Predicting Baseball Winners from Season Average Data

By Austin Chewning, Cameron Grondines, and Zack Miller

Introduction

Major League Baseball is widely considered America's pastime, and is a statistics heavy sport. In this project, we take advantage of a small fraction of that data to investigate the correlation between the winner of games and average team characteristics. There is a large amount of data that we were able to find, and much of it is free to use. This was greatly helpful in creating the dataset that we use in our analysis. We have two sources of data, Retrosheet and the Lahman Baseball Database. Retrosheet is utilized for individual game data, and the Lahman Baseball Database is used for season average statistics. We combine the two datasets for the years 2006-2016 such that our final dataset on which we will run our learning algorithms includes only the home team season average statistics, the away team season average statistics, and a class attribute of if the home team won the game. We included all of the non-identifier statistics available from the Lahman Baseball Database and used various attribute selection techniques to remove non-predictive attributes. This process will be discussed in the data preparation and data analysis sections. Because all of our attributes are numeric, we have two versions of this dataset. One is nominal where our class attribute is '1' = home win and '0' = home loss. The other is a fully numeric dataset where our interpretation is that the numeric prediction is the percentage chance of a home win. For all of the games in our dataset, we have either a 100% or 0% chance of the home team winning since the outcome of each game is known.

In our data analysis we find a baseline of 54.6% accuracy using ZeroR on our nominal dataset. This will be the number on which any learning scheme has to significantly improve. We find that using our numeric dataset, we have a correlation coefficient of 0.1284 using the simple linear regression algorithm. Upon further analysis, we find that it is difficult to improve upon these values by a large margin. We believe this to be because our dataset is inherently noisy. For example in the 2017 season (not available for this analysis) the New York Yankees faced the World Series Champion Houston Astros at home a total of seven times and won four of those matchups (http://mcubed.net/mlb/nyy/hou.shtml). In our dataset, each of these matchups would be a separate instance with identical season average data but four have the Yankees winning and three have the Astros winning. Further analysis of this and other issues we face is found in the results section.

Dataset Description

Our dataset is sourced from two baseball statistics repositories, Retrosheet and the Lahman Baseball Database. This section describes each dataset and the steps we took to combine them into the dataset used in our analysis.

Retrosheet

Our Retrosheet data includes 26,725 games from the 2006-2016 seasons. (The information used here was obtained free of charge from and is copyrighted by Retrosheet. Interested parties may contact Retrosheet at "www.retrosheet.org") This CSV dataset includes a total of 161 attributes that describe nearly every aspect of a single game. Many of these are not predictive on our project such as the umpiring crew or the attendance of the game. Others such as hits, doubles, triples, home runs, runs batted in, and many others encode the winner of the game too closely for the analysis that we want to perform. In many ways, using the number of hits, for example, to predict the winner in a new matchup would be overfitting. Therefore, we made the decision to overlay season average data with the attribute named Game_HomeWin that we derived from the Retrosheet game data by comparing the home and away team scores. In order to combine the datasets, we cleaned the game logs so that an instance takes the form of the instance shown in Table 1.

Table 1

Year	Game_HomeTeam	Game_AwayTeam	Game_HomeWin
2016	Tex	Hou	0

This instance represents one of the 2016 season games where the Texas Rangers faced the Houston Astros and the Rangers lost at home. We have an instance for each of the 26,725 games played in the 2006-2016 seasons.

Lahman Baseball Database

Our overlay data comes to us courtesy of the Lahman Baseball Database, of which we are using only a small fraction. The full database includes every player for each team for years dating back to the 1870's. We are using a table of season average data for each team over the 2006-2016 seasons that we are investigating. Some of the 48 attributes from the Lahman Baseball Database are identifiers for each team or are links to some of the other data tables included in the database. We are not using those attributes and will discuss further in the data preparation section. The other attributes are batting, pitching, and fielding statistics over the entire season. Table 2 lists all of the attributes in the Lahman Baseball Database along with their description and type.

