Predicting the Winner of NBA's 2021 Most Improved Player Award, and Finding Candidates for 2022

Sumitro Datta

September 05, 2021

Re-Introduction

For my HarvardX capstone project, I performed an analysis on whether I could predict last year's Most Improved Player and this year's candidates (found here). With the 2020-2021 regular season in the books, I decided to go back and improve on it. Major changes include:

- a more robust dataset (instead of using the IMPORTHTML function on Google Sheets multiple times, I learned how to webscrape)
- a couple of new variables (experience or seasons played in the league, and whether the player in question has been an All-Star selection in any prior season)
- different machine learning algorithms (a decision tree does not lend itself to use with this many granular decimals)

Introduction

As the beginning of a new NBA season nears, predictions start to roll in from pundits. In terms of major player awards, the Most Improved Player is the most "predicted by gut feeling".

- there are usually a select group of players that have a chance at Most Valuable Player and Defensive Player of the Year, with minor turnover year-over-year
- Sixth Man of the Year (given to the best performing player that does not start the game on the court) has morphed into a type: the high scoring guard that ironically often places in the top 5 in minutes played on the team
  - a guard has won 13 of the past 15 years, and the winner regardless of position has placed in the top 5 in minutes played on the team 12 of the 15 years
- Rookie of the Year, by definition, has turnover every year, but predictions tend towards players taken near the beginning of the draft

The goal of this analysis is to use machine learning to make a two-fold prediction.

1. Predict the winner of this year's (2021) MIP award.
2. Predict candidates of next year's (2022) MIP award.

On May 25, 2021, Julius Randle was revealed as the 2021 MIP. While disappointed I wasn't able to create a model in time, it'll serve as a good check.

Steps that were performed:

1. loading the data and swapping NA's with 0's
2. visualizing how MIP winners have performed and how much vote share they received
3. create a version of the data that shows statistical jumps (current season minus previous season)
4. separating out evaluation sets (the 2020 season) and test sets (the 1987 season for predicting the 2020 winner, the 1986 and 2019 seasons for predicting 2021 candidates)
5. training four models: a linear baseline, a k-nearest-neighbors model, a random forest model and a support vector machine model

Note: to simplify the Season column, I refer to each season by its second half. So you might also hear the current season referred to as the 2020-2021 season.

Methods/Analysis

Loading Packages

Let's start by loading required packages.

```r
if(!require(tidyverse))
  install.packages("tidyverse", repos = "http://cran.us.r-project.org")
if(!require(tidymodels))
  install.packages("tidymodels", repos = "http://cran.us.r-project.org")
#for variable importance
if(!require(vip))
  install.packages("vip", repos = "http://cran.us.r-project.org")
#ranger is random forest algorithm wrapper
if(!require(ranger))
  install.packages("ranger", repos = "http://cran.us.r-project.org")
#matrix stats
if(!require(matrixStats))
  install.packages("matrixStats", repos = "http://cran.us.r-project.org")
#kableExtra allows more customization of tables
if(!require(kableExtra))
  install.packages("kableExtra")
if(!require(RColorBrewer))
  install.packages("RColorBrewer", repos = "http://cran.us.r-project.org")
#for dark background plots
if(!require(ggdark))
  install.packages("ggdark", repos = "http://cran.us.r-project.org")
```

Downloading the Data

This is a dataset that I have compiled from Basketball-Reference. Each player season contains some identifying factors:

- a unique Season ID
- the player's name
- the player's age as of February 1st of that season (per Basketball-Reference recordkeeping)
- the team that the player played for (if they played for multiple teams, it was recorded as TOT)
- the actual season
- the birth year

    - this is to break ties, as there are instances of 2 players having the same name and playing in the same season as well as father and son playing in different eras under the same name

- the number of years of experience the player has accumulated
    - a rookie, or a player with no prior experience, is given the value 1
    - this value is incremented up every subsequent season that a player has played in; if a player is a rookie in the 1990 NBA season, skips the 1991 NBA season and comes back for the 1992 NBA season, his experience value in 1992 is 2, not 3

There are three comma-separated value files of player statistics:

- Player Per Game contains per game stats
- Per 100 Poss contains stats per 100 possessions
    - this adjusts for pace, which is the number of possessions teams use in 48 minutes (the typical length of a game, excluding overtimes)
    - this is necessary because the late 1990s/early 2000s were notoriously slow, so raw statistics were depressed
- Advanced contains advanced stats
    - for more information, refer to this *Basketball-Reference glossary*

The above three files will be combined into one dataframe. For players that played for multiple teams in one season, the partial seasons will be discarded. The shooting percentage columns will by multiplied by 100 to remain consistent with the other percent columns. In the previous year's project, we removed games played and games started columns as predictors due to various fluctuations in games played between seasons (reasons being contraction, expansion, lockouts, the COVID-19 pandemic, etc). We will convert the games started to a percentage of games played and we will change the games played to a percentage of maximum playable games. This maximum will differ for players who played for multiple teams in one season. Lastly, we will take out columns that are the result of linear combinations of other columns:

- field goals made are the sum of 2-point field goals made & 3-point field goals made, similarly with attempts
- total rebounds are the sum of defensive rebounds & offensive rebounds
- win shares & box plus-minus, two holistic advanced statistics, are the sum of their offensive & defensive components

```r
#specify columns because otherwise birth year is read as logical
cols_for_stats=cols(
  .default = col_double(),
  player = col_character(),
  hof = col_logical(),
  pos = col_character(),
  lg = col_character(),
  tm = col_character()
)

advanced=read_csv("Data/Advanced.csv",col_types = cols_for_stats)
per_poss=read_csv("Data/Per 100 Poss.csv",col_types = cols_for_stats)
per_game=read_csv("Data/Player Per Game.csv",col_types = cols_for_stats)
max_games_tots=per_game %>% filter(tm=="TOT") %>% group_by(season,lg,tm) %>%
  summarize(max_games_tot=max(g,na.rm = TRUE)) %>% ungroup()
max_games=per_game %>% filter(tm!="TOT") %>% group_by(season,lg) %>%
  summarize(max_games_non_tot=max(g,na.rm = TRUE)) %>% ungroup()
per_game_enhanced=left_join(per_game,max_games_tots) %>% left_join(.,max_games) %>%
  mutate(max_games_playable=coalesce(max_games_tot,max_games_non_tot)) %>%
  select(-c(max_games_non_tot,max_games_tot))
combined=left_join(per_game_enhanced,per_poss) %>% left_join(.,advanced) %>%
  mutate(tm=ifelse(tm=="TOT","1TOT",tm)) %>%
  group_by(player_id,season) %>% arrange(tm) %>% slice(1) %>%
  mutate(tm=ifelse(tm=="1TOT","TOT",tm)) %>% arrange(season,player) %>%
  mutate(across(c(fg_percent,x3p_percent,x2p_percent,e_fg_percent,ft_percent,ts_percent),
              ~.*100)) %>%
  mutate(g_percent=g/max_games_playable*100,gs_percent=gs/g*100,.before=g) %>%
  select(-c(g,gs,max_games_playable)) %>%
  select(-c(starts_with("fg_per_"),starts_with("fga_per_"),starts_with("trb_per_"),ws,bpm))
```

Anotther file that will be loaded is the Player Award Shares comma-separated values file. This file keeps track of both the number of first-place votes and the share of an award that a player received. Share equals the amount of points received divided by the maximum number of points possible. The file has multiple awards so it will be filtered for only MIP results. Up to the 2002 season, MIP voters were given one vote, and the player with the highest number of votes won the award. Starting in 2003, a point system was introduced, where:

- a first-place vote equals five points
- a second-place vote equals three points
- a third-place vote equals one point
- the player with the most ***points*** won the award

Theoretically, it is possible to have the most first-place votes and not win the award, but it has never happened in the history of the award.

```r
mip_share=read_csv("Data/Player Award Shares.csv",col_types = cols(
  .default = col_double(),
  award= col_character(),
  player = col_character(),
  pos = col_character(),
  tm = col_character(),
  winner = col_logical()
)) %>% filter(award=="mip") %>% select(season,seas_id:tm,share)
```

