NBA Free Agency Contract Prediction 2022

Sumitro Datta

June 23, 2022

Introduction

This is my third year attempting to predict NBA free agent contracts using machine learning. What ended up happening last year?

- **Kawhi Leonard** & **Chris Paul** both opted out of their player options & re-signed with their teams.
  - Kawhi signed a 4-year, `$176M` contract with the Los Angeles Clippers while CP3 agreed to a 4-year `$120M` deal with the Phoenix Suns
- Restricted free agents **John Collins** and **Jarrett Allen** cashed in, signing 5-year deals with their current teams.
  - Collins' deal with the Atlanta Hawks is worth `$125M`, while Allen got `$100M` from the Cleveland Cavaliers
- Restricted free agent **Lonzo Ball** and unrestricted free agents **DeMar DeRozan** & **Kyle Lowry** all switched teams via sign-and-trade
  - Lonzo (4-year,`$80M`) & DeRozan (3-year,`$85M`) joined forces on the Chicago Bulls with shooting guard Zach LaVine and center Nikola Vucevic. The two were former opponents in the Southwest division, with Lonzo a former member of the New Orleans Pelicans and DeRozan being a San Antonio Spur
  - Lowry joined up with Jimmy Butler & the Miami Heat on a 3-year,`$85M` contract (coincidentally the same contract as his best friend DeRozan) after 9 seasons with the Toronto Raptors. That Raptors tenure included 6 All-Star appearances, franchise records in 3-point field goals made, assists and steals, and of course an NBA championship in 2019.

The top of this year's class is headlined by 4 star guards holding player options, meaning they may not even hit the market if they pick their options up. **James Harden** & **Kyrie Irving** started the season together on the Brooklyn Nets along w/Kevin Durant as one of the presumptive championship favorites. However, issues plagued the duo. Due to his refusal to being vaccinated, Kyrie was not allowed to play in home games by New York City's mayor's mandate and ended up playing a total of 27 games. Harden struggled with conditioning & lack of explosiveness, and was traded at the deadline to the Philadelphia 76ers, teaming up with MVP candidate center Joel Embiid as well as his former Houston Rockets GM Daryl Morey. **Bradley Beal** of the Washington Wizards shot a career-low 30% from the three-point line, and his season ended in February after only 40 games due to required surgery on his left wrist. **Russell Westbrook** had a tumultuous season after being traded from the Wizards to the Los Angeles Lakers, attempting to adjust his ball-dominant playstyle alongside LeBron James. He faced numerous calls to be moved out of the starting five.

In terms of restricted free agents, the top players are **Deandre Ayton, Miles Bridges** & **Anfernee Simons**. Ayton was the man in the middle for the Phoenix Suns, serving a nice complement to the perimeter games of guards Chris Paul & Devin Booker. But after failing to reach a contract extension before the season and the Suns being handed an embarrassing home-court loss in Game 7 of the Western Conference Finals against the Dallas Mavericks, the former No. 1 overall pick is more likely to start next season in a different jersey. Bridges was given more opportunity, starting all 80 games he played for the Charlotte Hornets and leading the team in scoring at 20.2 points per game. Simons continues the great lineage of Portland Trail Blazers guards. He averaged 17 points per game in 57 games and 40% shooting from the 3-point line on almost 8 attempts a game, in a season where the Blazers star point guard Damian Lillard was limited to 29 games due to an abdominal injury and shooting guard CJ McCollum was traded to the New Orleans Pelicans at the trade deadline.

On the unrestricted free agent side, we've got **Zach LaVine, Jalen Brunson** & **Mitchell Robinson** topping the lists. LaVine finally made the playoffs last year for the first time in his eight-season career, with big help coming from free-agent reinforcements Ball, DeRozan & Alex Caruso. Now, the question is if LaVine believes in what VP of Basketball Ops Arturas Karnisovas & GM Marc Eversley are building, or is he looking for greener pastures? Brunson had a career year, moving into the Dallas Mavericks starting lineup as a secondary playmaker alongside superstar Luka Doncic. He averaged 16.3 points per game and 4.8 assists per game in the regular season but shone brighter in the postseason, averaging 21.6 PPG in the Mavs' run to the Western Conference Finals. Robinson has spent the past 4 years calling Madison Square Garden home, and set the season record for field goal percentage in 2020 with 74.2%. How much is a rim-running, shot-blocking, athletic freak worth these days to NBA teams?

What I wanted to do was predict what contracts this year's free agent class might get based off previous offseasons. Stars generally get star-type money, but in tiers below, contracts of comparable players usually come up in discussing contract value.

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 cleaning variable names
if(!require(janitor))
  install.packages("janitor", repos = "http://cran.us.r-project.org")
#for multinomial regression
if(!require(glmnet))
  install.packages("glmnet", 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")
#kknn is nearest neighbors algorithm wrapper
if(!require(kknn))
  install.packages("kknn", repos = "http://cran.us.r-project.org")
#kernlab is svm kernel wrapper
if(!require(kernlab))
  install.packages("kernlab", repos = "http://cran.us.r-project.org")
#for variable importance
if(!require(vip))
  install.packages("vip", repos = "http://cran.us.r-project.org")
#zoo allows rolling operations
if(!require(zoo))
  install.packages("zoo", repos = "http://cran.us.r-project.org")
#matrix stats
if(!require(matrixStats))
  install.packages("matrixStats", repos = "http://cran.us.r-project.org")
#rpart.plot shows the decision tree of an rpart result
if(!require(rpart.plot))
  install.packages("rpart.plot", repos = "http://cran.us.r-project.org")
#kableExtra allows more customization of tables
if(!require(kableExtra))
  install.packages("kableExtra", repos = "http://cran.us.r-project.org")
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")
#easy formatting of numbers as currency & percent
if(!require(formattable))
  install.packages("ggdark", repos = "http://cran.us.r-project.org")
```

Importing the Data

For the statistical data, I've scraped total and advanced stats from Basketball-Reference and stored them in .csv files. This was actually part of a larger project to scrape complete statistics for teams, players and awards (the Kaggle dataset resides here). To my knowledge, my dataset is unique in that it includes BAA stats and ABA stats, which is not really of use here.

The advanced stats I kept were cumulative (offensive win shares, defensive win shares and value over replacement player). For players who played on multiple teams in one season, I kept their total stats and discarded the team-specific season portions. There was an initial desire to use totals to bake in availability/body fragility, but the shortened seasons would cause the model to declare all players to be fragile and underestimate their contract.

In the first iteration of this project, we scaled games played and games started to a normal distribution due to fluctuations in games played between seasons caused by the COVID-19 pandemic. 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.

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

advanced<-read_csv("Data/Advanced.csv",col_types = cols_for_stats) %>%
  select(seas_id:mp,ows:ws,vorp) %>% mutate(ws_per_48=ws/mp*48,.before="vorp")
totals<-read_csv("Data/Player Totals.csv",col_types = cols_for_stats)
#max games per season for players on multiple teams
max_games_tots=totals %>% filter(tm=="TOT") %>% group_by(season,lg,tm) %>%
  summarize(max_games_tot=max(g,na.rm = TRUE)) %>% ungroup()
#max games per season for players on single team
max_games=totals %>% filter(tm !="TOT") %>% group_by(season,lg) %>%
  summarize(max_games_non_tot=max(g,na.rm = TRUE)) %>% ungroup()
#coalesce above two into one column in totals df
totals_enhanced=left_join(totals,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))
advanced_and_totals<-left_join(totals_enhanced,advanced) %>%
  #if player played for multiple teams in season, only take total row
  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(g_percent=g/max_games_playable,gs_percent=gs/g,.before=g) %>%
  select(-c(gs,max_games_playable)) %>%
  #filter for only last ten years for faster pre-processing
  filter(season > 2009) %>% ungroup()
```

A comment was raised last year regarding positional scarcity. Teams will overpay for the potential piece that puts them "over the hump", whether that be into playoff contention or the more loftier goal of championship contention. Teams might also panic to acquire a player that is deemed to be the last one in a talent tier above the remaining free agents in the same position.

The play-by-play data (available since the 1997 season) keeps track of the percentage of minutes played a player has played in each traditional position (point guard, shooting guard, small forward, power forward & center). I converted the percentages to raw minutes played at each position, and summed across all teams a player played for in the same season. I felt this method was more accurate than using the "totals" row.

```r
play_by_play<-read_csv("Data/Player Play By Play.csv") %>%
  filter(tm!="TOT") %>%
  select(seas_id:player,mp:c_percent)

#replace NA's with zeroes
play_by_play[is.na(play_by_play)] <- 0

pbp_pos_mins=play_by_play %>%
  #convert percents to minutes played at position
  mutate(across(pg_percent:c_percent,~./100*mp)) %>%
  rename_with(.fn=~gsub(x = ., pattern = "_percent", replacement = "_mp"),
              .cols=pg_percent:c_percent) %>%
  #sum season minutes across different teams
  group_by(season,player_id) %>%
  mutate(across(mp:c_mp,sum,.names="{col}_summed")) %>%
  slice(1) %>% ungroup() %>% select(-c(mp:c_mp))
```

I've updated the free agents training set from last year to include the 2021 free agents. As a quick refresh:

- scrape the Basketball-Reference free agents tracker
- removed retired players and players who signed from overseas (wouldn't have any contract year data) from the dataset
- set contract years to zero and salary to zero for players who:
  - went overseas (not many, considering the pandemic)
  - had explicitly non guaranteed first years in their contracts (training camp deals, two ways, ten days, exhibit 10s)
  - had blanks in their contract terms cell
- included option years and partially guaranteed years in my calculation of contract years
  - looked at it as both sides (player and team) intending to see out the contract
- gathered year 1 salary for remaining players using Spotrac
  - I'd previously used Capology, Basketball-Reference & Basketball-Insiders, but they were not needed since I got started with the data-gathering before/during the season rather than after the season
- general housekeeping like adding suffixes (Jr., II, III) to certain players to match up with statistical data

```r
past_free_agents<-read_csv("Data/2016-2021 Free Agents.csv")
```

The next file I used was salary cap history, scraped from RealGM. To somewhat normalize comparisons across years, I converted the first year salary to a percentage of the salary cap.

```
#subtract one from year to match up with offseason in which contract was signed
salary_cap_hist<-read_csv("Data/Salary Cap History.csv") %>% mutate(season=season-1)
#create variable of first year salary as percentage of cap
#easier to compare across years
past_free_agents<-past_free_agents %>% select(-c(terms,Source)) %>%
  left_join(.,salary_cap_hist) %>%
  mutate(first_year_percent_of_cap=yr_1_salary/cap) %>%
  select(-c(yr_1_salary,cap))
```

The last file loaded is our evaluation set: the 2022 free agent class, retrieved from Spotrac. I had to edit this dataset to match the Basketball-Reference names (mainly adding diacritics to European names). In addition, I filtered out players with options. Players who decline player options and players who have their team options declined with more than 3 years of experience become unrestricted free agents. Players with less than or equal to 3 years of experience and a declined team option become restricted free agents. I'll use this fact to see which players & teams might decline their option.

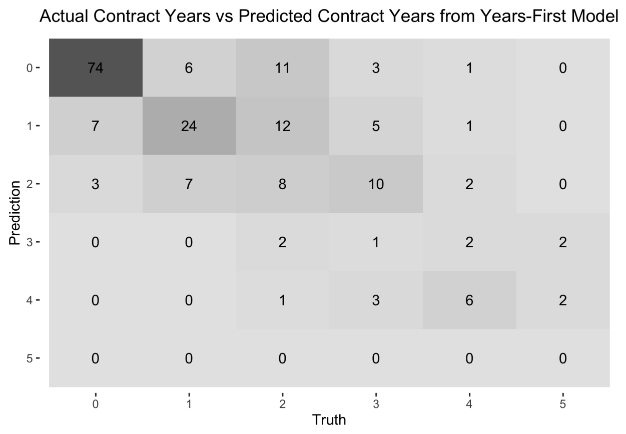
```
current_fa<-read_csv("Data/Free Agents 2022.csv")
#separate out options to compare what players options get if declined
current_fa_options<-current_fa %>% filter(str_detect(type,"PO|CO")) %>%
  select(-c(experience,contract_yrs)) %>%
  rename(option_type=type,option_amt=first_year_percent_of_cap) %>%
  mutate(option_amt=currency(option_amt,digits=0))
#make player options all declined (UFA's)
#make club options ufa or rfa depending on exp
current_fa<-current_fa %>%
  mutate(type=case_when((type=="PO"|(type=="CO" & experience >= 4))~"UFA",
                        (type=="CO" & experience < 4)~"RFA",
                        TRUE~type)) %>%
  group_by(player) %>% select(-experience) %>% slice(1) %>% ungroup() %>%
  mutate(first_year_percent_of_cap=NA)
```

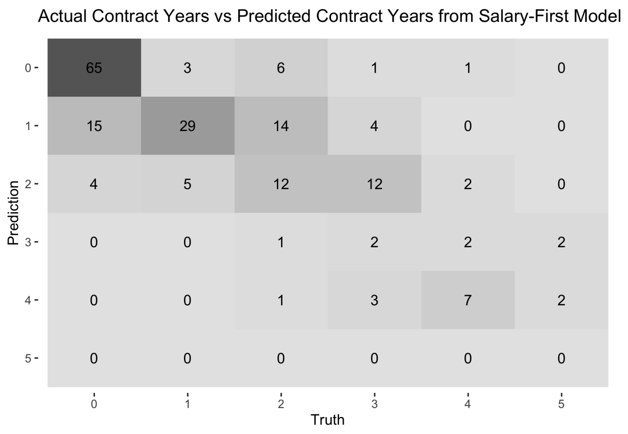
In the GitHub repository where this project is located, a file called `free agents.r` has more details on exactly how I scraped the train set, evaluation set and the salary cap history.

Retrospective on Last Year's Results

Before getting into pre-processing, we'll take a look at last year's results and see how the models performed. We remove players who ended up not hitting the market due to their option being picked up. Any player who was predicted but didn't end up in the free agent history must not have received a contract, so their NA's were replaced with zeroes.

