Practicum Report - Baltimore Orioles

Roberto Arturo Morales Cruz

Project Scope

Objective:

The Baltimore Orioles Baseball Analytics department is interested in creating a simple formula that can estimate when an American League manager removes his starting pitcher from a game. This model will mainly benefit the members of the front office and the coaching staff.

The manager has limited communication, so ideally the model would be able to be run manually by him or any member of the staff; to further emphasize that, the team has been using the codename of “clipboard model” .

Hypothesis

The likelihood that a generic American League manager would have removed his pitcher serves as a useful metric of how concerned the Baltimore Orioles Manager should be.

Problem description:

For each baseball game played, each team has an “starting” pitcher. He is expected to play most of it, giving the best possible odds of winning to his team before giving his place to another pitcher.

The decision of removal is based on a variety of factors, such as:

- The pitcher’s performance
- His fatigue level
- The upcoming hitters he is scheduled to face

The purpose of the project is to model the manager’s decision process for this situation.

Ideally, this will be a simple counting system in which pre-defined points accumulate when certain events happen during a pitcher’s start, like him throwing a pitch, giving up a hit, a walk or an inning concluding.

Typically, managers pull a pitcher when they have started to perform poorly. However, the team thinks that managers try to get ahead of this performance drop and as a result the pitcher being removed typically represents the manager's best estimate of this change point.

There is no preference for reducing either the false positives or false negatives. The tradeoff would be very dependent on the game state and the availability of other members of the bullpen. For instance, if the team has a one run lead and the whole bullpen available, they would be much more aggressive in pulling the starter than if having had a large lead and the bullpen threw a bunch of innings the previous day.

Constraints

There will be some cases in which a starting pitcher struggles immediately in the game and is removed in the first or second inning because of poor performance. Since the decision to remove the pitcher was not due to fatigue or the hitters he is scheduled to face next, these situations will need to be accounted for.

There are also "openers" (pitchers that only throw for the very first plate appearances). MLB teams used them last year at a higher rate than they did in the past. Accounting for openers is important since the manager's logic behind his removal is much different than the logic to remove a traditional starter.

The dataset

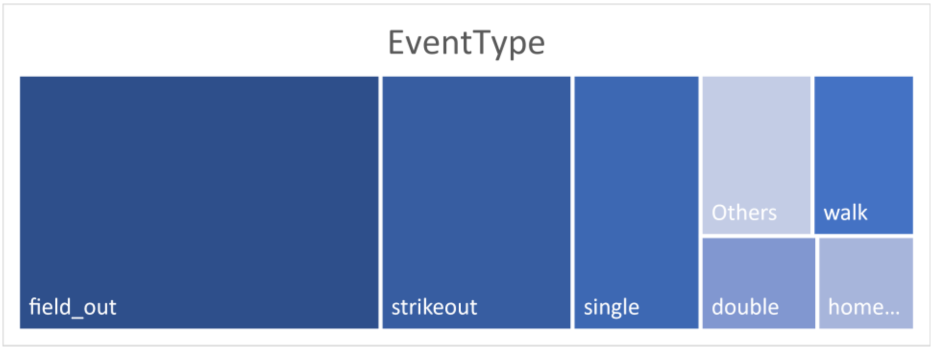
Baltimore Orioles Baseball Analytics department provided a dataset which contains play-by-play events, as well as information about the pitcher and the current batter.

Descriptive Analysis

The dataset includes records for 49521 plate appearances, with 30 variables each. These include ids for the game, the starter, game, and plate appearance. In addition, *last_batter* tells whether it is the last pitch thrown by the starter or not. That variable is going to serve as our target in the classification problem.

There are 256 pitchers, which totaled 2177 different starts. In other words, 4.4% of the plate appearances (2177/49521) ended up being the last batter for the pitcher.

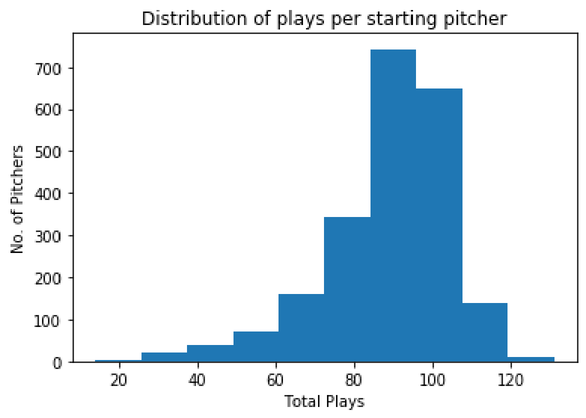
The distribution of the results from the plays is as follows:



Graph 1. The resulting *EventType* from each play on the dataset.