Table 2

Attribute	Description	Type
yearID	Year	Numeric
lgID	League	Nominal
teamID	Team	Nominal
franchID	Franchise	Nominal
DivID	Team's Division	Nominal
Rank	Position in final standings	Numeric
G	Games played	Numeric
GHome	Games played at home	Numeric
W	Wins	Numeric
L	Losses	Numeric
DivWin	Division Winner	Nominal
WCWin	Wild card Winner	Nominal
LgWin	League champion	Nominal
WSWin	World Series winner	Nominal
R	Runs scored	Numeric
AB	At bats	Numeric
H	Hits by batters	Numeric
2B	Doubles	Numeric
3B	Triplies	Numeric
HR	Home runs by batters	Numeric
BB	Walks by batters	Numeric
SB	Strikeouts by batters	Numeric
CS	Caught stealing	Numeric
HBP	Batters hit by pitch	Numeric
SF	Sacrifice flies	Numeric
RA	Opponents runs scored	Numeric
ER	Earned runs allowed	Numeric
ERA	Earned runs allowed	Numeric

CG	Complete games	Numeric
SHO	Shutouts	Numeric
SV	Saves	Numeric
IPOuts	Outs Pitched (innings pitched x3)	Numeric
HA	Hits allowed	Numeric
HRA	Homeruns allowed	Numeric
BBA	Walks allowed	Numeric
SOA	Strikeouts by pitchers	Numeric
E	Errors	Numeric
DP	Double plays	Numeric
FP	Fielding percentage	Numeric
name	Team's full name	Nominal
park	Name of team's home ballpark	Nominal
attendance	Home attendance total	Numeric
BPF	Three-year park factor for batters	Numeric
PPF	Three-year park factor for pitchers	Numeric
teamIDBR	Team ID used by Baseball Reference website	Nominal
TeamIDlahman45	Team ID used in Lahman database version 4.5	Nominal
teamIDretro	Team ID used by Retrosheet	Nominal

Overlaid Dataset

With much help from Dr. Treu, we used a SQL query to combine the Retrosheet game data with the season average statistics for the home and away team. The game log that looks like the instance in Table 1 includes three important pieces of information that we used to match each game with the correct season average statistics. The HomeTeam and AwayTeam attributes are included in the Lahman Baseball Database under the attribute teamIDretro. So we use the SQL query,

```
select * from (2016Games INNER JOIN TeamAvg on
2016Games.HomeTeam=TeamAvg.teamIDretro and yearID=2016) AS XYZ
INNER JOIN TeamAvg on XYZ,
```

to create a new instance that has (in order) the home team's season average statistics, the away team's season average statistics, and the information shown in Table 1 for each game. This new dataset has 100 attributes with a large number of identification attributes. These are removed so that we only have the numeric statistics for each team and the class attribute in its numeric form. Attributes removed include: yearID, leagueID, teamIDs, franchiseID, divisionWin, wcwWin, leagueWin, worldSeriesWin, ballpark information, year, Game_HomeTeam, and Game_AwayTeam. We are left with a total of 65 attributes once these are removed. We utilized Weka's CSV converter to create an ARFF file from our CSV.

Data Preparation

Our data preparation was fairly simple because we were dealing with all numeric attributes, no missing values, and the summary of each attribute in Weka demonstrated that we did not have any egregious outliers. As discussed in the previous section, we removed the unique identifiers in our dataset. Our data sources are reputable in the baseball statistics realm, so we are considering their data to be correct as given. On our fully numeric dataset, we normalized each attribute on a 0-1 scale. This is the base that we work on for any further analysis. Therefore, we split our data into test and training datasets. Since we have such a large amount of data, we decided to use a 90/10 split of our data where a randomized 90% is saved as the training data, and the remaining 10% is reserved as test data. We performed this split using Weka's randomize filter on the full dataset and then the remove percentage filter. At this point, we created a nominal version of both training and test datasets by changing the GameHomeWin attribute from numeric to having possibilities '1' and '0' where '1' = HomeWin and '0' = HomeLoss in a third and fourth ARFF file.

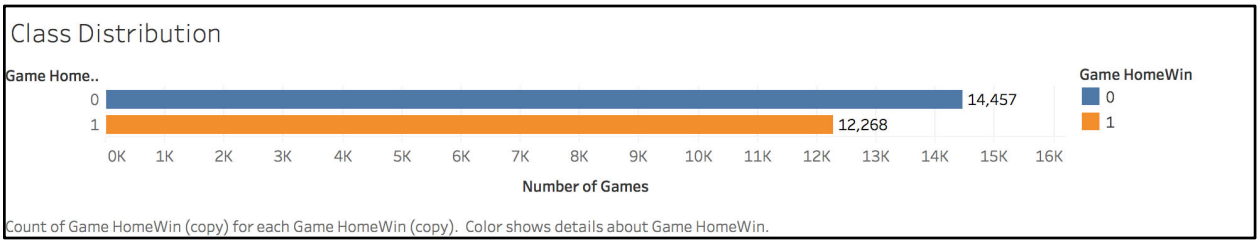
Data Analysis and Results

In our data analysis, we look at both the nominal and the numeric datasets in order to compare the results we get from each. Weka's attribute selection meta-learners are used in an attempt to find a more predictive set of attributes.