The years accuracy of the years-first model was 58.55%, while the years accuracy of the salary-first model was 59.59%. The 2020 models were at 49-51% accuracy, so there's been a significant improvement. Here's some confusion matrices on how each model handled the prediction of contract years.





The incorrect predictions were pessimistic, in that they skewed toward predicting less years than received as evidenced by the sum of the upper triangle being greater than the lower triangle (models forecasting no contract for players who received a one-year contract, one year for players who got two years, etc).

Here are the worst misses for both models.

player	yrs_y1s2	contract_yrs
Didi Louzada	0	4
Tim Hardaway Jr.	1	4
Jordan McLaughlin	0	3
David Nwaba	0	3
Dante Exum	0	3

player	yrs_s1y2	contract_yrs
Didi Louzada	0	4
Dante Exum	0	3
John Collins	3	5
Jarrett Allen	3	5
Richaun Holmes	2	4
Alex Caruso	2	4
Alec Burks	1	3
Spencer Dinwiddie	1	3
Garrett Temple	1	3
David Nwaba	1	3
Kendrick Nunn	4	2
Boban Marjanović	0	2
Gabe Vincent	0	2
Keita Bates-Diop	0	2
Max Strus	0	2
Thanasis Antetokounmpo	0	2
Trey Lyles	0	2
Garrison Mathews	2	0
Dennis Smith Jr.	2	0
Edmond Sumner	2	0
Armoni Brooks	2	0

After being drafted in 2019 as a "draft-and-stash" prospect, Louzada spent two years overseas with the Sydney Kings in the National Basketball League in Australia. He finally came over in 2021, but only played 3 games. New Orleans must have seen something they liked, and signed him to a 4-year, near minimum contract. Collins & Allen ended up staying with their current teams & getting full 5-year contracts. THJ had wildly different predictions between the two models last year, with a staggering total discrepancy of `$52M`, also attributing to a difference in contract years. Exum played only 6 games last year after straining his right calf, but Houston Rockets GM Rafael Stone got creative in re-signing Exum to a low-risk, high-reward contract ‘that would be three years, nonguaranteed and heavy on incentives that fall under “likely bonuses” definition’.

Let’s shift our focus to the salary predictions. First, the residual mean squared error of the years-first model was 0.0245718, while the salary-first model had an RMSE of 0.0248204.

As we did with the years models, let’s look at the most extreme salary misses.

player	y1s2_cap_percent	first_year_percent_of_cap
Spencer Dinwiddie	2.96%	15.25%
Tim Hardaway Jr.	9.58%	18.95%
Kyle Lowry	15.60%	24.00%
Kendrick Nunn	12.82%	4.45%
Gary Trent Jr.	6.73%	14.23%
Nicolas Batum	9.73%	2.82%
Will Barton	7.50%	13.90%
Dennis Schröder	11.34%	5.24%
Victor Oladipo	7.81%	2.13%
Andre Drummond	7.54%	2.14%

player	s1y2_cap_percent	first_year_percent_of_cap
Spencer Dinwiddie	3.84%	15.25%
Dennis Schröder	13.58%	5.24%
Andre Drummond	10.29%	2.14%
Gary Trent Jr.	6.57%	14.23%
Nicolas Batum	10.24%	2.82%
Victor Oladipo	9.47%	2.13%
Will Barton	7.39%	13.90%
Kyle Lowry	17.71%	24.00%
Tim Hardaway Jr.	12.92%	18.95%
Lauri Markkanen	8.21%	13.96%

Schröder miscalculated his market, rejecting a 4-year & `$84M` contract extension during the season from the Lakers and ended up settling for a one-year, prove-it contract for the mid-level exception from the Boston Celtics. Dinwiddie only played Brooklyn’s first three games last season before suffering a partial ACL tear, but Washington felt he had recovered enough to give him 3 years & `$54M` in a sign-and-trade. After being bought out by the Cavaliers and a forgettable stint on the Lakers, Drummond went to Philadelphia to back up former adversary Joel Embiid. Batum rewarded the Clippers, who took a flyer on him last year, by taking a below-market deal (much like Carmelo Anthony did with the Trail Blazers last offseason). Trent Jr. was mentioned in last year’s project as someone who was being majorly hampered by the models being exceedingly retrospective, because he was “inefficient with minimal contributions in rebounds and assists as well as a negative VORP”.

On a more positive note, here's some players on which the models were very close on. We'll restrict our view to contracts that had a first year salary that was greater than 5% of the salary cap, as it's easier to get close to minimum & near-minimum contract amounts.

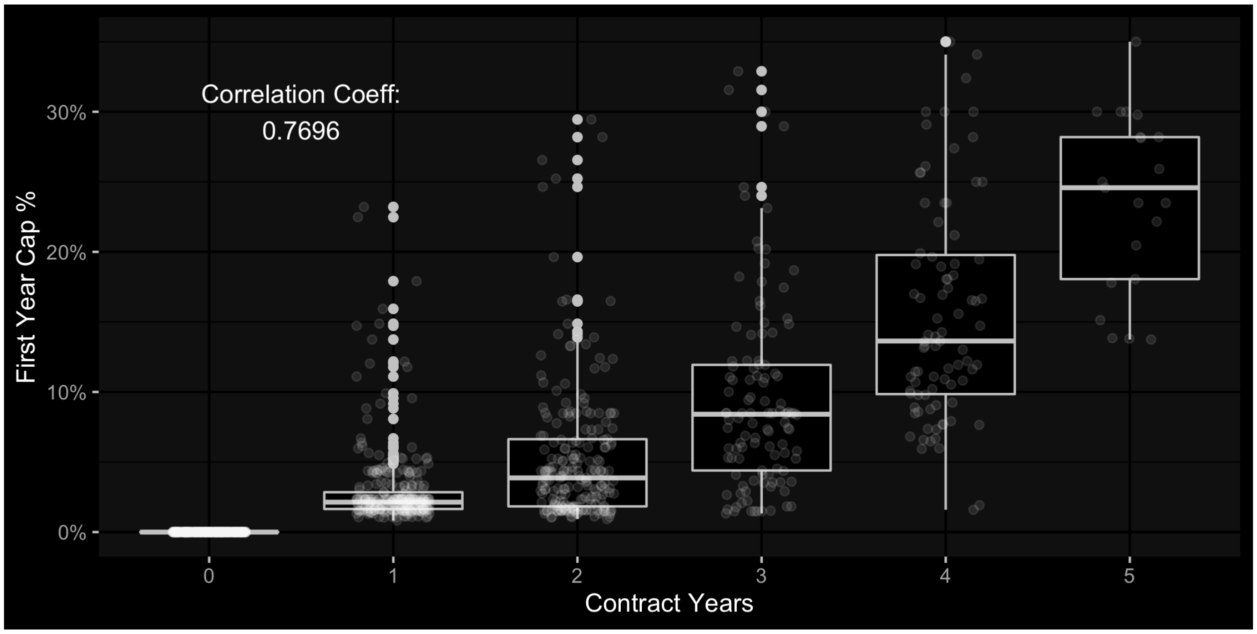
player	y1s2_cap_percent	first_year_percent_of_cap
Evan Fournier	15.35%	15.25%
Cameron Payne	5.61%	5.78%
Richaun Holmes	9.44%	9.24%
DeMar DeRozan	22.80%	23.13%
Chris Paul	26.71%	27.40%
Khem Birch	4.65%	5.65%
Lonzo Ball	15.47%	16.55%
Kelly Oubre Jr.	9.41%	10.67%
Patty Mills	3.81%	5.24%
Nerlens Noel	6.14%	7.83%

player	s1y2_cap_percent	first_year_percent_of_cap
Kelly Oubre Jr.	10.71%	10.67%
Patty Mills	5.05%	5.24%
Daniel Theis	7.91%	7.37%
Nerlens Noel	8.39%	7.83%
Mike Conley	18.01%	18.68%
Khem Birch	4.92%	5.65%
Cameron Payne	6.54%	5.78%
Alec Burks	7.60%	8.48%
Chris Paul	26.50%	27.40%
Devonte' Graham	10.76%	9.79%

Both Phoenix Suns point guards with the initials CP show up on both hit-lists. Other players that were near-spot on by both models include Brooklyn Nets point guard Patty Mills, Charlotte Hornets wing Kelly Oubre Jr, New York Knicks big Nerlens Noel and Toronto Raptors big Khem Birch.

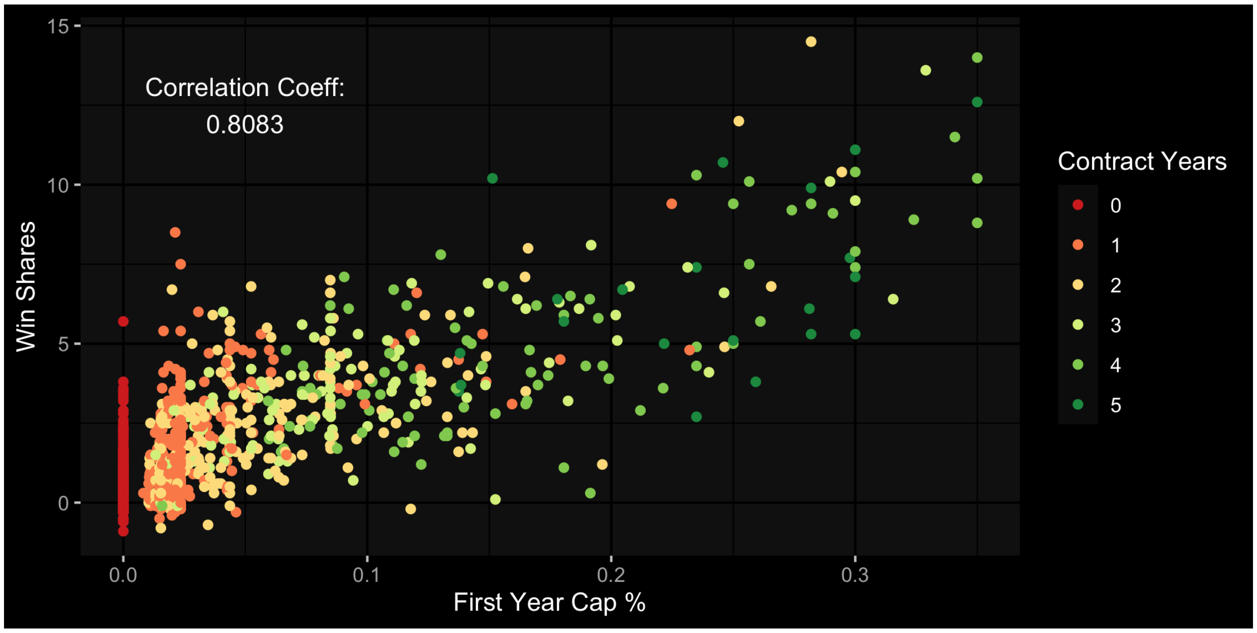
Data Exploration and Visualizations

Before we train models, let's see how some of the predictors and some of the targets interact. First, let's see how our targets correlate with each other. I've created a box and whisker plot, as well as added the points themselves in a transparent layer with some random variation to differentiate between points.



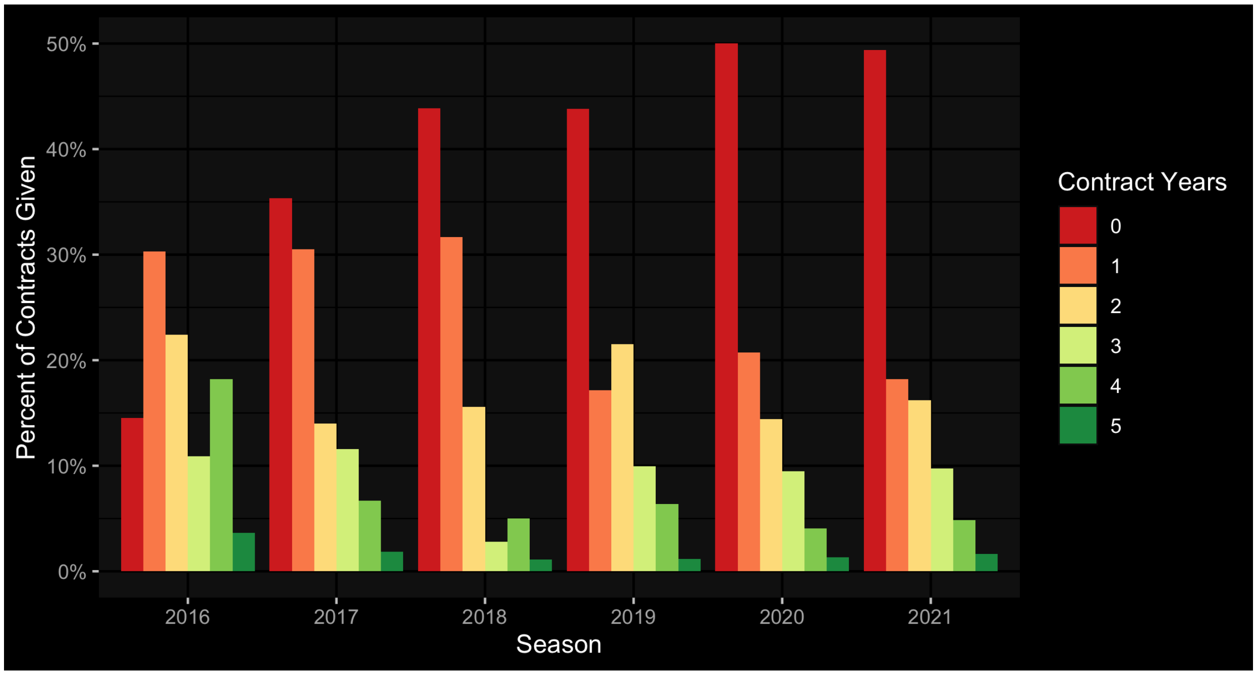
The correlation coefficient is 76.96%, which shows that the two targets are strongly and positively correlated. The median value of the first year cap % (middle line in each box) increases with an increase in contract length. The highest increase in first year cap % is between a 4-year and 5-year contract.

Next, let's see if an all-encompassing advanced statistic has a relationship with first year cap percentage. Win Shares represent how much a player has contributed to his team's wins by comparing his output to a marginal player. Higher win shares generally indicate a better player.