While the MIP share is tracked for the season in which the vote was received, it will also be tracked for the season prior. The latter column will help identify candidates for next season. For example, Pascal Siakam of the Toronto Raptors won the MIP award in 2019 with an MIP vote share of 0.938. This would be recorded in Siakam's 2019 season under `share` and in Siakam's 2018 Season under `share_next_seas`. Another column based on the MIP vote share is the cumulative share received by a player in previous seasons. This should lessen the future predictions of past MIP winners & those who finished as high runner-ups, as their improvement was already recognized.

```r
stats_and_share=left_join(combined,mip_share) %>%
  mutate(share=replace_na(share,0)) %>% group_by(player_id) %>% arrange(season) %>%
  mutate(share_next_seas=lead(share),
         share_before_current_seas=cumsum(share)-share) %>%
  ungroup()
```

The last file that will be loaded is the All-Star Selections comma-separated values file. I noticed last year that established players were factoring into my models more than I'd like. These players were under-performing due to injury and returning to their regular star level a year later. I initially thought that the number of All-Star selections a player has had prior to the current season could act as a proxy to a player's regular talent level. However, it might suffice to just determine whether or not a player has ever been an All-Star, effectively turning this into a categorical predictor with only 2 possible values rather than a numeric one with multiple. After joining the All-Star data with the statistics and vote share data, our last transformation is to remove seasons prior to 1984-1985 (which is one season before the inaugural awarding of the MIP award).

```
all_stars=read_csv("Data/All-Star Selections.csv") %>% group_by(player) %>%
  slice_min(season) %>% mutate(first_allstar="Yes") %>% ungroup() %>%
  select(-c(team,replaced,hof))
final_data=left_join(stats_and_share,all_stars) %>% group_by(player_id) %>%
  #move all-star appearance to next season
  mutate(all_stars_lag=lag(first_allstar)) %>%
  #add zero for easier filling of NA's
  mutate(all_stars_tot=ifelse(lead(all_stars_lag)=="Yes","No",NA)) %>%
  #combine two previous columns
  mutate(been_allstar_before=coalesce(all_stars_lag,all_stars_tot)) %>%
  fill(been_allstar_before,.direction="downup") %>%
  mutate(been_allstar_before=replace_na(been_allstar_before,"No")) %>%
  ungroup() %>% filter(season > 1984) %>%
  select(-c(all_stars_lag,all_stars_tot,first_allstar))
```

Let's see if there is any missing data.

```
colSums(is.na(final_data))
```

```
##              seas_id             season          player_id
##                    0                  0                  0
##               player         birth_year                hof
##                    0              15850                  0
##                  pos                age         experience
##                    0                  0                  0
##                   lg                 tm          g_percent
##                    0                  0                  0
##           gs_percent        mp_per_game         fg_percent
##                    0                  0                 54
##          x3p_per_game       x3pa_per_game        x3p_percent
##                    0                  0               2395
##          x2p_per_game       x2pa_per_game        x2p_percent
##                    0                  0                 90
##         e_fg_percent        ft_per_game       fta_per_game
##                   54                  0                  0
##           ft_percent       orb_per_game       drb_per_game
##                  504                  0                  0
##          ast_per_game       stl_per_game       blk_per_game
##                    0                  0                  0
##          tov_per_game        pf_per_game       pts_per_game
##                    0                  0                  0
##                   mp    x3p_per_100_poss   x3pa_per_100_poss
##                    0                  3                  3
##     x2p_per_100_poss   x2pa_per_100_poss     ft_per_100_poss
##                          3                          3                          3
##           fta_per_100_poss           orb_per_100_poss           drb_per_100_poss
##                          3                          3                          3
##           ast_per_100_poss           stl_per_100_poss           blk_per_100_poss
##                          3                          3                          3
##           tov_per_100_poss            pf_per_100_poss           pts_per_100_poss
##                          3                          3                          3
##                      o_rtg                      d_rtg                        per
##                         29                          3                          3
##                 ts_percent                     x3p_ar                       f_tr
##                         47                         54                         54
##                orb_percent                drb_percent                trb_percent
##                          3                          3                          3
##                ast_percent                stl_percent                blk_percent
##                          3                          3                          3
##                tov_percent                usg_percent                        ows
##                         38                          3                          0
##                        dws                      ws_48                       obpm
##                          0                          3                          0
##                       dbpm                       vorp                      share
##                          0                          0                          0
##            share_next_seas share_before_current_seas        been_allstar_before
##                       3016                          0                          0
```

Looks like there's a significant amount of missing data, but it can be broken down into bite-size parts:

- the NA's in the shooting percentages are due to a lack of attempts
- the consistent 3 NA's in the per-100-possessions and advanced stats are due to a lack of data, as those 3 players played less than a minute of game time
  - the rest of the NA's in per-100-possessions & advanced can also be attributed to a lack of data, but rather than minutes played, it is the absence of the stat itself
- all the players who don't need a birth year to separate them from another player with the same name also have an NA

The above NA's can all be converted to zeroes.

```
final_data[is.na(final_data)]<-0
```

Data Exploration and Visualizations

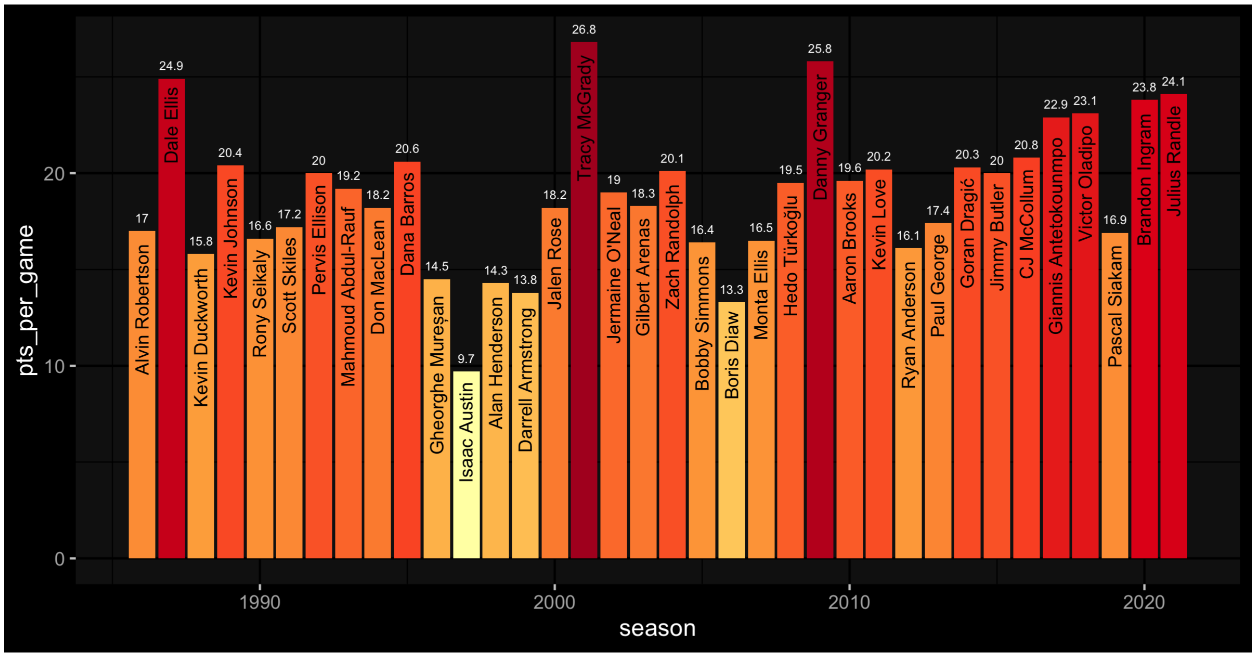
We'll start by seeing how much vote share MIP award winners have received.

