Stat441 Final project

Group 2

2022.12.01

## *1.Introduction：*

NBA has been a hot topic in recent years. More and more people around the world follow and fall in love with NBA. NBA_Player_Boxscore_2021-22 data collects information about the performance of different NBA players from 2021 to 2022. Players on different teams have different data such as game dates, points scored, etc. The data contained 35 variables to assess their performance ability.

## *2. Problem of interest：*

In the 2021-22 season, every player in the NBA had a great preference, however, the audience was focused on who was the best player (MVP) in this year. After statistical analysis, we have selected the three NBA star players that the audience is most interested in, and we intend to select the best player in the 2021-22 season from these three players, who are the Trae Young from the Hawks, Luka Doncic from Mavericks, and Giannis Antetokounmpo from Bucks.

Trae Young


Luka Doncic

Giannis Antetokounmpo

To evaluate whether a player is skilled or not, we choose the number of rebounds to evaluate, because the rebound is a very complicated basketball skill, which can fully reflect the strength of the player.

***3. Method(s) and main results：***

In the beginning, we wanted to select the best performers, and we started with the discriminant analysis. We chose three discriminant analysis methods which are Linear discriminant analysis (LDA), Mixture Discriminant analysis (MDA), and Flexible Discriminant analysis (FDA). Let's consider using points and rebounds to build our LDA classifier first. We will display the decision boundary on the scatterplot of points and rebounds for the training set and testing set separately.

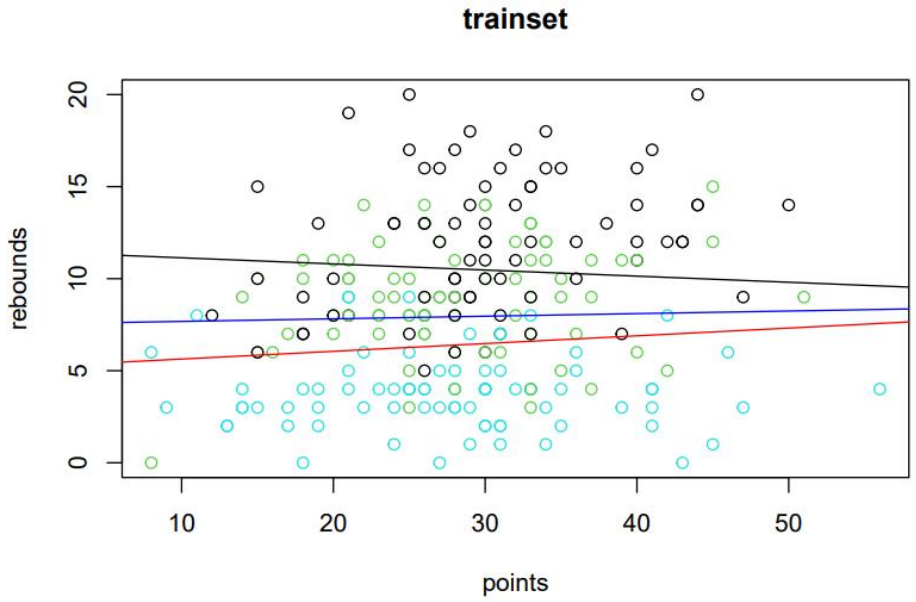
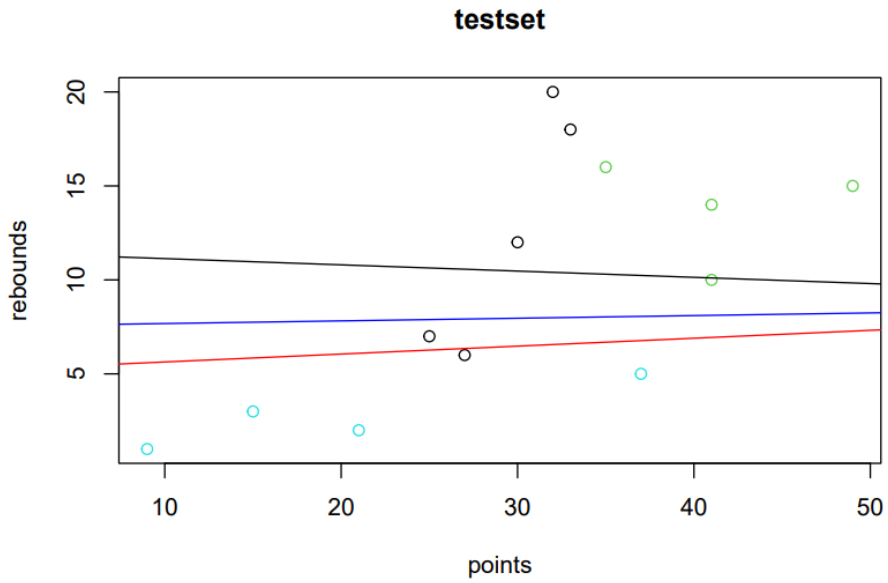


Figure1. The LDA classifier for the points and rebounds of three NBA players for trainset.

Figure2. The LDA classifier for the points and rebounds of three NBA players for testset.




By comparison of these two figures, we can obtain that using Linear discriminant analysis, we can find that given the same points, generally, Giannis Antetokounmpo has the most rebounds per game, while Trae Young has the least rebounds per game. Although the three players are all top scorers in the league, after LDA classification, we can conclude that Giannis and Luka Doncic have better performance in terms of rebounds.

Next, we used MDA and FDA for classification comparison. MDA is to make each class hypothetical. to be a Gaussian mixture of subclasses, The FDA is letting the non-linear combinations of predictors is used such as splines. Through calculation, we get the Misclassification Error of these two methods. The Misclassification Error of MDA was 0.85652 and that of FDA was 0.87391. The Misclassification Error of MDA is relatively small, so MDA is a better way to classify these three players.

As can be seen in figure1 and figure2, the black dots and green dots in the image overlapped, and they were mixed together, so we could not quickly determine which of Giannis and Luka Doncic had better performance. Therefore, in order to select the best performers more accurately, we choose to use Tree Classifier.

