# NBA FA Contract Prediction 2020

Sumitro Datta

July 13, 2020

## Introduction

Day 1 of NBA free agency is by far the biggest day of the offseason for the league. It was a doozy last season, with seismic shifts in the league's power structure. Arguably 2 of the top 5 players (when healthy) in the league, as well as an additional 3 within the top 25, changed teams.

This year's class? Not as highly regarded as last year. Anthony Davis is the crown jewel, while Brandon Ingram is the best restricted free agent available. The hype of this class was further deflated when players like Kyle Lowry and Draymond Green signed extensions with their current 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(caret))
  install.packages("caret", repos = "http://cran.us.r-project.org")
#Rborist and ranger are random forest algorithm wrappers
if(!require(Rborist))
  install.packages("Rborist", repos = "http://cran.us.r-project.org")
if(!require(ranger))
  install.packages("ranger", 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")
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")
```

## Importing the Data

There are five files I'll import.

```r
advanced<-read_csv("Advanced.csv") %>%
  mutate(tm=ifelse(tm=="TOT","1TOT",tm)) %>%
  group_by(player_id,season) %>% arrange(tm) %>% slice(1) %>%
  mutate(tm=ifelse(tm=="1TOT","TOT",tm)) %>% select(seas_id:mp,ows:ws,vorp)
totals<-read_csv("Player Totals.csv") %>%
  mutate(tm=ifelse(tm=="TOT","1TOT",tm)) %>%
  group_by(player_id,season) %>% arrange(tm) %>% slice(1) %>%
  mutate(tm=ifelse(tm=="1TOT","TOT",tm))
advanced_and_totals<-inner_join(totals,advanced)
```

For the statistical data, I've scraped total and advanced stats from Basketball-Reference and placed them in .csv files. The advanced stats I kept were cumulative (offensive win shares, defensive win shares and value over replacement player). This was actually part of a larger project to scrape complete statistics for teams, players and awards from 1947-2019. To my knowledge, my dataset is unique in that it includes BAA stats and ABA stats. For players who played on multiple teams in one season, I kept their total stats and discarded the team-specific season portions.

```r
free_agents<-read_csv("2016-2019 Free Agents.csv")
```

For training data, I initially wanted to take free agents from the season 2013 onwards, because that was the start of consecutive salary cap increases. However, due to the easy accessibility of Basketball-Reference's free agent tracker, I decided to use free agents from 2016-2019. This would only include players who signed during offseasons (no in season signings).

- 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 around 250 players who:
  - went overseas
  - 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 salaries for remaining 400 or so players:
  - decision to make between using year 1 cap hit or year 1 salary, but decision was made for me based off websites used to gather the data (Capology, Spotrac and Basketball-Insiders)
    - Capology = main source, but inconsistent with min/near-min contracts
    - Spotrac requires a premium login to view contract cap hits beyond the last and current contracts, but the cash earnings are free to view
    - When player was near-minimum & played for multiple teams in one season, Basketball-Insiders had transaction history in team salary archives (needed player's first team of season)
- fixed some small errors
  - Nikola Jokic was an RFA, not a UFA when he signed his 5-year extension
  - Alex Len accepted the qualifying offer rather than no contract
- added suffixes (Jr., II, III) to certain players to match up with statistical data

```r
fa_2020<-read_csv("Free Agents 2020.csv")
#separate out options to compare what players options get if declined
fa_2020_options<-fa_2020 %>% filter(str_detect(type,"PO|CO"))
#remove club options (unsure how to handle), make player options all declined (UFA's)
fa_2020<-fa_2020 %>% filter(type != "CO") %>%
  mutate(type=ifelse(type=="PO","UFA",type)) %>% group_by(player) %>% slice(1)
```

Our evaluation set is the 2020 free agent class which I got off Spotrac. I had to edit it to match the Basketball-Reference names (mainly adding diacritics to European names). In addition, I filtered out players with options. I'm unsure how to handle club options, but players who decline player options become unrestricted free agents. I'll use this fact to see which players might decline their option. Instead of first year percent of cap, I listed the actual monetary salary (taken from Spotrac) in the 2021 season for the player options, since the cap number is subject to change.

```r
salary_cap_hist<-read_csv("Salary Cap History.csv")
#create variable of first year salary as percentage of cap
#easier to compare across years
free_agents<-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 I used was salary cap history, also from Basketball-Reference. To somewhat normalize comparisons across years, I converted the first year salary to a percentage of the salary cap.

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.

## Pre-Processing

```r
create_data<-function(x){
  a<-advanced_and_totals %>% group_by(player_id) %>%
    #three year sum
    mutate(across(-c(1:10,fg_percent,x3p_percent,
                     x2p_percent:e_fg_percent,ft_percent),
                  list(three_yrs=~rollapplyr(.,3,sum,partial=TRUE)),
                  .names="{col}_last_3_yrs")) %>%
    inner_join(.,x) %>%
    mutate(ws_per_48_last_3_yrs=ws_last_3_yrs/mp_last_3_yrs*48) %>%
    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(hof,lg,pos,tm,ws,ws_last_3_yrs,
              trb,trb_last_3_yrs,fg,fga,fg_last_3_yrs,fga_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:pts),list(per_game=~./g)),
           .after="gs") %>%
    select(-c(mp,x3p:x3pa,x2p:x2pa,ft:fta,orb:pts)) %>%
    mutate(across(mp_last_3_yrs:pts_last_3_yrs,list(per_game=~./g_last_3_yrs)),
           .after="gs_last_3_yrs") %>%
    select(-c(mp_last_3_yrs:pts_last_3_yrs)) %>% ungroup() %>%
    #scale games & cumulative advanced stats (since short season due to COVID)
    mutate(across(starts_with("g")|starts_with("vorp")|
                    contains("dws")|contains("ows"),
                  list(scaled=~scale(.))),.keep="unused") %>%
    relocate(g_scaled,gs_scaled,.after="experience") %>%
    relocate(g_last_3_yrs_scaled,gs_last_3_yrs_scaled,
             .before="mp_last_3_yrs_per_game") %>%
    relocate(vorp_scaled,ows_scaled,dws_scaled,.after="ft_percent")
  return(a)
}
advanced_and_totals_train_set<-create_data(free_agents)
```

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 Whiteside). I initially wanted to use totals to bake in availability/body fragility, but the shortened season would cause the model to declare all players to be fragile and underestimate their contract. Stats other than games played, games started, and the advanced stats (OWS, DWS and VORP) were converted to per game. Percentages were left alone. Games played, games started, and the advanced stats (OWS, DWS and VORP) were scaled to have a normal distribution (mean of 0 and standard deviation of 1).

