Original NCAA Basketball Prediction Model

```r
# This code is an older version of the current model that I use for projecting NCAA
# basketball games on my website/twitter. I wrote an article that touches on elements of
# this code at collindougherty.com titled "How I Used Machine Learning to Solve March
# Madness".

# Very lightly edited, code below is more for educational purposes than anything. I plan to
# upload my current working model at some point prior to the 2022-23 NCAA season
```

```r
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --
```

```
## v ggplot2 3.3.6     v purrr   0.3.4
## v tibble  3.1.7     v dplyr   1.0.9
## v tidyr   1.2.0     v stringr 1.4.0
## v readr   2.1.2     v forcats 0.5.1
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```r
library(tidymodels)
```

```
## -- Attaching packages -------------------------------------- tidymodels 0.2.0 --
```

```
## v broom        0.8.0     v rsample      0.1.1
## v dials        0.1.1     v tune         0.2.0
## v infer        1.0.2     v workflows    0.2.6
## v modeldata    0.1.1     v workflowsets 0.2.1
## v parsnip      0.2.1     v yardstick    1.0.0
## v recipes      0.2.0
```

```
## -- 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()
## * Learn how to get started at https://www.tidymodels.org/start/
```

```r
library(zoo)
```

```
##
## Attaching package: 'zoo'
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```
library(hoopR)
library(corrr)
```

```
data <- load_mbb_team_box(seasons = 2022:2022) %>%
  separate(field_goals_made_field_goals_attempted, into = c("field_goals_made","field_goals_attempted")
  separate(three_point_field_goals_made_three_point_field_goals_attempted, into = c("three_point_field_g
  separate(free_throws_made_free_throws_attempted, into = c("free_throws_made","free_throws_attempted")
  mutate_at(12:34, as.numeric) %>%
  rename(team = team_short_display_name,
         fgm = field_goals_made,
         threes = three_point_field_goals_made,
         ftm = free_throws_made,
         fta = free_throws_attempted,
         fga = field_goals_attempted,
         threes_attempted = three_point_field_goals_attempted,
         fg_pct = field_goal_pct
         ) %>%
  arrange(game_date)
```

```
# important to group by in order to properly calculate rolling_means
teamdata <- data %>%
  group_by(team, season) %>%
  # creating some new stats that we will use
  mutate(
    team_score = ((fgm - threes) * 2) + (threes*3) + ftm,
    possessions = fga - offensive_rebounds + turnovers + (.475 * fta),
    ppp = team_score/possessions,
    turnover_pct = turnovers/(fga + 0.44 * fta + turnovers),
    free_throw_factor = ftm/fga,
    true_shooting = (team_score / (2*(fga + (.44 * fta)))) * 100,
    efg = (fgm + .5*threes)/fga,
    three_pct = threes/threes_attempted,
    three_pct = as.numeric(three_pct),
    # a binary variable to identify whether teams are home or away
    dummy_home_away = ifelse(home_away == "HOME", 1, 0),

    #rolling means, lag prevents model from knowing current results, align right is needed because roll
    r_three_pct = lag(cummean(three_pct), n = 1),
    r_fg_pct = lag(cummean(fg_pct), n=1),
    r_ppp = lag(cummean(ppp), n=1),
    r_true_shoot = lag(cummean(true_shooting), n=1),
    r_turnover_pct = lag(cummean(turnover_pct), n=1),
    r_ftf = lag(cummean(free_throw_factor), n=1),
    r_efg = lag(cummean(efg), n=1)
    ) %>%
        mutate(team_id = as.numeric(team_id),
               game_id = as.numeric(game_id))

excluded_vars <- c("team_uid", "team_alternate_color", "team_color", "team_display_name", "team_name",
teamdata <- teamdata %>% select(-excluded_vars) %>% ungroup()
```

