Predicting March Madness Selections

Caleb Skinner

May, 5, 2024

Abstract

Each year, a 10 member committee selects and seeds 68 universities to compete in the college basketball postseason tournament. Using seven years of data, I employ ensemble boosting trees to predict the committee's decisions. The model is highly accurate in predicting the 68 teams and moderately accurate in ranking the teams. A university's Strength of Record is the best predictor for their selection and ranking. In further analysis, I find that the model's selections perform slightly better than the committee's selections in the tournament, but more work is needed to validate these results.

Data Summary

Every year in March, 68 universities are selected to participate in the March Madness College Basketball Tournament. The teams are selected and ranked by a ten member committee. These rankings come with pervasive media attention and have a significant financial and marketing impact for the universities and their athletic programs. For these reasons, the committee's selections come under intense scrutiny and criticism. Bitter accusations of bias or malpractice from fans or university staff are commonplace and becoming more and more frequent. This problem is uniquely positioned in a time with growing technological advancements in the field of automated decision making. Computational tools and predictive metrics already equip human decision makers, and some have called for these decisions to become entirely automatic.

The march madness selection committee discloses several of the predictive metrics and summary statistics they use. In this paper, I seek to predict the committee's selections by training a model on their previous decisions with these metrics. I then analyze the committee's deviation from the model's predictions.

The predictors (see Table A1) can generally be divided into two categories: metrics designed to predict future success and summary statistics gauged to measure the team's resume. The committee intentionally developed the NCAA Evaluation Tool (NET) in 2019 to measure the school's resume and replace RPI as the primary resume tool. All summary statistics and predictive tools are ranks. I have seven years of data from 2017 to 2024 (the tournament was not held in 2020).

Techniques Employed

I conduct a two-step predictive process using Ensemble Boosting. In the first step, a classification boosting tree is training to classify the universities as selected or unselected by the committee. For each year, the 68 universities with the highest selection probability are predicted to make the tournament. In the second step, a classification regression tree is trained to rank the 68 selected universities. In this case, I treat the discrete outcome variable *seed* as continuous.

The selections are rankings within each year are correlated, so for each step, I use six years of data to train the model on the seventh year. I rotate this through each year to achieve a predicted value for every year. I standardize the predictors and use cross validation to tune the boosting algorithm with 1500 trees. The tuning method typically selects a tree depth of 5.

