부산대학교 통계학과 AIR

제 3회 KSPO X KBL

(국민체육진흥공단) (Korea Basketball League)

프로농구 데이터활용 경진대회

X

AIR 구성원 :
정재호
김도희
하도연

제출 일자 : 2019. 11. 30. (토)

목 차

✓주 제

" KBL 3년 시즌 Data를 기반으로
All-Star 선수 12명을 선발해 주세요.
단, 용병은 제외입니다. "

✓목 적

인기 투표가 아닌 Data 기반 우수 선수 또는 데이터로
증명할 수 있는 주제(캐릭터)가 있는 선수 선발

안정성 중시 선발 !!

✓분석 방향

포지션 별 가장 능력이 좋은 선수들은 어떤 선수 들인가?
(즉, 각 포지션에서의 능력치를 어떻게 평가할 것인가?)
+
All Star 전에서 자신의 능력치를 그대로 발휘할 것인가?
(즉, 경기 시에 환경의 악영향을 받을 가능성이 높은 선수는 누구인가?)
+
All Star 전에서 더 높은 능력치를 발휘할 수 있는가?
(즉, 각 선수들의 성장율이 어떻게 될 것인가?)

> 필요 패키지 설치

```
########## install.packages###########
install.packages('readxl')
install.packages('randomForest')
isntall.packages('caret')
install.packages('xlsx')
install.packages('prediction')  #for using prediction ftn
install.packages('ggplot2')
install.packages('dplyr')
install.packages('hrbrthemes')  # in ggplot2(linear trend)
install.packages('car') # vif ftn
install.packages('MLmetrics') # F1_Score ftn
install.packages('corrplot')
install.packages('car')
```

> 시즌 별 데이터 불러오기

```
> d1 <- readxl::read_excel("2-3. 2016-2017시즌_경기기록데이터.xlsx",
+                          sheet = "PLAYER_DAILY_LIST",
+                          col_names = T)
> d2 <- readxl::read_excel(path = "2-3. 2017-2018시즌_경기기록데이터.xlsx",
+                          sheet = "PLAYER_DAILY_LIST",
+                          col_names = T)
> d3 <- readxl::read_excel(path = "2018-2019시즌_경기기록데이터.xlsx",
+                          sheet = "PLAYER_DAILY_LIST",
+                          col_names = T)
```

> 승패여부를 나타내는 win 변수 생성

```
> d1$score<-as.integer(d1$score)
> winner=rep(0,270)
> win=rep(0,6480)
> for (i in 1:270){
+   game = d1[ which(d1[,3]==i) , ]
+   team1=unique(game$team_code)[1]
+   team2=unique(game$team_code)[2]
+   score_team1=sum (game[ which(game$team_code==team1), 38])
+   score_team2=sum (game[ which(game$team_code==team2), 38])
+   if (score_team1 > score_team2){
+     winner[i]=team1
+   }else{
+     winner[i]=team2
+   }
+   win[which(game$team_code==winner[i])+24*(i-1)]=1
+   win[which(game$team_code!=winner[i])+24*(i-1)]=0
+ }
> d1 <- cbind(d1,win)
```

각 선수들의 능력치를 평가하는 데에 있어서,
궁극적인 평가 방법이 될 수 있는 승패여부에 관한 데이터가 추가로 필요하다고 생각하여 생성.
각 포지션 별 승패 여부에 관해 영향을 미치는 변수들을 알아내기 위함.

* 해당 코드는 2016-2017년도의 경기 기록 데이터인 d1에서 win변수를 추가로 생성하는 과정이며, 각각 17-18년도와 18-19년도의 데이터인 d2 및 d3에서도 동일 과정으로 win 변수를 추가하였음.

> 데이터 병합 & 전처리

```
> seasonidx <- which(d3$season_code==31)
> d3 <- d3[-seasonidx,]
> gfindex<-which(d3$pos=='GF')
> d3$pos[gfindex]<-'GD'
> dpos <- which(d3$pos=="AB" | d3$pos == "AA" | d3$pos=="BC" | d3$pos == "AC")
> d3 <- d3[-dpos,]
```

➔ 17-18년도와 18-19년도에 중복하여 존재하던 season_code==31에 대한 중복된 값 제거

➔ pos 변수에 대한 오타 및 부정확한 정보를 담은 관측치 수정 및 제거

> 데이터 병합 & 전처리

```
> d<-rbind(d1,d2,d3)
> dim(d); dim(d1); dim(d2); dim(d3)
[1] 19440    43
[1] 6480   43
[1] 6480   43
[1] 6480   43
```

➔ 데이터 병합

```
> ##### 필요없는 데이터 삭제
> sum(!d$game_code=='01') # game_code 에 저장된 값 모두 '01'로 동일 => 따라서, 해당 변수 제거
[1] 1
> delete<-which(colnames(d)=='game_code')
> d<-d[,-delete]
>
> sum(!complete.cases(d$back_num))  # 결측치도 많이 포함할 뿐더러 등번호는 무의미한 변수라고 인지하여 제거
[1] 0
> delete<-which(colnames(d)=='back_num')
> d<-d[,-delete]
>
> sum(which(d$fb!='0')) # fb 변수에 저장된 값이 모두 '0'으로 동일 => 따라서, 해당 변수 제거
[1] 0
> delete<-which(colnames(d)=='fb')
> d<-d[,-delete]
```

➔ 'game_code', 'back_num', 'fb'
3가지의 무의미한 변수 제거

> 데이터 병합 & 전처리

```
> d$inputtime[1:30] # 입력시간은 무의미한 변수라고 인지하여 제거
 [1] "2016-10-22 15:53:06.670" "2016-10-22 15:53:06.670" "2016-10-22 15:53:06.670" "2016-10-22 15:5
 [5] "2016-10-22 15:53:06.670" "2016-10-22 15:53:06.670" "2016-10-22 15:53:06.670" "2016-10-22 15:5
 [9] "2016-10-22 15:53:06.670" "2016-10-22 15:53:06.670" "2016-10-22 15:53:06.670" "2016-10-22 15:5
[13] "2016-10-22 15:53:06.670" "2016-10-22 15:53:06.670" "2016-10-22 15:53:06.670" "2016-10-22 15:5
[17] "2016-10-22 15:53:06.670" "2016-10-22 15:53:06.670" "2016-10-22 15:53:06.670" "2016-10-22 15:5
[21] "2016-10-22 15:53:06.670" "2016-10-22 15:53:06.670" "2016-10-22 15:53:06.670" "2016-10-22 15:5
[25] "2016-10-22 17:47:35.793" "2016-10-22 17:47:35.793" "2016-10-22 17:47:35.793" "2016-10-22 17:4
[29] "2016-10-22 17:47:35.793" "2016-10-22 17:47:35.793"
> delete<-which(colnames(d)=='inputtime')
> d<-d[,-delete]
>
> delete=which(colnames(d)=='idf') # idf 변수도 모든 관측값에서 0의 값을 가져서 제거
> d<-d[,-delete]
>
> delete=which(colnames(d)=='foul_tot') # foul_tot 변수가 wof변수와 woft변수의 합과 동일함을 인지하여 제거
> d<-d[,-delete]
```