```
## Note: Using an external vector in selections is ambiguous.
## i Use `all_of(excluded_vars)` instead of `excluded_vars` to silence this message.
## i See <https://tidyselect.r-lib.org/reference/faq-external-vector.html>.
## This message is displayed once per session.
```

```
# nu1 <- teamdata %>% filter(team == "Nebraska")
# view(nu1)
```

```
opponentdata <- teamdata %>% #select(-excluded_vars) %>%
    rename(opponent = opponent_name,
          opp_id=opponent_id,
          opp_offensive_rebounds = offensive_rebounds,
          opp_defensive_rebounds=defensive_rebounds,
    opp_score = team_score,
    opp_possessions = possessions,
    opp_ppp = ppp,
    opp_true_shooting = true_shooting,
    opp_turnover_pct = turnover_pct,
    opp_fgm = fgm,
    opp_fga = fga,
    opp_fg_pct = fg_pct,
    opp_threes = threes,
    opp_threes_attempted = threes_attempted,
    opp_three_pct = three_pct,
    opp_ftm = ftm,
    opp_fta = fta,
    opp_free_throw_pct = free_throw_pct,
    opp_free_throw_factor = free_throw_factor,
    opp_total_rebounds = total_rebounds,
    opp_offensive_rebounds = offensive_rebounds,
    opp_defensive_rebounds = defensive_rebounds,
    opp_assists = assists,
    opp_steals = steals,
    opp_blocks = blocks,
    opp_turnovers = turnovers,
    opp_fouls = fouls,
    opp_three_point_field_goal_pct = three_point_field_goal_pct,
    opp_team_rebounds = team_rebounds,
    opp_team_turnovers = team_turnovers,
    opp_total_turnovers = total_turnovers,
    opp_total_technical_fouls = total_technical_fouls,
    opp_technical_fouls =technical_fouls,
    opp_flagrant_fouls = flagrant_fouls,
    opp_home_away = home_away,
    opp_efg = efg,
    opp_dummy_home_away = dummy_home_away,

    opp_r_three_pct = r_three_pct,
    opp_r_true_shoot = r_true_shoot,
    opp_r_ppp = r_ppp,
    opp_r_fg_pct = r_fg_pct,
    opp_r_ftf = r_ftf,
    opp_r_turnover_pct = r_turnover_pct,
    opp_r_efg = r_efg,
    # opp_largestLead = largestLead
    ) %>%
        mutate(opp_id = as.numeric(opp_id),
               game_id = as.numeric(game_id))


excluded_vars <- c("opponent", "opp_id", "largest_lead")
opponentdata <- opponentdata %>% select(-excluded_vars)
opponentdata <- opponentdata %>% rename(opponent_name = team,
                                        opponent_id = team_id)

# nu2 <- opponentdata %>% filter(opponent_id == 158)
# view(nu2)
```

```
# this is where we fix team name and opponent_name merging issues, they often differ in df data
teamdata <- teamdata %>% mutate(opponent_name = ifelse(opponent_name == "South Dakota State", "S Dakota
teamdata <- teamdata %>% mutate(opponent_name = ifelse(opponent_name == "Jacksonville State", "J'Ville S
teamdata <- teamdata %>% mutate(opponent_name = ifelse(opponent_name == "New Mexico State", "New Mexico


bothdata <- teamdata %>% inner_join(opponentdata)
```

```
## Joining, by = c("opponent_id", "opponent_name", "game_id", "season",
## "season_type", "game_date")
```

```
# nu3 <- bothdata %>% filter(team == "Nebraska")
# view(nu3)
```

```
bothdata <- bothdata %>% group_by(team, season) %>% mutate(
    orb_pct = offensive_rebounds / (offensive_rebounds + opp_defensive_rebounds),
    drb_pct = defensive_rebounds / (opp_offensive_rebounds + defensive_rebounds),
    r_orb_pct = lag(cummean(orb_pct), n=1),
    r_drb_pct = lag(cummean(drb_pct), n=1),
    team_scoring_margin = team_score - opp_score,
    r_score_margin = lag(cummean(team_scoring_margin), n=1),
    r_def_ppg = lag(cummean(opp_score), n=1),
    r_def_ppp = lag(cummean(opp_ppp), n=1),
    r_def_three_pct = lag(cummean(opp_three_pct), n=1),
