NBA Player Evaluation & Roster Construction

Graeme Ashley, Jose Smith, Timothy Wijaya
Professor Austin Pollok
July 8, 2024

Table of Contents

1. Abstract

This project leveraged advanced supervised learning techniques combined with optimization algorithms to create a useful metric for evaluating player performance. Using these techniques, we developed a performance metric aimed at maximizing wins, which was subsequently used to optimize basketball team rosters through the CVXPY optimization library. Our approach provides a systematic way to evaluate players and ensures optimal team compositions based on various constraints such as budget and positional needs.

The methodologies we developed have significant business implications, particularly in enhancing team performance, making informed decisions during free agency, and optimizing draft picks. The flexibility of our model allows it to be applied across different sports, including the WNBA, once relevant data becomes accessible. This adaptability ensures that sports teams can consistently make data-driven decisions to maximize their chances of success.

Additionally, our optimization model is designed to be highly adaptable, capable of incorporating new data sources and evolving with the needs of the sports industry. By focusing on maximizing wins, our metric ensures that every player selection is geared towards achieving the highest possible performance on the court. The potential applications of this model extend beyond basketball, providing valuable insights and optimization strategies for any team sport where player performance and team composition are critical factors.

2. Introduction

The landscape of professional sports is continuously evolving, with leagues frequently exploring opportunities for expansion to new markets. This growth necessitates the need for expansion teams to construct competitive rosters quickly. One of the critical challenges these teams face is evaluating a large pool of available players to identify those who can contribute most effectively to the team's success.

In this context, having a reliable and comprehensive metric for player evaluation becomes key. Traditional methods of player evaluation often rely on subjective assessments and basic statistics, which can lead to inconsistent and suboptimal decision-making. Advanced metrics that provide a more nuanced and holistic view of player performance are crucial for teams looking to build a competitive edge. By developing a comprehensive metric that quantifies player performance based on their contribution to wins, teams can make more informed and strategic decisions.

Our project addresses this need by combining supervised learning techniques with optimization algorithms to develop a performance evaluation metric. This metric is designed to provide a comprehensive view of a player's impact on the game, factoring in various performance indicators and their relative importance in contributing to wins.

Teams that adopt such advanced metrics can achieve several advantages such as enhanced decision-making as by using data-driven insights, teams can identify undervalued players who may not be recognized through traditional scouting methods. Also, they can create optimized rosters due to the optimization model that ensures that teams are composed of players who collectively maximize the chances of winning while also adhering to constraints like salary caps and positional requirements. With a reliable performance metric, teams can plan more effectively for free agency, drafts, and trades, making strategic moves that align with their long-term goals. Finally, there is an increased competitiveness as expansion teams can quickly become competitive, reducing the typical lag period associated with new teams in a league.

Moreover, the ability to simulate various scenarios and constraints allows teams to explore different strategies and choose the one that best fits their objectives and resources. This strategic flexibility is invaluable in a dynamic and fast-paced sports environment.

While this project focused on basketball, the principles and methodologies developed are highly adaptable. The core idea of using performance metrics to drive optimization can be applied to any sport, making this approach universally valuable. As the WNBA and other sports leagues increasingly embrace data analytics, our model stands ready to provide actionable insights and competitive advantages.

Optimization models in sports is not just about selecting the best players; it also involves managing team finances, predicting player development, and planning for future seasons. By integrating these factors, our model offers a comprehensive solution that can increase the success of how teams are built and managed.

The integration of supervised learning and optimization techniques in evaluating and selecting players represents a significant advancement in sports management. By adopting these methods, teams can ensure that their rosters are not only competitive but also strategically aligned to achieve maximum success. The future of sports analytics lies in leveraging such sophisticated models to make data-driven decisions, ultimately leading to better performance and greater success on the field or court.

3. Methods

Our primary objective is to identify the metrics that most significantly contribute to wins. To achieve this, we developed a model to predict a team's total wins and then mapped these metrics back to individual player performance to determine how players contribute to team success.