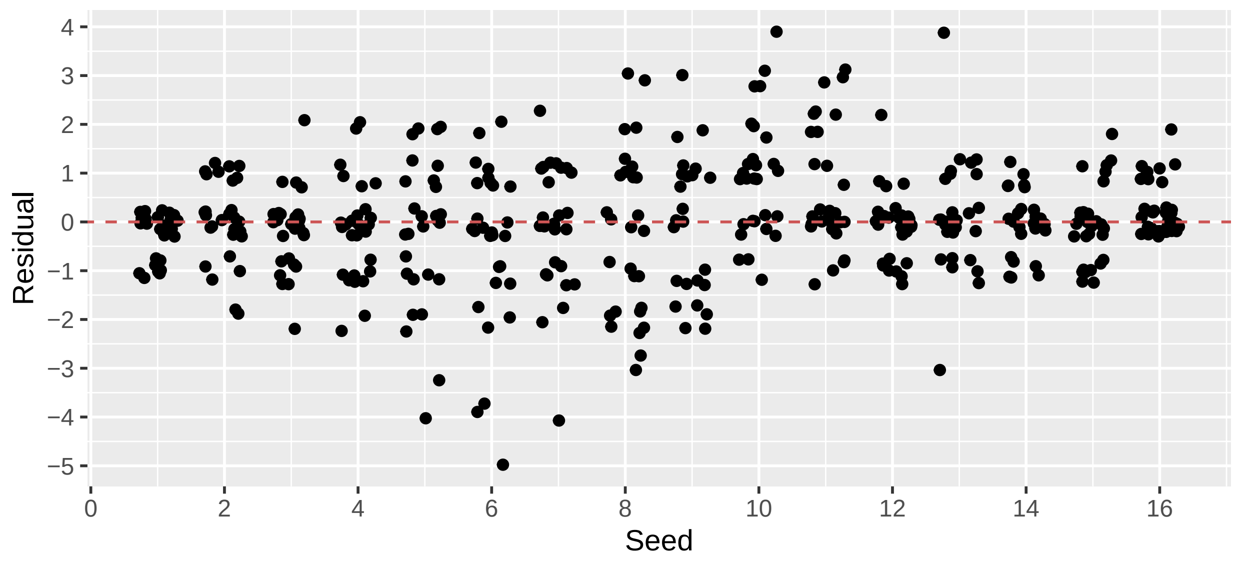
Results and Interpretation

Results

Overall, the first stage boosting algorithm has a high prediction accuracy. In seven years, the algorithm select 458 of the 476 teams (~96.2%). This amounts to about 2-3 misses each year (see Table A2 for yearly accuracy). For comparison, the top humans predicting the committee's selections tend to misclassify one or two universities each year.

The second stage boosting algorithm is less accurate. The model assigns each selected team a ranking between 1 and 16, and the Mean Squared Error of the model's rank and the committee's rank is 1.34 (see Table A3 for yearly errors). The model predicted the seed correctly for 211 of the 476 teams (~44.3%). The model is much more accurate at identifying the committee's highest and lowest ranked teams. The margins separating the mid-ranked teams are much closer, and the model has difficulty predicting how the committee will order them. I visualize the model's prediction error by seed in Figure 1. Noise is added to the values to show the density of the groups.

Figure 1: Stage 2 Error by Seed



I also visualize the boosting algorithm's variable importance measurement for stage 1 and stage 2 in Figure 2. The most important predictor both steps is clearly SOR. This is interesting, because in recent years, the committee has claimed to rely most heavily on the NET rankings. SOR is a very similar statistic to the NET and likely captures similar trends among the universities. The Conference Finalist and Conference Lost variables are important in stage 1, because universities automatically make the tournament if they win their conference.

Comparing Performance

The question naturally arises: if the model is training on metrics that the committee uses, then why do they produce different results? Does the committee simply have more information than the model? Or is the committee affected by meaningless noise and biases. One way to measure this is to evaluate the performance of the universities in the tournament relative to the committee and model's rankings. I do this in two ways. First, I measure *accuracy* as the proportion of tournament games that the higher seed wins. Second, I measure *error* as the seed differential if the lower seed wins. Error is zero if the higher seed wins.

I assess both the committee and the model's rankings by these criteria in Table 1. Interestingly, the model's rankings have both a smaller error and higher accuracy than the committee. The difference for both measurements is negligible and more research is needed to determine the statistical significance.