With a correlation coefficient of 80.83%, win shares are highly correlated with first year cap percentage. This shouldn't be too groundbreaking: better players get paid more.

Finally, let's see how many contracts of each length were given in each offseason.



It comes as no surprise that as contract length increases, the percent of contracts of that length given out decreases. Although in 2016, the amount of players who didn't receive contracts was lower than the amount who received 1, 2, or 4 year contracts. 2016 was the year of the cap spike, when the salary cap jumped from $ 70 million to $ 94 million. Similar to a lot of people with new-found money, teams spent somewhat recklessly. As it was also the first year of record with data, perhaps the number of true free agents was underestimated.

Pre-Processing

Pre-Processing Box Score & Advanced Stats

I used regular season stats, although I do understand that some players get paid on the strength of playoff performance. I started off with contract year stats, because there's anecdotal evidence that players exert more effort in their contract year (*cough cough Hassan Whiteside*). All stats except for the advanced stats (OWS, DWS and VORP) were converted to per game. Percentages were left alone.

In addition to using contract year stats, I summed the past two years and the contract year.

Why I settled on 3 years:

- Players do get paid on past performance, so just using contract year stats was out of the question
- 2 years opens up the possibility of a fluke year
  - Kawhi would have his nine game 2018 season bring down his averages significantly from his 2019 season with the Raptors: adding another year somewhat lessens this effect
- On the other hand, it's quite unlikely that teams factor in stats from more than 4 years ago, a lot would have changed

  - the Trail Blazers didn't pay Carmelo to recapture his form of his 2013 year where he led the league in scoring (I would hope)
- Another reason I settled on 3 years is that I can keep the same model for restricted free agents
  - my thought is that the rookie year is a bonus: great if you did well, but doesn't matter in the grand scheme of things if you did poorly
  - rookie extension is more based on how you improved over the course of that initial contract
  - For example, if Luka Dončić had a worse rookie year but had the same level of play that he has achieved in his second and third year (as well as next year), I highly doubt that Dallas would have offered him a significantly less amount of money due to that substandard rookie year

I performed the same processing on the three-year totals, using the three-year game total as the denominator for converting to per game. I had to calculate the three-year percentages, and also re-engineered the win shares per 48 minutes metric.

I removed categories that were linear combinations of one another. For example, total rebounds can be found by simply adding up offensive and defensive rebounds, and points are just the result of 2*(number of 2 point field goals made) & 3*(number of 3 point field goals made). I kept age and experience as predictor variables, but removed position because I felt it would ultimately reflect in the stats themselves.

The final step was to replace missing values in the shooting percentages with zeroes. These NA's were originally due to lack of attempts.

```
three_year_rolling_stats=advanced_and_totals %>% group_by(player_id) %>%
  #three year sum
  mutate(across(-c(1:9,fg_percent,x3p_percent,
                  x2p_percent:e_fg_percent,ft_percent),
                list(three_yrs=~rollapplyr(.,3,sum,partial=TRUE)),
                .names="{col}_last_3_yrs")) %>%
  mutate(ws_per_48_last_3_yrs=ws_last_3_yrs/mp_last_3_yrs*48) %>%
  mutate(fg_percent=ifelse(fga==0,0,fg/fga),
         x3p_percent=ifelse(x3pa==0,0,x3p/x3pa),
         x2p_percent=ifelse(x2pa==0,0,x2p/x2pa),
         e_fg_percent=ifelse(fga==0,0,(fg+0.5*x3p)/fga),
         ft_percent=ifelse(fta==0,0,ft/fta)) %>%
  mutate(fg_percent_last_3_yrs=
           ifelse(fga_last_3_yrs==0,0,fg_last_3_yrs/fga_last_3_yrs),
         x3p_percent_last_3_yrs=
           ifelse(x3pa_last_3_yrs==0,0,x3p_last_3_yrs/x3pa_last_3_yrs),
         x2p_percent_last_3_yrs=
           ifelse(x2pa_last_3_yrs==0,0,x2p_last_3_yrs/x2pa_last_3_yrs),
         e_fg_percent_last_3_yrs=
           ifelse(fga_last_3_yrs==0,0,
                  (fg_last_3_yrs+0.5*x3p_last_3_yrs)/fga_last_3_yrs),
         ft_percent_last_3_yrs=
           ifelse(fta_last_3_yrs==0,0,ft_last_3_yrs/fta_last_3_yrs)) %>%
  #remove categories that aren't predictive vars or linear combo of others
  select(-c(lg,pos,birth_year,tm,
            trb,trb_last_3_yrs,
            fg,fga,fg_last_3_yrs,fga_last_3_yrs,
            pts,pts_last_3_yrs)) %>%
  #convert contract year and last 3 year stats to per game (except games)
  mutate(across(c(mp,x3p:x3pa,x2p:x2pa,ft:fta,orb:pf),list(per_game=~./g)),
         .after="gs_percent") %>%
  select(-c(g,mp,x3p:x3pa,x2p:x2pa,ft:fta,orb:pf,ws)) %>%
  mutate(across(mp_last_3_yrs:pf_last_3_yrs,list(per_game=~./g_last_3_yrs)),
         .after="gs_percent_last_3_yrs") %>%
  select(-c(g_last_3_yrs,mp_last_3_yrs:pf_last_3_yrs,ws_last_3_yrs)) %>%
  ungroup() %>%
  #rescale games percentages over 3 years back to 0-1
  mutate(across(g_percent_last_3_yrs:gs_percent_last_3_yrs,~./3)) %>%
  replace_na(list(fg_percent=0,x3p_percent=0,x2p_percent=0,
                  e_fg_percent=0,ft_percent=0))
```

Pre-Processing Positions

I took a three-year rolling sum of minutes played for consistency with the previous stats pre-processing, and converted the totals back to percents.

With 3-year positional percentages in hand, it was time to assign the actual positions. Some players play one position almost exclusively, while other players are more flexible and alternate between positions. I set the baseline for a player to be considered at a "pure position" at 75%: if the player played at any position more than 75% of the time, they were deemed to be that position.

All other players were bucketed into combo positions based on the maximum of the following sums of two traditional positions:

- combo guard (point guard/shooting guard)
- small wing (shooting guard/small forward)
- big wing (small forward/power forward)
- big man (power forward/center)

Some players had multiple combo positions listed due to a small amount of minutes played and two players had no positions listed due to playing less than a full minute, so those players were added manually.

```
pbp_last_three_years=pbp_pos_mins %>% group_by(player_id) %>%
  #rolling 3 year sum of minutes played total & position
  mutate(across(mp_summed:c_mp_summed,
                list(three_yrs=~rollapplyr(.,3,sum,partial=TRUE)),
                .names="{col}_last_3_yrs")) %>% ungroup() %>%
  select(-c(mp_summed:c_mp_summed)) %>%
  #convert totals back to percents
  mutate(across(pg_mp_summed_last_3_yrs:c_mp_summed_last_3_yrs,
                ~./mp_summed_last_3_yrs)) %>%
  rename_with(.fn=~gsub(x = ., pattern = "_mp", replacement = "_percent"),
              .cols=pg_mp_summed_last_3_yrs:c_mp_summed_last_3_yrs) %>%
  select(-mp_summed_last_3_yrs)

#assign pure positions to players with 75% of time at one position
pure_position=pbp_last_three_years %>%
  pivot_longer(.,cols=c(pg_percent_summed_last_3_yrs:c_percent_summed_last_3_yrs),
               names_to = "pos",values_to = "percent_at_pos") %>% group_by(seas_id) %>%
  slice_max(percent_at_pos) %>% ungroup() %>%
  filter(percent_at_pos>0.75) %>% mutate(pos=word(pos,1,sep="_"))

combo_position=
  #remove players with pure positions
  anti_join(pbp_last_three_years,pure_position %>% select(1:4)) %>%
  #create buckets of in-between positions
  mutate(combo_guard=pg_percent_summed_last_3_yrs+sg_percent_summed_last_3_yrs,
         small_wing=sg_percent_summed_last_3_yrs+sf_percent_summed_last_3_yrs,
         big_wing=sf_percent_summed_last_3_yrs+pf_percent_summed_last_3_yrs,
         big_man=pf_percent_summed_last_3_yrs+c_percent_summed_last_3_yrs) %>%
  select(-c(pg_percent_summed_last_3_yrs:c_percent_summed_last_3_yrs)) %>%
  pivot_longer(.,cols=c(combo_guard:big_man),
               names_to = "pos",values_to = "percent_at_pos") %>%
  group_by(seas_id) %>% slice_max(percent_at_pos) %>% ungroup() %>%
  #4 players had 2 separate combo positions listed
  filter(!(player=="Terrance Roberson" & pos=="big_wing") &
           !(player=="Vince Hunter" & pos=="small_wing") &
           !(player=="Ty Jerome" & season==2020 & pos=="small_wing") &
           !(player=="Anthony Gill" & season==2021 & pos=="big_man"))

all_player_pos=bind_rows(pure_position,combo_position) %>%
  #2 players had 0 MP, so they were lost in both combo & pure
  add_row(seas_id=19914,season=2006,player_id=3589,
          player="Alex Scales",pos="sg",percent_at_pos=1) %>%
  add_row(seas_id=22403,season=2010,player_id=3882,
          player="JamesOn Curry",pos="pg",percent_at_pos=1) %>%
  filter(season>2009) %>% select(-c(seas_id,percent_at_pos)) %>%
  mutate(pos_group=case_when(pos %in% c("pg","sg","combo_guard")~"guard",
                             pos %in% c("sf","small_wing","big_wing")~"wing",
                             pos %in% c("pf","c","big_man")~"big"))
```

Combining Pre-Processing Dataframes

I joined the positional data and grouped by season and positional group to get the percentage of VORP a player has contributed to their position as a proxy for positional scarcity. While a player may not have had a raw high VORP compared to all offseasons, they could have a significant proportion of their positional group's VORP in that specific offseason and possibly induce a bidding war between teams due to unappetizing other options.

I changed contract years from a numeric column to a factor/category column. This changes its prediction from a regression problem to a classification problem. A 2.5-year contract doesn't make much sense, so it is in our best interest to discretize the years and store them as factors rather than round a regression result. However, in the next section, we'll see a use case that'll be undoing this last step.

```
train_set=inner_join(three_year_rolling_stats,past_free_agents) %>%
  left_join(.,all_player_pos) %>% group_by(season,pos_group) %>%
  mutate(position_vorp=sum(vorp_last_3_yrs)) %>% ungroup() %>%
  mutate(percent_of_pos_vorp=vorp_last_3_yrs/position_vorp) %>%
  mutate(contract_yrs=factor(contract_yrs,levels = 0:5)) %>%
  select(-c(ws,pos,pos_group,position_vorp)) %>%
  mutate(across(-c(seas_id:experience,type:contract_yrs),~round(.,digits=4)))
write_csv(train_set,"Data/Train Set.csv")
```

Training Models

There is no need for a subset of the training set to be withheld as a test set before running the models on the evaluation set, because there is built-in cross validation. In the first iteration of this project, I utilized leave-one-out cross validation since the dataset is relatively small (<1000 observations). How this works is that the model is run excluding one observation. Then, the model attempts to predict the result of that excluded observation. This is repeated for every observation. However, on subsequent runs, I've decided to use k-fold cross validation. I've achieved even better results while cutting down significantly on training time.

As we saw in data visualization, the two target variables (contract years and first year salary as a percentage of the salary cap) are fairly well correlated, as they have a Pearson correlation coefficient of 0.77. The way I chose to handle this is:

- predict one target first without the other as a predictor
- choose the best model (be that a single model or an ensemble of multiple models)
- use the first target's predictions as an input to predict the second target

One potential problem is compounding errors. If there's an incorrect year prediction, it might lead to an incorrect salary prediction. Initially, to alleviate this problem, I thought it would be sufficient to utilize tidymodels' ability to run multivariate models with more than one outcome. However, I realized that tuning wasn't yet implemented with multi-output regression, and I'm uneasy about choosing arbitrary parameters. An additional (minor) drawback is the outcomes must be of the same type, so we would undo the contract years conversion done in the last section.

The Models

I used a total of six models.

- linear regression model as a baseline for salary, and multinomial regression as a baseline for years
  - the separation is due to the classification/regression split
- k-nearest neighbors model: take the distance between the statistics of two players (the absolute value of the difference) and then take the average of the outcome variable of the k nearest neighbours
  - the intuition being that similar players get similar contracts
- decision tree model (`rpart`): maybe as a player passes certain statistical thresholds, their contract increases
  - only using for predicting the contract years; since there are so many different salary percentages, a solitary decision tree would either be useless or far too complicated
- random forest model (`ranger`): reduces instability by averaging multiple trees
  - costs interpretability as there is no tree diagram that is representative of the decisions made
- support vector machine model: 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
  - I believe this image from Wikipedia succinctly explains an SVM

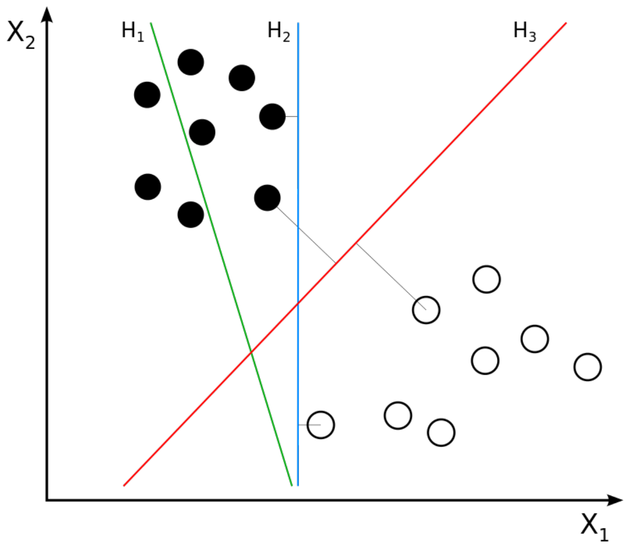


Figure 1: H1 does not separate the classes. H2 does, but only with a small margin. H3 separates them with the maximal margin. By User:ZackWeinberg, based on PNG version by User:Cyc - This file was derived from: Svm separating hyperplanes.png, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=22877598

Predicting Years First, then Salary

We'll start by predicting years first and then salary with years as an input.

