PREDICTING MAJOR LEAGUE BASEBALL WORLD SERIES WINNERS

Radhika Kulkarni

I. INTRODUCTION

Baseball is “America’s Pastime”, a game that has been played for over 150 years. Its roots run deep from generation to generation. Millions of fans pour into ballparks every year to see their beloved team and favorite players slug home runs, make highlight reel defensive plays, and just simply get to step back in time and be kids again for a few hours on a hot summer afternoon. At the end of the day, players, coaches, front office executives and fans all share a common desire: to win the World Series.

Countless factors go into one team’s success in a particular year. The players, their families, the millions of world-wide fans, the team management, coaching staff, weather, the techniques, the game rules, and just plain old luck. These factors are just a few of the enormous, complex equation of factors that go into the success of one team. The impacts are equally enormous. Think of all the people who we just mentioned. They, and the rest of America, waiting with bated breath in their living rooms, in sports bars and backyard barbecues, are incredibly invested in the outcome of this once a year World Series Championship. The problem we will address is how we can predict, using a team’s statistics in a given season, whether they are likely to win the World Series Championship. We have found a dataset called “Baseball Databank” off Kaggle, which contains a file titled “Teams.csv”. This file contains season statistics for each team from all the way back to 1871 until 2015. Which features, factors or variables will be most influential in determining with greatest accuracy the winner of the World Series?

II. BACKGROUND

Perhaps the most important factor that plays a role in every decision teams make today is sports analytics. The first true advocate for diving deep into analytics and finding values in players no one else could see was the 2002 Oakland A’s, whose efforts were famously brought to the spotlight in the 2011 film *Moneyball* starring Brad Pitt and Jonah Hill. In the two decades

since, teams have gone all in on the numbers movement, using analytics to decide which players to sign, trade, play more, play on a particular day, and overall, how to build a team of 26 players capable of bringing home a championship.

Much of the influence for our project was drawn from the article “Predicting Major League Baseball Championship Winners Through Data Mining” by Brandon Tolbert and Theodore Trafalis (2016). In their study, they applied SVMs with different kernels (linear, quadratic, cubic, Gaussian RBF) to find which model performed the best at predicting champions in the American League, National League, and World Series. This was a slightly different approach than our own study, but many of the conclusions they drew we expected to find similar results. Several of the features they found significant for model building validated what would be expected from our own baseball knowledge and justified our own choices in feature selection. They found Linear SVM and RBF SVM to be the best models for prediction, giving us a basis to compare our own model performance to. Although we used different datasets, both had imbalanced data and were linearly inseparable. In order to find the optimal boundary for our output, we had to find ways to make it balanced and linearly separable. Now we will discuss how we cleaned our data.

III. DESCRIPTIVE ANALYSIS

A. DATA CLEANING

Our original dataset “Teams.csv”, retrieved from a Kaggle titled “Baseball Databank”, was 2805 rows and 48 columns. There were 47 predictors and 1 target variable, the World Series winner. The original data covers the years 1871-2015, almost 150 years of baseball. However, baseball is an evolving sport and in this period of time many changes to the game were made. To allow our data to be on a level playing field, so to speak, we subsetted the data for the time period where the game rules and number of games played were mostly uniform. This subsetted age range is 1961-2015, around 55 years long. The reason why 1961 was chosen as the starting year is because previous to this year, the number of games played was not consistent across leagues and divisions. In 1961, the MLB Expansion brought about a 162-game schedule, or 18 games between each league opponent. This game schedule remains invariable until the current day, 2023 (although there is news that the MLB might change these rules soon).

In this study, we cleaned the data for null values, years in which there were anomalies in the dataset, and removed year ranges where the rules were uniform across the Major League Baseball franchises, leagues and teams. Some baseball events caused inconsistencies in our data, which we cleaned for as mentioned. These baseball events are the years 1981 and 1994, in which some turmoil arose in player vs owner relations. In 1981, there was a strike midseason, which shortened the number of games played to win the World Series. We removed this year because the World Series Championship was influenced by outside factors, and gameplay, rules and number of games was different compared to the rest of our data. We also observed many null values in 1994, which we discovered was due to a complete season strike, and there was no World Series Championship held in that year. Thus, we omitted this year from our dataset.

Other interesting developments in the data was the addition of expansion teams. Since baseball is ever changing, franchises have changed as well, and more teams have been added over the last 55 years. Expansion teams were added in 1962, 1969, 1977, 1993, and 1998, increasing the number of teams from 18 to the current 30. These teams were expansion teams: 1962 (NY Mets, Houston Astros), 1969 (Kansas City Royals, Montreal Expos [became Washington Nationals], San Diego Padres, Seattle Pilots [became Milwaukee Brewers]), 1977 (Seattle Mariners, Toronto Bluejays) , 1993 (Colorado Rockies, Florida Marlins), 1998 (Arizona Diamondbacks, Tampa Bay Devil Rays).

We subset the features based on the correlation plot (below) and dropped one of each pair of features that were highly correlated in most cases. We can see that there are some sparse correlations with other variables (sacrifice flies, hit by pitch, etc.), some very correlated. We used the plot as justification to drop one of each two pairs of variables (like earned run average and earned runs allowed, hits allowed and earned run average, etc.).

Additionally, we removed some features which we did not find important according to common baseball knowledge of gameplay statistics, and also variables which we saw had low correlation with the response variable World Series winner. As a result, we drop features games played, home games played, at bats, doubles, triples, homeruns, walks, batter strikeouts, stolen bases, caught stealing, hit by pitch, sacrifice flies, earned runs allowed, complete games, outs pitched, pitcher strikeouts, double plays, and fielding percentage. We found the numerical predictors of wins, losses, runs scored, hits, opponents' runs scored, earned run average, shutouts, saves, hits allowed, home runs allowed, walks allowed, errors, attendance to be most important (before more advanced feature selection techniques). We also found the categorical

predictors year, rank, league, franchise to be of significance. And of course we preserve our target variable of World Series winner.

Note: We include both wins and losses as important variables at this time for the exploratory data analysis. We will drop one of them in the later feature selection.

