All-NBA Modeling

AUTHOR
Ayush Batra

Loading Packages and Data

Here are the packages that I use in this notebook.

```
library(tidyverse)
library(tidymodels)
library(recipes)
library(nbastatR)
library(knitr)
Sys.setenv("VROOM_CONNECTION_SIZE" = 2*131072)
```

This next code block loads in the dataset of actual All-NBA voting shares along with the regular season player logs for all seasons from 2000 to 2024. The goal of this is to use regular season player stats to predict All-NBA voting shares from 2000 to 2023. At the end, we can use the model to predict the All-NBA voting shares for this year (if there were no 65 game minimum for awards, which is a new rule that began in the 2023-24 season).

```
allnba <- read_csv("data/allnba.csv")


player_logs <- game_logs(seasons = 2000:2024,
                         result_types = "player",
                         season_types = "Regular Season")
```

```
Acquiring NBA basic player game logs for the 1999-00 Regular Season
Acquiring NBA basic player game logs for the 2000-01 Regular Season
Acquiring NBA basic player game logs for the 2001-02 Regular Season
Acquiring NBA basic player game logs for the 2002-03 Regular Season
Acquiring NBA basic player game logs for the 2003-04 Regular Season
Acquiring NBA basic player game logs for the 2004-05 Regular Season
Acquiring NBA basic player game logs for the 2005-06 Regular Season
```

```
Acquiring NBA basic player game logs for the 2006-07 Regular Season
Acquiring NBA basic player game logs for the 2007-08 Regular Season
Acquiring NBA basic player game logs for the 2008-09 Regular Season
Acquiring NBA basic player game logs for the 2009-10 Regular Season
Acquiring NBA basic player game logs for the 2010-11 Regular Season
Acquiring NBA basic player game logs for the 2011-12 Regular Season
Acquiring NBA basic player game logs for the 2012-13 Regular Season
Acquiring NBA basic player game logs for the 2013-14 Regular Season
Acquiring NBA basic player game logs for the 2014-15 Regular Season
Acquiring NBA basic player game logs for the 2015-16 Regular Season
Acquiring NBA basic player game logs for the 2016-17 Regular Season
Acquiring NBA basic player game logs for the 2017-18 Regular Season
Acquiring NBA basic player game logs for the 2018-19 Regular Season
Acquiring NBA basic player game logs for the 2019-20 Regular Season
Acquiring NBA basic player game logs for the 2020-21 Regular Season
Acquiring NBA basic player game logs for the 2021-22 Regular Season
Acquiring NBA basic player game logs for the 2022-23 Regular Season
Acquiring NBA basic player game logs for the 2023-24 Regular Season
```

```
# variable for total number of All-NBA voting shares per season
# its 9 since 1st-team votes get 5 points, 2nd-team votes get 3 points,
# and 3rd-team votes get 1 point
TOTAL_SHARES = 9
```

Data Cleaning

This first part just generates some dataframes for player ids and team seasons.

```
# get dataframe of player ids and names
player_data <- player_logs %>%
  distinct(idPlayer, namePlayer) %>%
  distinct(idPlayer, .keep_all = TRUE)

# get team game counts
teams_data <- player_logs %>%
  distinct(slugTeam, idGame, yearSeason) %>%
  group_by(slugTeam, yearSeason) %>%
  summarize(teamG = n()) %>%
  ungroup()
```

```
`summarise()` has grouped output by 'slugTeam'. You can override using
the
`.groups` argument.
```

To predict All-NBA shares, I used player counting stats (scaled per 100 possessions) in addition to stats about playing time. First, we aggregate the stats for each player in each season.

```
# aggregate relevant player stats for each player season
player_stats <- player_logs %>%
  group_by(idGame, idTeam) %>%
  mutate(TmMIN = sum(minutes),
         TmPOSS = sum(fga) + sum(tov) + 0.44 * sum(tov) - sum(oreb)) %>%
  ungroup() %>%
  mutate(PlayerPOSS = minutes / (TmMIN / 5) * TmPOSS) %>%
  arrange(dateGame) %>%
  select(-pctFG, -pctFG3, -pctFG2, -pctFT, -hasVideo) %>%
  group_by(idPlayer, yearSeason) %>%
  summarize(G = n(),
            Tm = last(slugTeam),
            Min_Pct = sum(minutes) / sum(TmMIN / 5),
            WP = mean(isWin),
            POSS = sum(PlayerPOSS),
            across(fgm : pts, ~ sum(.x))) %>%
  ungroup() %>%
  left_join(teams_data, by = c("Tm" = "slugTeam", "yearSeason")) %>%
  mutate(pct_G = G / teamG) %>%
  left_join(player_data, by = "idPlayer")
```

```
`summarise()` has grouped output by 'idPlayer'. You can override using
the
`.groups` argument.
```

