Pitching substitutions in baseball

Spencer Lin (sl146)

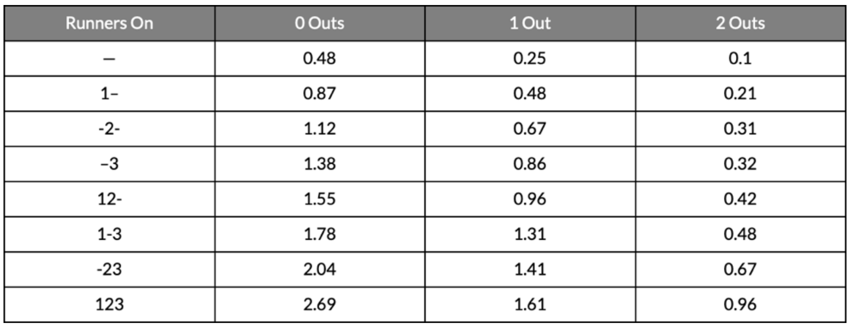
STAT 432 - Basics of Statistical Learning

December 11, 2025

Project Description and Abstract

In baseball, once a player is substituted out, they cannot return to the game. Therefore, the timing of making substitutions—especially pitching changes—is critical. Pitchers are the core of a baseball team's defensive strategy by controlling the pace and outcome of a game, and once a pitcher becomes fatigued, their performance declines, making it easier for the opposing team to score. Normally, a fatigued pitcher may experience reduced velocity, poorer command, and diminished ability to induce weak contact. Coaches must therefore decide whether it is optimal to change the pitcher or not.

The main goal of the project is to develop a model that is able to predict an ideal timing to substitute a pitcher. I use a metric called run value, which measures how much a single play increases or decreases the expected number of runs the opposing team will score. Run value is calculated by subtracting the run expectancy after the play and the run expectancy before the play plus the runs scored during this at bat. And run expectancy is the average number of expected runs per inning given the current number of outs and placement of baserunners.



Runners On	0 Outs	1 Out	2 Outs
—	0.48	0.25	0.1
1–	0.87	0.48	0.21
-2-	1.12	0.67	0.31
–3	1.38	0.86	0.32
12-	1.55	0.96	0.42
1-3	1.78	1.31	0.48
-23	2.04	1.41	0.67
123	2.69	1.61	0.96

Figure 1. Run Expectancy Matrix for the 2025 MLB Season from Fangraphs

The predictors used in the model are 1. The number of pitches used this game before the play. 2. Fastball quality (average fastball velocity this at bat - average fastball velocity whole game). 3. Fastball spin rate quality, defined as the difference between the pitcher's average fastball velocity in the current at-bat and their game-wide average. 4. Rolling pitcher effectiveness for the past 1, 2, 3 at-bats. (Numerical variable to evaluate pitchers dynamically, based on the results of in-game outcomes after each pitch).

By using a random forests model, it is possible to predict the run values of each play. And using the prediction values to find an adequate threshold to display a warning to see if runs are actually prevented compared to the original scenarios. The tuning of the model results in a mtry value of 6, and a ntree value of 300. Where the train R-squared being 0.53554, and the test R-squared being 0.301099. Moreover, when tuning the appropriate threshold for the warning to trigger, the best tradeoff value among a decent number of innings triggered, the accuracy, and the average runs scored after the warning is at 0.7

When comparing innings with model-generated warnings to innings without warnings, the results show substantial differences in run prevention. Innings flagged by the model averaged 0.515 runs, compared with 0.148 runs in innings with no warnings. In contrast, traditional substitution indicators, such as the presence of a runner in scoring position or a runner in scoring position with fewer than two outs, showed higher average runs allowed (0.669 and 0.807, respectively) in their warning conditions, and less separation between warning and non-warning innings. These findings suggest that the model-based approach provides a more sensitive and informative signal of impending run risk than conventional heuristics, offering the potential to guide more effective pitcher management strategies.

I used AI in this project to support me editing the code as well as creating summary plots and tables to enhance readability of the data. I also used AI in the document to check for grammatical errors to make the overall flow better.

