Phase 3 Project-George Ogden

Folder Explanations

Code Folder

The code folder contains the final notebook ('Final') which is an edited/cleaned up composite of the other notebooks in the folder. Particularly, Data_Update_I, _II, & _III.

The other notebooks were either not used for the final results or include experimenting/random code.

Data Folders

The 'Raw Data' folder contains excel, csv and txt files that were downloaded directly from baseball reference except for the odds_data file which came from my other data source.

'Base DFs' contains pickled dataframes built from webscrapped data or the afformentioned downloaded data. This covers all the team annual statistics, the various dataframes covering game information, individual types of team statistics (annual), and some dataframes that were built as part of my original feature data.

Similarly, 'Merged DFs' contains dataframes that were composites of the dataframes in 'Base DFs'. This includes the combined game information, all the team annual statistics and the final database--before any transformations and splitting.

The 'Primary Data' folder contains dicts and dataframes that have been saved out so that the final notebook can be run while skipping certain cells that take a long time to execute (e.g. some scrapping code takes awhile).

Other Folders

The image folder contains images used in the final presentaiton.

The glossaries folder contains word docs with definitions of each team statistic grabbed from baseball-reference. This includes those that were ultimately excluded as features.

Introduction to Over/Under Lines

Besides picking Winners & Losers, the most common form of sports gambling is to predict whether a the total number of runs (or points) scored will or will not go over a certain level.

As with other wager types, a sportsbook will set a 'line' that, based on their own beliefs, makes both outcomes equally likely. This line will come with 'odds' that will insure the book makes a profit, assuming both sides of the bet see even amounts bets are made. In order to keep both sides even, a book will adjust the line or the odds to entice gamblers to bet on the less-favored outcome. This iterative process continues until a closing line is set, usually when the game begins.

Most gambling research focuses on finding a better prediction for the total runs. While much of the inputs will be similiar, this project instead looks to predict when the book's estimation is too low (i.e. when the OVER bet wins).

Project Motivating Idea

I assume that sportsbooks are sophisticated participants and therefore use sophisticated models to predict the expected runs scored for each game. Of course, this makes our goal of predicting (downside) bias particularly difficult. We need to identify information that is availble before a game begins that creates ***upside uncertainty***.

What aspects of a baseball game should be considered? Obviously, we should consider the qualities of the two participating teams--both pitching & hitting. With pitching, it would make sense to differentiate between starting and relief pitching. We may also consider each team's fielding. Additionally, there may be some meta information about the game--such as how many games have been played in the given season--which hinders a sportsbook's prediction accuracy.

Ideally, I would incorporate each individual starter (pitcher & batting lineup) and available bench players (including relief pitchers) before every game. However, for the purposes of this project, collecting all this data for field players and relief pitchers is too large of a task. Therefore, I will use team-level data (for the time being) to approximate these features. Furthermore, the model should use somE data from prior season(s) but increasingly consider what has happended to date in the current season. Once again, this requires a computational complexity that, at least for a first pass, is beyond the scope of this project. (But more on this in the addendum.) Instead, the first model is trained with features captured from the season prior to the observation of interest (i.e. games in 2015, refer to statistics from the 2014 full season).

Data

Sources

Finding historical data for MLB games was surprisingly difficult--from what I found no legitimate sportsbook publishes such data. However, through other research citations, I was able to find a website, Sportsbook Review Online , which provides such data for all sports, including MLB going back to 2010. For all other data, I relied on Baseball-Reference which is widely-recognized as the most reliable and granular resource for all things MLB.

Collection Method

The gambling data was easily retrieved by downloading excel files directly from the website. These files, of course, included Over/Under opening & closing lines and odds along with other gambling data. They also provided data about the matchup--Home & Away team and pitchers--and the runs scored by each team.

Data from Baseball-Reference was a bit more difficult to retrieve. The annual year-end team statistics were easy to download directly from the website. However, any game specific-data required web scrapping (Although, I did later discover a not-widely-advertised API which had cummulative batting and pitching data for players and teams at any given time of the season.) Fortunately, the website follows strict templates so once one type of page was succesfully scrapped, it was just a matter of iterating to collect the rest.