Figure 2: Variable Importance

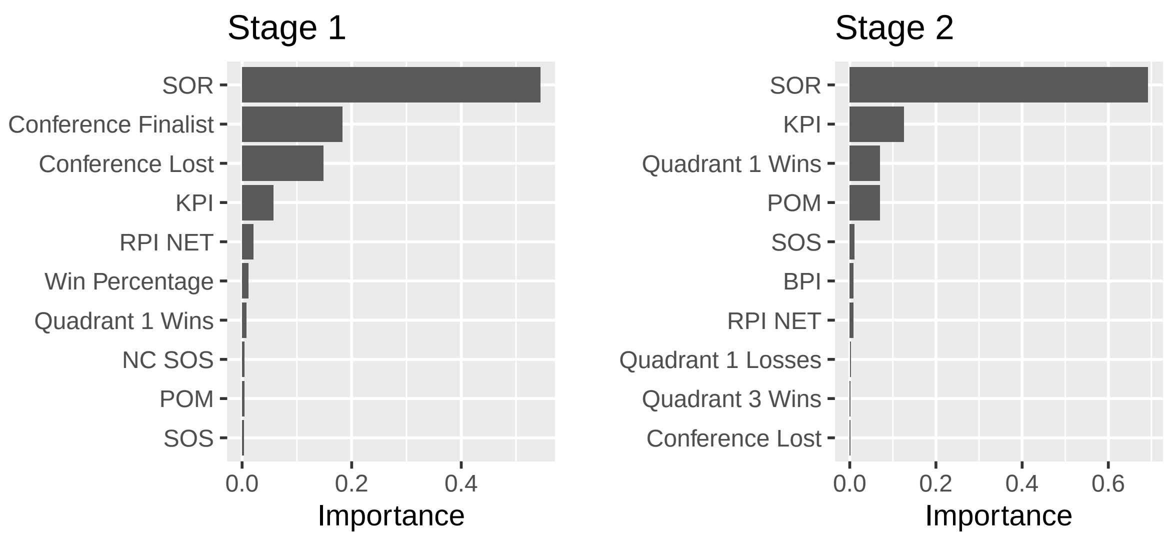


Table 1: Comparing Tournament Performance

Model Mean Error	Committee Mean Error	Model Accuracy	Committee Accuracy
1.405	1.511	0.707	0.699

Weakness/limitations/concerns

The model's prediction power is limited by the simplicity of the predictors. The committee is influenced by a variety of factors like brand names, recent success, or media coverage. The committee may also look at individual games to determine tiebreaks between schools. All of these influences are difficult to quantify.

Perhaps the largest weakness of this study lies in the model structure itself. The committee has a unique problem. They must select 68 teams exactly. The stage 1 model structure that I employ is tasked with predicting if a team will be selected or not. Some years there are more than 68 teams that qualify and other years there are fewer. A more advanced model could be trained to compare these teams directly. In a similar manner, the committee ranks the schools in an ordinal fashion. The stage 2 model treats the university's rank as continuous. In the future, improving both of these models would improve predictions.

Lastly, it is difficult to determine the variability of the model's predictions. Selections and rankings within each year are correlated, so traditional bootstrap methods are compromised. In the future, I would like to employ a by-year bootstrap method that could be used to determine prediction intervals for each universities rank and selection. This may also improve comparisons between the performance of the committee and model's selection.

Appendix

Table A1: Predictors

Predictor	Description
Win Percent	Proportion of Games Won
NC Win Percent	Non-Conference Proportion of Games Won
SOS	Strength of Schedule
NC SOS	Non-Conference Strength of Schedule
SOR	Strength of Record
RPI NET	Ratings Power Index (<2019), NCAA Evaluation Tool (since)
BPI	Basketball Power Index (Predictive)
KPI	Kevin Pauge Index (Predictive)
POM	Ken Pomeroy (Predictive)
Conference Tournament	Victorious Universities Receive Automatic Tournament bid
Previous Year	Wins in Previous Year's Tournament
Quadrant Record	Divides Record into 4 Quadrants by Opponent Skill

Table A2: Misclassifications by Year

Season	Misclassifications
2,017	1
2,018	2
2,019	3
2,021	2
2,022	3
2,023	3
2,024	4

Table A3: MSE by Year

Season	MSE	RMSE
2017	1.73	1.32
2018	0.91	0.95
2019	0.75	0.87
2021	1.05	1.02
2022	1.42	1.19