Baseball terminologies used in this paper:
At bat (AB): A batter's turn batting against a pitcher.
Scoring position: When there a runner is on second or third base. The distinction between being on first base and scoring position is that a runner is easier to score when in scoring position.
Spin Rate: How much spin, in revolutions per minute, a pitch was thrown with.

https://blogs.fangraphs.com/the-run-expectancy-matrix-reloaded-for-the-2020s/

Literature Review

1. Watkins, C. (2021). Pitcher effectiveness: A step forward for in game analytics ... https://digitalcommons.chapman.edu/cgi/viewcontent.cgi?article=1951&context=scs_articles

The article, "Pitcher Effectiveness: A Step Forward for In Game Analytics and Pitcher

Evaluation," proposes a new metric, Expected Pitcher Performance (EPP), to improve the evaluation and in-game analysis of baseball pitchers. In the article, A Random Forest Classifier was used to predict the immediate outcome of a pitch, classifying it as a strikeout, walk, hit, or an out. Moreover, A Random Forest Regressor was trained to predict the expected change in a starting pitcher's Game Score (a commonly used performance metric) based on the pitch outcome. The final EPP metric combines the predictions from both models: the predicted probability of a positive outcome (strikeout or out) and the expected change in Game Score. The study's results demonstrated that the EPP metric is a powerful tool for pitcher evaluation, surpassing the performance of traditional metrics. For example, the classifier achieved an accuracy of 84.78% in correctly predicting a pitch's final outcome. In the feature importance analysis revealed that the most critical factors in determining a pitcher's effectiveness are pitch type, pitch velocity, and pitch location.

2. Ganeshapillai, G., & Guttag, J. (n.d.). A Data-driven Method for In-game Decision Making in MLB When to Pull a Starting Pitcher. https://chbrown.github.io/kdd-2013-usb/kdd/p973.pdf

In this article, the researchers developed a data-driven model using regularized linear regression and multi-task learning to help Major League Baseball (MLB) managers decide when to replace a starting pitcher. The primary goal was to create a pitcher-specific predictor for the expected Pitcher's Total Bases (PTB) in the next inning. The regression output was then binarized to classify whether a run would be given up in the next inning. One of the keys to this approach was converting categorical variables (like batter and venue) into continuous priors which were then subjected to shrinkage when data support was low. The main finding is that the model developed by the researchers significantly outperformed a baseline "manager model" designed to mimic actual manager decision-making, where the researchers' model correctly predicted whether a run would be scored in 81% of the innings, compared to 75% for the manager model. For instance, if a manager had left a pitcher in a close game after the fourth inning but the model would have removed him, the pitcher would have given up a run 60% of the time. This is notable since runs are scored in only about 10% of innings overall.

These two papers give me a rough outline of what the model should contain. Pitcher effectiveness will be implemented into the final model and Random Forests is one of the regression models being used.

Data Processing and Summary Statistics

In the beginning of cleaning the data, only starting pitchers who pitched more than five innings were considered. The main reason of this choice is that starting pitchers this year pitches at an average of 5.35 innings, and these pitchers will have more data to train the model. First, dplyr functions like filter and group_by are used to create a data set to find out all the pitchers that started the games, another data set is then created to find the pitcher that ends the fifth inning. By matching these two datasets, it was possible to determine all pitchers who pitched at least five innings. There are three columns "on_1b", "on_2b", "on_3b" that originally contains a player's ID if he in on the base and NA otherwise. In order to make these columns useable, all the NA values are converted into 0 and the rest turns into 1.

Additional columns were then generated to represent the scenarios needed for calculating pitcher effectiveness. After having all the necessary columns, a column of pitcher effectiveness can be created. The lag function was then applied to compute rolling pitcher effectiveness scores for the previous one, two, or three at-bats. Rolling statistics are set to 0 for the first couple at bats that does not have a sufficient data yet.

Event	**Weight**
Single	-0.88
Double	-1.25
Triple	-1.58
Home Run	-2.03
Walk	-0.69
Hit by pitch	-0.72
Swinging Strike	+0.5
Out (except sacrifice fly)	+0.5
Other Events (i.e. Foul ball, error, etc.)	+0
Count	
Ahead of count	+0.5
Behind count	-0.5
Exit Velocity	
95+ mph exit velocity	-0.5
80 mph or less exit velocity	+0.5

Figure 2. Summary Statistics for Pitcher Effectiveness per Game

For pitch characteristics such as mean_ff (the average fastball velocity in each at-bat), mean_ff_spin (the average fastball spin rate in each at-bat), and mean_release_point (the average release height of the pitcher), missing values were handled using a forward-fill procedure from the zoo package. These variables contain NA values in situations where no fastballs were thrown during a given inning, so forward-filling ensures that each at-bat retains the most recent available information for these characteristics.

Before applying clustering algorithms, we standardized all numeric features using z-score normalization. This step is critical because our variables span vastly different scales—pitch_count ranges from 1 to 150+, while run_value ranges approximately from -1 to 1. Distance-based clustering algorithms like K-means are sensitive to scale, and without standardization, variables with larger magnitudes would dominate the distance calculations.

