Final Project_ss6

Seunggyun Shin

2022 11 30

Introduction Major League Baseball(MLB) is one of four outstanding sports leagues in the United States which are called "Big Four". With the fame of the sports and increasing demands of data analytics, importance of applications of data analytics in sports is increasing as well such as Sabermetrics. Especially in baseball, playing over 150 games per year, plenty of statistics are collected from MLB, which is the reason baseball is called "sports of statistics". However, it is difficult to make expectations about performances of players in the future as there are plenty of uncertainties in baseball. So, purpose of this project is to build the machine learning model to predict future performance of each player using and analyzing data collected throughout decades.

OPS(On-base Plus Slugging) is on of the most significant statistics used to evaluate abilities and productivity of batters.OPS is calculated through adding up OBP(On Base Percentage) and SLG(Slugging) which represent abilities of batters to get on bases and make extra-base hits to produce runs. In this project, by establishing three models to expect BA(Batting Average), IsoD(Isolated Discipline) and IsoP(Isolated Power), eventually expected OPS will be calculated.

- Batting Average = Hits / At-bat
- Isolated Discipline = On-base percentage - Batting Average (represents abilities of batters to distinguish strikes and balls)
- Isolated Power = Slugging - Batting Average (represents pure powers of batters to make extra-base hits)

Data collected from 1969 to 2019 will be used to build prediction models, and data collected on 2021 and 2022 will be used to make final prediction of 2023 season as 1969 is the cut-off of modern baseball (Stark, ESPN, 2006), and at least data from two seasons are needed to make prediction while 2020 season was shortened due to COVID-19 issue. In process to build models, only data of batters who satisfied at least half of minimum At-bat (half of 502) for more than two seasons in a row since statistics with small At-bat number will not be significant. In addition, setting regulation of the minimum At-bat(502) will drastically reduce the sample size to build models and will ignore some significant statistics. At-bat is fixed to 502 since 1962 as the number of games is fixed to 162 on 1962, and data of shortened seasons will be automatically ignored through filtering data only with certain At-bat number.

https://www.espn.com/mlb/columns/story?columnist=stark_jayson&id=2471349

Data Explanation In this project, three types of datasets will be used. 1. Batting dataset from Lahman library - Batting dataset contains data of batters from 1871 to 2021. 2. People dataset from Lahman library - People dataset contains personal information data of MLB player. 3. 2022 Major League batting data from Kaggle - 2022 batting data is collected from external source as the batting dataset only contains data until 2021.

Manipulating these three datasets, final data frame containing following variables are created.

https://www.kaggle.com/datasets/vivovinco/2022-mlb-player-stats

Prediction Variables - Season - BA (H / AB) - BABIP (H-HR) / (AB - HR + K + SF): Batting average of in-field hits - IsoD (OBP - BA) - IsoP (SLG - BA) - BBK (BB/K): Ratio of Base on Balls and Strike

outs - Age - Number of Seasons played - BA change from previous season(%) - BABIP change from previous season(%) - IsoD change from previous season(%) - IsoP change from previous season(%) - BBK change from previous season(%)

Response Variables - BA next season - IsoD next season - IsoP next season

- Data of actual number such as number of hits and home runs are not used as most of statistics are calculated through those numbers, and the number can vary depending on At-bat opportunities, which means multicollinearity issue can occur.
- IsoD, IsoP is used rather than OBP and SLG as OBP and SLG is calculated through adding IsoD and IsoP to batting average. If batting average increases, OBP and SLG will be increased automatically, which can also cause multicollinearity issue.
- Final OPS will be calculated through following equation. $OPS = 2 * BA + IsoD + IsoP$