We began our process by obtaining two identical datasets of NBA box scores per 100 possessions from the 2003-2024 seasons: one dataset for each team’s home statistics and their opponent’s statistics against them at home, and the other for each team’s away statistics and their opponent’s statistics against them while playing on the road. Each row contains a unique identifier, "TeamSeason," which is a string concatenating the season and the team name (e.g., '2022Lakers'), and that team's statistics per 100 possessions for that season. Since each team plays 41 games at home and 41 away, there are 41 games worth of statistics aggregated into each row (**Refer to Appendix 7.1**). Box score data was scraped from Game Summaries from nba.com, using the hoopR R package.

Our initial step in data cleaning was to separate the 'TeamSeason' column into two distinct columns for 'team' and 'season.' We then merged the 'home' and 'away' datasets on 'team' and 'season' and aggregated each team's away and home statistics so that each row corresponds to a full 82-game season's worth of performance. RAPM data was collected from the nbashotcharts.com, and every team's O-RAPM, D-RAPM, and Overall RAPM, were the product of a players minutes and their RAPM, done for entire rosters and summed together. We then modified the 'team' column to match the format of our master dataframe, and then joined it with our master dataframe on 'team' and 'season' to ensure that each team had impact data corresponding to each season's statistics. After merging, our master dataset included only the seasons from 2011-2022 (**Refer to Appendix 7.2)**.

We discovered a time series component in our data, as a team's total wins in the previous season (t-1) is a significant predictor for their wins in the current season (t) (**Refer to Appendix 7.12)**. To address this autocorrelation, we introduced a fixed effects model by creating dummy variables for each team and each season, which significantly increased the dimensionality of our dataset, and included two lags of total wins. Before running any models, our dataset contained 384 rows and 146 columns.

To avoid the autocorrelation issues that a random train-test split can cause with time series data, we assigned all seasons before 2020 to the training set and tested our models on the 2020-2022 seasons. The first model we ran was a standard linear regression without any variable selection, which yielded poor results as expected due to the high dimensionality and complexity of our dataset (**Refer to Appendix 7.3)**. Next, we applied regularized regression techniques in hopes that the regularization parameter would perform variable selection and improve overall model performance. After running Ridge, LASSO, and ElasticNet regression models and performing five-fold cross-validation, LASSO performed the best on the test set with an MSE of

15.25 and an R^2 of 88.04%. Ridge was the second best, with an MSE of 21.59 and an R^2 of 80.08%, followed by ElasticNet with an MSE of 34.59 and an R^2 of 72.89% (**Refer to Appendix 7.4)**. The relative similarity in our results across the three regularized regression models confirmed the predictive abilities of our metrics and positioned us to map the coefficients of our models to individual player statistics. However, with more rows of data we could achieve better performance and more similarity across our different regression models.

Next, we aimed to map our model coefficients to individual player statistics to identify which players possess the most winning skill sets. We obtained a dataset of 6,563 rows containing individual player statistics from the 2010-2022 seasons (**Refer to Appendix 7.5)**. To map the coefficients from our models to the player dataset, we needed to ensure the dimensions of the player dataset matched those of the dataset used in our models. We normalized the data using StandardScaler to ensure that the player statistics were on the same scale as the data in our models.

Non-predictive columns were dropped, and players who played fewer than 30 games in a season were filtered out. Additionally, we needed to impute data that was present in our team dataset but unavailable in our player dataset. For columns that existed in the master dataset but not in the player dataset, we imputed zeroes. Since home and away data was not available for individual player statistics, we assumed that a player's home statistics were equal to their performance on the road and imputed the data accordingly (**Refer to Appendix 7.6)**.

Finally, we computed the coefficients for players by taking the dot product of the array of coefficients from our models with the new player dataset. This resulted in each player having a coefficient value for each of the three models. A positive coefficient indicated that the player contributed positively to winning, while a negative coefficient indicated a negative contribution to winning games. To make the coefficients more interpretable, we standardized the coefficients to have an average of 100. Therefore, a player who neither adds to nor subtracts from their team's wins would have a value of 100, and players who contribute more to winning would have a coefficient above 100, and vice versa for those who detract from winning (**Refer to Appendix 7.7)**.