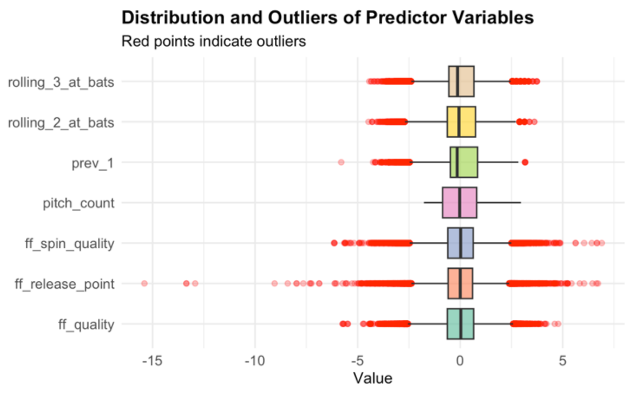
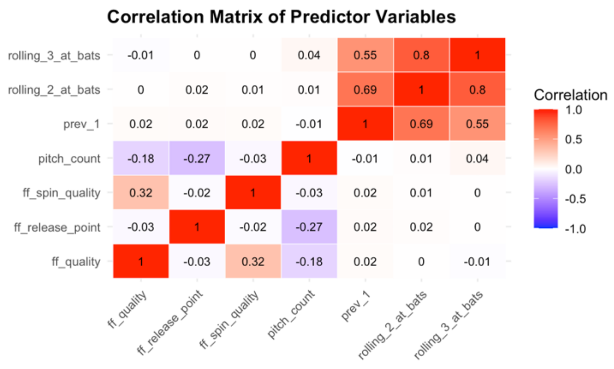


Figure 3. Distribution and Outliers of Predictor Variables

The plot above is a box plot for a few variables that I think is important of predicting the run value. From the plot, we can see that the spin quality of the fastballs spreads widely compare to the other variables. One of the reasons is that the recording of the spin rates in the games are still not fully stable. Another reason is that any subtle change in the grip from the pitcher or even a different baseball will end up having a distinct spin rate.



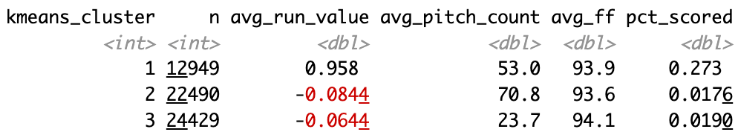
	ff_quality	ff_release_point	ff_spin_quality	pitch_count	prev_1	rolling_2_at_bats	rolling_3_at_bats
rolling_3_at_bats	-0.01	0	0	0.04	0.55	0.8	1
rolling_2_at_bats	0	0.02	0.01	0.01	0.69	1	0.8
prev_1	0.02	0.02	0.02	-0.01	1	0.69	0.55
pitch_count	-0.18	-0.27	-0.03	1	-0.01	0.01	0.04
ff_spin_quality	0.32	-0.02	1	-0.03	0.02	0.01	0
ff_release_point	-0.03	1	-0.02	-0.27	0.02	0.02	0
ff_quality	1	-0.03	0.32	-0.18	0.02	0	-0.01

Figure 4. Correlation Matrix of Predictor Variables

The figure above is the correlation matrix for a few variables that I think is important of predicting. As expected, the correlation for the three statistics about pitcher effectiveness are related to each other since they are all derived from overlapping sequences of the same pitch-by-pitch outcome data. In contrast, the remaining variables, such as pitch count, fastball quality, fastball spin-rate quality, and release-point variation, show relatively weak correlations with each other and with the effectiveness metrics.

Unsupervised Learning

1. K-means Clustering



```
kmeans_cluster     n avg_run_value avg_pitch_count avg_ff pct_scored
         <int> <int>         <dbl>           <dbl>  <dbl>      <dbl>
             1 12949         0.958            53.0   93.9     0.273
             2 22490       -0.0844            70.8   93.6     0.0176
             3 24429       -0.0644            23.7   94.1     0.0190
```

Figure 5. Summary Statistics by Cluster

The dataset contains no categorical variables relevant to clustering since it contains play by play data, so clustering was performed only on numerical features.

From figure 5, we can observe that the three clusters show very distinct pitcher performance and game-state profiles. The first cluster is showing when pitchers are performing poorly or are in trouble. Moderate pitch counts suggest they are still early–mid game, but the run value and scoring rate are higher than the other clusters, indicating hitters are succeeding in these at-bats. This could reflect innings where pitchers lose command early, or where the lineup is seeing the pitcher well.