There are two seasons that have a vote share of zero: 1985 (which was the season before the first awarding of the MIP) and 1987.

There was an MIP award given out in 1987 to Dale Ellis of the Seattle SuperSonics. However, when researching, I could not find any voting records for the season. I'll discuss what I plan to do with this season a little further in the report.

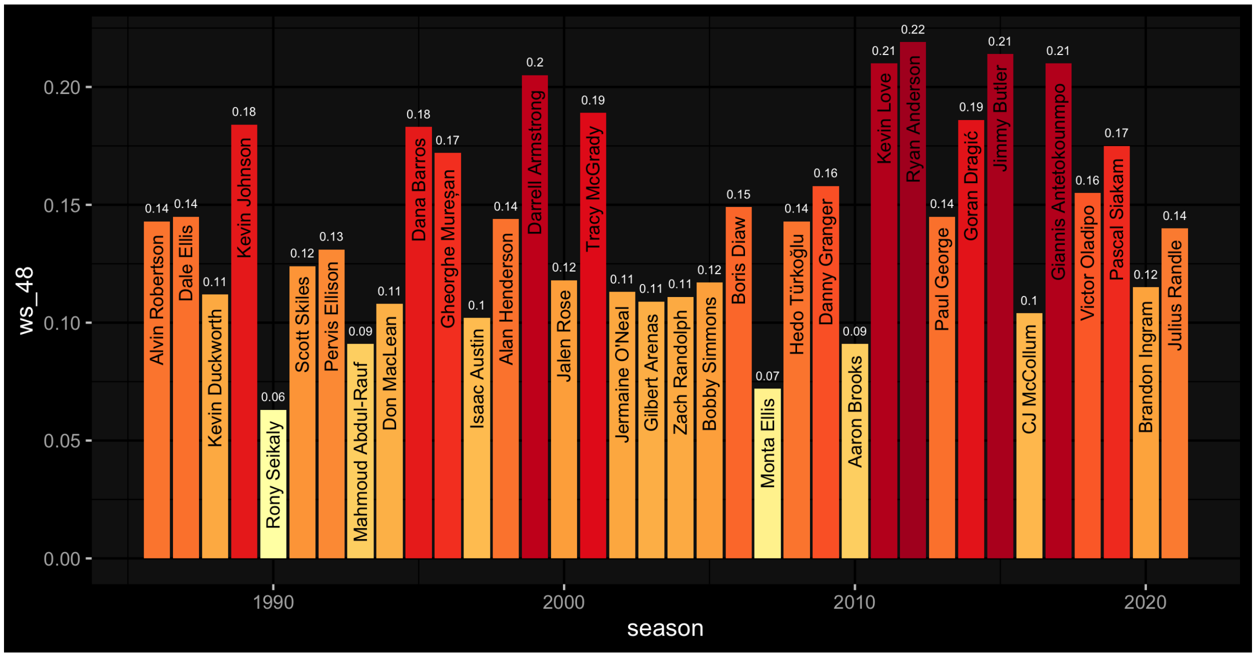
Secondly, we'll plot the scoring per game of the MIP winners. The two missing seasons (1985,1987) have been removed.

season	player	share
1985	Adrian Dantley	0.000
1986	Alvin Robertson	0.385
1987	Dale Ellis	0.000
1988	Kevin Duckworth	0.413
1989	Kevin Johnson	0.565
1990	Rony Seikaly	0.402
1991	Scott Skiles	0.260
1992	Pervis Ellison	0.417
1993	Mahmoud Abdul-Rauf	0.255
1994	Don MacLean	0.545
1995	Dana Barros	0.476
1996	Gheorghe Mureșan	0.442
1997	Isaac Austin	0.357
1998	Alan Henderson	0.284
1999	Darrell Armstrong	0.458
2000	Jalen Rose	0.264
2001	Tracy McGrady	0.597
2002	Jermaine O'Neal	0.413
2003	Gilbert Arenas	0.488
2004	Zach Randolph	0.626
2005	Bobby Simmons	0.624
2006	Boris Diaw	0.776
2007	Monta Ellis	0.546
2008	Hedo Türkoğlu	0.608
2009	Danny Granger	0.602
2010	Aaron Brooks	0.655
2011	Kevin Love	0.690
2012	Ryan Anderson	0.430
2013	Paul George	0.518
2014	Goran Dragić	0.648
2015	Jimmy Butler	0.829
2016	CJ McCollum	0.860
2017	Giannis Antetokounmpo	0.856
2018	Victor Oladipo	0.988
2019	Pascal Siakam	0.938
2020	Brandon Ingram	0.652
2021	Julius Randle	0.986



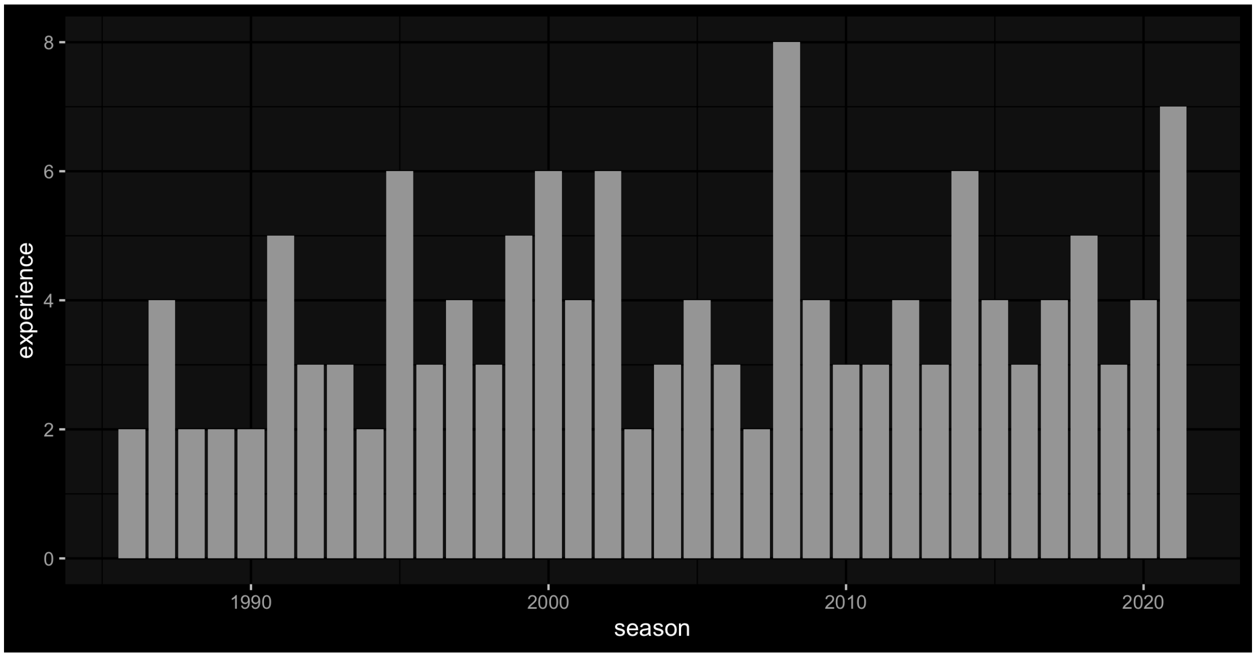
The lowest points per game average by an MIP is 9.7 (achieved by Isaac Austin in 1997), while the highest is 26.8 (achieved by Tracy McGrady in 2001).

Next, we'll plot the MIP winners by Win Shares per 48 minutes. Win Shares represent how much a player has contributed to his team's wins by comparing his output to a marginal player. Win Shares per 48 Minutes is a rate stat to compare how effective a player was during their time on the court, regardless of the actual minutes played.