    r_def_fg_pct = lag(cummean(opp_fg_pct), n=1),
    r_def_true_shoot = lag(cummean(opp_true_shooting), n=1),
    r_def_ftf = lag(cummean(opp_free_throw_factor), n=1),
    r_def_efg = lag(cummean(opp_efg), n=1),
    r_def_turnover_pct = lag(cummean(opp_turnover_pct), n=1),
    possession_diff = possessions - opp_possessions

    ) %>% ungroup() %>% na.omit()

# nu4 <- bothdata %>% filter(team == "Nebraska")
# view(nu4)
# osu4 <- bothdata %>% filter(team == "Ohio State")
# view(osu4)
```

```
opponent_rebound_data <- bothdata %>% select(game_id, team_id, game_date, season, orb_pct, drb_pct, r_o
  rename(opponent_id = team_id,
         opp_orb_pct = orb_pct,
         opp_drb_pct = drb_pct,
         opp_r_orb_pct = r_orb_pct,
         opp_r_drb_pct = r_drb_pct,
         opp_r_score_margin = r_score_margin,
    opp_r_def_ppg = r_def_ppg,
    opp_r_def_ppp = r_def_ppp,
    opp_r_def_three_pct = r_def_three_pct,
    opp_r_def_fg_pct = r_def_fg_pct,
    opp_r_def_true_shoot =r_def_true_shoot,
    opp_r_def_ftf = r_def_ftf,
    opp_r_def_efg = r_def_efg,
    opp_r_def_turnover_pct = r_def_turnover_pct,
    opp_possession_diff = possession_diff,
    opp_team_scoring_margin = team_scoring_margin
         )
bothdatawithrebounds <- bothdata %>% inner_join(opponent_rebound_data)# %>% na.omit()
```

```
## Joining, by = c("opponent_id", "game_id", "season", "game_date")
```

```
# nu5 <- bothdatawithrebounds %>% filter(team == "Nebraska")
# view(nu5)
```

```
# sums the nas in every column
# na_count <- sapply(msu, function(y) sum(length(which(is.na(y)))))
# na_count <- data.frame(na_count)
#na <- data %>% filter(is.na(team))
```

```
teamdatawithreboundsanddefense <- bothdatawithrebounds %>% group_by(team, season) %>% mutate(
  r_d_ppp = lag(cummean(opp_ppp), n=1),
  score_vs_avg = team_scoring_margin + opp_r_score_margin,
  r_score_vs_avg = lag(cummean(score_vs_avg), n=1),
  r_possession_diff = lag(cummean(possession_diff), n=1)
) %>% ungroup() #%>% na.omit()

# nu6 <- teamdatawithreboundsanddefense %>% filter(team == "Nebraska")
# view(nu6)
```

```
opponent_defense_data <- teamdatawithreboundsanddefense %>% select(game_id, team_id, game_date, season,
  rename(opponent_id = team_id,
         opp_r_d_ppp = r_d_ppp,
         opp_score_vs_avg = score_vs_avg,
         opp_r_score_vs_avg = r_score_vs_avg,
         opp_r_possession_diff = r_possession_diff)
bothdatawithreboundsanddefense <- teamdatawithreboundsanddefense %>% inner_join(opponent_defense_data)
```

```
## Joining, by = c("opponent_id", "game_id", "season", "game_date")
```

```r
# nu7 <- bothdatawithreboundsanddefense %>% filter(team == "Nebraska")
# view(nu7)
# osu7 <- bothdatawithreboundsanddefense %>% filter(team == "Ohio State")
# view(osu7)
```

```r
games <- bothdatawithreboundsanddefense

games <- games %>% mutate(score_margin = team_score - opp_score,
  TeamResult = as.factor(case_when(
    team_score > opp_score ~ "W",
    opp_score > team_score ~ "L"
)))# %>% na.omit()
```

```r
modelgames <- games %>% select(game_id, game_date, team, opponent_name, season, dummy_home_away,

                               r_true_shoot, opp_r_true_shoot,
                               r_turnover_pct, opp_r_turnover_pct,
                               r_ftf, opp_r_ftf,
                               r_orb_pct, opp_r_orb_pct,
                               r_fg_pct, opp_r_fg_pct,
                               r_three_pct, opp_r_three_pct,
                               r_ppp, opp_r_ppp,

                               r_d_ppp, opp_r_d_ppp,