Table A3: MSE by Year

Season	MSE	RMSE
2023	1.78	1.34
2024	1.77	1.33
Total	1.34	1.16

Code

```r
# knitr
knitr::opts_chunk$set(
  comment = "#", fig.height = 3,
  cache = FALSE,  collapse = TRUE,
  error = TRUE, echo = FALSE,
  message = FALSE, warning = FALSE)

# libraries
library("tidymodels")
library("janitor")
library("tidyverse")
library("rvest")
library("xgboost")
library("rpart.plot")
library("vip")
library("flextable")
library("here")
library("patchwork")

# flextable
set_flextable_defaults(
  font.size = 10, theme_fun = theme_apa,
  padding = 1,
  background.color = "#EFEFEF",
  text.align = "center")

# load data
team0 <- read_csv(here("Project/full_team.csv")) %>%
  select(-`...1`)

team <- team0 %>% filter(ineligible !=1) %>%
  select(-ap, -team_id, -conf_abbrev, -wins, -losses, -nc_wins,
         -nc_losses, -sag, -ineligible, -conf_classification) %>%
  mutate(make_tournament = factor(make_tournament)) %>%
  filter(season > 2016) %>%
  mutate(index = if_else(season < 2020, season - 2016, season - 2017))

# stage 1 boosting

# model
class_boost_spec <- boost_tree(trees = 1500, tree_depth = tune()) %>%
    set_engine("xgboost") %>%
    set_mode("classification")

# tuning grid
boost_tuning_grid <- grid_regular(
  tree_depth(range = c(3, 6)),
  levels = 10)

# model each season
s1_total_pred <- tibble()
for(i in 1:7){
  team_train <- team %>% filter(index != i)
  team_test <- team %>% filter(index == i)
  set.seed(1128)
  team_folds <- vfold_cv(team_train, v = 5, strata = make_tournament)

  s1_recipe <- recipe(make_tournament ~ ., data = team_train) %>%
    step_rm(season, team_name, seed) %>%
    step_normalize(all_numeric_predictors()) %>%
    step_dummy(all_nominal_predictors())

  # fine tune the model
  s1_boost_tune_results <- tune_grid(
    object = workflow() %>%
      add_recipe(s1_recipe) %>%
      add_model(class_boost_spec),
    resamples = team_folds,
    grid = boost_tuning_grid,
    metrics = metric_set(accuracy))

  s1_boost_best_params <- s1_boost_tune_results %>% select_best(metric = "accuracy")

  s1_boost_final_fit <- finalize_workflow(
    workflow() %>%
      add_recipe(s1_recipe) %>%
      add_model(class_boost_spec),
    s1_boost_best_params) %>%
    fit(data = team_train)

  s1_boost_predictions <- augment(s1_boost_final_fit, new_data = team_test)

  s1_total_pred <- bind_rows(s1_total_pred, s1_boost_predictions)
}

# transformation to final predictions
s1_final_predictions <- s1_total_pred %>%
  arrange(season, desc(.pred_1)) %>%
  group_by(season) %>%
  mutate(
    pred_make_tournament = if_else(rank(.pred_0) < 69, 1, 0))

# take predicted teams
order <- s1_final_predictions %>%
  select(-.pred_1, -.pred_0, -.pred_class)

# regression boosting model
reg_boost_spec <- boost_tree(trees = 1500, tree_depth = tune()) %>%
  set_engine("xgboost") %>%
  set_mode("regression")

# run boosting model
s2_total_pred <- tibble()
for(i in 1:7){
  order_train <- order %>% filter(make_tournament == 1, index != i) %>%
    select(-contains("tournament"))
  order_test <- order %>% filter(pred_make_tournament == 1, index == i) %>%
    select(-contains("tournament"))
  set.seed(1128)
  order_folds <- vfold_cv(order_train, v = 5, strata = seed)

