NBAcase_B_Model

Team

5/6/2021

Contents

EDA

```r
df = read.csv('NBA_Data.csv')
#summary(df)

#df_2 = read.csv('NBADataa_test.csv')
#df_2 = df_2[-c(4,5,6),]
#df_test = df_2[,-c(1,2,33,35)]
```

```r
library(ggplot2)
theme_set(
  theme_classic() +
    theme(legend.position = "top")
  )

#ggplot(df, aes(x=Deal.Average.Salary)) +
#    geom_histogram(binwidth=1000000, colour="black", fill="white")
#ggplot(df, aes(x=Age)) +
#    geom_histogram(binwidth=1, colour="black", fill="white")
#ggplot(df, aes(x=G)) +
#    geom_histogram(binwidth=1, colour="black", fill="white")
#ggplot(df, aes(x=MP)) +
#    geom_histogram(binwidth=100, colour="black", fill="white")
#ggplot(df, aes(x=FG)) +
#    geom_histogram(binwidth=10, colour="black", fill="white")
#ggplot(df, aes(x=eFG.)) +
#    geom_histogram(binwidth=0.01, colour="black", fill="white")

library(ggcorrplot)
corr = round(cor_pmat(df[-1]),2)
```

```r
ggcorrplot(corr, type = "lower")
corr = as.data.frame(corr)
corr
```

Question: Which three metrics are most correlated with AAV and which three are most correlated with each other? Please include this in your summary.

Answer: Deal_Year, Age, X3PAr are most correlated with AAV(Salary). Pairs with the highest corr coef: Deal_Year & FT, X3PAr & RFA, FG & AST, X3P & DRB, etc. Variables with highest corr with others: Deal_Year, Age, RFA

Modeling

Multivariate Linear Reg - Stepwise

```r
#df_model = df[,-c(1,3,4)]
#reg = lm(Deal.Average.Salary~Age+G+GS+MP+FGA+FG.+X3P.
#         +eFG.+FTA+FT.+ORB+TRB+STL+BLK+TOV+PF+PTS+X3PAr+FTr+USG.+OWS+DWS+WS
#         +Deal_Year*FT+X3PAr*RFA+FG*AST+X3P*DRB, df_model)
#summary(reg)
#Stepwise
null1 = lm(Deal.Average.Salary ~ 1, df_model)
full1 = lm(Deal.Average.Salary ~ ., df_model)
stepwise1 =step(null1, scope=list(lower=null1, upper=full1), direction="both")
summary(stepwise1)

reg = lm(formula = Deal.Average.Salary ~ WS + G + FGA + RFA + ORB +
    USG. + DWS + FG + X3PAr, data = df_model)

summary(reg)
#plot(reg)

y_predicted_mlr = predict(reg, newx = x)
sst_mlr = sum((df_model$Deal.Average.Salary - mean(df_model$Deal.Average.Salary))^2)
sse_mlr = sum((y_predicted_mlr - df_model$Deal.Average.Salary)^2)
rsq_mlr = 1 - sse_mlr/sst_mlr
print(paste("R-Square - MLR:", rsq_mlr))
print(paste("Res Sum of Sq - MLR:", sse_mlr))

y_pred_reg = predict(reg, df_test[,c("WS", "G", "FGA", "RFA", "ORB", "USG.", "FG",  "X3PAr", "DWS")])
print(paste("Predicted Value - MLR:", df_2$Player, y_pred_reg))
```

```r
df_JG = read.csv('JG_data.csv')
df_tJG = df_JG[,-1]

y_pred_reg = predict(reg, df_tJG[,c("WS", "G", "FGA", "RFA", "ORB", "Deal_Year", "USG.", "Age", "DWS")]
print(paste("Predicted Value - JG:", df_JG$Player, y_pred_reg))
```