In our tree classifier, five variables were selected for tree classification, which are pts, reb, ast, stl, blk which are points, rebounds, assists, steals, Blocked Shots respectively.

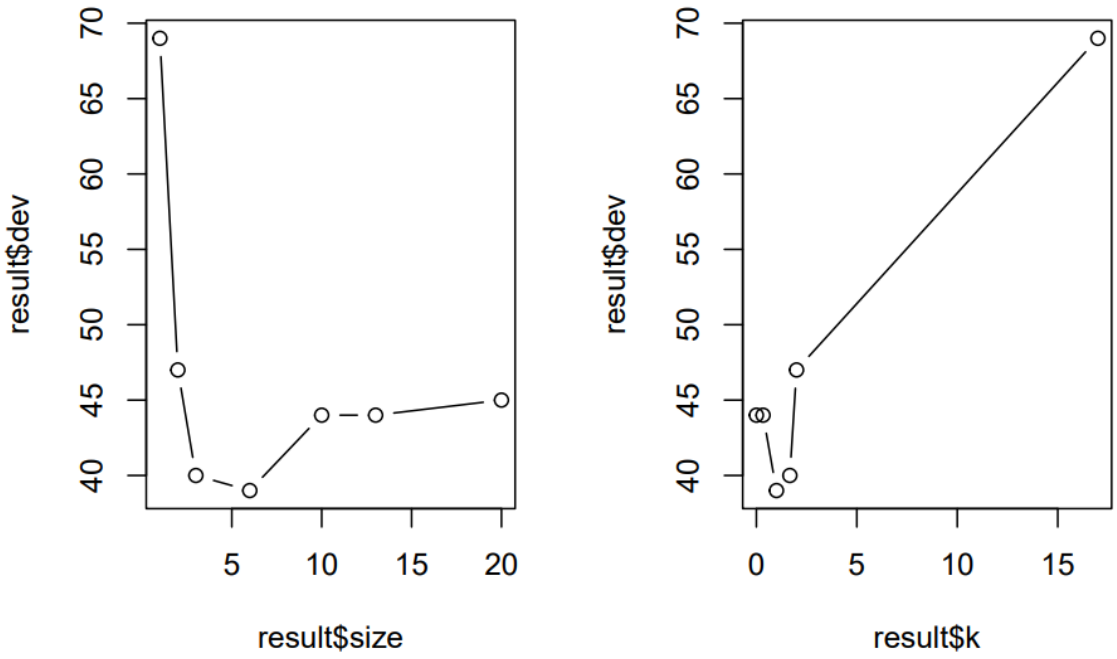



Figure 3. The error rate for both size and k

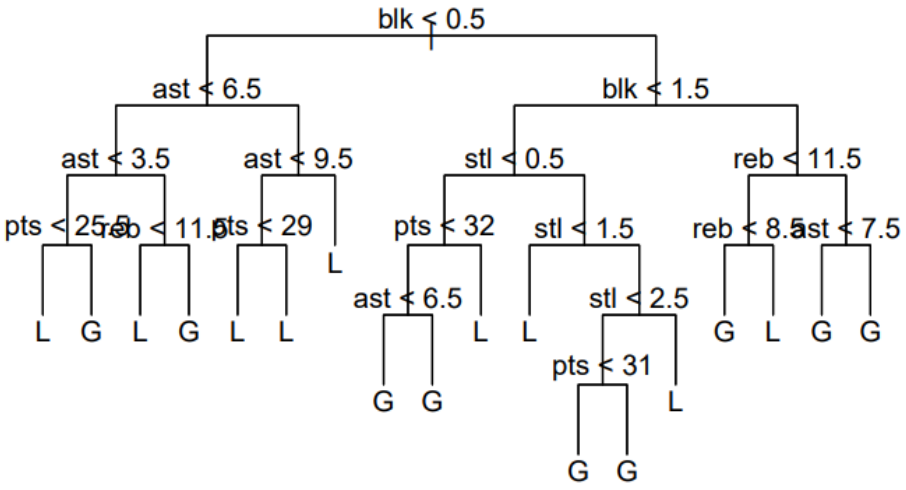



Figure4. The tree plot which does not prune.

From figure4, we obtain that the tree classification diagram derived from the five variables to judge which Luka Doncic or Giannis Antetokounmpo is strong in basketball performance. However, intuitively, figure4 is too complicated and messy. Not conducive to intuitive judgment and analysis. Therefore, we extracted the three best variables blk, ast and reb and re-classified them to get a pruned tree, Figure5.

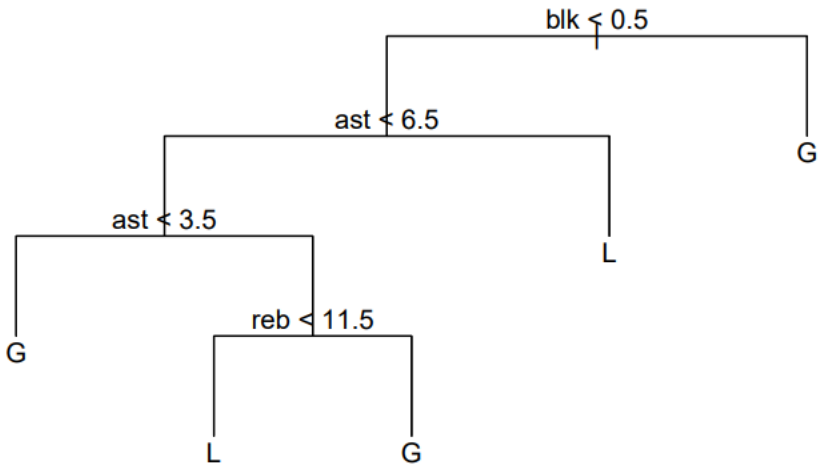



Figure5. The tree plot which is prune.

