OAKLAND UNIVERSITY™

Oakland University Advanced Baseball Analytics

Predicting Runs Batted In

Submitted by

Aswini Sivakumar

In Partial Fulfillment of the Requirements for the Degree of

Masters in Business Analytics

April 2024

Table of Contents

Abstract 3
Organization/ Industry Description 4
Problem Description 4
Organizational Sponsors 5
System Capabilities 5
Business Benefits 6
Project Deliverables 6
Methods Used for Analysis 7
    Data Preparation 7
    Exploratory Data Analysis 7
    Feature Extraction 7
    Model fitting 8
Analysis Results 9
    Accuracy Score 9
    Confusion matrix 9
    Classification report 11
Solution Evaluation 12
Lessons Learned 12
Limitations & Future work 12
Appendixes 13
    Appendix A : Understanding Basic counts in baseball (Baseball Prospectus, 2024) 13
    Appendix B : Understanding sabermetrics (MLB.com, 2024) 15
    Appendix C : Libraries and functions used 17
    Appendix D : Player encoding mapping 18
    Appendix E : EDA for RBI 19
    Appendix F: Coefficient of logistic regression 20
    Appendix G: Variable importance graph 22
    Appendix H : First 3 decision trees from random forest 23
References 24

Abstract

The Oakland University Athletics have a successful baseball team that has won their conference championships regularly since 1970. Recognizing the significance of sabermetrics in modern baseball, the integration of advanced statistical analysis such as sabermetrics can offer profound insights into player performance and strategic decision-making (Thorn & Palmer, 1985). They currently use a third-party analytics platform, 643 charts (643 Charts, 2023), to track player performance metrics like Runs Batted In (RBIs), wOBA (weighted On Base Average), wRAA (weighted Runs above average), OBP (On-Base Percentage), BABIP (Batting average on balls in play), SPD (Speed score) (MLB.com, 2024) and many more. However, this external system is expensive. To gain more advanced analytics capabilities in a cost-effective way, Oakland University wants to build an in-house solution for calculating advanced statistics.

Additionally, conducting predictive modeling on batter's RBIs can provide valuable insights to help the team make strategic decisions and improve their performance. An in-house analytics system would allow them to quickly incorporate the latest game data and metrics, thereby supporting timely, data-driven decision making (Lewis, 2003).

Organization/ Industry Description

The Oakland University Athletics have a great baseball team and have won the conference championships in the first 10 places since 1970. The team is committed to providing its student-athletes with the opportunity to compete at the highest level while also pursuing their academic goals.

Problem Description

In the ever-evolving world of baseball, data analytics has become a crucial tool for teams seeking to gain a competitive edge (Kelly, 2019). The Oakland University Baseball team recognized the potential of leveraging data to enhance their player evaluation, game strategy, and decision-making processes. This project aimed to develop a technology solution that would empower the coaching staff with valuable insights and predictive capabilities.

At present, they depend on a third-party system, 643 charts (643 Charts, 2023), to monitor a player's performance, spot trends, and comprehend the influence each player has on the game. However, this application comes at a hefty price.

However, having a solution built in-house for baseball analytics would help the team gain access to more advanced statistical metrics. Also, it can be updated quickly with the recent matches played. This would be cost-effective and help the team make more informed and timely decisions on improving the strategy for their upcoming games.

Organizational Sponsors

This project involved several key organizational stakeholders from both the athletic and technology domains who provided comprehensive perspectives to properly frame the analytics needs and create an effective solution.

Role	Name	Responsibility
Coaching Staff	Jordon Banfield	Subject matter expert involved in articulating the scope of the project.
Project Guides	Prof. Venugopal Balijepally Prof. Vijayan Sugumaran	Provided technical oversight and mentorship. Helped scope the project and define requirements.
Athletic Communication Coordinator	Michael Reedy	Provides the current XML files containing player statistics for each game played during this season.
Student Contributors	Aswini Sivakumar Lauren Goralczyk	Extracted useful statistics from XML file. Developed models for predicting certain batter outcomes.

System Capabilities

This project provides Oakland University Athletics with the ability to calculate a comprehensive set of sabermetrics (Appendix B) and exploration of the pitch sequences. By analyzing the intricate patterns and characteristics of these sequences, the coaching staff can gain a deeper understanding of player's batting strategies and tendencies (Kelly, 2019).

The system is designed to accept any number of XML files containing the game details as input and extract relevant tags and features (Hanson, 2020). This information is then converted into structured formats like data frames, which can then be downloaded as a csv file. This data processing capability forms the foundation for advanced analytics and predictive modeling.

The data from the csv file contains basic counting stats (Appendix A), which are used to compute more advanced stats by applying respective formulas.

The system leverages machine learning techniques, particularly utilizing the scikit-learn library, to develop predictive models. The main focus is on forecasting a batter's likelihood of achieving Runs Batted In (RBIs), but the modeling capabilities can be expanded to cover other key performance indicators.

Business Benefits

The implementation of this project is expected to deliver several key business benefits to the Oakland University Baseball team. Improved player evaluation is a primary advantage, as the insights gained from advanced stats provide information on various aspects of a player's performance (Kelly, 2019). For example, these stats indicate whether the batter gets aggressive and swings the bat on the very first pitch, how many times the player was caught stealing while attempting a stolen base, how many times the player grounded into double plays, or how many times the batter swung the bat and missed the ball.

