DSCapstone Report

Carlos Aguilar

06/23/2020

Introduction

This project will implement a predictor system for the ALL NBA team members for the National Basketball Association. At the end of an NBA season, the coaches, media and, players select the best players of the year to form three teams known as the ALL NBA teams. These teams have a spot for the Center position, two spots for the Forward positions and, two spots for the Guard positions. Therefore, if a player is Guard, he could only be eligible to take one of the Guard positions.

The data used in the project is a combination of two datasets. The first one is a dataset retrieved from [Kaggle NBA-players-stats] (https://www.kaggle.com/drgilermo/nba-players-stats) contains statistical information about all the players in the league since 1950. The dataset has 20376 observations and 53 variables, including the season's totals and advanced statistics.

The second one is a generated dataset from scraping the official NBA site listing the members of the ALL NBA teams since 1947. The code for scraping the website is written in Python and is beyond the scope of this project. However, if you are interested in taking a look at the code, you can find it on the file web_scraper.py in this repository. It is necessary to add that the program only recovers the members of the First and Second ALL NBA Teams because the Third team didn't exist until the late 80s. Also, both datasets can be found in the Data directory of this project.

To create the predictor system, we are going to analyze and wrangle the data from both datasets. Then, merge the datasets to identify the selected players for the ALL NBA teams, creating a binary classification problem. Next, determine the best predictors from the 53 variables. Finally, compare three different binary classification algorithms: logistic regression, knn, and decision trees to decide the best predicting algorithm.

Methods

First, we will install the libraries required in the project and load the datasets from the Data folder.

```
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")
if(!require(ggrepel)) install.packages("ggrepel", repos = "http://cran.us.r-project.org")
```

```
## Loading required package: ggrepel
```

```
seasons_stats <- read_csv("data/seasons_stats.csv", col_types = cols())
all_nba <- read_csv("data/all_nba_teams.csv", col_types = cols())
```

The previous code creates two Data Frames, **all_nba** and **seasons_stats**. The next step is to clean and merge the Data Frame. The **all_nba** needs a couple of replacements on the players' names to match the **seasons_stats** names. Additionally, in 1952 two players tie on the first team, which was marked on the NBA site and needs to be removed from the players' names. Finally, we add a variable call all_nba and set it to 1 for all the registers, indicating that all those players are members of one of the ALL NBA teams.

```r
# Cleaning all_nba table to join with seasons_stats
all_nba <- all_nba %>% mutate(all_nba=1)
all_nba$Player = gsub(" \\(tie\\)", "", all_nba$Player) # Remove tie text
# Changing name differences between data sets
all_nba$Player = gsub("Neil Johnson", "Neil Johnston", all_nba$Player)
all_nba$Player = gsub("KarlMalone", "Karl Malone", all_nba$Player)
all_nba$Player = gsub("Benard King", "Bernard King", all_nba$Player)
all_nba$Player = gsub("World B. Free", "World B.", all_nba$Player)
all_nba$Player = gsub("Jo Jo White", "Jo Jo", all_nba$Player)
all_nba$Player = gsub("Norm Van Lier", "Norm Van", all_nba$Player)
all_nba$Player = gsub("Gus Johnsson", "Gus Johnson", all_nba$Player)
all_nba$Player = gsub("Elgin baylor", "Elgin Baylor", all_nba$Player)
all_nba$Player = gsub("Richie Griffin", "Richie Guerin", all_nba$Player)
```

For the **seasons_stats** Data Frame, there is a little bit more work to be done. First, we need to remove the asterisk (*) that some names have at the end of the name. Second, we need to remove duplicated registers generated when a player changes teams. In theory, a player should only have one record in one particular season, but if we take a look at Dikembe Mutombo's entry for the 2001 season, we find three entries for that year.

```r
seasons_stats %>%
  filter(Player=='Dikembe Mutombo*' & Year==2001) %>%
  select(Year, Player, Tm, G)
```

```
## # A tibble: 3 x 4
##    Year Player           Tm        G
##   <dbl> <chr>            <chr> <dbl>
## 1  2001 Dikembe Mutombo* TOT      75
## 2  2001 Dikembe Mutombo* ATL      49
## 3  2001 Dikembe Mutombo* PHI      26
```

