DataBall

Betting on the NBA with Data

Kevin Lane

September 14, 2017

Outline

Background

- Sports analytics began in professional baseball, most notably with the work of Bill James[1]
  - James coined the term sabermetrics as "the search for objective knowledge about baseball"
  - James selected the name to honor the Society for American Baseball Research (SABR)
- Gained widespread adoption after Billy Beane implemented James' ideas and led the Oakland Athletics to a record winning streak[2]
- Analytics has since spread to other sports and its impact is evidenced by several examples:
  - MLB's increased attention to on-base percentage beginning in the Moneyball era of the early 2000s
  - The rise of the three-point shot and subsequent fall of the midrange jumper in the NBA
  - Increased use of short, high percentage passes in the NFL

[1] James 1985.

[2] Lewis 2003.

Background

What makes sports an attractive testbed for machine learning?

According to Nate Silver, "sports nerds have it easy."[3]

1. "Sports has awesome data."
2. "In sports, we know the rules."
3. "Sports offers fast feedback and clear marks of success."

Why the NBA?

- Easily the most deterministic of the major American sports
- The NBA provides a wealth of advanced stats and player tracking data on their website
- The season is long enough at 82 games that sample size is not as much of a concern as in the NFL, who claim to have parity, but also only play 16 regular season games

[3] Silver 2015.

Process

- I used several algorithms from the popular Python machine learning library scikit-learn[4] to predict NBA game winners against the spread
  - Logistic Regression
  - Support Vector Machine
  - Random Forest
  - Multilayer Perceptron
- I collected box scores, point spreads, and over/under lines from the 1990-91 season through 2016-17
- I used the 2016-17 season as my test set and trained models with prior seasons
- All models are trained with stats averaged over a rolling window
  - The stats were shifted so that stats were not used to predict the game from which they were obtained
  - This provides a realistic scenario for making predictions in real time, such as in betting

[4] Pedregosa et al. 2011.