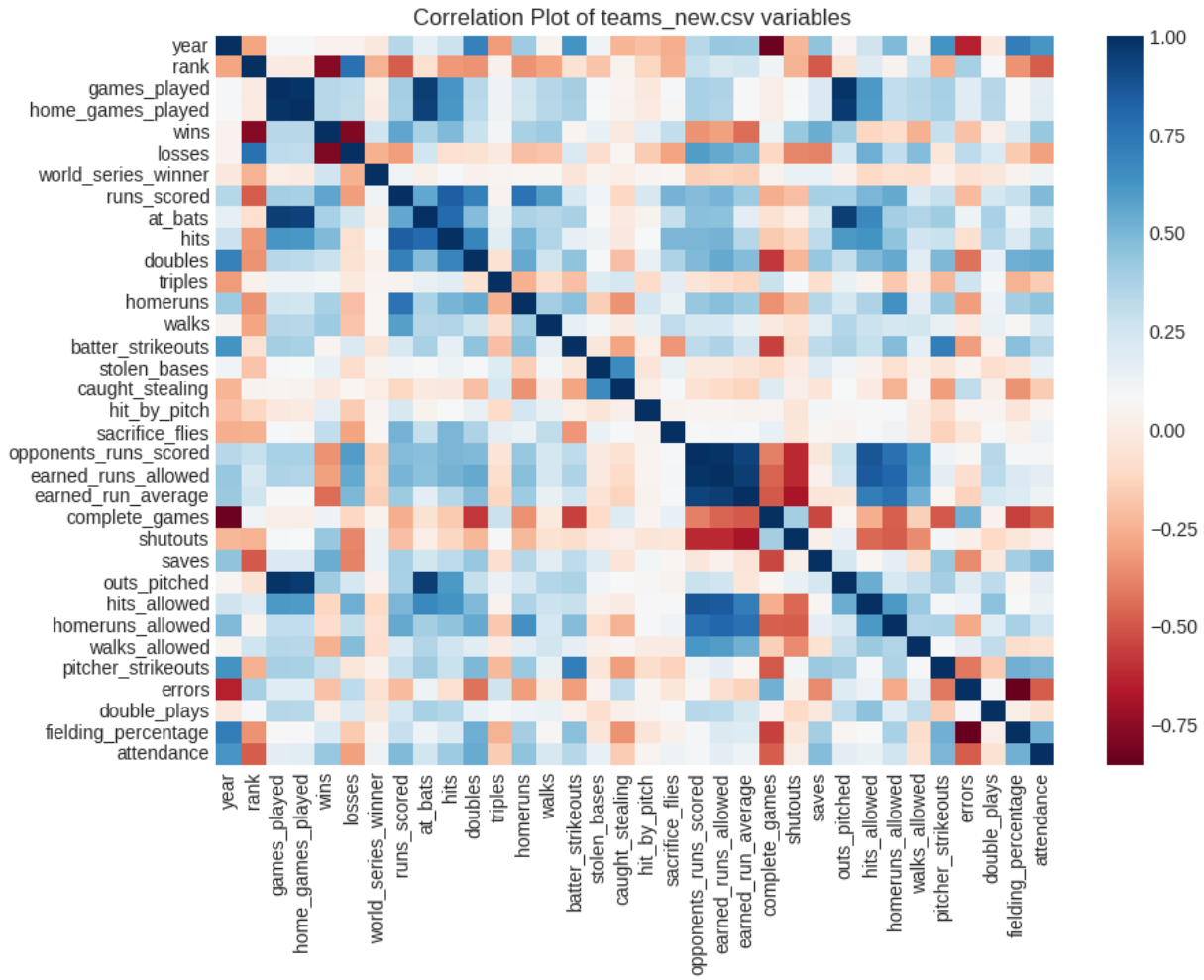


Figure 1: Correlation Matrix

After the cleaning, we have 1392 rows × 18 columns in our dataset. Each row in the dataset is a team summary statistics for that year's season. For example, the first entry in the Figure 2 below is the 1961 season, Baltimore Orioles team and their season's statistics, and whether they won the World Series.

	year	league	franchise	rank	wins	losses	runs_scored	hits	opponents_runs_scored	earned_run_average	shutouts	saves	hits_allowed	homeruns_allowed	walks_allowed	errors	attendance	world_series_winner
0	1961	AL	BAL	3	95	67	691	1393	588	3.22	21	33	1226	109	617	126	951089.0	0
1	1961	AL	BOS	6	76	86	729	1401	792	4.29	6	30	1472	167	679	143	850589.0	0
2	1961	AL	CHW	4	86	76	765	1475	726	4.06	3	33	1491	158	498	128	1146019.0	0
3	1961	NL	CHC	7	64	90	689	1364	800	4.48	6	25	1492	165	465	183	673057.0	0
4	1961	NL	CIN	1	93	61	710	1414	653	3.78	12	40	1300	147	500	134	1117603.0	0

Figure 2: First 5 entries of cleaned data

B. PREPROCESSING AND FORMULATION OF OUR TESTING DATA

We were interested to see how our models would perform on new data, after they are trained on our original dataset that covers years 1961-2015. As the year of our study is 2023, and since the dataset we have is until 2015, we thought why not test our models on data available post 2015? We obtained data from baseball-reference.com and RetroSeasons.com on years 2016-2019, and 2021-2022. 2020 was omitted due to the pandemic. Our testing set contains 180 rows and 18 columns.

Our next step is to transform the features of both the training/validation dataset and the testing dataset. We will perform one-hot encoding using pandas.get_dummies() function to convert the categorical features into numerical. We will also scale all the quantitative features using MinMaxScaler() from sklearn.preprocessing so all our numeric features will be on the same scale for training and testing purposes. We will now give the description of the features in our dataset.

C. DESCRIPTION OF VARIABLES

After cleaning, our variables are limited to:

- Year
    - Year in which the season was played
- League
    - Which league the franchise is associated with. There are two leagues in Major League Baseball: the National League (NL) and American League (AL). The winners of each league play each other in the World Series.
- Franchise
    - The franchise each team is associated with. This is denoted by the team's three letter abbreviation (Ex: Boston Red Sox are BOS, New York Yankees are NYY). Franchise differs from team because a franchise could relocate or rebrand their

team name throughout its existence (Ex: The Atlanta Braves were previously the Milwaukee Braves, and prior to that the Boston Braves, but all are affiliated with the same franchise). This simplifies our model over using teams.