                               r_score_vs_avg, opp_r_score_vs_avg,

                               r_def_ppg, opp_r_def_ppg,
                               r_def_ppp, opp_r_def_ppp,
                               r_def_three_pct, opp_r_def_three_pct,
                               r_def_fg_pct, opp_r_def_fg_pct,
                               r_def_true_shoot, opp_r_def_true_shoot,
                               r_def_ftf, opp_r_def_ftf,
                               r_def_efg, opp_r_def_efg,
                               r_def_turnover_pct, opp_r_def_turnover_pct,
                               r_possession_diff, opp_r_possession_diff,
                               #r_blocks, opp_r_blocks,

                               #rolling_sos, opponent_rolling_sos,
                               score_margin,
                               TeamResult) %>% na.omit()

averages <- summarize_all(modelgames, mean)
```

```
## Warning in mean.default(team): argument is not numeric or logical: returning NA
```

```
## Warning in mean.default(opponent_name): argument is not numeric or logical:
## returning NA
```

```
## Warning in mean.default(TeamResult): argument is not numeric or logical:
## returning NA
```

```
modelgames <- rbind(modelgames, averages)
modelgames <- modelgames %>% mutate(game_id = ifelse(is.na(team), 9999999999999, game_id),
                                    team = ifelse(is.na(team), "average_team", team),
                                    opponent_name = ifelse(is.na(opponent_name), "average_team", oppone
```

```
game_split <- initial_split(modelgames, prop = .8)
game_train <- training(game_split)
game_test <- testing(game_split)

game_recipe <-
  recipe(TeamResult ~ ., data = game_train) %>%
  update_role(game_id, game_date, team, opponent_name, season, score_margin,
              new_role = "ID") %>%
  step_normalize(all_predictors())

summary(game_recipe)
```

```
## # A tibble: 44 x 4
##    variable           type    role      source
##    <chr>              <chr>   <chr>     <chr>
##  1 game_id            numeric ID        original
##  2 game_date          date    ID        original
##  3 team               nominal ID        original
##  4 opponent_name      nominal ID        original
##  5 season             numeric ID        original
##  6 dummy_home_away    numeric predictor original
##  7 r_true_shoot       numeric predictor original
##  8 opp_r_true_shoot   numeric predictor original
##  9 r_turnover_pct     numeric predictor original
## 10 opp_r_turnover_pct numeric predictor original
## # ... with 34 more rows
```

```
log_mod <-
  logistic_reg() %>%
  set_engine("glm") %>%
  set_mode("classification")

rf_mod <-
  rand_forest() %>%
  set_engine("ranger") %>%
  set_mode("classification")
```

```
# svm_mod <-
#   svm_poly() %>%
#   set_engine("kernlab") %>%
#   set_mode("classification")
#
# xg_mod <-   boost_tree(
#   trees = tune(),
#   learn_rate = tune(),
#   tree_depth = tune(),
#   min_n = tune(),
#   loss_reduction = tune(),
#   sample_size = tune(),
#   mtry = tune(),
#   ) %>%
#   set_mode("classification") %>%
#   set_engine("xgboost")
```

```
log_workflow <-
  workflow() %>%
  add_model(log_mod) %>%
  add_recipe(game_recipe)

rf_workflow <-
  workflow() %>%
  add_model(rf_mod) %>%
  add_recipe(game_recipe)
```

```
# svm_workflow <-
#   workflow() %>%
#   add_model(svm_mod) %>%
#   add_recipe(game_recipe)
#
# xg_wflow <-
#   workflow() %>%
#   add_model(xg_mod) %>%
#   add_recipe(game_recipe)
```

```