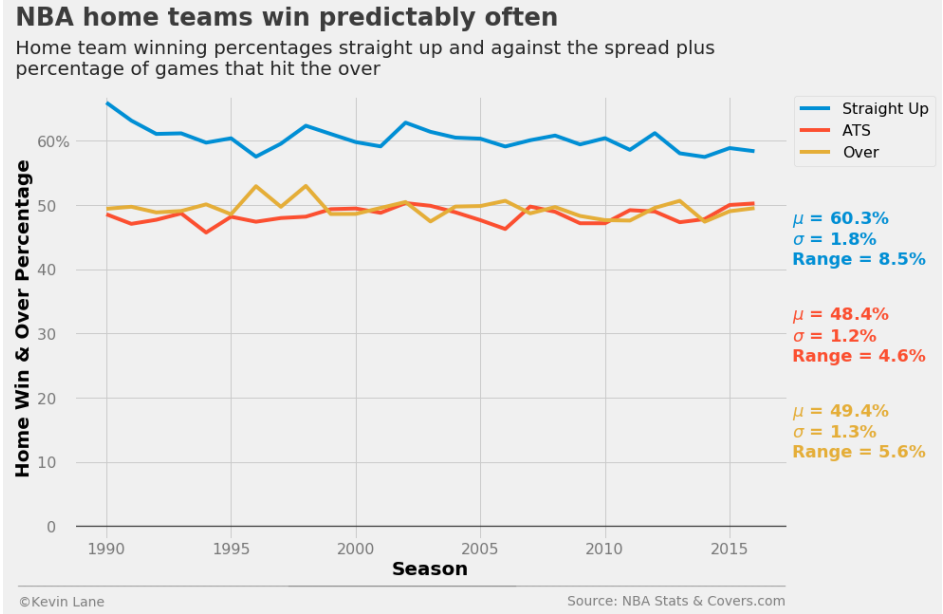
Data Wrangling

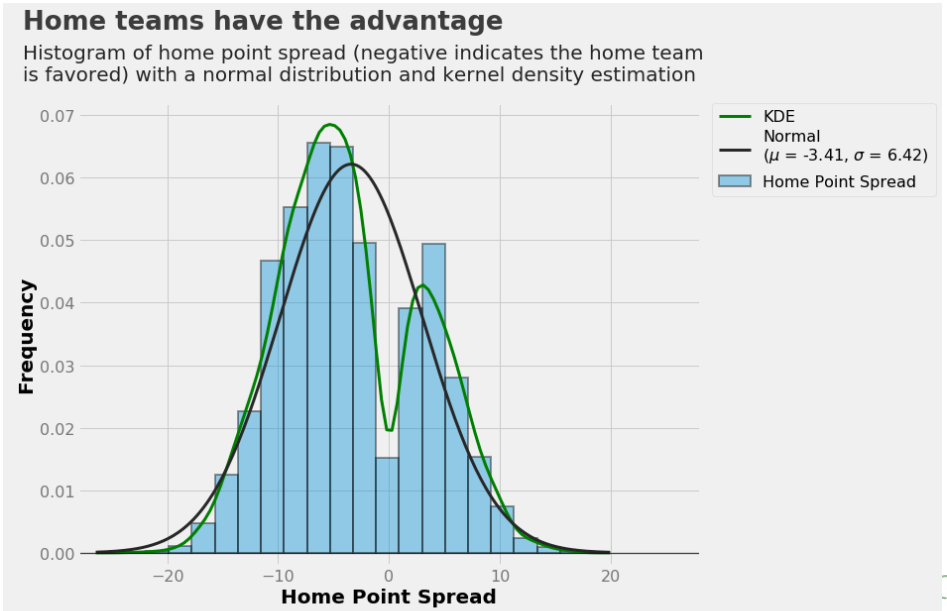
- I collected stats from the NBA's stats website `stats.nba.com`
  - The site exposes a wealth of information in JSON format through various web API endpoints
  - I utilized the Python module nba_py to format URLs and collect stats
- I scraped point spreads and over/under lines from covers.com using the Python web scraping framework Scrapy
- I stored all data to a SQLite database using Python's built-in support
- I used the basic box score stats to calculate more advanced stats
  - Offensive/defensive ratings (points scored/allowed per 100 possessions), which requires an estimate for the number of possessions
  - Simple Rating System (SRS), which is a team's average margin of victory adjusted for its strength of schedule
  - Oliver's four factors[5], which include effective FG%, TOV%, OREB%, and free throw rate
  - Weighted four factors, which is just sum of the four factors weighted according to Oliver's assigned weights

[5] Oliver 2004.

Data Exploration

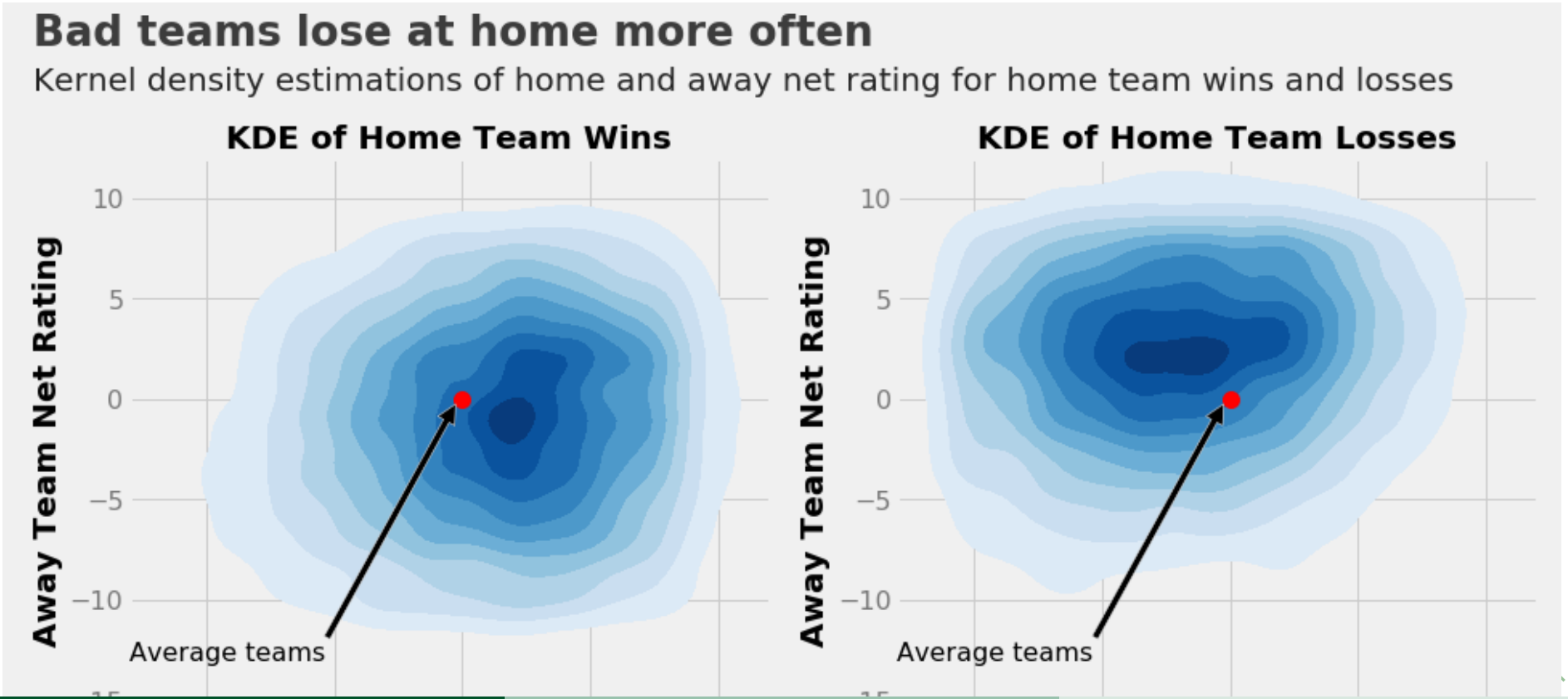
- The top figure shows that the home team winning percentage is remarkably consistent
- Teams only win about half the time against the spread (ATS)
  - This provides a tougher problem than picking game winners straight up
- The bottom figure shows that home point spreads favor the home teams
  - The betting lines indicate home court advantage is about 3.4 points
  - The distribution is bimodal because oddsmakers rarely set a spread of zero