  s2_recipe <- recipe(seed ~ ., data = order_train) %>%
    step_rm(season, team_name, index) %>%
    step_normalize(all_numeric_predictors()) %>%
    step_dummy(all_nominal_predictors())

  # fine tune the model
  s2_boost_tune_results <- tune_grid(
    object = workflow() %>%
      add_recipe(s2_recipe) %>%
      add_model(reg_boost_spec),
    resamples = order_folds,
    grid = boost_tuning_grid,
    metrics = metric_set(rmse))

  s2_boost_best_params <- select_best(s2_boost_tune_results, metric = "rmse")

  s2_boost_final_fit <- finalize_workflow(
    workflow() %>%
      add_recipe(s2_recipe) %>%
      add_model(reg_boost_spec),
    s2_boost_best_params) %>%
    fit(data = order_train)

  s2_boost_predictions <- augment(s2_boost_final_fit, new_data = order_test)

  s2_total_pred <- bind_rows(s2_total_pred, s2_boost_predictions)
}

# transform seeds
predicted_seeds <- s2_total_pred %>%
  arrange(season, .pred) %>%
  group_by(season) %>%
  mutate(
    .pred_a = if_else(conf_result == "champion", NA, .pred),
    play_in = case_when(
      rank(-.pred_a) < 3 & (rank(-.pred_a)+1)%%2 == 0 ~ "anchor2",
      rank(-.pred_a) < 3 & (rank(-.pred_a)+1)%%2 == 1 ~ "float2",
      rank(-.pred_a) < 5 & (rank(-.pred_a)+1)%%2 == 0 ~ "anchor1",
      rank(-.pred_a) < 5 & (rank(-.pred_a)+1)%%2 == 1 ~ "float1",
      .default = NA),
    .pred_c = case_when(
      str_detect(play_in, "float") ~ NA,
      .default = .pred),
    rank_c = rank(.pred_c),
    pred_seed_a = case_when(
      rank_c > 60 ~ 16,
      rank_c > 56 ~ 15,
      rank_c > 52 ~ 14,
      rank_c > 48 ~ 13,
      rank_c > 44 ~ 12,
      rank_c > 40 ~ 11,
      rank_c > 36 ~ 10,
      rank_c > 32 ~ 9,
      rank_c > 28 ~ 8,
      rank_c > 24 ~ 7,
      rank_c > 20 ~ 6,
      rank_c > 16 ~ 5,
      rank_c > 12 ~ 4,
      rank_c > 8 ~ 3,
      rank_c > 4 ~ 2,
      .default = 1),
    pred_seed_b = if_else(is.na(play_in), NA, pred_seed_a),
    pred_seed = case_when(
      play_in == "float1" ~ quantile(pred_seed_b, na.rm = TRUE, probs = 0),
      play_in == "float2" ~ quantile(pred_seed_b, na.rm = TRUE, probs = 1/3),
      .default = pred_seed_a)) %>%
  select(-.pred_a, -play_in, -.pred_c, -rank_c, -pred_seed_a, -pred_seed_b) %>%
  ungroup()

# seed rmse/mse
seed_rmse <- predicted_seeds %>% filter(!is.na(seed)) %>%
  group_by() %>%
  summarise(
    MSE = mean((pred_seed - seed)^2),
    RMSE = sqrt(mean((pred_seed - seed)^2))) %>%
  mutate(Season = "Total")

# residual by seed
predicted_seeds %>%
  mutate(residual = seed - pred_seed) %>%
  ggplot() +
  geom_jitter(aes(x = seed, y = residual), width = .3, height = .3) +
  geom_abline(slope = 0, linetype = "dashed", color = "indianred3") +
  scale_y_continuous(n.breaks = 9) +
  scale_x_continuous(n.breaks = 9) +
  labs(x = "Seed", y = "Residual")

# variable importance stage 1
p1 <- vip(s1_boost_final_fit) +
  scale_x_discrete(
    labels = c(
      "sor" = "SOR",
      "conf_result_finalist" = "Conference Finalist",
      "conf_result_lost" = "Conference Lost",
      "kpi" = "KPI",
      "rpi_net" = "RPI NET",
      "win_perc" = "Win Percentage",
      "rpi_net_wq1" = "Quadrant 1 Wins",
      "non_conf_sos" = "NC SOS",
      "pom" = "POM",