- Rank
    - Each team's ranking in their division. Division realignments have happened multiple times since 1961, but there are currently 6 divisions with 5 teams each. Rankings scale from 1 to 5 (1 being the division winner, and 5 being last in the division).
- Wins
    - A team's win total for that season
- Losses
    - A team's loss total for that season
- Runs Scored
    - The number of runs scored by a team's offense for that season
- Hits
    - The number of hits by a team's offense for that season
- Opponent's Runs Scored
    - The number of runs allowed by a team's pitching staff during that season
- Earned Run Average (ERA)
    - The number of earned runs a team allows per nine innings. Earned runs are any runs that are scored without the aid of an error. The formula is 9 x earned runs / innings pitched.
- Shutouts
    - The number of games a team did not allow the opposing team to score.
- Saves
    - The number of saves by a team's pitching staff during that season. A save occurs when a pitcher comes in in the final inning of the game and gets the final three outs when his team is winning by 3 or less runs, without allowing the opposing team to tie or take the lead. Each team typically has a specific pitcher for this situation, called a "closer". Most winning teams have a good closer that comes in and gets lots of saves.
- Hits Allowed
    - The number of hits allowed by a team's pitching staff for that season
- Home runs allowed

    - The number of home runs allowed by a team's pitching staff for that season
- Walks Allowed
    - The number of walks (base on balls) allowed by a team's pitching staff for that season
- Errors
    - The number of defensive errors committed by a team's defense for that season
- Attendance
    - A team's total attendance for home games for that season
- **World Series Winner**
    - Takes value of 0 if the team did not win the World Series in that season or 1 if the team did win the World Series in that season. This is our response variable.

IV. EXPLORATORY DATA ANALYSIS

In order to glean some insights from the data, we performed exploratory data analysis through visualizing the features and examining the relationships between them. Since our target variable is the World Series winner, we graphed its univariate distribution to determine the split of our majority and minority class. We see from Figure 3 that the ratio of the majority class, losers, and the minority class, winners, is heavily unbalanced. This makes sense because there are around 30 teams which qualify for the World Series each season, but only one of the teams will be the World Series Winner. So each season will generate 29 entries in the majority class compared to just one entry in the minority category. We will need to deal with this imbalance in our data in our modeling later, because models trained on this data would be biased towards categorizing every new data point as a loser rather than a winner. We want to predict winners correctly so we will address this issue by using weighting, since our winning class has only 53 wins (and even less in our training data), we would obtain a comparatively small dataset if we were to balance the dataset by taking the same number of samples from the majority class.

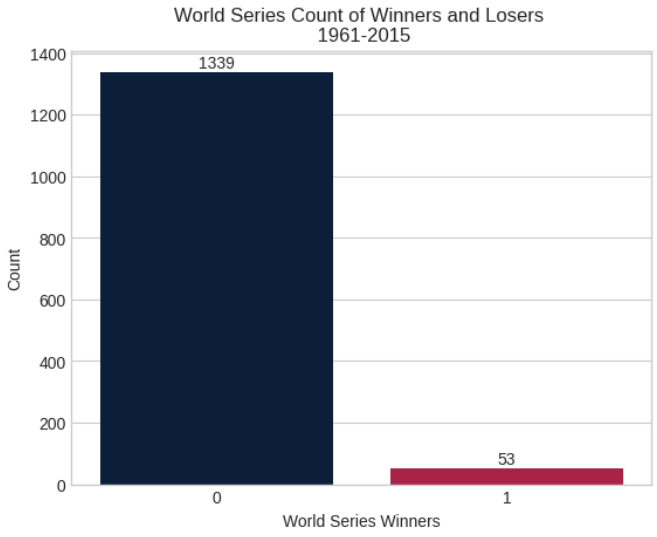


Figure 3: World Series Univariate Distribution

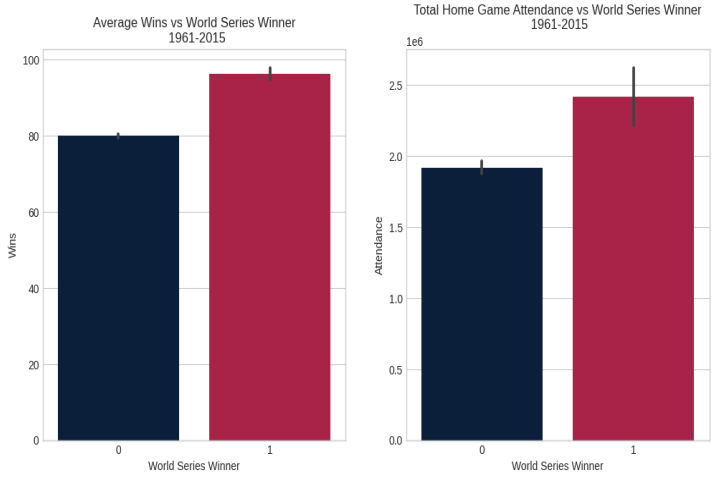


Figure 4: Average Wins, Home Game Attendance vs World Series

In Figure 4, we wanted to see the interactions of the World Series winner with other major predictors. We were interested in the wins of a team and whether they won the final game, the World Series. From the first plot in Figure 4, we observe that World Series winners averaged a higher number of wins than losers. This makes sense because the teams that go on to win the World Series were not knocked out of the competition earlier, so they played more games compared to the losers.

We also were curious about how the attendance of the games is influenced by World Series winners and losers. Fan loyalty is a big part of the game of baseball, and having a strong fan base can boost a team in both spirit and revenue. There is an argument that well-loved (and well-funded) teams are more likely to win the World Series, and the fans show how they love their teams, whether they win or lose, through their physical home game attendance. In the second graph of Figure 4, we note that the losers have home game attendance which is smaller by half a million people than winners' home game attendance. Fans want to see their team win at home games, because when their team wins, they feel like they have won too. The World Series games also have a larger draw of fans since it is the championship game. Still, a half million difference in fans indicates that fans of losing teams stick to their guns and still cheer on their teams regardless of what the outcome might be at their home games.