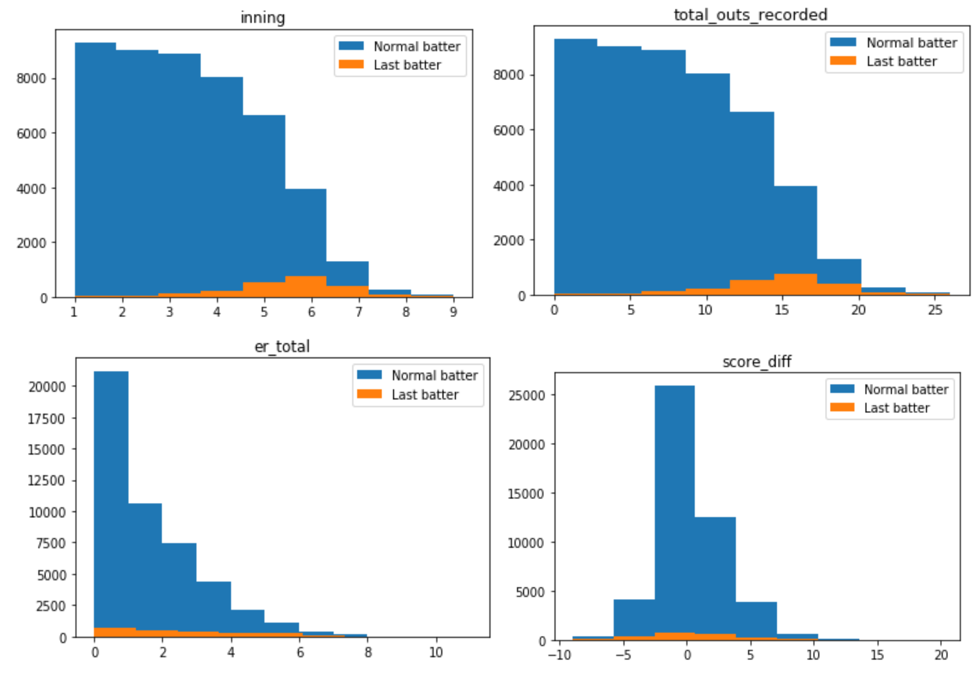
Two variables, *top_bot* and *home_away* have perfect collinearity. That makes total sense since the home team pitches on the top of the inning.

The number of pitches thrown by starter per game lies between 14 and 131, with a mean of 89. The distribution is as follows:



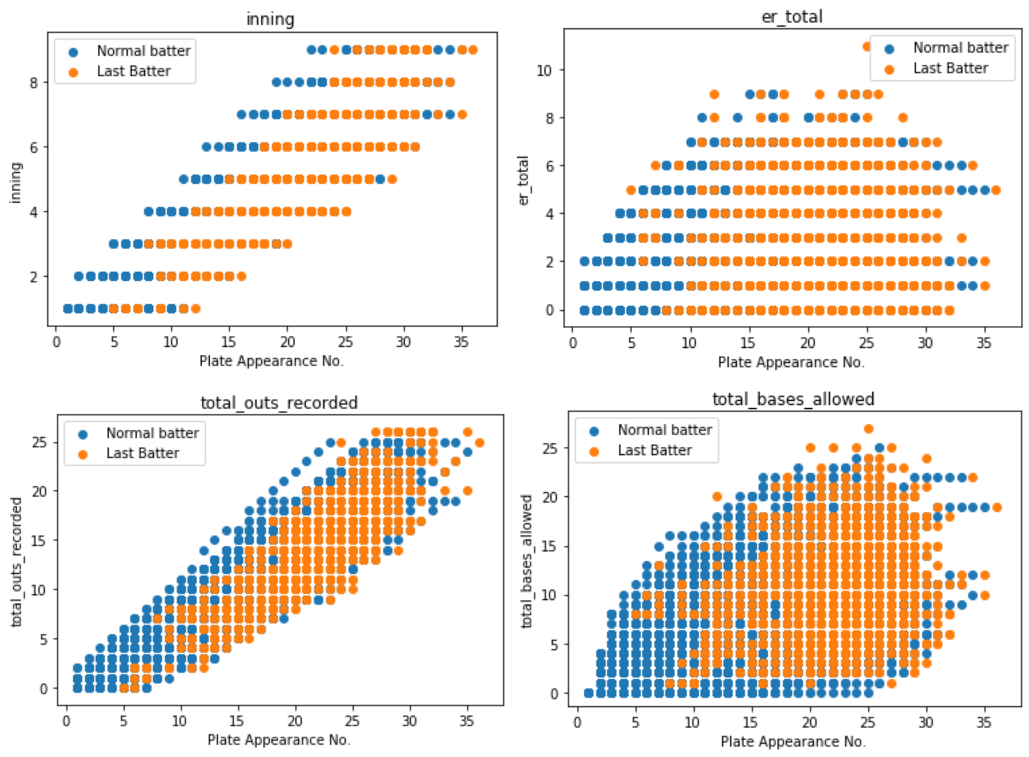
Graph 2. How many pitches did every starter threw?