Secondly, the enhanced understanding of players' batting patterns and the ability to predict RBI outcomes provide insights into evaluating a player's ability to contribute to the team's offense by driving in runs (Barnes et al., 2024). All of these insights can eventually help the coaching staff develop more effective game strategies, determine the order of line-ups during a game, and facilitate player development (Kelly, 2019). Ultimately, the data-driven approach enhances the quality of the game played by empowering the coaching staff to make more informed and evidence-based decisions, giving the team a fair chance of winning (Barnes et al., 2024).

Project Deliverables

This project has two primary deliverables. The first deliverable is an Excel file that consolidates the basic statistics extracted from the XML files, along with advanced statistics computed from the basic counts (Hanson, 2020).

The second deliverable is a predictive model designed to forecast RBI outcomes. This project employed and compared the performance of multiple machine learning models, including Random Forest Classifier, Logistic Regression, and Decision Tree Classifier. The feature importance analysis and coefficient visualization helped identify the key factors influencing RBI outcomes, such as at-bats, number of balls and strikes, and base-loading situations.

Methods Used for Analysis

Data Preparation

The project utilizes the Python library xml.etree.ElementTree to extract the roots of the XML tags (Almazan, 2023) containing play-by-play information. Required counts, such as at-bats, number of balls, strikes, doubles, triples, hits, and others, were identified (Hanson, 2020). Subsequently, all extracted variables were saved to a dataframe, resulting in a dataset comprising 830 rows from 24 games played this season.

Exploratory Data Analysis

Performed visualization to understand the count of the various levels of RBI. RBI consisted of 5 levels, which were combined to create 2 levels - 0 and 1. RBI levels 1, 2, 3, and 4 were combined to form level 1. Overall, there were 712 records for RBI level 0 and 118 for level 1.

Feature Extraction

Performed label encoding using scikit-learn's LabelEncoder library to convert all categorical variables, such as player_name, batter_side, and pitcher_prof, which are represented as strings, into numerical format (Pedregosa et al., 2011). Additionally, to incorporate the categorical variable player_name, which had 18 levels, into the logistic regression model, 18 dummy variables were created using the get_dummies function from the pandas library. The column containing level 0 was dropped, as the model considers it as the reference class.

Model Fitting

The data is split into 80% training data and 20% test data using the train_test_split function from scikit-learn's model selection library (Pedregosa et al., 2011). Since the classes of RBI are imbalanced, a frequency percentages table was developed to verify if the proportions of classes are equally split between the train and the test set. Approximately 85% of the records belonged to class 0, while 15% of the records belonged to class 1 in both the datasets.

```
Frequency percentages for y1_train:
c_rbi
0    85.843373
1    14.156627
Name: count, dtype: float64

Frequency percentages for y1_test:
c_rbi
0    85.542169
1    14.457831
Name: count, dtype: float64
```

Various predictive models, such as random forest classifier, decision tree, and logistic regression, were employed using the fit and predict method of scikit-learn's library (Pedregosa et al., 2011). The predictors included in the models were player_name, strikes, balls, first_occ, second_occ, third_occ, batter_side, context, outs, and ab. The encoded player_name is included as a single predictor in the random forest and decision tree models, as they have the capability of handling categorical variables with multiple levels without requiring dummy encoding. However, the dummy variables created for player_name are included in the logistic regression model.

Analysis Results

Accuracy Score

When comparing the 3 models developed, the accuracy scores for the training and test data were obtained using scikit-learn's accuracy_score function (Pedregosa et al., 2011).

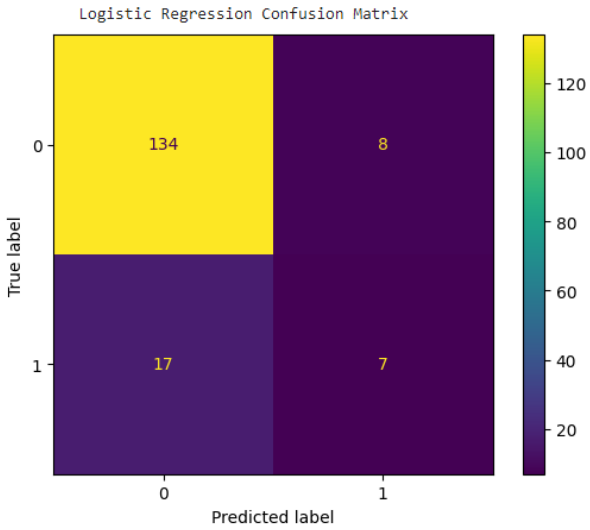
Model Name	Training accuracy	Test accuracy
Logistic Regression	89 %	85 %
Random Forest	99 %	86 %
Decision Tree	99 %	84 %