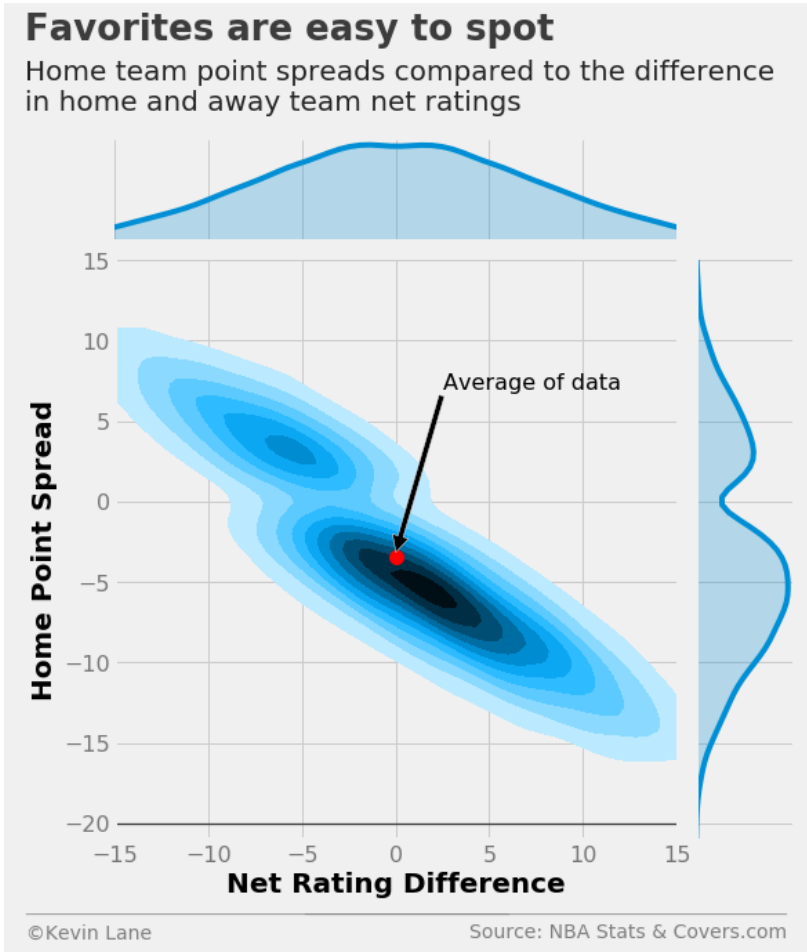
Data Exploration

- The plots below show kernel density estimations (KDE) of net rating split between home team wins and losses
- The dark region to the bottom right of the origin for home team wins shows above-average home teams tend to beat below-average visitors
- The opposite appears in the KDE of home team losses



Data Exploration

- The plot to the right shows a kernel density estimation (KDE) comparing home point spread with the difference in net rating between the two teams playing in the game
- The highest density of points occurs at a positive net rating difference and a negative point spread
  - A positive net rating difference indicates the home team is stronger
  - A negative point spread means the home team is favored



Feature Selection

- Picking winners ATS is much harder than picking winners straight up
- Simply picking the favorite wins ATS will only be right half the time