The second cluster is showing pitchers in deep pitch count but still effective at finishing batters. Their velocity may be slightly slower, but the run value and scoring probabilities remain extremely low. This may represent low-leverage innings with few threats, despite the high pitch count.

The final cluster is showing pitcher dominating in early games. Low pitch counts and highest fastball velocity indicate early innings, where pitchers have full stamina. The low run value and low run scoring probability suggest early-game advantage for pitchers.

Figure 6 in the next shows the visualization of the three clusters. The three clusters form distinct groups in the feature space, indicating that the algorithm successfully detected meaningful structure in the data.

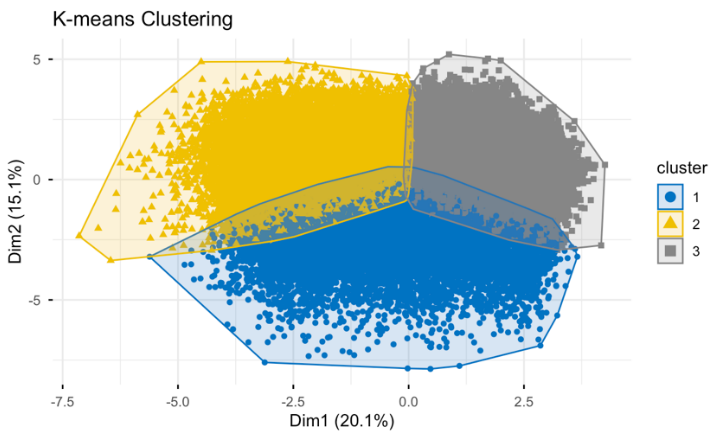
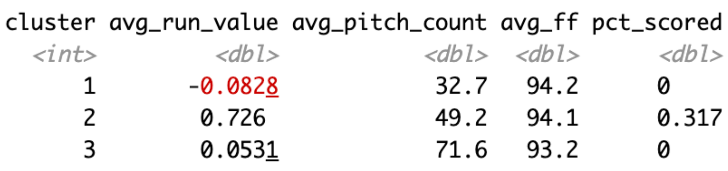


Figure 6. Visualization of K-means clusters

2. Hierarchical Clustering



```
cluster avg_run_value avg_pitch_count avg_ff pct_scored
  <int>         <dbl>           <dbl>  <dbl>      <dbl>
      1       -0.0828            32.7   94.2      0
      2        0.726             49.2   94.1      0.317
      3        0.0531            71.6   93.2      0
```

Figure 7. Summary Statistics by Hierarchical Clustering

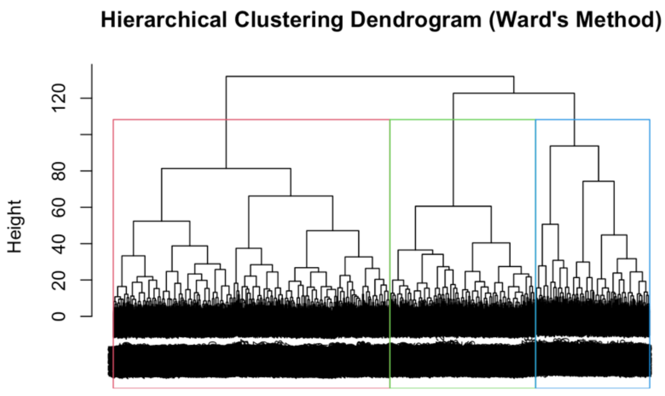


Figure 8. Hierarchical Clustering Dendrogram

From figure 7, it is observable that similar to the K-means clustering, the three clusters are sharing a similar outcome. Cluster 1 represents early game situations, characterized by negative average run value (–0.083), and relatively low pitch counts. Cluster 3 shows a similar low-risk profile, but with pitchers who have thrown substantially more pitches on average (71.6) and slightly lower fastball velocities, suggesting a late-game but still stable performance state. In contrast, Cluster 2 clearly captures high-risk situations. This cluster exhibits a substantially higher average run value (0.726), and mid-range pitch counts. These results indicate that hierarchical clustering successfully separates innings into interpretable performance states, with Cluster 2 identifying the conditions in which pitchers are most vulnerable to allowing runs.