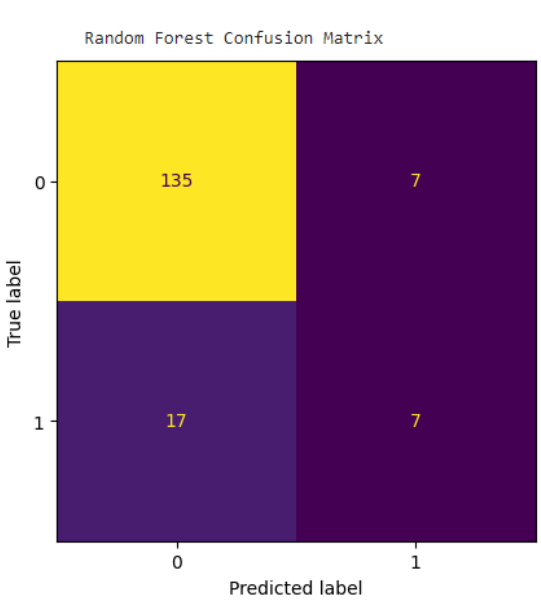
Based on the accuracy score, the Random Forest model has the highest accuracy on the test data. However, on developing the confusion matrix using scikit-learn's confusion_matrix function, the below results were obtained.

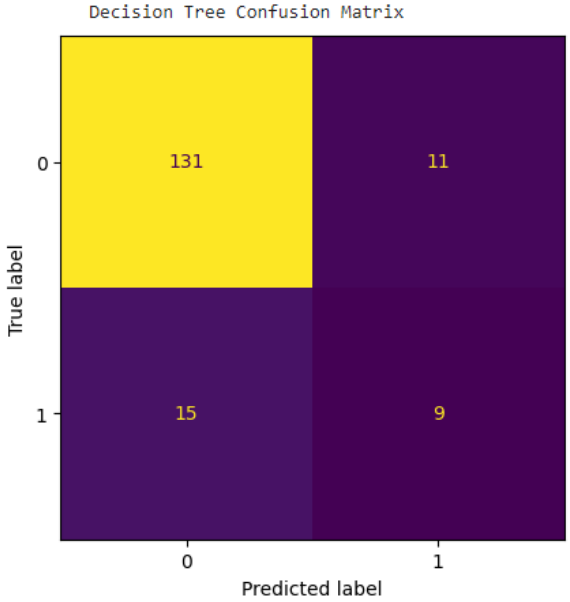
Confusion matrix

The confusion matrix helps identify how many of the classifier model's predictions were correct and how many were incorrect (Sokolova & Lapalme, 2009). The rows represent the true labels, while the columns represent the predicted labels (Sokolova & Lapalme, 2009). The numbers on the diagonal indicate the number of times the predicted value matches the true value, while the numbers in the other cells represent the number of times the classifier predicted incorrectly (Sokolova & Lapalme, 2009).

The project aims to identify when RBI is 1. The Logistic Regression and random forest classifier models accurately predicted class 1 seven out of 24 times, whereas the decision tree model predicted class 1 accurately nine times out of 24. This indicates that the Decision Tree is the best model among the three, even though its test accuracy is slightly lower compared to the other models.


Logistic Regression Confusion Matrix
134
8
17
7
0
1
True label
0
1
Predicted label
120
100
80
60
40
20


Random Forest Confusion Matrix
135
7
17
7
0
1
True label
0
1
Predicted label


Decision Tree Confusion Matrix
131
11
15
9
0
1
True label
0
1
Predicted label

Classification report

The classification report helps to comprehend the model performance even better by closely examining metrics such as precision, recall, and F1-score.

Classification report for Decision Tree [[131 11]
[15   9]]:

	precision	recall	f1-score	support
0	0.90	0.92	0.91	142
1	0.45	0.38	0.41	24
accuracy			0.84	166
macro avg	0.67	0.65	0.66	166
weighted avg	0.83	0.84	0.84	166

Precision is the metric that indicates how often the classifier predicted a class correctly when that class is predicted (Hossin & Sulaiman, 2015). For instance, the precision of class 1 is 0.45. This value represents the number of correctly identified "1s" divided by the total number of times the classifier predicted "1," whether correctly or incorrectly. Therefore, this indicates that the classifier correctly identified "1s" 45% of the time it predicted a "1".

Recall is the metric that tells how often a class is predicted correctly when it actually belonged to the class (Hossin & Sulaiman, 2015). For instance, the recall for class 1, which is also known as specificity is 0.38. This indicates that only 38% of the time, the number of actual 1s that the classifier correctly identified as such. The recall for class 0, which is also known as sensitivity is 92%.

F1 score is the combination of precision and recall metrics. If the value of either one of the metrics is low, then we can expect a lower F1 score. This is the quickest measure that tells if a classifier is actually good at predicting the correct class (Hossin & Sulaiman, 2015).

Overall, for class 1, the precision is 0.45, indicating that the classifier mis predicted more 0s as 1s than predicting the 1s correctly. Additionally, the recall is very low, suggesting that the classifier missed predicting some of the 1s that were actually "1s". This imbalance in the classification is clearly depicted by the low F1 score for this class.

This model can be improved further by adding additional records and including more useful predictors.

Solution Evaluation

The completed deliverables of this project have been recognized by the coaching staff Jordon. Feedback was provided on the variables included in the model to predict RBI, with suggestions to include variables deemed important for driving in runs through RBI. As the season is still going, the coaching staff can utilize the stats and the model to devise strategies for the upcoming games.