Data Manipulations for analysis

```
data = Batting %>%
  filter(yearID >= 1969 & yearID < 2020) %>%
  group_by(playerID, yearID) %>%
  replace_na(list(HBP = 0 , SF = 0)) %>%
  summarise(yearID = yearID,
            G = sum(G),
            AB = sum(AB),
            R = sum(R),
            H = sum(H),
            X2B = sum(X2B),
            X3B = sum(X3B),
            HR = sum(HR),
            RBI = sum(RBI),
            SB = sum(SB),
            CS = sum(CS),
            BB = sum(BB),
            SO = sum(SO),
            HBP = sum(HBP),
            SH = sum(SH),
            SF = sum(SF)) %>%
  mutate(X1B = H - (X2B + X3B + HR)) %>%
  mutate(SBpct = (SB / (SB + CS))) %>%
  mutate(OBP = (H + BB + HBP) / (AB + BB + HBP + SF)) %>%
  mutate(SLG = (X1B + 2* X2B + 3 * X3B + 4 * HR) / AB) %>%
  mutate(OPS = OBP + SLG) %>%
  mutate(AVG = H/AB) %>%
  mutate(BABIP = (H - HR) / (AB - HR - SO + SF)) %>%
  mutate(IsoD = OBP - AVG) %>%
  mutate(IsoP = SLG - AVG) %>%
  mutate(BBK = BB/SO) %>%
  arrange(playerID)
```

Prediction variables needed are calculated through equations using numbers from the original data set.

```
Names = People %>%
  mutate(fullname = paste(nameFirst, nameLast)) %>%
```

```
  dplyr::select(playerID, birthYear, debut, fullname)

debutseason = c()
for (i in 1:nrow(Names)){
  debutseason[i] = strsplit(Names$debut, split = "-", fixed = T)[[i]][1]
}

Names$debutseason = as.integer(debutseason)
Names = Names %>%
  dplyr::select(-debut)
```

Data frame containing full names, birth years, debut seasons and playerIDs is created through processing Names data set to calculate number of seasons played and age. Also, playerID is used to inner join two data set - Names and Batting.

```
#Leave data with at least 3 seasons (to compare with previous season and next season, get response vari
data5 = data4 %>%
  filter(AB >= 502 *1/2) %>%
  distinct() %>%
  arrange(playerID, yearID) %>%
  group_by(playerID) %>%
  mutate(Avg_chg = (AVG - lag(AVG)) * 100/lag(AVG),
         BABIP_chg = (BABIP - lag(BABIP)) * 100/lag(BABIP),
         IsoD_chg = (IsoD - lag(IsoD)) * 100/lag(IsoD),
         IsoP_chg = (IsoP - lag(IsoP)) * 100/lag(IsoP),
         BBK_chg = (BBK - lag(BBK)) * 100/lag(BBK)) %>% # Prediction variables
  mutate(Avg_next = lead(AVG),
         IsoD_next = lead(IsoD),
         IsoP_next = lead(IsoP)) %>% #response variables
  drop_na() %>%
  dplyr::select(-c(minAB_pct, minAB, AB))


#Final dataset for modeling
head(data5, 10)
```