Nominal Dataset

We began our analysis on the nominal dataset with ZeroR which picks the class attribute value that occurs the most often and uses that value to predict all new instances. In our case it would be whether the home team won or lost. Using this we were able to get a baseline accuracy of 54.6% which we compare to our new models to see if we obtainin a higher accuracy. This result is shown in the distribution of the two classes shown in Figure 1. This also confirms for us that our data is well balanced between home win and home loss class attributes.

Figure 1



The next algorithm that we ran was OneR which creates a rule based on one attribute for predicting the class attribute value of new instances. We ran this algorithm with four different bucket sizes to see how our accuracy would change which can be seen in the Table 3 below. The best accuracy we obtained from running OneR was 57.4% using a minimum bucket size of 500.

Table 3

Bucket Size	**Accuracy**
10	56.6%
100	56.8%
500	57.4%
1000	56.9%

The next algorithm that we ran was J48 which is a tree building algorithm that builds a decision tree by splitting on the most predictive attributes at each level. Using this algorithm we obtained an accuracy of 56.4%.

Next we ran a Sequential Minimal Optimization (SMO) algorithm which breaks down a large quadratic programing problem into smaller quadratic programing problems. These problems are then solved analytically making the memory required linear with respect to the training set size allowing the use of much larger training sets such as ours. The goal of this algorithm is to find a hyperplane that divides the data into two classes, perfect for our application. This algorithm produces a model that gives us a 58.2% accuracy. This model is shown in the appendix and the confusion matrix below in Table 4.

Table 4
SMO Confusion Matrix

	Predicted Home Team Loss	Predicted Home Team Win
Home Team Loss	1056	403
Home Team Win	713	500

We use two Naive Bayes algorithms which predict the probability of a new instance based on probabilities obtained from training instances while assuming all attributes are independent. Using the normal Naive Bayes algorithm, we obtained a 56.7% accuracy when evaluating the model. We then ran the Naive Bayes Multinomial algorithm which runs the naive bayes algorithm with the assumption that distributions are multinomial and not some other distribution. With this algorithm we got an accuracy of 58.1%. The entire model is shown in the appendix and the confusion matrix in Table 5.

Table 5
Multinomial Naive Bayes Confusion Matrix

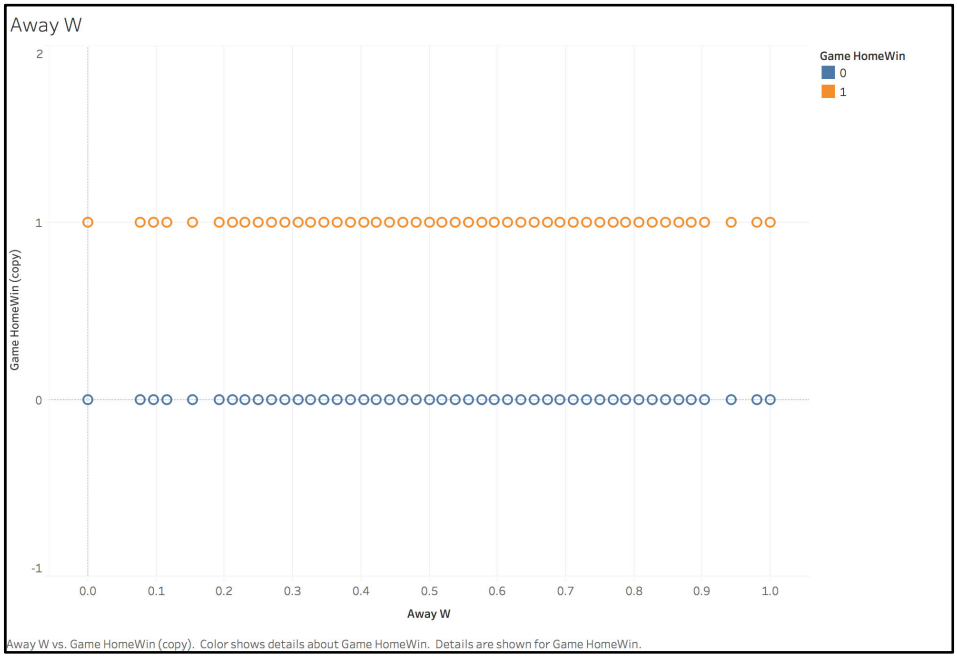
	Predicted Home Team Loss	Predicted Home Team Win
Home Team Loss	1153	306
Home Team Win	813	400