Lessons Learned

One of the key lessons learned during the project execution was the efficiency of using Python libraries for data extraction and aggregation, compared to the traditional approach of importing XML files into Excel. The xml.etree.ElementTree library helped to easily scrape the necessary attributes from the XML files and aggregate the data into a structured dataframe. This approach proved to be more streamlined and less time-consuming than struggling to import the XML files into Excel and manually extract the required information (Almazan, 2023). This project helped in recognizing the power of Python's data manipulation capabilities, which enables a quick transformation of the raw data into a format suitable for analysis and model development.

Limitations & Future work

While the current solution provides valuable insights and predictive capabilities, there is much more scope for improvement. The project is designed to only accept XML files, and if the file format changed in the future, the code would need to be tweaked accordingly. Additionally, the current dataset does not contain detailed pitch data, such as the type of pitch (fastball, curveball, slider, or changeup) thrown. If such pitch data were available, the project could be further enhanced by analyzing and developing more sophisticated models to predict RBI or other such useful outcomes based on the specific pitch characteristics.

In the future, the Oakland University Baseball team should consider expanding the project to incorporate additional data sources, such as scouting reports, trackman analysis, and player biomechanics. By integrating these diverse datasets, the coaching staff could gain a more comprehensive understanding of player performance and develop even more accurate predictive models. Additionally, the team should explore the possibility of automating the data extraction and update processes, ensuring the solution remains relevant and up to date as the baseball season progresses.

Appendixes

Appendix A: Understanding Basic counts in baseball (Baseball Prospectus, 2024)

The counts specified in this section are extracted from the XML file.

Abbreviations	Full Form
AB	At Bats
R	Runs
H	Hits
1B	Single
2B	Doubles
3B	Triples
HR	Homeruns
RBI	Runs Batted In
BB	Walks
HBP	Hit By Pitch
SO	Strike Outs
GDP	Grounded into Double Play
SF	Sacrifice Flies
SH	Sacrifice Hits
SB	Stolen bases
CS	Caught Stealing
PO	Put Outs
A	Assists
E	Errors
KL	Strikeout looking
ROE	Reached On Error
PU	Pop Ups
Ground	Ground Balls
Fly	Fly Balls
BUNT	Bunts
Bunt_1B	Singles bunts
Squeeze	Bunts and runners on third base
Num_Pitches	Total number of pitches
INF_1B	Infield singles
LD_count	Line Drives

IFFB_count	Infield fly balls
base_1_3_Single	Advanced to third base from first base when batter hit single
base_2_h_Single	Advanced to home from second base when batter hit single
base_1_h_Double	Advanced to home from first base when batter hit double
c_base_1_Single	On first base when batter hit single
c_base_2_Single	On second base when batter hit single
c_base_1_Double	On first base when batter hit double
sb2	Stolen base to second base
sb3	Stolen base to third base
cs2	Caught stealing while advancing to second base
cs3	Caught stealing while advancing to third base
SBA2	Stolen base attempts to second base
SBA3	Stolen base attempts to third base
ab_first_loaded	At bats when first base is occupied
swingcount	Number of times batter swinged
SW1	Number of times batter swung at the first pitch
Take1K	When last pitch in the pitch sequence is a strikeout
SW_Last	Number of times batter swung at the last pitch (foul or missed)

Appendix B: Understanding sabermetrics (MLB.com, 2024)

The following sabermetrics are calculated using the specified formulas in an excel file.

Sabermetrics	Definitions	Formula
PA	Plate Appearance	AB + BB + SH + SF + HBP
BA	Batting Average	H/AB
OBP	On base percentage	(H + BB + HBP)/(AB + BB + HBP + SF)
TB	Total Bases	(Singles) + (2 * Doubles)+ (3 *Triples) + (4* HR)
SLG	Slugging Percentage	Total Bases / AB
OPS	On-base Plus Slugging	OBP + SLG
ISO	Isolated Power	SLG - BA
BIP	Balls In Play	AB-K-HR+SF
BABIP	Batting average on balls in play	(H-HR)/ BIP
QAB	Quality At Bats	TB+BB+HBP+RBI+R+(0.5*SB)+(0.5*ROE)
K/BB	Strikeouts to walk ratio	K/BB
GB%	Percentage of Ground Ball	GB / BIP
FB%	Percentage of fly balls	FB / BIP
LD%	Percentage of Line Drives	LD / BIP
IFFB%	In field fly balls rate	IFFB / BIP
PU	Pop ups rate	PU / BIP
P/PA	Pitches Per Plate Appearance	Total Pitches / PA
S/M	Swing and miss	SW_Last / Swingcount
SW	Swing Percentage	Swingcount / Total pitches
HR/FB	Home run to fly ball ratio	HR / FB
SOr	Strikeout rate	SO / PA
BBr	Walk rate	BB / PA
wOBA	Weighted On base average	((0.694*BB)+(0.726*HBP)+(0.888*1B)+(1.252*2B)+(1.578*3B)+(2.017*HR))/(AB+BB+SF+HBP)
wRAA	Weighted runs above average	((wOBA - wOBA of the entire league) / annual wOBA scale) x PA
RC	Runs created	((H + BB)*TB)/(AB+BB)
GDPr	GDP rate	GDP / ab_first_loaded
SPD	Speed score	
1st-3rd on 1B	Advance 2 bases (1-3) on a single	base_1_3_Single/c_base_1_Single
2nd-H on 1B	Advance 2 bases (2-h) on a single	base_2_h_Single/c_base_2_Single
1st-H on 2B	Advance 3 bases (1-h) on a double	base_1_h_Double/c_base_1_Double
pos	Fielding position	Average(f1,f2,f3,f4,f5,f6)
GP	Total Number of games played	
f1	stolen base percentage	(((SB + 3) / (SB + CS + 7)) - 0.4) * 20
f2	Stolen base attempts	SQRT((SB+CS)/(Singles + BB + HBP))/0.07