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

Why I settled on 3 years:

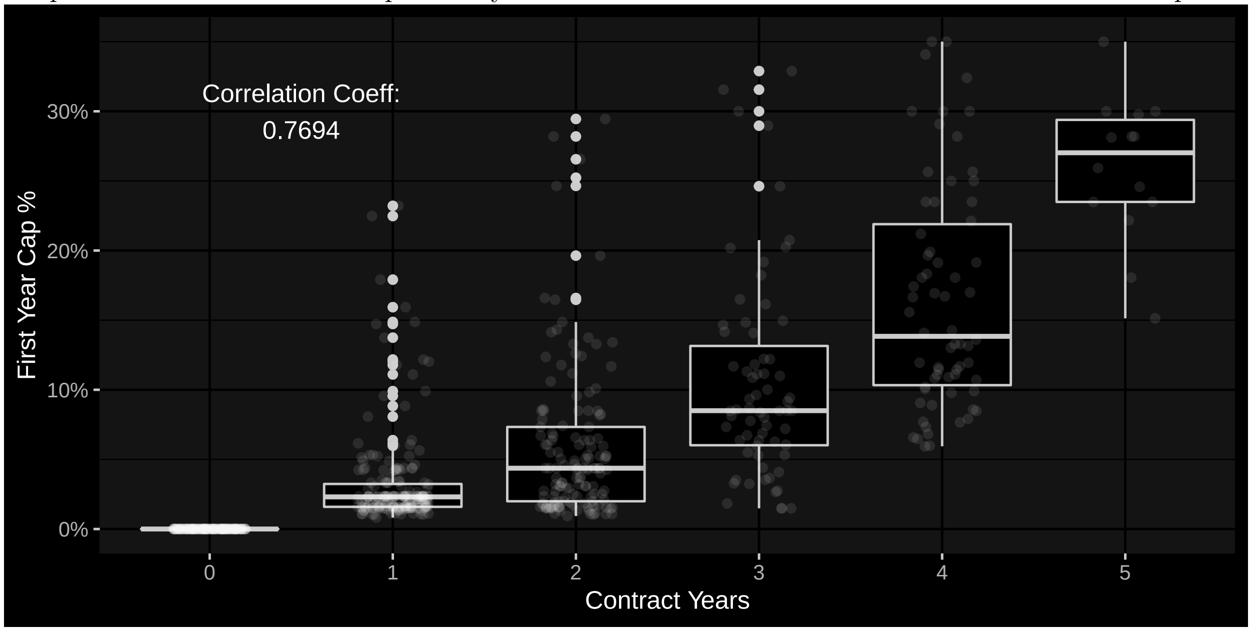
- Players do get paid on past performance, so just using contract year stats was out of the question
- 2 years opens up the possibility of a fluke year
  - Kawhi would have his nine game season bring down his averages significantly from his Raptors season: adding another year somewhat lessens this effect
- 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 Knicks didn't pay Derrick Rose to recapture his form of his MVP year (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 Donovan Mitchell 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 Utah would offer 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. I kept age and experience as predictor variables, but removed position because I felt it would ultimately reflect in the stats themselves.

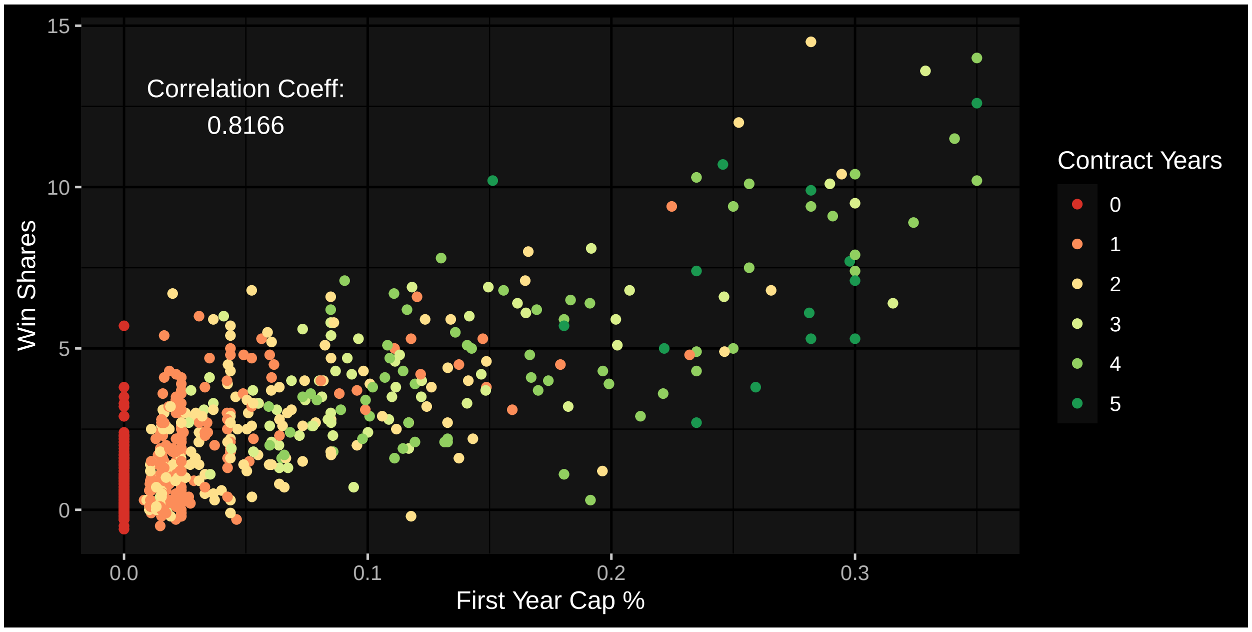
## Data Exploration and Visualizations

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.



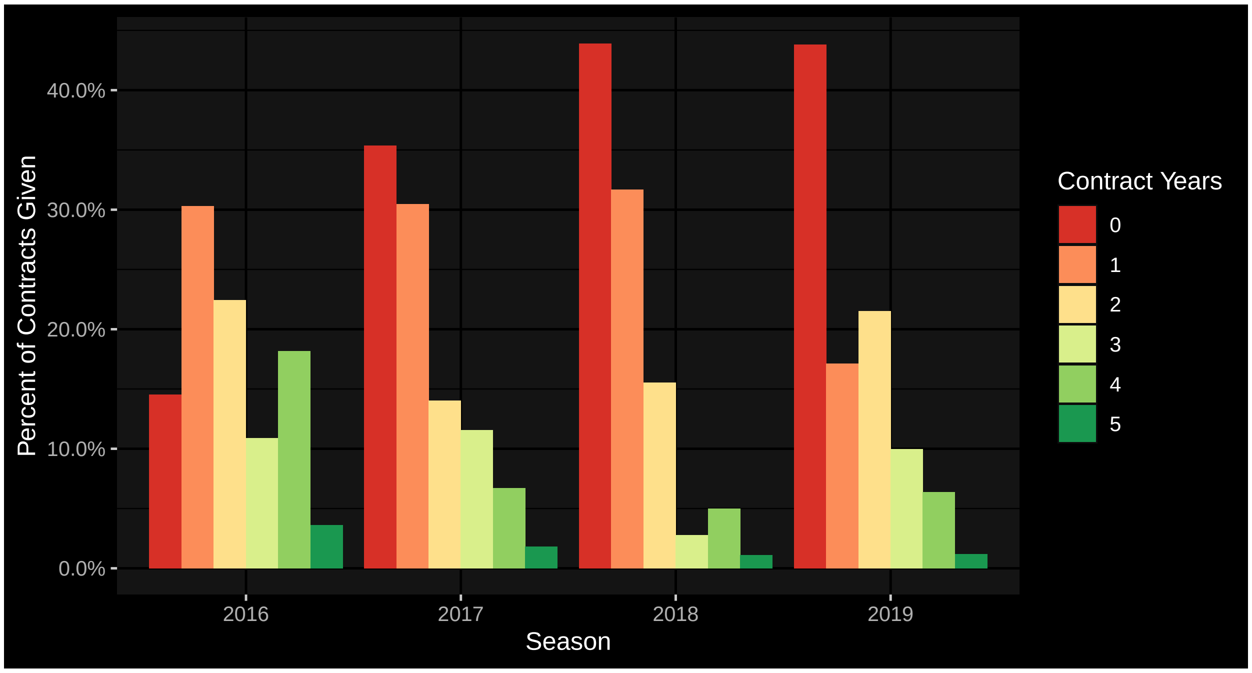
The correlation coefficient is 0.77, which shows that the two targets are strongly and positively correlated. The median value of the first year cap % (middle line in each box) increases with an increase in contract length.

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 0.8166, 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.

## 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. Since the dataset is relatively small (760 observations), I decided to utilize leave-one-out cross validation. 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.

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.

### The Models

I used a total of six models.