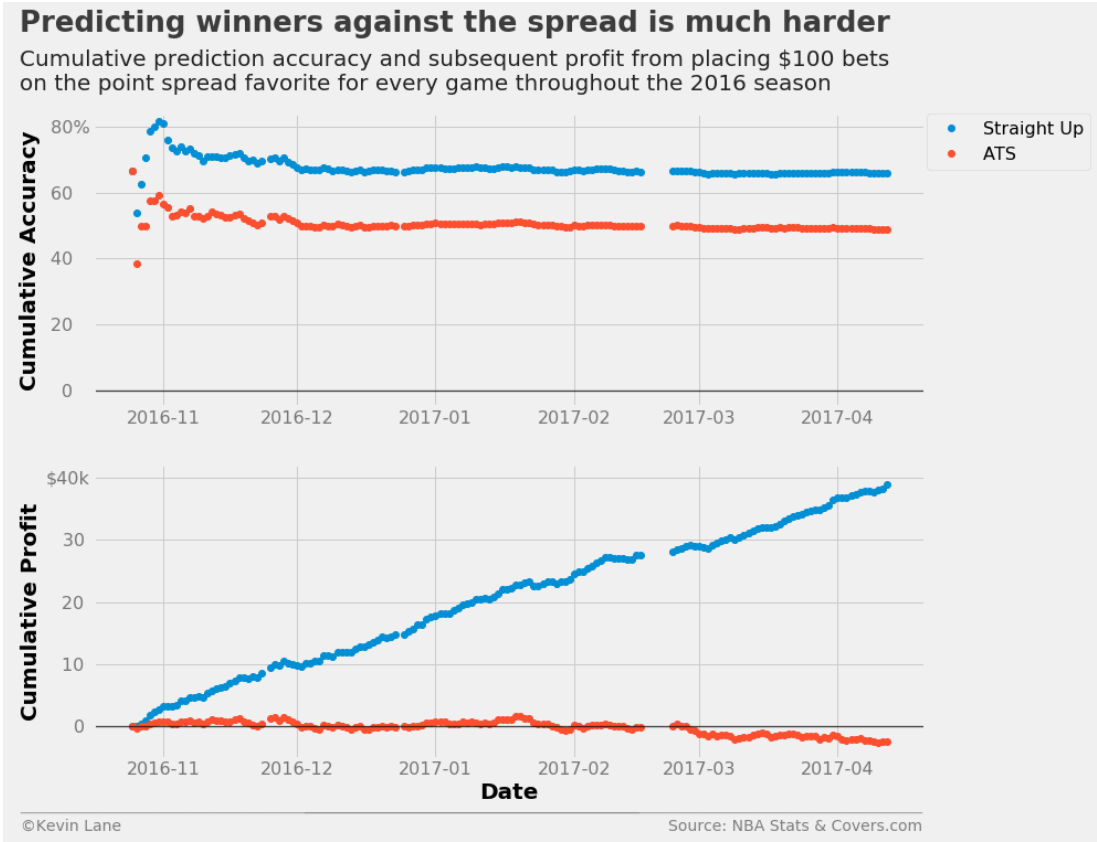


Figure 1: Comparison of Betting on Games Straight Up and ATS

Feature Selection

- The plots below show cross-validation ROC and precision/recall curves for various metrics
- None of the models perform particularly well