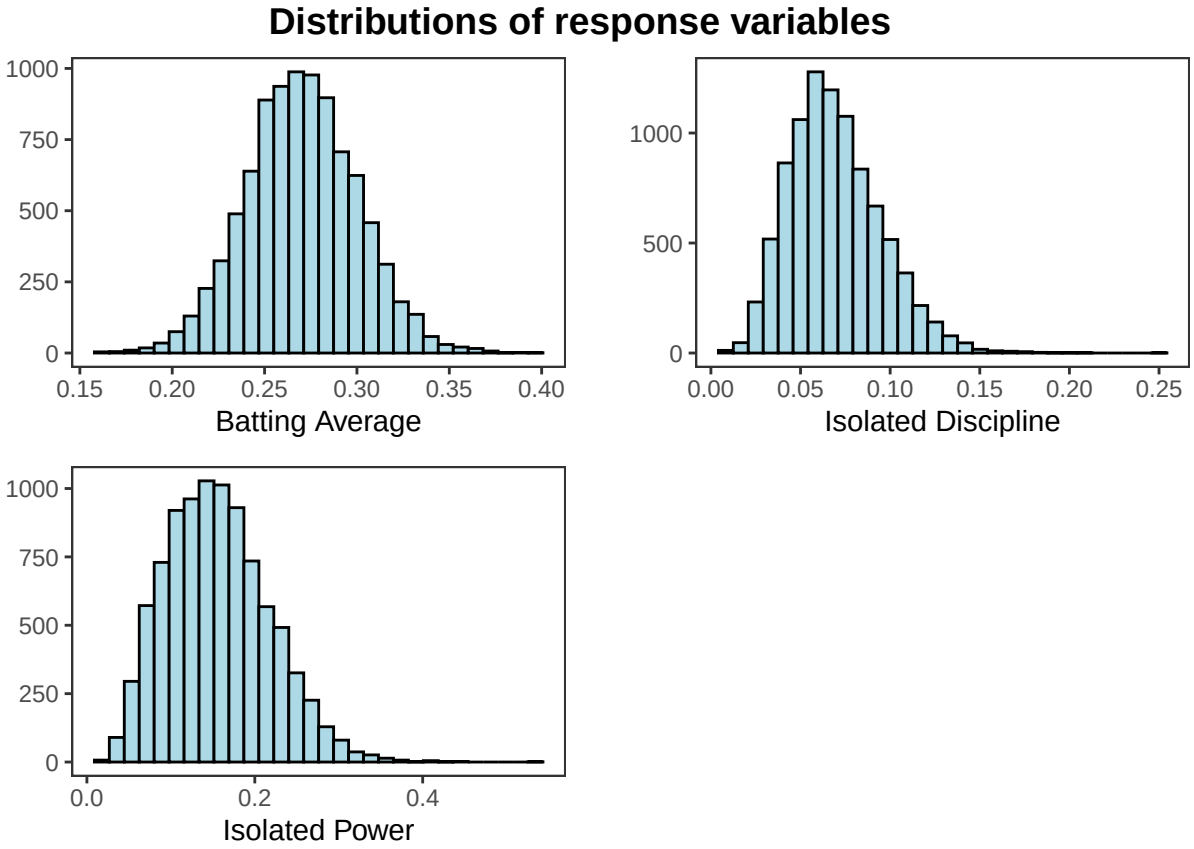
```
## # A tibble: 10 x 18
## # Groups:   playerID [3]
##    playerID  fulln~1 yearID   AVG BABIP   IsoD  IsoP   BBK   age seasons Avg_chg
##    <chr>     <chr>    <int> <dbl> <dbl>  <dbl> <dbl> <dbl> <dbl>   <dbl>   <dbl>
##  1 aaronha01 Hank A~   1970 0.298 0.276 0.0862 0.275 1.17     37      17  -0.456
##  2 aaronha01 Hank A~   1971 0.327 0.291 0.0828 0.341 1.22     38      18   9.66
##  3 aaronha01 Hank A~   1972 0.265 0.235 0.125  0.249 1.67     39      19 -19.0
##  4 aaronha01 Hank A~   1973 0.301 0.256 0.101  0.342 1.33     40      20  13.6
##  5 aaronha01 Hank A~   1974 0.268 0.242 0.0736 0.224 1.34     41      21 -11.1
##  6 aaronha01 Hank A~   1975 0.234 0.238 0.0977 0.120 1.37     42      22 -12.4
##  7 abbotku01 Kurt A~   1995 0.255 0.302 0.0628 0.198 0.327    27       3   2.20
##  8 abbotku01 Kurt A~   1996 0.253 0.343 0.0541 0.175 0.222    28       4  -0.643
##  9 abbotku01 Kurt A~   1997 0.274 0.354 0.0408 0.159 0.206    29       5   8.17
## 10 abreubo01 Bobby ~   1999 0.335 0.391 0.110  0.214 0.965    26       4   7.47
## # ... with 7 more variables: BABIP_chg <dbl>, IsoD_chg <dbl>, IsoP_chg <dbl>,
## #   BBK_chg <dbl>, Avg_next <dbl>, IsoD_next <dbl>, IsoP_next <dbl>, and
## #   abbreviated variable name 1: fullname
```

After joining two datasets and calculating variables needed, data was grouped by playerID to filter players filled half of At-bat number for three seasons as data of three seasons is necessary to establish prediction models - two seasons to create prediction variables and the third season to create response variables.