Next, we examine the relationships between the predictor features and compare them to the World Series losers and winners. In Figure 5 below, we see that teams that have been ranked higher within their division win more. There are 6 divisions, and 5 teams in each division compete to obtain rank 1, which is the rank with the most wins. Hence, rank 1 and 2 winners have more wins than ranks 3 through 5. We also note that the rank 1 and 2 teams are the only

teams that win the World Series, with rank 1 World Series winners having a greater median number of wins than rank 2 World Series winners.

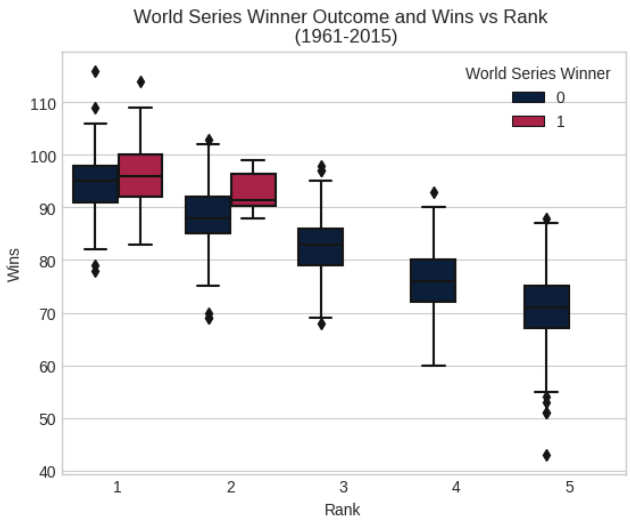


Figure 5: Wins vs Rank and WSW Outcome

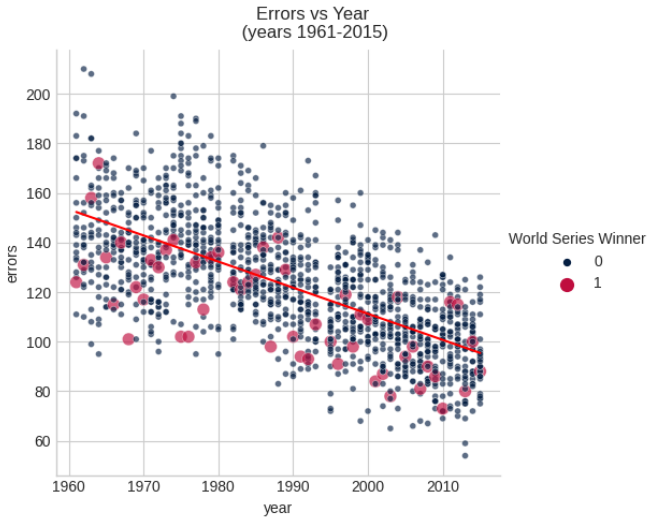


Figure 6: Errors vs Year and WSW Outcome

In baseball statistics it has been said that the errors variable is statistically significant. In Figure 6 above, we deal with the quantitative variable of Errors against the Year, and observe the teams that won the World Series (red).The main trend is clear; the errors of the game are decreasing over the years from 1961 to 2015. This is because playing conditions have become better over the years and players have focused more attention on their fielding ability. Less errors means less people allowed on base, and less runs allowed for the opposing team. When playing conditions improve, like when games are played during the night (regulated temperature) and played on artificial turf, then the number of errors decrease, and it is possible that a team then is less likely to lose.

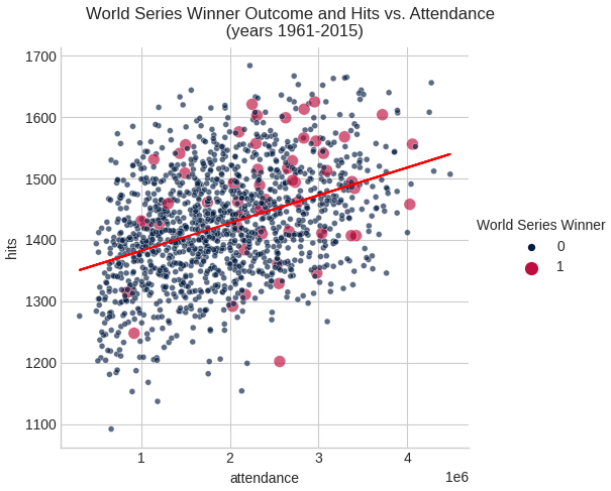


Figure 7: Hits vs Attendance and WSW Outcome

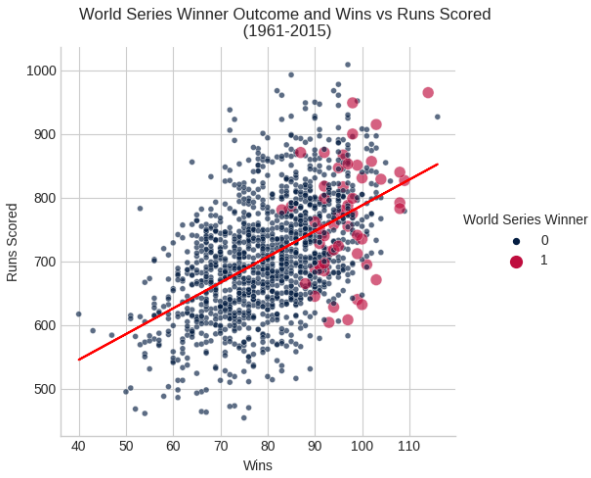


Figure 8: Wins vs Runs Scored and WSW Outcome

We now turn our attention to five important variables in baseball–Hits, Attendance, Wins, Runs Scored, and Opponents Runs Scored–which interact with each other and the World Series winner outcome (WSW). The Hits vs Attendance plot in Figure 7 shows how the number of hits increases with the attendance of the game. This effect can be two-fold; firstly, the more people attend the game, the players may be more motivated to perform, causing greater number of hits by a team's offense. Secondly, the audience may be greater for teams who hit more, since people like to see their team hit the ball more often, an action which is more engaging for an audience to watch. What is interesting here is that a team with a high number of hits does not necessarily mean it is the World Series winner; while the World Series winners do seem to have more hits above 1400, many World Series winners had lower number of hits. This may imply that the variable alone may not be as significant as in combination with others.