Some of the most notable distribution of all numerical and categorical variables is are shown. In orange, the characteristics of the play that ended up being the last one for the pitcher. In blue, all other pitches.



Graph 3. Distribution of notable variables.

However, it is important to remark that the observations are not independent from each other. That is because they correspond to a row per play. Therefore, most of the numerical variables (for instance, *score_diff*) will have the same value for several consecutive plays until there is a change. This causes a direct relationship: with more plate appearances, more an increase on cumulative statistics. It will become more evident when plotting numerical variables against the number of plate appearance.



Graph 4. Notable variables vs Number of Plate Appearance

Only 57% of the end of pitchers were changed at the inning that they were playing. Therefore, it may be interesting to develop new variables which are the cumulative not of the whole game, but also on the specific inning. This may be influential since pitchers have a rest between innings; therefore, the performance may slightly reset.

Data Wrangling

Feature Engineering

The original dataset contained 30 variables, organized as follows:

- 4 id variables
- 15 categorical variables
- 10 numerical variables
- 1 Target variable

From these, derived features will be created. However, it is important to note that features will be simple enough to be calculated on paper, as required for the clipboard model. This include:

An indicator variable for the most common values of *EventType*. That is, the most common results from the plays: field_out, strikeout, single, walk, double or home_run.

Cumulative parameters, of both the indicator and numerical variables. For each one, these two variables were created:

- *Cum_<var>*, which is the total <var> obtained during all the game, until the moment of the play.
- *Inning_cum_<var>*, which is the total <var> obtained during the current inning.

For instance, *cum_strikeout* is the total number of strikeouts obtained until that point of the game, whereas *inning_cum_strikeouts* is the number of singles gained on that inning.

Plays between variables, of indicator variables. The hypothesis here is that certain events will occur more or less frequently when the player performance gets worse. For each one, these two variables were created:

- *Pas_since_<var>*, which is the number of plate appearances since <var> happened last.
- *has_have_<var>*, which tells if a <var> has been obtained at that point of the game.

For instance, *pas_since_walk* is how many plate appearances have passed since the last walk allowed, whereas *has_have_walk* becomes 1 after the first walk is allowed.

*Hot_cold_* variables, which are an indicator of the previous result obtained by the next batter.

There are 99 variables at the end of the feature engineering.

Missing values treatment

As the *pas_since_ and hot_cold* parameters depend on previous results, they will have missing values at the beginning for each starter. Therefore, a value of 0 will be inputted. After all, *hot_cold* are indicator features anyway.

On the other side, the missing values for *next_batter_hand* were asked to the team's Analytics Department, and they were manually inputted.

Outliers treatment

The candidate values for outliers were determined using the following rule: each value that was at least three standard deviations away from the mean was to be separately analyzed. This rule resulted in 2287 candidate values: 1.19% of the database. They mostly appear on the following variables:

Feature	Large Values
tying_run_on	195
walk	178
hot_cold_walk	159
double	127
points_allowed	99
hot_cold_double	95
hot_cold_home_run	79
home_run	78

Table 1. Variables with more candidates to Outliers.