3. Spectral Clustering

```
cluster     n avg_run_value avg_pitch_count avg_ff pct_scored
  <int> <int>         <dbl>           <dbl>  <dbl>      <dbl>
      1  1769      -0.00498            45.5   93.9      0
      2    88       1.41               48.0   93.7      1
      3   143       0.734              52.9   94.4      0.392
```

Figure 9. Summary Statistics by Spectral Clustering

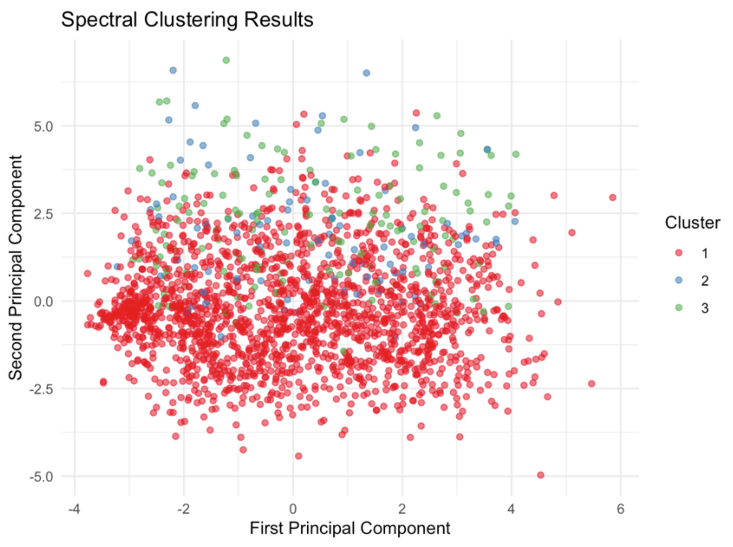


Figure 10. Spectral Clustering Results

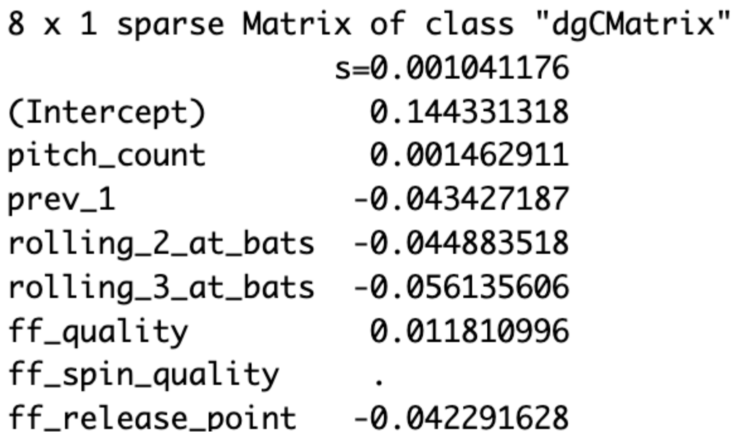
From the table and plot above, the results suggest that spectral clustering does not separate pitcher-performance patterns as cleanly as the previous clustering methods since the clusters are not distinct among each other. The table further provides proof this fact since the pitch count and fastball velocity are similar to each other. Although the average run value is very different from each other, the separation among clusters in terms of pitch count, and fastball velocity is weaker. Moreover, the size of the second and third clusters are smaller than the first cluster stating that the second and third cases are rare comparing to the first scenario.

Overall, although spectral clustering highlights a small collection of high-impact events, it does not provide well-separated or balanced clusters in the context of the pitcher-related variables considered. Therefore, it performs less effectively than the k-means and hierarchical clustering approaches for identifying interpretable clusters of pitcher performance.

Prediction Models

In this section, the five models selected are lasso regression, KNN regression, SVM regression, random forest regression, and gradient boosting(xgboost).

1. Lasso Regression



```
8 x 1 sparse Matrix of class "dgCMatrix"
                    s=0.001041176
(Intercept)           0.144331318
pitch_count           0.001462911
prev_1               -0.043427187
rolling_2_at_bats    -0.044883518
rolling_3_at_bats    -0.056135606
ff_quality            0.011810996
ff_spin_quality       .
ff_release_point     -0.042291628
```

Figure 11. Summary Statistics by Lasso Regression

The first algorithm applied is lasso regression, chosen over ridge regression because lasso performs variable selection in addition to regularization. Since some predictors were expected to have minimal contribution to the response, lasso is beneficial for shrinking uninformative coefficients exactly to zero, making a better model.

As shown in Figure 11, the coefficient for ff_spin_quality is reduced to zero (displayed as "." in the output), indicating that this variable does not meaningfully improve the model under the optimal penalty parameter. This confirms the initial expectation that certain features may not be predictive and supports the use of lasso for feature elimination.