      "sos" = "SOS")) +
  labs(title = "Stage 1")

# variable importance stage 2
p2 <- vip(s2_boost_final_fit) +
    scale_x_discrete(
    labels = c(
      "rpi_net_wq1" = "Quadrant 1 Wins",
      "sor" = "SOR",
      "conf_result_lost" = "Conference Lost",
      "kpi" = "KPI",
      "rpi_net" = "RPI NET",
      "bpi" = "BPI",
      "rpi_net_lq1" = "Quadrant 1 Losses",
      "rpi_net_wq3" = "Quadrant 3 Wins",
      "pom" = "POM",
      "sos" = "SOS")) +
  labs(title = "Stage 2")

p1 + p2

# combining predicted and actual seed
simple_seed <- predicted_seeds %>% select(team_name, season, pred_seed) %>%
  full_join(team %>% filter(!is.na(seed)) %>% select(team_name, season, seed),
            by = join_by(team_name, season)) %>%
  mutate(
    deviance = case_when(
      is.na(pred_seed) ~ 1,
      .default = pred_seed - seed),
    direction = case_when(
      is.na(pred_seed) ~ "over",
      pred_seed > seed ~ "over",
      pred_seed < seed ~ "under",
      .default = "match"))

# grabbing actual results
compare_tourney <- read_csv(here("Project/full_tourney.csv")) %>%
  select(season, winner, contains("team_name")) %>%
  filter(season > 2016) %>%
  left_join(simple_seed %>% rename_with(~ paste0("t1_", .x)),
            by = join_by(season == t1_season, t1_team_name)) %>%
  left_join(simple_seed %>% rename_with(~ paste0("t2_", .x)),
            by = join_by(season == t2_season, t2_team_name)) %>%
  mutate(
    play_in = case_when(
      t1_seed == 16 & t2_seed == 16 ~ "play-in",
      t1_seed == 11 & t2_seed == 11 ~ "play-in",
      t1_seed == 12 & t2_seed == 12 ~ "play-in",
      t1_seed == 10 & t2_seed == 10 ~ "play-in",
      .default = NA)) %>%
  rowwise() %>%
  mutate(
    model_result = case_when(
```

```r
      winner == "t1" & is.na(t1_pred_seed) & !is.na(t2_pred_seed) ~ 0,
      winner == "t1" & !is.na(t1_pred_seed) & is.na(t2_pred_seed) ~ 1,
      winner == "t1" & t1_pred_seed < t2_pred_seed ~ 1,
      winner == "t1" & t2_pred_seed < t1_pred_seed ~ 0,
      winner == "t2" & !is.na(t1_pred_seed) & is.na(t2_pred_seed) ~ 0,
      winner == "t2" & is.na(t1_pred_seed) & !is.na(t2_pred_seed) ~ 1,
      winner == "t2" & t2_pred_seed < t1_pred_seed ~ 1,
      winner == "t2" & t1_pred_seed < t2_pred_seed ~ 0,
      .default = NA),
    model_error = case_when(
      winner == "t1" & is.na(t1_pred_seed) & !is.na(t2_pred_seed) ~ max(12 - t2_pred_seed, 0),
      winner == "t1" & is.na(t2_pred_seed) ~ 0,
      winner == "t1" ~ max(t1_pred_seed - t2_pred_seed, 0),
      winner == "t2" & is.na(t2_pred_seed) & !is.na(t1_pred_seed) ~ max(12 - t1_pred_seed, 0),
      winner == "t2" & is.na(t1_pred_seed) ~ 0,
      winner == "t2" ~ max(t2_pred_seed - t1_pred_seed, 0),
      .default = 0),
    committee_result = case_when(
      winner == "t1" & t1_seed < t2_seed ~ 1,
      winner == "t1" & t2_seed < t1_seed ~ 0,
      winner == "t2" & t2_seed < t1_seed ~ 1,
      winner == "t2" & t1_seed < t2_seed ~ 0,
      .default = NA),
    committee_error = case_when(
      winner == "t1" ~ max(t1_seed - t2_seed, 0),
      winner == "t2" ~ max(t2_seed - t1_seed, 0),
      .default = 0))

# compute accuracy
compare_acc <- compare_tourney %>%
  group_by() %>%
  # group_by(season) %>%
  summarise(