Figure5 shows the classification of players' performance ability more intuitively. We can also know from the figure5 that players on the right side of the branch have stronger performance ability than those on the left side. We can judge that Giannis Antetokounmpo has stronger basketball ability.

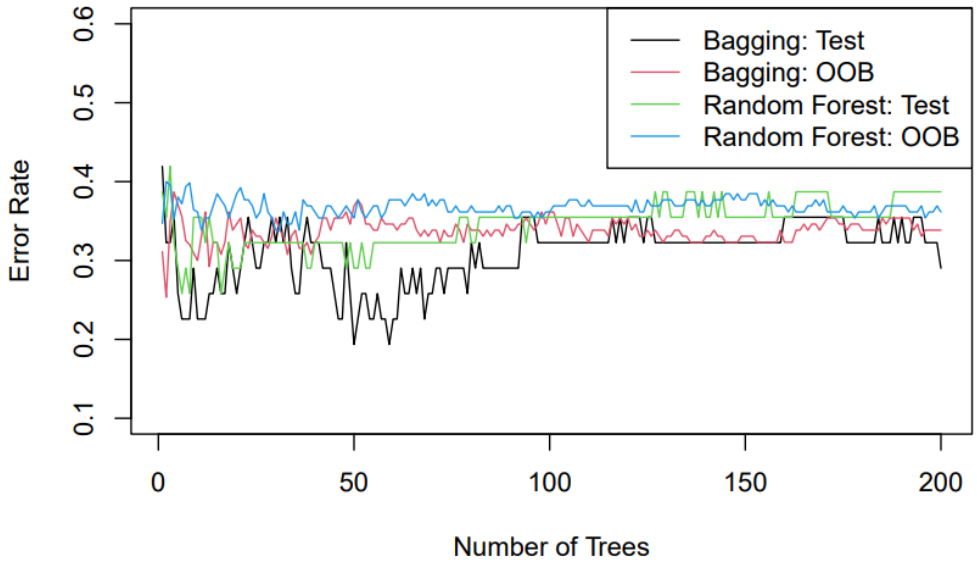



Figure6. Error rate of Test Bagging, OOB (out-of-bag) Bagging, Test Random Forest and OOB (out-of-bag) Random Forest.

By Figure6, we obtain the error rate of Test Bagging, OOB (out-of-bag) Bagging, Test Random Forest and OOB (out-of-bag) Random Forest these four trees, and we get the error rates are 2, 50, 11, 6 respectively. The lowest error rate is Test Bagging tree, so we prefer using test bagging tree to do the classify for choose the MVP between player Luka Doncic and Giannis Antetokounmpo.

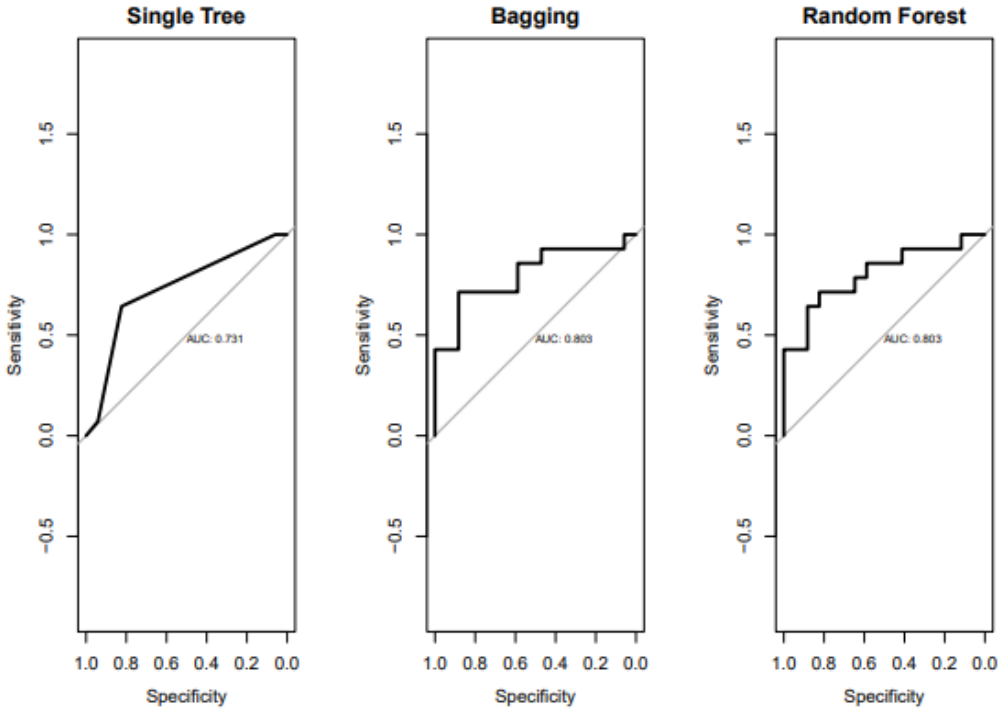



Figure7. Comparison based on ROC curve with Single tree, Bagging and Random Forest methods.

From figure7, we can obtain the AUC value for Single tree, Bagging and Random Forest methods. Since the higher the AUC value of the classifier, the higher the accuracy, so we obtain that Random Forest is the best classifier which has the highest AUC value. However, in our case, these three classifier's AUC value are very closed, and 0.5 <AUC< 1, so according to the results of ROC curve, all three classification methods can be used and if these three classifiers set thresholds properly, they can have predictive value.

## *4. Conclusion*

In conclusion, we used LDA, MDA, FDA, and tree classification to screen three NBA star players, Trae Young, Luka Doncic and Giannis Antetokounmpo, who is the most expressive

basketball player in 2021-2022.Though our categorized, compared error rate and screened, finally, Giannis Antetokounmpo's basketball performance is relatively strong.