Dataset & Feature Engineering

The initial odds data had 12,144 observations (2 rows per game in the date range eventually consolidated). Some (4 games) of the over/under observations had to be manipulated by using the opening as the closing line. I also had to discard 31 games because the odds data did not have the correct score and therefore I was not confident in relying upon the over/under info for said observations.

In the final dataset, I ended up not using much of the game-level info (such as the Home PLate umpire and day/night game indicator) that was collected. I also ended up dropping a number of features from the various team statistic datasets because they were a function of other features (slugging % is a function of the various types of hits); were used to engineer new features (e.g. an offense's succesful and unsuccesful steals were combined because I posited that attempting steals was a good way to capture volatility in a game and that teams which attempted often were also likely to be more succesful); had strong correlation numbers to one or more other features (see below); or added no marginal information to whether the over would hit (e.g. the # of wins a pitcher has is unlikely to have any information on runs scored that is not already captured elsewhere).

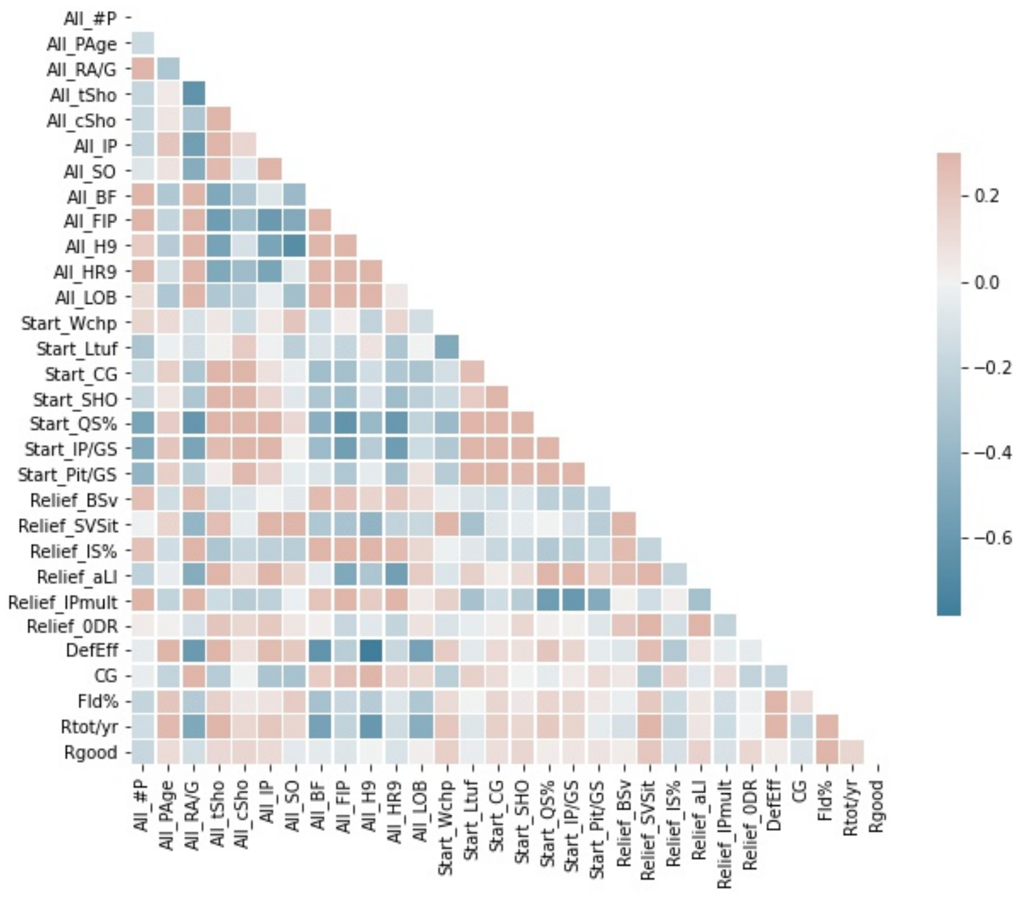
As a result of this culling, the final dataset still had 12,166 observations and 56 features:

- 2 game-related: closing over/under line and cli which measures the importance of the game for the two teams.
- 22 batting-related: including 5 "advanced" metrics
- 24 pitching-related: 7 related to starting pitching, 5 related to relief pitching & 12 totals
- 5 fielding-related The 51 team-related features were needed for both teams in a given observation; ultiamtely, leaving the final model to incorporate 104 features.