With caret, everything could be thrown into the train function: cross-validation, tuning, data, etc. Tidymodels is more explicit in its steps. The standard 10-fold cross validation is how the data is going to be resampled. In addition, we'll stratify selection, so there is sufficient data to predict all contract year lengths. A hypothetical (albeit unlikely) scenario that could occur without stratification is a fold could include all instances of restricted free agents (which are more rare) in the test portion, and none in the train portion. The chosen algorithm would be confused as to how to predict something for which it has no training.

Recipes is tidymodels' pre-processing package. The first step to change the roles of some variables. The season_id, season, player_id & player variables are of no use as predictors, but are useful to identify observations, so they are given the "id" role. Since we are predicting years first, the first_year_percent_of_cap variable will be removed. Finally, we will convert the free agent type column to a numeric column.

```
set.seed(100)
cv=vfold_cv(train_set,v=10,strata=type,repeats=5)
y1_recipe=recipe(contract_yrs~.,data=train_set) %>%
  update_role(seas_id:player,new_role="id") %>%
  step_rm(first_year_percent_of_cap) %>% step_dummy(type)
```

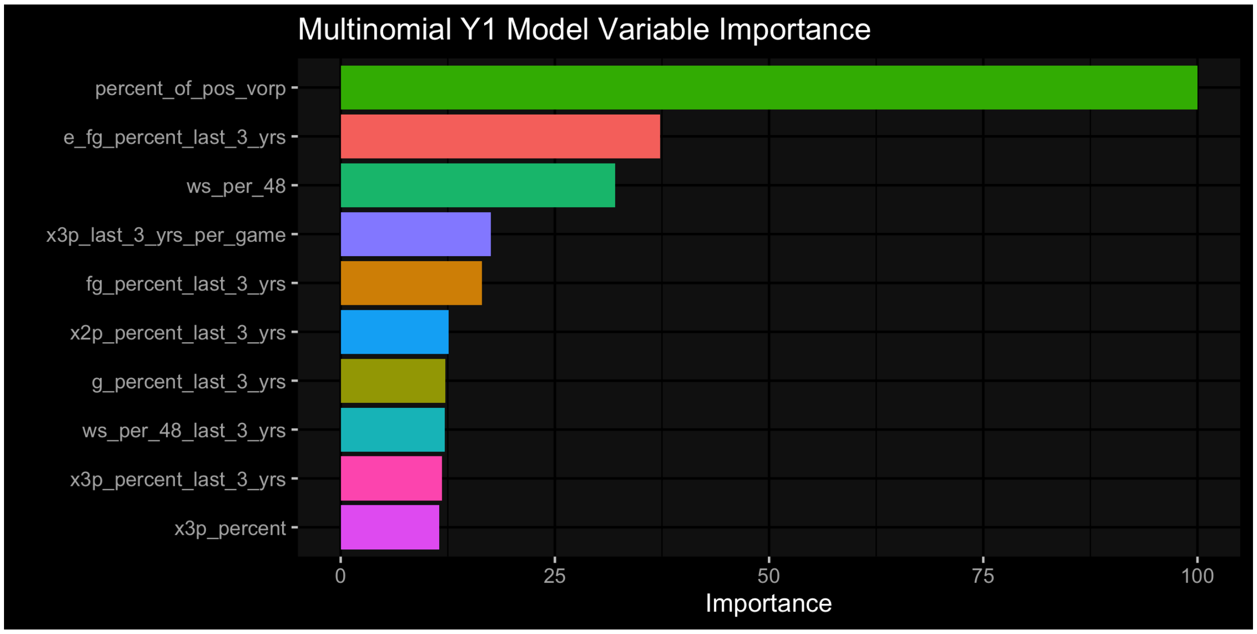
Initially, accuracy was chosen as the metric to determine the best submodel by cross-validation. However, with the inherent imbalance of the outcome classes, the F1 score is a better metric. As an extreme example, if there were only two classes with a 90:10 split, a classifier could achieve 90% accuracy by simply predicting the more populous class for every case. On the other hand, the F1 score attempts to minimize both false positives & false negatives. However, the default averaging for F1 is macro-averaging, which gives equal weights to all classes and is exactly the problem we were trying to distance ourselves from by not choosing accuracy. We need to change the averaging function to be macro-weighted. In order to include it within yardstick's metric_set, we have to create a new function wrapping the metric, set the options within the wrapper & formalize it as a new metric.

```r
macro_weight_f1 <- function(data,truth,estimate,beta = 1,estimator="macro_weighted",
                            na_rm = TRUE,
                            event_level = yardstick_event_level(),...){
  f_meas(data=data,
         truth = !! rlang::enquo(truth),
         estimate = !! rlang::enquo(estimate),
         estimator = "macro_weighted",
         beta=beta,
         na_rm=na_rm,
         event_level=event_level,
         ...
         )
}
macro_weight_f1 <- new_class_metric(macro_weight_f1,"maximize")
```

First up is the multinomial regression model.

```r
wf=workflow() %>% add_recipe(y1_recipe) %>%
  add_model(multinom_reg(penalty=0) %>%
              set_engine("glmnet") %>%
              set_mode("classification"))
tune_multinom=wf %>% tune_grid(resamples=cv,
                               metrics=metric_set(macro_weight_f1))
wf=wf %>% finalize_workflow(tune_multinom %>% select_best())
fit_y1_multinom=fit(wf,train_set)
```

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



The new percent of positional VORP variable is the most important variable, which is a good start.

Next up is the k-nearest neighbors model.

```r
wf=workflow() %>% add_recipe(y1_recipe) %>%
  add_model(nearest_neighbor(neighbors = tune()) %>%
  set_engine("kknn") %>% set_mode("classification"))
tune_knn=wf %>% tune_grid(resamples=cv,
                         grid=expand.grid(neighbors=5:50),
                         metrics=metric_set(macro_weight_f1))
wf=wf %>% finalize_workflow(tune_knn %>% select_best())
fit_y1_knn=fit(wf,train_set)
tune_knn %>% select_best() %>% knitr::kable()
```

neighbors	.config
50	Preprocessor1_Model46

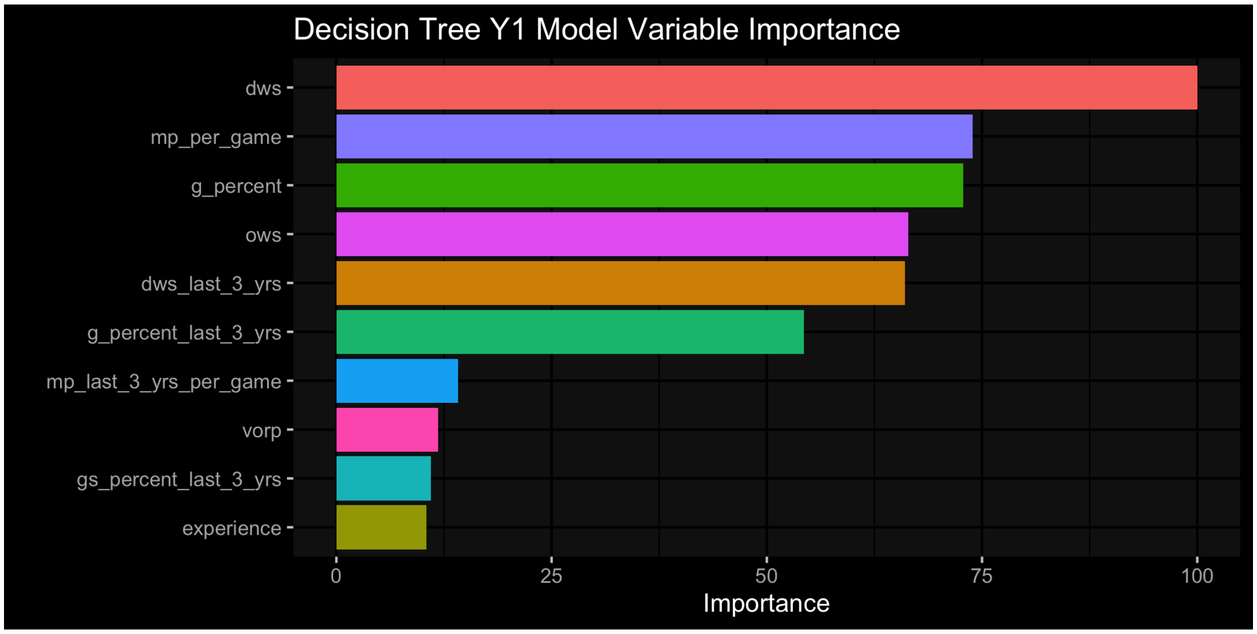
Looks like the best knn model is right at the upper limit of the tune grid with using 50 closest comparable players.

The next model is the decision tree model. We can tune the cost complexity parameter (cp) in this model. CP is the minimum for how much the residual sum of squares must improve for another partition to be added. A CP that is too high will have too few branches, while a CP that is too low will be difficult to follow since there are many branches. We lean towards the lower end of the spectrum. We will also change the min_n, which is the minimum number of data points in a node that are required for the node to be split further. The default is 20, but the 5-year portion of the dataset is small, so we will decrease min_n to 1. Since there exists an element of randomness in choosing samples to model a decision tree, we need to set a seed to keep the work reproducible.

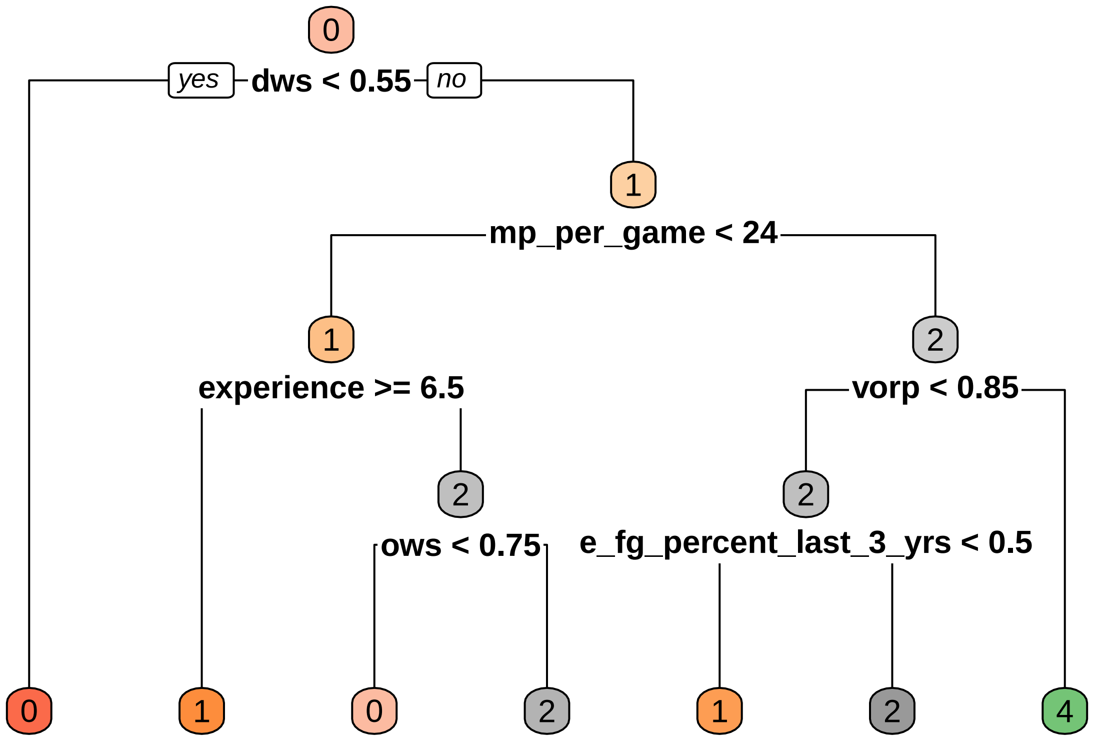
```r
set.seed(100,sample.kind = "Rounding")
wf=workflow() %>% add_recipe(y1_recipe) %>%
  add_model(decision_tree(cost_complexity = tune(),min_n=tune()) %>%
  set_engine("rpart") %>% set_mode("classification"))
tune_tree=wf %>% tune_grid(resamples=cv,
                         grid=expand.grid(cost_complexity=0.1^seq(2,5),
                                          min_n=seq(1:10)),
                         metrics=metric_set(macro_weight_f1))
wf=wf %>% finalize_workflow(tune_tree %>% select_best())
fit_y1_tree=fit(wf,train_set)
tune_tree %>% select_best() %>% knitr::kable()
```

cost_complexity	min_n	.config
0.01	1	Preprocessor1_Model01

Let’s get the most important variables of the decision_tree model.



The individual contract-year components of the holistic advanced stat of win shares rank within the top-4 most important variables. Durability and availabilty are also highly desirable characteristics, as evidenced by the high values of contract-year minutes per game, contract-year games played percentage and last-3-years games played percentage.