***Contribution of each group member:***

Robin: LDA MDA FDA

William: First part of tree classification

Meko: Second part of tree classification

Jesy: Text and Content Writing, Tandem

## *Appendix: R codes and output:*

# project_rl

Robin

2022-11-30

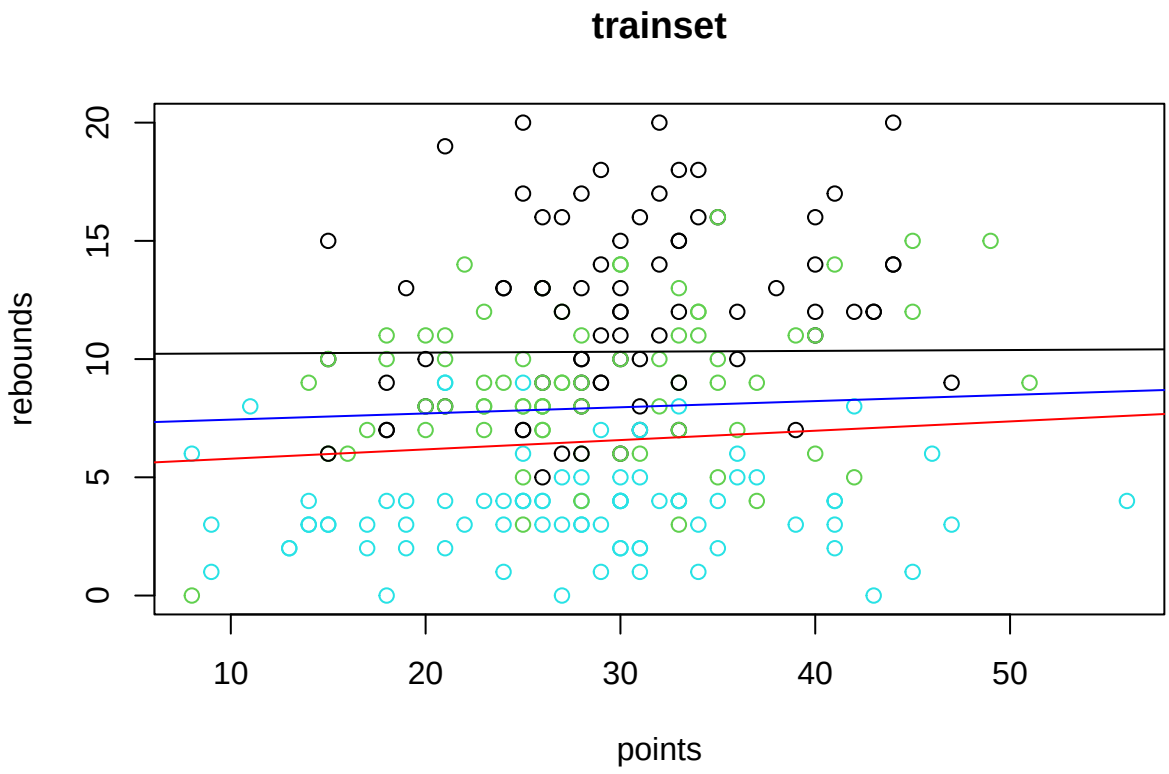
```
library(mda)
```

```
## Loading required package: class
```

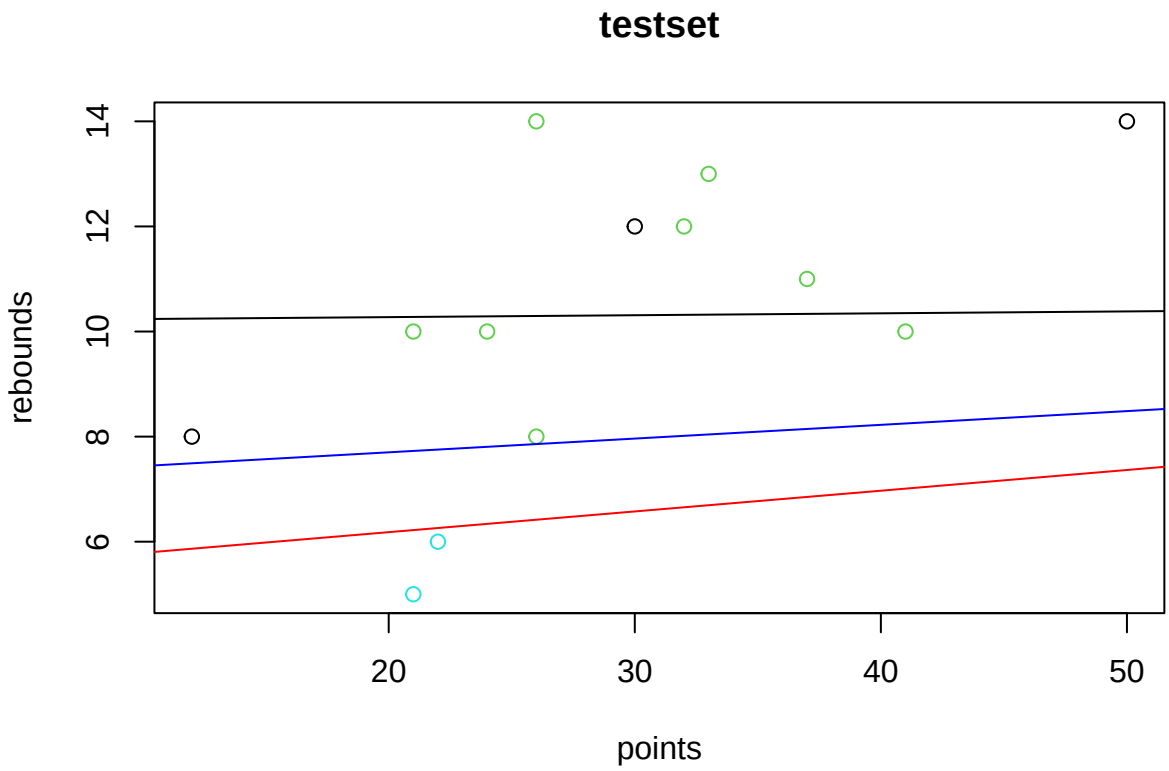
```
## Loaded mda 0.5-3
```

```
data<- read.csv("NBA_Player_Boxscore_2021-22.csv", stringsAsFactors=TRUE)
data = subset(data, athlete_display_name %in% c("Trae Young","Luka Doncic", "Giannis Antetokounmpo"))
subdata = data.frame(data$pts,data$reb)
train = sample(dim(subdata)[1], 230)
data.train = subdata[train,]
data.test = subdata[-train,]
names.train = data[train,1]
names.test = data[-train,1]
## LDA
u1 = as.vector(apply(data.train[names.train=="Trae Young",],MARGIN=2,FUN=mean))
u2 = as.vector(apply(data.train[names.train=="Luka Doncic",],MARGIN=2,FUN=mean))
u3 = as.vector(apply(data.train[names.train=="Giannis Antetokounmpo",],MARGIN=2,FUN=mean))
pi1 = mean(names.train=="Trae Young")
pi2 = mean(names.train=="Luka Doncic")
pi3 = mean(names.train=="Giannis Antetokounmpo")
n1 = sum(names.train=="Trae Young")
n2 = sum(names.train=="Luka Doncic")
n3 = sum(names.train=="Giannis Antetokounmpo")
pooled=1/(n1+n2+n3-3)*((n1-1)*
              cov(data.train[names.train=="Trae Young",])
              +(n2-1)*cov(data.train[names.train==
                                "Luka Doncic",])
              + (n3-1)*cov(data.train[names.train==                    "Giannis Antetokounmpo",]))
a1=log(pi1/pi2)-1/2*t(u1+u2)%*%solve(pooled)%*%(u1-u2)
a2=log(pi1/pi3)-1/2*t(u1+u3)%*%solve(pooled)%*%(u1-u3)
a3=log(pi2/pi3)-1/2*t(u2+u3)%*%solve(pooled)%*%(u2-u3)
b1=solve(pooled)%*%(u1-u2)
b2=solve(pooled)%*%(u1-u3)
b3=solve(pooled)%*%(u2-u3)
int1=-a1/b1[2]
int2=-a2/b2[2]
int3=-a3/b3[2]
slope1=-b1[1]/b1[2]
slope2=-b2[1]/b2[2]
slope3=-b3[1]/b3[2]
```

```
plot(data.train,col=names.train,xlab ="points",
     ylab = "rebounds", main = "trainset")
abline(a=int1,b=slope1,col = "red")
abline(a=int2,b=slope2,col = "blue")
abline(a=int3,b=slope3,col = "black")
```




```
plot(data.test,col=names.test,xlab ="points",
     ylab = "rebounds", main = "testset")
abline(a=int1,b=slope1,col = "red")
abline(a=int2,b=slope2,col = "blue")
abline(a=int3,b=slope3,col = "black")
```