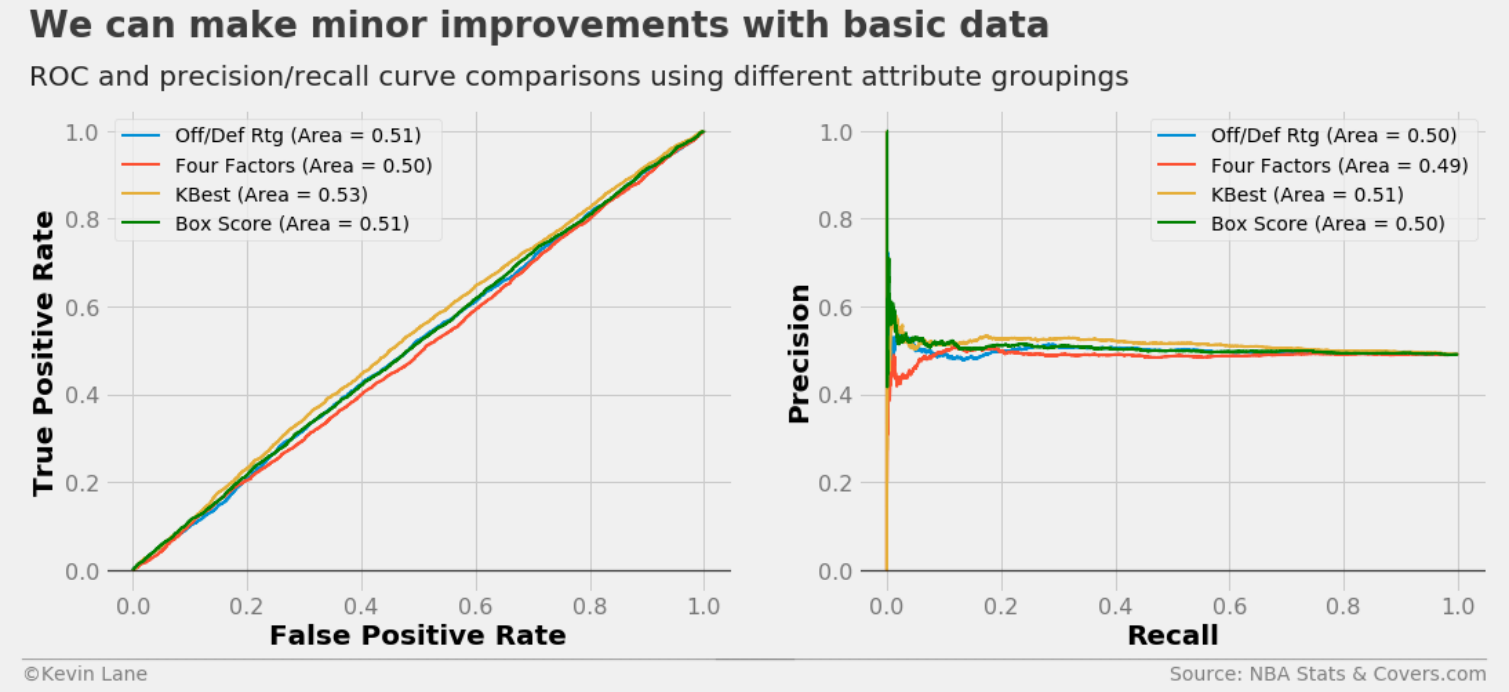


Figure 2: ROC and Precision/Recall Curve Feature Comparison

Parameter Tuning

- I used the Python module hyperopt to optimize model parameters
- It provides a flexible framework for optimizing any scikit-learn classifier

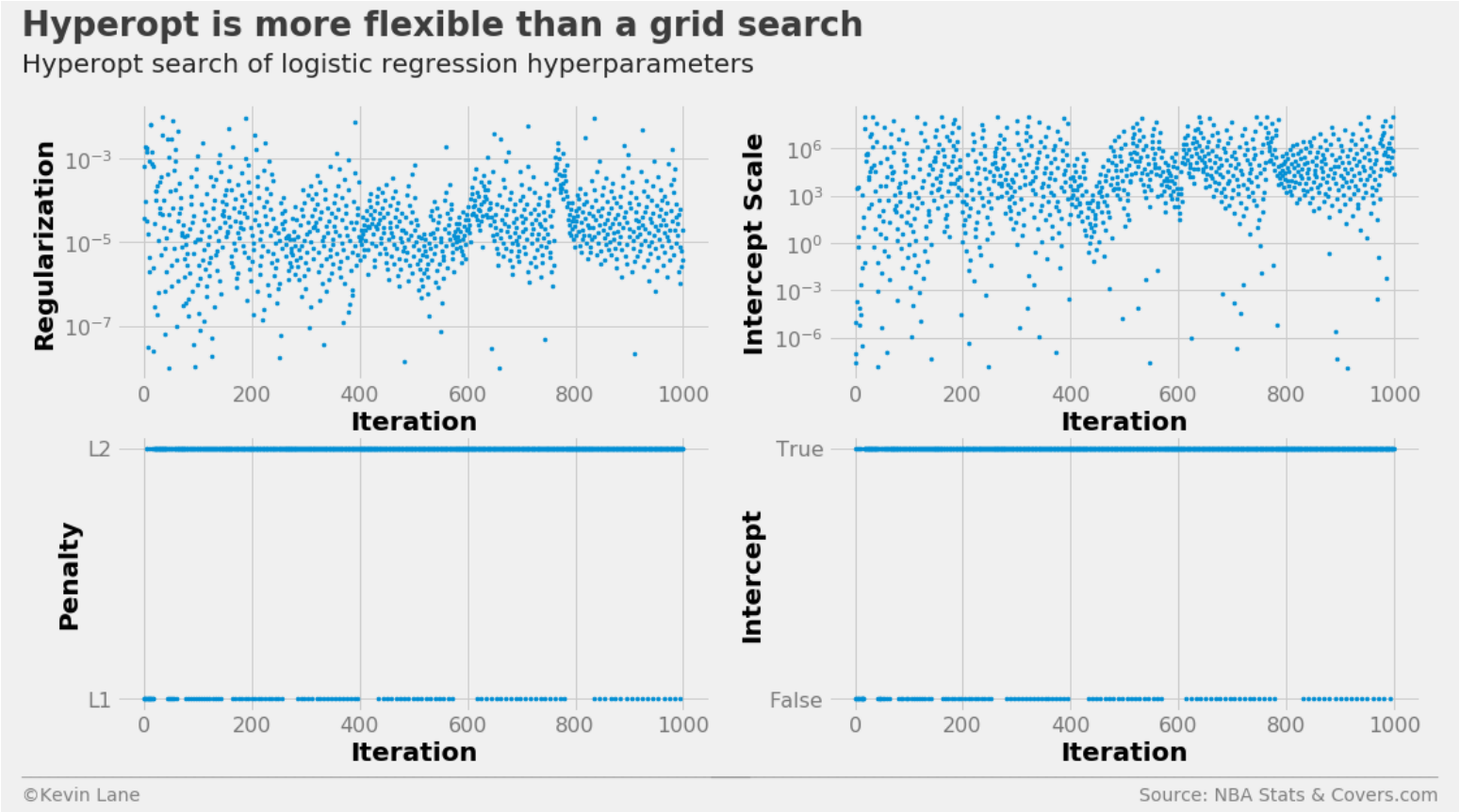


Figure 3: Logistic Regression Hyperopt Parameter Tuning

Parameter Tuning

- Parameter tuning yielded marginal benefits for logistic regression
- None of the optimized models performed substantially better than models with default parameters