- linear regression model as a baseline
- 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: 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 models (`ranger` and `Rborist`): reduce instability by averaging multiple trees
  - also don't need cross validation, since the out of bag estimate is good enough
  - costs interpretability as there is no tree diagram that is representative of the decisions made
  - using `ranger` and `Rborist` rather than `randomForest` to decrease computational time
- 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

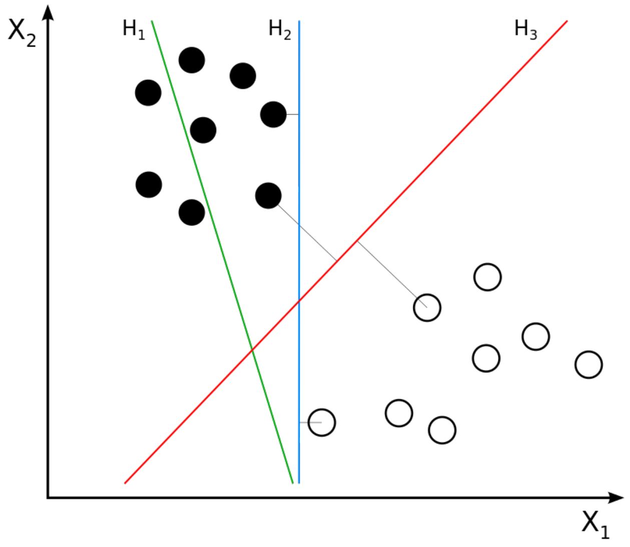


Figure 1: H1 does not separate the classes. H2 does, but only with a small margin. H3 separates them with the maximal margin. By User:ZackWeinberg, based on PNG version by User:Cyc - This file was derived from: Svm separating hyperplanes.png, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=22877598

### Predicting Years First, then Salary

I'll start by predicting years first and then salary with years as an input. First up is the linear model.

```
lin_yrs<-train(contract_yrs~.,
               data=(advanced_and_totals_train_set %>% ungroup() %>%
                       select(-c(seas_id:birth_year,
                                 first_year_percent_of_cap))),
               method="lm",
               trControl=trainControl(method="loocv"))
```

Let's get the top 5 most important variables of the linear model.

```
##                               Overall                      Vars
## typeUFA                     100.00000                   typeUFA
## ows_scaled                   44.06727                ows_scaled
## dws_scaled                   42.73433                dws_scaled
## x3pa_last_3_yrs_per_game     35.36471 x3pa_last_3_yrs_per_game
## age                          31.86516                       age
```

Whether a player is an unrestricted free agent is the most important factor for a linear model. This checks out, as RFAs are restricted to a maximum of 25% of the cap, whereas a UFA can go to 35%. Following that are the separate components of win shares in the contract year, the number of 3-point shots attempted per game over the last three years and age. Seeing the 3PA/game is reassuring, since the importance of the 3-point shot has exploded in recent years.

Next up is the k-nearest neighbors model. We can tune the parameter k to change how many players to compare to. We check values of k from 5 to 25.

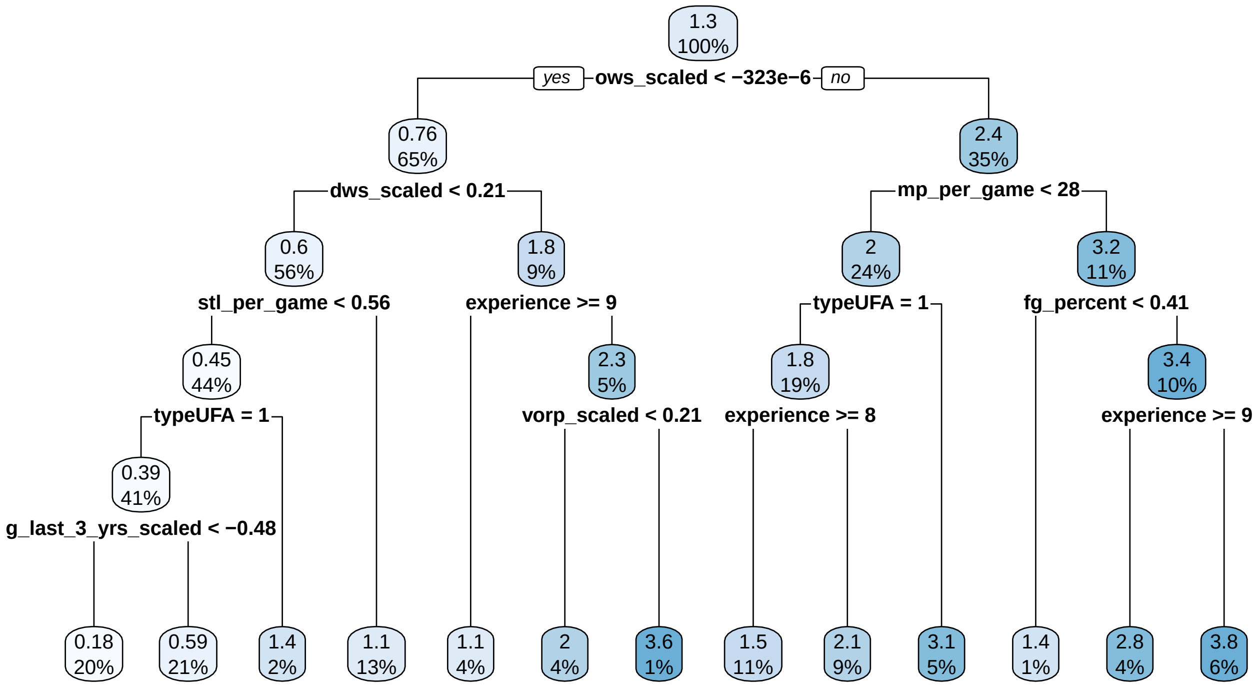
```
knn_yrs<-train(contract_yrs~.,
               data=(advanced_and_totals_train_set %>% ungroup() %>%
                       select(-c(seas_id:birth_year,first_year_percent_of_cap))),
               method="knn",
               tuneGrid=data.frame(k=c(5:25)),
               trControl=trainControl(method="loocv"))
knn_yrs$finalModel
```

```
## 5-nearest neighbor regression model
```

Looks like the best knn model is with using 5 closest comparable players.

The next model is the decision tree model. We can tune the 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. 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.

```
set.seed(100,sample.kind = "Rounding")
rpart_yrs<-train(contract_yrs~.,
           data=(advanced_and_totals_train_set %>% ungroup() %>%
                   select(-c(seas_id:birth_year,first_year_percent_of_cap))),
           method="rpart", tuneGrid = data.frame(cp=seq(0.005,0.02,0.001)),
           trControl=trainControl(method="loocv"))
```



The decision tree maximizes its prediction when a player does all of the following:

- has above average offensive win shares in the contract year
- plays more than 28 minutes per game in the contract year
- shoots better than 41% from the field in the contract year
- has less than 9 years of experience

The decision tree minimizes its prediction when a player does all of the following:

- has below average offensive win shares in the contract year
- has at most slightly above average defensive win shares in the contract year
- steals the ball less than 0.56 times per game in the contract year
- is an unrestricted free agent
- has played almost half a standard deviation less games than the average in the last 3 years

What's immediately noticeable is that the decision tree model will not predict any players to get 5 year contracts.

