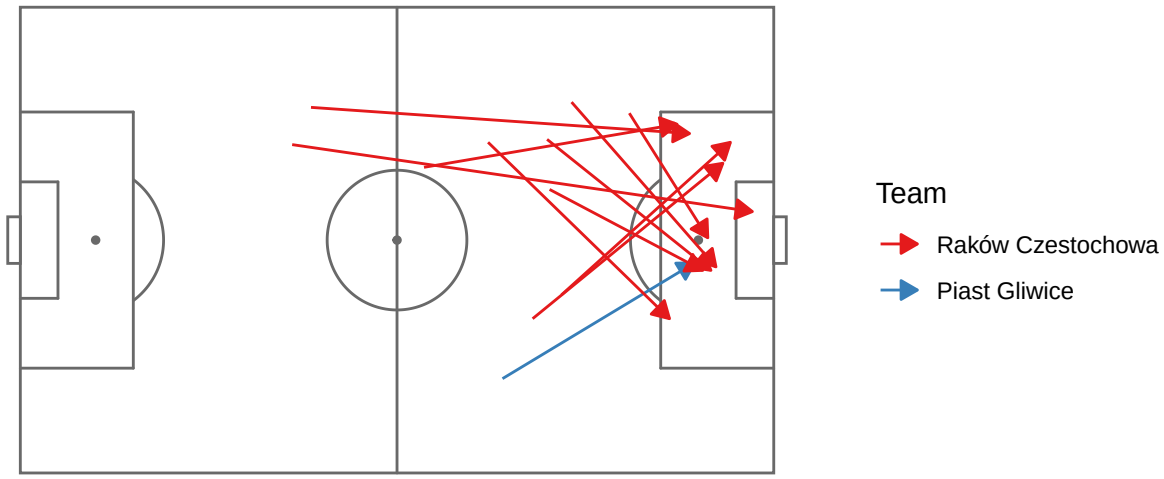
Piast wygrana: 0.83
Piast remis: 0.48
Piast przegrana: -0.33
Piast przegrana: -0.5

Pytanie 2:

Policz liczbę podań z półprzestrzeni do pola karnego dla obu drużyn.

```r
# define half spaces and penalty area
penalty_x <- 102
penalty_y_min <- 18
penalty_y_max <- 62

# divide y (width) by 5 spaces (left, left half-space, middle, right half-space, right)
left_half_spaces_y_min <- 16
left_half_spaces_y_max <- 32
right_half_spaces_y_min <- 48
right_half_spaces_y_max <- 64

# select data
passes <- data.frame(d_events[d_events$type$id == 30,]$possession_team$name,
                     map_dbl(d_events[d_events$type$id == 30,]$location, 1),
                     map_dbl(d_events[d_events$type$id == 30,]$location, 2),
                     map_dbl(d_events[d_events$type$id == 30,]$pass$end_location, 1),
                     map_dbl(d_events[d_events$type$id == 30,]$pass$end_location, 2),
                     d_events[d_events$type$id == 30,]$pass$outcome$name)
colnames(passes) <- c('Team', 'pass_start_x', 'pass_start_y', 'pass_end_x', 'pass_end_y', 'pass_outcome

# get passes from half spaces to the penalty box for each team
passes_half_spaces <- passes %>%
  filter(pass_start_x < penalty_x,
    (pass_start_y > left_half_spaces_y_min & pass_start_y < left_half_spaces_y_max) |
           (pass_start_y > right_half_spaces_y_min & pass_start_y < right_half_spaces_y_max),
         pass_end_x > penalty_x & pass_end_x < 120,
         pass_end_y > penalty_y_min & pass_end_y < penalty_y_max)

teams <- unique(passes$Team)

# get the number of passes for each team
#nrow(passes_half_spaces[passes_half_spaces$Team == teams[1],])
#nrow(passes_half_spaces[passes_half_spaces$Team == teams[2],])

# Convert Team to a factor and reverse the levels
passes_half_spaces$Team <- factor(passes_half_spaces$Team)
passes_half_spaces$Team <- factor(passes_half_spaces$Team, levels = rev(levels(passes_half_spaces$Team))

# make a pitch to plot passes
ggplot(passes_half_spaces) +
  annotate_pitch(dimensions = pitch_statsbomb) +
  geom_segment(aes(x = pass_start_x, y = pass_start_y, xend = pass_end_x, yend = pass_end_y, colour = Te
               arrow = arrow(length = unit(0.25, "cm"),
                             type = "closed")) +
  scale_color_brewer(palette = "Set1") +
  theme_pitch() +
  labs(title = 'Podania z półprzestrzeni do pola karnego')
```

Podania z półprzestrzeni do pola karnego



Odpowiedź:

Aby zdefiniować półprzestrzenie, podzieliłem szerokość boiska na pięć części i wybrałem dwie pomiędzy prawą a środkową oraz lewą a środkową. Co więcej, półprzestrzenie kończą się na wysokości linii pola karnego. W tym meczu Raków miał 10 podań z półprzestrzeni do pola karnego, a Piast miał tylko jedno. Wszystkie podania są pokazane na grafice.