There is a little bit of cleaning to do with the All-NBA data. Some of the names don't match up with the game logs data. The code below edits the names so that they do match up.

```
# see which names have to be cleaned
allnba %>%
  filter(Season >= 2000) %>%
  anti_join(player_stats, by = c("Player" = "namePlayer",
```

```
                                  "Season" = "yearSeason")) %>%
  distinct(Player)
```

```
# A tibble: 16 × 1
   Player
   <chr>
 1 Steve Smith
 2 Peja Stojaković
 3 Manu Ginóbili
 4 Nenad Krstić
 5 Hedo Türkoğlu
 6 Nenê
 7 Ömer Aşık
 8 J.J. Hickson
 9 Nikola Peković
10 Nikola Vučević
11 Goran Dragić
12 Nikola Jokić
13 Kristaps Porziņģis
14 Luka Dončić
15 Karl-Anthony Towns*
16 Ben Simmons*
```

```
# clean some of the names
allnba2 <- allnba %>%
  select(Player, Season, Share) %>%
  mutate(Player = str_remove(Player, "[*]")) %>%
  mutate(Player = case_when(
    Player == "Steve Smith" ~ "Steven Smith",
    Player == "Peja Stojaković" ~ "Peja Stojakovic",
    Player == "Manu Ginóbili" ~ "Manu Ginobili",
    Player == "Nenad Krstić" ~ "Nenad Krstic",
    Player == "Hedo Türkoğlu" ~ "Hedo Turkoglu",
    Player == "Nenê" ~ "Nene",
    Player == "Ömer Aşık" ~ "Omer Asik",
    Player == "J.J. Hickson" ~ "JJ Hickson",
    Player == "Nikola Peković" ~ "Nikola Pekovic",
    Player == "Nikola Vučević" ~ "Nikola Vucevic",
    Player == "Goran Dragić" ~ "Goran Dragic",
    Player == "Nikola Jokić" ~ "Nikola Jokic",
    Player == "Kristaps Porziņģis" ~ "Kristaps Porzingis",
```

```
    Player == "Luka Dončić" ~ "Luka Doncic",
    .default = Player
  ))

# verify that all names match up
allnba2 %>%
  filter(Season >= 2000) %>%
  anti_join(player_stats, by = c("Player" = "namePlayer",
                                 "Season" = "yearSeason")) %>%
  distinct(Player)
```

```
# A tibble: 0 × 1
# ℹ 1 variable: Player <chr>
```

Next, we join the All-NBA voting shares data to the player stats, and normalize the counting stats to be per 100 possessions. In addition, we filter out players that did not play enough.

```
# join all-nba stats to player stats
player_stats <- player_stats %>%
  left_join(allnba2, by = c("namePlayer" = "Player",
                            "yearSeason" = "Season")) %>%
  mutate(Share = ifelse(is.na(Share), 0, Share))

# normalize stast per 100 possessions
player_stats2 <- player_stats %>%
  select(-minutes, -fgm, -fga, -pts, -treb) %>%
  mutate(across(fg3m : pf, ~ 100 * .x / POSS))

# filter so only have players that played enough
# must have played 40 percent of team minutes and 30% of team games
player_stats2 <- player_stats2 %>%
  filter(Min_Pct > 0.4, pct_G > 0.3)
```

Next, we do some data processing. For the relevant variables, we transform the per 100 numbers into z-scores (grouped by year) to account for the changes across seasons. For example, there were a lot more 3-point attempts per 100 in 2023 than in 2000.