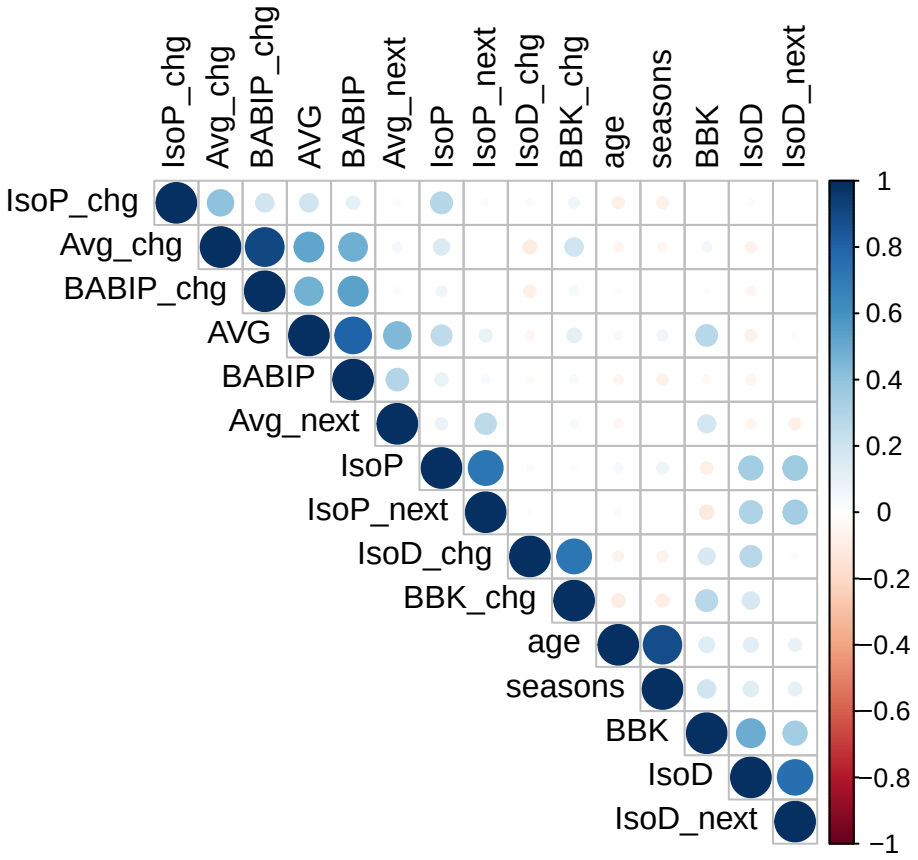
Final Data set for data modeling is above.

Data exploration

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



Distributions of three response variables are above. All three variables are seemingly normal distributed while IsoD and IsoP are bit right skewed that they are very appropriate for applying to data modeling.



Graphic above is the correlation plot of the data set. You can see general correlation between variables.

```
vif(mod_vif)
```

```
##       AVG     BABIP      IsoD      IsoP       BBK       age   seasons   Avg_chg
## 10.441789  7.441908  3.700210  2.726436  4.567115  4.776738  4.992958 17.561406
## BABIP_chg  IsoD_chg  IsoP_chg   BBK_chg
## 12.992701  3.944507  2.022385  4.629324
```

According to variance inflation factor analysis, there are some variables with multicollinearity.

```
vif(mod_vif2)
```

```
##       AVG     BABIP      IsoD      IsoP       BBK       age   seasons BABIP_chg
##  9.489841  6.897952  3.600876  2.703982  4.336910  4.776107  4.981229  1.476846
##  IsoD_chg  IsoP_chg   BBK_chg
##  2.398114  1.176965  2.423689
```

By removing BA_chg, multicollinearity issue can be removed. This result is used in further linear regression modeling, and regression models that can resolve this issue are tested as well.

```
#train, test split (7:3)
df = df[ ,-1]
test = sample(nrow(df), nrow(df) * 0.3, replace = F)
train = df[-test ,]
test = df[test, ]
```

Modeling *Modeling for Batting Average*

- Best linear model: stepwise AIC

Model(BA)	R^2	MSE
Linear Regression	0.268	0.000678
KNN	-0.039	0.00096
Random Forest	0.240	0.000684
Lasso Regression	0.268	0.000679
Xgboost	0.250	0.000695

Best: Linear Regression

Modeling for Isolated Discipline

Model(IsoD)	R^2	MSE
Linear Regression	0.606	0.0002507
KNN	-0.023	0.000652
Random Forest	0.587	0.000261
Lasso Regression	0.606	0.0002508
Xgboost	0.601	0.000254

Best: Linear Regression

Modeling for Isolated Power

```
MSE(test_IsoP$IsoP_next, lassopred_IsoP, test_IsoP)
```

```
## [1] "R_square : 0.579680622249021     MSE :  0.00159067280278939"
```

Model(IsoP)	R^2	MSE
Linear Regression	0.578	0.00163
KNN	-0.013	0.00387
Random Forest	0.550	0.00171
Lasso Regression	0.573	0.00154
Xgboost	0.561	0.00167

