Analyzing the pitcher performance through years

Chen Wang

Ⅰ Abstract

This project focuses on forecasting two key aspects in baseball performance : score difference and the reward system associated with pitcher performance. Through a meticulous data preprocessing pipeline, feature engineering, and model selection, we tailored our approach to address each objective.

Ⅱ Introduction

In the realm of baseball analytics, predicting performance metrics such as score differences and reward systems hold significant value. This project delves into the complexities of forecasting these metrics for individual pitchers in each game. By leveraging advanced machine learning techniques, we aim to provide insights that contribute to a deeper understanding of player performance and strategic decision making in baseball.

Ⅲ Motivation

The motivation behind this project lies in the intersection of data mining and real-world applications in baseball analytics. Accurate prediction of score differences and pitcher performance rewards enhances our ability to decipher patterns and trends in player behavior. Such insights can be invaluable for coaches, analysts, and team management, aiding in strategic planning and performance optimization. The project's relevance extends beyond the realm of sports, showcasing the practical implications of data mining methodologies in extracting meaningful information from complex datasets.

Ⅳ Related Work

Several studies in the realm of baseball analytics have laid the foundation for our approach, emphasizing the importance of various factors in predicting pitcher performance and game outcomes. One analysis focuses on the correlation between a pitcher's Earned Run Average and their average fastball speed, utilizing linear regression. Additionally, a study addresses challenges in predicting starting pitcher performance, considering factors such as skill volatility, defensive impact, and luck on batted-ball outcomes, utilizing regression tree methods. Furthermore, a method for analyzing a baseball pitcher's performance based on statistical data mining emphasizes the extraction of pitchers, pitches and results. These references collectively inform our project, where we predict score differences and forecast reward systems by leveraging various features and choosing types of models to use for the prediction.

Ⅴ Methodology

1. Data

We have a collection of data from 90 players, covering 92 different features. This data comes from a total of 258 games that took place in the year 2022. In our dataset, there are around 226,000 data points, and some of the features include things like 'release speed', 'pitch type', 'release spin rate', 'zone', and more. More detailed features of our data are in the Appendix.

As we delve into the dataset, our analysis revolves around two primary objectives, each centered on the performance of individual players in every game. The initial goal is to predict score differences, providing insights into how one team may outperform the other. The second objective is to forecast outcomes linked to the reward system. This entails using various features to predict the consequences and rewards associated with pitched balls, offering

valuable insights into the potential outcomes of specific pitches. We will explore each of these objectives in more detail.

2. Preprocessing

- Handling Missing Values
  We took steps to enhance the quality of our data by addressing missing values. Specifically, we removed rows where important features such as ‘pitcher’, ’batter’, ’effective speed’, ’release spin rate’ and ‘release extension’ had null values. This resulted in the removal of 500 data points out of the initial 226,000

- Label Encoding
  The ‘pitch type’ feature, indicating different pitching postures, was consolidated into a single categorical element. Recognizing the significance of pitch types for both pitchers and individual pitches, we transformed the singular ‘pitch type’ feature into multiple categorical features. Using label encoding, we expanded and organized the distinct pitch types.

	index	# of rows
0	FF	71496
1	SI	35392
2	SL	33980
3	CH	27319
4	CU	20410
5	FC	18173
6	KC	7405
7	ST	5831
8	FS	4504
9	SV	873
10	CS	63
11	PO	10

Figure 1. The numbers of distinct pitch types appear in the overall games

	index	new added colum names
0	26	pitch_SI
1	27	pitch_CU
2	28	pitch_FC
3	29	pitch_FF
4	30	pitch_CH
5	31	pitch_CS
6	32	pitch_SL
7	33	pitch_ST
8	34	pitch_FS
9	35	pitch_KC
10	36	pitch_PO
11	37	pitch_SV

Figure 2. The expanded pitch types after implementing label encoding

- Identify Pitcher’s Home Team
  In our dataset, we have information about the ‘Home Team’ and ‘Away Team’ for each game. To calculate score differences accurately, we need to figure out if the pitcher is playing for the home team.
  To do this, we count how many times each pitcher appears as part of the home and away teams. Then, we determine the home team associated with the maximum count for each pitcher.