To model All-NBA voting shares, I split the process into 2 parts: 1 model for giving the probability of having at least 1 vote for All-NBA, and a different model for the magnitude of the share if the player received at least 1 vote. To ensure the predictions for the magnitude are between 0 and 1, I transformed the response variable by using a logit transformation. The small adjustments I made were to subtract .001 from the numerator (so that shares of 1 would become .999 and the minimum share of .002 would still be above 0 at .001; the minimum share is .002 since there are 100 voters and the max points per voter is 5 points, so there are 500 points maximum, and the minimum number of points is 1, so 1/500 is .002). The denominator adds .001 so that shares of 1 are divided by .001 within the logarithm instead of dividing by 0. Overall, the formula for the transformed share is:

S: voting shares

S_T: transformed voting shares

$$S_T = \log\left(\frac{S - .001}{1.001 - S}\right)$$

Again, just to summarize, the only reason for the adding/subtracting of .001 was to ensure there was no dividing by 0 or taking the log of 0.

```
# processing for data to predict magnitude of share
player_stats3 <- player_stats2 %>%
  filter(yearSeason != 2024) %>%
  select(idPlayer, yearSeason, pct_G, Min_Pct : pf, Share) %>%
  # convert stats to z-scores
  group_by(yearSeason) %>%
  mutate(across(fg3m : pf, ~ (.x - mean(.x)) / sd(.x))) %>%
  ungroup() %>%
  select(-POSS) %>%
  # only include those with at least 1 vote
  filter(Share > 0) %>%
  # transform the share so predictions are bounded between 0 and 1
  mutate(trans_Share = log((Share - 0.001) / (1.001 - Share)))


# processing for data to predict whether player got a vote or not
player_stats4 <- player_stats2 %>%
```

```
  filter(yearSeason != 2024) %>%
  select(idPlayer, yearSeason, pct_G, Min_Pct : pf, Share) %>%
  # convert stats to z-scores
  group_by(yearSeason) %>%
  mutate(across(fg3m : pf, ~ (.x - mean(.x)) / sd(.x))) %>%
  ungroup() %>%
  select(-POSS) %>%
  # create binary variable for if player got at least 1 vote
  mutate(hasShare = ifelse(Share > 0, "Yes", "No"),
         hasShare = factor(hasShare, levels = c("Yes", "No")))
```

Modeling

Magnitude Fit

Now it is time to specify and fit the models. Again, the models are split into 2 parts: 1 model assesses the expected magnitude of an All-NBA voting share if a player received at least 1 vote, and the other model assesses the probability of receiving at least 1 vote. We can get a final prediction for expected All-NBA awards share by multiplying the probability by the magnitude. First, we will fit the model to predict the magnitude of the share

```
# train-test split
set.seed(123)
mag_split <- initial_split(player_stats3)
mag_train <- training(mag_split)
mag_test <- testing(mag_split)

# specify model as linear regression
mag_spec <- linear_reg() %>%
  set_engine("lm")

# formula for model
mag_rec <- recipe(trans_Share ~ ., data = mag_train) %>%
  update_role(idPlayer, yearSeason, new_role = "id") %>%
  step_rm(Share)

mag_wflow <- workflow() %>%
```

```
  add_model(mag_spec) %>%
  add_recipe(mag_rec)

# fit model
mag_fit <- mag_wflow %>%
  fit(mag_train)

# display output
mag_fit %>%
  tidy() %>%
  kable(digits = 4)
```

term	estimate	std.error	statistic	p.value
(Intercept)	-24.2269	1.0878	-22.2717	0.0000
pct_G	4.5935	0.6285	7.3089	0.0000
Min_Pct	10.4638	1.2703	8.2372	0.0000
WP	9.8088	0.6134	15.9912	0.0000
fg3m	1.4006	0.4678	2.9941	0.0029
fg3a	-0.2520	0.4757	-0.5298	0.5964
fg2m	2.5503	0.2893	8.8144	0.0000
fg2a	-1.1992	0.2801	-4.2815	0.0000
ftm	0.7233	0.2220	3.2583	0.0012
fta	0.1172	0.2396	0.4891	0.6249
oreb	0.6526	0.1393	4.6832	0.0000
dreb	0.6993	0.0922	7.5839	0.0000
ast	1.1916	0.1022	11.6634	0.0000
stl	0.0302	0.0740	0.4078	0.6835
blk	0.6564	0.0839	7.8275	0.0000
tov	-0.0372	0.1126	-0.3304	0.7412
pf	-0.4846	0.1140	-4.2512	0.0000