➔ 'inputtime', 'idf', 'foul_tot'
3가지의 무의미한 변수 추가로 제거

- foul_tot(한경기 총 파울) = wft(파울_자유투유) + woft(파울_자유투무)
에 대한 수식을 만족.
즉, 우변에 존재하는 두 변수만으로도 좌변이 갖는 정보를 담아낼 수 있으므로 제거

01

02

03

> 변수 형태 지정

```
> str(d)
'data.frame':   16626 obs. of  37 variables:
 $ season_code: chr  "29" "29" "29" "29" ..
 $ game_no    : chr  "1" "1" "1" "1" ...
 $ team_code  : chr  "60" "60" "30" "30" ..
 $ home_away  : chr  "2" "2" "1" "1" ...
 $ player_no  : chr  "260005" "290248" "290
 $ pos        : chr  "C" "C" "C" "C" ...
 $ away_team  : chr  "30" "30" "60" "60" ..
 $ start_flag : chr  "0" "1" "0" "1" ...
 $ play_min   : chr  "7" "14" "10" "35" ...
 $ play_sec   : chr  "53" "45" "50" "48" ..
 $ fg         : chr  "0" "3" "3" "2" ...
 $ fg_a       : chr  "0" "6" "6" "5" ...
 $ ft         : chr  "0" "0" "0" "1" ...
 $ ft_a       : chr  "0" "1" "0" "2" ...
 $ threep     : chr  "0" "1" "1" "2" ...
 $ threep_a   : chr  "0" "1" "1" "5" ...
 $ dk         : chr  "0" "1" "0" "0" ...
 $ dk_a       : chr  "0" "1" "0" "0" ...
 $ pp         : chr  "0" "3" "3" "2" ...
 $ pp_a       : chr  "0" "6" "6" "5" ...
 $ o_r        : chr  "0" "3" "1" "3" ...
 $ d_r        : chr  "2" "1" "0" "4" ...
 $ a_s        : chr  "0" "0" "0" "4" ...
 $ s_t        : chr  "0" "0" "0" "1" ...
 $ b_s        : chr  "0" "0" "1" "0" ...
 $ gd         : chr  "0" "0" "0" "0" ...
 $ t_o        : chr  "1" "0" "2" "0" ...
 $ wft        : chr  "0" "1" "1" "1" ...
 $ woft       : chr  "2" "0" "2" "1" ...
 $ tf         : chr  "0" "0" "0" "0" ...
 $ ef         : chr  "0" "0" "0" "0" ...
 $ p_score    : chr  "0" "6" "6" "4" ...
 $ score      : int  0 9 9 11 5 30 6 0 0 17
 $ inout      : chr  "3" "3" "3" "3" ...
 $ inout1     : chr  "3" "4" "2" "3" ...
 $ fo         : chr  "0" "0" "0" "0" ...
 $ win        : num  0 0 1 1 1 1 1 1 1 0 ..
```

```
> str(d)
'data.frame':   16626 obs. of  37 variables:
 $ season_code: Factor w/ 3 levels "29","31","33": 1 1 1 1 1 1
 $ game_no    : Factor w/ 270 levels "1","10","100",..: 1 1 1 1
 $ team_code  : Factor w/ 10 levels "06","10","16",..: 8 8 4 4
 $ home_away  : Factor w/ 2 levels "1","2": 2 2 1 1 1 1 1 1 1 2
 $ player_no  : Factor w/ 287 levels "210074","210080",..: 14 3
 $ pos        : Factor w/ 3 levels "C","FD","GD": 1 1 1 1 2 2 2
 $ away_team  : Factor w/ 10 levels "06","10","16",..: 4 4 8 8
 $ start_flag : Factor w/ 2 levels "0","1": 1 2 1 2 2 2 2 1 1 1
 $ play_min   : int  7 14 10 35 24 31 24 3 13 30 ...
 $ play_sec   : int  5 45 50 48 15 58 33 14 37 36 ...
 $ fg         : num  0 3 3 2 1 13 0 0 0 6 ...
 $ fg_a       : num  0 6 6 5 4 25 0 0 1 11 ...
 $ ft         : int  0 0 0 1 0 4 0 0 0 2 ...
 $ ft_a       : int  0 1 0 2 0 6 0 0 0 2 ...
 $ threep     : int  0 1 1 2 1 0 2 0 0 1 ...
 $ threep_a   : int  0 1 1 5 1 2 4 0 2 4 ...
 $ dk         : int  0 1 0 0 0 1 0 0 0 2 ...
 $ dk_a       : int  0 1 0 0 0 1 0 0 0 2 ...
 $ pp         : int  0 3 3 2 0 8 0 0 0 5 ...
 $ pp_a       : int  0 6 6 5 2 14 0 0 0 6 ...
 $ o_r        : int  0 3 1 3 0 2 1 0 0 1 ...
 $ d_r        : int  2 1 0 4 0 9 6 0 1 9 ...
 $ a_s        : int  0 0 0 4 6 2 1 0 0 1 ...
 $ s_t        : int  0 0 0 1 0 0 1 0 1 1 ...
 $ b_s        : int  0 0 1 0 0 0 0 0 0 3 ...
 $ gd         : int  0 0 0 0 0 0 0 0 0 0 ...
 $ t_o        : int  1 0 2 0 4 0 0 0 0 5 ...
 $ wft        : int  0 1 1 1 1 0 1 0 1 0 ...
 $ woft       : int  2 0 2 1 1 1 0 0 1 1 ...
 $ tf         : int  0 0 0 0 0 0 0 0 0 0 ...
 $ ef         : int  0 0 0 0 0 0 0 0 0 0 ...
 $ p_score    : int  0 6 6 4 0 16 0 0 0 10 ...
 $ score      : int  0 9 9 11 5 30 6 0 0 17 ...
 $ inout      : int  3 3 3 3 3 2 1 2 3 2 ...
 $ inout1     : int  3 4 2 3 3 3 2 2 2 1 ...
 $ fo         : int  0 0 0 0 0 0 0 0 0 0 ...
 $ win        : Factor w/ 2 levels "0","1": 1 1 2 2 2 2 2 2 2 1
```