The next models are random forests. Since `ranger` and `Rborist` are *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. In addition, `ranger` and `Rborist` have a maximum of 3 tuning parameters:

- the number of variables to split at each decision node (mtry in ranger, predFixed in Rborist)
  - the default is the rounded square root of the number of variables, which in this case would be `round(sqrt(80))`
- the minimum number of observations needed to split a node (min.node.size in ranger, minNode in Rborist)
  - the default is 5 for a regression problem like this
- splitrule is the rule used to split a node
  - the default is to minimize variance

Unfortunately, due to limited computing capacity, we'll be using the defaults.

```r
set.seed(10,sample.kind="Rounding")
ranger_yrs<-train(contract_yrs~.,
             data=(advanced_and_totals_train_set %>% ungroup() %>%
                     select(-c(seas_id:birth_year,first_year_percent_of_cap))),
             method="ranger", importance="permutation")
set.seed(3,sample.kind="Rounding")
rborist_yrs<-train(contract_yrs~.,
                  data=(advanced_and_totals_train_set %>% ungroup() %>%
                          select(-c(seas_id:birth_year,first_year_percent_of_cap))),
                  method="Rborist", importance="permutation")
get_5_top_important(ranger_yrs)
```

```
##                Overall        Vars
## dws_scaled  100.00000  dws_scaled
## typeUFA      95.00442     typeUFA
## ows_scaled   90.73358  ows_scaled
## mp_per_game  61.24319 mp_per_game
## vorp_scaled  58.59191 vorp_scaled
```

```r
get_5_top_important(rborist_yrs)
```

```
##                  Overall         Vars
## ows_scaled   100.00000   ows_scaled
## dws_scaled    56.76480   dws_scaled
## mp_per_game   50.29692  mp_per_game
## vorp_scaled   41.87863  vorp_scaled
## pts_per_game  14.74880 pts_per_game
```

Both algorithms take heavy use of the scaled contract year advanced stats, and also have contract year minutes per game as a top 5 variable. The type of free agent is a significant element in the ranger model.

Finally, we train the support vector machine on the model. There are two tuning parameters:

- 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
  - estimated using internal `sigest` function
- 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

As with the random forests, we will stay with the defaults due to limited computing capacity.

```r
set.seed(2,sample.kind="Rounding")
svm_yrs<-train(contract_yrs~.,
               data=(advanced_and_totals_train_set %>% ungroup() %>%
                       select(-c(seas_id:birth_year,first_year_percent_of_cap))),
               method="svmRadial", trControl=trainControl(method="loocv"))
```

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 | MAE | RMSE |
|---|---:|---:|---:|---:|---:|
| Linear | 0.4552632 | 0.0828947 | 5 | 0.7295992 | 0.7295992 |
| KNN | 0.4973684 | 0.0947368 | 2 | 0.7817544 | 0.7817544 |
| Decision Tree | 0.4486842 | 0.0750000 | 0 | 0.7533149 | 0.7533149 |
| Ranger | 0.8171053 | 0.0000000 | 1 | 0.7348781 | 0.9199267 |
| Rborist | 0.8526316 | 0.0000000 | 2 | 0.7549106 | 0.9447599 |
| SVM | 0.5947368 | 0.0750000 | 0 | 0.7029836 | 0.7029836 |

Mean absolute error is the measure of the average difference between forecasts, while the residual mean squared error penalizes large errors. The random forests provide by far the best performance, being above 80% accuracy when no other model is above 60%. However, they have a hard time distinguishing max contract year players. This is somewhat understandable, as the fifth year is only accessible to players resigning with their current team. The models could get confused seeing similar players in stats, but one signed for five years and one signed for four. To alleviate this, I propose:

- if a player is predicted by ANY model to be a 5 year player, their contract year prediction is 5
- else, use a median of Rborist, ranger and SVM

Why am I including SVM? Well, it has the best performance among the other models. I'm assuming that the random forest models will generally agree with each other. In the off chance they don't, I'll rely on the SVM to break the tie.

```
rborist_yrs_vec=predict(rborist_yrs,newdata=advanced_and_totals_train_set %>%
                          ungroup() %>%
                   select(-c(seas_id:birth_year,first_year_percent_of_cap)))
ranger_yrs_vec=predict(ranger_yrs,newdata=advanced_and_totals_train_set %>%
                         ungroup() %>%
                         select(-c(seas_id:birth_year,
                                   first_year_percent_of_cap)))
svm_yrs_vec=predict(svm_yrs,newdata=advanced_and_totals_train_set %>%
                      ungroup() %>%
                         select(-c(seas_id:birth_year,
                                   first_year_percent_of_cap)))
linear_yrs_vec=predict(lin_yrs,newdata=advanced_and_totals_train_set %>%
                         ungroup() %>%
                         select(-c(seas_id:birth_year,
                                   first_year_percent_of_cap)))
rpart_yrs_vec=predict(rpart_yrs,newdata=advanced_and_totals_train_set %>%
                        ungroup() %>%
                        select(-c(seas_id:birth_year,
                                  first_year_percent_of_cap)))
knn_yrs_vec=predict(knn_yrs,newdata=advanced_and_totals_train_set %>%
                      ungroup() %>%
                        select(-c(seas_id:birth_year,
                                  first_year_percent_of_cap)))

advanced_and_totals_sal_predict=advanced_and_totals_train_set %>%
  select(-contract_yrs) %>%
  add_column(contract_yrs=
               ifelse(round(rowMaxs(cbind(rborist_yrs_vec,ranger_yrs_vec,
                                          svm_yrs_vec,linear_yrs_vec,
                                          rpart_yrs_vec,knn_yrs_vec)))>=5,5,
                      round(rowMedians(cbind(rborist_yrs_vec,ranger_yrs_vec)))))
```

Now I can run through the models for salary using the predicted years as an input. The only model I won't reuse is the decision tree. Since there's so many different salary percentages, a decision tree model would be useless or far too complicated.

```
lin_sal_second<-train(first_year_percent_of_cap~.,
               data=(advanced_and_totals_sal_predict %>% ungroup() %>%
                       select(-c(seas_id:birth_year))),
               method="lm",
               trControl=trainControl(method="loocv"))
get_5_top_important(lin_sal_second)
```

```
##                           Overall                 Vars
## contract_yrs            100.00000         contract_yrs
## dws_last_3_yrs_scaled    21.48918 dws_last_3_yrs_scaled
## x2pa_per_game            20.89302        x2pa_per_game
## gs_scaled                19.59373            gs_scaled
## ows_last_3_yrs_scaled    18.72897 ows_last_3_yrs_scaled
```

Contract years dwarf all other variables in terms of importance. While the contract year win shares are key in determining the length of the contract, the three-year win shares are driving factors in the salary portion.

```
knn_sal_second<-train(first_year_percent_of_cap~.,
               data=(advanced_and_totals_sal_predict %>% ungroup() %>%
                       select(-c(seas_id:birth_year))),
               tuneGrid=data.frame(k=c(5:25)),
               method="knn",
               trControl=trainControl(method="loocv"))
knn_sal_second$finalModel
```

```
## 8-nearest neighbor regression model
```

The KNN salary model takes the eight closest comparables compared to five for the KNN years model. This checks out, since there's a wider range for salary.

```
set.seed(10,sample.kind="Rounding")
ranger_sal_second<-train(first_year_percent_of_cap~.,
               data=(advanced_and_totals_sal_predict %>% ungroup() %>%
                       select(-c(seas_id:birth_year))),
               method="ranger",importance="permutation")

set.seed(20,sample.kind="Rounding")
rborist_sal_second<-train(first_year_percent_of_cap~.,
                  data=(advanced_and_totals_sal_predict %>% ungroup() %>%
                          select(-c(seas_id:birth_year))),
                  method="Rborist",importance="permutation")
```

```
get_5_top_important(ranger_sal_second)
```

```
##                  Overall         Vars
## contract_yrs 100.000000 contract_yrs
## vorp_scaled   12.993926  vorp_scaled
## mp_per_game   10.947835  mp_per_game
## pts_per_game  10.550708 pts_per_game
## gs_scaled      5.700334    gs_scaled
```

```
get_5_top_important(rborist_sal_second)
```

```
##                          Overall                  Vars
## mp_per_game            100.00000           mp_per_game
## contract_yrs            92.36370          contract_yrs
## pts_per_game            74.48223          pts_per_game
## vorp_scaled             74.17610           vorp_scaled
## ows_last_3_yrs_scaled   36.11859 ows_last_3_yrs_scaled
```