In an effort to improve these results we used the meta-learner Attribute Selected Classifier and ran OneR and the Naive Bayes algorithms, and the SMO algorithms again to see what could be learned. These algorithms were chosen because they gave the best accuracies thus far in our analysis. We used the Correlation-based Feature Subset (CFS) selection which evaluates the worth of a subset of attributes by considering the individual predictive ability of each feature along with the degree of redundancy between them to select attributes. When we ran OneR with a minimum bucket size of 500 (based on our testing shown in Table 3) using the CFS attribute selection we obtained the same accuracy of 57.4%. Using the Correlation Attribute Evaluator for attribute selection and the OneR Attribute Evaluator gave an accuracy of 56.9%. We then ran Naive Bayes using the CFS evaluator to remove attributes and got a new accuracy of 56.3%. The Naive Bayes Multinomial algorithm, also using the CFS evaluator to remove attributes and got a new accuracy of 57.6%. Lastly, the SMO algorithm in conjunction with the CFS evaluator gives an accuracy of 58.1%.

Numeric Dataset

We begin our analysis on the numeric dataset using the Simple Linear Regression algorithm which forms a baseline prediction by producing a model that is a function of one input attribute. This model gave us a correlation coefficient of 0.1284. We then ran the linear regression algorithm which produces a model that is a function of multiple input attributes. This model gave another disappointing correlation coefficient of 0.1526. Since our model performed so poorly, we plotted the relationship of the most predictive attribute from OneR, Away Wins, and the results are shown in Figure 2. We can immediately see that there is not a linear relationship between our data and the change of the home team winning. Similar analysis with other attributes shows the same pattern, no linear correlation. Therefore, we decided to discontinue our numeric estimation.

Figure 2



Results

We find that our data is unable to reliably predict the winner of a baseball game, given the season average statistics of the teams playing. However, there are some interesting takeaways from our analysis. While it is not a particularly strong predictor, it is interesting that a team's ability to win games on the road is most closely tied to the winner of a game. This is shown in our results using OneR. In the most predictive OneR model, the rule generated was:

if Away_W<0.4519229 then Game_HomeWin=1, if Away_W>0.4519229 then Game_HomeWin=0.

This is also notable because our data is skewed slightly towards the home team losing.

The SMO algorithm produces nearly the same accuracy of the Multinomial Naive-Bayes algorithm. However, it more evenly distributes the false positives and false negatives compared to the Multinomial Naive-Bayes. This makes some intuitive sense because it is built as an algorithm to classify binary data like ours. In general, false positives and false negatives are equally undesired in our models so this does not make the SMO a better algorithm, necessarily.

In general, our model suffers greatly from the noise demonstrated by Figure 2. Since team's face each other more than once over the course of a season, while the season average statistics remain the same for each matchup, the winner of the game has a high potential to be split somewhat evenly between the two teams. A possible way to mitigate this issue that was not tested would be to compile the results of their games so that each instance is the season average statistics for each team and the class attribute is either the team that is more likely to win a new game or the percentage chance of one of the teams winning.

Conclusion

The main purpose of this project was to see if we could predict the winner of a Major League Baseball game given Retrosheet's game data and Lahman's Baseball Database season average statistics. In the preprocessing stage we combined these two datasets to create a dataset that lined up each game being played with the two teams seasonal averages. We then created two versions of this dataset, one with nominal values and one with numeric values. With this newly created datasets we ran both classification and numeric estimation algorithms on the nominal and numeric datasets, respectively.

Overall, many of the classification algorithms only beat a coin flip and our ZeroR baseline by a few digits and the numeric estimation algorithms produced poor correlation coefficients. The poor correlation coefficients along with data visualization lead us to realize that our data is nonlinear. The best model we were able to create was with using the Sequential Minimal Optimization algorithm. With this model it produced, we get an accuracy of 58.2%. We discovered that baseball is a very tough sport to try to predict, mainly because there is no one attribute that will determine whether a team will win or not out of the season average data. After conducting these tests with the dataset we created we hypothesize that better results may be yielded by utilizing the averages of individual players rather than the season averages of the teams. For example, the

individual matchups of pitchers and batters may have a larger predictive value than the average tendencies of a team’s bullpen and their opponent’s hitters.