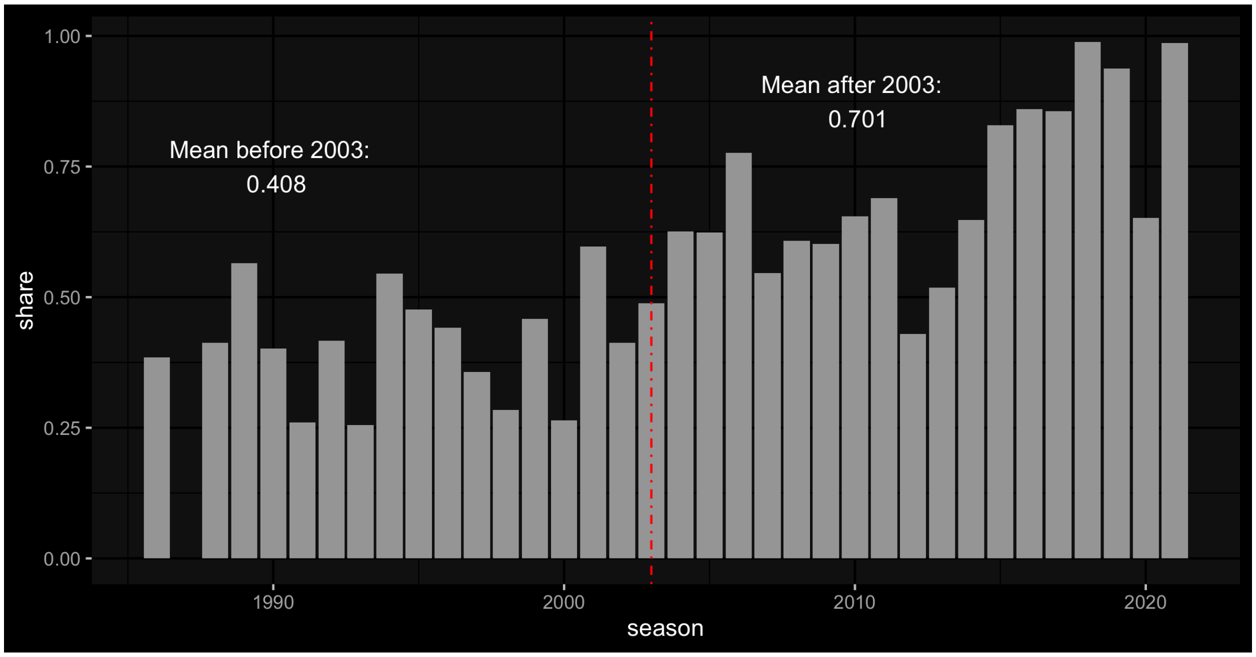
The lowest win shares per 48 minutes by an MIP is 0.06 (achieved by Rony Seikaly in 1990), while the highest is 0.22 (achieved by Ryan Anderson in 2012).

Thirdly, we'll plot the experience of the winners.



In the first few years of awards, second-year players ruled the roost, winning 4 of the first 5 awards. Recently, the trend has been to recognize a player in their third or fourth season.

Finally, we'll plot how the max voting share has evolved over the life of the award, placing a vertical line at the 2003 season when the voting system changed.



Looks like the average jumps up significantly after the voting system was changed. In recent years, there's been more consensus.

Preprocessing

Before we fit our model, we should do some pre-processing. We've already done some pre-processing by handling missing values.

For the 2021 award winner, we realize that players must have at least one prior season played in order to garner consideration. So we will subset out all players who have only one season played in the database. Removing one-season players has the unfortunate side effect on removing some players who played their last games in 1985, but had a career before then. This shouldn't be too worrisome, as no MIP has won in their final season in the league.

In addition, statistical jumps from the previous season play a much bigger part than the current season's statistics in determining the winner. A player going from 20 points per game to 21 points per game will not merit as much campaigning as a player going from 8 PPG to 18 PPG.

Finally, we will also remove some extraneous information from the data because we don't want to predict based off of a player's team or position. While age was used as a predictor variable last year, we'll remove it since we have our new experience variable. Experience could also be looked at as "league age" and expected improvement is more closely tied to experience than player age.

```
one_seasoners<-final_data %>% group_by (player_id) %>%
  mutate(n=n_distinct(season)) %>% filter(n==1) %>% select(seas_id,player_id,player)
full_jumps_data<-final_data %>%
  #remove the one-season players
  filter(!(seas_id %in% one_seasoners$seas_id)) %>%
  group_by(player_id) %>%
  #arrange seasons from first to last within each player
  arrange(season,.by_group=TRUE) %>%
  #subtract prior season from current season
  mutate_at(.vars=c(colnames(final_data[12:68])),
            .funs=funs(`diff`=.-lag(.,default=first(.)))) %>%
  #do not keep the raw stats, only differences
  select(seas_id:tm, g_percent_diff:pts_per_game_diff,x3p_per_100_poss_diff:vorp_diff,
         share,share_before_current_seas,been_allstar_before) %>%
  #remove first season from every group (no difference)
  slice(-1) %>% arrange(seas_id) %>% ungroup() %>%
  #do not keep irrelevant information
  select(seas_id:player,experience,g_percent_diff:been_allstar_before)
```

For predicting candidates in 2022, it's a much more diverse subset. So we'll just remove some of the irrelevant factors from the data.

```
full_data<-final_data %>% select(seas_id:player,experience,
                                 g_percent:been_allstar_before) %>%
  select(-c(mp,share))
```

Creating an Evaluation Set

Our evaluation set is straightforward: it is players who have played in the 2021 season. So we segment it off here.

```
eval_2021<-full_data %>% filter (season==2021)
full_data<-full_data %>% filter (season != 2021)
eval_jumps_2021<-full_jumps_data %>% filter (season == 2021)
full_jumps_data<-full_jumps_data %>% filter (season != 2021)
```

As discussed earlier, I could not find any voting records for the 1987 season. All I could find was the winner (Dale Ellis of the Seattle SuperSonics). Instead of removing the 1987 data, I realize that I could use it as a test set or additional performance metric before applying the algorithm to find the 2020 season winner. Ideally, the algorithm would project Ellis to be the winner, and also provide me with the other candidates from that season. In the same vein, I'll cast the 1986 season and 2020 season as test sets in the journey to predict the 2021 candidates. By moving the test sets to the bottom of their respective datasets, I can use the `initial_time_split` function from tidymodels to segment them out.

```r
full_data_tidy<-full_data %>%
  mutate(train_or_test=ifelse(season %in% c(1986,2020),"test","train")) %>%
  arrange(desc(train_or_test)) %>% select(-train_or_test)
full_jumps_data_tidy<-full_jumps_data %>%
  mutate(train_or_test=ifelse(season == 1987,"test","train")) %>%
  arrange(desc(train_or_test)) %>% select(-train_or_test)

full_jumps_split=initial_time_split(
  full_jumps_data_tidy,prop=full_jumps_data_tidy %>%
    summarize(test_prop=1-sum(season==1987)/n()) %>% pull()
  )
jumps_trainer=training(full_jumps_split)
jumps_tester=testing(full_jumps_split)

full_split=initial_time_split(
  full_data_tidy,prop=full_data_tidy %>%
    summarize(test_prop=1-sum(season %in% c(1986,2020))/n()) %>% pull()
  )
trainer=training(full_split)
tester=testing(full_split)
```