f3	Triples factor	(Triples/(AB-HR-K))/0.0016
f4	Runs scored factor	(((R - HR) / (H+ BB + HBP - HR)) - 0.1) * 0.25
f5	GDP factor	(0.063 - (GDP / (ABS - HR - K))) / 0.007
f6	Defensive position and range factor	P: f6 = 0 C: f6 = 1 1B: f6 = 2 2B: f6 = (((PO + A) / GP) / 4.8) * 6 3B: f6 = (((PO + A) / GP) / 2.65) * 4 SS: f6 = (((PO + A) / GP) / 4.6) * 7 OF: f6 = (((PO + A) / GP) / 2.0) * 6

Appendix C: Libraries and Functions Used

Libraries/ Functions used	Usage
xml.etree.ElementTree	Utilized for parsing and manipulating XML data (Almazan, 2023) (Hanson, 2020)
parse()	To parse the XML files and obtain the root of the XML tree
getroot()	Returns the top-level element of the XML document, which serves as the starting point for navigating and extracting data from the XML structure
findall()	To locate the specific tags within the XML tree that contained the desired information, such as the "team" and "player" tags
attrib or attrib.get	Dictionary-like property of the Element objects
find()	To locate specific child elements, such as the "pitches" tag within the "play" tag
os	Used to interact with the operating system (Oliphant, 2007)
os.listdir()	To get a list of all the files in the directory
endswith()	To filter the file list and with respect to project only process the XML files
os.path.join()	To construct the full path to the XML files by combining the directory path and the file names
re	To match and manipulate patterns in text such as pitch sequence (Oliphant, 2007)
re.compile()	To validate patterns like Bunts in the action attribute
pandas	To facilitate working with dataframes (McKinney, 2010)
get_dummies()	To create binary dummy variables for each level in a categorical column
DataFrame()	To create a dataframe from the extracted data
groupby()	To perform data transformation such as aggregate the sum of counts obtained
apply()	To perform data manipulation functions on a column
drop()	To drop columns or rows based on specified conditions
matplotlib	To perform data visualization (Hunter, 2007)
pyplot	To create bar plot for showing data distribution and visualizing model coefficients
seaborn	To perform informative statistical data visualization (Waskom, 2021)
Scikit-learn - fit and predict	To develop efficient machine learning models (Pedregosa et al., 2011)
train_test_split	Function of model_selection used to split data into training and test sets
LabelEncoder	Function of preprocessing used to encode categorical variables as numerical values
RandomForestClassifier	Function of ensemble that combines multiple decision trees for prediction task
LogisticRegression	Function of linear_model used to predict classification problems
DecisionTreeClassifier	Function of tree that creates tree-like model of decisions to make

	predictions
export_graphviz	Function of tree that used to visualize and export the decision tree structure
accuracy_score	Function of metrics used to calculate accuracy of the model
confusion_matrix	Function of metrics used to generate a confusion matrix
ConfusionMatrixDisplay	To visualize confusion matrix for the predictive models
classification_report	Function of metrics used to generate a performance report for the model

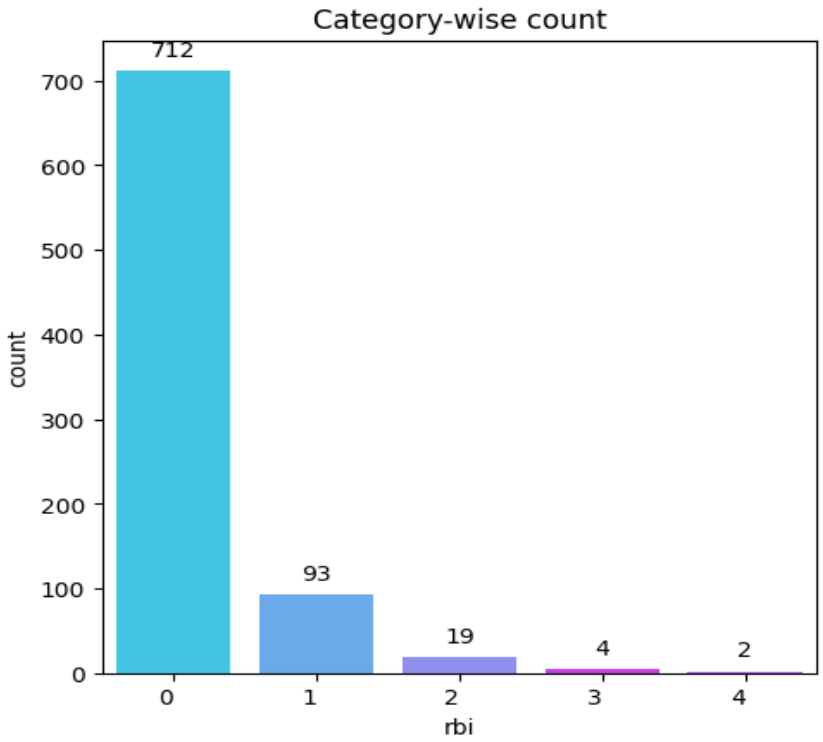
Appendix D: Player Encoding Mapping.