Best: Lasso Regression

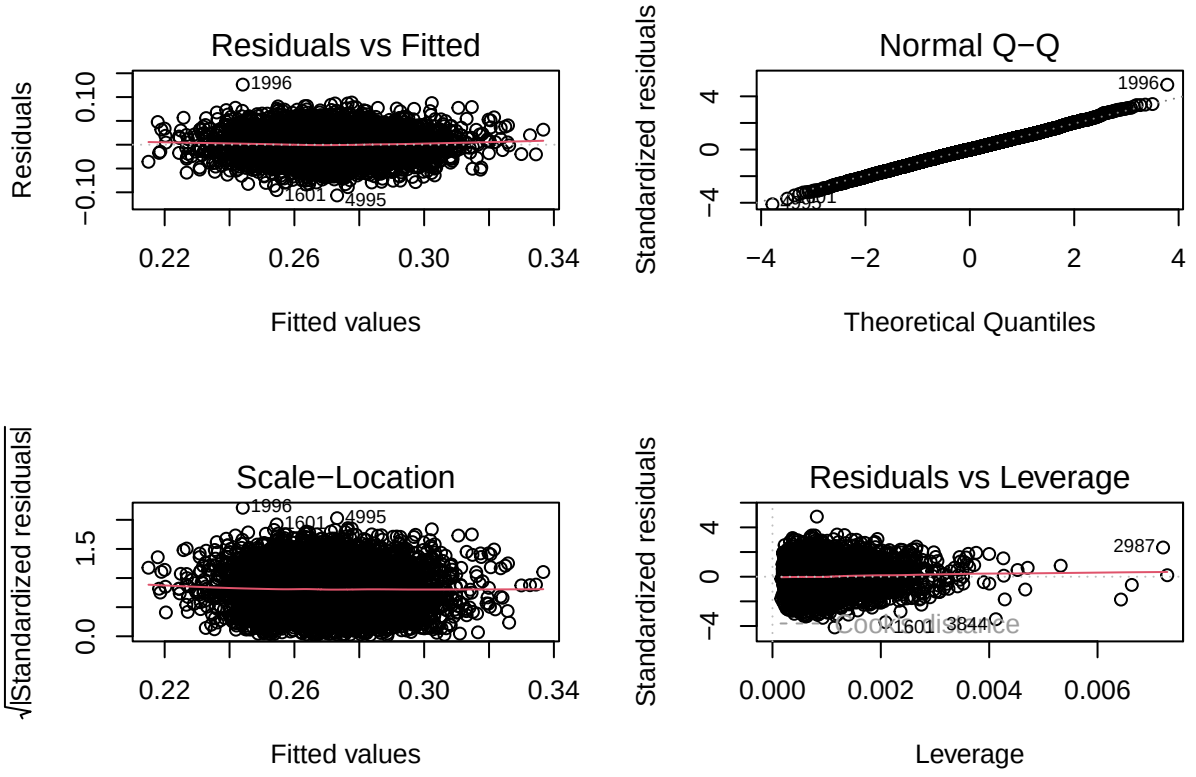
```
final_IsoP_model = lasso_best_IsoP
coef(final_IsoP_model)
```

```
## 13 x 1 sparse Matrix of class "dgCMatrix"
##                         s0
## (Intercept)  0.0853256948
## AVG         -0.0315186906
## BABIP        .
## IsoD         0.2498863436
```