# lm_mod <-
#   linear_reg() %>%
#   set_engine("lm")
#
# lm_workflow <-
#   workflow() %>%
#   add_model(lm_mod) %>%
#   add_recipe(game_recipe)
#
# lm_fit <-
#   lm_workflow %>%
#   fit(data = game_train)
```

```
# lmpredict <- lm_fit %>% predict(new_data = game_train) %>%
#   bind_cols(game_train)
```

```
# metrics(lmpredict, score_margin, estimate = .pred)
```

```
# lmtestpredict <- lm_fit %>% predict(new_data = game_test) %>%
#   bind_cols(game_test)
#
# metrics(lmtestpredict, score_margin, estimate = .pred)
```

```r
log_fit <-
  log_workflow %>%
  fit(data = game_train)
```

```r
# rf_fit <-
#   rf_workflow %>%
#   fit(data = game_train)
```

```r
# svm_fit <-
#   svm_workflow %>%
#   fit(data = game_train)
#
# xgb_grid <- grid_latin_hypercube(
#   trees(),
#   tree_depth(),
#   min_n(),
#   loss_reduction(),
#   sample_size = sample_prop(),
#   finalize(mtry(), game_train),
#   learn_rate(),
#   size = 30
# )
# game_folds <- vfold_cv(game_train)
# xgb_res <- tune_grid(
#   xg_wflow,
#   resamples = game_folds,
#   grid = xgb_grid,
#   control = control_grid(save_pred = TRUE)
# )
# best_acc <- select_best(xgb_res, "accuracy")
# final_xgb <- finalize_workflow(
#   xg_wflow,
#   best_acc
# )
#
# xg_fit <-
#   final_xgb %>%
#   fit(data = game_train)
```

```r
logpredict <- log_fit %>% predict(new_data = game_train) %>%
  bind_cols(game_train)

logpredict <- log_fit %>% predict(new_data = game_train, type="prob") %>%
  bind_cols(logpredict)
```

```r
# rfpredict <- rf_fit %>% predict(new_data = game_train) %>%
#   bind_cols(game_train)
#
# rfpredict <- rf_fit %>% predict(new_data = game_train, type="prob") %>%
#   bind_cols(rfpredict)

# svmpredict <- svm_fit %>% predict(new_data = game_train) %>%
#   bind_cols(game_train)
#
# svmpredict <- svm_fit %>% predict(new_data = game_train, type="prob") %>%
#   bind_cols(svmpredict)
#
# xgpredict <- game_train %>%
#   bind_cols(predict(xg_fit, game_train))
```

```r
metrics(logpredict, TeamResult, .pred_class)
```

```
## # A tibble: 2 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy binary         0.700
## 2 kap      binary         0.401
```

```r
# metrics(rfpredict, TeamResult, .pred_class)
```

```r
# metrics(svmpredict, TeamResult, .pred_class)
```

```r
# metrics(xgpredict, truth = TeamResult, estimate = .pred_class)
```

```r
logtestpredict <- log_fit %>% predict(new_data = game_test) %>%
  bind_cols(game_test)

logtestpredict <- log_fit %>% predict(new_data = game_test, type="prob") %>%
  bind_cols(logtestpredict)

metrics(logtestpredict, TeamResult, .pred_class)
```

```
## # A tibble: 2 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy binary         0.688
## 2 kap      binary         0.376
```

```r
# rftestpredict <- rf_fit %>% predict(new_data = game_test) %>%
#   bind_cols(game_test)
#
# rftestpredict <- rf_fit %>% predict(new_data = game_test, type="prob") %>%
#   bind_cols(rftestpredict)
#
# metrics(rftestpredict, TeamResult, .pred_class)
```

```r
# svmtestpredict <- svm_fit %>% predict(new_data = game_test) %>%
#   bind_cols(game_test)
#
# svmtestpredict <- svm_fit %>% predict(new_data = game_test, type="prob") %>%
#   bind_cols(svmtestpredict)
#
# metrics(svmtestpredict, TeamResult, .pred_class)
# xgtestresults <- game_test %>%
#   bind_cols(predict(xg_fit, game_test))
#
# metrics(xgtestresults, truth = TeamResult, estimate = .pred_class)
```

```
logtestpredict %>%
  conf_mat(TeamResult, .pred_class)
```

```
##            Truth
## Prediction   L   W
##          L 473 212
##          W 228 497
```

```
# rftestpredict %>%
#   conf_mat(TeamResult, .pred_class)
```

```
# svmtestpredict %>%
#   conf_mat(TeamResult, .pred_class)
```