Training Models

Models to Predict the 2021 Winner

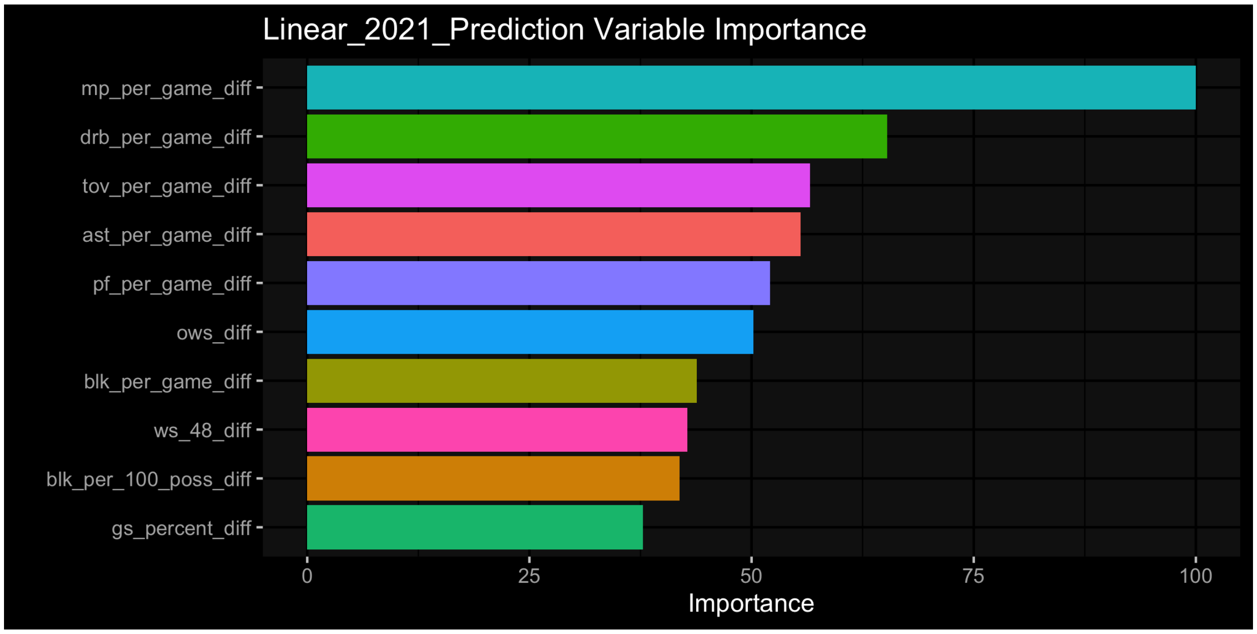
With caret, everything could be thrown into the train function: cross-validation, tuning, data, etc. Tidymodels is more explicit in its steps. We'll start off with common steps across all models. The standard 10-fold cross validation is how the data is going to be resampled. Recipes is tidymodels' pre-processing package. There's two pre-processing steps to perform. The first step is to convert the been_allstar_before column into a numeric column. The second pre-processing step is to update the role of the rest of the identifying information so that it's easier to match predictions to players.

```r
this_year_cv=vfold_cv(jumps_trainer,v=10)
this_year_recipe<-recipe(share ~ .,data=full_jumps_split) %>%
  step_dummy(been_allstar_before) %>%
  update_role(seas_id:player,new_role="id")
```

Our first model is a simple linear regression, which will serve as a baseline.

```r
wf <- workflow() %>% add_recipe(this_year_recipe) %>%
  add_model(linear_reg() %>% set_engine("lm") %>% set_mode("regression"))
tune_lin <- wf %>%
  tune_grid(resamples=this_year_cv,metrics=metric_set(rmse))
wf <- wf %>% finalize_workflow(tune_lin %>% select_best(metric="rmse"))
lin_this_yr=fit(wf,jumps_trainer)
```

Let's get the most important variables of the linear model.



The most important variable is the difference in minutes per game. As a player gets more time on the court, there's more opportunity to rack up positive stats like rebounds, assists and blocks but also negative stats like personal fouls and turnovers. Offensive improvement is easier to recognize than defensive improvement, so it makes sense that we see offensive win shares in the top 10 and not defensive win shares.

Next up is the k-nearest neighbors model. The intuition behind using this model is that maybe Most Improved Player winners are most similar to other winners. We can tune the neighbors parameter to change how many neighbors to compare to. We check values from 5 to 50.

```r
wf <- workflow() %>% add_recipe(this_year_recipe) %>%
  add_model(nearest_neighbor(neighbors = tune()) %>%
  set_engine("kknn") %>% set_mode("regression"))
tune_knn <- wf %>% tune_grid(resamples=this_year_cv,
                             grid=expand.grid(neighbors=5:50),
                             metrics=metric_set(rmse))
wf <- wf %>% finalize_workflow(tune_knn %>% select_best(metric="rmse"))
knn_this_yr=fit(wf,jumps_trainer)
tune_knn %>% select_best(metric="rmse") %>% knitr::kable()
```

neighbors	.config
38	Preprocessor1_Model34

Looks like the optimal amount of neighbors is 38.

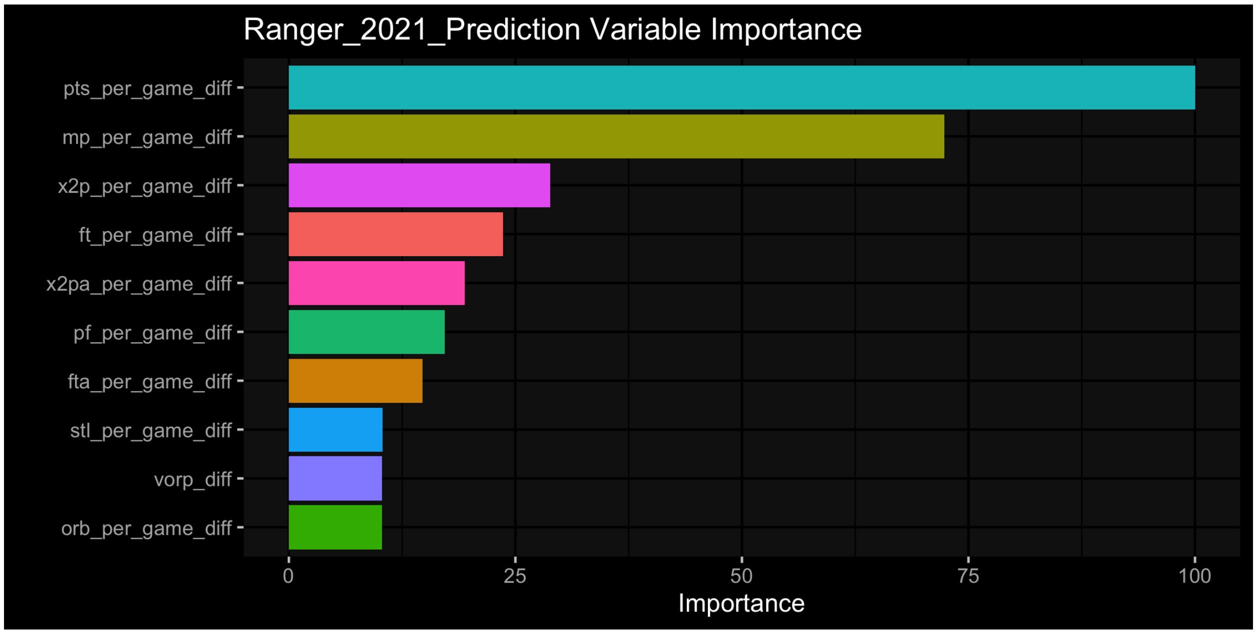
The next model is a random forest which return the average of multiple decision trees. The intuition behind using models from the decision tree family is that maybe there are certain statistical thresholds that a potential MIP winner needs to meet in order to merit consideration. As a player passes more of these checkpoints, their vote share increases. Random forests are better than decision trees in that they reduce instability by averaging multiple trees. Unfortunately, the cost is interpretability. There is no tree diagram that is representative of the decisions made. With *random* forest algorithms, we need to set a seed to keep the work reproducible.

