1 Model Development

Variable Selection using "`woeBinning`"

Ing. Peter Tomko, M.A.

29/8/2020

Note:

In this step, the preliminary data are no longer stored within database, but three local objects are created in the folder `1 model development/preliminary_data`, namely `1a variable selection - binning.RData`, `1b elastic net modelling.RData` and `1c bayesian multilevel model.RData`. If the folder is not within the folder structure, please create it manually.

Introduction

Previous steps, described in the other documents, described the process of how we will get final data set suitable for the modelling stage. In this document, I will describe, how the final variables are prepared and how they enter the model.

The modelling, in general consists of two steps

- binning algorithm is applied on the data in order to select variables with highest `IV`, i.e. Information Value(please see `1a variable selection - binning.Rmd`)
- after elimination of variables, `glmnet` model is estimated, cross validation is applied in order to get best hyper parameters (variables are reduced in this step as well) (please see `1b elastic net modelling.Rmd`)
- cca. 7 - 10 variables are then selected for the final Bayesian Multilevel Model using `brms` package (`1c bayesian multilevel model.R`)

Downloading Views

```
con <- DBI::dbConnect(odbc::odbc(), "betting_ds", bigint = "integer")
```

```
select * from v_last_matches;
```

```
select * from v_match_stats;
```

Target Variable Preparation

In this case, two types of target variable might be used - number of goals (`n_goals`) or match result (`match_results`, i.e. 1 for `win`, 0 for `loss` or -1 for `Draw`), depending on your preferences.

```
select distinct season, league, team, match_id, created_at, n_goals
from v_match_stats;
```

Preparation of Modelling Data

This step prepares final data set that might be used for modeling stage. For the purpose of this exercise (primarily due to the lack of suitable hardware), only subset of all available leagues is provided, namely `E0` (Premier League), `SP1` (La Liga), `D1` (Bundesliga), and `F1` (Ligue 1).

```
modelling_data <-
  v_last_matches %>%

  # - removing too historical seasons
  filter(!(season %in% c("9495", "9596", "9697", "9798", "9899", "9900",
                         "0001", "0102", "0203", "0304", "0405", "0506",
                         "0607", "0708", "0809", "0910"))) %>%
  filter(hist_category %in% c("last_10", "last_20", "last_30",
                              "last_40", "last_50")) %>%

  # - get data
  group_by(league, team, is_home) %>%
  nest() %>%
  rename(h_data = data) %>%

  left_join(., v_match_stats) %>%
  group_by(league, team, is_home) %>%
  nest(data = !c(h_data, league, team, is_home)) %>%
  rename(data = h_data,
         match_data = data) %>%

  group_by(team, is_home) %>%
  mutate(hist_data =
           map2(data, match_data,
                function(h_data, m_data) {

                  temp_df <-
                    h_data %>%
                    mutate(days_diff = created_at - last_n) %>%
                    map_df(., rep, .$days_diff) %>%
                    group_by(created_at, last_n) %>%
                    mutate(between_date = last_n + c(1:n()) - 1) %>%
                    as.data.frame() %>%

                    left_join(., m_data %>%
                                rename(match_date = created_at),
                              by = c("between_date" = "match_date",
                                     "season" = "season")) %>%
                    filter(!(is.na(match_id))) %>%

                    group_by(season, created_at, last_n, hist_category) %>%
                    summarise(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()

                  return(temp_df)
                })) %>%
  select(-data, - match_data) %>%
  unnest(c(hist_data)) %>%
  as.data.frame() %>%

  gather(., var_name, var_value,
         -c(league, season, team, is_home,
            created_at, last_n, hist_category)) %>%
  mutate(var_name_n = paste(var_name, hist_category, sep = "__")) %>%

  select(-hist_category, -var_name, -last_n) %>%
  distinct() %>%
  as.data.frame() %>%
  spread(var_name_n, var_value) %>%
  mutate_if(is.character, as.factor) %>%
  as.data.frame()
```

Data Split by season

- if 2020/2021 season already started, it is not used in any modeling procedures
- season 2019/2020 used for applying the betting strategy
- season 2018/2019 for testing the predictive performance of model
- other seasons used for training the predictive model

```
master_data <- modelling_data %>%
  left_join(., target_vars %>%
              mutate(n_goals_cat = ifelse(n_goals > 2.5,
                                          "Over 2.5", "Under 2.5")) %>%
              mutate_if(is.character, as.factor)) %>%

  # - split season
  mutate(data_type = ifelse(season %in% test_season, "Val", "Train")) %>%
```

```r
  # - nest
  group_by(data_type) %>%
  nest()

DBI::dbDisconnect(con)
rm(con)
rm(target_vars)
rm(modelling_data)
gc()
```

```
##            used  (Mb) gc trigger   (Mb)  max used   (Mb)
## Ncells  2443671 130.6    3847000  205.5   3847000  205.5
## Vcells 67663905 516.3  346484964 2643.5 433012797 3303.7
```

Variable Selection using woeBinning

```r
no_binning_vars <-
  c("is_home", "created_at", "match_id", "n_goals", "n_goals_cat")
binning_vars <- names(master_data$data[[1]] %>%
                        select(-one_of(no_binning_vars)))

binning_model <-
  woe.binning(df = master_data %>%
                filter(data_type %in% "Train") %>%
                unnest(c(data)) %>%
                as.data.frame() %>%
                select(-data_type) %>%
                select(one_of(binning_vars), n_goals_cat),
              target.var = "n_goals_cat",
              pred.var = binning_vars)

binning_output <-
  map_df(woe.binning.table(binning_model),
         ~as.data.frame(.x), .id = "variable") %>%
  mutate(variable = str_replace_all(variable, "WOE Table for ", "")) %>%
  as.data.frame() %>%
  group_by(variable) %>%
  mutate(total_iv = sum(as.numeric(IV))) %>%
  arrange(desc(total_iv))

# ----- Apply Binning -----
master_data <- master_data %>%
  group_by(data_type) %>%
  mutate(binned_data =
           map(data, function(i_data){

             woe.binning.deploy(i_data %>% as.data.frame(),
                                binning = binning_model,
                                min.iv.total = min_iv,
                                add.woe.or.dum.var = "woe") %>%
               select(one_of(no_binning_vars), n_goals_cat, contains("woe"))

           }))
```

```r
# ---- cleaning the output -----
rm(v_match_stats)
rm(v_last_matches)
rm(min_iv)

save.image(output_path)
```