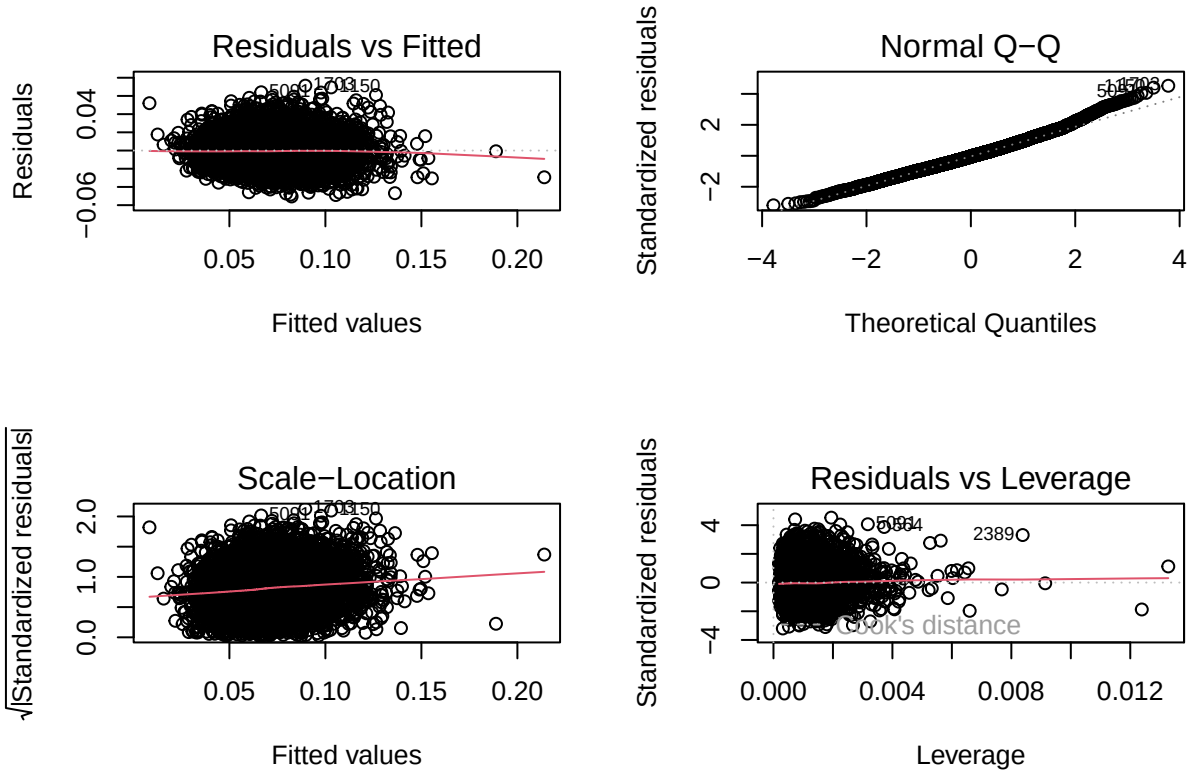
```
## IsoP          0.7553075372
## BBK          -0.0139243974
## age          -0.0016183534
## seasons       0.0004472935
## Avg_chg      -0.0010758773
## BABIP_chg     0.0008333508
## IsoD_chg     -0.0001775272
## IsoP_chg     -0.0002537520
## BBK_chg       0.0001438887
```

5 types of regression modeling methods were used. Linear regression models using all the variables, VIF analysis and stepwise AIC were created and linear model with the highest R^2 value and the lowest MSE(Mean Square Error) was chosen. K Nearest Neighbors regression did not have effective prediction that it had negative R^2 values for all three predictions. Random Forest, Lasso Regression and Xgboost modeling were used to deal with multicollinearity issue of the data set. By comparing R^2 and MSE values, the best modeling method was chosen.

```
#Plots for Batting Average model
par(mfrow = c(2,2))
plot(final_BA_model)
```




```
#Plots for Isolated Discipline model
par(mfrow = c(2,2))
plot(final_IsoD_model)
```



According to assumption check using plots, for both linear regression models chosen for batting average and isolated discipline, both models met normality assumption.