It turns out that when a player changes teams in the middle of a season, we find a register for each team. Furthermore, there is a third record accumulating the stats for the season; this can be verified by looking at the games played variable (G). If we sum the 26 games played at Philadelphia and the 49 games played in Atlanta, we get 75 games played for the whole season. Since we are going to predict if the player made the ALL NBA team at the end of the season, we don't care about the incomplete statistics, and we can remove those registers. Finally, the **seasons_stats** dataset has several columns in NA that we removed at the end of the code.

```r
# Cleaning seasons_stats data
# 1. Removing * from players name in seasons_stats
seasons_stats$Player = gsub("\\*", "", seasons_stats$Player)

# 2. Removing Player with multiple seasons, we just keep the TOT register
#    which has the stats for the complete season.
changed_players <- seasons_stats %>%
  filter(Tm=='TOT') %>%
  select(Year, Player) %>%
  inner_join(seasons_stats, by = c('Year', 'Player'))

stats_to_remove <- changed_players %>%
  filter(Tm!='TOT') %>%
```

```
  select(Year, Player, Tm)
seasons_stats <- seasons_stats %>%
  anti_join(stats_to_remove, by = c('Year', 'Player', 'Tm'))

# 3. Removing NA columns
seasons_stats <- seasons_stats %>%
  select(
    -'X1', -'3PAr', -'ORB%', -'DRB%', -'TRB%', -'AST%', -'STL%', -'BLK%', -'TOV%',
    -'USG%', -'blanl', -'blank2', -'OBPM', -'DBPM', -'BPM', -'VORP', -'3P',
    -'3PA', -'3P%', -'ORB', -'DRB', -'STL', -'BLK'
  )
```

Next, we join both tables by Year and Player, replace NAs for 0s, exclude seasons before 1952 because there is not enough information. The resulting table has information from 1952 to 2017. Additionally, we change the all_nba column to be a factor, facilitating the following training and remove the temporal Data Frames.

```
# Merge the two datasets
full_set <- seasons_stats %>%
  left_join(all_nba, by = c('Year', 'Player')) %>%
  filter(Year>=1952)
# Replace NAs with 0
full_set[is.na(full_set)] <- 0
# Make all_nba column a factor
full_set$all_nba <- as.factor(full_set$all_nba)
# Remove temporal objects
rm('changed_players', 'stats_to_remove')
```

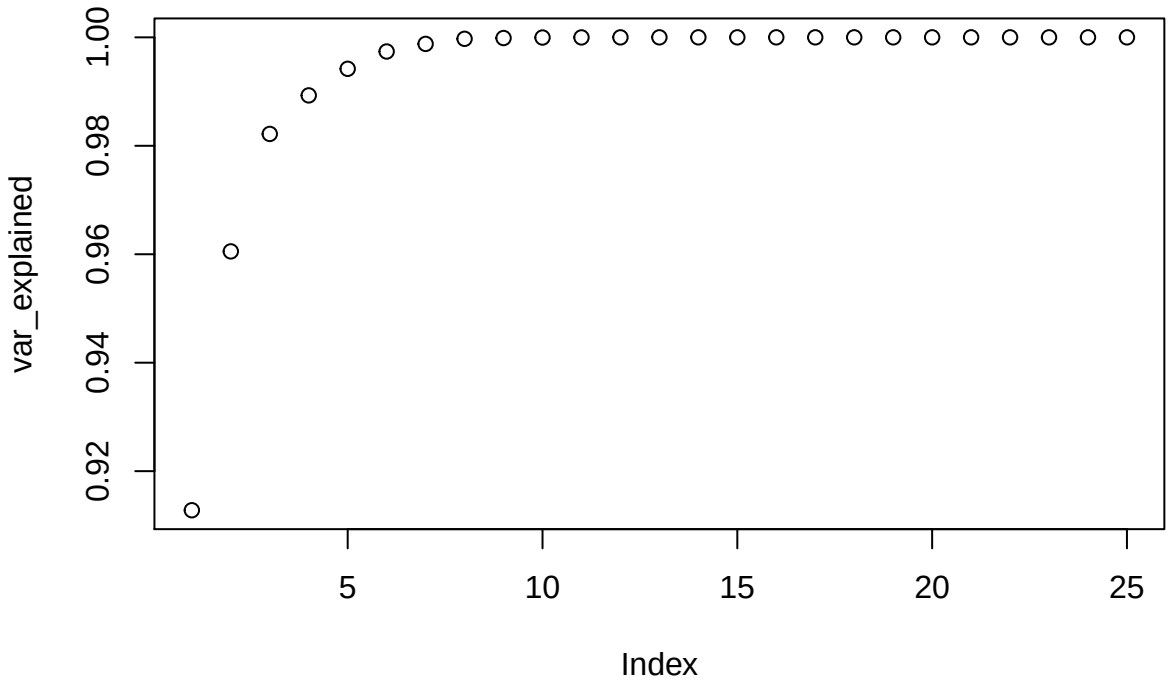
Now that we cleaned our data is time to start the analysis. The first step will be to split the **full_set** into a **train_set** and a **test_set**. Given the nature of the dataset, we are not going to create a random validation set. Instead, we are going to take the last three seasons available 2015, 2016, and 2017, train our models using all the seasons before that one, and then we are going to validate the model against these three seasons.

```
train_set <- full_set %>% filter(Year<2015)
test_set <- full_set %>% filter(Year>=2015)
```