We can also see the model's predictiveness on both the train and test sets.

```
# add training set predictions
mag_pred_train <- predict(mag_fit, new_data = mag_train) %>%
  # convert predictions back into units of shares
  mutate(yhat = exp(.pred) / (1 + exp(.pred))) %>%
  bind_cols(mag_train)

# add test set predictions
mag_pred_test <- predict(mag_fit, new_data = mag_test) %>%
  mutate(yhat = exp(.pred) / (1 + exp(.pred))) %>%
  bind_cols(mag_test)

# scores on train and test sets
rsq(mag_pred_train, truth = Share, estimate = yhat)
```

```
# A tibble: 1 × 3
  .metric .estimator .estimate
  <chr>   <chr>          <dbl>
1 rsq     standard       0.729
```

```
rsq(mag_pred_test, truth = Share, estimate = yhat)
```

```
# A tibble: 1 × 3
  .metric .estimator .estimate
  <chr>   <chr>          <dbl>
1 rsq     standard       0.789
```

We see that there is an R^2 of 0.729 on the training set and 0.789 on the testing set. The score is actually better on the test set, so there is not an indication of overfitting. In addition, an r-squared of about 0.8 on the test set is a sign of a solid fit.

Indicator Fit

Now we model the probability of a player receiving at least 1 vote.

```
# train test split
set.seed(456)
ind_split <- initial_split(player_stats4)
ind_train <- training(ind_split)
```

```r
ind_test <- testing(ind_split)

# specify logistic regression
ind_spec <- logistic_reg() %>%
  set_engine("glm")

# formula
ind_rec <- recipe(hasShare ~ ., data = ind_train) %>%
  update_role(idPlayer, yearSeason, new_role = "id") %>%
  step_rm(Share)

ind_wflow <- workflow() %>%
  add_model(ind_spec) %>%
  add_recipe(ind_rec)

# fit model
ind_fit <- ind_wflow %>%
  fit(ind_train)

# display output
ind_fit %>%
  tidy() %>%
  kable()
```

term	estimate	std.error	statistic	p.value
(Intercept)	25.2632405	1.3582007	18.6005213	0.0000000
pct_G	-5.5350053	0.6887382	-8.0364433	0.0000000
Min_Pct	-17.6828894	1.3301104	-13.2943016	0.0000000
WP	-9.6180641	0.7519450	-12.7909144	0.0000000
fg3m	-3.1466764	0.5630664	-5.5884644	0.0000000
fg3a	2.3300871	0.5710913	4.0800603	0.0000450
fg2m	-2.5482296	0.3862027	-6.5981656	0.0000000
fg2a	1.2937169	0.3535506	3.6592131	0.0002530
ftm	-1.4430627	0.3486530	-4.1389658	0.0000349
fta	0.3547781	0.3714348	0.9551558	0.3394989

term	estimate	std.error	statistic	p.value
oreb	-0.3549364	0.1638473	-2.1662631	0.0302911
dreb	-0.9495266	0.1276626	-7.4377815	0.0000000
ast	-1.0388236	0.1393115	-7.4568423	0.0000000
stl	-0.1319595	0.0935022	-1.4112976	0.1581569
blk	-0.8737038	0.1062868	-8.2202467	0.0000000
tov	-0.0431550	0.1459284	-0.2957273	0.7674383
pf	0.5037171	0.1352592	3.7240876	0.0001960

And again we can look at the predictions. This time, since its a logistic regression, we use area under the ROC curve to measure predictive accuracy.

```
# predictions for train and test set
ind_pred_train <- predict(ind_fit, new_data = ind_train, type = "prob")
  bind_cols(ind_train)

ind_pred_test <- predict(ind_fit, new_data = ind_test, type = "prob") %
  bind_cols(ind_test)

# scores for both sets
roc_auc(ind_pred_train, truth = hasShare, .pred_Yes)
```

```
# A tibble: 1 × 3
  .metric .estimator .estimate
  <chr>   <chr>          <dbl>
1 roc_auc binary         0.981
```

```
roc_auc(ind_pred_test, truth = hasShare, .pred_Yes)
```

```
# A tibble: 1 × 3
  .metric .estimator .estimate
  <chr>   <chr>          <dbl>
1 roc_auc binary         0.980
```

We get an ROC AUC of about 0.98 for both train and test sets. This is an indicator that this fit is pretty good at determining which players get an All-NBA vote and which ones don't. In addition, the model is unlikely to be overfit as the train and test set accuracies are very similar.