Using Linear discriminant analysis, we can find that given the same points, generally, Giannis Antetokounmpo has the most rebounds per game, while Trae Young has the least rebounds per game. Although the three players are all top scorers in the league, after LDA classification, we can conclude that Giannis and Luka Doncic have better perfermance in terms of rebounds.

```
## MDA
model.mda <- mda(data.reb~., data = data.train )
model.mda
```

```
## Call:
## mda(formula = data.reb ~ ., data = data.train)
##
## Dimension: 1
##
## Percent Between-Group Variance Explained:
##  v1
## 100
##
## Degrees of Freedom (per dimension): 2
##
## Training Misclassification Error: 0.86522 ( N = 230 )
##
## Deviance: 1209.577
```

```
## FDA
model.fda <- fda(data.reb~., data = data.train )
model.fda
```

```
## Call:
## fda(formula = data.reb ~ ., data = data.train)
##
## Dimension: 1
##
## Percent Between-Group Variance Explained:
##  v1
## 100
##
## Degrees of Freedom (per dimension): 2
##
## Training Misclassification Error: 0.86522 ( N = 230 )
```

Comparing the misclassification error, $0.83478 < 0.87391$, so MDA is a better approach for classifying the three players.

# Untitled

Weikang Jiang

2022-11-25

```
library(tree)
```

```
## Warning: package 'tree' was built under R version 4.2.2
```

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.2 --
## v ggplot2 3.3.6      v purrr   0.3.5
## v tibble  3.1.8      v dplyr   1.0.10
## v tidyr   1.2.1      v stringr 1.4.0
## v readr   2.1.3      v forcats 0.5.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
data<-read.csv("NBA_Player_Boxscore_2021-22.csv")
data = subset(data, athlete_display_name %in% c("Luka Doncic", "Giannis Antetokounmpo"))
data[which(data$athlete_display_name == "Luka Doncic"), ]$athlete_display_name = "L"
data[which(data$athlete_display_name == "Giannis Antetokounmpo"), ]$athlete_display_name = "G"

train = sample(dim(data)[1], 130)
data.train = data[train,]
data.test = data[-train,]