Random forest algorithms require an explicit call for variable importance, so we'll ask for permutation importance. A simplified explanation for permutation importance is shuffling a predictor's values and seeing how much the error increases. As a predictor's importance increases, it is difficult for the rest of the model to compute accurate predictions without it. There are 3 tuning parameters:

- trees is the number of decision trees to create
  - the default is 500, which we'll keep
- mtry is the number of variables to split at each decision node
  - the default is the rounded square root of the number of variables, which in this case would be `round(sqrt(59))`
  - we will try all integers between 5 & 10
- min_n is the minimum number of observations needed to split a node
  - the default is 5 for a regression problem like this, which we'll keep

```
set.seed(20,sample.kind = "Rounding")
wf<-workflow() %>% add_recipe(this_year_recipe) %>%
  add_model(rand_forest(mtry = tune()) %>%
  set_engine("ranger",importance="permutation") %>% set_mode("regression"))
tune_forest<-wf %>% tune_grid(resamples=this_year_cv,
                              grid=expand.grid(mtry=5:10),
                              metrics=metric_set(rmse))
wf <-wf %>% finalize_workflow(tune_forest %>% select_best(metric="rmse"))
ranger_this_yr=fit(wf,jumps_trainer)
tune_forest %>% select_best(metric="rmse") %>% knitr::kable()
```

mtry	.config
7	Preprocessor1_Model3



The top 8 variables all heed from the per-game subset, with the advanced stat of value over replacement player sneaking into the number 9 spot. The top 2 variables of difference in points per game and difference in minutes per game are in a separate tier of importance from the others.

Finally, we train the support vector machine on the model. SVMs attempt to separate classes with a hyperplane. Support vectors are the points closest to the hyperplane, named as such because the hyperplane would change if those points were removed.

There are two tuning parameters:

- rbf_sigma: determines how well to fit the training data (higher=fit closer to training)
  - with a high sigma, every workaday big with middling stats might be predicted to get close to Mozgov money
- cost: tradeoff between smoothness of boundaries and correct classification
  - with a high cost, leads to too wiggly of a boundary, and might not generalize to test sets
  - tests using C=0.25, C=0.5 and C=1

```r
set.seed(100,sample.kind = "Rounding")
wf <- workflow() %>% add_recipe(this_year_recipe) %>%
  add_model(svm_rbf(rbf_sigma=tune(),cost=tune()) %>%
  set_engine("kernlab") %>% set_mode("regression"))
tune_svm <- wf %>% tune_grid(resamples=this_year_cv,
                        grid=expand.grid(rbf_sigma=0.1^seq(2:5),
                                         cost=c(0.25,0.5,1)),
                        metrics=metric_set(rmse))
wf <- wf %>% finalize_workflow(tune_svm %>% select_best(metric="rmse"))
svm_this_yr=fit(wf,jumps_trainer)
tune_svm %>% select_best(metric="rmse") %>% knitr::kable()
```

cost	rbf_sigma	.config
1	0.01	Preprocessor1_Model10

Unfortunately, there is no concept of variable importance for an SVM model.

We'll evaluate performance by checking RMSE, the predicted vote share of Dale Ellis in the 1987 test set and the top 3 candidates in 1987. Remember, Ellis won the 1987 MIP but there were no voting records found during research.

Method	RMSE	Ellis Vote Share	Top 3 Candidates	Top 3 Vote Shares
Linear	0.0345003	0.0628871	Michael Jordan	0.0771842
Linear	0.0345003	0.0628871	Dale Ellis	0.0628871
Linear	0.0345003	0.0628871	Otis Thorpe	0.0406809
KNN	0.0316558	0.2568028	Dale Ellis	0.2568028
KNN	0.0316558	0.2568028	Michael Jordan	0.2059979
KNN	0.0316558	0.2568028	Otis Thorpe	0.1754788
Ranger	0.0316984	0.3314636	Dale Ellis	0.3314636
Ranger	0.0316984	0.3314636	Michael Jordan	0.2183658
Ranger	0.0316984	0.3314636	Bill Cartwright	0.1966584
SVM	0.0335963	0.0859808	Dale Ellis	0.0859808
SVM	0.0335963	0.0859808	Michael Jordan	0.0603610
SVM	0.0335963	0.0859808	Otis Thorpe	0.0507833

The vote shares are quite low, but especially so for both the linear & SVM models. The highest shares in the linear and SVM models are less than half the 3rd highest share in the knn model. Ellis & Michael Jordan are present in all 4 top 3s, while the third candidate varies. In the linear model, Jordan even finishes first. This is somewhat misleading, because Jordan broke his foot the previous year and missed 64 games. He came back with a vengeance, upping his points per game by 14.4. But Jordan was already a star, and stars are implicitly excluded from winning MIP. I'd hoped that including the variable of being an All-Star prior to the current season would alleviate this problem, but it wasn't a big factor in the models.

Thorpe is a worthy candidate: he increased his scoring by nine points and WS/48 by 0.04 while decreasing his turnover percentage. Cartwright's case is similar to Jordan as a player who didn't play often in the previous season due to injury.

Models to Predict 2022 MIP Candidates

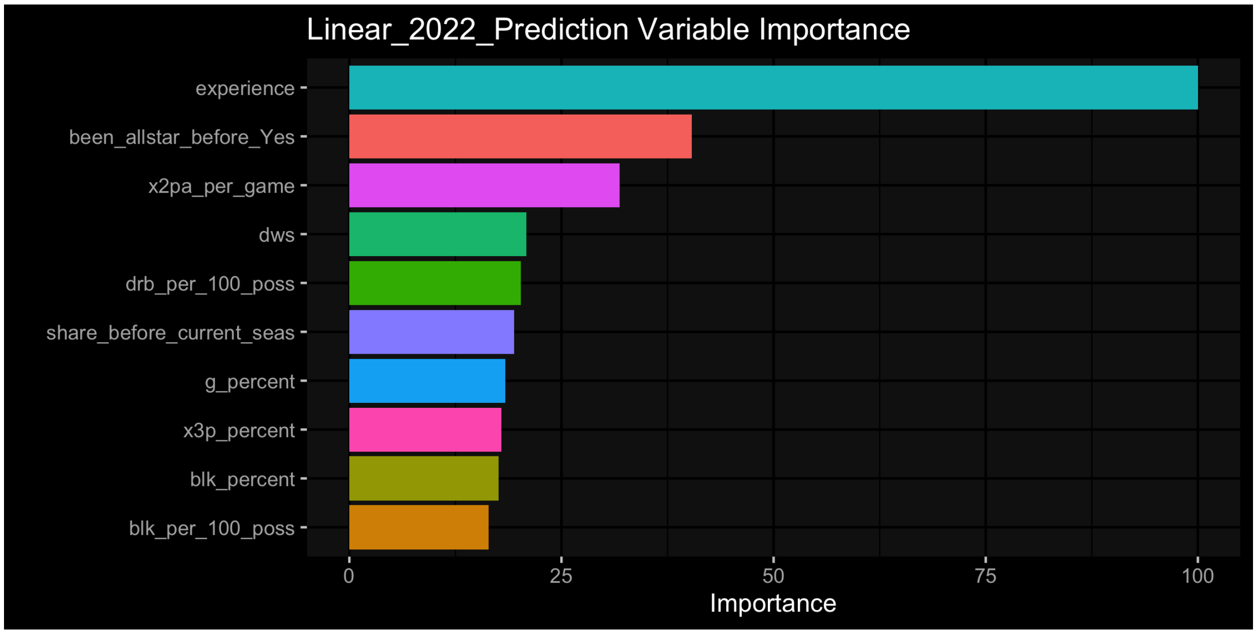
We will apply the same algorithms we used for predicting the 2021 winner. We'll continue to evaluate performance using RMSE but now we have 2 test sets: 1986 and 2020.

```r
next_year_cv=vfold_cv(trainer,v=10)
next_year_recipe<-recipe(share_next_seas ~ .,data=full_split) %>%
  step_dummy(been_allstar_before) %>%
  update_role(seas_id:player,new_role="id")
```

Once again, we'll start with the linear regression.

```r
wf <- workflow() %>% add_recipe(next_year_recipe) %>%
  add_model(linear_reg() %>% set_engine("lm") %>% set_mode("regression"))
tune_lin <- wf %>% tune_grid(resamples=next_year_cv,metrics=metric_set(rmse))
wf <- wf %>% finalize_workflow(tune_lin %>% select_best(metric="rmse"))
lin_next_yr=fit(wf,trainer)
```

Let's get the most important variables of the linear model.