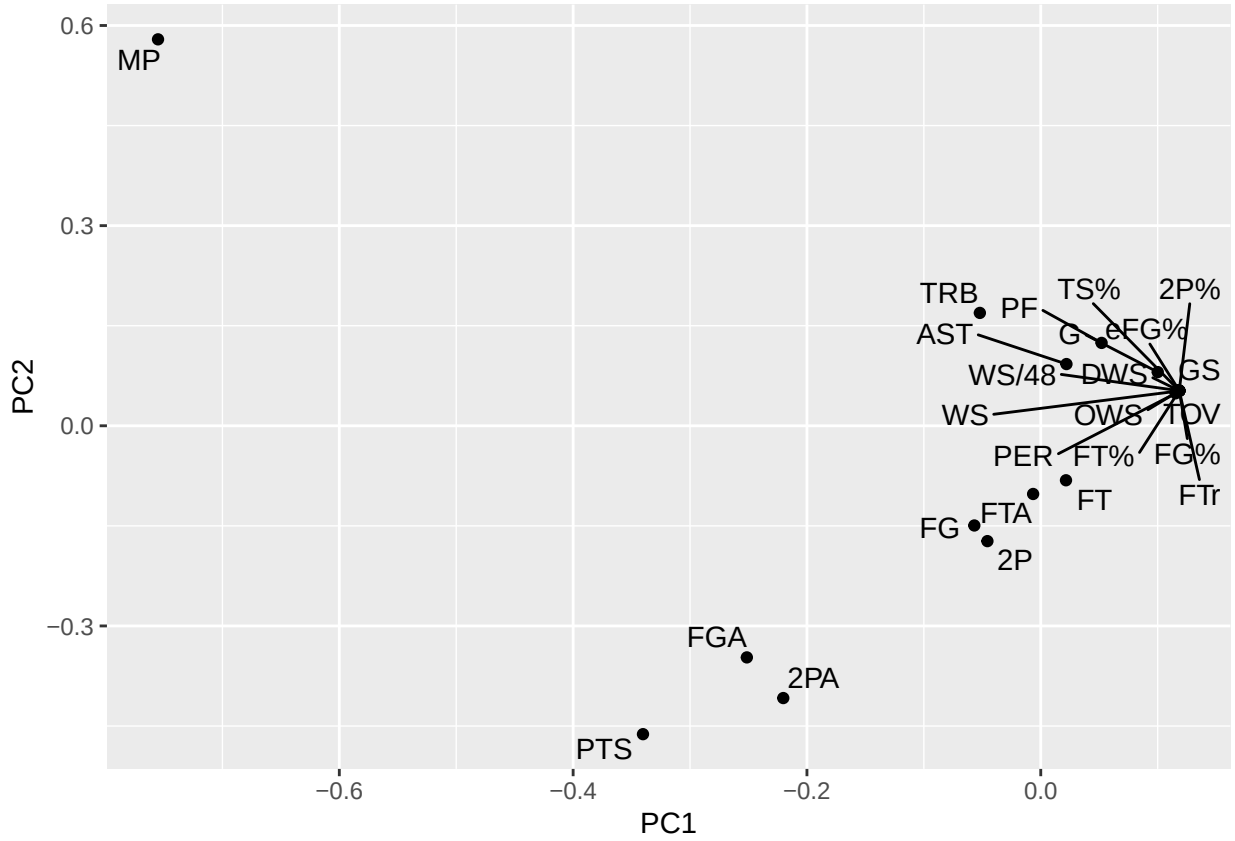
Next, we will use Principal Component Analysis to find the most representative variables in our data.

```
# Finding Principal Components using PCA (Principal Components Analysis)
train_matrix <- train_set %>% select(-Year, -Player, -Age, -Pos, -Tm, -all_nba) %>% as.matrix()
train_matrix <- sweep(train_matrix, 1, rowMeans(train_matrix))
pca <- prcomp(train_matrix)

# Looking how may variables explain the data
var_explained <- cumsum(pca$sdev^2/sum(pca$sdev^2))
plot(var_explained)
```




```
# Looking for information clusters
pcs <- data.frame(pca$rotation, name = colnames(train_matrix))
pcs %>%  ggplot(aes(PC1, PC2)) + geom_point() +
  geom_text_repel(aes(PC1, PC2, label=name),
                  data = pcs)
```