- Creating a new feature ‘reward’

Drawing from the insights in the referenced paper[4], we introduced a new feature called ‘reward’ by assigning specific points to each pitch. For example, if a pitched ball is hit by the batter and safely reaches first base, it gets a reward point of ‘-1’. If a ball results in a strikeout, the pitch is awarded a score of ‘1’.
We’ve identified a total of 59 distinct rewards based on different outcomes for each pitch.

events	description	reward
single	hit_into_play	-1
	foul	0
	blocked_ball	0
	ball	-1
	called_strike	0
field_out	hit_into_play	1
strikeout	swinging_strike	1
double	hit_into_play	-2
strikeout	swinging_strike_	1
	swinging_strike	1
walk	blocked_ball	-1
sac_fly	hit_into_play	-1
strikeout	called_strike	1
home_run	hit_into_play	-4

Figure 3. Part of reward system
(You can find the detailed reward table in the Appendix for reference)

- Grouping pitches by pitcher and dates
  Each date corresponds to a single game, and within each game, multiple pitchers may have played. To better analyze pitcher performance, we grouped the pitches based on the pitcher’s identity and the date of the game.
  During this grouping process, we calculated the average score for features like ‘spin rate’ and ‘release speed’ to capture an overall performance metric for each pitcher in a given game. Additionally, we determined the maximum values of ‘post away score’ and ‘post home score’ to capture the final scores of the last game and the pitcher played in. To provide a comprehensive measure of the pitcher’s performance, we summed up the rewards associated with each pitch. This grouping strategy allows us to evaluate the overall effectiveness of each pitcher in the context of individual games.

	game_date	pitcher	release_speed	release_pos_x	release_pos_z	zone	pfx_x	pfx_z	plate_x	plate_z	...
0	2022-04-07	425794	82.396296	-1.156667	6.359136	8.790123	-0.064074	0.366049	0.062963	2.374938	...
1	2022-04-07	425844	84.888095	-1.186429	6.215476	9.392857	-0.084524	0.598810	-0.110714	2.143571	...
2	2022-04-07	506433	90.113043	-1.784783	5.627500	9.543478	0.098152	0.693696	-0.010326	2.085543	...
3	2022-04-07	571578	87.013158	2.206842	6.367237	9.881579	0.424605	0.805921	-0.125263	2.182763	...
4	2022-04-07	608331	87.591667	1.307857	6.072976	8.333333	-0.124167	0.249524	-0.332619	2.638452	...

Figure 4. Grouped data snapshot

3. Define Target Variable
   Since we have two distinct objectives, we’ve established two separate target variables to align with each purpose.
   a. Score Difference in Each Pitcher for Each Game
      Calculated the score difference based on the team the pitcher belongs to. If the pitcher is part of the home team, the score difference is computed as follows :

‘ diff score = post home score – post away score’. Conversely, if the pitcher is in the away team, the score difference is calculated as : ‘diff score = post away score – post home score’.

b. Summed Up Reward Scores
For the second objective, we focused on the total reward scores. This involved adding up the individual reward scores associated with each pitch, providing an overall measure of a pitcher’s performance in terms of the assigned rewards.

4. Data Split

After grouping data by pitcher and date, we have a total of 2600 rows for total. We utilize the train test split function to split our dataset into training and testing sets. The variables X_train, X_test represent the features, while y_train and y_test correspond to the target variables, for each diff score, reward. Test size parameter is set to 0.1, indicating a 10% portion of the data reserved for testing.

5. Modeling

We used multiple regression models to predict the target variables. The models we chose are :

c. Linear Regression
d. LASSO Regression
e. SVR
f. Random Forest Regressor
g. KNN Regressor
h. MLP Regressor (neural network)
i. Decision Tree Regressor
j. Gradient Boosting Regressor

- Hyperparameter Tuning
  To optimize our model’s performance, we employed GridSearchCV, a technique that systematically explores various combinations of hyperparameters defined in a given dictionary.
  For more detailed information about the specific parameters and their ranges, refer to the Appendix. The GridSearchCV process enables us to fine-tune our model effectively by evaluating numerous combinations and selecting the optimal configuration based on performance metrics.