Appendix

Sequential Minimal Optimization Model

-0.0553 * (normalized) Home_Rank
+ -0.2836 * (normalized) Home_G
+ 0.1629 * (normalized) Home_GHome_
+ 0.6159 * (normalized) Home_W
+ -0.6247 * (normalized) Home_L
+ 0.2654 * (normalized) Home_R
+ 1.1674 * (normalized) Home_AB
+ -0.5241 * (normalized) Home_H
+ 0.0377 * (normalized) Home_2B
+ -0.0991 * (normalized) Home_3B
+ -0.2096 * (normalized) Home_HR
+ 0.2174 * (normalized) Home_BB
+ 0.0048 * (normalized) Home_SO
+ 0.2784 * (normalized) Home_SB
+ 0.0924 * (normalized) Home_CS
+ 0.1078 * (normalized) Home_HBP
+ -0.0331 * (normalized) Home_SF
+ -0.536 * (normalized) Home_RA
+ 0.1547 * (normalized) Home_ER
+ 0.3057 * (normalized) Home_ERA
+ 0.0999 * (normalized) Home_CG
+ 0.1326 * (normalized) Home_SHO
+ 0.6803 * (normalized) Home_SV
+ -0.1059 * (normalized) IHome_Pouts
+ -0.2017 * (normalized) Home_HA
+ 0.2707 * (normalized) Home_HRA
+ -0.1628 * (normalized) Home_BBA
+ -0.0246 * (normalized) Home_SOA
+ 0.3492 * (normalized) Home_E
+ 0.0528 * (normalized) Home_DP
+ 0.1797 * (normalized) Home_FP
+ -0.8344 * (normalized) Home_BPF
+ 0.6528 * (normalized) Home_PPF
+ 0.2831 * (normalized) Away_Rank
+ -0.3254 * (normalized) Away_G
+ 0.2235 * (normalized) Away_Ghome
+ -1.4518 * (normalized) Away_W
+ 1.3271 * (normalized) Away_L
+ 0.3848 * (normalized) Away_R
+ 0.3407 * (normalized) Away_AB
+ -0.0544 * (normalized) Away_H
+ 0.2154 * (normalized) Away_2B
+ 0.0535 * (normalized) Away_3B
+ -0.3536 * (normalized) Away_HR
+ 0.1559 * (normalized) Away_BB
+ 0.2932 * (normalized) Away_SO
+ 0.1127 * (normalized) Away_SB
+ -0.0375 * (normalized) Away_CS
+ 0.1323 * (normalized) Away_HBP

+ -0.2313 * (normalized) Away_SF
+ 0.7364 * (normalized) Away_RA
+ -0.1411 * (normalized) Away_ER
+ -0.4361 * (normalized) Away_ERA
+ 0.1085 * (normalized) Away_CG
+ 0.1304 * (normalized) Away_SHO
+ 0.6559 * (normalized) Away_SV
+ 0.3111 * (normalized) Away_IPouts
+ -0.1002 * (normalized) Away_HA
+ 0.0014 * (normalized) Away_HRA
+ -0.3474 * (normalized) Away_BBA
+ 0.2155 * (normalized) Away_SOA
+ -0.4612 * (normalized) Away_E
+ 0.0575 * (normalized) Away_DP
+ -0.4312 * (normalized) Away_FP
- 1.5957