The ranger model mimics the linear model in that the contract years is by far the most substantial factor when predicting salary. There is a big 4 in terms of variable importance for the rborist model: contract year minutes per game, contract length, contract year points per game and contract year VORP.

```
set.seed(30,sample.kind="Rounding")
svm_sal_second<-train(first_year_percent_of_cap~.,
                   data=(advanced_and_totals_sal_predict %>% ungroup() %>%
                           select(-c(seas_id:birth_year))),
                   method="svmRadial", trControl=trainControl(method="loocv"))
```

| Method | Off By >5% | MAE | RMSE |
|---|---|---|---|
| Linear | 0.0750000 | 0.0206442 | 0.0206442 |
| KNN | 0.1078947 | 0.0232765 | 0.0232765 |
| Ranger | 0.0184211 | 0.0161127 | 0.0287658 |
| Rborist | 0.0065789 | 0.0164040 | 0.0290209 |
| SVM | 0.0421053 | 0.0197899 | 0.0197899 |

With the MAEs being relatively similar, I'll take the median of the models.

### Predicting Salary First, then Years

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

```
lin_sal<-train(first_year_percent_of_cap~.,
               data=(advanced_and_totals_train_set %>% ungroup() %>%
                       select(-c(seas_id:birth_year,contract_yrs))),
               method="lm",
               trControl=trainControl(method="loocv"))
get_5_top_important(lin_sal)
```

```
##                           Overall                 Vars
## typeUFA                 100.00000              typeUFA
## x2pa_per_game            98.49682        x2pa_per_game
## dws_last_3_yrs_scaled    71.83384 dws_last_3_yrs_scaled
## dws_scaled               71.40868           dws_scaled
## x3p_percent              70.89221          x3p_percent
```

The most important factor here is the type of free agent a player is, which is similar to the linear years-first model. Right on the heels of free agent type 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 warrants them taking that increased volume of shots.

```
knn_sal<-train(first_year_percent_of_cap~.,
               data=(advanced_and_totals_train_set %>% ungroup() %>%
                       select(-c(seas_id:birth_year,contract_yrs))),
               method="knn",
               tuneGrid=data.frame(k=c(5:25)),
               trControl=trainControl(method="loocv"))
knn_sal$finalModel
```

```
## 12-nearest neighbor regression model
```

The KNN salary-first includes even more comparables than the KNN salary-second model (12 vs 8).

```
set.seed(10,sample.kind="Rounding")
ranger_sal<-train(first_year_percent_of_cap~.,
             data=(advanced_and_totals_train_set %>% ungroup() %>%
                     select(-c(seas_id:birth_year,contract_yrs))),
             method="ranger", importance="permutation")
set.seed(3,sample.kind="Rounding")
rborist_sal<-train(first_year_percent_of_cap~.,
                  data=(advanced_and_totals_train_set %>% ungroup() %>%
                          select(-c(seas_id:birth_year,contract_yrs))),
                  method="Rborist", importance="permutation")
```

```
get_5_top_important(ranger_sal)
```

```
##                 Overall         Vars
## vorp_scaled   100.00000  vorp_scaled
## mp_per_game    89.77262  mp_per_game
## ows_scaled     73.36771   ows_scaled
## pts_per_game   72.50753 pts_per_game
## dws_scaled     69.55954   dws_scaled
```

```
get_5_top_important(rborist_sal)
```

```
##                           Overall                  Vars
## vorp_scaled             100.00000           vorp_scaled
## mp_per_game              94.06695           mp_per_game
## pts_per_game             89.87324          pts_per_game
## ows_scaled               33.27641            ows_scaled
## vorp_last_3_yrs_scaled   18.42221 vorp_last_3_yrs_scaled
```

Contract year VORP and contract year minutes per game are the two most significant variables in both random forests. Contract year points per game rounds out the big 3 for the Rborist model.

```
set.seed(2,sample.kind="Rounding")
svm_sal<-train(first_year_percent_of_cap~.,
               data=(advanced_and_totals_train_set %>% ungroup() %>%
                       select(-c(seas_id:birth_year,contract_yrs))),
               method="svmRadial", trControl=trainControl(method="loocv"))
```

| Method | Off By >5% | MAE | RMSE |
|---|---|---|---|
| Linear | 0.1092105 | 0.0243321 | 0.0243321 |
| KNN | 0.1223684 | 0.0247301 | 0.0247301 |
| Ranger | 0.0131579 | 0.0230175 | 0.0351948 |
| Rborist | 0.0078947 | 0.0232042 | 0.0353620 |
| SVM | 0.0644737 | 0.0233510 | 0.0233510 |

The MAE range for the salary-first model is much smaller than the equivalent for the salary second model. I'll take the median, as I did for the previous.

```
rborist_sal_vec=predict(rborist_sal,newdata=advanced_and_totals_train_set %>%
                          ungroup() %>%
                   select(-c(seas_id:birth_year,contract_yrs)))
ranger_sal_vec=predict(ranger_sal,newdata=advanced_and_totals_train_set %>% ungroup() %>%
                         select(-c(seas_id:birth_year,contract_yrs)))
svm_sal_vec=predict(svm_sal,newdata=advanced_and_totals_train_set %>% ungroup() %>%
                         select(-c(seas_id:birth_year,contract_yrs)))
linear_sal_vec=predict(lin_sal,newdata=advanced_and_totals_train_set %>% ungroup() %>%
                         select(-c(seas_id:birth_year,contract_yrs)))
knn_sal_vec=predict(knn_sal,newdata=advanced_and_totals_train_set %>% ungroup() %>%
                        select(-c(seas_id:birth_year,contract_yrs)))

advanced_and_totals_yrs_predict=advanced_and_totals_train_set %>%
  select(-first_year_percent_of_cap) %>%
  add_column(first_year_percent_of_cap=rowMedians(cbind(
    rborist_sal_vec,ranger_sal_vec,svm_sal_vec,linear_sal_vec,knn_sal_vec)))
```

Now I can run through the models for years using the predicted salary as an input.

```r
lin_yrs_second<-train(contract_yrs~.,
               data=(advanced_and_totals_yrs_predict %>% ungroup() %>%
                       select(-c(seas_id:birth_year))),
               method="lm",
               trControl=trainControl(method="loocv"))
get_5_top_important(lin_yrs_second)
```

```
##                              Overall                      Vars
## first_year_percent_of_cap 100.00000 first_year_percent_of_cap
## typeUFA                    40.56444                   typeUFA
## gs_scaled                  28.41306                 gs_scaled
## vorp_scaled                27.42094               vorp_scaled
## g_last_3_yrs_scaled        21.59877       g_last_3_yrs_scaled
```