VI Evaluation

In our evaluation process, we adopt the average of results from various matches as our measurement standard, aligning with the optimization goal of MSE. Given the size of our dataset, each variable X corresponds to multiple target values y. The primary aim of MSE is to minimize the discrepancy between our predicted values and the average values of these multiple y’s. This choice of MSE as our evaluation metric ensures a robust assessment of our prediction performances, emphasizing the precision and closeness of our predictions to the overall trends in the data.

VII Results

- Predicting the Score Difference
  MLP Regressor emerged as the top performer for predicting the score difference. It exhibited the smallest Mean Square Error, indicating superior accuracy in our forecasting. The best parameters for this model were found to be :

  Activation function: 'relu'
  Alpha (regularization term): 0.0001
  Hidden layer sizes: (50,)
  Learning rate: 'adaptive'

- Predicting the Reward System
  The Linear Regression model demonstrated superior performance. Its effectiveness in capturing the relationship between features and reward scores positioned it as the most suitable model.

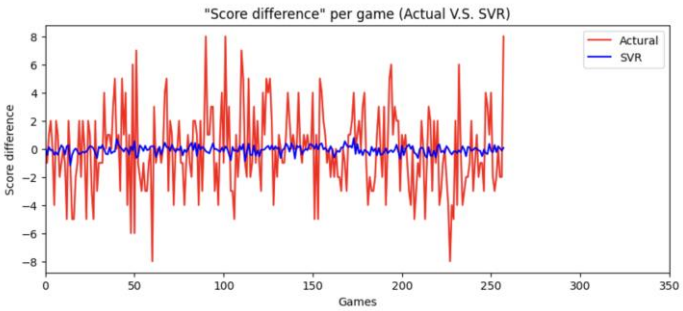
	Score difference_MSE	reward_MSE
Linear regression	8.388365	54.845142
LASSO	8.390942	56.320243
SVR	8.289725	58.538512
Random Forest	8.366556	60.175789
K-nearest neighbor	9.667866	73.982142
Neural Network	8.424831	67.632269
Decision Tree	10.870371	87.449264
Gradient Boost	8.343152	57.299927

Figure 5. The MSE results for ‘score difference’ and ‘reward’ to compare between predicting the score differences or rewards of each

In the concluding phase, we present distinct visualizations showcasing the results obtained from the two forecasting models. The graphical representations show us the overall performance of the predicted reward scores closely correlates with the actual data, surpassing the performance of the score difference.
When predicting ‘score difference’, the standard deviation of the actual target value (y) is 2.92, whereas the standard deviation for the predicted target values is 0.49. On the other hand, predicting the ‘reward’ system reveals a more consistent performance. The standard deviation of the actual values is 8.28, and the standard deviation of the predicted target values is 3.01. This suggests that predicting reward scores exhibits a more stable and reliable performance compared to predicting the score difference, providing valuable insights into the effectiveness of our forecasting models.

	Score_difference	Reward
Actual_std	2.900000	8.280000
Prediction_std	0.490000	3.010000
Ratio	5.890000	2.750000

Figure 6.The standard deviation of the predictions and the actual data from both methods.



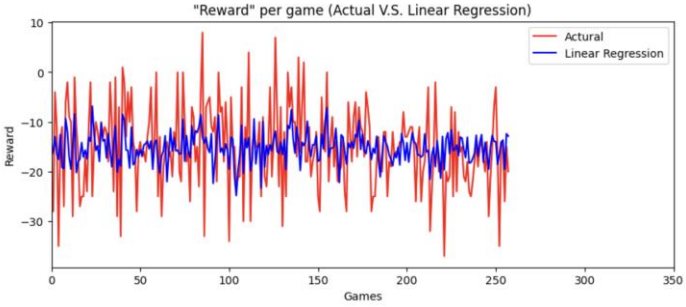


Figure 7 and Figure 8. Visualizations of two methods.

VIII Conclusion / Discussions

In conclusion, this project successfully tackled the dual objectives of predicting score differences and forecasting the reward system in baseball performance. The MLP Regressor and Linear Regression models emerged as top performers for predicting score differences and reward scores, respectively.

Our comprehensive analysis revealed the contrasting performance of the two systems, with the ‘reward’ system showcasing greater stability and reliability compared to predicting score differences. This insight not only contributes to a deeper understanding of player performance but also holds practical implications for strategic decision-making in baseball.