> 율 변수 생성

```
> ##### 율 데이터 생성
> n<-dim(d)[1] ; p<-dim(d)[2]
> fg_r<-rep(0,n) ; ft_r<-rep(0,n) ; threep_r<-rep(0,n) ; dk_r<-rep(0,n) ; pp_r<-rep(0,n)
> d<-cbind(d,fg_r,ft_r,threep_r,dk_r,pp_r)  # 5가지의 새로운 율변수 생성
>
> fg_a0<-which(d$fg_a==0)
> d$fg_r[-fg_a0]<-d$fg[-fg_a0]/d$fg_a[-fg_a0]
> d$fg_r[fg_a0]<-0
>
> ft_a0<-which(d$ft_a==0)
> d$ft_r[-ft_a0]<-d$ft[-ft_a0]/d$ft_a[-ft_a0]
> d$ft_r[ft_a0]<-0
>
> threep_a0<-which(d$threep_a==0)
> d$threep_r[-threep_a0]<-d$threep[-threep_a0]/d$threep_a[-threep_a0]
> d$threep_r[threep_a0]<-0
>
> dk_a0<-which(d$dk_a==0)
> d$dk_r[-dk_a0]<-d$dk[-dk_a0]/d$dk_a[-dk_a0]
> d$dk_r[dk_a0]<-0
>
> pp_a0<-which(d$pp_a==0)
> d$pp_r[-pp_a0]<-d$pp[-pp_a0]/d$pp_a[-pp_a0]
> d$pp_r[pp_a0]<-0
```

➔ 성공과 시도의 정보를 담은 변수들에 대해 시도 대비 성공율의 정보를 담은 5가지의 새로운 율변수 생성

➔ 최대한 적은 변수들로 최대한의 정보를 담아내기 위하여 변수들의 차원 축소를 실시하기 위하여 생성함.

01

02

03

> 시간 데이터 변수 생성

```
> ##### 시간 데이터 생성
> d$playtime<-d$play_min*60+d$play_sec
> d$playtime<-as.integer(d$playtime)
> ##### playtime==0 인 선수 제거
> playtime0<-which(d$play_min=='0' & d$play_sec=='0')
> d<-d[-playtime0,]
```

➔ 'play_min' 과 'play_sec' 에 대한 정보를 하나의 playtime 변수로 나타냄

> 16-17년도의 fo(피파울) 값 예측

```
> #fo 값 채워넣기(by using RF with Factorizing)
> season29<-which(d$season_code==29)
> d23<-d[-season29,]
> d23$fo <- as.factor(d23$fo)
> rffactidx <- which(colname=='season_code' |colname=='game_no'|colname=='player_no'|colname=='away_team'|c
+                    |colname=='fg_r'|colname=='ft_r'|colname=='threep_r'|colname=='dk_r'|colname=='pp_r')
> rffact <- d23[,-rffactidx]
> set.seed(201711505)
> rffactrf <- randomForest(fo~., data=rffact)
> set.seed(201711505)
> rffact_1 <- predict(rffactrf, d[season29,], type='response')
> table(rffact_1)
rffact_1
   0    1    2    3    4    5    6    7    8    9   10   11   12   13   14   16
2581  988  812  410  312  212  103   21   29    1    5    2    1    0    0    0
```

➔ 16-17년도를 제외한 다른 시즌 경기의 피파울 값은 자료 조사가 되었음.
선수의 능력치를 평가하기 위해서는 피파울 횟수는 매우 중요한 요소가 되며,
randomForest 분석 방법을 이용한 예측값을 사용하기로 결정.

> 16-17년도의 예측된 값을 포함한 fo(피파운) 값 분포 확인

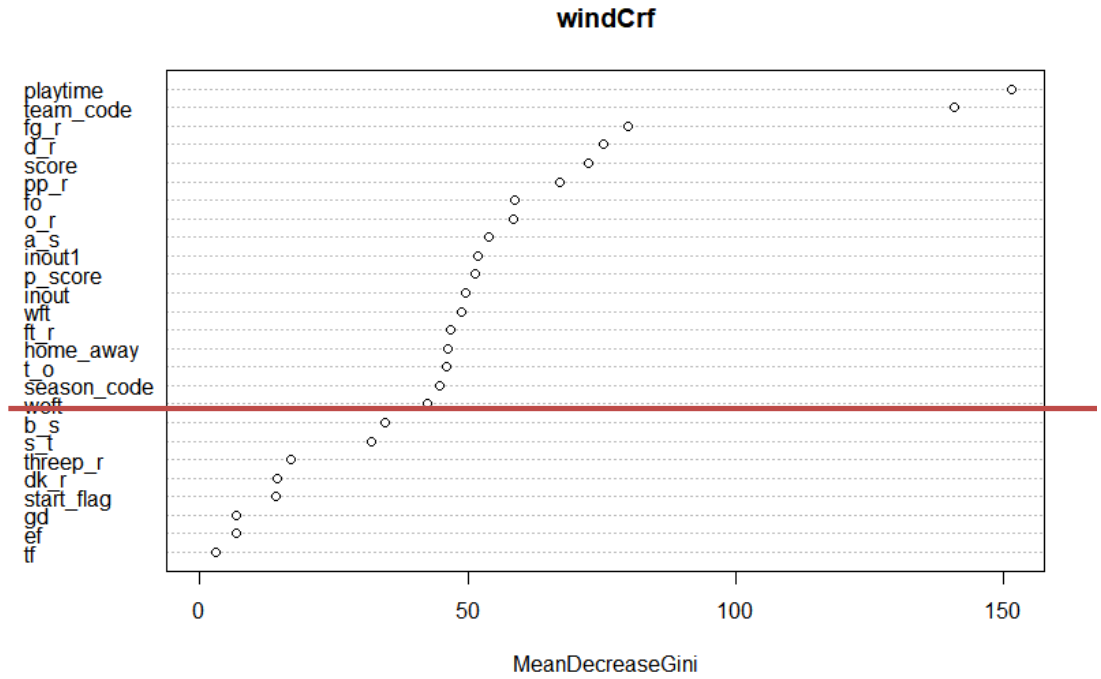
```
> d$fo[season29]<-rffact_1
> dim(d)
[1] 16626    43
> summary(d$fo)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  0.000   1.000   1.000   1.996   3.000  16.000
> d$fo <- as.integer(d$fo)
```

➔ 최소값은 0회, 최대값은 16회까지 나타나는 것을 알 수 있다.

Q1. 포지션 별 가장 능력이 좋은 선수들은 어떤 선수 들인가?
(즉, 각 포지션에서의 능력치를 어떻게 평가할 것인가?)