treemodel = tree(as.factor(athlete_display_name)~pts+reb+ast+stl+blk,
                 data.train, split ="gini")
tree.pred=predict(treemodel,data.train,type="class")
table(tree.pred,data.train$athlete_display_name)
```

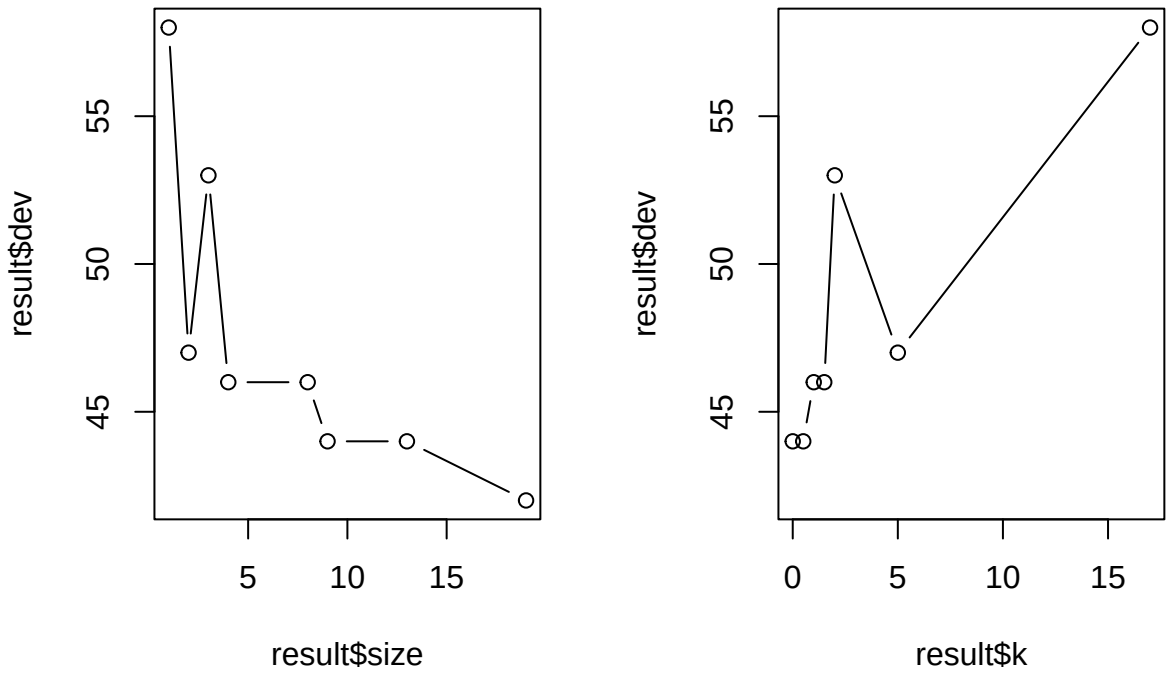
```
##
## tree.pred  G  L
##         G 47 16
##         L 18 49
```

```
result = cv.tree(treemodel, FUN = prune.misclass)
result
```

```
## $size
## [1] 19 13  9  8  4  3  2  1
```

```
##
## $dev
## [1] 42 44 44 46 46 53 47 58
##
## $k
## [1] -Inf  0.0  0.5  1.0  1.5  2.0  5.0 17.0
##
## $method
## [1] "misclass"
##
## attr(,"class")
## [1] "prune"         "tree.sequence"
```

```
par(mfrow = c(1,2))
plot(result$size,result$dev,type = "b")
plot(result$k, result$dev, type ="b")
```

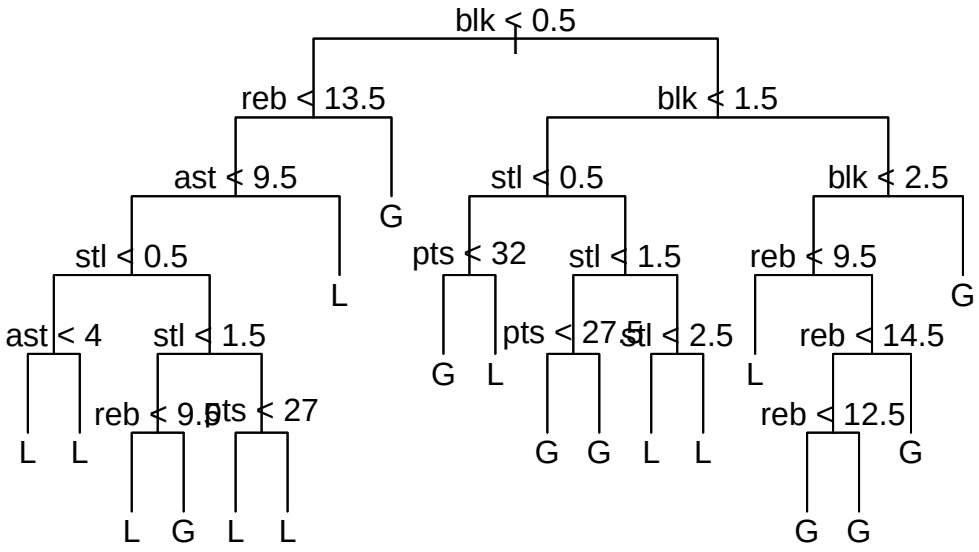



```
opt.size = result$size[which.min(result$dev)]
newtree = prune.misclass(treemodel,best=opt.size)

summary(newtree)
```

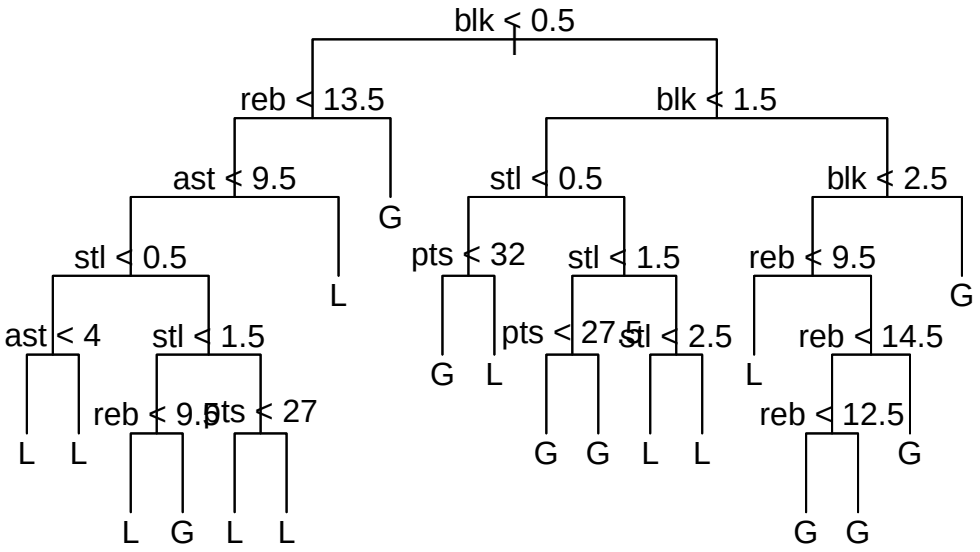
```
##
## Classification tree:
## tree(formula = as.factor(athlete_display_name) ~ pts + reb +
```