The figure below depicts the mapping of the encoded player name to their actual names.

	player_name	player_name_orig
0	15	R. Bussey
1	2	B. Heidal
2	9	I. Cleary
3	3	B. Nigh
4	12	L. Day
5	7	G. Arseneau
6	0	A. Orr
7	6	E. Larsen
8	16	S. Griffith
27	17	T. Rice
34	5	D. Gaskins
44	10	J. Lauinger
47	8	H. Griffith
76	14	P. Gunnell
79	1	B. Clark
80	13	L. Pollock
153	4	C. Hain
158	11	J. Lux

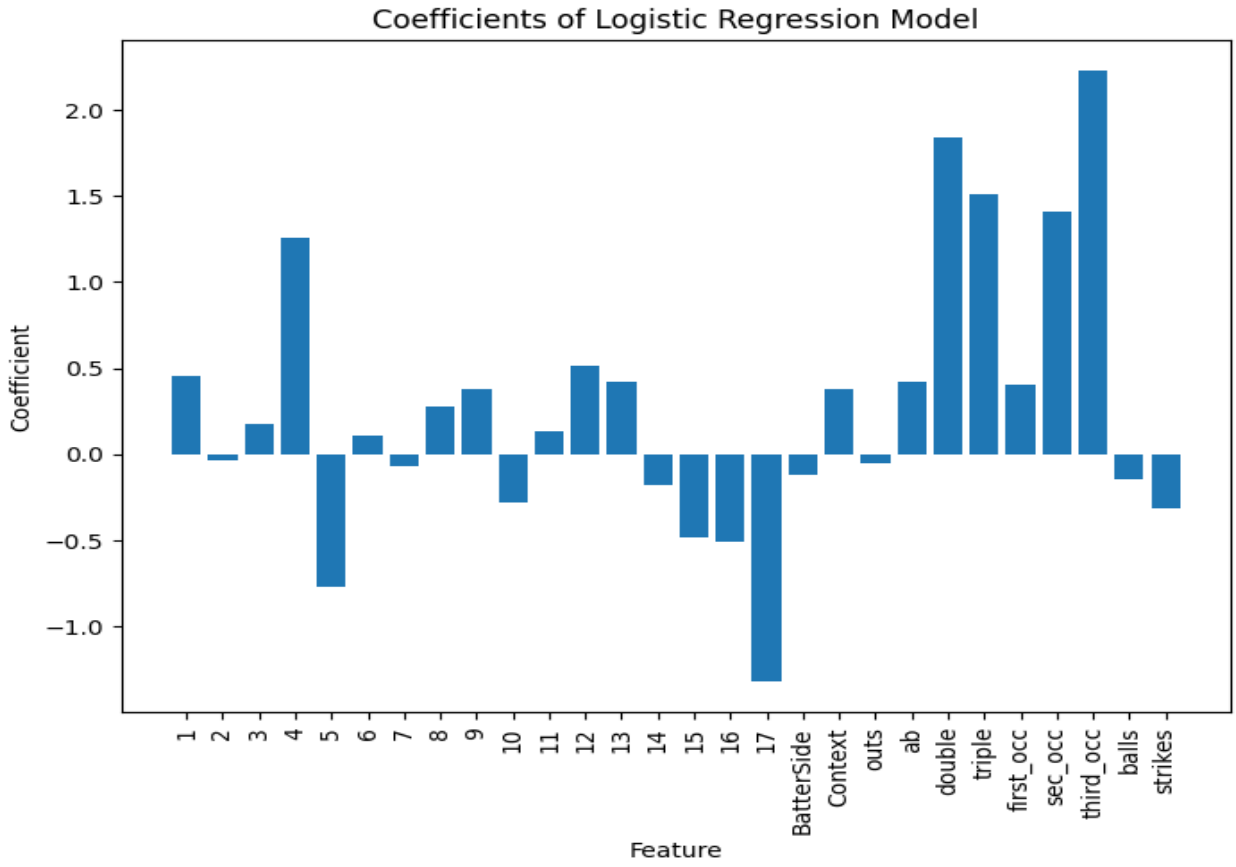
Appendix E: EDA for RBI

The below bar chart shows the counts for the various levels of RBI.



Appendix F: Coefficient of logistic regression

From the coefficients graph, it is evident that having a runner on third base increases the batter's chance of driving in runs (RBI).