Lastly, we combined a dataset containing player positions with our dataframe, since positional constraints are crucial to the optimization problem. To perform our optimization, we aimed to maximize the sum of player coefficients for our team, as an increase in player coefficients would result in more team wins. Using CVXPY, we ran an optimization on our entire player dataset with 10 decision variables, as we wanted our roster to consist of 10 players. We also included positional constraints, ensuring at least two guards, two centers, and two forwards on our roster. We conducted an optimization for each of our metrics, resulting in three separate 10-man rosters, each maximizing the player metric predicted by the three regularized regression models. We also introduced salary constraints by merging a salary dataset with our player dataset, which provided interesting and practical results that are detailed in the following "Results" section. Salaries were taken from a Github dataset from erikgregorywebb from 2010-2020, and the remaining years were parsed from spotrac.com

4. Results

Upon investigating the coefficients for our models predicting total wins, we observed a consistent trend where the most important predictors, by absolute value, centered around increasing field goals attempted while decreasing opponent field goal attempts. Given that our data is standardized to 100 possessions, we can infer that teams playing at a fast offensive pace and a slow defensive pace are more likely to win games. Additionally, our LASSO model exhibited extreme sparsity, with 119 out of 143 coefficients being zeroed out (**Refer to Appendix 7.8)**.

In our first optimization problem, forming the optimized roster for the 2011 - 2022 seasons, we were satisfied with the results. The roster included many of the best players of the generation, validating that our metric effectively identifies successful players and that the optimization is functioning correctly (**Refer to Appendix 7.9)**. Both the Ridge and Elastic Net models yielded similar roster constructions, resulting in 35.6% and 34.7% more wins than an average roster, respectively. However, the LASSO model produced intriguing results, selecting mostly forwards and centers, with Giannis Antetokounmpo and Josh Smith being the only guards—both of whom could also be considered forwards due to their physical attributes. The LASSO model favored taller and more physical teams, reminiscent of the early-mid 2000s, while the Ridge and Elastic Net models aligned more with the fast-paced, three-point shooting rosters that have dominated in recent years. Interestingly, the LASSO model achieved the best R^2, lowest MSE, and highest predicted wins above average for the optimized roster (39.1%). We aim to further investigate why the LASSO model prefers one archetype of player and how we can adjust it to select a more practical roster.

Next, we implemented salary and season constraints to construct optimal rosters for the 2023-24 season based on 2022-2023 statistics and salaries under different salary cap values. These rosters more closely resemble practical NBA 10-man rosters in today's league, though the financial aspect is idealized, as it would be impossible to construct these rosters given that all the players are currently under contract (**Refer to Appendix 7.10)**.

Lastly, we reduced our decision variables from ten to four and adjusted the salary cap constraint to reflect what certain teams have available to spend this offseason. This approach yielded the most practical results, as these players are currently in the league and available for new contracts or as trade targets (**Refer to Appendix 7.11**). In the future, we plan to integrate more granular financial data, such as contract structure and duration, and designate which players are free agents or trade targets. This would enable us to provide tangible recommendations to teams for free agent and trade evaluations.

5. Discussion

This project demonstrated the power and flexibility of combining supervised learning and optimization techniques to evaluate and select basketball players, ultimately constructing competitive rosters that maximize team performance. However, the potential applications of our optimization model extend far beyond the scope of what we have currently implemented. There are several ways we could enhance and expand our model to provide even more comprehensive and actionable insights for sports teams.