```
##     ast + stl + blk, data = data.train, split = "gini")
## Number of terminal nodes:  19
## Residual mean deviance:  1.066 = 118.3 / 111
## Misclassification error rate: 0.2462 = 32 / 130
```

```
par(mfrow = c(1,1))


#not prune tree plot
plot(treemodel, type ="uniform")
text(treemodel, pretty =0)
```




```
##new tree plot
plot(newtree, type ="uniform")
text(newtree, pretty =0)
```




```
num.tree=200
library(randomForest)
```

```
## Warning: package 'randomForest' was built under R version 4.2.2
```

```
## randomForest 4.7-1.1
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:dplyr':
##
##     combine
```

```
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
bag =randomForest(as.factor(athlete_display_name)~pts+reb+ast+stl+blk,
                  data=data.train, mtry=5,
                  ntree=num.tree, importance=TRUE,
```

```
                  xtest=select(data.test, pts, reb, ast, stl, blk),
                  ytest=as.factor(data.test$athlete_display_name),
                  keep.forest=TRUE)
bag
```

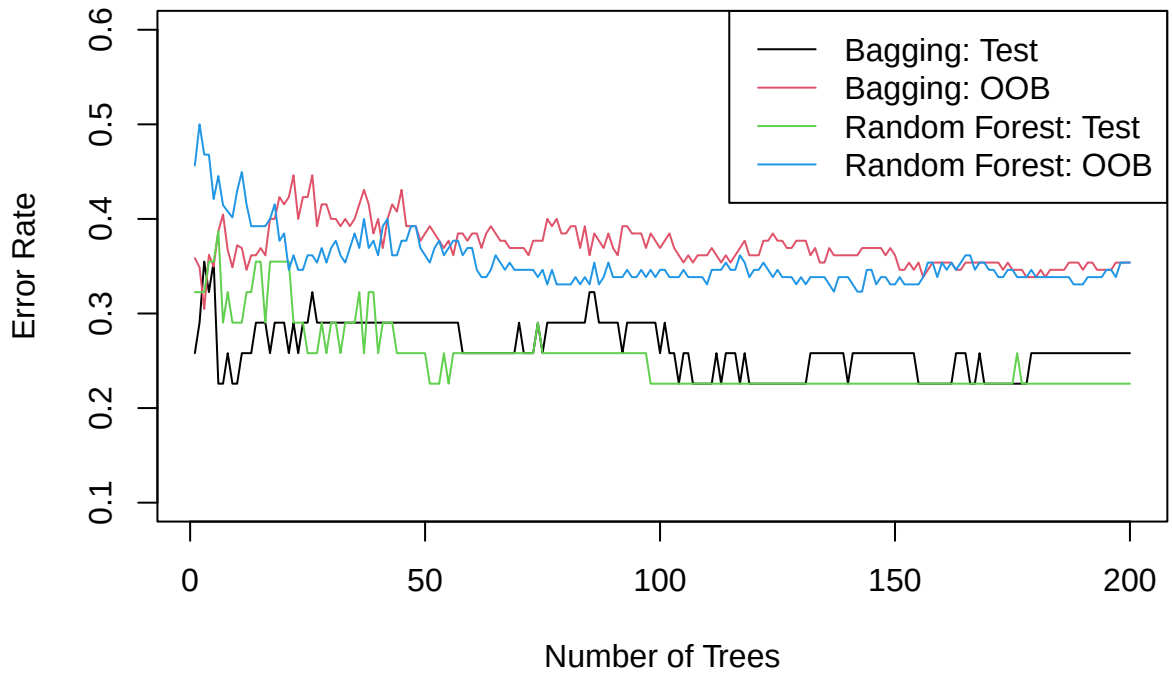
```
##
## Call:
##  randomForest(formula = as.factor(athlete_display_name) ~ pts +      reb + ast + stl + blk, data = da
##                Type of random forest: classification
##                      Number of trees: 200
## No. of variables tried at each split: 5
##
##         OOB estimate of  error rate: 35.38%
## Confusion matrix:
##    G  L class.error
## G 42 23   0.3538462
## L 23 42   0.3538462
##                 Test set error rate: 25.81%
## Confusion matrix:
##   G  L class.error
## G 9  6       0.400
## L 2 14       0.125
```

```
rf =randomForest(as.factor(athlete_display_name)~pts+reb+ast+stl+blk,
                  data=data.train, mtry=5,
                  ntree=num.tree, importance=TRUE,
                  xtest=select(data.test, pts, reb, ast, stl, blk),
                 ytest=as.factor(data.test$athlete_display_name),
                 keep.forest=TRUE)
rf
```

```
##
## Call:
##  randomForest(formula = as.factor(athlete_display_name) ~ pts +      reb + ast + stl + blk, data = da
##                Type of random forest: classification
##                      Number of trees: 200
## No. of variables tried at each split: 5
##
##         OOB estimate of  error rate: 35.38%
## Confusion matrix:
##    G  L class.error
## G 41 24   0.3692308
## L 22 43   0.3384615
##                 Test set error rate: 22.58%
## Confusion matrix:
##   G  L class.error
## G 9  6      0.4000
## L 1 15      0.0625
```

```
plot(1:num.tree, bag$test$err.rate[,1], ylim=c(0.1, 0.6), type="l",
     xlab="Number of Trees", ylab="Error Rate")
lines(1:num.tree, bag$err.rate[,1], type="l", col=2)
```