The decision tree does not predict any 3-year or 5-year contracts. The decision tree maximizes its prediction when a player does all of the following:

- has defensive win shares above 0.55 in the contract year

- plays more than 24 minutes per game in the contract year
- has a value over replacement player above 0.85

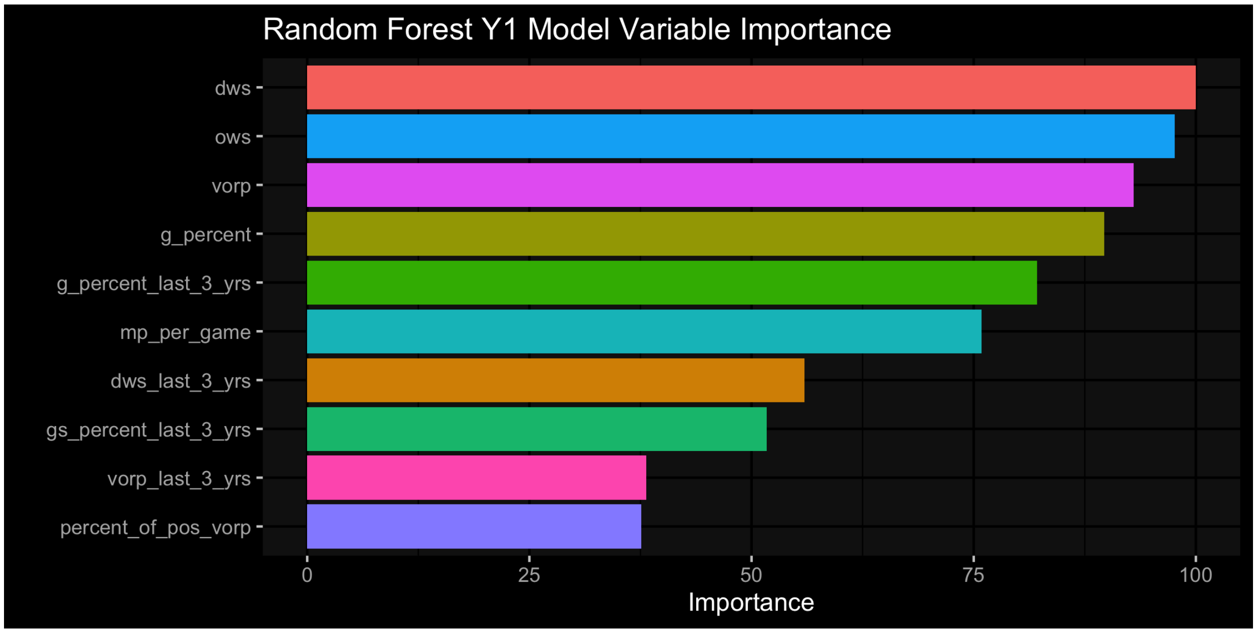
The next model is a random forest. Since `ranger` is a *random* forest algorithm, we need to set a seed to keep the work reproducible.

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(55))`
  - we will try all integers between 5 & 10
- min_n is the same as the decision tree
  - the default is 1 for a classification problem like this, which we'll keep

```
set.seed(100,sample.kind = "Rounding")
wf=workflow() %>% add_recipe(y1_recipe) %>%
  add_model(rand_forest(mtry = tune()) %>%
  set_engine("ranger",importance="permutation") %>% set_mode("classification"))
tune_forest=wf %>% tune_grid(resamples=cv,
                        grid=expand.grid(mtry=5:10),
                        metrics=metric_set(macro_weight_f1))
wf=wf %>% finalize_workflow(tune_forest %>% select_best())
fit_y1_forest=fit(wf,train_set)
tune_forest %>% select_best() %>% knitr::kable()
```

mtry	.config
8	Preprocessor1_Model4



The contract year versions of the holistic advanced stats are the top 3 most important variables in the random forest model. The other two variables in the top 5 are percentage of games played in the contract year and percentage of games played in the past 3 years, while the new percent of positional VORP variable is top 10.

Finally, we train the support vector machine on the model. 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

```
set.seed(100,sample.kind = "Rounding")
wf=workflow() %>% add_recipe(y1_recipe) %>%
  add_model(svm_rbf(rbf_sigma=tune(),cost=tune()) %>%
  set_engine("kernlab") %>% set_mode("classification"))
tune_svm=wf %>% tune_grid(resamples=cv,
                        grid=expand.grid(rbf_sigma=0.1^seq(2:5),
                                         cost=c(0.25,0.5,1)),
                        metrics=metric_set(macro_weight_f1))
wf=wf %>% finalize_workflow(tune_svm %>% select_best())
fit_y1_svm=fit(wf,train_set)
```

```
## maximum number of iterations reached 0.0008977438 0.000883558
```

```
tune_svm %>% select_best() %>% knitr::kable()
```

cost	rbf_sigma	.config
0.25	1e-04	Preprocessor1_Model04

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

Let's look at some performance metrics, and see which models we want to take along as inputs for predicting salary.

Method	Correct Predict %	Off by >1 Yr	Max Year Predicts	F1 Score
Multinomial	61.37%	13.04%	20	0.5099835
KNN	65.28%	14.43%	2	0.5052750
Decision Tree	54.93%	17.52%	0	0.4833842
Random Forest	98.13%	1.30%	21	0.5148873
SVM	40.91%	35.37%	0	0.5796736

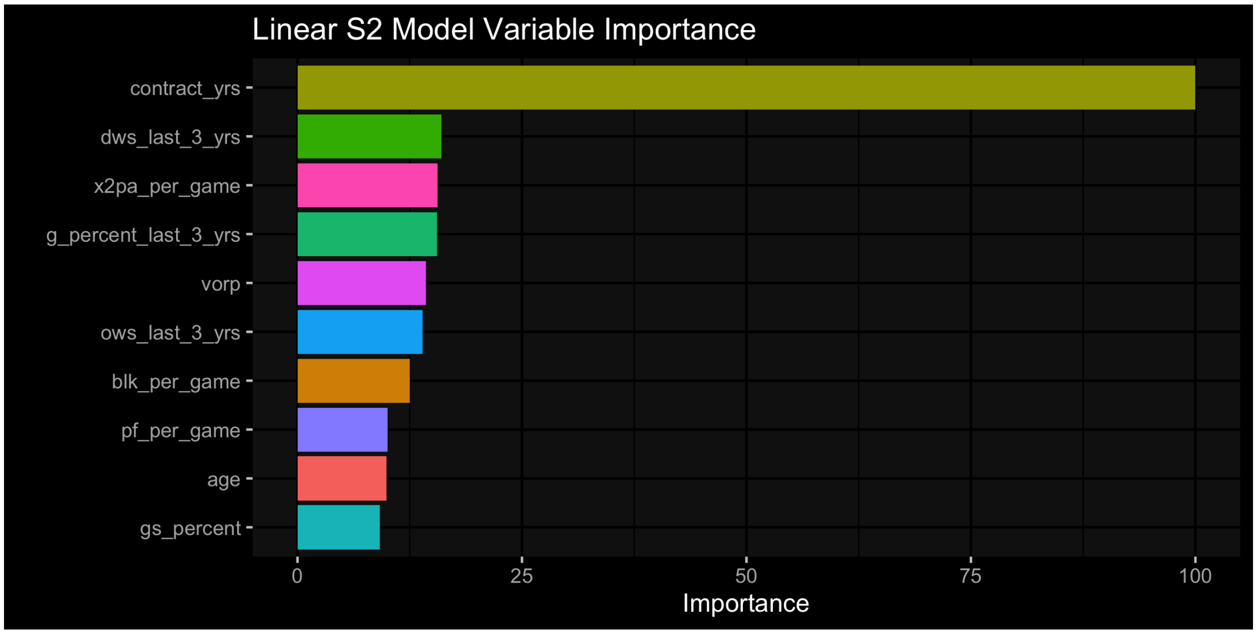
The SVM has the highest F1 score, but the lowest accuracy. In fact, there are almost as many predictions more than 1 year off as there are correct predictions. The random forest stands far above the rest with 98% accuracy! It also has the highest F1-score barring the SVM. In 2020's project, the random forests had the best performance as well, but had difficulty distinguishing 5-year contracts. Shifting to a classification problem seems to have solved that. The multinomial model had significantly more max-year predictions than KNN, whereas KNN pipped the multinomial model in F1-score and had ~4% better accuracy. While anticlimactic, I think the best course of action is to simply use the random forest model to make contract-year predictions. We'll convert the prediction column to use it as a numeric predictor.

```r
train_set_after_y1=train_set %>% select(-contract_yrs) %>%
  bind_cols(contract_yrs=
              as.numeric(
                as.character(
                  predict(fit_y1_forest,new_data=train_set) %>% pull()
                  )
                )
            )
cv=vfold_cv(train_set_after_y1,v=10,strata=type,repeats=5)
s2_recipe=recipe(first_year_percent_of_cap~.,data=train_set_after_y1) %>%
  update_role(seas_id:player,new_role="id") %>% step_dummy(type)
```

Now I can run through the models for salary using the predicted years as an input. The models I won't reuse are the multinomial model and the decision tree. The multinomial model is for classification only, while we are attempting to predict a continuous outcome in the first year salary as a percentage of the salary cap. We'll sub in a linear regression model as our new baseline. Since there's so many different salary percentages, a decision tree model would be useless or far too complicated.

```r
wf=workflow() %>% add_recipe(s2_recipe) %>%
  add_model(linear_reg() %>%
              set_engine("lm") %>%
              set_mode("regression"))
tune_lin=wf %>% tune_grid(resamples=cv,metrics=metric_set(rmse,mae))
wf=wf %>% finalize_workflow(tune_lin %>% select_best(metric="rmse"))
fit_s2_lin=fit(wf,train_set_after_y1)

prettify_vip(fit_s2_lin,"Linear S2 Model Variable Importance")
```



Contract years dwarf all other variables in terms of importance.

```
wf=workflow() %>% add_recipe(s2_recipe) %>%
  add_model(nearest_neighbor(neighbors=tune()) %>%
              set_engine("kknn") %>%
              set_mode("regression"))
tune_knn=wf %>% tune_grid(resamples=cv,
                        grid=expand.grid(neighbors=5:50),
                        metrics=metric_set(rmse,mae))
wf=wf %>% finalize_workflow(tune_knn %>% select_best(metric="rmse"))
fit_s2_knn=fit(wf,train_set_after_y1)
tune_knn %>% select_best(metric="rmse") %>% knitr::kable()
```

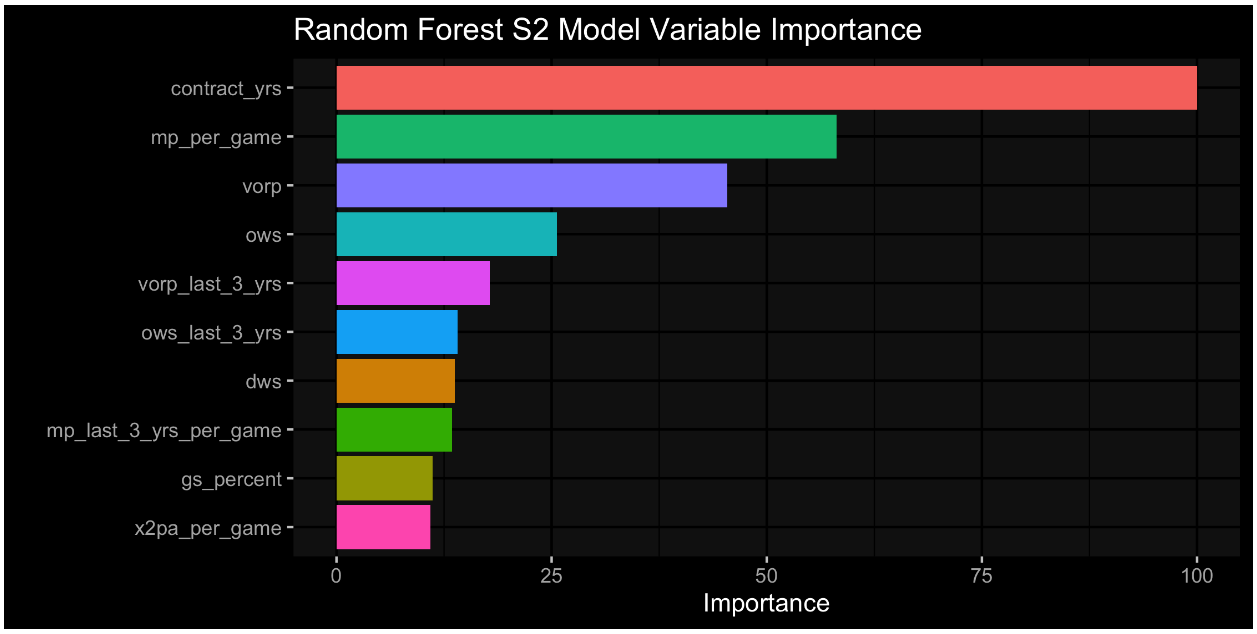
neighbors	.config
24	Preprocessor1_Model20

The best knn salary model is around the middle of the tune grid, using 24 comparables.

```
set.seed(100,sample.kind = "Rounding")
wf=workflow() %>% add_recipe(s2_recipe) %>%
  add_model(rand_forest(mtry = tune()) %>%
  set_engine("ranger",importance="permutation") %>% set_mode("regression"))
tune_forest=wf %>% tune_grid(resamples=cv,
                        grid=expand.grid(mtry=5:10),
                        metrics=metric_set(rmse,mae))
wf=wf %>% finalize_workflow(tune_forest %>% select_best(metric="rmse"))
fit_s2_forest=fit(wf,train_set_after_y1)
tune_forest %>% select_best(metric="rmse") %>% knitr::kable()
```

mtry	.config
10	Preprocessor1_Model6

```
prettify_vip(fit_s2_forest,"Random Forest S2 Model Variable Importance")
```



Like the linear model, contract years are by far the most important variable in the random forest model, although not to the same extent.

```r
set.seed(100,sample.kind = "Rounding")
wf=workflow() %>% add_recipe(s2_recipe) %>%
  add_model(svm_rbf(rbf_sigma=tune(),cost=tune()) %>%
  set_engine("kernlab") %>% set_mode("regression"))
tune_svm=wf %>% tune_grid(resamples=cv,
                        grid=expand.grid(rbf_sigma=0.1^seq(2:5),
                                         cost=c(0.25,0.5,1)),
                        metrics=metric_set(rmse,mae))
wf=wf %>% finalize_workflow(tune_svm %>% select_best(metric="rmse"))
fit_s2_svm=fit(wf,train_set_after_y1)
tune_svm %>% select_best(metric="rmse") %>% knitr::kable()
```

cost	rbf_sigma	.config
1	0.01	Preprocessor1_Model10

Here are the performance metrics for the salary portion of the Y1S2 model.