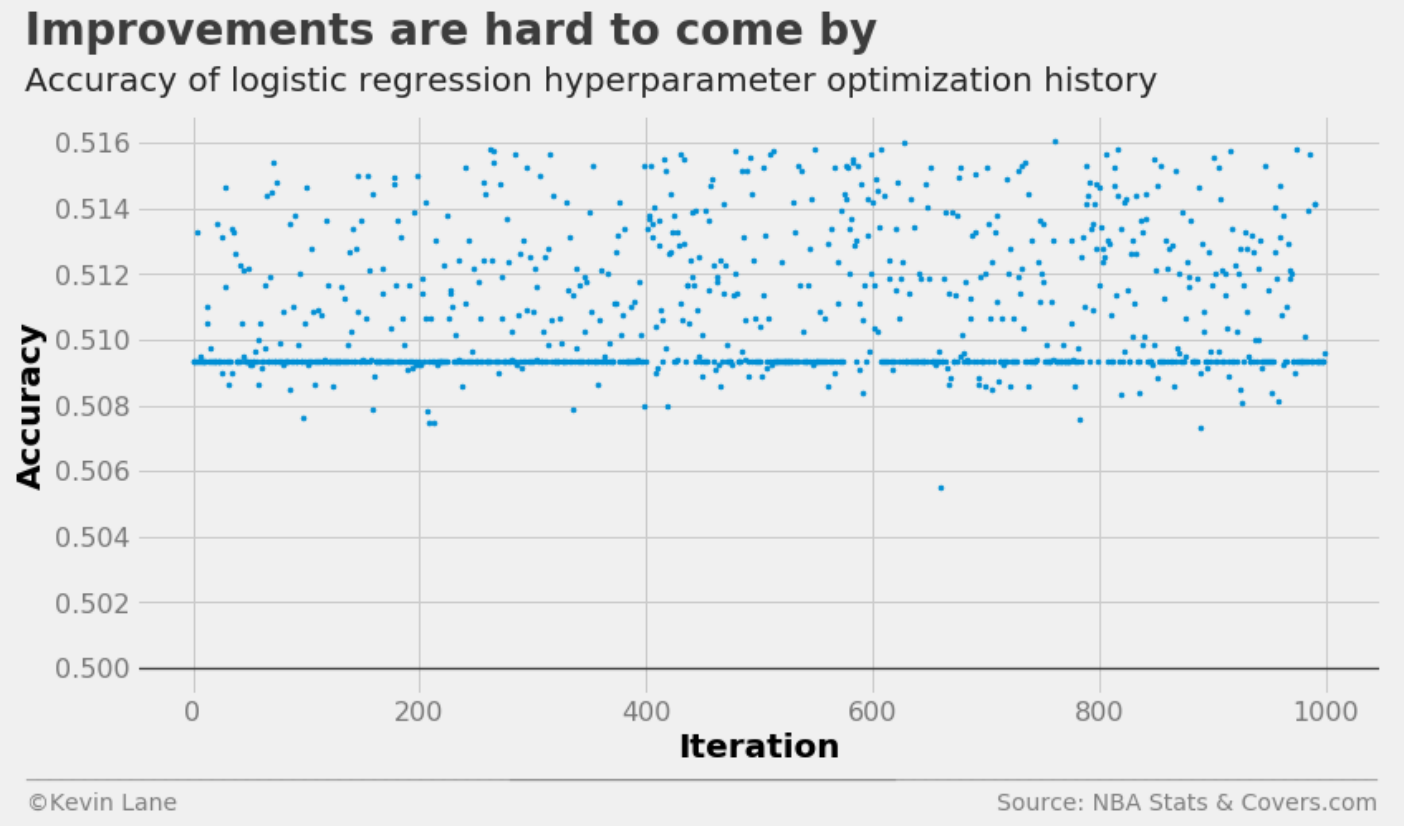


Figure 4: Logistic Regression Accuracy History

Model Performance

- Logistic regression does a decent job at predicting home wins, but struggles with home losses
  - The confusion matrix shows the model tends to predict the home team wins
- None of the models performed particularly well
  - The maximum accuracy was below 55%
- Any accuracy above 50% will result in a profit
  - A 1% improvement over the course of a season results in about $2,500 extra when betting $100 on every game