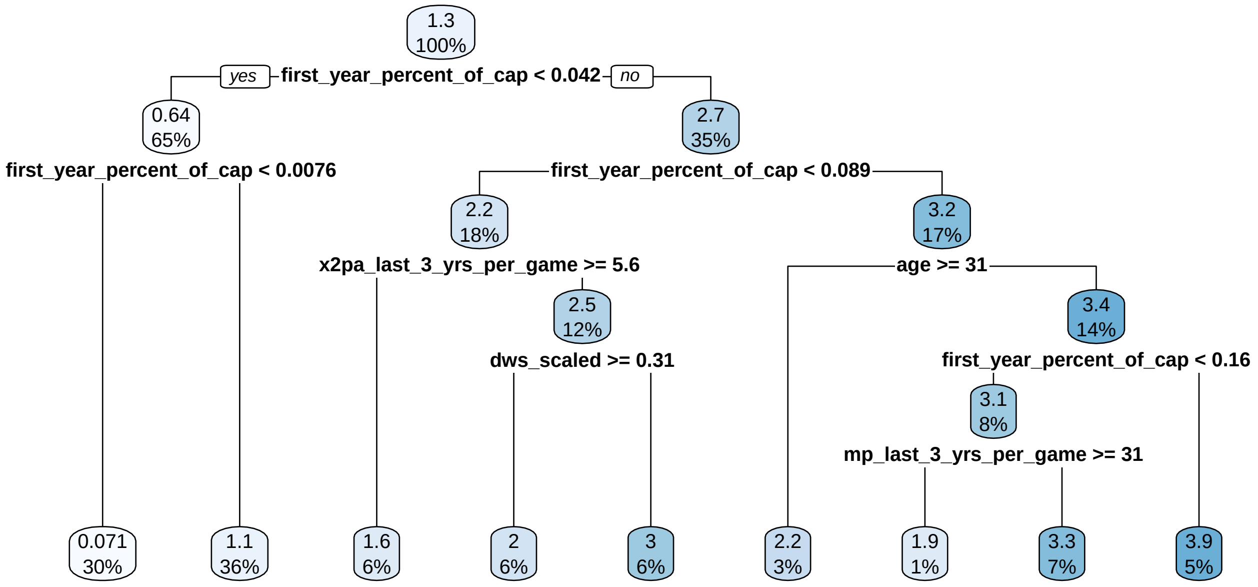
As we saw in the years-first model, the predicted target dwarfs all other variables in importance of predicting the second target.

```r
knn_yrs_second<-train(contract_yrs~.,
               data=(advanced_and_totals_yrs_predict %>% ungroup() %>%
                       select(-c(seas_id:birth_year))),
               method="knn",
               tuneGrid=data.frame(k=c(5:25)),
               trControl=trainControl(method="loocv"))
knn_yrs_second$finalModel
```

```
## 5-nearest neighbor regression model
```

Just like the years-first KNN model, the years-second KNN model only takes the 5 most similar players into account.

```r
set.seed(100,sample.kind="Rounding")
rpart_yrs_second<-train(contract_yrs~.,
               data=(advanced_and_totals_yrs_predict %>% ungroup() %>%
                       select(-c(seas_id:birth_year))),
               method="rpart", tuneGrid = data.frame(cp=seq(0.005,0.02,0.001)),
               trControl=trainControl(method="loocv"))
```


1.3
100%
yes
first_year_percent_of_cap < 0.042
no
0.64
65%
2.7
35%
first_year_percent_of_cap < 0.0076
first_year_percent_of_cap < 0.089
2.2
18%
3.2
17%
x2pa_last_3_yrs_per_game >= 5.6
age >= 31
2.5
12%
3.4
14%
dws_scaled >= 0.31
first_year_percent_of_cap < 0.16
3.1
8%
mp_last_3_yrs_per_game >= 31
0.071
30%
1.1
36%
1.6
6%
2
6%
3
6%
2.2
3%
1.9
1%
3.3
7%
3.9
5%

The singular decision tree again has trouble with predicting max contract length. Salary makes up 50% (4 of 8) of the decisions in the tree.

The decision tree minimizes its prediction when a player has a predicted salary of less than 0.76% of the salary cap.

The decision tree maximizes its prediction when a player is less than 31 years old and their predicted salary is above 16% of the cap.

```
set.seed(10,sample.kind="Rounding")
ranger_yrs_second<-train(contract_yrs~.,
               data=(advanced_and_totals_yrs_predict %>% ungroup() %>%
                       select(-c(seas_id:birth_year))),
               method="ranger", importance="permutation")
set.seed(20,sample.kind="Rounding")
rborist_yrs_second<-train(contract_yrs~.,
               data=(advanced_and_totals_yrs_predict %>% ungroup() %>%
                       select(-c(seas_id:birth_year))),
               method="Rborist", importance="permutation")
```

```
get_5_top_important(ranger_yrs_second)
```

```
##                              Overall                    Vars
## first_year_percent_of_cap 100.000000 first_year_percent_of_cap
## typeUFA                     4.085861                   typeUFA
## dws_scaled                  3.659244                dws_scaled
## mp_per_game                 3.331912               mp_per_game
## ows_scaled                  3.187064                ows_scaled
```

```
get_5_top_important(rborist_yrs_second)
```

```
##                               Overall                    Vars
## first_year_percent_of_cap 100.0000000 first_year_percent_of_cap
## experience                  3.2033402                experience
## age                         2.0311189                       age
## pts_per_game                1.6695068              pts_per_game
## x2pa_per_game               0.9169872             x2pa_per_game
```

These are easily the biggest differences between most important & second-most important variables I've seen!

```
set.seed(30,sample.kind="Rounding")
svm_yrs_second<-train(contract_yrs~.,
                   data=(advanced_and_totals_yrs_predict %>% ungroup() %>%
                           select(-c(seas_id:birth_year))),
                   method="svmRadial", trControl=trainControl(method="loocv"))
```

| Method | Correct Predict % | Off By >1 Yr | Max Year Predicts | MAE | RMSE |
|---|---|---|---|---|---|
| Linear | 0.5802632 | 0.0697368 | 22 | 0.6379354 | 0.6379354 |
| KNN | 0.4973684 | 0.0947368 | 2 | 0.7814474 | 0.7814474 |
| Decision Tree | 0.7013158 | 0.0473684 | 0 | 0.5506388 | 0.5506388 |
| Ranger | 0.8210526 | 0.0000000 | 2 | 0.5392439 | 0.7545000 |
| Rborist | 0.8526316 | 0.0052632 | 11 | 0.5308783 | 0.7647890 |
| SVM | 0.7039474 | 0.0355263 | 6 | 0.6755156 | 0.6755156 |

All models achieved at least the same if not a better correct prediction percentage than when predicting years first. The number of max year predictions has substantially increased (and in the linear model's case, has outstripped the actual number of max contracts given out). I'll take the median of all models except KNN, which has the lowest prediction accuracy by around 8%.

## Results

Now it's time to test them on the 2020 data. I'll refer to the years first, salary second model as Y1S2 and the salary first, years second model as S1Y2.

```
advanced_and_totals_eval_set<-create_data(fa_2020)
```

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

| player | type |
|---|---|
| Andre Roberson | UFA |
| Cam Reynolds | RFA |
| Isaiah Hicks | RFA |
| Jontay Porter | RFA |
| Kobi Simmons | RFA |
| Kyle Alexander | RFA |
| Ray Spalding | RFA |
| Shaq Buchanan | RFA |

### Years First, Salary Second

As a reminder, the Y1S2 model had trouble distinguishing max contract players. So if any of the models predict 5 years, that is the prediction for that player. Otherwise, I'm using the median of the Rborist, ranger and svm models. For the salary portion, I take the median of all 5 models used (ranger, Rborist, svm, knn and linear).

```
lin_2020_yrs_first=predict_yrs(lin_yrs)
knn_2020_yrs_first=predict_yrs(knn_yrs)
rpart_2020_yrs_first=predict_yrs(rpart_yrs)
ranger_2020_yrs_first=predict_yrs(ranger_yrs)
rborist_2020_yrs_first=predict_yrs(rborist_yrs)
svm_2020_yrs_first=predict_yrs(svm_yrs)

yrs_first_vec=ifelse(round(rowMaxs(cbind(
  rborist_2020_yrs_first,ranger_2020_yrs_first,svm_2020_yrs_first,
  lin_2020_yrs_first,rpart_2020_yrs_first,knn_2020_yrs_first)))>=5,5,
  round(rowMedians(cbind(rborist_2020_yrs_first,ranger_2020_yrs_first,
                         svm_2020_yrs_first))))
y1s2<-advanced_and_totals_eval_set %>%
  select(-contract_yrs) %>% add_column(contract_yrs=yrs_first_vec)

lin_2020_sal_second=predict_sal(lin_sal_second)
knn_2020_sal_second=predict_sal(knn_sal_second)
ranger_2020_sal_second=predict_sal(ranger_sal_second)
rborist_2020_sal_second=predict_sal(rborist_sal_second)
svm_2020_sal_second=predict_sal(svm_sal_second)

sal_second_vec=rowMedians(cbind(rborist_2020_sal_second,ranger_2020_sal_second,
                                lin_2020_sal_second,svm_2020_sal_second,
                                knn_2020_sal_second))
```