In Figure 8, we explore the connection between wins and runs scored. As the number of wins increases, the number of runs scored also increases. Runs are scored when a player runs around all the bases and crosses the plate. A team which has greater wins also tends to have more runs scored. However, while the World Series winners had greater numbers of total games won, around half of the winners had runs scored below 700 and the regression line. So runs scored alone in combination with other variables may be a good predictor of the World Series winner, which lines up with baseball statistics.

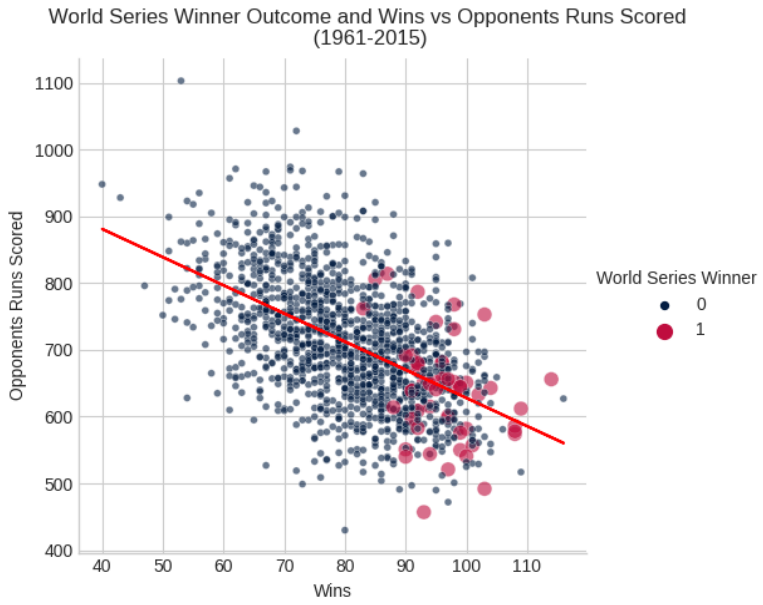


Figure 9: Wins vs Opponents Runs Scored and WSW Outcome

In our last figure above, Figure 9, we look into the association between wins and opponents runs scored. The opponent's runs scored is a measure of how many times a team allowed an opposing team to score a run. For a baseball team to win, they want to minimize this statistic while maximizing their own runs scored. Opponents' runs scored thus has an inverse relationship with wins as opposed to runs scored for this reason. However, the gameplay for each type of statistic is separate, so the correlation between the variables is not of importance. In Figure 9 we observe that as the number of wins increases, the opponents runs scored decreases. The winningest of teams allows fewer runs to be scored on them by their opponents. World Series winners are among those who win the most and allow the fewest opponents scored runs.

Now that we have explored the associations between the features and with the target variable, we will now proceed to model building, analysis, and our results.

V. MODELING, ANALYSIS, AND RESULTS

A. FEATURE SELECTION

Like most optimization processes, finding our optimal feature matrix was an iterative process. Initially, as discussed previously, we removed any attributes that we deemed unnecessary from our knowledge of baseball. Next, we approached this problem from a more statistical perspective by examining the correlation plot and dropping any especially correlated features.

After removing these features from the original 48, we fit all of the following models. After yielding mediocre results, we revisited our feature matrix. By examining the Gini importance from our Random Forest classifier, we found that the majority of the splits among the trees in the random forest were distributed across 13 features: *wins*, *hits*, *hits_allowed*, *runs_scored*, *opponents_runs_scored*, *shutouts*, *attendance*, *rank_1*, *homeruns_allowed*, *saves*, *errors*, *walks_allowed*, and *earned_run_average*. After discovering this, we refit each model using this new feature matrix and saw a universal increase in performance.

B. METRICS

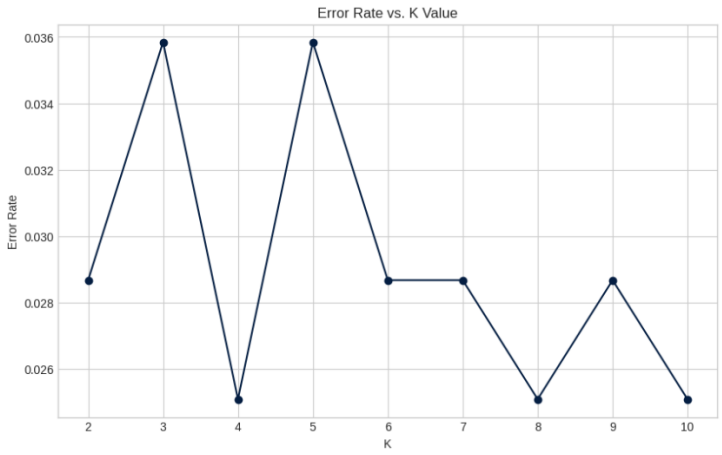
To address the balance issue within our data and effectively be able to determine which of the models we built would best answer our research question, we needed to establish which

metric we would use. Intuitively, accuracy and misclassification rate would be very misleading. Since the data consists largely of World Series losers, even a model that predicts every observation to be a loser will still have a very high accuracy.

For our purposes, we decided to be more tolerant of false positives and focus our efforts more on correctly predicting all of the actual World Series winners. In other words, we are not prioritizing precision. As, in this case, false positives represent teams that have more similarities with most (or all) of the actual World Series winners than they do with the other World Series losers. Practically, knowing which teams performed more similarly to winners might be insightful and interesting to industry professionals. That being said, a balance is ideal (high precision, high recall, and high F1). In order to compare the following methods, though, a consideration of primarily recall and secondarily F1 (of the winner class) will be used.