A1. 각 포지션 별 우승을 결정짓는 데에 중요한 변수들 결정
➔
우승 여부의 불순도를 줄이는 정도를 나타내는 Gini 계수로 각 포지션 별 중요 변수 결정 및 가중치 결정
➔
Radar Chart를 이용하여 가중치들의 부호 결정
➔
포지션 별로 선수들의 능력치 평가 지표가 되는 수식 결정

> 포지션 C의 Gini 계수 생성



➔ 실선까지의 변수 선택

```
> sort(varImpPlot(windCrf)[,1], decreasing=T)
   playtime   team_code        fg_r         d_r       score       pp_r          fo         o_r         a_s      inout1
151.543808  140.829525   79.791871   75.332084   72.401295   67.199769   58.648707   58.569915   53.972680   51.788820
    p_score       inout         wft        ft_r  home_away         t_o season_code        woft         b_s         s_t
 51.413745   49.649301   48.740513   46.792426   46.173187   46.082477   44.655723   42.561359   34.595226   32.116463
   threep_r        dk_r  start_flag          gd          ef          tf
 16.900545   14.353161   14.129491    6.935297    6.843794    2.933853
```

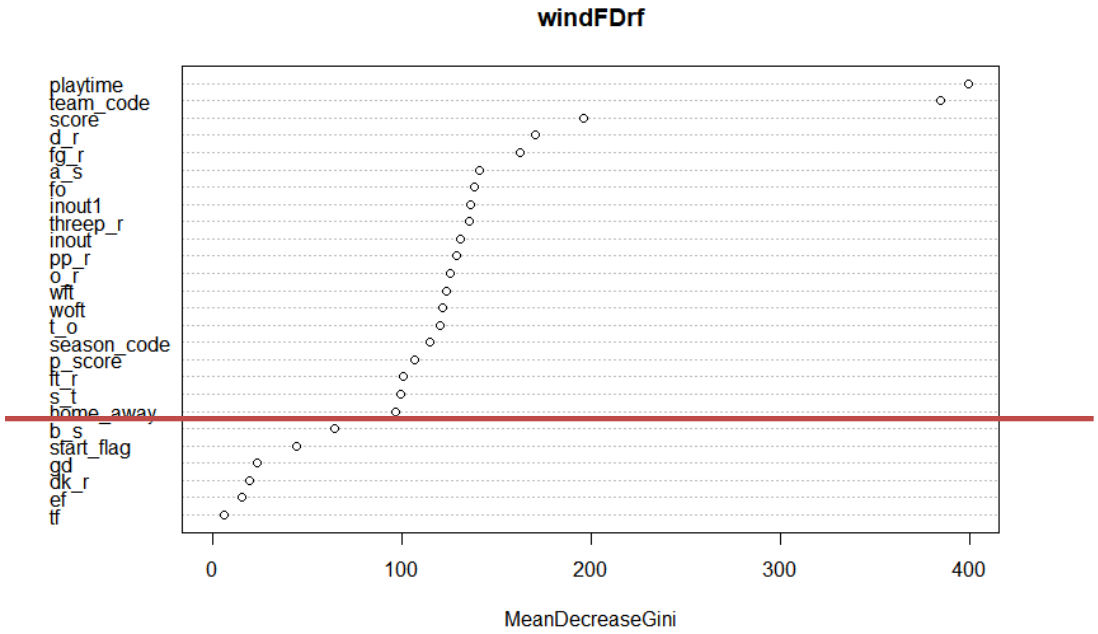
➔ 각 변수에 대한 지니계수 확인

01

02

03

> 포지션 FD의 Gini 계수 생성



➔ 실선까지의 변수 선택

```
> sort(varImpPlot(windFDrf)[,1])
        tf         ef       dk_r         gd start_flag        b_s  home_away        s_t       ft_r    p_score
  5.714936  14.992846  19.098085  23.230792  44.335552  64.458332  96.577264  99.268552 100.312672 106.559219
season_code        t_o       woft       wft        o_r       pp_r      inout    threep_r      inout1         fo
114.468264 119.889349 121.316648 123.572511 125.170657 129.021764 130.753824 135.399479 135.846677 137.895666
        a_s       fg_r        d_r      score  team_code   playtime
140.824525 162.555495 170.031018 195.860992 384.682062 399.540797
```