```
lines(1:num.tree, rf$test$err.rate[,1], ylim=c(0.1, 0.6), type="l", col =3)
lines(1:num.tree, rf$err.rate[,1], type="l", col=4)
legend("topright", lty=c(1,1,1,1), col=c(1:4),
       legend=c("Bagging: Test", "Bagging: OOB", "Random Forest: Test", "Random Forest: OOB"))
```




```
which.min(bag$err.rate[ ,1])
```

```
## [1] 3
```

```
which.min(bag$test$err.rate[ ,1])
```

```
## [1] 6
```

```
which.min(rf$err.rate[ ,1])
```

```
## [1] 137
```

```
which.min(rf$test$err.rate[ ,1])
```

```
## [1] 51
```

```
pred.prob.tree=predict(newtree, data.test)[,2]
pred.prob.rf=predict(rf, newdata=data.test, type="prob")[ ,2]
pred.prob.bag=predict(bag, newdata=data.test, type="prob")[ ,2]
library(pROC)
```

```
## Type 'citation("pROC")' for a citation.
```

```
##
## Attaching package: 'pROC'
```

```
## The following objects are masked from 'package:stats':
##
##     cov, smooth, var
```

```
par(mfrow=c(1,3))
roc(data.test$athlete_display_name, pred.prob.tree, plot=TRUE, print.auc=
      TRUE, main="Single Tree")
```

```
## Setting levels: control = G, case = L
```

```
## Setting direction: controls < cases
```

```
##
## Call:
## roc.default(response = data.test$athlete_display_name, predictor = pred.prob.tree,     plot = TRUE, p
##
## Data: pred.prob.tree in 15 controls (data.test$athlete_display_name G) < 16 cases (data.test$athlete_
## Area under the curve: 0.7521
```

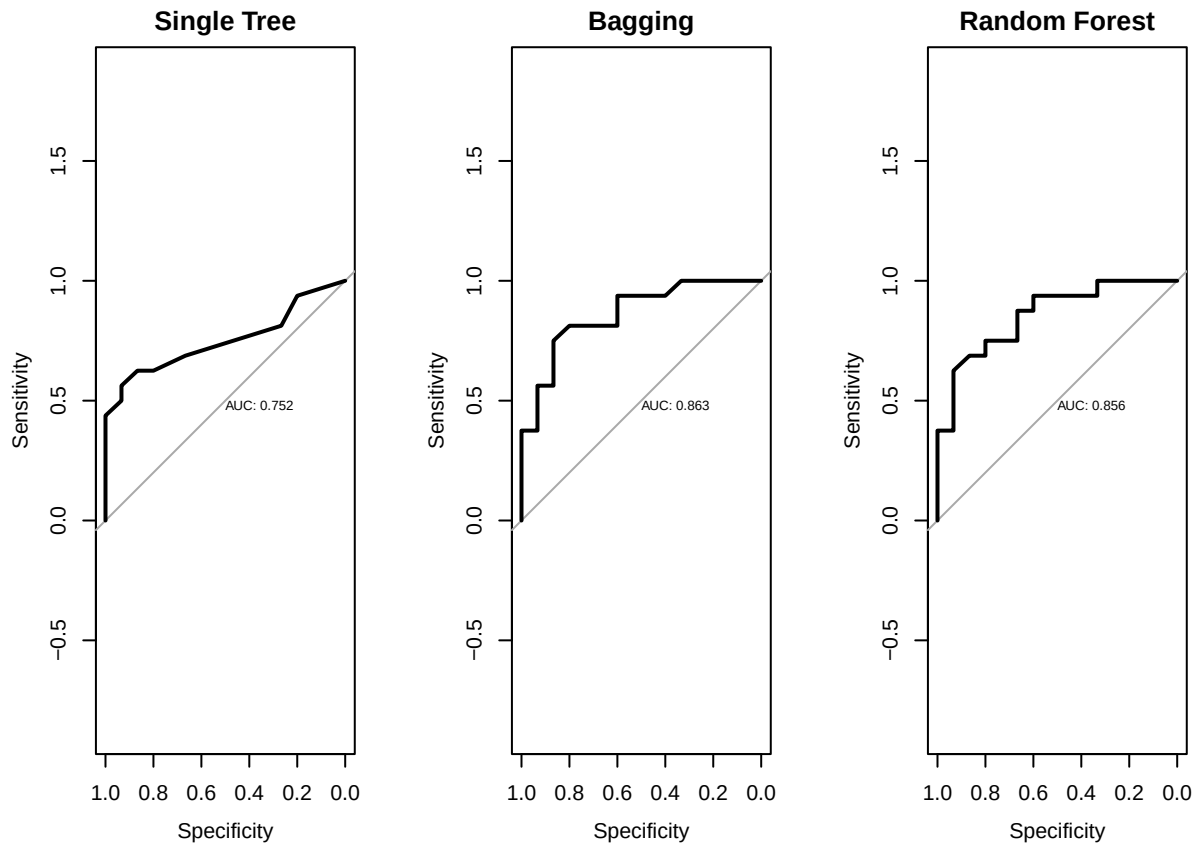
```
roc(data.test$athlete_display_name, pred.prob.bag, plot=TRUE, print.auc=
      TRUE, main="Bagging")
```

```
## Setting levels: control = G, case = L
## Setting direction: controls < cases
```

```
##
## Call:
## roc.default(response = data.test$athlete_display_name, predictor = pred.prob.bag,     plot = TRUE, pr
##
## Data: pred.prob.bag in 15 controls (data.test$athlete_display_name G) < 16 cases (data.test$athlete_d
## Area under the curve: 0.8625
```

```
roc(data.test$athlete_display_name, pred.prob.rf , plot=TRUE, print.auc=
      TRUE, main="Random Forest")
```

```
## Setting levels: control = G, case = L
## Setting direction: controls < cases
```




```
##
## Call:
## roc.default(response = data.test$athlete_display_name, predictor = pred.prob.rf,     plot = TRUE, pr
##
## Data: pred.prob.rf in 15 controls (data.test$athlete_display_name G) < 16 cases (data.test$athlete_d
## Area under the curve: 0.8562
```