Two of the new variables of experience and previous All-Star recognition are the most important variables in the linear model, with the third of accumulated vote share in previous seasons placing 6th.

```r
wf <- workflow() %>% add_recipe(next_year_recipe) %>%
  add_model(nearest_neighbor(neighbors = tune()) %>%
  set_engine("kknn") %>% set_mode("regression"))
tune_knn <- wf %>% tune_grid(resamples=next_year_cv,
                             grid=expand.grid(neighbors=5:50),
                             metrics=metric_set(rmse))
wf <- wf %>% finalize_workflow(tune_knn %>% select_best(metric="rmse"))
knn_next_yr=fit(wf,trainer)
tune_knn %>% select_best(metric="rmse") %>% knitr::kable()
```

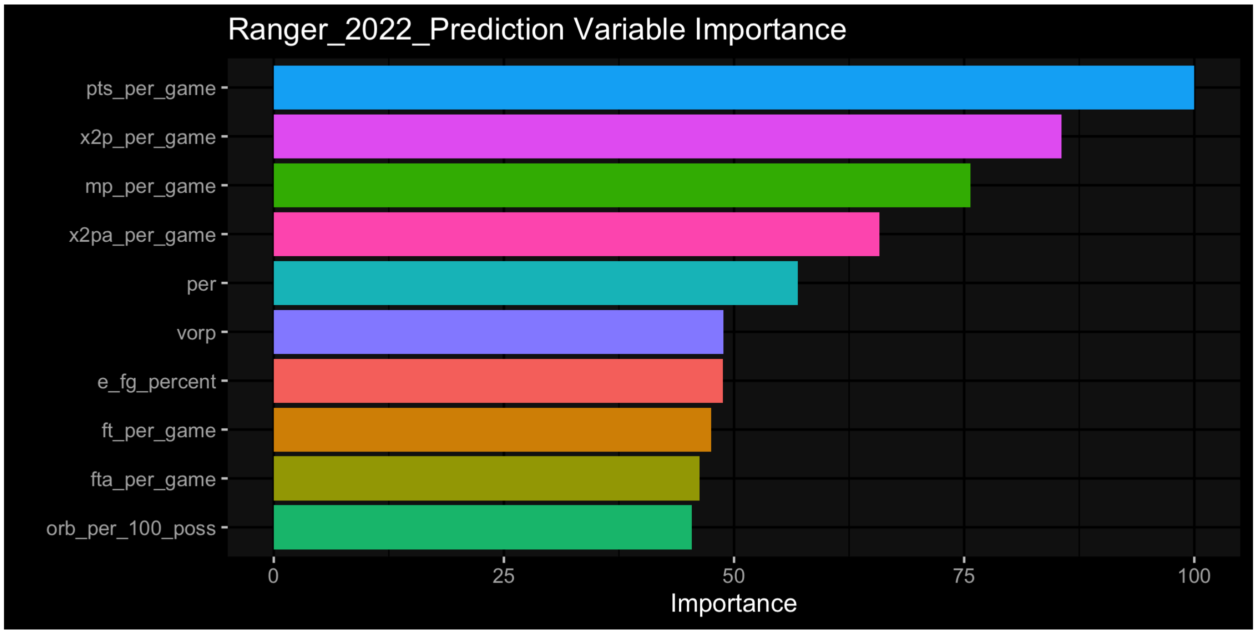
neighbors	.config
50	Preprocessor1_Model46

The maximum number of neighbors tested seems to produce the best results, with 50 neighbors needed to determine a player's MIP share in the next season.

```r
wf <- workflow() %>% add_recipe(next_year_recipe) %>%
  add_model(rand_forest(mtry = tune()) %>%
  set_engine("ranger",importance="permutation") %>% set_mode("regression"))
tune_forest<-wf %>% tune_grid(resamples=next_year_cv,
                              grid=expand.grid(mtry=5:10),
                              metrics=metric_set(rmse))
wf <-wf %>% finalize_workflow(tune_forest %>% select_best(metric="rmse"))
ranger_next_yr=fit(wf,trainer)
tune_forest %>% select_best(metric="rmse") %>% knitr::kable()
```

mtry	.config
5	Preprocessor1_Model1

Here's the most important variables in the random forest model.



The top 4 most important variables are from the per-game subset.

```r
wf <- workflow() %>% add_recipe(next_year_recipe) %>%
  add_model(svm_rbf(rbf_sigma=tune(),cost=tune()) %>%
  set_engine("kernlab") %>% set_mode("regression"))
tune_svm<-wf %>% tune_grid(resamples=next_year_cv,
                           grid=expand.grid(rbf_sigma=0.1^seq(2:5),
                                            cost=c(0.25,0.5,1)),
                              metrics=metric_set(rmse))
wf <-wf %>% finalize_workflow(tune_svm %>% select_best(metric="rmse"))
svm_next_yr=fit(wf,trainer)
tune_svm %>% select_best(metric="rmse") %>% knitr::kable()
```

cost	rbf_sigma	.config
0.5	0.1	Preprocessor1_Model05

Let's check up on our performance metrics. We'll be checking two separate sets of candidates: 1987 using 1986 season stats and 2021 using 2020 season stats.

Method	RMSE	1987 Top Candidates	1987 Top Vote Shares	2021 Top Candidates	2021 Top Vote Shares
Linear	0.0324409	Charles Barkley	0.0164642	Luka Dončić	0.0181696
Linear	0.0324409	Alvin Robertson	0.0122835	Trae Young	0.0147355
Linear	0.0324409	Karl Malone	0.0104052	James Harden	0.0139819
KNN	0.0330052	Alton Lister	0.0401656	Bam Adebayo	0.0851442
KNN	0.0330052	Kevin McKenna	0.0378416	Kelly Oubre Jr.	0.0658284
KNN	0.0330052	Benoit Benjamin	0.0344605	Jaylen Brown	0.0599741
Ranger	0.0326502	John Stockton	0.0541831	Luka Dončić	0.0636569
Ranger	0.0326502	Spud Webb	0.0405791	Trae Young	0.0480464
Ranger	0.0326502	Doc Rivers	0.0322000	Jaren Jackson Jr.	0.0470269
SVM	0.0324897	John Stockton	0.0085273	Shai Gilgeous-Alexander	0.0089525
SVM	0.0324897	Dominique Wilkins	0.0084857	Bam Adebayo	0.0064649
SVM	0.0324897	Charles Oakley	0.0066425	Brandon Clarke	0.0063947

The RMSEs are similar across all 4 models. Stockton is the only player who appears in multiple top 3s for 1987, while Adebayo, Young & Dončić accomplish the same feat for 2021. There have been no repeat MIP winners, which does not bode well for the inclusion of Robertson (1986). Barkley & Wilkins received votes in the previous year, finishing second & fourth. On the 2021 side, every listed candidate save for Oubre, Jackson Jr. & Clarke has received votes in prior seasons.

Harden has already won an MVP award, and there's not much feasible improvement that can be done above that level. The aforementioned trio of players who appeared on multiple 2021 model top 3s have all been All-Star selections before. Brown made his first All-Star team in 2021, but it could be more than reasonably argued that his leap happened in the season prior. SGA was given the keys to the offense of the tanking Thunder, but he was shut down for the second half of the season. Clarke & Jackson Jr. are Memphis teammates, with JJJ missing almost the entire season after meniscus surgery and Clarke failing to take advantage of that absence by seizing a starting lineup spot. Oubre was traded to the Warriors, with the thinking being that he would benefit from Steph Curry's gravity pulling away defenders, but he opened the season in a brutal shooting slump.