The charts hint two conclusions: The first one being that with just four variables, we could explain more than 98% of the data. The second one is that we have three defined clusters, one defined by the minutes played (MP), the second determined by the points scored (PTS), and the third one defined by everything else. Taking a look at the statistics found in the third cluster, we can see that most of them are related to the offensive part of the game: AST: assistances to score, TS%: true shooting percentage, FG%: field goal average, FT%: free-throws percentage, etc. However, two statistics may impact the final result, OWS Offensive Win Shares, and DWS Defensive Win Shares. Both of them encapsulates several other statistics into one number that express the player's contribution to wins in the offensive and the defensive ends, respectively. Therefore, we will add OWS and DWS to the MP and PTS variables to train our models.

Each observation on the **train_set** corresponds to the statistics of a player in a particular year. For example, if the player played 12 seasons, they should have 12 observations on the data set. The variable PTS has the total points scored on that season, MP the total minutes played, OWS is the Offensive Win Share contribution, and DWS the Defensive Win Share contribution. Finally, the all_nba variable has a 1 if the player made it to one of the ALL NBA teams for that season and 0 if they didn't. We will ignore the other columns for the rest of the project.

Since the outcome variable (all_nba) is binary, we will use three binary classification algorithms to find the best result: logistic regression, Knn, and decision trees. We also will use the F1 score to measure the algorithm success, analyzing the confusion matrix because the data's prevalence is high since most of the players don't make it to the ALL NBA teams at the end of the season. We start with the logistic regression approach.

```
# Generating a basic logistic regression model
fit <- glm(all_nba ~ MP + PTS + OWS + DWS, data = train_set, family = binomial)
y_hat <- ifelse(predict(fit, train_set) > 0, 1, 0) %>% factor()
# F1 Score
```

```r
F_meas(y_hat, factor(train_set$all_nba))
```

```
## [1] 0.9895909
```

```r
confusionMatrix(y_hat, train_set$all_nba %>% factor())
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 17778   286
##          1    88   345
##
##                Accuracy : 0.9798
##                  95% CI : (0.9776, 0.9818)
##     No Information Rate : 0.9659
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6385
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.9951
##             Specificity : 0.5468
##          Pos Pred Value : 0.9842
##          Neg Pred Value : 0.7968
##              Prevalence : 0.9659
##          Detection Rate : 0.9611
##    Detection Prevalence : 0.9766
##       Balanced Accuracy : 0.7709
##
##        'Positive' Class : 0
##
```