LASSO

```r
library(caTools)
set.seed(123)
split = sample.split(df$Deal.Average.Salary, SplitRatio = 0.70)
```

```r
df_train = subset(df, split == TRUE)
df_test = subset(df, split == FALSE)

#install.packages("glmnet")
library(glmnet)
```

```
## Loading required package: Matrix
## Loaded glmnet 4.1-1
```

```r
lambdas = 10^seq(2, -3, by = -.1)

x_train = data.matrix(df_train[, -c(1,2)])

lasso_reg = cv.glmnet(x_train, df_train$Deal.Average.Salary, alpha = 1)
lambda_best = lasso_reg$lambda.min

reg_lasso = glmnet(x_train, df_train$Deal.Average.Salary, alpha=1, lambda = lambda_best)
coef(reg_lasso)
```

```
## 32 x 1 sparse Matrix of class "dgCMatrix"
##                         s0
## (Intercept)  1.228923e+09
## Deal_Year   -6.062005e+05
## Age         -3.549166e+04
## G           -1.256735e+05
## GS           2.048672e+04
## MP           .
## FG           .
## FGA          1.428391e+04
## FG.          .
## X3P          1.754752e+04
## X3PA         .
## X3P.        -1.298366e+06
## eFG.         .
## FT           .
## FTA          .
## FT.          .
## ORB          .
## DRB         -2.746186e+03
## TRB          .
## AST          6.206511e+03
## STL          .
## BLK          .
## TOV         -1.756029e+04
## PF          -1.049853e+04
## PTS          .
## X3PAr       -1.062662e+06
## FTr         -4.542440e+05
## USG.        -3.841536e+04
## OWS          .
## DWS          1.111473e+06
## WS           1.425767e+06
## RFA          1.571643e+06
```

```r
# testing
x_test = data.matrix(df_test[, -c(1,2)])
y_pred_LASSO = predict(reg_lasso, s = lambda_best, newx = x_test)

sst = sum((df_test$Deal.Average.Salary - mean(df_test$Deal.Average.Salary))^2)
sse = sum((y_pred_LASSO - df_test$Deal.Average.Salary)^2)
rsq = 1 - sse/sst
print(paste("R-Square - LASSO:", rsq))
```

```
## [1] "R-Square - LASSO: 0.742299569765037"
```

```r
print(paste("Res Sum of Sq - LASSO:", sse))
```

```
## [1] "Res Sum of Sq - LASSO: 2985615246839328"
```

```r
#print(paste("PV:", df_test$Player, y_pred_LASSO, df_test$Deal.Average.Salary))

Predicted = y_pred_LASSO
df_result = df_test[,c(1,2)]
df_result$Predicted = y_pred_LASSO
df_result$Error = df_result$Predicted-df_result$Deal.Average.Salary
df_result$Status = ifelse(df_result$Error<0, 'Overpaid', 'Underpaid')
df_result$AbsErrLess1Mil = ifelse(abs(df_result$Error)<1000000, 'Yes', 'No')

head(df_result)
```

```
##                 Player Deal.Average.Salary        1          1        1  1
## 2        Tobias Harris            36000000 22503247 -13496753 Overpaid No
## 4         Kevin Durant            41063925 31423713  -9640212 Overpaid No
## 5     Kristaps Porzingis          31650600 15942059 -15708541 Overpaid No
## 8         Kyrie Irving            34122650 29237305  -4885345 Overpaid No
## 11       Kawhi Leonard            34379100 28219269  -6159831 Overpaid No
## 16       Julius Randle            20700000 15503483  -5196517 Overpaid No
```

```r
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
df_result %>%
  group_by(Status) %>%
  summarise(Count = length(Player))%>%
  mutate(Percentage = Count / sum(Count))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)

## # A tibble: 2 x 3
##   Status[,"1"] Count Percentage
##   <chr>        <int>      <dbl>
## 1 Overpaid        71      0.493
```

```
## 2 Underpaid       73      0.507
```

```
df_result %>%
  group_by(AbsErrLess1Mil) %>%
  summarise(Count = length(Player))%>%
  mutate(Percentage = Count / sum(Count))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
## # A tibble: 2 x 3
##   AbsErrLess1Mil[,"1"] Count Percentage
##   <chr>                <int>      <dbl>
## 1 No                     102      0.708
## 2 Yes                     42      0.292
```

```
df_JG = read.csv('JG_data.csv')
df_tJG = df_JG[,-c(1,3,4)]

df_tJG = data.matrix(df_tJG[, -c(1)])

y_pred_LASSO_JG = predict(reg_lasso, s = lambda_best, newx = df_tJG)

print(paste("Predicted (LASSO) - JG:", df_JG$Player, y_pred_LASSO_JG))
print(paste("Actual - JG:", df_JG$Player, df_JG$Deal.Average.Salary))
```

Random Forest

```
library(randomForest)
reg_rf = randomForest(x = df_model[-1], y = df_model$Deal.Average.Salary,
                      ntree = 200)

y_predicted_rf = predict(reg_rf, newx = x)
sst_rf = sum((df_model$Deal.Average.Salary - mean(df_model$Deal.Average.Salary))^2)
sse_rf = sum((y_predicted_rf - df_model$Deal.Average.Salary)^2)
rsq_rf = 1 - sse_rf/sst_rf
print(paste("R-Square - RF:", rsq_rf))
print(paste("Res Sum of Sq - RF:", sse_rf))

y_pred_rf = predict(reg_rf, df_test)
print(paste("Predicted Value - RF:", df_2$Player, y_pred_rf))
```

Conclusion

LASSO Regression achieved the highest performance (highest residual sum of square and r square). The combo of predictors selected by LASSO can be found above with their respective coefficients.

Notes: coefficients can be directly interpreted by unit changes for Multivariate Reg and Lasso Reg, no transformations were applied to modeling. Random Forest does not support predictor interpretation.