C. K-NEAREST NEIGHBORS CLASSIFIER

Of the five classifiers we chose to model our data, we fit a K-Nearest Neighbors model first. While not specifically designed for unbalanced data, the nature of KNN is unaffected by the size of each class as–unlike the other models–it doesn't have to learn or transform the data. When fitting the classifier, we decided to leave many of the hyperparameters as their default. Most notably, we left the weights hyperparameter so that all points within each neighborhood are weighted equally. We also left the metric to compute the distance as Minkowski distance (where p = 2) or L2/Euclidean distance. Additionally, we found that the default number of neighbors (5), was not suitable for our data. As seen in the plot below, when K = 4, 8, and 10 the classifier yields the lowest error.



While this plot served as inspiration, we acknowledged that the accuracy and error rate aren’t always the most telling in data as unbalanced as ours. We also acknowledged that

because of the number of observations in our validation and testing sets, it would make the most sense to keep K relatively small. This is because a large K would only work well if the 2 classes are very well-separated and all of the World Series winners sit very close together in the feature space. Additionally, there also has to be at least K/2 + 1 total winners in each dataset otherwise it would not be possible for any observation to be truly classified as a winner. Thus, we looked at the confusion matrices and classification reports of several models using other reasonably small K values and found that K = 4 would be the best.

As for the performance of our KNN classifier, it was not able to predict many (or any) true winners in any of the three datasets. As evidenced in the chart below, not only was it incapable of predicting true winners, it was rather unsuccessful in predicting any winners (true or not). Numerically, the recall and F1 scores are unsatisfactory. Therefore, we concluded this isn't the most helpful model in terms of predicting World Series winners. That being said, it was still informative; it gave us a little bit more of a clue as to how the observations exist in the 13-dimensional space. In other words, it confirmed our suspicion that the 2 classes aren't well-separated as most of the actual winners are closer to mostly loser observations.

		TRAINING DATA		VALIDATION DATA		TEST DATA	
KNN CONFUSION MATRICES		**Predicted**		**Predicted**		**Predicted**	
		Loser	**Winner**	**Loser**	**Winner**	**Loser**	**Winner**
Actual	**Loser**	1065	2	271	1	176	0
	Winner	40	6	6	1	6	0

Winner Class Precision	0.75	0.50	0.00
Winner Class Recall	0.13	0.14	0.00
Winner Class F1 Score	0.22	0.22	0.00

D. COMPLEMENT NAIVE BAYES CLASSIFIER

For our second method, we decided to call upon the illustrious Naive Bayes classifier. Of the five different types of Naive Bayes classifiers within Python's sklearn library, there exists an adaption of Multinomial Naive Bayes specifically for imbalanced data: Complement Naive Bayes. Procedurally, the Complement Naive Bayes algorithm calculates the probability of an observation *not* belonging to the class for each class. After calculating this probability for both classes, it selects the *minimum* of the 2 values. Logically, as we want to select the class that the observation most likely belongs to, we should select the class that is associated with the *lowest* probability of it *not* belonging to that particular class.

For our Complement Naive Bayes classifier, we employed GridSearchCV() and a dictionary of varying values for many of the hyperparameters to determine the most promising iteration. The three hyperparameters we tuned were alpha, fit_prior, and norm. As the weights used in the Complement Naive Bayes algorithm require dividing one count by a different count (to put it very simply), it is possible that the weight could be undefined. To combat this, alpha is employed as a smoothing factor. In our hyperparameter tuning process, GridSearchCV() informed us that alpha = 25 would be optimal. Additional hyperparameters found to be ideal were fit_prior = True (the default) and norm = False (the default). These indicate that the classifier should learn the class prior probabilities while training and that no second normalization of the weights is needed.

		TRAINING DATA		**VALIDATION DATA**		**TEST DATA**	
COMPLEMENT NAIVE BAYES CONFUSION MATRICES		**Predicted**		**Predicted**		**Predicted**	
		Loser	**Winner**	**Loser**	**Winner**	**Loser**	**Winner**
Actual	**Loser**	918	149	230	42	143	31
	Winner	6	40	0	7	1	5

Winner Class Precision	0.21	0.14	0.14
Winner Class Recall	0.87	1.00	0.83

Winner Class F1 Score	0.34	0.25	0.24

Immediately, we can note an improvement in performance across all three datasets in comparison to KNN. Not only is there an increase in F1 scores, the recall values are much closer to what we're looking for. This means that the Complement Naive Bayes classifier was able to correctly identify almost every single one of the actual winners in data that it was not trained on (i.e. the validation and testing datasets).

E. SUPPORT VECTOR MACHINE WITH LINEAR KERNEL

For the first of our 2 Support Vector Machine models, we fit an SVM with a linear kernel. Of the kernel functions, the linear kernel is perhaps the simplest as it is only a dot product between a pair of observations after they've been transformed into higher dimensional vectors ($\langle x, x' \rangle$). This transformation aims to take linearly-inseparable data and transform it so that a single hyperplane can separate the classes.

For the hyperparameters of the SVM, we directly passed our kernel (kernel = 'linear') and the class weights (class_weight = {0: 1, 1: 30.0}). The class weights served to lessen the impact of the class imbalance. We used a 1 to 30 ratio as there are roughly 30 losers for each winner in our data. To tune our remaining hyperparameters for this method, we also used GridSearchCV(). We tested varying values of C (the regularization parameters) and gamma (the kernel coefficient). We quickly discovered gamma is insignificant when a linear kernel is used, so GridSearchCV() instructed us to use gamma = 1. Additionally, we also found that this method performed better with a rather small C value. This means that the SVM's cost for misclassified observations is low (as seen primarily in the false positives of each dataset).