Stockton's underlying stats primed him for a breakout, but it wasn't until 1988 that he started the majority of games for the Utah Jazz and received MIP votes. This will be a recurring caveat, since we can't predict opportunity/playing time. Rivers did improve significantly in advanced stats, but his points per game moved up marginally. He helped the team through playmaking and defense. Lister is the first candidate we've seen that received extra opportunity. While he increased his per game statistics due to being given 8 more minutes of playing time per game, his rate statistics dropped.

Results

We’ve trained our models, and now it’s time to test them on the 2021 data.

2021 Winner

We won’t use the linear or SVM model to generate predictions, as their predictions were too low (and in the linear case, didn’t meet expectations of Dale Ellis finishing first).

player	knn_p	ranger_p	mean
Jerami Grant	0.1579969	0.1043269	0.1311619
Stephen Curry	0.0380930	0.1531875	0.0956403
Khyri Thomas	0.0529103	0.1110552	0.0819828
Michael Porter Jr.	0.0516106	0.0723733	0.0619920
Kevin Porter Jr.	0.0565003	0.0630083	0.0597543

Grant has the highest prediction under the KNN model as well as the highest mean prediction between the two models. Curry receives the highest vote share under the ranger model, which is a cause for concern as he’s already won 2 MVP trophies. The other problem with his candidacy is exactly the same as that of Michael Jordan: he only played 5 games last year due to injury. Khyri Thomas, with the second-highest share in the ranger model, played 8 games last year and only 5 games this year for a tanking Rockets team. The Porters (no relation) both made strides in their second year in the league.

Let’s now take a look at the actual results.

player	share	knn_p	ranger_p	mean
Julius Randle	0.986	0.0503808	0.0391402	0.0447605
Jerami Grant	0.280	0.1579969	0.1043269	0.1311619
Michael Porter Jr.	0.276	0.0516106	0.0723733	0.0619920
Christian Wood	0.088	0.0021850	0.0371984	0.0196917
Zach LaVine	0.040	0.0014786	0.0010797	0.0012792
Chris Boucher	0.020	0.0018908	0.0116686	0.0067797
Jaylen Brown	0.020	0.0193955	0.0105430	0.0149692
Mikal Bridges	0.016	0.0100035	0.0120575	0.0110305
Zion Williamson	0.012	0.0126874	0.0264801	0.0195837
Clint Capela	0.006	0.0003012	0.0081963	0.0042488
Darius Garland	0.006	0.0087302	0.0282922	0.0185112
Jordan Clarkson	0.006	0.0010037	0.0019250	0.0014643
Jordan Poole	0.006	0.0130107	0.0221063	0.0175585
Luguentz Dort	0.006	0.0133381	0.0297190	0.0215286
Nikola Vučević	0.006	0.0001666	0.0046214	0.0023940
Kyle Anderson	0.004	0.0439481	0.0376396	0.0407938
Miles Bridges	0.004	0.0010789	0.0004051	0.0007420
RJ Barrett	0.004	0.0039513	0.0020668	0.0030090
Andrew Wiggins	0.002	0.0000516	0.0000040	0.0000278
Bojan Bogdanović	0.002	0.0000000	0.0000030	0.0000015
Lonzo Ball	0.002	0.0001652	0.0005754	0.0003703
Richaun Holmes	0.002	0.0000737	0.0004914	0.0002826
Shai Gilgeous-Alexander	0.002	0.0079800	0.0268802	0.0174301
T.J. McConnell	0.002	0.0003169	0.0027389	0.0015279
Terry Rozier	0.002	0.0004794	0.0008099	0.0006447

Randle, the actual winner, didn’t place in the top 5 of predictions, which is not ideal to say the least. Small consolation in Grant & MPJ finishing 2nd and 3rd, but not a great showing overall.

2022 Candidates

With the RMSEs being closer, we'll use all four models.

Model	**player**	**experience**	**Predicted Vote Share**
Linear	Jaren Jackson Jr.	3	0.0139608
Linear	Elijah Bryant	1	0.0135664
Linear	LaMelo Ball	1	0.0129919
Linear	Anthony Edwards	1	0.0121777
Linear	Chris Boucher	4	0.0120494
KNN	Xavier Tillman Sr.	1	0.0874916
KNN	Andrew Wiggins	7	0.0635783
KNN	Kevin Porter Jr.	2	0.0482502
KNN	Brandon Clarke	2	0.0471425
KNN	Tyrese Haliburton	1	0.0456044
Ranger	Elijah Bryant	1	0.0475808
Ranger	Robert Williams	3	0.0469542
Ranger	Mikal Bridges	3	0.0432811
Ranger	Jaren Jackson Jr.	3	0.0407263
Ranger	Luka Dončić	3	0.0406589
SVM	Kevin Porter Jr.	2	0.0072510
SVM	Tyrese Haliburton	1	0.0068021
SVM	Bogdan Bogdanović	4	0.0064650
SVM	Jakob Poeltl	5	0.0062204
SVM	Brandon Clarke	2	0.0059940

player	mean
Xavier Tillman Sr.	0.0311095
Kevin Porter Jr.	0.0262286
Andrew Wiggins	0.0210931
Elijah Bryant	0.0199203
Tyrese Haliburton	0.0196798

Save for Andrew Wiggins and Jakob Poeltl, every player in any of the four top 5s has a maximum of 4 years of experience. Jackson & Clarke show up once again, and their teammate Xavier Tillman Sr makes an appearance. If all three improve, there might be a case of vote-splitting. Bryant played only one game, which was the last game of the season for the Milwaukee Bucks as they rested their main players to gear up for the playoffs. Ball, Edwards & Haliburton were the top 3 players in Rookie of the Year voting, so improvement is kind of "expected" for them.

Williams & Bridges are young players that excel in their roles, but it's somewhat difficult to project where they can get those extra shot attempts to boost their scoring in Boston & Phoenix respectively. They're at best the 4th scoring options on their teams: Devin Booker, Chris Paul & Deandre Ayton are at the top of the pecking order for the Suns, while Jayson Tatum, Jaylen Brown and Dennis Schroder figure to soak up a majority of touches for the Celtics. Bogdanovic is in a similar situation in Atlanta, with arguably an even deeper roster and therefore a more muddled hierarchy.

The best bet in my opinion is Kevin Porter Jr. After being traded mid-season from the Cavaliers to the Rockets, he was given the keys to the offense. With a full offseason under his belt and the Rockets still in rebuild mode, there's not much shot creating competition apart from rookie Jalen Green.

Conclusion

Using a database of individual player seasons from 1984 to 2021, we set out to predict both the 2021 Most Improved Player award winner, as well as MIP candidates for 2022. Using the mean of a knn model & a ranger random forest that were trained on statistical jumps data (current season statistics minus previous season statistics), **Jerami Grant** was projected to be the 2021 winner. Rounding out the top 5 are Stephen Curry, Khyri Thomas, Michael Porter Jr and Kevin Porter Jr. The actual winner was Julius Randle, while Grant & MPJ finished second and third. Using the mean and median for four models (the previously mentioned two as well as linear & SVM) trained on regular season data, the top 5 candidates for 2022 are Xavier Tillman Sr, Kevin Porter Jr, Andrew Wiggins, Elijah Bryant and Tyrese Haliburton.

Both models suffer from the inability to quantify intangibles such as narrative and popularity. Julius Randle plays in New York, which is the NBA's biggest media market. While I'd hoped using the games played percentage would be able to somewhat help in distinguishing playing too few games due to injury or genuinely bad play, Steph Curry still finished second in projected voting despite already claiming two Most Valuable Player awards earlier in his career. The 2021 candidates model is limited by the fact that we can't predict opportunity.

By nature, the per game & per 100 possessions statistics are very correlated. I didn't remove highly correlated variables by design, as I wanted the models themselves to perform feature selection and determine what the most important variables were.

Further improvements could be only focusing on seasons after 2003, which is when the voting system changed from only first-place votes to three-place ballots. In addition, it might be better (if possible) to compare the current season to a player's career up to that point rather than just the previous season. This would immensely help with the injury-or-bad-player conundrum.