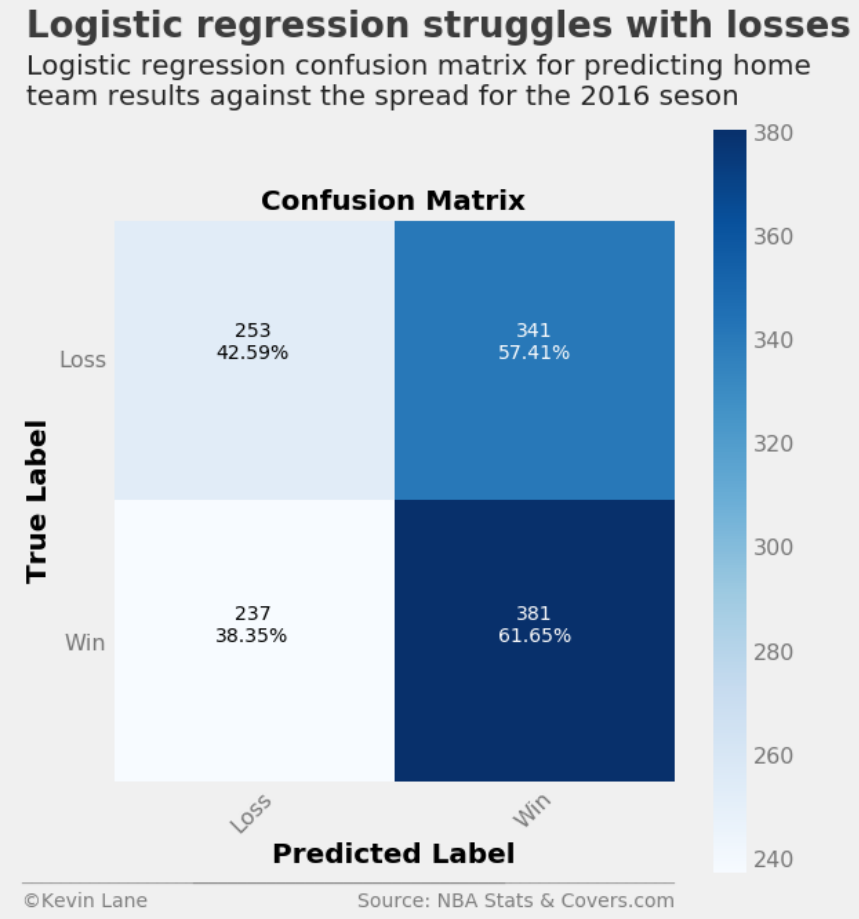


Figure 5: Logistic Regression Confusion Matrix

Model Performance

- The neural network was the highest earning model
- All models with optimized parameters returned a profit

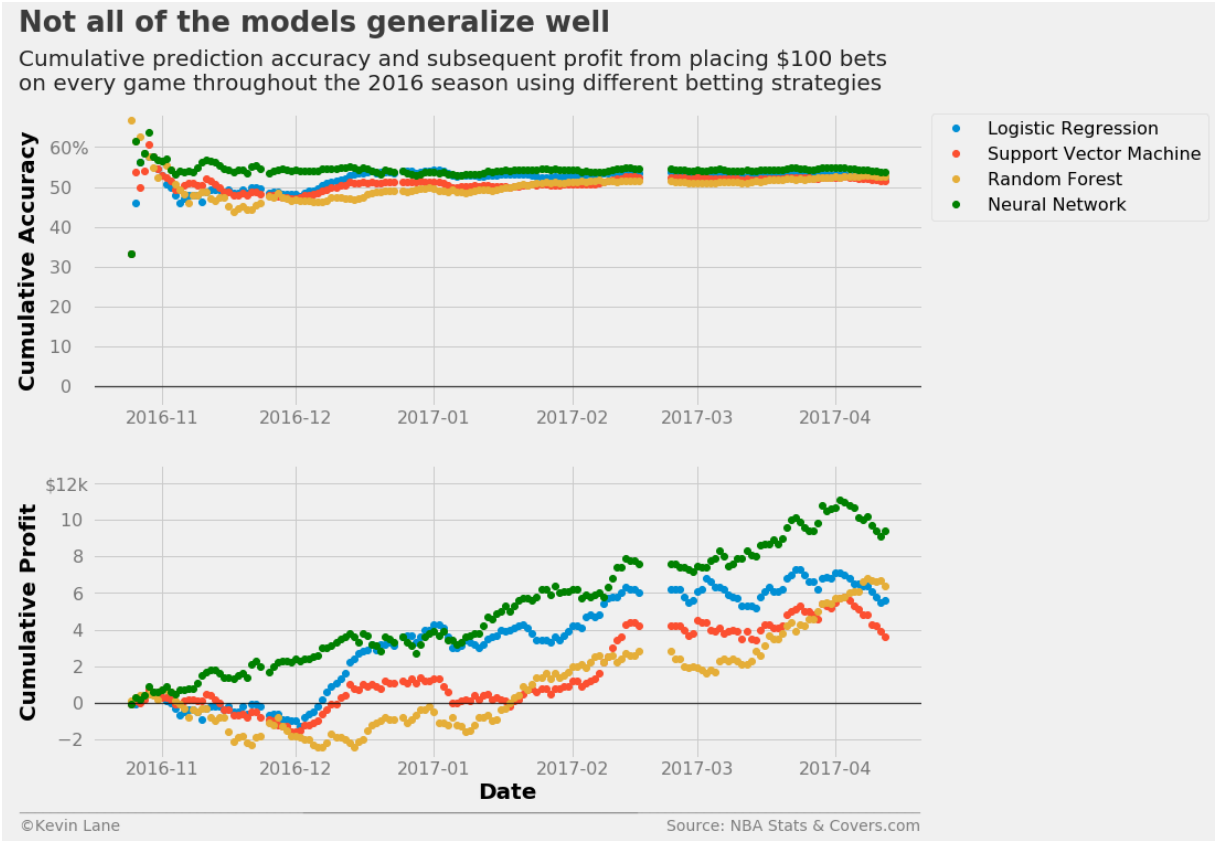


Figure 6: Model Performance Comparison

Model Performance

- The hyperparameters optimized prior to the 2016 season did not work well for the logistic regression model
- Logistic regression with default parameters earned almost double