Pytanie 3:

Jaki jest wzrost najniższego zawodnika, który oddał strzał głową po stałym fragmencie gry?

```
# get the data for head shots and type of play patterns
shots <- data.frame(d_events[which(d_events$shot$body_part$id == 37), ]$play_pattern$id,
                    d_events[which(d_events$shot$body_part$id == 37), ]$player$name)
# change column names
colnames(shots) <- c('Play_Pattern', 'Name')

# select only headers from set pieces
shots <- shots %>%
  filter(Play_Pattern %in% c(2,3,4,7,9))

# organize lineups for both teams
lineups1 <- data.frame(d_lineup$lineup[1]) %>%
  select(player_name, player_height)
lineups2 <- data.frame(d_lineup$lineup[2])  %>%
  select(player_name, player_height)
lineups <- rbind(lineups1, lineups2)

# combine data frames with headers and lineups
result <- shots %>%
  inner_join(lineups, by = c("Name" = "player_name")) %>%
  arrange(player_height) # arrange it from smallest

# print names of players with lowest height
print(result[result$player_height == min(result$player_height),]$Name)
```

Odpowiedź:

W tym meczu było dwóch piłkarzy, którzy oddali strzały głową i są tego samego wzrostu (najniżsi z wszystkich, którzy oddali strzały głową po stałym fragmencie gry):

- `Jakub Czerwiński - 183`
- `Efstratios Svarnas - 183`

Pytanie 4:

Jak zautomatyzowałbyś raport z odpowiedziami na powyższe pytania? (nie musisz tego robić, wystarczy opisanie procesu i narzędzi)

Odpowiedź:

Kod napisany w odpowiedziach pozwala odpowiedzieć na te pytania w każdym innym meczu, który mógłby być wyciągnięty z StatsBomb API. To znaczy, ten dokument HTML albo PDF może być generowany automatycznie i dostarczany członkom sztabu oraz każdej innej zainteresowanej osobie. To znaczy kod by się komunikował z StatsBomb API, tworzył grafiki i odpowiedzi, które by były przedstawione w PDF albo HTML. Jak w tym przykładzie, użyłbym R i R markdown do tworzenia tych dokumentów. Co więcej, aby zautomatyzować proces i umożliwić interakcję z danymi, z łatwością mógłbym stworzyć małą aplikację (R Shiny), która komunikowałaby się z StatsBomb API i generowała raporty na każdy wybrany mecz przez użytkownika. Jako przykład chciałbym pokazać aplikację, którą stworzyłem, aby porównać style gry między zespołami w różnych sezonach: https://arturczeszumski.shinyapps.io/playstyleapp/

Pytanie 5:

Jakie trzy metryki zespołowe (z tych oferowanych przez Statsbomb) uznałbyś za niezbędne w raporcie pomeczowym? Krótko opisz dlaczego?

Odpowiedź:

1. Expected Goals (xG): Ta metryka ocenia jakość oddanych strzałów przez drużynę, biorąc pod uwagę różne czynniki, takie jak odległość do bramki, kąt strzału oraz rodzaj akcji prowadzącej do strzału. xG pozwala lepiej zrozumieć, czy drużyna tworzy wartościowe sytuacje strzeleckie, co jest kluczowe dla oceny efektywności ofensywy.
2. Pressures: Metryka ta mierzy liczbę sytuacji, w których zawodnicy drużyny wywierają presję na przeciwników posiadających piłkę. Wysoki poziom presji może świadczyć o agresywnej grze defensywnej i skuteczności w odbiorze piłki. Jest to ważne dla oceny, jak dobrze drużyna radzi sobie w obronie oraz jak efektywnie odzyskuje piłkę.
3. On-Ball Value (OBV): Metryka ta ocenia wartość, jaką każde zagranie piłki (podanie, drybling, strzał itp.) dodaje do potencjalnej zdolności drużyny do zdobycia bramki lub zapobieżenia stracie bramki. OBV jest zaawansowaną metryką analityczną, która pozwala na bardziej szczegółową ocenę wkładu poszczególnych zawodników w grę zespołu. Dzięki OBV można zidentyfikować, które zagrania mają największy wpływ na przebieg meczu, zarówno w ataku, jak i w obronie.

Dodatkowo uważam, że metryka PPDA jest bardzo ważna i można ją łatwo wyliczyć z danych StatsBomb.

PPDA (Passes Per Defensive Action): PPDA mierzy liczbę podań przeciwnika, jakie drużyna pozwala wykonać, zanim podejmie akcję defensywną (np. odbiór, przechwyt, faul). Niższa wartość PPDA oznacza bardziej intensywną presję. Ta metryka jest kluczowa dla oceny, jak agresywnie drużyna naciska na przeciwnika, co może być wskaźnikiem jej defensywnej strategii i skuteczności w odbiorze piłki. PPDA oblicza się, dzieląc liczbę podań przeciwnika przez liczbę akcji defensywnych drużyny w określonej strefie boiska.