For the categorical variables with 2 levels, the first encountered level will be considered as the base reference level by the model. Therefore, the reference class for batter side is 0, which means right-handed and the reference class for context (pitcher proficiency) is 1, which means left-handed.

	variable	coefficient	odds_ratio				
0	intercept	-3.281825	0.037560	14	14	-0.180422	0.834918
1	1	0.450749	1.569487	15	15	-0.485475	0.615405
2	2	-0.031007	0.969469	16	16	-0.506756	0.602447
3	3	0.172779	1.188604	17	17	-1.320277	0.267061
4	4	1.255700	3.510295	18	BatterSide	-0.121726	0.885391
5	5	-0.769952	0.463035	19	Context	0.382035	1.465264
6	6	0.110043	1.116326	20	outs	-0.055265	0.946235
7	7	-0.069926	0.932463	21	ab	0.422005	1.525016
8	8	0.276857	1.318978	22	double	1.836355	6.273629
9	9	0.377100	1.458050	23	triple	1.513639	4.543232
10	10	-0.279468	0.756186	24	first_occ	0.402243	1.495175
11	11	0.137340	1.147218	25	sec_occ	1.408019	4.087848
12	12	0.513947	1.671878	26	third_occ	2.225519	9.258285
13	13	0.421086	1.523615	27	balls	-0.143054	0.866707
				28	strikes	-0.313228	0.731083

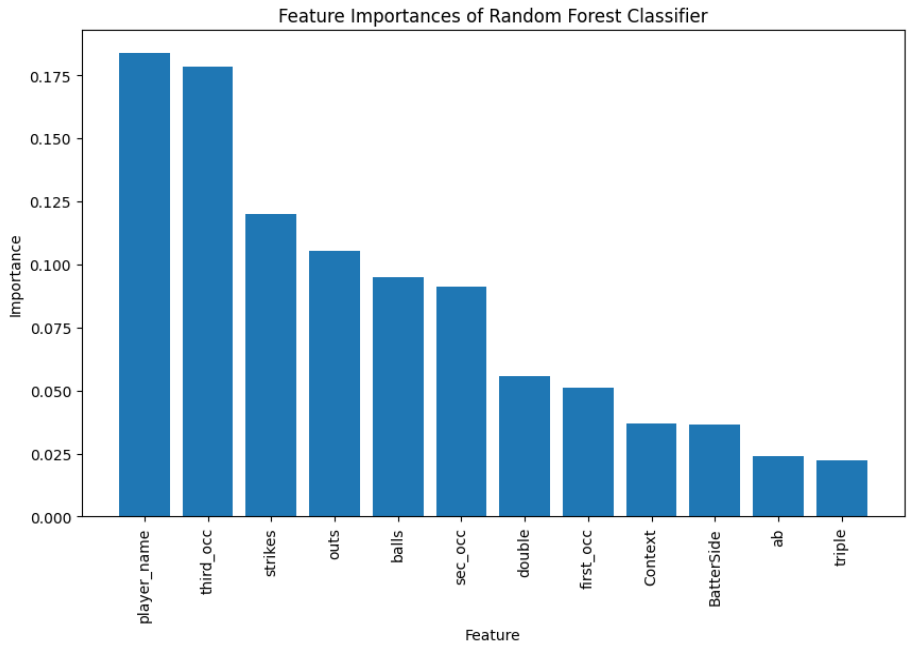
Here are the interpretations of few variables from the model-
The odds of a batter scoring an RBI when third base is occupied are 9.25 times higher compared to when third base is unoccupied. This aligns with the expectation that having a runner on third base creates more opportunities for RBIs, as even a simple sacrifice fly or ground ball can drive in the run.

The odds of a left-handed batter scoring an RBI are 11.5% times lower than the odds for a right-handed batter.

The odds of a batter scoring an RBI are 49% higher when the first base is occupied compared to when the first base is empty.