By integrating injury data, we can evaluate the risk associated with each player and factor this into our optimization model. This would allow teams to balance the potential performance benefits of a player with their injury risk, leading to more resilient and reliable team compositions. Adding more detailed contract information, such as contract length, player options, and trade clauses, would enable a more nuanced analysis of player acquisitions and their long-term financial impact. This would help teams make smarter financial decisions and better plan for future seasons. Additionally, by including data on which players are currently available in the market or are realistic trade targets, teams can use our model to explore various acquisition scenarios. This would allow for dynamic and real-time optimization, providing teams with actionable insights during critical decision-making periods like free agency and trade deadlines.

The utility of our optimization model for sports teams cannot be overstated. Teams can make more informed and objective decisions by leveraging comprehensive data and sophisticated algorithms. This reduces the reliance on subjective assessments and increases the likelihood of successful player acquisitions and team compositions. With constraints such as salary caps and positional needs, our model ensures that resources are allocated in the most efficient way possible. This helps teams maximize their performance within their financial and structural limits. The ability to simulate various scenarios and constraints allows teams to explore different strategies and choose the one that best fits their objectives and resources. This strategic flexibility is invaluable in a dynamic and fast-paced sports environment. By adopting advanced analytics and optimization techniques, teams can gain a competitive edge over their competition. This can be particularly beneficial for expansion teams or teams undergoing rebuilding phases, as it enables them to quickly become competitive.

In conclusion, the integration of supervised learning and optimization techniques in evaluating and selecting players represents a significant advancement in sports management. Our model not only provides a systematic and data-driven approach to building winning teams but also offers flexibility and adaptability to meet the evolving needs of sports organizations. As we continue to refine and expand our model, the potential applications across different sports and contexts are vast. By incorporating additional data sources such as injury records, detailed contract information, and market availability, our model can provide even deeper insights and more powerful optimization capabilities. The future of sports analytics lies in leveraging such sophisticated models to make data-driven decisions, ultimately leading to better performance and greater success on the field or court.

In the ever-competitive world of professional sports, the ability to make informed and strategic decisions can be the difference between winning and losing. Our optimization model offers a valuable tool for teams looking to enhance their decision-making processes, optimize their resources, and build championship-winning rosters.

6. References

Works Cited

nbashotcharts. "RAPM Dataset." nbashotcharts.com, http://nbashotcharts.com/home. Accessed 11 July 2024.

hoopR. "HoopR Package." sportsdataverse.org, https://hoopr.sportsdataverse.org/. Accessed 11 July 2024.

Spotrac. "NBA Salaries." spotrac.com, https://www.spotrac.com/nba/. Accessed 11 July 2024.

Webb, Erik Gregory. "NBA Salaries." GitHub, 2017, https://github.com/erikgregorywebb/datasets/blob/master/nba-salaries.csv. Accessed 11 July 2024.

7. Appendix

7.1 - Raw data frame of home team statistics

	TeamSeason	team_score1	assists1	blocks1	defensive_rebounds1	fast_break_points1	field_goals_made1	field_goals_attempted1	flagrant_fouls1
0	200376ers	103.863568	21.299211	3.696118	31.004844	6.328605	39.540483	88.254783	0.106363
1	2003Bucks	105.473642	22.815880	4.356000	31.687246	5.073147	39.363369	89.537056	0.106244
2	2003Bulls	96.884025	21.042915	5.681077	30.978432	5.553699	37.372828	86.362568	0.127378
3	2003Cavaliers	92.775680	20.588223	5.430144	31.988952	6.124996	35.077185	84.025685	0.025735
4	2003Celtics	100.867010	19.103195	3.959766	31.811903	3.772480	35.129545	86.338950	0.133776