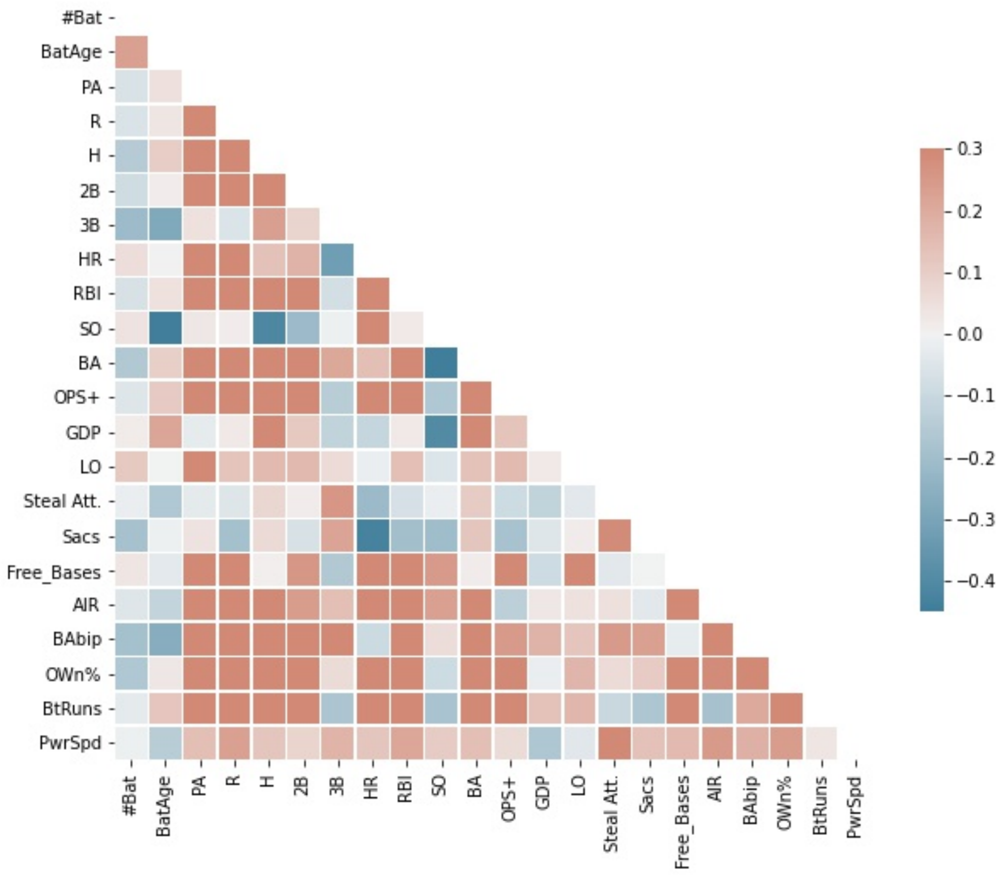
Exploratory Analysis

Annual Team Statistics

I looked at the summary stats and correlation matrices for the Defensive (Ptiching + Fielding) to look for any autocorrelation concerns--I assumed that the data collected directly from baseball reference was all correct, so no outliers or corrupted data.



Initial versions of this heatmap showed that the starter's Game Score (a mesaure of pitching quality that includes # of HRs, doubles, etc), relievers' games w/ multiple innings pitched; and overall strikeout/walk ratio appear to have strong negative correlations with a number of other features. As a result, these were removed from the dataset.



