

# Betting on Upcoming Matches

Ing. Peter Tomko, M.A.

7/9/2020

```
select distinct created_at, dateclosed, fullname, sport_league,  
trim(BOTH from t_home_team) as t_home_team,  
trim(BOTH from t_away_team) as t_away_team,  
rate from v_upcoming_matches;
```

```
select distinct tipsport_league, results_league, results_team, string_agg(tipsport_team, '') as tipsport_team  
(select distinct created_at, tipsport_league, results_league, results_team, tipsport_team, last_update  
from  
(select created_at,  
trim(both tipsport_league) as tipsport_league,  
trim(both results_league) as results_league,  
trim(both results_team) as results_team,  
trim(both tipsport_team) as tipsport_team,  
max(created_at) over(partition by tipsport_league, results_league, results_team, tipsport_team) as last_update  
from t_mapping_team) as m_data  
where created_at = last_update) actual_mapping  
group by tipsport_league, results_league, results_team;
```

```
select distinct * from v_match_stats;
```

```
d_upcoming_matches <- v_upcoming_matches %>%  
  as.data.frame() %>%  
  
  # - create match ID  
  group_by(created_at, fullname, sport_league, t_home_team, t_away_team) %>%  
  mutate(tip_match_id = c(1:n())) %>%  
  as.data.frame() %>%  
  
  gather(., is_home, tipsport_team,  
    -created_at, -dateclosed, -fullname,  
    -sport_league, -rate, -tip_match_id) %>%  
  mutate(is_home = ifelse(is_home == "t_home_team", 1, 0)) %>%  
  rename(tipsport_league = sport_league) %>%  
  
  # - get mapped team names  
  left_join(., t_mapping_team, by = c("tipsport_league" = "tipsport_league",  
    "tipsport_team" = "tipsport_team")) %>%  
  
  # - check how many time team was mapped  
  group_by(created_at, fullname, tipsport_league,  
    results_league, is_home, tipsport_team) %>%  
  mutate(t_mapped = n()) %>%  
  as.data.frame() %>%
```

```

# - assign not in data
mutate(map_missing = ifelse(is.na(results_team), 1, 0)) %>%
as.data.frame() %>%
distinct() %>%

# - get last match
left_join(., v_match_stats %>%
  select(team, created_at) %>%
  distinct() %>%
  left_join(., t_mapping_team, by = c("team" = "results_team")) %>%
  group_by(team, tipsport_team) %>%
  summarise(last_match = max(created_at)),
  by = c("results_team" = "team",
        "tipsport_team" = "tipsport_team")) %>%
as.data.frame() %>%

# - keep only latest naming convention (i.e. one match one mapping to HomeTeam and AwayTeam)
group_by(created_at, dateclosed, fullname, tip_match_id,
  is_home, tipsport_team, results_league) %>%
mutate(l_n_matchdate = max(last_match)) %>%
mutate(name_matches = ifelse(l_n_matchdate == last_match, 1, 0)) %>%
as.data.frame() %>%
filter(name_matches == 1) %>%
select(-t_mapped, -map_missing, -last_match, -l_n_matchdate, -name_matches) %>%

# - nesting in order to join historical statistics
group_by(dateclosed, is_home, results_league, results_team, tip_match_id) %>%
nest() %>%
rename(tipsport_data = data) %>%

# - transform dateclosed to match date
mutate(match_date = as.Date(dateclosed))

d_upcoming_matches <- d_upcoming_matches %>%
mutate(row_id = row_number()) %>%
group_by(row_id) %>%
mutate(hist_data =
  pmap(list(match_date, results_team, is_home),
    function(i_md, i_rt, i_h) {

      # - set inputs
      output_temp <- list()
      count <- 1
      last_n_v <- c(10, 20, 30, 40, 50)

      for(i in 1:length(last_n_v)){

        # - subset the data from which the history is taken
        temp_data <-
          v_match_stats %>%
          dplyr::filter(created_at < i_md &
            is_home == i_h &
            results_team %in% i_rt) %>%