Preprocess data for final prediction

```
a = data2022$Name
a = gsub('\\?', ' ', a)
a = gsub('\\*', '', a)
a = gsub('\\#' , "", a)

data2022$Name = a
colnames(data2022)[2] = "fullname"
df2022 = inner_join(data2022, Names, by = "fullname")
df2022$yearID = 2022


df2022_2 = df2022 %>%
  filter(Lg != "MLB") %>%
  group_by(playerID, yearID) %>%
  replace_na(list(HBP = 0 , SF = 0)) %>%
  summarise(yearID = yearID,
            G = sum(G),
            AB = sum(AB),
            R = sum(R),
            H = sum(H),
            X2B = sum(X2B),
            X3B = sum(X3B),
```

```
            HR = sum(HR),
            RBI = sum(RBI),
            SB = sum(SB),
            CS = sum(CS),
            BB = sum(BB),
            SO = sum(SO),
            HBP = sum(HBP),
            SH = sum(SH),
            SF = sum(SF),
            fullname = fullname,
            debutseason = debutseason,
            birthYear = birthYear) %>%
  mutate(X1B = H - (X2B + X3B + HR)) %>%
  mutate(SBpct = (SB / (SB + CS))) %>%
  mutate(OBP = (H + BB + HBP) / (AB + BB + HBP + SF)) %>%
  mutate(SLG = (X1B + 2* X2B + 3 * X3B + 4 * HR) / AB) %>%
  mutate(OPS = OBP + SLG) %>%
  mutate(AVG = H/AB) %>%
  mutate(BABIP = (H - HR) / (AB - HR - SO + SF)) %>%
  mutate(IsoD = OBP - AVG) %>%
  mutate(IsoP = SLG - AVG) %>%
  mutate(BBK = BB/SO)


df2022_3 = df2022_2 %>%
  mutate(seasons = yearID - debutseason + 1) %>%
  mutate(age = yearID - birthYear + 1) %>%
  mutate(minAB = ifelse(AB >= 502, yes = 1, no = 0))

df2022_4 = df2022_3 %>%
  summarise(fullname = fullname,
            yearID = yearID,
            AB = AB,
            AVG = AVG,
            BABIP = BABIP,
            IsoD = IsoD,
            IsoP = IsoP,
            BBK = BBK,
            seasons = seasons,
            age = age,
            minAB = minAB) %>%
  distinct()
```

For final prediction, as data of 2022 season is collected separately, data manipulation was necessary to standardize formats of data sets and to combine them. Due to encoding error, to eliminate exclamation marks in the player name, string management was used.

```
df_2122 = rbind(data2021_4, df2022_4) %>%
  group_by(playerID) %>%
  mutate(cnt = n()) %>%
  arrange(playerID) %>%
  filter(cnt >= 2)
```

```
#Final dataset for final prediction
df2122_2 = df_2122 %>%
  filter(AB >= 100) %>%
  distinct() %>%
  arrange(playerID, yearID) %>%
  group_by(playerID) %>%
  mutate(Avg_chg = (AVG - lag(AVG)) * 100/lag(AVG),
         BABIP_chg = (BABIP - lag(BABIP)) * 100/lag(BABIP),
         IsoD_chg = (IsoD - lag(IsoD)) * 100/lag(IsoD),
         IsoP_chg = (IsoP - lag(IsoP)) * 100/lag(IsoP),
         BBK_chg = (BBK - lag(BBK)) * 100/lag(BBK)) %>% # Prediction variables
  filter(yearID == 2022)

df2122_2 = na.omit(df2122_2)
```

Final data set was created by joining two data sets.

```
#make final prediction
pred_final_BA = predict(final_BA_model, newdata = df2122_2)
pred_final_IsoD = predict(final_IsoD_model, newdata = df2122_2)

IsoP_final_x = df2122_2 %>%
  dplyr::select(-c(fullname, AB, minAB, cnt, yearID))
IsoP_final_x = IsoP_final_x[ , -1]

pred_final_IsoP = predict(final_IsoP_model, newx = as.matrix(IsoP_final_x))[,1]
```

```
#Final dataset
df2122_2$BA_2023 = pred_final_BA
df2122_2$IsoD_2023 = pred_final_IsoD
df2122_2$IsoP_2023 = pred_final_IsoP


final = df2122_2 %>%
  mutate(OBA_2023 = round(BA_2023 + IsoD_2023, 3),
         SLG_2023 = round(BA_2023 + IsoP_2023, 3),
         OPS_2023 = round(OBA_2023 + SLG_2023, 3)) %>%
  dplyr::select(fullname, BA_2023, OBA_2023, SLG_2023, OPS_2023)
```

```
## Adding missing grouping variables: ‘playerID‘
```

```
final = final[ , -1]
final$BA_2023 = round(final$BA_2023, 3)
```

Final predictions were made using final data set, and those predictions were combined into the data as the result.