Then we try with knn

```r
# Generating a basic Knn model
train_knn <- train(all_nba ~ MP + PTS + OWS + DWS, method = "knn", data = train_set)
knn_y_hat <- predict(train_knn, train_set)
# F1 Score
F_meas(knn_y_hat, factor(train_set$all_nba))
```

```
## [1] 0.9869726
```

```r
confusionMatrix(knn_y_hat, train_set$all_nba)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 17766   369
```

```
##          1   100   262
##
##                Accuracy : 0.9746
##                  95% CI : (0.9723, 0.9769)
##     No Information Rate : 0.9659
##     P-Value [Acc > NIR] : 4.009e-12
##
##                   Kappa : 0.5156
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.9944
##             Specificity : 0.4152
##          Pos Pred Value : 0.9797
##          Neg Pred Value : 0.7238
##              Prevalence : 0.9659
##          Detection Rate : 0.9605
##    Detection Prevalence : 0.9804
##       Balanced Accuracy : 0.7048
##
##        'Positive' Class : 0
##
```

Finally we try and plot the decision tree.

```
# Using rpart to create a decision tree
train_rpart <- train(all_nba ~ MP + PTS + OWS + DWS, method = "rpart", data = train_set)
rpart_y_hat <- predict(train_rpart, train_set)
# F1 score
F_meas(rpart_y_hat, factor(train_set$all_nba))
```

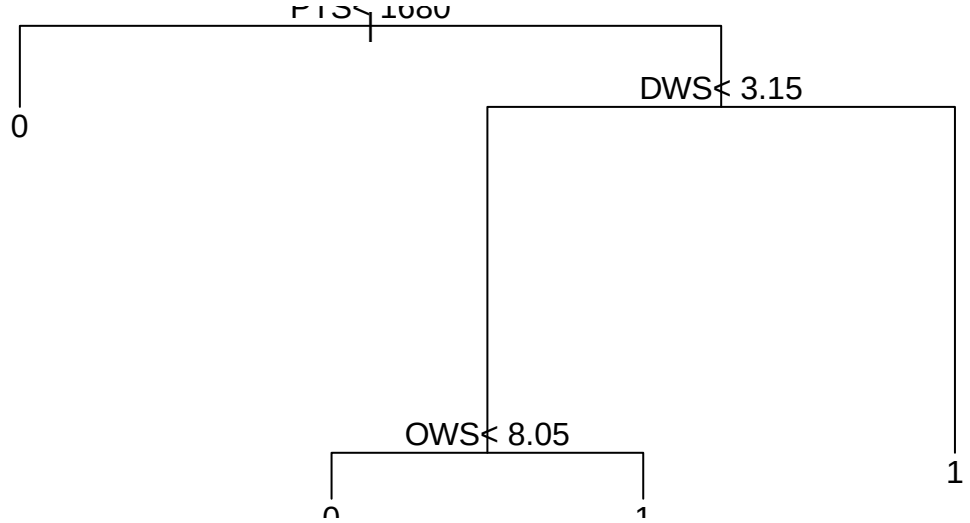
```
## [1] 0.9872931
```

```
confusionMatrix(rpart_y_hat, train_set$all_nba %>% factor())
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 17715   305
##          1   151   326
##
##                Accuracy : 0.9753
##                  95% CI : (0.973, 0.9775)
##     No Information Rate : 0.9659
##     P-Value [Acc > NIR] : 5.886e-14
##
##                   Kappa : 0.576
##
##  Mcnemar's Test P-Value : 7.785e-13
##
##             Sensitivity : 0.9915
##             Specificity : 0.5166
```

```
##          Pos Pred Value : 0.9831
##          Neg Pred Value : 0.6834
##              Prevalence : 0.9659
##          Detection Rate : 0.9577
##    Detection Prevalence : 0.9742
##       Balanced Accuracy : 0.7541
##
##        'Positive' Class : 0
##
```

```
# Plotting the tree
plot(train_rpart$finalModel)
text(train_rpart$finalModel)
```