Final Fitting

Lastly, we will generate the final expected predictions by fitting both models on all the players and multiplying together to get the expected share.

```
player_stats5 <- player_stats2 %>%
  group_by(yearSeason) %>%
  mutate(across(fg3m : pf, ~ (.x - mean(.x)) / sd(.x))) %>%
  ungroup()

# get final predictions
final_mags <- predict(mag_fit, new_data = player_stats5) %>%
  mutate(yhat_mag = exp(.pred) / (1 + exp(.pred)))
final_inds <- predict(ind_fit, new_data = player_stats5, type = "prob")
final_stats <- player_stats2 %>%
  bind_cols(final_mags) %>%
  bind_cols(final_inds)

# multiply indicator and expected
final_stats <- final_stats %>%
  mutate(exp_Share = .pred_Yes * yhat_mag,
         # will use this in next step
         adj_Share = exp_Share)
```

There is one last thing I want to do to make the predicted All-NBA results better. If we look at the total predicted All-NBA shares per year, we see that it is not always close to 9, which is the actual total shares per year (It is 9 because a 1st team vote is 5 points, a 2nd team vote is 3 points, and a 3rd team vote in 1 point; add them up and you get 9).

```
final_stats %>%
  group_by(yearSeason) %>%
```

```
  summarize(predicted_shares = sum(adj_Share)) %>%
  tail(15)
```

```
# A tibble: 15 × 2
   yearSeason predicted_shares
        <dbl>            <dbl>
 1       2010             7.12
 2       2011             7.61
 3       2012             6.45
 4       2013             6.32
 5       2014             6.41
 6       2015             6.75
 7       2016             8.19
 8       2017             8.84
 9       2018             9.69
10       2019             8.12
11       2020             7.46
12       2021             9.16
13       2022             7.84
14       2023             7.25
15       2024             9.09
```

To make predictions better, we can normalize them so that there are approximately 9 total predicted shares per year.

```
# 10 iterations
for (i in c(1:10)) {
  # for each player, multiply by a factor but ensure share doesn't
  # exceed 1
  final_stats <- final_stats %>%
    group_by(yearSeason) %>%
    mutate(mult = TOTAL_SHARES / sum(adj_Share), # multiplicative facto
           adj_Share = adj_Share * mult,
           adj_Share = ifelse(adj_Share > 1, 1, adj_Share)) %>%
    ungroup() %>%
    select(-mult)
}
```

Now, after normalizing, the total predicted shares per season is much closer to 9 for all seasons.

```r
final_stats %>%
  group_by(yearSeason) %>%
  summarize(predicted_shares = sum(adj_Share)) %>%
  tail(15)
```

```
# A tibble: 15 × 2
   yearSeason predicted_shares
        <dbl>            <dbl>
 1       2010             9.00
 2       2011             9.00
 3       2012             9.00
 4       2013             9.00
 5       2014             9.00
 6       2015             9.00
 7       2016             9.00
 8       2017             9.00
 9       2018             9
10       2019             9.00
11       2020             9.00
12       2021             9
13       2022             9.00
14       2023             9.00
15       2024             9
```

Finally, we gather the results by team and season and save the rersults for 2024.

```r
# gather totals by team/season
team_stats <- final_stats %>%
  group_by(Tm, yearSeason) %>%
  summarize(n = n(),
            Shares = sum(Share),
            exp_Shares = sum(exp_Share)) %>%
  ungroup()
```

```
`summarise()` has grouped output by 'Tm'. You can override using the
`.groups`
argument.
```

```r
# get team predictions for 2024
allnba24 <- team_stats %>%
```

```r
  filter(yearSeason == 2024) %>%
  select(Tm, exp_Shares)

# see top few teams in total All-NBA shares for 2024
allnba24 %>%
  arrange(-exp_Shares) %>%
  head()
```

```
# A tibble: 6 × 2
  Tm    exp_Shares
  <chr>      <dbl>
1 MIL        1.07
2 PHI        1.05
3 DEN        1.00
4 DAL        0.988
5 OKC        0.984
6 BOS        0.803
```

```r
# save stats as csv
write_csv(allnba24, "data/allnba_2024.csv")
```