2. KNN Regression

In KNN regression, a cross-validation is made to find an optimal k value for the regression. And as shown in Figure 12 in the next page, the lowest RMSE happens when k = 200. The full RMSE and R-squared values will be shown in the end of this section comparing with other models.

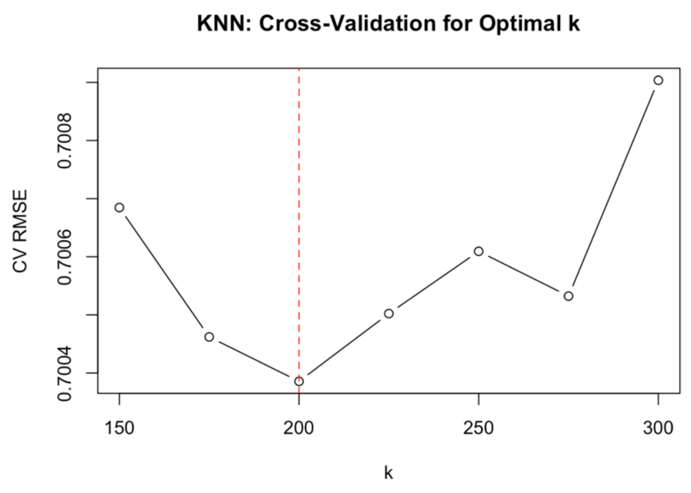
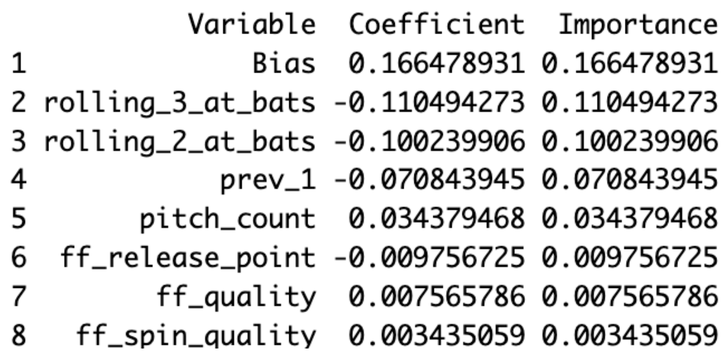


Figure 12. Cross Validation for KNN Regression

3. SVM Regression



```
           Variable  Coefficient  Importance
1              Bias  0.166478931 0.166478931
2 rolling_3_at_bats -0.110494273 0.110494273
3 rolling_2_at_bats -0.100239906 0.100239906
4            prev_1 -0.070843945 0.070843945
5       pitch_count  0.034379468 0.034379468
6  ff_release_point -0.009756725 0.009756725
7        ff_quality  0.007565786 0.007565786
8   ff_spin_quality  0.003435059 0.003435059
```

Figure 13. Summary Statistics by SVM Regression

Figure 13 is the importance of the variables from the SVM regression. We can see that the linear SVM model assigns the largest coefficients to the rolling at-bat variables and the previous at-bat indicator, suggesting that short-term pitcher performance has the strongest predictive power for current run value. Pitch-level characteristics such as fastball velocity, spin rate, and release point receive much smaller weights, indicating that they contribute marginally compared to recent outcomes. The bias term is moderate in magnitude and primarily serves to shift the model's baseline prediction rather than providing interpretable importance. The parameter tuned are cost = 10 and epsilon = 0.1.

4. Random Forests

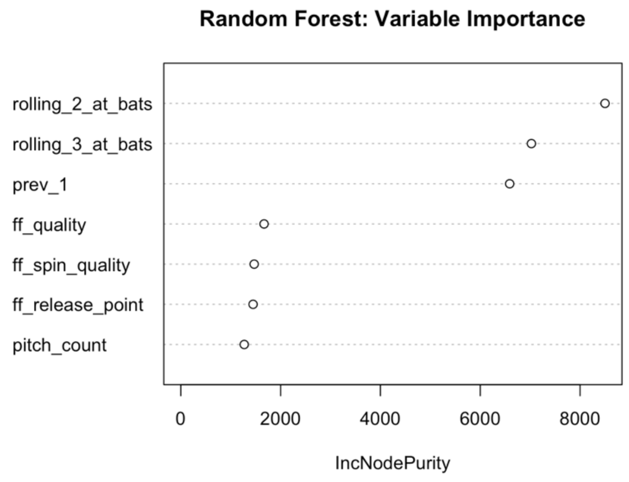
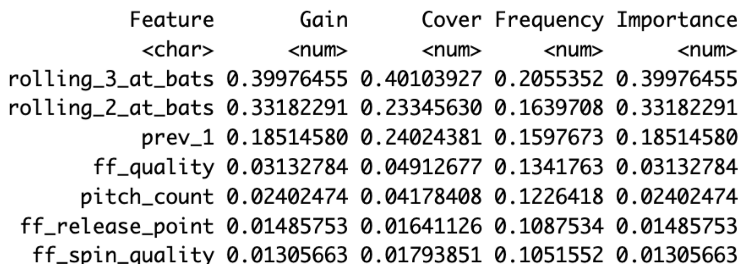