    `Model Accuracy` = mean(model_result, na.rm = TRUE),
    `Committee Accuracy` = mean(committee_result, na.rm = TRUE))

# mean error and accuracy
compare_tourney %>%
  filter(is.na(play_in)) %>%
  group_by() %>%
  # group_by(season) %>%
  summarise(
    `Model Mean Error` = mean(model_error),
    `Committee Mean Error` = mean(committee_error)) %>%
  bind_cols(compare_acc) %>%
  flextable() %>%
  align(align = "center") %>%
  width(j = c(1:4), width = 1.5) %>%
  colformat_double(digits = 3) %>%
  set_caption(caption = "Comparing Tournament Performance")

# clean names
clean <- function(vector){
  vector = str_replace_all(vector, "_", " ")
  vector = str_to_title(vector)
  vector = str_replace_all(vector, "Perc", "Percent")
  vector = str_replace_all(vector, "pi", "PI")
  vector = str_replace_all(vector, "Sos", "SOS")
  vector = str_replace_all(vector, "Sor", "SOR")
  vector = str_replace_all(vector, "Pom", "POM")
  vector = str_replace_all(vector, "Nc", "NC")
  vector = str_replace_all(vector, "Non Conf", "NC")
  vector = str_replace_all(vector, "Net", "NET")
  vector = str_replace_all(vector, "Conf Result", "Conference Tournament")
  vector = str_replace_all(vector, "Prev Year Result", "Previous Year")
}

# descriptors table
tibble(
  predictor = team %>% select(-team_name, -season, -contains("rpi_net_"), -is_net, -make_tournament, -i
    relocate(rpi_net, .after = sor) %>% colnames(),
  description = c("Proportion of Games Won",
                  "Non-Conference Proportion of Games Won",
                  "Strength of Schedule",
                  "Non-Conference Strength of Schedule",
                  "Strength of Record",
                  "Ratings Power Index (<2019), NCAA Evaluation Tool (since)",
                  "Basketball Power Index (Predictive)",
                  "Kevin Pauge Index (Predictive)",
                  "Ken Pomeroy (Predictive)",
                  "Victorious Universities Receive Automatic Tournament bid",
                  "Wins in Previous Year's Tournament"
                  )) %>%
  mutate(
    predictor = clean(predictor)) %>%
  add_row(
    predictor = "Quadrant Record", description = "Divides Record into 4 Quadrants by Opponent Skill"
  ) %>%
  rename_with(str_to_title) %>%
  flextable() %>%
  align(align = "left") %>%
  width(j = 1, width = 2) %>%
  width(j = 2, width = 4) %>%
  set_caption(caption = "Predictors")

# misclassifications by year
s1_final_predictions %>%
  filter(make_tournament != pred_make_tournament) %>%
  summarise(
    misclassifications = sum(make_tournament == "1")) %>%
  rename_with(~str_to_title(.)) %>%
  flextable() %>%
  align(align = "center") %>%
  colformat_double(j = 1, digits = 0) %>%
  width(j = 2, width = 1.4) %>%
  set_caption(caption = "Misclassifications by Year")
```

```r
# MSE by year
predicted_seeds %>% filter(!is.na(seed)) %>%
  group_by(season) %>%
  summarise(
    MSE = mean((pred_seed - seed)^2),
    RMSE = sqrt(mean((pred_seed - seed)^2))) %>%
  mutate(Season = as.character(season)) %>%
  select(-season) %>%
  bind_rows(seed_rmse) %>%
  relocate(Season, .before = MSE) %>%
  flextable() %>%
  align(align = "center") %>%
  set_caption(caption = "MSE by Year")
```