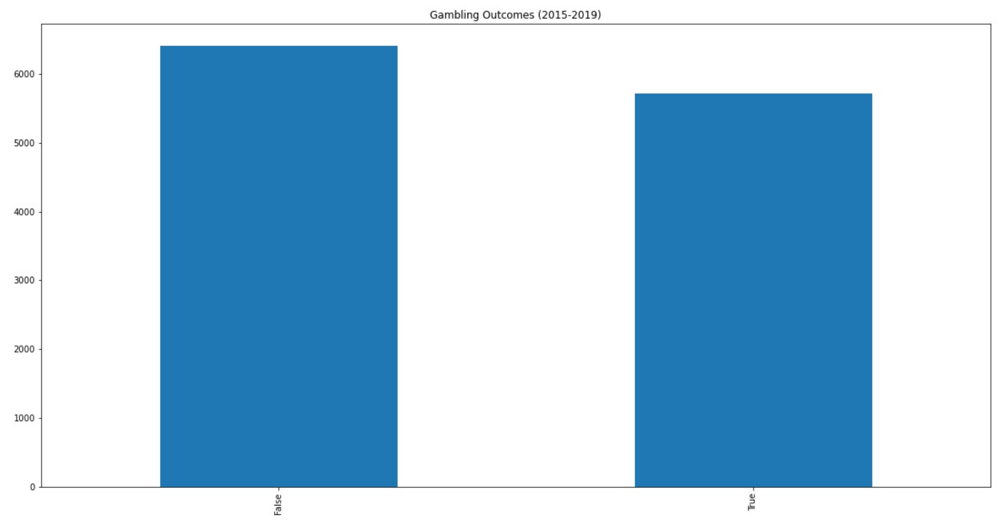
For offensive numbers, no such issues appeared as all correlations were between -0.4 and 0.3, roughly.

Without including a greater number of years, testing for autocorrelation across these different features is not useful. But one could collect more data to check for pervasiveness of these statistics year-to-year. (For the purpose of this project, there is a relationship between consecutive years.)

Game Info

Because the statistical data was so large, I focused only on game-related info for further exploration.