Method	Off By >5%	Within 2%	MAE	RMSE
Linear	5.70%	72.13%	0.0177654	0.0257611
KNN	6.52%	77.75%	0.0176845	0.0292287
Random Forest	0.33%	92.99%	0.0145995	0.0250257
SVM	4.40%	85.66%	0.0146166	0.0253318

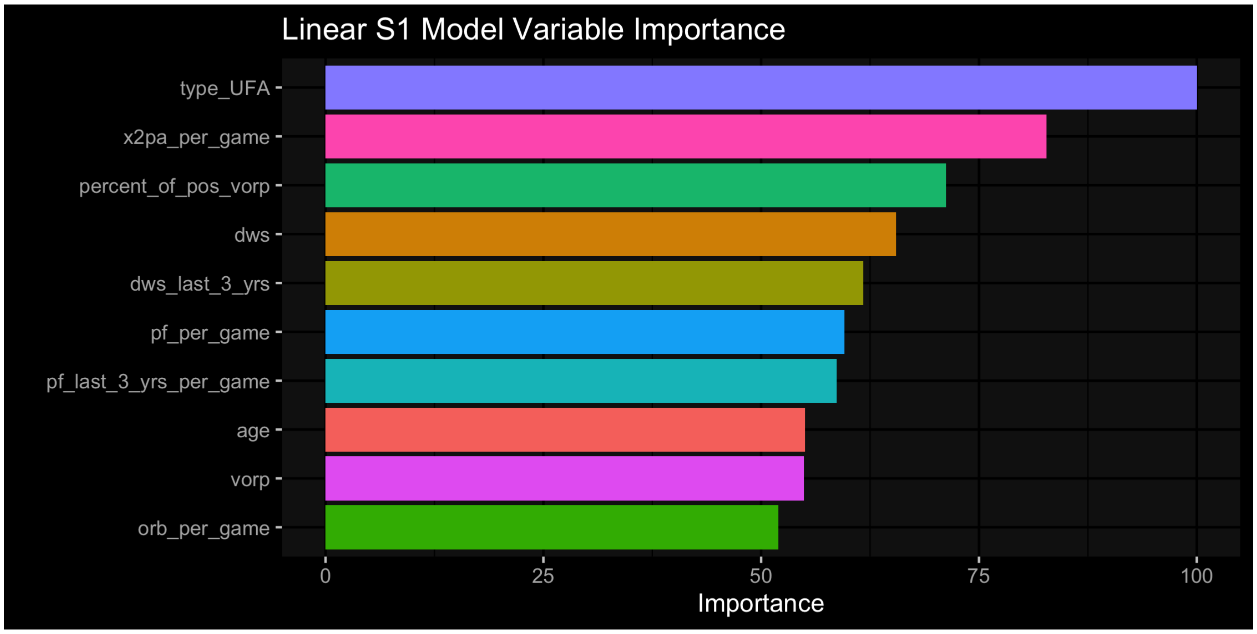
Mean absolute error is the measure of the average difference between forecasts, while the residual mean squared error penalizes large errors. The random forest has the best RMSE, and also tops the leaderboard of maximizing predictions within 2% of actual value, and minimizing predictions that are more than 5% away. The linear model also has a very close RMSE to the random forest & SVM, but it suffers on the dataset-specific metrics. The KNN model has the worst RMSE, but still performs better on predictions than the linear model. With the SVM & random forest being so close in traditional metrics as well as the SVM being no slouch on dataset-specific metrics, we will take the mean of the SVM & random forest as our salary prediction in the Y1S2 model.

Predicting Salary First, then Years

Now I'll switch up the order, predicting salary first and then years.

```r
set.seed(100)
cv=vfold_cv(train_set,v=10,strata=type,repeats=5)
s1_recipe=recipe(first_year_percent_of_cap~.,data=train_set) %>%
  update_role(seas_id:player,new_role="id") %>%
  step_rm(contract_yrs) %>% step_dummy(type)
```

```r
wf=workflow() %>% add_recipe(s1_recipe) %>%
  add_model(linear_reg() %>%
              set_engine("lm") %>%
              set_mode("regression"))
tune_lin=wf %>% tune_grid(resamples=cv,metrics=metric_set(rmse,mae))
wf=wf %>% finalize_workflow(tune_lin %>% select_best(metric="rmse"))
fit_s1_lin=fit(wf,train_set)
prettify_vip(fit_s1_lin,"Linear S1 Model Variable Importance")
```



The most important factor here is the type of free agent a player is. Next on the importance graph is amount of 2-point shots attempted per game in the contract year. 2PA/G is a measure of scoring opportunity: leaders in this category are usually those also near the top in points per game, as their talent (usually) warrants them taking that increased volume of shots.

```r
wf=workflow() %>% add_recipe(s1_recipe) %>%
  add_model(nearest_neighbor(neighbors=tune()) %>%
              set_engine("kknn") %>%
              set_mode("regression"))
tune_knn=wf %>% tune_grid(resamples=cv,
                          grid=expand.grid(neighbors=5:50),
                          metrics=metric_set(rmse,mae))
wf=wf %>% finalize_workflow(tune_knn %>% select_best(metric="rmse"))
fit_s1_knn=fit(wf,train_set)
tune_knn %>% select_best(metric="rmse") %>% knitr::kable()
```

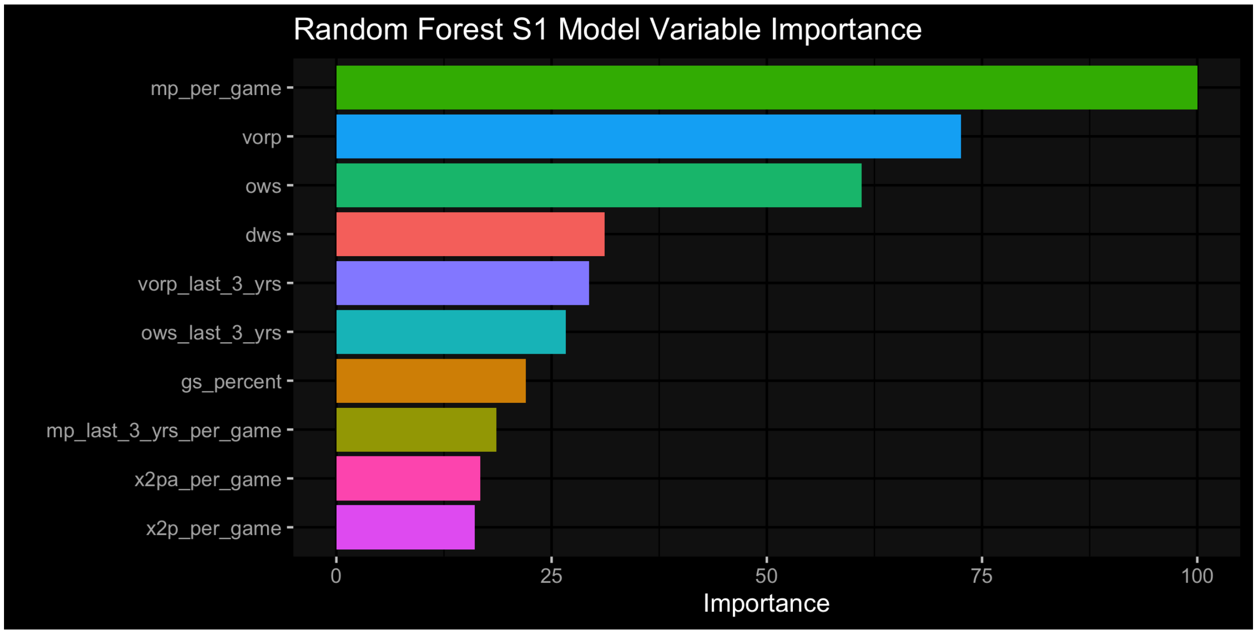
neighbors	.config
27	Preprocessor1_Model23

The best KNN salary-first model is much like the KNN salary-second model in that the tuned neighbor parameter is in the middle of the tune grid, taking the 27 closest players in similarity to construct a player's salary prediction.

```r
set.seed(100,sample.kind = "Rounding")
wf=workflow() %>% add_recipe(s1_recipe) %>%
  add_model(rand_forest(mtry = tune()) %>%
  set_engine("ranger",importance="permutation") %>% set_mode("regression"))
tune_forest=wf %>% tune_grid(resamples=cv,
                          grid=expand.grid(mtry=5:10),
                          metrics=metric_set(rmse,mae))
wf=wf %>% finalize_workflow(tune_forest %>% select_best(metric="rmse"))
fit_s1_forest=fit(wf,train_set)
tune_forest %>% select_best(metric="rmse") %>% knitr::kable()
```

mtry	.config
10	Preprocessor1_Model6

```
prettify_vip(fit_s1_forest,"Random Forest S1 Model Variable Importance")
```



The salary-first forest is a tad mode widely clustered at the top than the salary-second forest, having 3 variables >50% importance vs 2 respectively. Excluding contract years in the salary-second forest, the top 3 variables of importance follow the exact same order in both forests.

```
set.seed(100,sample.kind = "Rounding")
wf=workflow() %>% add_recipe(s1_recipe) %>%
  add_model(svm_rbf(rbf_sigma=tune(),cost=tune()) %>%
  set_engine("kernlab") %>% set_mode("regression"))
tune_svm=wf %>% tune_grid(resamples=cv,
                          grid=expand.grid(rbf_sigma=0.1^seq(2:5),
                                           cost=c(0.25,0.5,1)),
                          metrics=metric_set(rmse,mae))
wf=wf %>% finalize_workflow(tune_svm %>% select_best(metric="rmse"))
fit_s1_svm=fit(wf,train_set)
tune_svm %>% select_best(metric="rmse") %>% knitr::kable()
```

cost	rbf_sigma	.config
1	0.01	Preprocessor1_Model10

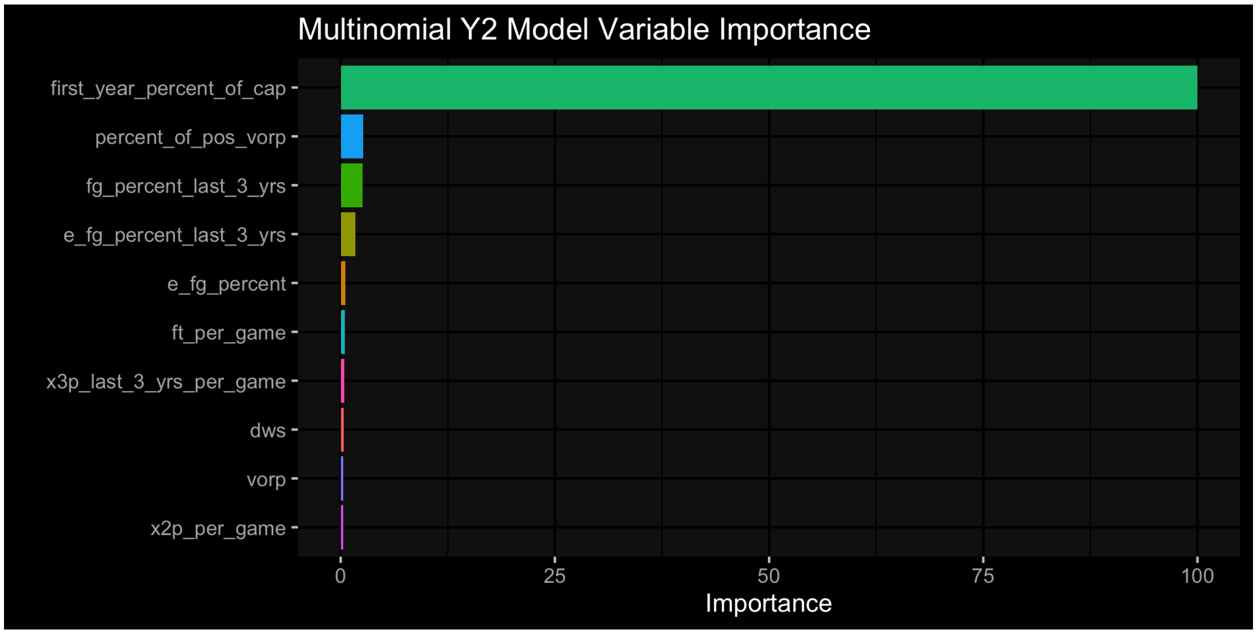
Below are the performance metrics for the salary-first models.

Method	Off By >5%	Within 2%	MAE	RMSE
Linear	8.56%	63.49%	0.0218176	0.0314814
KNN	7.82%	73.68%	0.0202803	0.0323535
Random Forest	0.81%	90.71%	0.0196987	0.0306023
SVM	6.11%	78.32%	0.0189661	0.0300713

Once again, the random forest finishes second to the SVM model in traditional metrics but beats it out in our dataset-specific metrics. All 4 models performed worse when predicting salary-first vs salary-second. With the SVM dipping below 80%, we will use the random forest as our salary prediction.

```r
train_set_after_s1=train_set %>% select(-first_year_percent_of_cap) %>%
  bind_cols(first_year_percent_of_cap=predict(fit_s1_forest,new_data=train_set) %>%
              pull())
cv=vfold_cv(train_set_after_s1,v=10,strata=type,repeats=5)
y2_recipe=recipe(contract_yrs~.,data=train_set_after_s1) %>%
  update_role(seas_id:player,new_role="id") %>% step_dummy(type)
```

```r
wf=workflow() %>% add_recipe(y2_recipe) %>%
  add_model(multinom_reg(penalty=0) %>%
              set_engine("glmnet") %>%
              set_mode("classification"))
tune_multinom=wf %>% tune_grid(resamples=cv,metrics=metric_set(macro_weight_f1))
wf=wf %>% finalize_workflow(tune_multinom %>% select_best())
fit_y2_multinom=fit(wf,train_set_after_s1)
prettify_vip(fit_y2_multinom,"Multinomial Y2 Model Variable Importance")
```



The multinomial years-second model essentially discards all other variables in favor of the predicted first-year-percent-of-cap variable.

```r
wf=workflow() %>% add_recipe(y2_recipe) %>%
  add_model(nearest_neighbor(neighbors = tune()) %>%
  set_engine("kknn") %>% set_mode("classification"))
tune_knn=wf %>% tune_grid(resamples=cv,
                          grid=expand.grid(neighbors=5:50),
                          metrics=metric_set(macro_weight_f1))
wf=wf %>% finalize_workflow(tune_knn %>% select_best())
fit_y2_knn=fit(wf,train_set_after_s1)
tune_knn %>% select_best() %>% knitr::kable()
```