5 rows × 50 columns

7.2 - Master data frame after cleaning

	season	team	team_score_home	assists_home	blocks_home	defensive_rebounds_home	fast_break_points_home	field_goals_made_home
1	2010	76ers	106.379759	21.826870	6.007636	32.556666	19.203449	40.610571
2	2011	76ers	104.245712	23.362721	4.197989	33.792507	17.104849	40.441496
3	2012	76ers	103.461156	23.249698	5.812424	34.177056	15.743367	41.284821
4	2013	76ers	117.513286	26.916451	5.720918	38.952262	13.598904	46.580153
5	2014	76ers	117.262895	24.282297	4.873866	37.105206	17.841831	44.793149
...	...	...	...	...	...	...	...	...
409	2018	Wizards	125.057466	4.747174	25.017318	38.152138	15.231731	46.889269
410	2019	Wizards	126.401906	29.273765	4.694849	36.343655	16.956690	46.589473
411	2020	Wizards	129.485215	29.035509	5.011523	34.830082	16.600669	46.920380
412	2021	Wizards	126.932556	28.262086	4.751486	38.968356	14.501289	46.743515
413	2022	Wizards	124.446309	28.727245	5.647482	39.849328	12.793277	46.822240

384 rows × 146 columns

7.3 - Linear regression results in predicting total wins

```
# Linear Regression #

model = LinearRegression()
scaler = StandardScaler()

X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

model.fit(X_train, y_train)
y_pred = model.predict(X_test)

lr_coefs = model.coef_
mse = mean_squared_error(y_test, y_pred)
print("Mean Squared Error on Test Data:", mse)
r2 = r2_score(y_test, y_pred)
print('R2 score for LR is ', r2)
print( ' ')
```

```
Mean Squared Error on Test Data: 13183.356620488805
R2 score for LR is  -102.31694219346683
```

7.4 - Regularized regression results in predicting total wins

```
#Ridge

scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

alphas = [1e-15, 1e-10, 1e-8, 1e-5, 1e-4, 1e-3, 1e-2, 0.1, 0.5, 1.0, 5.0, 10.0, 20.0, 50.0, 100.0, 500.0, 1000.0]

param_grid = {'alpha': alphas}

ridge_model = Ridge()
grid_search = GridSearchCV(estimator=ridge_model, param_grid=param_grid, scoring='neg_mean_squared_error', cv=5)
grid_search.fit(X_train_scaled, y_train)

best_alpha = grid_search.best_params_['alpha']
ridge_model = Ridge(alpha=best_alpha)
ridge_model.fit(X_train_scaled, y_train)
y_pred = ridge_model.predict(X_test_scaled)

ridge_coefs = ridge_model.coef_

print("Best Alpha:", grid_search.best_params_['alpha'])
mse = mean_squared_error(y_test, y_pred)
print("Mean Squared Error on Test Data:", mse)
r2 = r2_score(y_test, y_pred)
print('R2 score for Ridge is ', r2)
print( ' ' )
```

```
Best Alpha: 5.0
Mean Squared Error on Test Data: 21.59126477052915
R2 score for Ridge is  0.8307909344791822
```

```
## Lasso ##

lasso_model = Lasso(max_iter = 100000, tol=6e-2)

grid_search = GridSearchCV(estimator=lasso_model, param_grid=param_grid, scoring='neg_mean_squared_error', cv=5)

grid_search.fit(X_train, y_train)
best_lasso_model = grid_search.best_estimator_

y_pred = best_lasso_model.predict(X_test)
r2 = r2_score(y_test, y_pred)

test_error = mean_squared_error(y_test, y_pred)
print("Best Alpha:", grid_search.best_params_['alpha'])
print("Lasso Regression Test Error (MSE):", test_error)
print('Lasso R2 Score: ', r2)
print( ' ' ' ' ' ' ' ' ' ' ')
lasso_coefs = best_lasso_model.coef_
```

```
Best Alpha: 1.0
Lasso Regression Test Error (MSE): 15.254912790767476
Lasso R2 Score:  0.8804484329491147
```

```
# Elastic Net

scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

elastic_net_model = ElasticNet(tol = 6e-2)

param_grid = {
    'alpha': [1e-15, 1e-10, 1e-8, 1e-5, 1e-4, 1e-3, 1e-2, 0.1, 0.5, 1.0, 5.0, 10.0, 20.0, 50.0, 100.0, 500.0,
             1000.0],
    'l1_ratio': [0.1, 0.3, 0.5, 0.7, 0.9, 0.95, 1.0]
}

grid_search = GridSearchCV(estimator=elastic_net_model, param_grid=param_grid, scoring='neg_mean_squared_error',
                           cv=5)

grid_search.fit(X_train_scaled, y_train)

best_elastic_net_model = grid_search.best_estimator_

y_pred = best_elastic_net_model.predict(X_test_scaled)

r2 = r2_score(y_test, y_pred)
test_error = mean_squared_error(y_test, y_pred)

print("Best Parameters:", grid_search.best_params_)
print("Elastic Net Regression Test Error (MSE):", test_error)
print('Elastic Net R2 Score: ', r2)
print('')

elastic_net_coefs = best_elastic_net_model.coef_
```

```
Best Parameters: {'alpha': 0.1, 'l1_ratio': 0.1}
Elastic Net Regression Test Error (MSE): 34.58979156188198
Elastic Net R2 Score:  0.7289224893052675
```