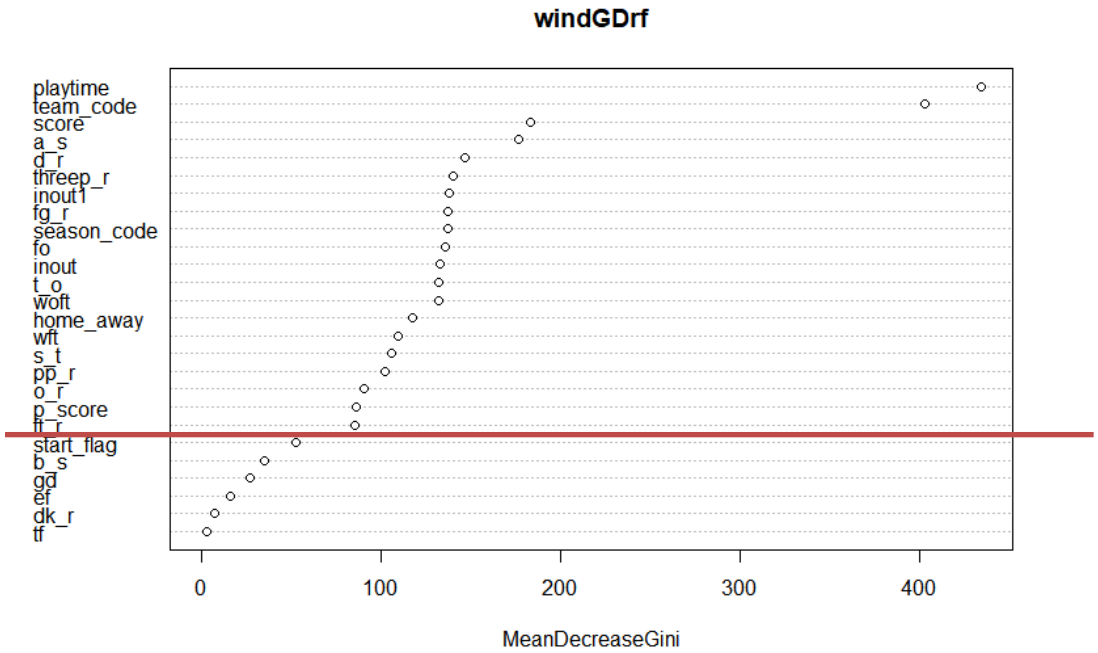
➔ 각 변수에 대한 지니계수 확인

01

02

03

> 포지션 GD의 Gini 계수 생성

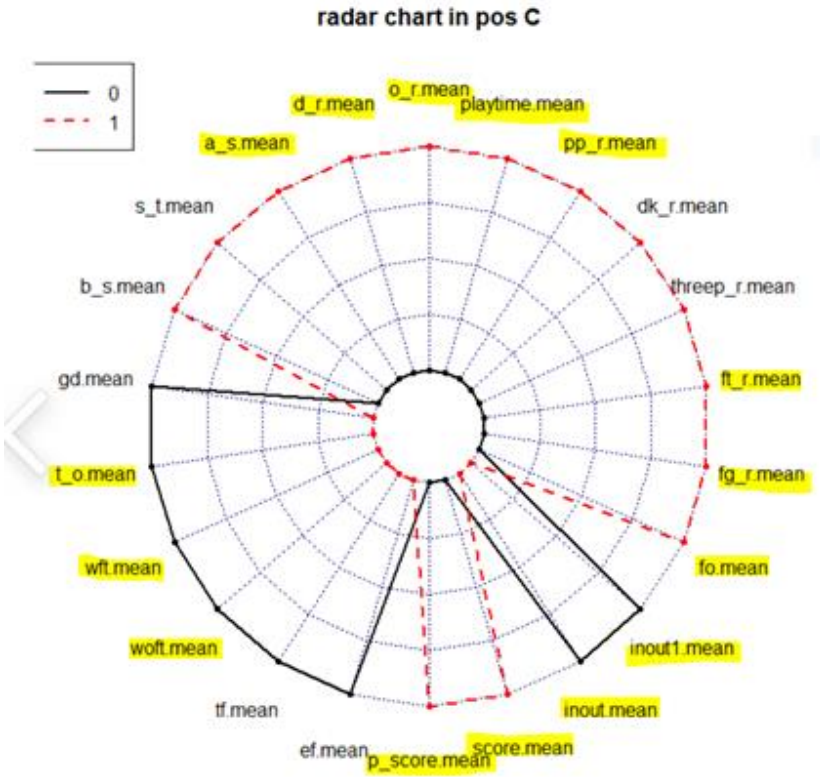


➔ 실선까지의 변수 선택

```
> sort(varImpPlot(windGDrf)[,1])
        tf       dk_r         ef         gd        b_s start_flag      ft_r    p_score        o_r       pp_r
  2.801639   7.051646  16.111310  27.140393  34.667351  52.625413 85.184219  85.884981  90.717257 102.316679
       s_t        wft  home_away       woft        t_o      inout         fo season_code       fg_r      inout1
105.640738 109.128989 117.100282 131.847042 132.027408 132.831578 135.386746 136.995175 137.082746 137.977829
  threep_r        d_r        a_s      score  team_code   playtime
139.942833 146.851146 176.470882 183.095485 403.260214 434.576274
```

➔ 각 변수에 대한 지니계수 확인

> 포지션 C의 Radar Chart 생성

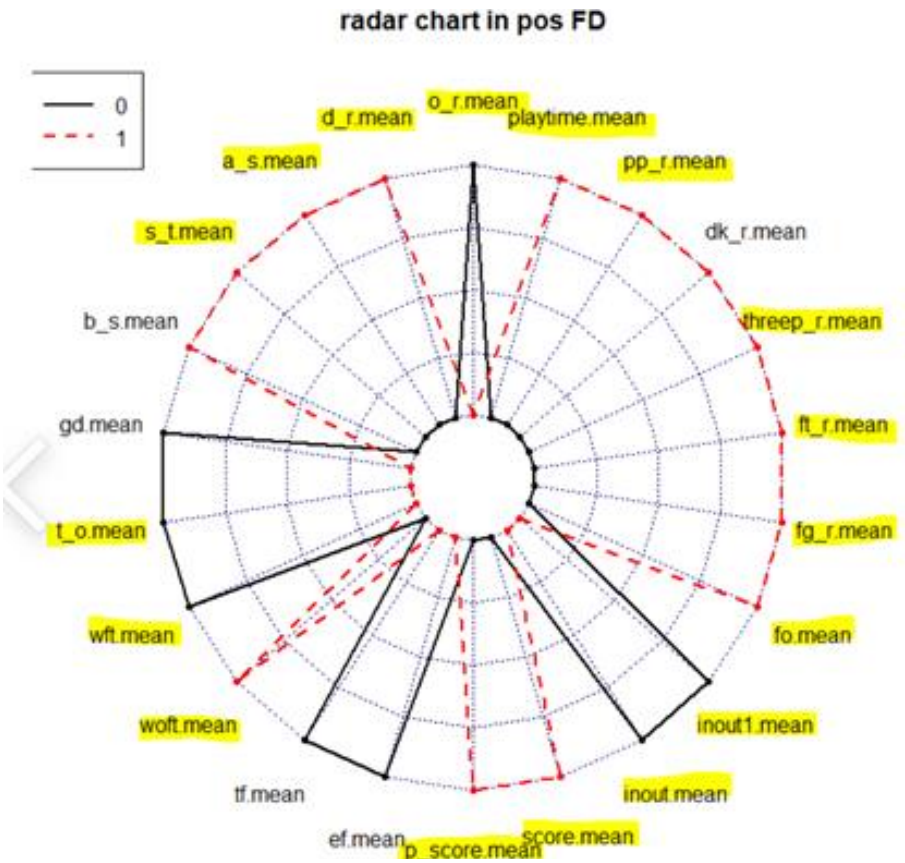


C formula

$$playtime \times 151.54381 + fg\_r \times 79.79187 + d\_r \times 75.33208 + score \times 72.40129 + pp\_r \times 67.19977 + fo \times 58.64871 + o\_r \times 58.56991 + a\_s \times 53.97268 + inout1 \times (-51.78882) + p\_score \times 51.41374 + inout \times (-49.64930) + wft \times (-48.74051) + ft\_r \times 46.79243 + t\_o \times (-46.08248) + woft \times (-42.56136)$$

