Johnny Addis - Player WAR Projection Model

PDF of code, Results sheet

Overview:

WAR stands for “Wins Above Replacement”. It is a statistic in baseball that determines how many wins a player is worth compared to a replacement-level player at his position.

I began my model by using the pybaseball module to take data from FanGraphs. I then used the *groupby* function to assign a unique ID to each player and make sure I was only using data from players who had at least 200 plate appearances in each season. I then dumped the data to .csv file.

Next, I set up a target for my model to predict. I defined this as “Next_WAR”. This is essentially the players WAR in the following year, but is also what I am trying to predict.

To clean the data, I removed players with only one season of data (can’t predict a next season if there isn’t one). I then counted up all the missing values I had in my columns. These were displayed as “NaN”. My new data frame contained only columns that had no missing values and the “Next_WAR” column. Finally, I converted all strings in my data frame into numbers, or simply removed them if they weren’t necessary.

Now things got a little tricky, after cleaning the data there were still 132 columns. This could lead to over-fitting the data. To try and simplify this, I used SequentialFeatureSelector from the Sci Kit Learn module to decide which columns were most important. This feeds into the Ridge Regression model. From here, I used a for-loop to generate predictions for “Next_WAR” and then tried to make the data more readable to put into an Excel sheet.

Below are some results and screenshots of my work. I identify changes between versions 1.0 and 2.0.

1.0:

Mean squared difference of my predicted “Next_WAR” values compared to actual: 2.8

```
[53]: from sklearn.metrics import mean_squared_error

      mean_squared_error(predictions["actual"],predictions["prediction"])

[53]: 2.800447128497333

[54]: batting["Next_WAR"].describe()

[54]: count    5575.000000
      mean        1.794511
      std         1.997104
      min        -3.400000
      25%         0.300000
      50%         1.500000
      75%         2.900000
      max        11.900000
      Name: Next_WAR, dtype: float64

[57]: 2.800447128497333 ** .5

[57]: 1.6734536529277806
```

Notes from above: The mean of all “Next_War” values is 1.795, with a standard deviation of 1.997. So, having a mean squared difference of 2.8 is not very accurate. Taking the square root of the mean squared difference gives us 1.673. This is now inside of the S.D. but still isn’t great.

2.0:

Changes: Version 1.0 only told the algorithm how the player did in the current season (Ex: to predict 2010 season WAR, algorithm used 2009 season only). This doesn’t take into account trends with a player’s performance that span over multiple seasons. I defined a function called *player_history.* This function has new columns that display how a player’s WAR is trending based on age and another column that displays the ratio of a player’s WAR in the current season compared to their WAR in the previous season. I applied this to the batting data frame. The new predictions were based on these new columns. The new mean squared difference was 2.7. Taking the square root gives us 1.646. This is a small improvement from version 1.0.

```
predictions = backtest(batting, rr, new_predictors)
```

```
mean_squared_error(predictions["actual"],predictions["prediction"])
```

```
2.6442590311967966
```

```
2.712051960597786 ** .5 #2.0 avg. diff
```

```
1.6468308840308363
```

Final Remarks:

Overall, I am satisfied with my margin of error for my model. If I were to do this project again, I would more clearly define what my goal was. I started off with a broad goal to just “predict player’s WARs”. The deeper into the project I got, I realized how many blockers there were and parameters I didn’t originally consider. I decided to predict players from 2002-2022 instead of current players this season because I wanted as many data points as possible to determine the accuracy of my model. A better understanding of baseball sabermetrics would also have helped as I would have been able to determine which stats had the biggest impact on WAR as opposed to letting SequenceFeatureSelector decide. Below are the coefficients for each stat used in the Ridge Regression model. A bigger coefficient means it had a bigger impact on the prediction.

```
Age             -2.904808
WAR             -2.177549
AVG+            -2.090964
K%+             -1.176602
BU              -0.949863
O-Swing%        -0.787144
PH              -0.630941
war_diff        -0.594740
SH              -0.594470
3B              -0.384123
war_corr        -0.130468
wRC+            -0.057921
player_season    0.013822
Pos              0.228871
Def              0.373032
F-Strike%        0.644815
Oppo%+           0.698548
O-Contact%       0.811264
Spd              0.835289
SB               0.859852
Med%             1.117624
IBB              1.720475
war_season       3.381125
Hard%+           3.403152
dtype: float64
```