## Salary First, Years Second

The salary portion of the S1Y2 is equivalent to the one in Y1S2: take the median of all 5 models used (ranger, Rborist, svm, knn and linear). Since the S1Y2 didn't have trouble with max contracts, I take the median of all models except KNN, which has the lowest prediction accuracy by around 8%.

```r
lin_2020_sal_first=predict_sal(lin_sal,first=TRUE)
knn_2020_sal_first=predict_sal(knn_sal,first=TRUE)
ranger_2020_sal_first=predict_sal(ranger_sal,first=TRUE)
rborist_2020_sal_first=predict_sal(rborist_sal,first=TRUE)
svm_2020_sal_first=predict_sal(svm_sal,first=TRUE)

sal_first_vec=rowMedians(cbind(rborist_2020_sal_first,ranger_2020_sal_first,
                               lin_2020_sal_first,svm_2020_sal_first,
                               knn_2020_sal_first))

s1y2<-advanced_and_totals_eval_set %>%
  select(-first_year_percent_of_cap) %>%
  add_column(first_year_percent_of_cap=sal_first_vec)

lin_2020_yrs_second=predict_yrs(lin_yrs_second,first=FALSE)
rpart_2020_yrs_second=predict_yrs(rpart_yrs_second,first=FALSE)
ranger_2020_yrs_second=predict_yrs(ranger_yrs_second,first=FALSE)
rborist_2020_yrs_second=predict_yrs(rborist_yrs_second,first=FALSE)
svm_2020_yrs_second=predict_yrs(svm_yrs_second,first=FALSE)

yrs_second_vec=round(rowMedians(cbind(rborist_2020_yrs_second,
                                      ranger_2020_yrs_second,
                                      svm_2020_yrs_second,
                                      lin_2020_yrs_second,
                                      rpart_2020_yrs_second)))
```

## 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_S1Y2 | n |
|---:|---:|
| 0 | 5 |
| 1 | 90 |
| 2 | 55 |
| 3 | 26 |
| 4 | 5 |

| yrs_Y1S2 | n |
|---:|---:|
| 0 | 4 |
| 1 | 92 |
| 2 | 55 |
| 3 | 26 |
| 4 | 3 |
| 5 | 1 |

It doesn't really make any sense for only 4 to 5 players to not return from this year, with the rookie draft bringing in at least 30 new players. Let's take a random sample of players who are projected to receive one year contracts.

| player | yrs_Y1S2 | yr1_cap_percent_Y1S2 |
|---|---:|---:|
| Josh Gray | 1 | 0.0099228 |
| Michael Carter-Williams | 1 | 0.0246286 |
| Raul Neto | 1 | 0.0152454 |
| Frank Jackson | 1 | 0.0058673 |
| Josh Jackson | 1 | 0.0165391 |

| player | yrs_S1Y2 | yr1_cap_percent_S1Y2 |
|---|---:|---:|
| Max Strus | 1 | 0.0048668 |
| Kyle O'Quinn | 1 | 0.0124405 |
| Paul Watson | 1 | 0.0077279 |
| DaQuan Jeffries | 1 | 0.0071303 |
| Emmanuel Mudiay | 1 | 0.0212275 |

Aha! The problem is that the NBA has a minimum salary, but some players are projected to get even less than that. My solution to alleviate the problem:

- the projected salary cap (before the COVID-19 shutdown) was \$115 million
- the projected minimum contract (from RealGM) is \$1,523,320
- therefore, the minimum first year percent of cap in 2021 is 1.3246261%
- all players with less than the minimum first year percent of cap will have both the contract years and the first year cap percentage converted to zero

```
eval_set<-eval_set %>%
  mutate(yrs_Y1S2=ifelse(yr1_cap_percent_Y1S2<=0.01325,0,yrs_Y1S2)) %>%
  mutate(yr1_cap_percent_Y1S2=ifelse(yrs_Y1S2==0,0,yr1_cap_percent_Y1S2)) %>%
  mutate(yrs_S1Y2=ifelse(yr1_cap_percent_S1Y2<=0.01325,0,yrs_S1Y2)) %>%
  mutate(yr1_cap_percent_S1Y2=ifelse(yrs_S1Y2==0,0,yr1_cap_percent_S1Y2))
```

| yrs_Y1S2 | n |
|---:|---:|
| 0 | 52 |
| 1 | 44 |
| 2 | 55 |
| 3 | 26 |
| 4 | 3 |
| 5 | 1 |

| yrs_S1Y2 | n |
|---:|---:|
| 0 | 63 |
| 1 | 32 |
| 2 | 55 |
| 3 | 26 |
| 4 | 5 |

Much better, although it's odd that there's more two-year contracts given out than one-year contracts.

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 \$115 million.

```
eval_set<-eval_set %>%
  mutate(total_Y1S2=
           round(yr1_cap_percent_Y1S2*115*((1.05)^(yrs_Y1S2)-1)/0.05,2),
         total_S1Y2=
           round(yr1_cap_percent_S1Y2*115*((1.05)^(yrs_S1Y2)-1)/0.05,2))
```

**Player Option Decisions**

| player | Y1S2 Cap % | yrs_Y1S2 | total_Y1S2 | S1Y2 Cap % | yrs_S1Y2 | total_S1Y2 | 2021 Option |
|---|---|---|---|---|---|---|---|
| Gordon Hayward | 0.1788989 | 3 | 64.86 | 0.1812430 | 3 | 65.71 | 34.19 |
| Andre Drummond | 0.2221571 | 3 | 80.54 | 0.2000980 | 4 | 99.18 | 28.75 |
| Anthony Davis | 0.3014812 | 5 | 191.58 | 0.2868036 | 3 | 103.98 | 28.75 |
| Otto Porter | 0.0829733 | 2 | 19.56 | 0.0792502 | 2 | 18.68 | 28.49 |
| DeMar DeRozan | 0.2781874 | 3 | 100.85 | 0.2657557 | 3 | 96.35 | 27.74 |
| Nicolas Batum | 0.0185742 | 1 | 2.14 | 0.0260451 | 1 | 3.00 | 27.13 |
| Tim Hardaway Jr. | 0.1551284 | 3 | 56.24 | 0.1519132 | 3 | 55.07 | 18.98 |
| Evan Fournier | 0.2139512 | 4 | 106.05 | 0.1969257 | 4 | 97.61 | 17.00 |
| James Johnson | 0.0242445 | 1 | 2.79 | 0.0316826 | 1 | 3.64 | 15.83 |
| Kelly Olynyk | 0.1076968 | 3 | 39.04 | 0.0867777 | 2 | 20.46 | 12.20 |
| Tony Snell | 0.0694058 | 2 | 16.36 | 0.0766659 | 2 | 18.07 | 12.18 |
| Jerami Grant | 0.1321223 | 3 | 47.90 | 0.1139293 | 3 | 41.30 | 9.35 |
| Kentavious Caldwell-Pope | 0.1173450 | 3 | 42.54 | 0.1037241 | 3 | 37.60 | 8.49 |
| Jabari Parker | 0.0696616 | 2 | 16.42 | 0.0626748 | 2 | 14.78 | 6.50 |
| Rodney Hood | 0.0634435 | 2 | 14.96 | 0.0710488 | 2 | 16.75 | 6.00 |
| Robin Lopez | 0.0240291 | 1 | 2.76 | 0.0225893 | 1 | 2.60 | 5.01 |
| Avery Bradley | 0.0526229 | 2 | 12.41 | 0.0433099 | 2 | 10.21 | 5.01 |
| Enes Kanter | 0.0623681 | 2 | 14.70 | 0.0591980 | 2 | 13.96 | 5.01 |
| JaMychal Green | 0.0494627 | 2 | 11.66 | 0.0438166 | 2 | 10.33 | 5.01 |
| JaVale McGee | 0.0751264 | 2 | 17.71 | 0.0715280 | 2 | 16.86 | 4.20 |
| Stanley Johnson | 0.0165870 | 1 | 1.91 | 0.0000000 | 0 | 0.00 | 3.80 |
| Wesley Matthews | 0.0673011 | 2 | 15.87 | 0.0680355 | 2 | 16.04 | 2.69 |
| Rajon Rondo | 0.0283151 | 1 | 3.26 | 0.0301455 | 1 | 3.47 | 2.62 |
| Austin Rivers | 0.0531737 | 2 | 12.54 | 0.0473989 | 2 | 11.17 | 2.37 |
| Willie Cauley-Stein | 0.0742538 | 2 | 17.51 | 0.0773235 | 2 | 18.23 | 2.29 |
| Mike Muscala | 0.0198408 | 1 | 2.28 | 0.0138775 | 1 | 1.60 | 2.28 |
| James Ennis | 0.0441484 | 2 | 10.41 | 0.0332182 | 2 | 7.83 | 2.13 |
| Mario Hezonja | 0.0153306 | 1 | 1.76 | 0.0144861 | 1 | 1.67 | 1.92 |