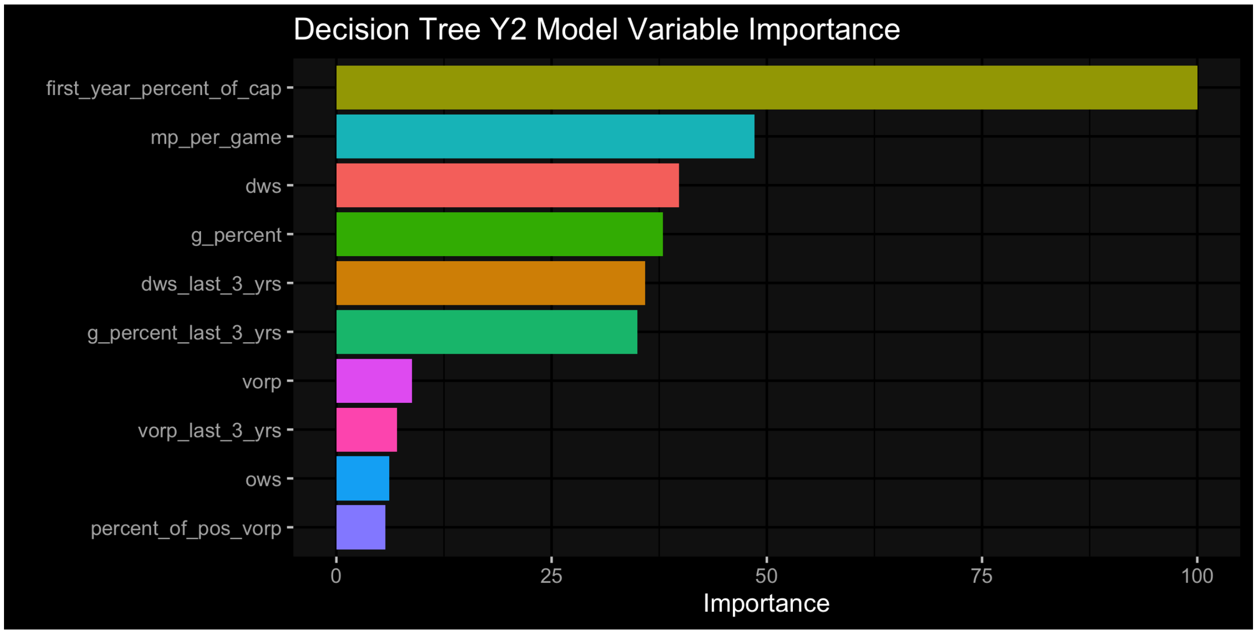
neighbors	.config
47	Preprocessor1_Model43

47 neighbors are needed in order for the KNN years-second model to classify a player's contract years.

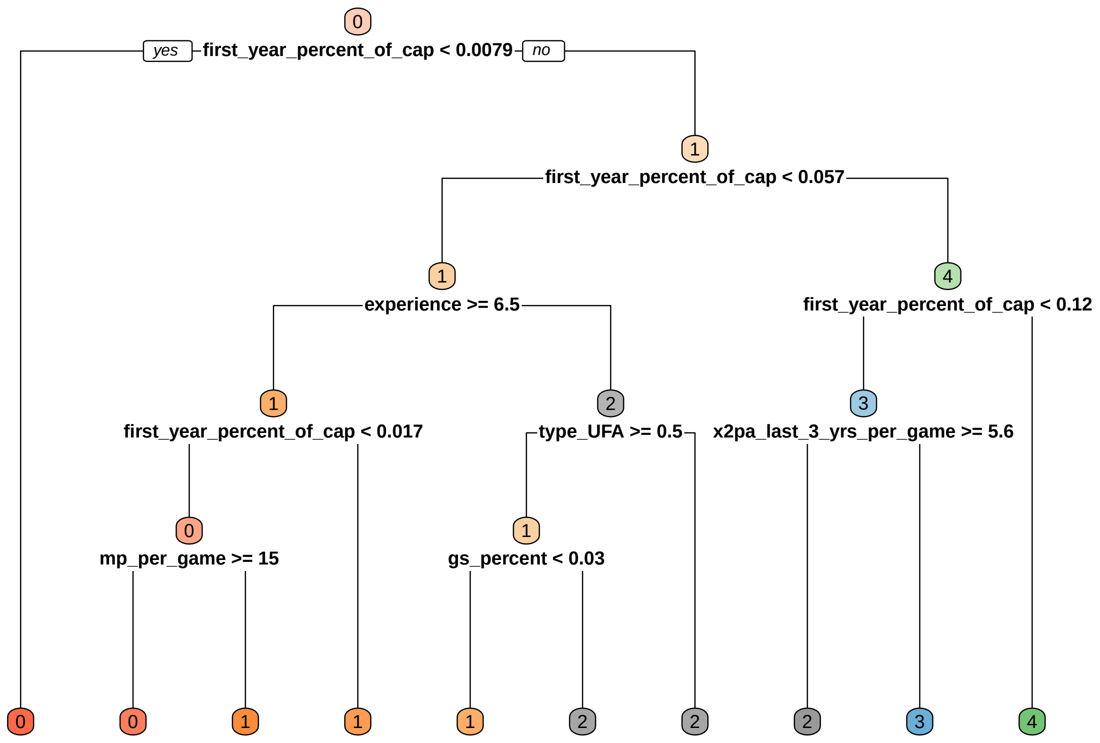
```
set.seed(100,sample.kind = "Rounding")
wf=workflow() %>% add_recipe(y2_recipe) %>%
  add_model(decision_tree(cost_complexity = tune(),min_n=tune()) %>%
  set_engine("rpart") %>% set_mode("classification"))
tune_tree=wf %>% tune_grid(resamples=cv,
                        grid=expand.grid(cost_complexity=0.1^seq(2,5),
                                         min_n=seq(1:10)),
                        metrics=metric_set(macro_weight_f1))
wf=wf %>% finalize_workflow(tune_tree %>% select_best())
fit_y2_tree=fit(wf,train_set_after_s1)
tune_tree %>% select_best() %>% knitr::kable()
```

cost_complexity	min_n	.config
0.01	1	Preprocessor1_Model01

```
prettify_vip(fit_y2_tree,"Decision Tree Y2 Model Variable Importance")
```



First-year-percent-of-cap shoots up to 1st in importance in the years-second decision tree, much like we saw in the years-second multinomial model. Contract-year minutes per game is the second most important variable, but doesn't even reach 50% of first-year-percent-of-cap's importance.

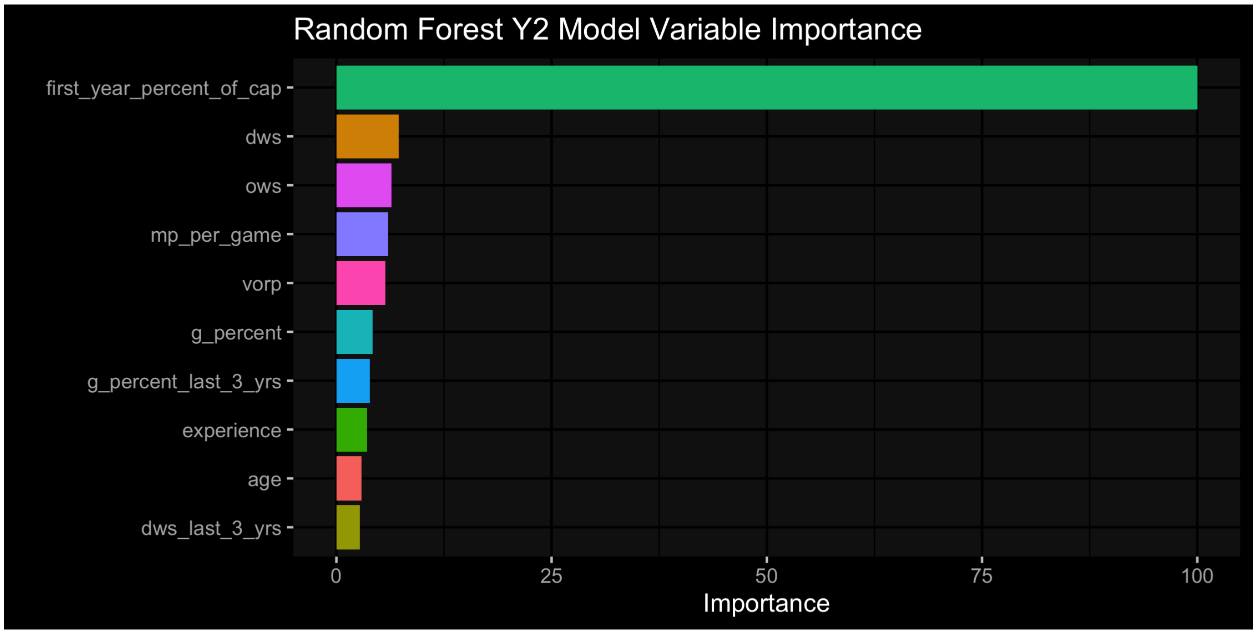

0
yes
first_year_percent_of_cap < 0.0079
no
1
first_year_percent_of_cap < 0.057
1
experience >= 6.5
4
first_year_percent_of_cap < 0.12
1
first_year_percent_of_cap < 0.017
2
type_UFA >= 0.5
3
x2pa_last_3_yrs_per_game >= 5.6
0
mp_per_game >= 15
1
gs_percent < 0.03
0
0
1
1
1
2
2
2
3
4

The years-second decision tree predicts some three year contracts, but still can't seem to figure out the division between 4- and 5-year contracts. It maximizes its prediction at 4 contract years if the first-year-percent-of-cap is greater than 12%.

```
set.seed(100,sample.kind = "Rounding")
wf=workflow() %>% add_recipe(y2_recipe) %>%
  add_model(rand_forest(mtry = tune()) %>%
  set_engine("ranger",importance="permutation") %>% set_mode("classification"))
tune_forest=wf %>% tune_grid(resamples=cv,
                        grid=expand.grid(mtry=5:10),
                        metrics=metric_set(macro_weight_f1))
wf=wf %>% finalize_workflow(tune_forest %>% select_best())
fit_y2_forest=fit(wf,train_set_after_s1)
tune_forest %>% select_best() %>% knitr::kable()
```

mtry	.config
10	Preprocessor1_Model6

```
prettify_vip(fit_y2_forest,"Random Forest Y2 Model Variable Importance")
```



The variable importance graph looks like the multinomial model with a slightly less severe division between the first-year-percent-of-cap variable importance and the rest of the variable importances.

```
set.seed(100,sample.kind = "Rounding")
wf=workflow() %>% add_recipe(y2_recipe) %>%
  add_model(svm_rbf(rbf_sigma=tune(),cost=tune()) %>%
  set_engine("kernlab") %>% set_mode("classification"))
tune_svm=wf %>% tune_grid(resamples=cv,
                          grid=expand.grid(rbf_sigma=0.1^seq(2:5),
                                           cost=c(0.25,0.5,1)),
                          metrics=metric_set(macro_weight_f1))
wf=wf %>% finalize_workflow(tune_svm %>% select_best())
fit_y2_svm=fit(wf,train_set_after_s1)
```

```
## maximum number of iterations reached 0.0008944529 0.000881132
```

```
tune_svm %>% select_best() %>% knitr::kable()
```

cost	rbf_sigma	.config
0.25	1e-04	Preprocessor1_Model04

Last but not least, here are the years performance metrics of the S1Y2 model.

Method	Correct Predict %	Off by >1 Yr	Max Year Predicts	F1 Score
Multinomial	80.52%	4.73%	21	0.6830092
KNN	65.77%	13.53%	3	0.5167366
Decision Tree	71.31%	6.11%	0	0.6721096
Random Forest	99.76%	0.24%	21	0.6752496
SVM	40.91%	35.37%	0	0.5797525

The random forest is second in F1 score, but has less than one percent of predictions that are off by more than one year. We will use the random forest as the sole model to generate Y2 predictions.

Results

Now it's time to run on the 2022 data.

Since the models depend on contract year stats, these players were removed from the evaluation set:

player	type
D.J. Stewart Jr.	RFA
Dru Smith	RFA
John Wall	UFA
Kendrick Nunn	UFA
Quincy Douby	RFA
T.J. Warren	UFA

Years First, Salary Second

As a reminder, we're using a random forest as our years prediction, and using the mean of a random forest and SVM to predict salary.

```
eval_set=inner_join(three_year_rolling_stats,current_fa) %>%
  left_join(.,all_player_pos) %>% group_by(season,pos_group) %>%
  mutate(position_vorp=sum(vorp_last_3_yrs)) %>% ungroup() %>%
  mutate(percent_of_pos_vorp=vorp_last_3_yrs/position_vorp) %>%
  mutate(contract_yrs=factor(contract_yrs,levels = 0:5)) %>%
  select(-c(pos,pos_group,position_vorp)) %>%
  mutate(across(-c(seas_id:experience,type:contract_yrs),~round(.,digits=4)))
write_csv(eval_set,"Data/Eval Set.csv")
```

```
eval_y1_predict=as.numeric(as.character(
  predict(fit_y1_forest,new_data=eval_set) %>% pull()))
eval_s2_predict_forest=predict(fit_s2_forest,
                               new_data=eval_set %>% select(-contract_yrs) %>%
                                 bind_cols(contract_yrs=eval_y1_predict))
eval_s2_predict_svm=predict(fit_s2_svm,
                               new_data=eval_set %>% select(-contract_yrs) %>%
                                 bind_cols(contract_yrs=eval_y1_predict))
eval_s2_predict=tibble(forest=eval_s2_predict_forest %>% pull(),
                       svm=eval_s2_predict_svm %>% pull()) %>% rowwise() %>%
  mutate(m=mean(c_across(forest:svm))) %>% ungroup() %>% pull(m)
```

Salary First, Years Second

For the S1Y2 model, we are using random forests for each component.

```
eval_s1_predict=predict(fit_s1_forest,new_data=eval_set) %>% pull()
eval_y2_predict=as.numeric(as.character(
  predict(fit_y2_forest,new_data=eval_set %>% select(-first_year_percent_of_cap) %>%
          bind_cols(first_year_percent_of_cap=eval_s1_predict)) %>%
  pull()))
```

Player Option Decisions and Selected Free Agents

Before I look at which players might be accepting or declining their option and then some selected high-profile free agents, let's look at the distribution of contract years.

yrs_Y1S2	n
0	103
1	42
2	25
3	13
4	6
5	1

yrs_S1Y2	n
0	69
1	63
2	37
3	14
4	6
5	1

Two years ago on the first iteration of this project, both the Y1S2 & the S1Y2 had only 4-5 players being out of the league. This didn't make sense, as the annual rookie draft brings in at least 30 new players. The problem was that there were players who were predicted to receive 1 year contracts but receiving salaries that were less than the minimum. My solution was to convert any salary predictions less than the minimum to zero, and any years predictions to zero where the salary prediction is zero.