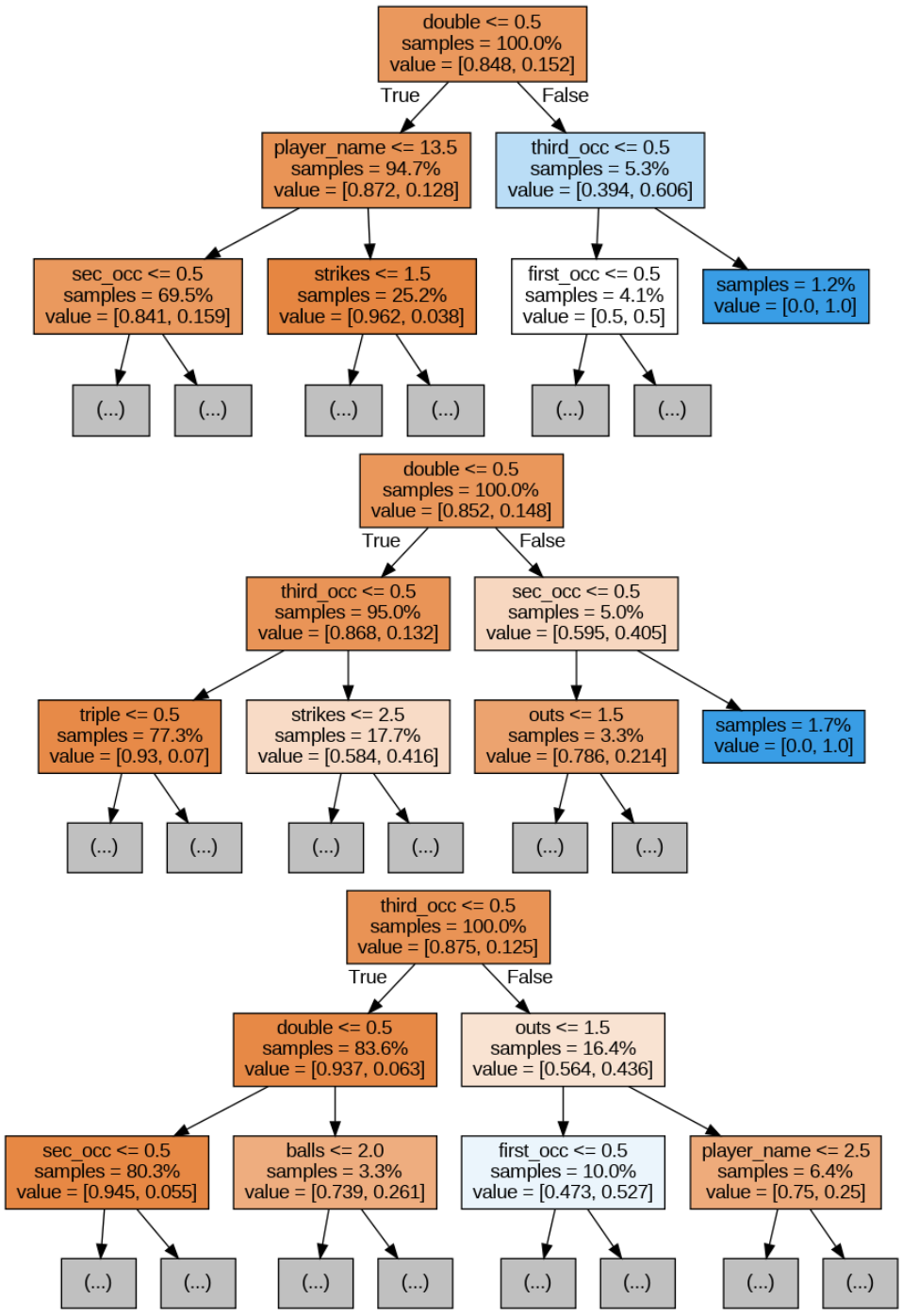
Appendix G: Variable importance graph

The variable importance graph shows that player_name and having a runner on third base are the two most important predictors of RBI.



player_name	0.183655
third_occ	0.178407
strikes	0.120014
outs	0.105272
balls	0.094921
sec_occ	0.091138
double	0.055565
first_occ	0.051104
Context	0.037060
BatterSide	0.036648
ab	0.024030
triple	0.022186

Appendix H: First 3 decision trees from random forest



References

643 Charts. (2023). *Interactive Stats – 6–4–3 Charts.* Retrieved April 9, 2024 from 6–4–3 Charts: https://643charts.com/web-application/

Almazan, W. (2023, November 23). *Processing XML Files with Python's xml.etree.ElementTree.* Retrieved April 11, 2024 from DevNet Journey: https://devnetjourney.com/processing-xml-files-with-pythons-xmletreeelementtree

Barnes, S., Bjarnadóttir, M., Smolyak, D., & Thiele, A. (2024). A data-driven optimization approach to baseball roster management. *Annals of Operations Research, 335*, 33-58.

Baseball Prospectus. (2024). *Glossary.* Retrieved April 11, 2024 from Baseball Prospectus: https://legacy.baseballprospectus.com/glossary/

Hanson, C. R. (2020, March 1). *Getting started using Python's ElementTree to navigate XML files.* Retrieved April 11, 2024 from CoreyHanson: https://coreyhanson.com/blog/getting-started-using-pythons-elementtree-to-navigate-xml-files/

Hossin, M., & Sulaiman, M. (2015, March). A review on evaluation metrics for data classification evaluations. *International Journal of Data Mining & Knowledge Management Process, 5*(2), 1 - 11.

Hunter, J. D. (2007, June 18). Matplotlib: A 2D Graphics Environment. *Computing in Science & Engineering, 9*(9), 90 - 95.

Kelly, M. (2019, May 27). *Sabermetrics in Baseball: A Casual Fans Guide.* Retrieved April 11, 2024 from MLB.com: https://www.mlb.com/news/sabermetrics-in-baseball-a-casual-fans-guide

Lewis, M. (2003). Moneyball: The Art of Winning an Unfair Game. W. W. Norton & Company.

McKinney, W. (2010, January). Data Structures for Statistical Computing in Python. *Python in Science Conference, 445*, 51-56.

MLB.com. (2024). *Advanced Stats | Glossary.* Retrieved April 5, 2024 from MLB.com: https://www.mlb.com/glossary/advanced-stats

Oliphant, T. (2007, June). Python for Scientific Computing. *Computing in Science & Engineering, 9*(3), 10-20.

Pedregosa, F., Varoquaux, G., & Gramfort, A. (2011). Scikit-learn: Machine learning in Python. *Journal of machine learning research, 12*(Oct), 2825-2830.

Sokolova, M., & Lapalme, G. (2009). A systematic analysis of performance measures for classification tasks. *Information Processing & Management, 45*(4), 427-437.

Thorn, J., & Palmer, P. (1985). *The Hidden Game of Baseball.* Doubleday.

Waskom, M. L. (2021). seaborn: statistical data visualization. *Journal of Open Source Software, 6*(60), 3021.