7.5 - Raw data frame of player statistics

	Unnamed: 0	playerId	Player	GP	Min	FGM	FGA	FG%	3PM	3PA	...	X	playerName	LA_RAPM	LA_RAPM__Def	LA_RAPM__Off	RAPM
0	1	201985	AJ Price	50	15.90	2.28	6.40	0.356	0.82	2.98	...	3974	AJ Price	1.50	0.39	1.10	0.04
1	2	201166	Aaron Brooks	59	21.76	3.73	9.95	0.375	1.19	4.00	...	3090	Aaron Brooks	-1.81	-4.77	2.96	-2.66
2	3	201189	Aaron Gray	41	12.95	1.37	2.41	0.566	0.00	0.00	...	3156	Aaron Gray	0.99	0.04	0.96	0.60
3	4	1733	Al Harrington	73	22.80	3.85	9.25	0.416	1.60	4.49	...	2591	Al Harrington	-0.54	-0.99	0.45	0.35
4	5	201143	Al Horford	77	35.11	6.66	11.96	0.557	0.03	0.05	...	2923	Al Horford	0.98	-0.01	1.00	1.05

5 rows × 34 columns

7.6 - Players data frame after transforming it to have the same dimensions as the master dataframe we used to run our regression models

	playerId	playerName	season	team	team_score_home	assists_home	blocks_home	defensive_rebounds_home	fast_break_points_home
0	201985	AJ Price	NaN	NaN	-0.600666	0.011513	-1.005770	-1.136435	0.0
1	201166	Aaron Brooks	NaN	NaN	0.127857	0.932836	-0.937475	-1.215836	0.0
2	201189	Aaron Gray	NaN	NaN	-1.162766	-0.963066	-0.391119	-0.223320	0.0
3	1733	Al Harrington	NaN	NaN	0.082006	-0.435835	-0.732592	0.219059	0.0
4	201143	Al Horford	NaN	NaN	0.900533	0.666557	1.316244	2.175734	0.0
...	...	...	...	...	...	...	...	...	...
6557	1629139	Yuta Watanabe	NaN	NaN	-0.755201	-0.728741	-0.391119	-0.688385	0.0
6558	1628380	Zach Collins	NaN	NaN	0.272203	0.352349	0.724358	0.803226	0.0
6559	203897	Zach LaVine	NaN	NaN	2.520605	1.092603	-0.527708	0.462934	0.0
6560	1630192	Zeke Nnaji	NaN	NaN	-0.809543	-0.989694	-0.072412	-0.988975	0.0
6561	1630533	Ziaire Williams	NaN	NaN	-0.733124	-0.664834	-0.687062	-0.807487	0.0