Figure 14. Graph of the Variable Importance of Random Forest

For the tuning of the regression model, the final parameters are ntree = 300, mtry = 3, and nodesize = 60. The training RMSE is 0.5881 and the training R^2 is 0.3483.

Similar to the previous regression method, the random forests model also has a high emphasis on the previous outcome of the at bats. However, the pitch count in the random forests model has the lowest importance whereas the SVM regression has it right after the pitcher effectiveness variables, indicating that its predictive influence is less stable across methods.

5. Gradient Boosting



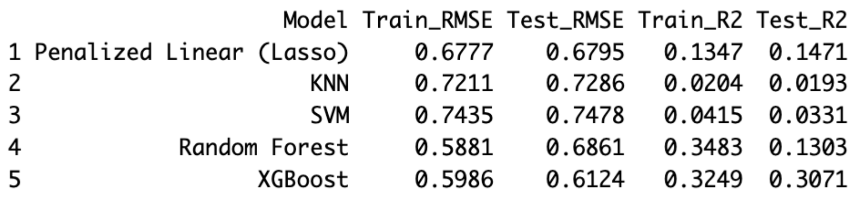
```
            Feature       Gain      Cover Frequency Importance
             <char>      <num>      <num>     <num>      <num>
  rolling_3_at_bats 0.39976455 0.40103927 0.2055352 0.39976455
  rolling_2_at_bats 0.33182291 0.23345630 0.1639708 0.33182291
             prev_1 0.18514580 0.24024381 0.1597673 0.18514580
         ff_quality 0.03132784 0.04912677 0.1341763 0.03132784
        pitch_count 0.02402474 0.04178408 0.1226418 0.02402474
   ff_release_point 0.01485753 0.01641126 0.1087534 0.01485753
    ff_spin_quality 0.01305663 0.01793851 0.1051552 0.01305663
```

Figure 15. Summary Statistics by Gradient Boosting

The last prediction model implemented is gradient boosting using the xgboost library. For the tuning for gradient boosting, the final values are eta = 0.01 and max_depth = 6 with training RMSE of 0.5986, and a training R^2 of 0.3249.

Compared to the Random Forests model, the Gradient Boosting model shows an even stronger reliance on recent batter performance. The top three features account for more than 90% of the total model gain, indicating that boosting methods concentrate importance more aggressively than ensembles of trees. Interestingly, while Random Forests assigned meaningful importance to pitcher-related variables such as pitch count, spin quality, and release point, the Gradient Boosting model places only minimal weight on these characteristics. This suggests that, after accounting for the batter's recent outcomes, additional information about pitch attributes contributes little to improving prediction accuracy in the boosting framework. Another notable difference is that Random Forests has a higher importance on the most recent at-bat, whereas Gradient Boosting places greater value on multi-at-bat rolling averages, implying that boosted decision trees extract slightly longer-term trends in hitter performance.

6. Performance Comparison



```
                      Model Train_RMSE Test_RMSE Train_R2 Test_R2
1 Penalized Linear (Lasso)     0.6777    0.6795   0.1347  0.1471
2                       KNN     0.7211    0.7286   0.0204  0.0193
3                       SVM     0.7435    0.7478   0.0415  0.0331
4             Random Forest     0.5881    0.6861   0.3483  0.1303
5                   XGBoost     0.5986    0.6124   0.3249  0.3071
```