		TRAINING DATA		**VALIDATION DATA**		**TEST DATA**	
SVM LINEAR CONFUSION MATRICES		**Predicted**		**Predicted**		**Predicted**	
		Loser	**Winner**	**Loser**	**Winner**	**Loser**	**Winner**
Actual	**Loser**	918	149	230	42	143	31
	Winner	6	40	0	7	1	5

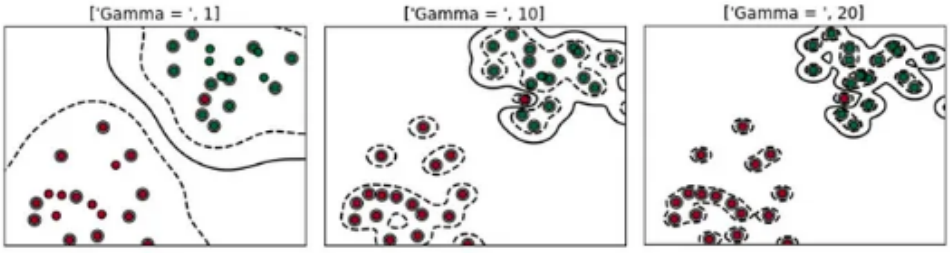
Winner Class Precision	0.21	0.14	0.14
Winner Class Recall	0.87	1.00	0.83
Winner Class F1 Score	0.34	0.25	0.24

If you recall the confusion matrices and classification reports for our Complement Naive Bayes model, you can see that this SVM performed identically on each of the datasets. As stated for the Complement Naive Bayes evaluation, the recall and F1 score are up to the mark. This is especially the case when compared to the results yielded by our best KNN model.

F. SUPPORT VECTOR MACHINE WITH RADIAL BASIS FUNCTION KERNEL

For our fourth model, we decided to test another Support Vector Machine. While we established that the linear kernel function did well when separating the 2 classes, we sought to outperform it using a different kernel. Instead of using a linear kernel, we chose to use a more complex kernel: Gaussian radial basis function. As the name suggests, the form of this kernel resembles that of a Gaussian probability density function (RBF kernel = $\exp(-\gamma||x - x'||^2)$).

By continuing to utilize GridSearchCV(), we determined that gamma = 1 is still our optimal gamma value. Like the previous SVM, we set the class weights equal to the 1:30 ratio to help the classifier handle the class imbalance. Unlike the previous SVM, gamma is quite meaningful for an SVM with an RBF kernel. Essentially, a gamma value of 1 indicates that there is little influence on the decision boundary when a new observation is added. Visually, this means that the decision boundary and tolerance don't hug each of the points tightly as seen in the image below.



Additionally, we found that a larger C value (C = 10) would be optimal. The difference between the previous SVM's C value (1) and this SVM's C value is evident in the corresponding

confusion matrices. As a larger C value increases the cost of misclassified data, we see much fewer false positives and false negatives. This is why there is an increase in each datasets' F1 score.

Therefore, this model is overall good. It is also the most balanced of the models in terms of being able to predict both winners and losers without misclassifying many observations. This is well reflected in the precision, recall, and F1 scores for the winner class. But–as stated before–we care to maximize the recall or correctly identify all of the actual World Series winners. And since it cannot predict as many or more actual World Series winners as Complement Naive Bayes and other SVM models do, we decided to rank this model below them.

		TRAINING DATA		VALIDATION DATA		TEST DATA	
SVM RBF CONFUSION MATRICES		**Predicted**		**Predicted**		**Predicted**	
		Loser	**Winner**	**Loser**	**Winner**	**Loser**	**Winner**
Actual	**Loser**	997	70	247	25	165	9
	Winner	0	46	2	5	4	2

Winner Class Precision	0.40	0.17	0.18
Winner Class Recall	1.00	0.71	0.33
Winner Class F1 Score	0.57	0.27	0.24

G. RANDOM FOREST CLASSIFIER

For our fifth and final model, we chose to fit a Random Forest classifier as we were made aware that it is something used often for MLB classification problems. Typically with Random Forest classifiers, it is difficult to overfit the data. Like several of the other models, though, the class imbalance was quite the obstacle for this classifier.

Like the previous 2 methods, we made sure to pass the same class weights (class_weight = {0: 1, 1: 30.0}) to put more importance on the winner class. After many brute force attempts to alter the default hyperparameters, we found that no matter whether we changed the maximum depth or minimum number of samples needed to split a node, we still saw abysmal performance on the validation dataset. As anything we attempted to tune had little to no effect on the performance, we used the default hyperparameters.

Intuitively, using the default hyperparameters (notably max_depth = None and min_samples_split = 2) implies that each tree can go as deep as it needs to in order to achieve leaf purity. This means that it's likely that every leaf that was categorized as a World Series winner might just be a single observation split from losers by a niche or overall unimportant feature. Before even examining the classification reports or confusion matrices, we know we will see some severe overfitting. In other words, we should expect great training performance and abysmal validation and testing performance.

.

		TRAINING DATA		VALIDATION DATA		TEST DATA	
RF CONFUSION MATRICES		**Predicted**		**Predicted**		**Predicted**	
		Loser	**Winner**	**Loser**	**Winner**	**Loser**	**Winner**
Actual	**Loser**	1067	0	272	0	173	1
	Winner	0	46	7	0	5	1

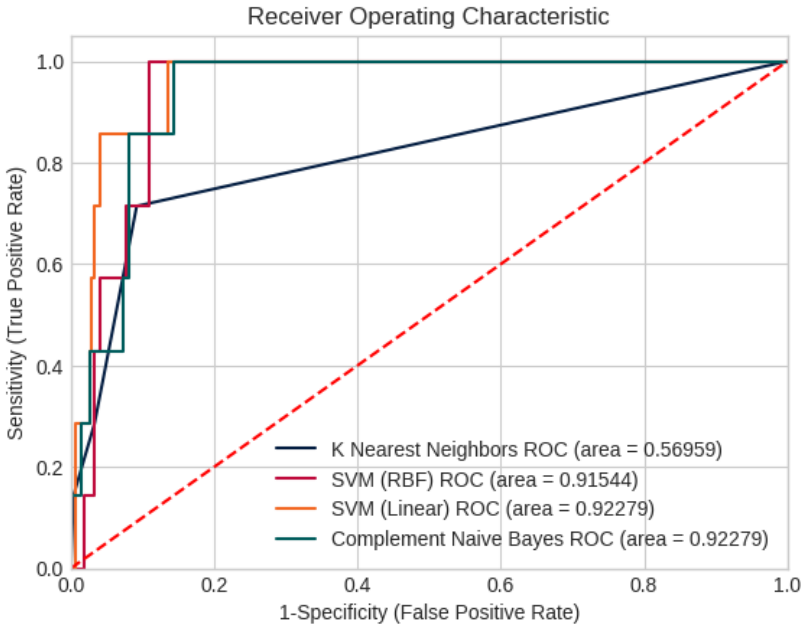
Winner Class Precision	1.00	0.00	0.50
Winner Class Recall	1.00	0.00	0.17
Winner Class F1 Score	1.00	0.00	0.25