```

```

as.data.frame() %>%
  arrange(desc(created_at)) %>%
  as.data.frame()

if(nrow(temp_data) < last_n_v[i]){
  temp_data <- temp_data[1:nrow(temp_data), ]
}

else{
  temp_data <- temp_data[1:last_n_v[i], ]
}

temp_data$hist_category <- paste("last_",
                                last_n_v[i], sep = "")

output_temp[[count]] <-
  temp_data %>%
  as.data.frame() %>%
  dplyr::select(-season, -league, -created_at,
               -team, -is_home, -match_id) %>%
  as.data.frame() %>%
  dplyr::group_by(hist_category) %>%
  dplyr::summarise(team = i_rt,
                  is_home = i_h,
                  league = temp_data$league[1],
                  season = current_season,
                  match_results = mean(match_results),
                  avg_total_goals = mean(total_goals),
                  n_goals = mean(n_goals),
                  n_shots = mean(n_shots),
                  n_shots_ontarget = mean(n_shots_ontarget),
                  n_fauls = mean(n_fauls),
                  n_corners = mean(n_corners),
                  n_yellow_cards = mean(n_yellow_cards),
                  n_red_cards = mean(n_red_cards),
                  r_shots_goals = mean(r_shots_goals),
                  r_st_goals = mean(r_st_goals),
                  r_fauls_goals = mean(r_fauls_goals),
                  r_corners_goals = mean(r_corners_goals),
                  r_yellow_goals = mean(r_yellow_goals),
                  r_red_goals = mean(r_red_goals),
                  r_team_odds = mean(r_team_odds),
                  r_draw_odds = mean(r_draw_odds),
                  r_ah_advantage =
                    mean(o_strength_ah)/(mean(o_strength) + 1),
                  r_ah_advantage_season =
                    mean(o_strength_ah)/(mean(o_strength_season) + 1),
                  r_season_strength =
                    mean(o_strength_season)/(mean(o_strength) + 1),
                  r_season_strength_ah =
                    mean(o_strength_season_ah)/(mean(o_strength_ah) + 1)) %>%
  as.data.frame()

```

```

        count <- count + 1
      }

      temp_return <-
        bind_rows(output_temp) %>%
        as.data.frame() %>%
        gather(., var_name, est_value,
              -hist_category, -league, -team, -season, -is_home) %>%
        as.data.frame() %>%
        mutate(n_var_name = paste(var_name, "__",
                                   hist_category, sep = "")) %>%
        select(n_var_name, est_value, league, team, season, is_home) %>%
        spread(n_var_name, est_value)
      return(temp_return)
    })
  })

```

```
load("C:/Users/Peter/Desktop/ds_projects/betting_data_science/6 glm models/1 model development/prelimin
```

```

rm(master_data)
rm(binning_output)
rm(binning_vars)
rm(no_binning_vars)
rm(output_path)
rm(test_season)

```

```

d_upcoming_matches <- d_upcoming_matches %>%
  group_by(row_id) %>%
  mutate(binned_data = map(hist_data, function(i_df){

    test_df <- i_df %>%
      as.data.frame() %>%
      mutate_if(is.character, as.factor)

    woe.binning.deploy(test_df,
                      binning = binning_model,
                      min.iv.total = 0.015,
                      add.woe.or.dum.var = "woe") %>%
    select(is_home, season, contains("woe.")) %>%
    rename_all(~stringr::str_replace_all(., "__", "_")) %>%
    rename_all(~stringr::str_replace_all(., "woe.", "")) %>%
    rename_all(~stringr::str_replace_all(., ".binned", "")) %>%
    as.data.frame() %>%
    mutate(is_home = ifelse(is_home == 1, "yes", "no")) %>%

    as.data.frame()
  })
})