Results

Then, we compare the algorithms using the **test_set**

```
lg_results <- ifelse(predict(fit, test_set) > 0, 1, 0) %>% factor()
knn_results <- predict(train_knn, test_set)
dt_results <- predict(train_rpart, test_set)

results <- tibble(method=character(), score=double())
results <- bind_rows(results,
```

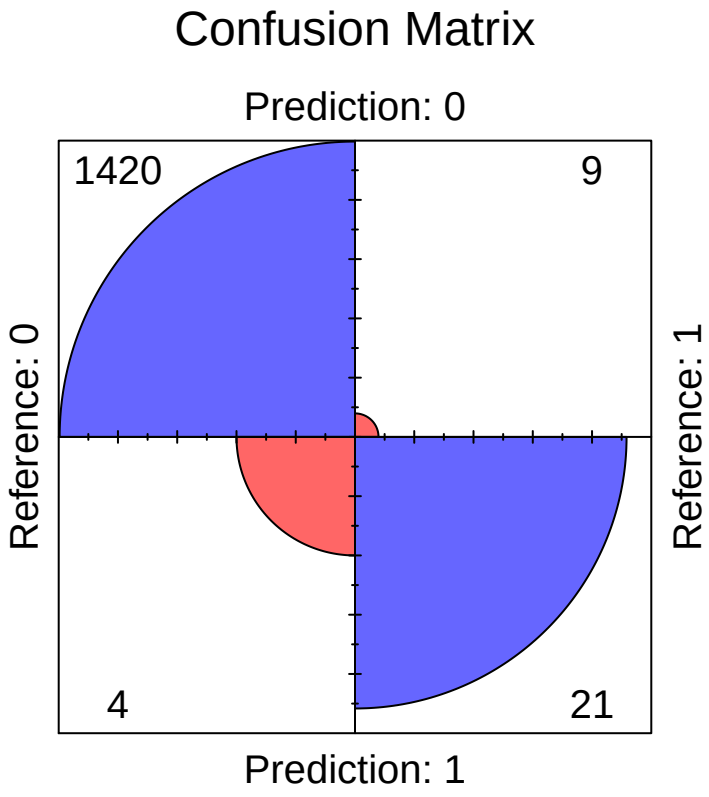
```
  tibble(method='Logistic Regression', score=F_meas(lg_results, factor(test_set$all_nba))),
  tibble(method='Knn', score=F_meas(knn_results, factor(test_set$all_nba))),
  tibble(method='Decision Tree', score=F_meas(dt_results, factor(test_set$all_nba)))
)

results
```

```
## # A tibble: 3 x 2
##   method              score
##   <chr>               <dbl>
## 1 Logistic Regression 0.995
## 2 Knn                 0.992
## 3 Decision Tree       0.992
```

If we plot the confusion matrix we can see that even if the accuracy is really high, there is a lot of room for improvement in the algorithm

```
# Plotting the confusion matrix
fourfoldplot(confusionMatrix(lg_results, test_set$all_nba)$table,
             color = c("#FF6666", "#6666FF"),
             conf.level = 0, margin = 1, main = "Confusion Matrix")
```



There are several changes we could make that will be explained in the next section

Conclusions

In this project, we created a tool that predicts with extreme accuracy the members of the ALL NBA teams in a season given their season stats. From a poll of 1954 observations, the instrument was able to predict which players would be on the ALL NBA team on their correspondent year with an accuracy higher than 99%. Since the presentation of the awards are around two months after the season ends, this project can deliver insight about the chosen players a lot earlier than the awards ceremony. On the other hand, several improvements could be made to improve the results and make more sense; we list some of them below as futute work for the project.

1. Select ten players per year; the tool uses a statistics threshold without considering how many players are awarded per year.
2. Considering position bias, generally, the Guards are judge different than Forwards and Centers. For example, the guards probably have more assists and the centers more rebounds.
3. Incorporate defensive stats. DWS may not be enough to incorporate the defense part of the game into the analysis. The most significant example of this is Ben Wallace, an 00s center, which excels at defensive tasks while not so good on the offensive end. Sadly, defensive data was not provided by Kaggle's dataset.