Multinomial Naive Bayes Model

The independent probability of a class

---

0 0.5403866140095615
1 0.4596133859904386

The probability of a word given the class

	0	1
Home_Rank	0.015233970738079026	0.012835093007382501
Home_G	0.017016208968340118	0.017069497917592498
Home_GHome_	0.01752840847205262	0.017541399604571562
Home_W	0.019428433739428985	0.021344358182958
Home_L	0.016081611269661292	0.014171563051059254
Home_R	0.01563610016635965	0.01637731570895863
Home_AB	0.01818407735794917	0.018585766634534092
Home_H	0.01769262365911398	0.01820972282437688
Home_2B	0.014667544047257964	0.014982550633538788
Home_3B	0.016146461605691213	0.016112372206472502
Home_HR	0.014952179460788197	0.015527945923255812
Home_BB	0.014817747638778135	0.01542595365689034
Home_SO	0.016612000126860062	0.016431669356611358
Home_SB	0.0147191578990608	0.01481499696631549
Home_CS	0.013360444022987254	0.01318454539343982
Home_HBP	0.012673702078663458	0.013064507138839745
Home_SF	0.013997737410515184	0.014396447802445544
Home_RA	0.015843167426666555	0.014677356730627833
Home_ER	0.01603157157138938	0.01491090470265902
Home_ERA	0.015877276255227144	0.014717376509314007
Home_CG	0.008386218669748088	0.008669189287265438
Home_SHO	0.013035834975396158	0.013925372251836897
Home_SV	0.01388284958642702	0.015023226762463888

Home_IPouts	0.01448470566642443	0.015387188707952384
Home_HA	0.020968116853702954	0.020091521637095863
Home_HRA	0.014695463691056274	0.014217883296938924
Home_BBA	0.0164634109758948	0.015665939395701704
Home_SOA	0.015098378140089746	0.01589188380652O107
Home_E	0.017579545331653172	0.01697896652492399
Home_DP	0.019143583494232237	0.018754779676301812
Home_FP	0.016929059840010183	0.017529141992237852
Home_BPF	0.013312765674043244	0.013380580487959199
Home_PPF	0.012990089527116364	0.012873796993601833
Away_Rank	0.013030249939094673	0.015513334251904261
Away_G	0.017131183353261328	0.01697575311845464
Away_Ghome	0.01752570320052326	0.01750750316943189
Away_W	0.021174483560203076	0.019213283569499667
Away_L	0.01437207836589183	0.01626021075772416
Away_R	0.016352047784908508	0.015481737881955848
Away_AB	0.01837250035576572	0.018298097822908546
Away_H	0.018142268126721997	0.017611700710696714
Away_2B	0.014932816697665796	0.014631698116473234
Away_3B	0.01608965082782689	0.016173727955062993
Away_HR	0.015538769085049916	0.01477465632301555
Away_BB	0.015454108589455038	0.014636060475090007
Away_SO	0.016384610532625115	0.01673401989714239
Away_SB	0.014809538229630605	0.01477012155341337
Away_CS	0.013236400558678753	0.013372503512332662
Away_HBP	0.013015270088800232	0.012678880102934564
Away_SF	0.014388086739160009	0.013894621717377742
Away_RA	0.014855079307559067	0.015888058894781364
Away_ER	0.015082937531976422	0.01606394057984834
Away_ERA	0.01491061285527148	0.015893651261429866
Away_CG	0.008661405842716748	0.008344701487598012
Away_SHO	0.01377965845549748	0.013035439177347381
Away_SV	0.014786606304503704	0.013908806064979205
Away_IPouts	0.015092517075990537	0.014632781409127875
Away_HA	0.020221593850791378	0.021002892437497478
Away_HRA	0.014266276136922695	0.014731939824237447
Away_BBA	0.015831067552155343	0.016443380306476442
Away_SOA	0.01567914278056678	0.015151616042105001
Away_E	0.01707891670395409	0.017625700209442863
Away_DP	0.01887467856764529	0.019120192143012027
Away_FP	0.017459294658586203	0.0168581744540943

Time taken to build model: 0.05 seconds
=== Evaluation on test set ===
Time taken to test model on supplied test set: 0.1 seconds

=== Summary ===

Correctly Classified Instances	1553	58.1213 %
Incorrectly Classified Instances	1119	41.8787 %
Kappa statistic	0.1244	
Mean absolute error	0.4863	
Root mean squared error	0.4919	
Relative absolute error	97.9852 %	
Root relative squared error	98.7877 %	

Total Number of Instances 2672

=== Detailed Accuracy By Class ===

	TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Class
	0.790	0.670	0.586	0.790	0.673	0.136	0.593	0.625	0
	0.330	0.210	0.567	0.330	0.417	0.136	0.593	0.527	1
Weighted Avg.	0.581	0.461	0.577	0.581	0.557	0.136	0.593	0.580	

References

Lahman's Dataset:
http://www.seanlahman.com/baseball-archive/statistics/

Retrosheet Dataset:
http://www.retrosheet.org/gamelogs/index.html

Yankees-Astros Match-ups
http://mcubed.net/mlb/nyy/hou.shtml