➔ 포지션 C에서 선수들에 대한 능력치 평가 지표 수식

> 포지션 FD의 Radar Chart 생성

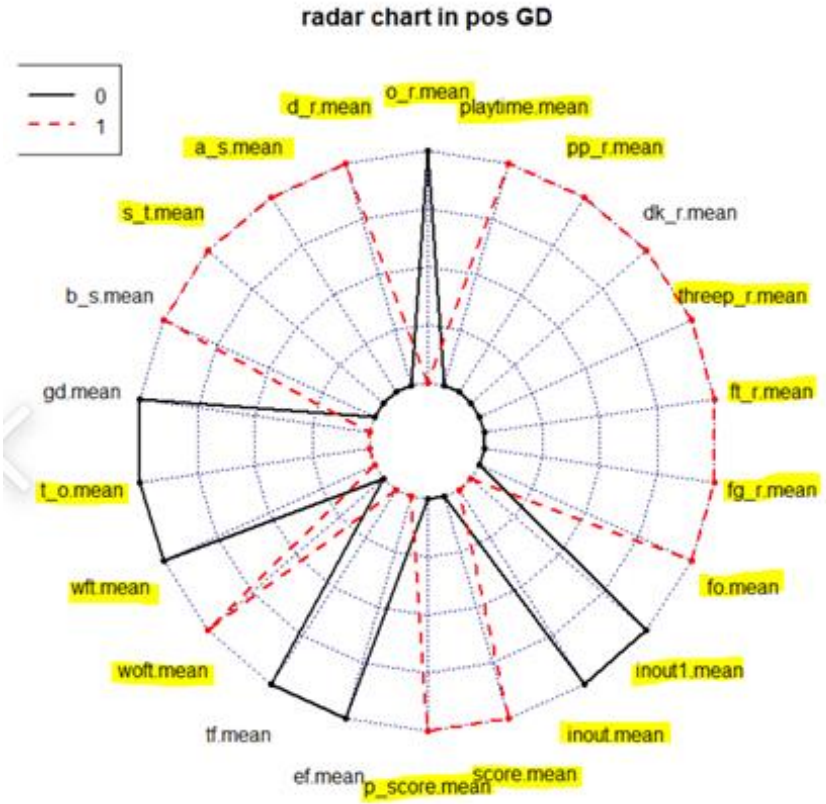


FD formula

$$playtime \times 399.54080 + score \times 195.86099 + d\_r \times 170.03102 + fg\_r \times 162.55550 + a\_s \times 140.82453 + fo \times 137.89567 + inout1 \times (-135.84668) + threep\_r \times 135.39948 + inout \times (-130.75382) + pp\_r \times 129.02176 + o\_r \times (-125.17066) + wft \times (-123.57271) + woft \times 121.31665 + t\_o \times (-119.88935) + p\_score \times 106.55922 + ft\_r \times 100.31267 + s\_t \times 99.26855$$

➔ 포지션 FD에서 선수들에 대한 능력치 평가 지표 수식

> 포지션 GD의 Radar Chart 생성



GD formula

$$playtime \times 434.57627 + score \times 183.09549 + a\_s \times 176.47088 + d\_r \times 146.85115 + threep\_r \times 139.94283$$
$$+ inout1 \times (-137.97783) + fg\_r \times 137.08275 + fo \times 135.38675 + inout \times (-132.83158) + t\_o \times (-132.02741) + woft \times 131.84704$$
$$+ wft \times (-109.12899) + s\_t \times 105.64074 + pp\_r \times 102.31668 + o\_r \times (-90.71726) + p\_score \times 85.88498 + ft\_r \times 85.18422$$

➔ 포지션 GD에서 선수들에 대한 능력치 평가 지표 수식

> 평균 값을 이용한 각 변수에 대한 선수들의 값 단일화

```
> head(playercar)
       pos       o_r       d_r       a_s       s_t       b_s        gd       t_o       wft
240074  FD 1.0000000 1.5714286 0.6285714 0.2000000 0.20000000 0.02857143 0.5714286 1.0857143
290218  FD 1.2264151 3.4339623 3.4339623 0.8679245 1.24528302 0.05660377 1.4905660 1.5660377
290227  FD 1.4500000 2.8500000 0.9000000 0.2500000 0.40000000 0.20000000 1.4000000 0.9000000
290232  FD 1.2500000 0.5625000 0.5000000 0.2500000 0.00000000 0.00000000 0.5000000 0.2500000
290351  GD 0.3571429 0.8571429 0.5000000 0.4285714 0.07142857 0.04761905 0.1428571 0.4285714
290407  GD 0.8409091 2.3181818 3.9090909 1.2045455 0.13636364 0.06818182 1.7727273 0.8636364
            woft   tf         ef   p_score     score    inout   inout1        fo      fg_r      ft_r
240074 0.8857143 0.00 0.00000000 1.6000000  4.028571 2.571429 2.285714 0.4285714 0.3771429 0.2142857
290218 0.6226415 0.00 0.03773585 3.0566038  8.264151 2.433962 2.830189 1.4339623 0.4661276 0.2914196
290227 0.6500000 0.05 0.05000000 4.9000000 12.000000 3.250000 3.450000 2.3000000 0.4510714 0.4732143
290232 0.2500000 0.00 0.00000000 0.8750000  2.375000 1.687500 1.625000 0.2500000 0.3125000 0.0937500
290351 0.6428571 0.00 0.02380952 0.4761905  2.785714 1.904762 1.880952 0.7142857 0.1817460 0.1726190
290407 0.5454545 0.00 0.00000000 6.1363636 14.068182 3.500000 3.840909 3.3181818 0.5081626 0.7006563
        threep_r dk_r      pp_r  playtime
240074 0.1683333  0.0 0.3833333  903.3714
290218 0.3452830  0.0 0.4918239 1590.3208
290227 0.2989286  0.1 0.4984524 1360.6000
290232 0.1520833  0.0 0.3125000  597.6250
290351 0.1426587  0.0 0.1341270  646.2381
290407 0.3088745  0.0 0.5316173 1680.0682
```

➔ 해당 수치를 포지션에 맞는 수식에 대입하여 선수들의 순위 결정

> 순위권에서 제거시켜야할 외국인 용병들 제거

```
> length(notkorea_idx)
[1] 50
> ## 외국인 용병들 확인
> table( player_code[notkorea_idx,3] )
```

글렌 코지	기디 팟츠	네이트 밀러	다니엘 러츠	대릴 먼로	데이빗 로건
1	1	1	1	1	1
듀안 섬머스	디제이 존슨	랜디 컬페퍼	레이션 테리	르브라이언 내쉬	리온 윌리엄스
1	1	1	1	1	1
마커스 랜드리	마커스 쏜튼	마커스 킨	마커스 포스터	마퀴스 티그	머피 할로웨이
1	1	1	1	1	1
미카일 매킨토시	버논 해밀턴	벤 음발라	브랜든 브라운	섀넌 쇼터	쉐인 깁슨
1	1	1	2	1	1
스테이시 오그먼	스테판 무디	아이라 클라크	아이반 아스카	애런 헤인즈	오데리언 바셋
1	1	1	1	1	1
윌리엄 다니엘스	유진 펠프스	저스틴 덴트몬	저스틴 에드워즈	저스틴 틸먼	제이슨 시거스
1	1	1	2	1	1
제임스 메이스	제쿠안 루이스	조쉬 그레이	조쉬 에코이언	조엘 헤르난데즈	찰스 로드
1	1	1	1	1	1
칼 홀	큐제이 피터슨	크리스토퍼 로프튼	투 할로웨이	프랭크 로빈슨	필립 허버드
1	1	1	1	1	1