```

```
load("C:/Users/Peter/Desktop/ds_projects/betting_data_science/6 glm models/1 model development/prelimin
```

```

d_upcoming_matches <- d_upcoming_matches %>%
  group_by(row_id) %>%
  mutate(p_lambda = map(binned_data, function(i_df){

    data.frame("Lambda" = c(predict(glm_m_final,

```

```

                                newdata = i_df,
                                type = "response"),
predict(hurdle_model,
        newdata = i_df,
        type = "response")),
      "Model" = c("glm", "hurdle"))
})) %>%
unnest(c(p_lambda))

bet_data <- d_upcoming_matches %>%
  select(-hist_data, -binned_data) %>%
  unnest(c(tipsport_data)) %>%

  # - get lambdas
as.data.frame() %>%
select(-row_id, -match_date, -results_league, -results_team) %>%
mutate(is_home = ifelse(is_home == 1, "HomeTeam", "AwayTeam")) %>%
as.data.frame() %>%
distinct() %>%
as.data.frame() %>%

  # - get HomeTeam and AwayTeam
group_by(dateclosed, tip_match_id, created_at, tipsport_league,
          fullname, Model, rate) %>%
nest() %>%

group_by(dateclosed, tip_match_id, created_at, tipsport_league,
          fullname, Model, rate) %>%
mutate(t_team = map(data, function(i_df) data.frame(t_rows = nrow(i_df)))) %>%
unnest(c(t_team)) %>%
filter(t_rows == 2) %>%
select(-t_rows) %>%
filter(str_detect(fullname, "5") == T) %>%

  # - spread table
distinct() %>%
as.data.frame() %>%
rowwise() %>%
mutate(t_b_goals = as.numeric(str_extract(fullname, "\\d+\\.\\d*")),
       t_interval = ifelse(str_detect(fullname, "Více") == T, "Over", "Under"))

get_pred_func <-
function(i_data, i_rate, i_t_b_goals, i_interval){
  # i_data <- bet_data$data[[1]]
  # i_t_b_goals <- 2.5
  # i_interval <- "Under"

  temp <-
  expand_grid("HomeTeam" = i_data$tipsport_team[i_data$is_home %in% "HomeTeam"],
             "AwayTeam" = i_data$tipsport_team[i_data$is_home %in% "AwayTeam"],
             "HTG" = c(0:10),
             "ATG" = c(0:10),
             "HTL" = i_data$Lambda[i_data$is_home %in% "HomeTeam"],
             "ATL" = i_data$Lambda[i_data$is_home %in% "AwayTeam"]) %>%

```

```

    filter(HTG + ATG < i_t_b_goals) %>%
    mutate(prob = dpois(HTG, HTL) * dpois(ATG, ATL)) %>%
    group_by(HomeTeam, AwayTeam) %>%
    summarise(p_over = 1 - sum(prob),
              p_under = sum(prob))

    if(i_interval %in% "Under"){
      temp <- temp %>% select(-p_over) %>% rename(p_prob = p_under)
    }else{
      temp <- temp %>% select(-p_under) %>% rename(p_prob = p_over)
    }

    temp <- temp %>%
      mutate(kelly_criterion = (i_rate * p_prob - 1)/i_rate)

    return(temp)
  }

output_ls <- list()
for(i in 1:nrow(bet_data)){
  output_ls[[i]] <- get_pred_func(i_data = bet_data$data[[i]],
                                  i_rate = bet_data$rate[i],
                                  i_t_b_goals = bet_data$t_b_goals[i],
                                  i_interval = bet_data$t_interval[i])
}

temp_df <- output_ls %>%
  bind_rows() %>%
  cbind(., bet_data %>%
    select(-data, -t_b_goals, -t_interval, -tip_match_id)) %>%
  mutate(opt_stake = kelly_criterion * kelly_fraction * bet_stake,
         analysis_time = Sys.time())

names(temp_df) <- tolower(names(temp_df))
if(nrow(temp_df) > 0) {
  dbWriteTable(con, "t_optimal_bets", temp_df,
              row.names = F, append = T)}

```