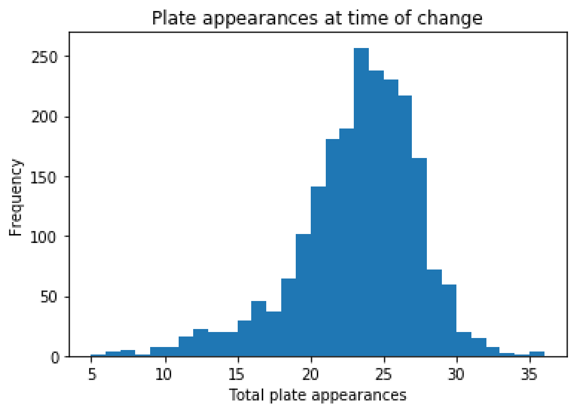
However, a deeper analysis reveals that they are not true outliers; they are either boolean variables, of which one of the two values is predominant over the other, or a derived variable from of those. Thus, they will not be classified as outliers. This was the case for all the variables, except on three cases:

PostVisTeamScore	PostHomeTeamScore	score_diff
20	21	20
17	15	15
16	14	15
16	14	14
15	12	14
15	12	13
15	11	13
15	11	12

Table 2. Extreme values from scoring variables.

As seen, the three variables related to the score have outliers on one start each. In addition, the largest *PostHomeTeamScore* and *score_diff* happened on the same game. Thus, being particularly good games, the values will be substituted with a cap: one score more than the second largest. Thus, the best scoring games will still be on the top, but not too large so that it will not have a large effect on the model.

Finally, we will attempt to eliminate both opener pitchers and starters whose abysmal performance led to a soon exit. We will be looking for an inflection point on the number of plate appearances.



Graph 5. Number of plate appearances at the time the pitcher was pulled

10 may be the ideal plate appearance for the cutoff. However, a deeper examination revealed that most of them were on the first inning. The score difference for the pitchers that were pulled at the first inning was always negative; they were changed because of really bad performance. Thus, pitchers that lasted one inning will be eliminated.

<u>Modelling</u>

Data split

The remaining dataset will be split into a training and a validation/test dataset. This was done randomly, with the 80% of the data into the first and the remaining 20% unto the second.

The training set was further down sampled, as the target was unbalanced. The final training sample included 60% of "normal" plays, and 40% of plays where last_batter=1.

The down sample also helped to reduce the bias, as previously there was a high correlation between the observations from the same start. By not including all the plays, the chances of consecutive plays being used for modeling was reduced; therefore, less similar observations.

Metrics

To measure the effectiveness of the model, the following metrics will be used:

- Accuracy: how many of the test plays are correctly predicted?
- Precision: how many of the predicted changes were actual changes?
- ROC AUC: how capable is the classifier of distinguishing between classes?
- F1 score: how balanced is the tradeoff between false positives and false negatives?

Algorithm selection

To accomplish the requirement that the model should be simple enough to be scored on paper, the following algorithms were chosen:

- Decision Tree
- Logistic Regression
- Support Vector Machine – Linear Kernel

They were chosen due to the simplicity of their decision boundaries. They may be estimated manually, if keeping only a few variables to work with.

Preliminary models were fitted and tested on our defined datasets. This was done on both R and Python, testing different parameters, and using all the features. The “best” looking results on the testing dataset are as follows:

	Decision Tree	**Logistic R**	**SVM**
Accuracy	91.01%	89.13%	88.91%
Precision	29.31%	26.93%	26.62%
AUC	83.93%	88.63%	88.85%
F1 score	42.34%	41.25%	40.97%

Table 3. Results from preliminary models

The three algorithms performed similar. In particular, the SVM and the Logistic Regression had a similar prediction on the testing set. That was expected, as they both aim to generate the best “hyperplane” that separates the two classes. However, the logistic regression runs noticeably faster, and has a more concise formula to determine the decision boundary (more on it later). Therefore, it will be chosen over the SVM with a linear kernel.

On the other hand, the Decision Tree performed slightly better on most of the metrics while using less variables. That is because the maximum depth was set at four. Thus, the Decision Tree will be used as a base model, while we attempt to have a better model by using a Logistic Regression with few variables.

Multicollinearity Analysis

The first variable reduction will be done by analyzing variables with high collinearity. Multicollinearity exists whenever one of our predictors is highly correlated with one or more of the other independent variables. That is, similar information may be given by two or more features. It may become a problem because it undermines the statistical significance of an independent variable.

Variance Inflation Factor (VIF) will be given as a measure of multicollinearity. To calculate it, a linear regression model is fitted on every independent variable using the other features as predictors. For each fitted regression i, R_i^2 is computed. Therefore:

$$VIF_i = \frac{1}{1 - {R_i}^2}$$

Variables with high VIF are not desired; as a rule of thumb, every VIF higher than 6 should be analyzed.