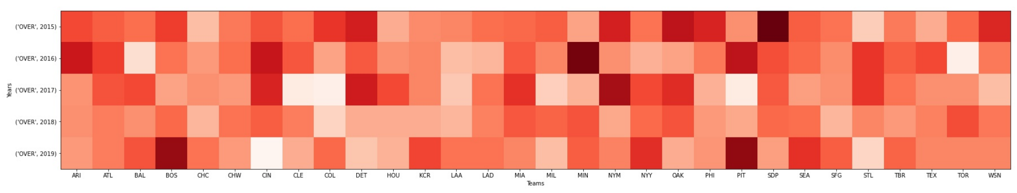
Class Balance

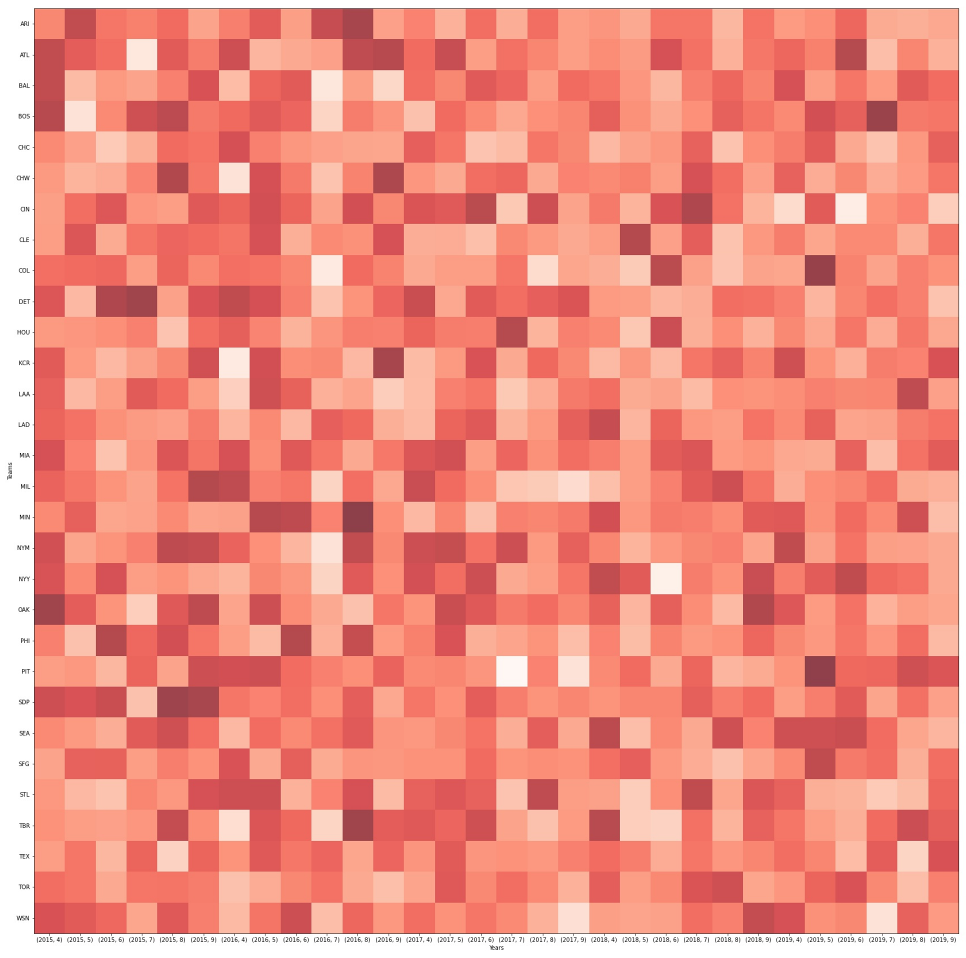


There are a bit more False outcomes which makes sense since a tie is included in that class. But nothing to worry about wrt class imbalance.

Possible Feature Importance

I looked at splitting the data between team, year, month, and the level of the O/U line and did not find much: the monthly splits did show a slightly higher % of games in April going over (but still only 49%). However, when combining team with year and with month, there appears to be certain "hot spots"


('OVER', 2015)
('OVER', 2016)
('OVER', 2017)
('OVER', 2018)
('OVER', 2019)
Years
ARI ATL BAL BOS CHC CHW CIN CLE COL DET HOU KCR LAA LAD MIA MIL MIN NYM NYY OAK PHI PIT SDP SEA SFG STL TBR TEX TOR WSN
Teams



This does support that there may be qualities in a specific team that hold for a long (e.g. a year) or short (e.g. a month) time horizon and do create bias in the bookmaker's model.

Training

For logistic regression, I used a grid search on the following hyperparameters:

- max iterations: Ultimately went with 1000 vs 500, 150, and 5000.
- penalty score: None. These models were very quick to remove almost all

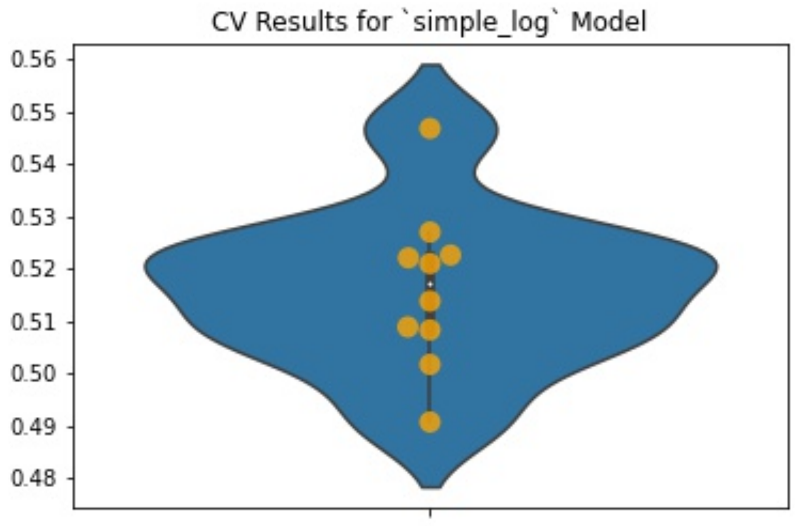
features with any regularization penalty.
- Solver: lbfgs.

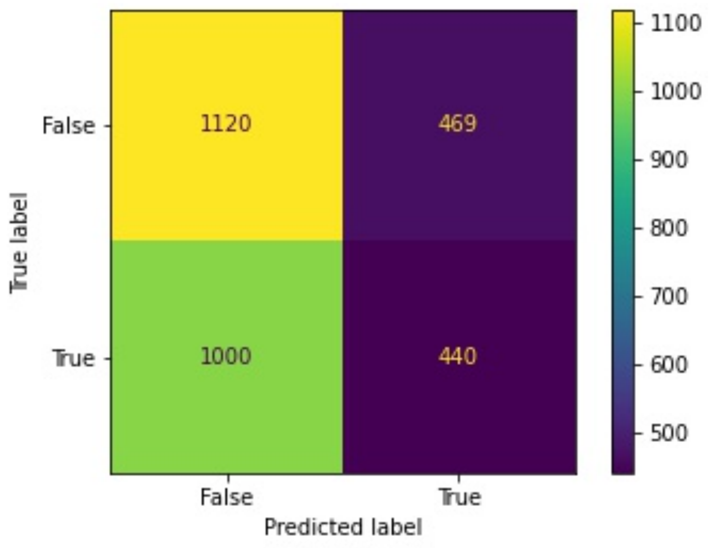
For the random forests, I used a grid search on:

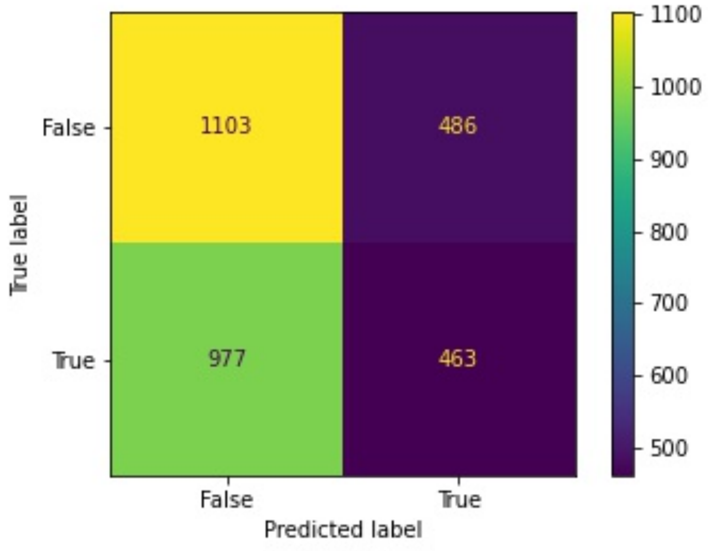
- ## of Estimators: 700 vs. 300, 400, 500, 600, 800.

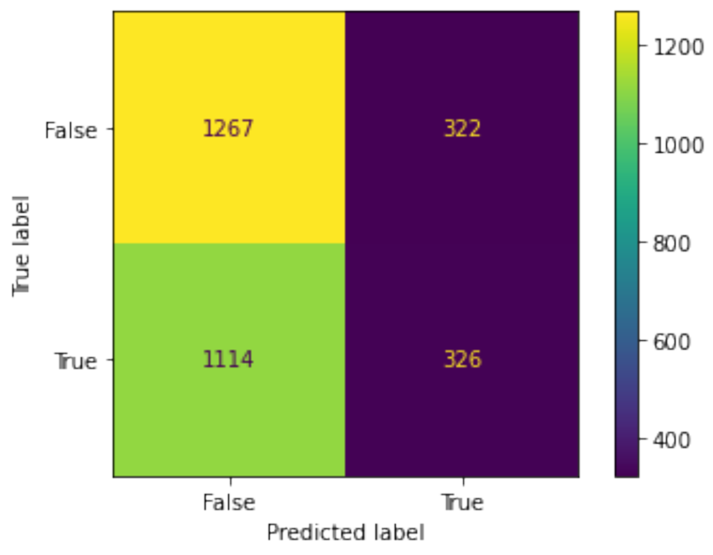
- Minimum Samples / Leaf: 80 vs.25, 50, 75 and 90.
- Maximum Leaf Nodes: 50 vs range(50,100,5).
- Maximum Features: 5 vs auto & 6

Final Results




False
True
True label
1120
469
1000
440
False
True
Predicted label
1100
1000
900
800
700
600
500


False
True
True label
1103
486
977
463
False
True
Predicted label
1100
1000
900
800
700
600
500



For all four of these models, the overall results are dissapointingly (but not unexpectedly) poor. What is notable however, is that the random forest model does have a better precision metric, even though all the grid models were set to optimize accuracy. It may be better to actually look at each model's record on True predictions because that is when a gambler bets. This may be done by changing the scoring parameter in the GridSearchCV and/or customizing the class weights as a parameter in the models. As long as this does not discourage the model from making bets then as long as you are right more often than not when one actually places a bet (assuming consistent odds), you can come out on top.

Conclusions & Further Research

Ultimately, it is not all that surprising that this model failed to succeed. Only using team-level data from the prior season is not going to carry over, especially as the season progresses. Even creating better approximations (by including YTD, player-level data or both) might only marginally improve the results because these is a fairly effecient market--bookmakers not only have their own complex models but also get to see and adjust to the behavior of their bettors, some of whom have very precise complex models themselves.