Initially, we focused on comprehensive pitching performance in each game, calculating average values for various features while addressing outliers to ensure data accuracy. However, we recognized that the outcome of a game is not solely determined by the pitcher. Factors like batter and fielder performance, as well as the impact of a proficient catcher, play pivotal roles. This interconnectedness emphasizes the need for a holistic approach in evaluating individual player contributions.

Furthermore, we considered the non-independence of each pitched ball, acknowledging the gradual decline in a player’s physical strength throughout a game. This factor underscores the importance of accounting for evolving player conditions, especially in the latter stages of a game. It's evident that relying solely on pitcher-specific datasets may not suffice for accurate predictions, highlighting the collective nature of team efforts.

In the context of predicting "score difference," normalization of outliers may lead to less optimal outcomes due to the centralized dataset. On the contrary, the "reward" system, with its comprehensive consideration of multiple factors, shows promise for more accurate predictions. To enhance predictive accuracy further, future work could explore incorporating additional parameters like batter and pitcher performance or strategic decisions made by the coach. This holistic approach holds potential for refining and improving the precision of predictions within the "reward" system.

References

[1] Using linear regression to predict a pitcher's performance. [Link]
[2] Predicting pitcher performance. [Link]
[3] Predicting 2015 starting pitcher performance using regression trees. [Link]
[4] A Method of Analyzing a Baseball Pitcher's Performance Based on Statistical Data Mining. [Link]