➔ 외국인 용병들 확인 후 제거

> 포지션별 능력치 상위 10위권 선수들 확인

player_no	kname
290750	김종규
290450	오세근
290781	김준일
290987	이종현
290549	김민욱
290248	하승진
290110	송창무
290439	유성호
290491	김승원
290786	주지훈

➔ C

player_no	kname
290787	이승현
290897	송교창
291091	양홍석
290417	최진수
291084	안영준
290993	최준용
290280	문태영
290119	김영환
290661	문성곤
290284	허일영

➔ FD

player_no	kname
290381	이정현
290440	이관희
290776	허웅
291085	허훈
290742	이대성
290407	김선형
290505	김시래
210074	양동근
290991	천기범
291008	박지훈

➔ GD

All Star 전에서 갑자기 실력발휘
못하면 어떡하지?????

Q2. All Star 전에서 자신의 능력치를 그대로 발휘할 것인가?
(즉, 경기 시에 환경의 악영향을 받을 가능성이 높은 선수는 누구인가?)

A2. 모든 시즌의 경기에서 선수들의 경기 데이터 값들의 변동 확인
→
변수의 수가 많으며, 변수마다 단위를 통일시켜 비교가능하도록 하기위해,
PCA를 이용하여 8개로 축소 후 편차 계산
→
분산의 합이 큰 선수는 환경의 악영향을 상대적으로 많이 받는 선수 이며,
경기 기록의 변동성이 큰 선수로 위험성이 높은 선수로 판단
→
경기마다 자신의 능력치를 발현시킬 확률이 적은 상위 25%의 선수들은
순위권에서 제거

01

02

03

> PCA 실시 후 주성분 점수 확인

```
> round(gofall, 2)
 [1] 37.88  9.98  6.63  5.87  5.08  4.13  3.81  3.61  3.27  3.20  2.68  2.60  2.19  2.09  1.53  1.49  0.90
[18]  0.72  0.61  0.57  0.50  0.33  0.18  0.08  0.05  0.00  0.00
> sum(round(gofall, 2)[1:8])
[1] 76.99
```

➔ 원 데이터 d에 존재하는 모든 변수들이 갖고 있는 정보의 76.99%를 갖는 원 변수들의 선형 결합으로 이루어진 새로운 변수 8가지 선정

```
> round(Pall, 3)
      P1     P2     P3     P4     P5     P6     P7     P8
1  2.759 -0.067 -1.327  0.703  0.321 -0.244 -0.720  0.811
2 -0.580 -1.270 -1.952  0.918 -1.358 -1.415  0.931 -0.240
3  0.225 -0.261 -0.872  0.539 -0.517  0.656  0.166  0.050
4 -1.185  2.073 -0.067  0.395 -0.331  0.250  0.676 -0.022
5  0.871  1.546 -0.517  0.297 -0.038  0.582 -0.742  0.887
7 -7.494 -2.090 -0.094  1.074  0.342 -0.567  0.679  0.534
8  1.442  0.772  1.078 -0.328 -0.959  0.047  0.916 -0.665
9  3.314 -1.218 -0.185  0.120  0.303 -0.456  0.263  0.066
10 2.576  0.088 -0.723  0.281 -0.072 -0.130  0.033  0.119
11 -5.196 -1.959  0.130  0.865 -2.947 -1.547 -0.794  0.229
12 -0.534  1.201 -2.289  1.289  1.168  0.527 -1.232  1.135
14 2.273  0.089  1.025 -0.099  0.201 -1.041  0.090  0.262
15 1.867  0.945  0.381  0.309 -0.491  0.417 -1.058  0.931
17 2.502 -0.384  0.282  0.210 -0.026  0.351 -1.095  0.756
19 1.042  0.095  0.051 -0.093  0.356  0.269 -2.251 -2.753
20 1.879 -1.284  0.624 -0.031  0.976 -0.588  0.193  0.338
```

> PCA 주성분을 이용한 편차 계산

```
> summary(sort_devsum)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
     38     712    1161    1104    1498    2094
> which(sort_devsum==1500)
290979
    69
> sort_devsum[1:69]
291107 290395 291180 291171 291011 291108 290978 291058 290367 290981 290774 291132 291139 291133 291055
  2094   2066   2064   2045   2017   2007   1981   1979   1954   1953   1929   1901   1898   1894   1886
290861 291137 291134 291014 291129 290371 290982 291140 290261 290325 291138 291052 291057 290993 290524
  1877   1877   1875   1872   1838   1828   1818   1814   1800   1788   1783   1780   1759   1748   1746
291145 291233 291109 290738 291172 290984 290742 290983 290412 290794 291144 290381 290797 230080 290280
  1745   1740   1736   1734   1710   1699   1693   1692   1688   1678   1656   1652   1640   1635   1634
290542 290750 291220 290505 290976 290106 291056 215053 291136 291234 290790 290485 290407 290862 290765
  1632   1628   1622   1621   1614   1608   1604   1601   1587   1584   1583   1581   1566   1565   1557
290740 290558 290909 235070 290450 290553 220194 290417 290979
  1523   1523   1523   1512   1508   1506   1504   1502   1500
```

➔ 가장 큰 편차를 갖는 즉, 가장 불안정한 점수분포를 띄는 선수 번호는 291107로 2094 값의 편차를 갖는 것을 볼 수 있음.