Chosen variables were eliminated iteratively. At each step, VIF is computed for all features and one or more of the variables with the highest VIF is carefully removed. It is important to supervise it, since we may accidentally remove the two (or more) variables that give us the same information, or we may prefer one over the other.

For instance, *post_runner_on_first, post_runner_on_second*, *post_runner_on_third* and *runners_on_base* had infinite VIF on the first step. The latter is a combination of the first three. To work with less variables, *runners_on_base* was chosen.

In the end, the number of predictors went down to 68 from the original 98.

Variable selection

To reduce the number of variables and possibly improve the performance, three approaches were done:

1. Using Lasso (L1) regularization.
2. Using Recursive Feature Elimination (RFE)
3. Estimating the Information Value (IV) for each variable.

Given the simplicity requirement for his project, only models using a maximum of 6 variables are given. In addition, the base model uses only four features and already showed a good performance. The final model to be selected must combine simplicity and performance.

Three models with few variables were generated. Their performance was as follows:

	Lasso	**RFE**	**IV**
Accuracy	83.98%	81.90%	84.80%
Precision	16.66%	13.54%	19.71%
AUC	76.27%	71.15%	83.54%
F1 score	26.74%	22.05%	31.80%

Table 5. Different approaches for feature selection in Logistic Regression

As seen, after testing their performance on the validation set, they were not as effective as the decision tree. Therefore, we will continue to look for a better model.

The next approach was an exhaustive feature selection. Models will be fitted using the different variables combinations (using, at most, 6 variables). The most important metrics will be F1_score, as well as AIC and BIC to prefer simpler models. 5-fold Cross validation will be used to get the best coefficients.

However, it is not computationally efficient to try every different combination. By only keeping 6 of the 69 variables, for instance, there are 109,453,344 different models to be fitted. Therefore, this will be done using only the important features; those given as the principal variables for the previous three approaches. In total, 15 features were carefully selected.

The performance of the best Logistic Regression is:

```
Accuracy: 87.95%
Precision: 24.10%
AUC: 85.85%
F1 score: 37.41%
```

Which is great, considering the only the use of six variables: *pbp_idx, cum_points_allowed, cum_walk, end_of_inning, field_out* and *strikeout*. However, the base model is still better.

The Decision Tree model indicated a cutoff at 19.5 or 18.5 (depending on the seed used) plate appearances per pitcher as the initial branch. Could it mean that it is a good idea separating the plays into two groups?

The training will be repeated. This time, the model will be trained using only the plays after the 18th. This number was chosen as it makes more sense: there are 9 players at the batting order, and differences may show between “normal” plays and “last” plays after the pitcher faces the starting batter for a third time.

These are the results from that Logistic Regression:

		Predicted		
		0	**1**	**Total**
True	**0**	9202	283	9485
	1	171	224	395
	Total	9373	507	9880

```
Accuracy: 95.40%
Precision: 44.18%
AUC: 76.86%
F1 score: 49.66%
```

By focusing on the plays after the 18th plate appearance, the model captured better the differences. Even though the model had a small drop on the AUC score, it performed better on the three other metrics.

More complex models were fitted; this was simpler and had a highly similar performance (52% vs 50% F1 score, by the cost of adding two more variables).

There is a better tradeoff between False Positives and False Negatives; therefore, this model will be selected. In addition, the z value indicates that it is highly possible that each variable used adds predictive value.

Given the use of 0.5 as threshold for making the decision, the boundary is given as:

$$\ln\left(\frac{P}{1-P}\right) = \ln\left(\frac{0.5}{1-0.5}\right) = \ln(1) = 0$$
$$= 0.1022\, pitch\_total - 0.1362\, cum\_strikeout + 1.7942\, end\_of\_inning - 9.3939$$

=>

$$9.3939 = 0.1022\, pitch\_total - 0.1362\, cum\_strikeout \ + 1.7942\, end\_of\_inning$$

=>

$$10 = 0.1022\, pitch\_total - 0.1362\, cum\_strikeout \ + 1.7942\, end\_of\_inning + 0.6061$$

Which is approximately equal to:

$$\underline{\boldsymbol{1000 = 10\, pitch\ total - 13\, cum\ strikeout\ + 180\, end\ of\ inning + 60}}$$

Therefore:

- Each pitch thrown adds 10 points
- Each out by strike subtracts 13 points
- If it is the end of inning, 180 points will be added.
- 60 points added, nevertheless.

When this sum is 1000, half of the pitchers will be changed. The higher the sum, the higher probability of being changed and vice versa.