Conclusion

```
#Top 10 - Batting Average
head(final %>% arrange(desc(BA_2023)), 10) %>%
  dplyr::select(fullname, BA_2023)
```

```
## # A tibble: 10 x 2
##    fullname        BA_2023
##    <chr>             <dbl>
##  1 Luis Arraez       0.298
##  2 Freddie Freeman   0.292
##  3 Tim Anderson      0.288
##  4 Luis Robert       0.288
##  5 Trea Turner       0.288
##  6 Yordan Alvarez    0.286
##  7 Nico Hoerner      0.285
##  8 Xander Bogaerts   0.284
##  9 Wander Franco     0.284
## 10 Jeff McNeil       0.284
```

```
#Top 10 - On base average
head(final %>% arrange(desc(OBA_2023)), 10) %>%
  dplyr::select(fullname, OBA_2023)
```

```
## # A tibble: 10 x 2
##    fullname         OBA_2023
##    <chr>               <dbl>
##  1 Juan Soto           0.414
##  2 Aaron Judge         0.392
##  3 Yordan Alvarez      0.38
##  4 Freddie Freeman     0.375
##  5 Mike Trout          0.375
##  6 Paul Goldschmidt    0.367
##  7 Alex Bregman        0.366
##  8 Bryce Harper        0.365
##  9 Brandon Nimmo       0.365
## 10 Jose Altuve         0.364
```

```
#Top 10 - Slugging
head(final %>% arrange(desc(SLG_2023)), 10) %>%
  dplyr::select(fullname, SLG_2023)
```

```
## # A tibble: 10 x 2
##    fullname         SLG_2023
##    <chr>               <dbl>
##  1 Aaron Judge         0.644
##  2 Mike Trout          0.616
##  3 Yordan Alvarez      0.604
##  4 Byron Buxton        0.584
##  5 Kyle Schwarber      0.563
##  6 Paul Goldschmidt    0.559
##  7 Austin Riley        0.558
##  8 Shohei Ohtani       0.553
##  9 Pete Alonso         0.552
## 10 Mookie Betts        0.547
```

```
#Top 10 - OPS
head(final %>% arrange(desc(OPS_2023)), 10) %>%
  dplyr::select(fullname, OPS_2023)
```

```
## # A tibble: 10 x 2
##    fullname         OPS_2023
##    <chr>               <dbl>
##  1 Aaron Judge         1.04
##  2 Mike Trout          0.991
##  3 Yordan Alvarez      0.984
##  4 Juan Soto           0.949
##  5 Paul Goldschmidt    0.926
##  6 Byron Buxton        0.918
##  7 Kyle Schwarber      0.914
##  8 Austin Riley        0.912
##  9 Bryce Harper        0.909
## 10 Shohei Ohtani       0.906
```

Above are top 10 players of 4 predictions for 2023 season. Comparing them to records of 2022, they seem quite reasonable. However, there are so many unexpected situations in baseball. So, it is recommended to use it as references rather than highly depending on the predictions.

Below are predictions of players with my personal interest.

```
## # A tibble: 1 x 5
##   fullname     BA_2023 OBA_2023 SLG_2023 OPS_2023
##   <chr>          <dbl>    <dbl>    <dbl>    <dbl>
## 1 Ha-Seong Kim   0.243    0.316    0.442    0.758
```

```
## # A tibble: 1 x 5
##   fullname    BA_2023 OBA_2023 SLG_2023 OPS_2023
##   <chr>         <dbl>    <dbl>    <dbl>    <dbl>
## 1 Ji-Man Choi   0.236    0.339    0.461      0.8
```

```
## # A tibble: 1 x 5
##   fullname      BA_2023 OBA_2023 SLG_2023 OPS_2023
##   <chr>           <dbl>    <dbl>    <dbl>    <dbl>
## 1 Shohei Ohtani   0.263    0.353    0.553    0.906
```

```
## # A tibble: 1 x 5
##   fullname   BA_2023 OBA_2023 SLG_2023 OPS_2023
##   <chr>        <dbl>    <dbl>    <dbl>    <dbl>
## 1 Mike Trout   0.279    0.375    0.616    0.991
```

```
## # A tibble: 1 x 5
##   fullname     BA_2023 OBA_2023 SLG_2023 OPS_2023
##   <chr>          <dbl>    <dbl>    <dbl>    <dbl>
## 1 Bryce Harper    0.28    0.365    0.544    0.909
```

```
## # A tibble: 1 x 5
##   fullname      BA_2023 OBA_2023 SLG_2023 OPS_2023
##   <chr>           <dbl>    <dbl>    <dbl>    <dbl>
## 1 Justin Turner   0.266     0.34     0.48     0.82
```