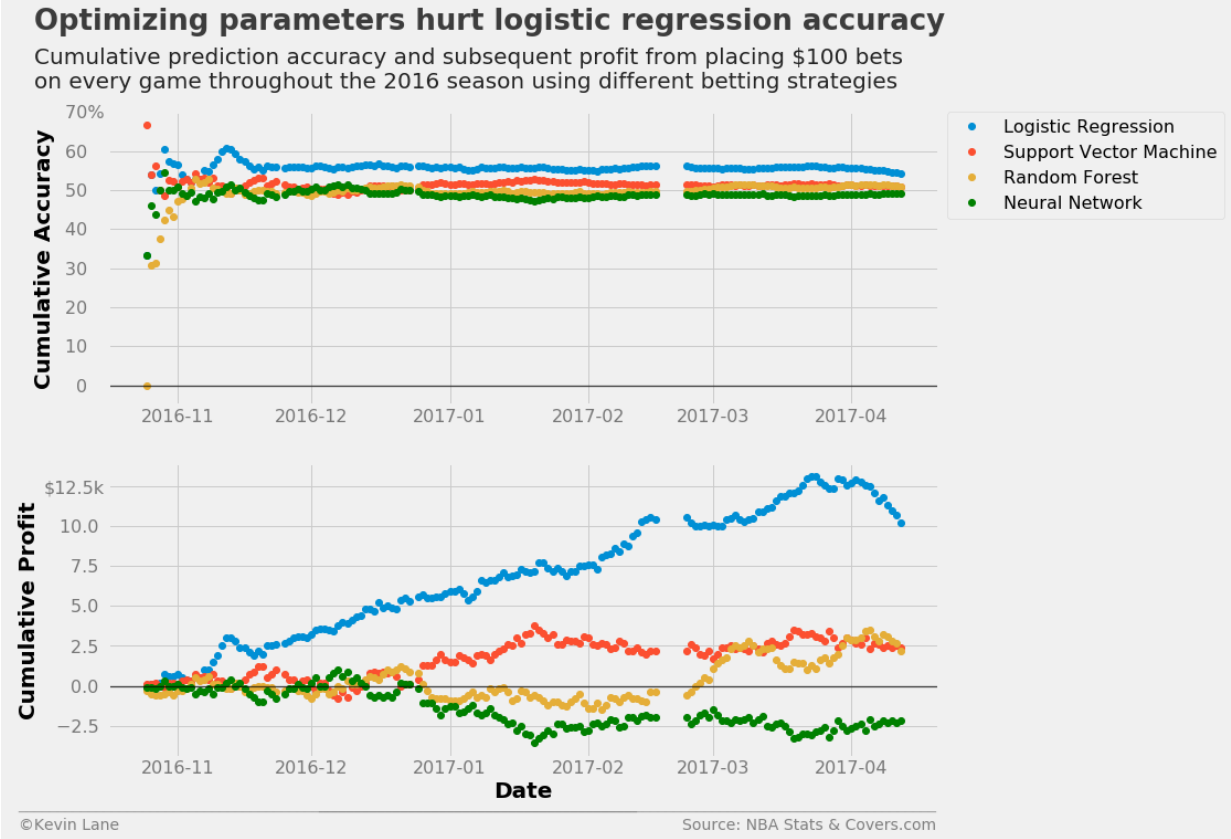


Figure 7: Default Model Performance Comparison

Future Work

- Incorporate player stats to adjust predictions as rosters fluctuate and players sit for rest or injury
- Test how well models generalize by predicting other seasons
- Calculate stats relative to league average to control for opponent strength and league-wide changes in game strategy
- Include a dummy variable indicating if teams are playing the second game of a back-to-back
- Predict games against over/under lines

References

[1] James, Bill. *The Bill James Historical Baseball Abstract*. Villard, 1985.

[2] Lewis, Michael. *Moneyball: The Art of Winning an Unfair Game*. W. W. Norton & Company, 2003.

[3] Silver, Nate. *Rich Data, Poor Data*. Feb. 2015. URL: https://fivethirtyeight.com/features/rich-data-poor-data/.

[4] Pedregosa, F. et al. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[5] Oliver, Dean. *Basketball on Paper: Rules and Tools for Performance Analysis*. Potomac Books, 2004.