As you can see in the above chart, our hypothesis was confirmed: even the best Random Forest model is severely overfit and is not generalizable in the least. We can immediately note a record low recall and F1 score for the validation dataset. At first glance, the F1 score for the testing set is rather high. After examining this a little closer, we can see that the

F1 score is brought up largely because of the precision (i.e. of the predicted winners, half of them are correct). Overall, our Random Forest was equally poor at predicting World Series winners as the KNN model. Upon further research, our findings were validated as we discovered that Random Forest classifiers often perform poorly on imbalanced data.

VI. DISCUSSION

Lastly, to compare the five above models, we will expand on the recall and F1 score and compare the models using their area under the curve (AUC). To visualize the relative performance of the models, we have plotted the ROC curve below. We chose to exclude Random Forest from this plot as the area under its curve is equal to 0.5. This means that it essentially has no predicting power and would just overlap with the red dashed baseline. The ROC curve also validates our prior conclusion that the SVM with a linear kernel, the Complement Naive Bayes, and the SVM with a RBF kernel outperform both KNN and Random Forest by a landslide.



To conclude, baseball is a sport with a long and vibrant history. Millions of fans watch their favorite team every year in hopes of seeing them hoist the World Series trophy at the end of the season. A key aspect that goes into every decision made by teams in today's game is

sports analytics. In our study of season-by-season team data, we drew many conclusions on what attributes play the most into a team's success, and built a number of models based on these attributes to predict World Series winners.

From our EDA, we determined, logically, that WS winners win more games per year and tend to have higher attendance figures than losers. We also found that only first or second ranked teams in their division have won the World Series. Rules were changed in 2022 to add an additional wild card team, meaning the third ranked team in the division now has a chance to win the championship. This is something we could explore in the future when the sample size becomes larger. Teams that score more runs and allow less runs are more likely to win more games and therefore, more likely to win the World Series. One of the most interesting results from the visualizations was errors vs year. The number of errors by a team have trended downward since 1961, indicating that defense as a whole has improved and teams with better defenses are more likely to win the World Series.

From our model building, we found our two best models for prediction were the Complement Naive Bayes classifier and the SVM with a linear kernel. They actually produced identical results, meaning their respective methods lead them to find very similar (or perhaps identical) decision boundaries. Despite predicting all 7 winners correctly on the validation set, the models' F1 score may have suffered from the cost of a somewhat high number of false positives (bringing down the precision). However, they still produced the best testing results. In the article discussed previously in the background section of our report, other researchers also found linear SVM to be the best model, validating our own findings. SVM with an RBF kernel was the next best model, and was perhaps the most balanced model in terms of precision and recall. It produced the highest validation F1 score of any of the models. However, it did not perform as well as the first two in predicting true positives on the testing set. Our other two models, KNN and Random Forest, were not good models for classification on an imbalanced dataset. Due to their construction, both models struggled to predict any winners.

Overall, our results produced an outcome we were satisfied with. Because we found models that performed very well, we could build off of our work and explore other areas of the game that might have hidden value. Something we can consider is using expected statistics from this season to attempt to predict this year's World Series winner. This is something we can look into in the future.

VII. REFERENCES

Chen, L. (2017). Difference in Gamma Values for SVM RBF. Towards Data Science. Retrieved April 30, 2023, from https://towardsdatascience.com/support-vector-machine-simply-explained-fee28eba5496.

ESPN Internet Ventures. (n.d.). *MLB World Series Winners*. ESPN. Retrieved April 5, 2023, from https://www.espn.com/mlb/worldseries/history/winners

Kalist, D. E., & Spurr, S. J. (2006). Baseball errors. Journal of Quantitative Analysis in Sports, 2(4). https://doi.org/10.2202/1559-0410.1043

Major League Baseball - seasons. RetroSeasons. (2022, May 20). Retrieved April 16, 2023, from https://www.retroseasons.com/leagues/mlb/history/seasons/

Tolbert, B., & Trafalis, T. (n.d.). *Predicting Major League Baseball Championship Winners through Data Mining*. Academic journals | Athens Institute for Education & Research . Retrieved April 10, 2023, from https://www.athensjournals.gr/sports/2016-3-4-1-Tolbert.pdf, https://doi.org/10.30958/ajspo.3.4.1

Sklearn.ensemble.randomforestclassifier. scikit. (n.d.). Retrieved May 1, 2023, from https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

Sklearn.naive_bayes.Complementnb. scikit. (n.d.). Retrieved May 1, 2023, from https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.ComplementNB.html

Sklearn.neighbors.kneighborsclassifier. scikit. (n.d.). Retrieved May 1, 2023, from https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html

Sklearn.svm.SVC. scikit. (n.d.). Retrieved May 1, 2023, from https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html

Sportradar. (n.d). *History of all Major League Baseball: National League, American League, Negro league and more*. Baseball Reference. Retrieved May 1, 2023, from https://www.baseball-reference.com/leagues/

Sportradar. (n.d.). *World Series Winners*. Baseball Reference. Retrieved April 10, 2023, from https://www.baseball-reference.com/postseason/world-series.shtml

Sports, O. S. (2019, November 17). *Baseball databank*. Kaggle. Retrieved March 25, 2023, from https://www.kaggle.com/datasets/open-source-sports/baseball-databank?select=Teams.csv

Ziegler, Z. (2021, May 8). *Using machine learning in Python to Predict World Series winners*. Medium. Retrieved April 16, 2023, from https://eeziegle.medium.com/using-machine-learning-in-python-to-predict-world-series-winners-4d04b229bba3