For the 2022-2023 season, the salary cap is $122,000,000. The minimum contract (from RealGM) is $1,616,044. Therefore the minimum first year percent of cap in 2021 is 1.32%.

```
first_yr_min=1616044/122000000
final_results<-final_results %>%
  mutate(yrs_Y1S2=ifelse(yr1_cap_percent_Y1S2<=first_yr_min,0,yrs_Y1S2)) %>%
  mutate(yr1_cap_percent_Y1S2=ifelse(yrs_Y1S2==0,0,yr1_cap_percent_Y1S2)) %>%
  mutate(yrs_S1Y2=ifelse(yr1_cap_percent_S1Y2<=first_yr_min,0,yrs_S1Y2)) %>%
  mutate(yr1_cap_percent_S1Y2=ifelse(yrs_S1Y2==0,0,yr1_cap_percent_S1Y2))
```

yrs_Y1S2	n
0	103
1	42
2	25
3	13
4	6
5	1

yrs_S1Y2	n
0	84
1	49
2	36
3	14
4	6
5	1

The Y1S2 predictions remain unchanged, while some 1-year and one 2-year contract in the S1Y2 predictions got converted to 0-year due to the salary being below the minimum threshold.

Another task is to make the information more digestible. Contracts are usually reported in terms of total value, and players usually get five percent raises per year, so I'll add a column for each method for total contract value assuming the projected cap value of $122 million.

```
final_results<-final_results %>%
  mutate(total_Y1S2=
           round(yr1_cap_percent_Y1S2*122000000*((1.05)^(yrs_Y1S2)-1)/0.05,digits=-4),
         total_S1Y2=
             round(yr1_cap_percent_S1Y2*122000000*((1.05)^(yrs_S1Y2)-1)/0.05,digits=-4))
```

Player Option Decisions

player	Y1S2 Cap %	yrs_Y1S2	total_Y1S2	S1Y2 Cap %	yrs_S1Y2	total_S1Y2	Option Type	2021 Option
James Harden	22.12%	3	$85,090,000	28.81%	3	$110,810,000	PO	$42,782,880
Russell Westbrook	9.45%	1	$11,530,000	12.17%	1	$14,840,000	PO	$41,358,814
Bradley Beal	13.15%	2	$32,890,000	15.13%	2	$37,840,000	PO	$35,073,168
Kyrie Irving	17.76%	2	$44,430,000	20.03%	2	$50,090,000	PO	$34,122,650
JaMychal Green	2.56%	1	$3,130,000	3.44%	1	$4,200,000	PO	$8,200,000
P.J. Tucker	7.16%	2	$17,900,000	8.44%	2	$21,100,000	PO	$7,175,000
Ivica Zubac	7.60%	1	$9,280,000	10.73%	3	$41,260,000	CO	$7,000,000
Patty Mills	6.39%	2	$15,990,000	7.48%	2	$18,720,000	PO	$6,037,250
Pat Connaughton	8.28%	3	$31,840,000	8.46%	3	$32,550,000	PO	$5,333,333
Hamidou Diallo	2.78%	1	$3,400,000	4.41%	1	$5,390,000	CO	$5,200,000
Cory Joseph	2.59%	1	$3,160,000	3.72%	1	$4,540,000	PO	$5,032,500
Jeff Green	3.70%	1	$4,510,000	4.81%	1	$5,870,000	PO	$4,500,000
Bobby Portis	13.66%	4	$71,830,000	12.17%	4	$64,000,000	PO	$4,456,290
Mike Muscala	2.97%	1	$3,620,000	3.85%	1	$4,700,000	CO	$3,500,000
Nicolas Batum	6.36%	2	$15,910,000	7.45%	2	$18,620,000	PO	$3,249,280
Trey Burke	0.00%	0	$0	0.00%	0	$0	PO	$3,150,000
Frank Jackson	0.00%	0	$0	2.01%	2	$5,030,000	CO	$3,075,000
Trey Lyles	5.19%	2	$12,970,000	5.26%	2	$13,160,000	CO	$2,562,500
Tony Bradley	0.00%	0	$0	1.55%	1	$1,890,000	PO	$1,912,787
Sviatoslav Mykhailiuk	0.00%	0	$0	0.00%	0	$0	PO	$1,803,969
Thanasis Antetokounmpo	0.00%	0	$0	0.00%	0	$0	PO	$1,803,969
Shake Milton	3.91%	2	$9,790,000	2.79%	2	$6,970,000	CO	$1,664,676
Stanley Johnson	1.66%	1	$2,020,000	2.02%	1	$2,470,000	CO	$1,620,069
Isaiah Roby	4.88%	2	$12,200,000	4.55%	2	$11,370,000	CO	$1,600,201
Jalen McDaniels	2.78%	2	$6,950,000	1.98%	2	$4,960,000	CO	$1,532,398
Naz Reid	5.65%	2	$14,130,000	5.81%	2	$14,540,000	CO	$1,532,398
Jae'Sean Tate	6.70%	2	$16,760,000	6.64%	2	$16,610,000	CO	$1,481,839
Luguentz Dort	7.50%	2	$18,770,000	7.69%	2	$19,240,000	CO	$1,346,733
Austin Reaves	4.56%	3	$17,550,000	3.24%	3	$12,470,000	CO	$1,244,388
Luka Garza	0.00%	0	$0	0.00%	0	$0	CO	$1,244,388
Oshae Brissett	4.35%	2	$10,870,000	4.45%	2	$11,140,000	CO	$1,229,523
Dean Wade	3.51%	2	$8,780,000	3.20%	2	$8,000,000	CO	$1,225,201
Jaylen Nowell	4.19%	2	$10,490,000	4.22%	2	$10,560,000	CO	$1,175,151
Wenyen Gabriel	0.00%	0	$0	0.00%	0	$0	CO	$954,267
Carsen Edwards	0.00%	0	$0	0.00%	0	$0	CO	$946,211
Sam Hauser	0.00%	0	$0	0.00%	0	$0	CO	$938,628
Juwan Morgan	0.00%	0	$0	0.00%	0	$0	CO	$917,432

Let's look at the elephant in the room first: Beal is in-line for a max or near-max contract in real life. He is a gifted scorer, finishing as the runner-up in the scoring race in both 2020 (30.5 to Harden's 34.3) and 2021 (31.3 to Stephen Curry's 32). His per-year actual salary may very well eclipse the total contract values predicted here. The reasoning behind the low prediction is Beal's games played percentages (49% in the contract year and 70% over the last three years). This in turn affected his offensive & defensive win shares as well as his VORP, which are all cumulative stats. Another interesting case is Bobby Portis, who's projected for 4-year contracts with a total value ranging between \$ 64M & \$ 72M. It was rumored he gave the Milwaukee Bucks a "hometown discount" in an attempt to fortify the team to win a second consecutive championship, while retaining flexibility to hit the market by making the second year a player option.

Selected Free Agents

Now let's see what the models have predicted for this year's other notable free agents. I'll present them five at a time and comment on some projections. We'll look at restricted free agents first.

player	age	Y1S2 Cap %	yrs_Y1S2	total_Y1S2	S1Y2 Cap %	yrs_S1Y2	total_S1Y2
Miles Bridges	23	23.55%	4	$123,840,000	21.92%	4	$115,290,000
Deandre Ayton	23	17.08%	5	$115,160,000	16.23%	5	$109,440,000
Mo Bamba	23	11.47%	4	$60,320,000	9.52%	3	$36,620,000
Anfernee Simons	22	7.69%	2	$19,230,000	9.13%	4	$48,010,000
Collin Sexton	23	6.51%	1	$7,940,000	9.56%	2	$23,920,000

Ayton is the lone 5-year prediction in the entire test set for both models, but Miles Bridges pips him in total value even with only contracts predicted for 4 years. Simons follows in Gary Trent Jr's footsteps from last year in that he's a Portland guard capable of scoring in bunches who had a mini-breakout but only played 70% of his contract year games (Gary was traded to Toronto in his contract year). I'm confident in saying that Simons, much like GTJ last year, will eclipse his projections.

player	age	Y1S2 Cap %	yrs_Y1S2	total_Y1S2	S1Y2 Cap %	yrs_S1Y2	total_S1Y2
Caleb Martin	26	7.55%	3	$29,050,000	6.68%	2	$16,700,000
Cody Martin	26	7.41%	3	$28,510,000	6.62%	3	$25,450,000
Amir Coffey	24	6.93%	3	$26,640,000	6.41%	2	$16,030,000
Nicolas Claxton	22	5.39%	3	$20,740,000	5.08%	2	$12,700,000
Donte DiVincenzo	25	3.87%	1	$4,730,000	4.49%	1	$5,480,000

The Martin twins (Cody stayed in Charlotte with the Hornets, while Caleb moved south to Miami) also have extremely similar projections, differing by 0.14% or $0.17M at most!

On to the unrestricted free agents.

player	age	Y1S2 Cap %	yrs_Y1S2	total_Y1S2	S1Y2 Cap %	yrs_S1Y2	total_S1Y2
Zach LaVine	26	24.66%	4	$129,660,000	24.84%	4	$130,610,000
Jalen Brunson	25	18.09%	4	$95,120,000	16.77%	4	$88,170,000
Mitchell Robinson	23	11.11%	4	$58,410,000	11.67%	3	$44,870,000
Montrezl Harrell	28	8.99%	2	$22,480,000	10.67%	2	$26,690,000
Chris Boucher	29	9.36%	3	$36,000,000	8.69%	3	$33,410,000

LaVine is the crown jewel of this class, and it shows with him holding the highest average prediction of first-year-percent-of-cap across guaranteed free agents. Brunson's star-making turn in the playoffs has increased interest in his services around the league. Harrell has now played for 4 teams in 3 years: he is a known quantity at this point, ~25 minutes a game of efficient offense (he has shot better than 60% from the field in all but one season in his career) and passable to below-average defense.

player	age	Y1S2 Cap %	yrs_Y1S2	total_Y1S2	S1Y2 Cap %	yrs_S1Y2	total_S1Y2
Malik Monk	23	7.77%	2	$19,440,000	9.46%	4	$49,720,000
Jusuf Nurkić	27	7.36%	2	$18,420,000	9.77%	2	$24,420,000
Tyus Jones	25	8.25%	3	$31,750,000	8.77%	3	$33,730,000
Dennis Schröder	28	6.67%	1	$8,140,000	9.11%	1	$11,110,000
Otto Porter	28	7.55%	3	$29,040,000	7.46%	3	$28,700,000

Monk was a bright spot in the Lakers' season, scoring 13.8 points per game on 47.3% shooting from the field, which were both career-highs. Tyus Jones provided a steady hand for the Memphis Grizzlies during

Ja Morant's various injury absences, and might be looking for a team that can provide a more consistent starting role.

player	age	Y1S2 Cap %	yrs_Y1S2	total_Y1S2	S1Y2 Cap %	yrs_S1Y2	total_S1Y2
Kevon Looney	25	7.49%	3	$28,820,000	7.10%	2	$17,760,000
Delon Wright	29	6.70%	3	$25,750,000	7.22%	3	$27,780,000
Joe Ingles	34	7.10%	3	$27,300,000	5.97%	2	$14,940,000
Ricky Rubio	31	5.74%	1	$7,010,000	7.22%	1	$8,810,000
Bruce Brown	25	5.26%	1	$6,420,000	7.30%	3	$28,080,000

Looney played his role exceedingly well in the Warriors' road to their 4th championship in 8 years, leading the team in rebounding and being defensively stout. The models don't know that Ingles tore his ACL in February of last year, but I'm still surprised that they would predict Ingles for a three-year contract considering his advanced age.

Conclusion

Using a combination of contract year stats and last-three-years stats, we set out to predict NBA free agent contracts for the 2021 offseason. Since we had correlated targets in contract length and first year percent of salary cap, we predicted one target first and used its predictions as an input for the second target. The contract years were predicted as a classification problem, while the salary was predicted as a regression model We used six models: linear (for salary) & multinomial (for contract length) as baselines, k-nearest-neighbors, decision tree (only for predicting contract length), a random forest and a support vector machine. James Harden, Zach LaVine, Miles Bridges & Deandre Ayton are all projected to get above \$ 100 million in total contract value.

The models suffer from being unable to quantify intangibles like playing reputation, team fit, within-season role changes, or willingness to take a reduced salary to be on a championship contender. The models also can't determine which team will sign which player. That highly depends on a sequence of events: if Team A signs this player, they don't have enough money to resign Player B, who then goes to Team C for less money, etc.

Since there were low predictions on Beal and Kyrie, maybe I need to add some sort of "starpower indicator" like recent All-Star or All-NBA appearances to pad up the predictions. Perhaps I should have implemented a time factor or weighted recent years more heavily, as team decision makers may have gotten smarter. I didn't remove highly correlated predictors by design, as I wanted the models themselves to perform feature selection and determine what the most important variables were.

Future work could include trying more models, like boosting (in which models are added sequentially, with later models in the sequence attempting to correct the errors of earlier models). Another example might be adding a draft pedigree variable, as teams might be more willing to:

A. take a chance on a player with mediocre surface-level stats, as the team could believe the player was misused and the inherent talent is still there
B. pay up for a player with good stats, as the high pick of the player could signal to the player's higher potential

A final option might be predicting a third target: whether a contract will end in an option year. Star players are more likely to demand the last year of their contract as a player option in order to take ownership of their future.