While some might be surprised at the DeRozan and Drummond projections, we have to note that the model doesn't take into account intangibles like playing reputation, team fit, or willingness to take a reduced salary to be on a championship contender.

If Anthony Davis re-signs with the Lakers, he is eligible for eight percent raises rather than five. That would bring the Y1S2 total to $202 million, which is exactly what he is eligible for. In terms of the three year projection, Davis could sign that shorter contract with the last year being an option in order to be eligible for the 35% max contract after 10 years of service (but the model wouldn't know that).

Jerami Grant and surprisingly (to me at least) Evan Fournier look like players set to cash in after declining their option. Players like Nicolas Batum, James Johnson, Otto Porter and Stanley Johnson (who's predicted to be out of the league by the S1Y2 model) would be wise to accept their 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.

| player | Y1S2 Cap % | yrs_Y1S2 | total_Y1S2 | S1Y2 Cap % | yrs_S1Y2 | total_S1Y2 |
|---|---|---|---|---|---|---|
| Brandon Ingram | 0.2548324 | 4 | 126.31 | 0.2465663 | 4 | 122.21 |
| Danilo Gallinari | 0.2173188 | 3 | 78.79 | 0.2149018 | 3 | 77.91 |
| Fred VanVleet | 0.1961214 | 4 | 97.21 | 0.1888455 | 4 | 93.60 |
| Montrezl Harrell | 0.1920150 | 3 | 69.61 | 0.1782263 | 4 | 88.34 |
| Hassan Whiteside | 0.1941521 | 3 | 70.39 | 0.1723291 | 3 | 62.48 |

Assuming Ingram (a first-time All-Star this season) doesn't make one of the three All-NBA teams, his first year contract salary is capped at 25%, which is right in line with predictions. However, I (among many others) expect Ingram to get the full five year extension.

The Heat wanted to acquire Gallinari this trade deadline, but ultimately were unable to broker a deal. Gallinari's career has been marred by injuries, but he has played over 80% of games the past two seasons. He's averaging 19 points per game this year, on 44% field goal shooting and 40% three-point shooting.

After his breakout in last year's Eastern Conference Finals and NBA Finals, VanVleet has blossomed again in a starting role this season. Even so, Masai Ujiri and Bobby Webster might balk at paying him $ 24-25 million per year.

Harrell and Whiteside are the first cases of dissension between the two models. The S1Y2 model projects Harrell to get 4 years, while the Y1S2 model only projects three. For Whiteside, the difference is almost 8 million dollars less in salary under S1Y2.

| player | Y1S2 Cap % | yrs_Y1S2 | total_Y1S2 | S1Y2 Cap % | yrs_S1Y2 | total_S1Y2 |
|---|---|---|---|---|---|---|
| Marcus Morris | 0.1675723 | 3 | 60.75 | 0.1587575 | 3 | 57.56 |
| Joe Harris | 0.1539417 | 3 | 55.81 | 0.1586893 | 3 | 57.53 |
| Serge Ibaka | 0.1536285 | 3 | 55.70 | 0.1416515 | 3 | 51.35 |
| Bogdan Bogdanović | 0.1284454 | 3 | 46.57 | 0.1216268 | 3 | 44.09 |
| Jordan Clarkson | 0.1357167 | 3 | 49.20 | 0.1062462 | 3 | 38.52 |

Morris reneged on a verbal 2-year, $ 20 million contract with the Spurs to sign a 1-year, $ 20 million deal with the Knicks. He was the focal point of the Knicks offense and a hot commodity at the trade deadline, ultimately ending up with the championship-contending Clippers.

In February, Zach Lowe on his Lowe Post podcast thought that Joe Harris had a chance to double his current salary of $ 8 million. The models are even more optimistic, projecting his first year salary as over $ 17 million.

Bogdanovic has been mentioned as the second-best RFA in the class. He might be squeezed out of Sacramento, considering the recent contracts to Buddy Hield and Harrison Barnes as well as upcoming contracts to De'Aaron Fox and Marvin Bagley III.

Clarkson has the largest discrepancy in salary ($ 11 million) between the two models.

| player | Y1S2 Cap % | yrs_Y1S2 | total_Y1S2 | S1Y2 Cap % | yrs_S1Y2 | total_S1Y2 |
|---|---|---|---|---|---|---|
| Dāvis Bertāns | 0.1170676 | 3 | 42.44 | 0.1080289 | 3 | 39.16 |
| Goran Dragić | 0.1030788 | 2 | 24.30 | 0.1104918 | 2 | 26.05 |
| Dario Šarić | 0.1045437 | 3 | 37.90 | 0.1021354 | 3 | 37.03 |
| Christian Wood | 0.1121054 | 3 | 40.64 | 0.0925856 | 3 | 33.57 |
| Derrick Favors | 0.0981608 | 2 | 23.14 | 0.1064310 | 3 | 38.59 |

Bertans has shot the lights out this season, earning himself a raise over his current salary of $ 7 million.

Wood has flashed star-like potential since being inserted into the starting lineup in place of the departed Andre Drummond. But the key here is *12* games started. Wood played limited minutes in the first four months of the season. His contract situation, as well as that of Malik Beasley (who flourished after being traded to Minnesota mid-season), are intriguing to say the least.

## Conclusion

Using a combination of contract year stats and last-three-years stats, we set out to predict NBA free agent contracts for the 2020 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. We used six models: linear, k-nearest-neighbors, decision tree (only for predicting contract length), 2 random forests and a support vector machine. Anthony Davis, Brandon Ingram, DeMar DeRozan, Andre Drummond, Evan Fournier and Fred VanVleet are all projected to get above $ 90 million in total contract value.

The models suffer from being unable to quantify intangibles like playing reputation, team fit, 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 salary and contract length were correlated, maybe I should have predicted them as a tuple instead of sequentially (unfortunately, caret doesn't have that capability of multi-target regression). 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.

Further work could include looking at the contract years as a classification problem with 6 classes (0, 1, 2, 3, 4, 5). Another example of future work is to try more models, like boosting (in which models are added sequentially, with later models in the sequence attempting to correct the errors of earlier models). 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.