Appendix

[1] Features we use to do the prediction

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 225456 entries, 0 to 225455
Data columns (total 38 columns):
 #   Column             Non-Null Count   Dtype
---  ------             --------------   -----
 0   game_date          225456 non-null  datetime64[ns]
 1   pitcher            225456 non-null  object
 2   home_team          225456 non-null  object
 3   away_team          225456 non-null  object
 4   post_home_score    225456 non-null  int64
 5   post_away_score    225456 non-null  int64
 6   player_name        225456 non-null  object
 7   release_speed      225456 non-null  float64
 8   release_pos_x      225456 non-null  float64
 9   release_pos_z      225456 non-null  float64
 10  zone               225456 non-null  float64
 11  pfx_x              225456 non-null  float64
 12  pfx_z              225456 non-null  float64
 13  plate_x            225456 non-null  float64
 14  plate_z            225456 non-null  float64
 15  vx0                225456 non-null  float64
 16  vy0                225456 non-null  float64
 17  vz0                225456 non-null  float64
 18  ax                 225456 non-null  float64
 19  ay                 225456 non-null  float64
 20  az                 225456 non-null  float64
 21  effective_speed    225456 non-null  float64
 22  release_spin_rate  225456 non-null  float64
 23  release_extension  225456 non-null  float64
 24  release_pos_y      225456 non-null  float64
 25  reward             225456 non-null  int64
 26  pitch_SI           225456 non-null  int32
 27  pitch_CU           225456 non-null  int32
 28  pitch_FC           225456 non-null  int32
 29  pitch_FF           225456 non-null  int32
 30  pitch_CH           225456 non-null  int32
 31  pitch_CS           225456 non-null  int32
 32  pitch_SL           225456 non-null  int32
 33  pitch_ST           225456 non-null  int32
 34  pitch_FS           225456 non-null  int32
 35  pitch_KC           225456 non-null  int32
 36  pitch_PO           225456 non-null  int32
 37  pitch_SV           225456 non-null  int32
dtypes: datetime64[ns](1), float64(18), int32(12), int64(3), object(4)
memory usage: 55.0+ MB
```

[2] The reward table

events	description	reward
single	hit_into_play	-1
	foul	0
	blocked_ball	0
	ball	-1
	called_strike	0
field_out	hit_into_play	1
strikeout	swinging_strike	1
double	hit_into_play	-2
strikeout	swinging_strike_blocked	1
	swinging_strike	1
walk	blocked_ball	-1
sac_fly	hit_into_play	-1
strikeout	called_strike	1
home_run	hit_into_play	-4
	swinging_strike_blocked	1
walk	ball	-1
hit_by_pitch	hit_by_pitch	0
grounded_into_double_play	hit_into_play	1
force_out	hit_into_play	1
strikeout_double_play	swinging_strike	1
	foul_tip	0
double_play	hit_into_play	-2
strikeout	foul_tip	1
field_error	hit_into_play	0
caught_stealing_2b	called_strike	1
	foul_bunt	0
triple	hit_into_play	-3
sac_fly_double_play	hit_into_play	0
caught_stealing_3b	ball	1
sac_bunt	hit_into_play	0
fielders_choice	hit_into_play	0
	missed_bunt	0
caught_stealing_2b	swinging_strike	1
other_out	ball	0
caught_stealing_2b	ball	1
strikeout_double_play	called_strike	1
fielders_choice_out	hit_into_play	0
caught_stealing_home	called_strike	1
catcher_interf	hit_into_play	1
strikeout	missed_bunt	1
strikeout	foul_bunt	1
other_out	blocked_ball	0
catcher_interf	foul	1
pickoff_3b	ball	1
caught_stealing_2b	blocked_ball	1
caught_stealing_home	ball	1
	bunt_foul_tip	0
pickoff_1b	swinging_strike	1
	pitchout	0
strikeout_double_play	swinging_strike_blocked	1
catcher_interf	ball	1
pickoff_1b	ball	1
catcher_interf	swinging_strike	1
triple_play	hit_into_play	-3
pickoff_1b	blocked_ball	1
pickoff_2b	swinging_strike	1
other_out	swinging_strike	1
strikeout_double_play	foul_tip	1
pickoff_1b	called_strike	1

[3] Hyperparameter Grids for Model Tuning

1. LASSO Regression:
- Alpha values: [0.01, 0.1, 1.0, 10.0]
- Max iterations: [100, 500, 1000]

2. Support Vector Regressor (SVR):
- Kernel options: ['linear', 'poly', 'rbf']
- C values: [0.1, 1, 10]
- Epsilon values: [0.01, 0.1, 1]

3. Random Forest Regressor:
- Number of estimators: [10, 50, 100]
- Maximum depth: [None, 10, 20]
- Minimum samples split: [2, 5, 10]
- Minimum samples leaf: [1, 2, 4]

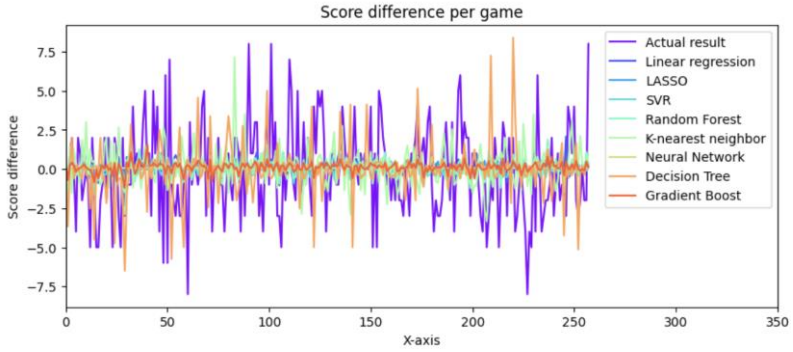
4. K-Nearest Neighbors Regressor (KNNR):
- Number of neighbors: [3, 5, 7]
- Weight options: ['uniform', 'distance']
- P values: [1, 2]

5. Multi-Layer Perceptron (MLP) Regressor:
- Hidden layer sizes: [(50,), (100,), (50, 50)]
- Activation functions: ['relu', 'tanh']
- Alpha values: [0.0001, 0.001, 0.01]
- Learning rate options: ['constant', 'inscaling', 'adaptive']

6. Decision Tree Regressor:
- Maximum depth: [None, 10, 20, 30]
- Minimum samples split: [2, 5, 10]
- Minimum samples leaf: [1, 2, 4]

7. Gradient Boosting Regressor (GBR):
- Number of estimators: [50, 100, 200]
- Learning rates: [0.01, 0.1, 0.2]
- Maximum depth: [3, 4, 5]
- Minimum samples split: [2, 5, 10]
- Minimum samples leaf: [1, 2, 4]
- Subsample values: [0.8, 1.0]

[4] All results from models we use for “score difference” comparing to the actuarial results



[5] All results from models we use for “reward” comparing to the actuarial results