Figure 16. Table of Model Performance Comparison

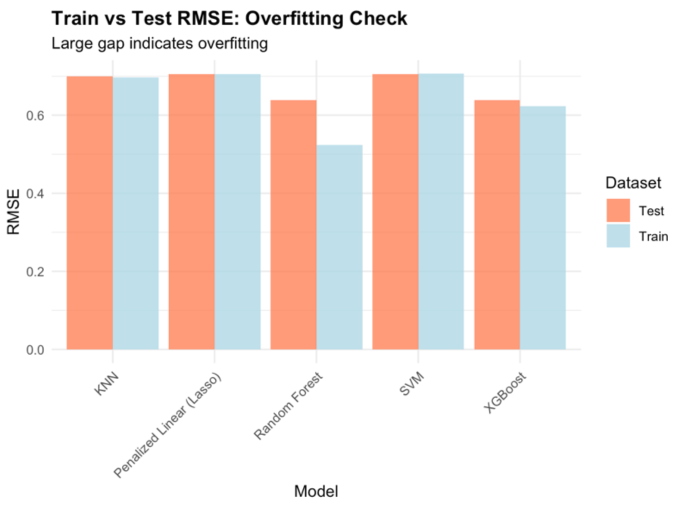
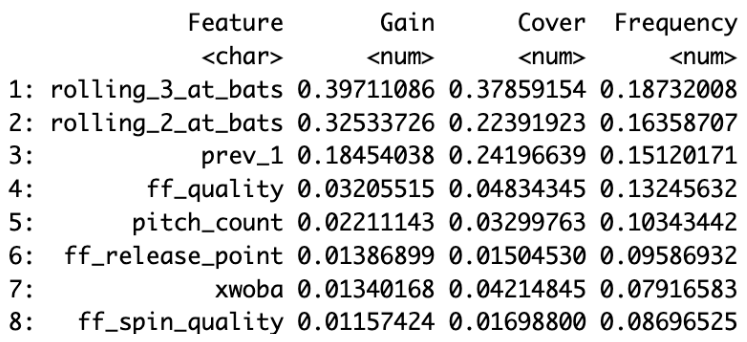


Figure 17. Plot of Train vs. Test RMSE

From this table above, in the training RMSE, Random Forest and XGBoost have lower values comparing to the rest of the three. Also, the two regression models have a higher training R^2 than the rest of the model meaning that these two models perform better than the rest. However, Random Forest exhibits a substantial drop in performance when evaluated on the test set (Train $R^2 = 0.3483$ vs. Test $R^2 = 0.1303$), suggesting significant overfitting. In contrast, XGBoost maintains similar performance between training and testing (Train $R^2 = 0.3249$ vs. Test $R^2 = 0.3071$), indicating good generalization.

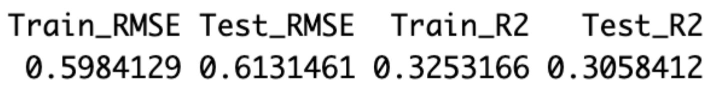
Feature Engineering Challenge

Since the current model does not consider the batters' performance at all, adding some statistics of the batter might be a good feature to add into the model. Therefore, a new variable xwOBA which it is statistics showing how good a batter performs is implemented. xwOBA combines quality of contact (EV, LA), strikeout likelihood, and walk rate into a single measure of batter talent. It is interpretable as the expected run value of each plate appearance. Thus, including xwOBA adds information about the batter's underlying skill independent of game randomness.



```
             Feature       Gain      Cover  Frequency
              <char>      <num>      <num>      <num>
1: rolling_3_at_bats 0.39711086 0.37859154 0.18732008
2: rolling_2_at_bats 0.32533726 0.22391923 0.16358707
3:            prev_1 0.18454038 0.24196639 0.15120171
4:        ff_quality 0.03205515 0.04834345 0.13245632
5:       pitch_count 0.02211143 0.03299763 0.10343442
6:  ff_release_point 0.01386899 0.01504530 0.09586932
7:             xwoba 0.01340168 0.04214845 0.07916583
8:   ff_spin_quality 0.01157424 0.01698800 0.08696525
```

Figure 18. Summary Statistics by Gradient Boosting with xwOBA



```
Train_RMSE Test_RMSE  Train_R2   Test_R2
 0.5984129 0.6131461 0.3253166 0.3058412
```

Figure 19. Model Performance with xwOBA

After adding the variable, the regression model still does not consider it important enough with the gain being as low as the lowest variables like ff_release_point. This suggests that batter performance does not play a strong predictive role in this specific modeling setup, likely because the dataset is dominated by recent pitch-by-pitch context rather than batter skill.

Advantages of adding this variable is that it captures batter skill not present in the original model, and it combines multiple inputs into one meaningful number. Some of the extensions that can be made is that like pitcher effectiveness, rolling xwOBA can be added to improve the model. Moreover, when an elite pitcher is pitching, batters are expected to perform worse. Interaction term between batter xwOBA and pitcher quality can also be taken into consideration.