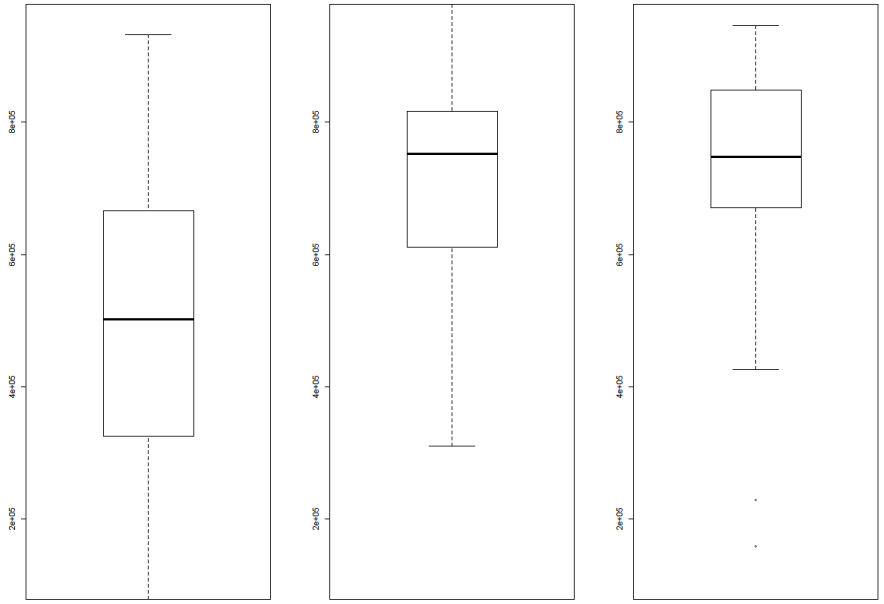
> 포지션별 상위 10위권 선수들 중 제거 고려 선수 확인

```
> # C  : 290750(김종규) / 290450(오세근)
> # FD : 290993(최준용) / 290280(문태용) /
> #      290417(최진수)
> # GD : 290742(이대성) / 290381(이정현) /
> #      290505(김시래) / 290407(김선형)
```

Q3. All Star 전에서 더 높은 능력치를 발휘할 수 있는가?
(즉, 각 선수들의 성장율이 어떻게 될 것인가?)

A3. A2)에 대한 처리과정 중 편차가 큰 이유가 성장율이 큰 선수인 경우 고려.
➔
이러한 선수들은 시간이 지날수록 현재 갖는 능력치보다 상승 성향을 보일 가능성이 큰 선수로 인지.
즉, All Star 전에서도 또한 더 높은 능력치 발휘할 것으로 예상가능
➔
편차가 크나 성장하는 정도가 큰 선수들은 순위권에서 제거 X, 포함 O

> 제거 고려 대상 중 유일하게 성장율이 높아진 선수



```
> mean(resu29);mean(resu31);mean(resu33)
[1] 496456.2
[1] 707734.9
[1] 733952
```

290417 최진수

➔ 성장하고 있는 선수로서, 제거 대상에서 제외

> 최종 선발된 All Star 선수 12명 명단

```
> finalC;finalFD;finalGD
    player_no  kname
139    290781 김준일
39     290987 이종현
    player_no  kname
107    290787 이승현
223    290897 송교창
22     291091 양홍석
114    290417 최진수
202    291084 안영준
    player_no  kname
132    290440 이관희
81     290776   허웅
21     291085   허훈
60     210074 양동근
145    290991 천기범
```

> 최종 선발된 All Star 선수: 포지션 C

서울 삼성 썬더스

C

NO 31. 김준일

1992.05.07

201cm

휘문고 - 연세대

PPG	RPG	APG
11.2	5.0	2.3

울산 현대모비스 피버스

C

NO 32. 이종현

1994.02.05

203cm

경복고등학교 - 고려대학교

PPG	RPG	APG
-	-	-

> 최종 선발된 All Star 선수: 포지션 FD

고양 오리온 오리온스

FD

NO 33. 이승현

1992.04.16
197cm
용산고 - 고려대

PPG 8.0 RPG 5.7

전주 KCC 이지스

FD

NO 2. 송교창

1996.07.03
198cm
삼일상고

PPG 15.4 RPG 4.8 APG 3.5

부산 KT 소닉붐

FD

NO 11. 양홍석

1997.07.02
195cm
부산중앙고등학교 - 중앙대학교

PPG 11.3 RPG 5.7 APG 1.7

고양 오리온 오리온스

FD

NO 23. 최진수

1989.05.11
203cm
사우스랜트고등학교 - 메릴랜드대학교

PPG 8.7 RPG 4.1 APG 1.2

서울 SK 나이츠

FD

NO 8. 안영준

1995.06.28
194.6cm
경복고등학교 - 연세대학교

PPG 8.6 RPG 5.0 APG 2.5

> 최종 선발된 All Star 선수: 포지션 GD

서울 삼성 썬더스

GD

NO 7. 이관희

1988.04.29
190cm
낙생고 - 연세대

PPG 12.4 | RPG 3.6

원주 DB 프로미

GD

NO 6. 허웅

1993.08.05
185cm
용산고 - 연세대

PPG 8.7 | RPG 1.5 | APG 1.0

부산 KT 소닉붐

GD

NO 2. 허훈

1995.08.16
180cm
용산고등학교 - 연세대학교

PPG 15.8 | RPG 3.3 | APG 6.9

울산 현대모비스 피버스

GD

NO 6. 양동근

1981.09.14
180cm
대방초 - 삼선중 - 용산고 - 한양대

PPG 8.8 | RPG 2.4 | APG 3.2

서울 삼성 썬더스

GD

NO 11. 천기범

1994.05.28
186cm
부산중앙고등학교 - 연세대학교

PPG 4.3 | RPG 2.5 | APG 3.4

✓주 제

“ 선발된 선수들의 Stat. 이 포함된 MD를 설계해 주세요.”

✓ MD image

서울삼성
김준일 선수

LEGEND GAME

20190201 서울삼성 vs 원주DB

슈퍼리바운드 : 603 / 페인트존 득점률: 71%
2점슛득점률: 67% / 자유투 성공률: 50%

KBL
KOREAN BASKETBALL LEAGUE

1 JAN

일	월	화	수	목	금	토
			1 신정	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25 설날
26	27 대체 휴일	28	29	30	31	

✓ 기대 효과

➔ 선수 개인마다의 MD를 만들기보다는 하나의 MD에 12명 선수들의 특성을 잘 담아낼 수 있는 상품인 달력을 선택

➔ All Star 전에 참여할 12명 선수들의 특색을 담아낼 MD 로써, 각 Monthly 마다 특정 선수의 LEGEND 경기 기록을 담아냄.

➔ 다른 아이디어로 나왔던 어플, 다이어리, 머그컵, 키링 보다 경제적으로 효율적이며 Targeting 스펙트럼이 더욱 넓어 훨씬 대중적인 MD임.

> 최종 선발된 All Star 선수 12명 명단

✓ MD

```
> finalC;finalFD;finalGD
    player_no  kname
139    290781 김준일
39     290987 이종현
    player_no  kname
107    290787 이승현
223    290897 송교창
22     291091 양홍석
114    290417 최진수
202    291084 안영준
    player_no  kname
132    290440 이관희
81     290776   허웅
21     291085   허훈
60     210074 양동근
145    290991 천기범
```

THANK YOU ☺