4877 rows × 151 columns

7.7 - Player coefficients after standardizing the data around mean 100

	playerId	playerName	ridge_normalized	lasso_normalized	elastic_net_normalized
0	201985	AJ Price	95.0	89.0	96.0
1	201166	Aaron Brooks	100.0	88.0	97.0
2	201189	Aaron Gray	97.0	104.0	92.0
3	1733	Al Harrington	113.0	99.0	108.0
4	201143	Al Horford	106.0	123.0	105.0
...	...	...	...	...	...
6557	1629139	Yuta Watanabe	92.0	94.0	94.0
6558	1628380	Zach Collins	110.0	108.0	101.0
6559	203897	Zach LaVine	125.0	95.0	124.0
6560	1630192	Zeke Nnaji	96.0	98.0	91.0
6561	1630533	Ziaire Williams	110.0	92.0	107.0

4877 rows × 5 columns

7.8 - Coefficients from Ridge, LASSO, and ElasticNet Models in predicting total wins. Please refer to 'model_coefficients.csv' in the GitHub data repository for the full list of the coefficients.

Feature	Lasso Coefficient	Ridge Coefficient	ElasticNet Coefficient
team_score_home	0.892144429	3.342493623	1.758066719
assists_home	0	0.632690905	0.651591674
blocks_home	0	-0.104929571	-0.349812654
defensive_rebounds_home	0	1.873159256	0.895915476
fast_break_points_home	-0.014331635	-0.252971954	-0.180730967
field_goals_made_home	0	3.460160551	2.039043794
field_goals_attempted_home	0	0.922795422	0.117862326
flagrant_fouls_home	0	1.376076059	-0.057847736
fouls_home	0	-0.084456647	0.146817922
free_throws_made_home	0	0.668774631	0.670181964
free_throws_attempted_home	0	0.53094126	0.230341494
offensive_rebounds_home	0	-0.359642025	0.193660916
points_in_paint_home	0	-0.360501341	-0.157967455
steals_home	0	0.348839313	0.162844659
team_turnovers_home	0	-0.560479761	0.930444865
technical_fouls_home	0	0.145173649	0.258508117
three_point_field_goals_made_home	0	2.396279798	1.439118932
three_point_field_goals_attempted_home	-0.05188102	0.102592941	0.186388776
total_rebounds_home	0	0.840244913	-0.423771078
total_technical_fouls_home	0	0.145173649	0.28768047
turnover_points_home	0	-0.006184319	-0.418170159

7.9 - Optimized roster for seasons 2010 to 2022, maximizing sum of player coefficients generated from Ridge, ElasticNet and LASSO models

```
Roster for ridge_normalized:
      playerId          playerName position
4529      2216        Zach Randolph   Center
38911   201935         James Harden    Guard
40864   201939        Stephen Curry    Guard
42433  1628415        Dillon Brooks  Forward
43124  1628991    Jaren Jackson Jr.   Center
43476  1626157  Karl-Anthony Towns  Forward
49066   203897           Zach LaVine    Guard
53619  1630217         Desmond Bane    Guard
53684  1628378     Donovan Mitchell    Guard
54865  1630163          LaMelo Ball    Guard
Solve value for ridge_normalized: 1356.0
```

```
Roster for elastic_net_normalized:
      playerId         playerName position
4529      2216      Zach Randolph   Center
27552   201939      Stephen Curry    Guard
38911   201935       James Harden    Guard
40473   202331        Paul George    Guard
43124  1628991  Jaren Jackson Jr.   Center
46271  1628415      Dillon Brooks  Forward
50002  1630217       Desmond Bane    Guard
50778  1628369       Jayson Tatum  Forward
53684  1628378   Donovan Mitchell    Guard
54865  1630163        LaMelo Ball    Guard
Solve value for elastic_net_normalized: 1347.0
```

```
Roster for lasso_normalized:
      playerId             playerName position
2503    201567             Kevin Love  Forward
4519      2216          Zach Randolph  Forward
5760      2730          Dwight Howard  Forward
6813      2746             Josh Smith    Guard
19551   201599         DeAndre Jordan  Forward
29792   202355       Hassan Whiteside   Center
37262   203083         Andre Drummond   Center
42794   203507  Giannis Antetokounmpo    Guard
45823   203991           Clint Capela   Center
52267   203497            Rudy Gobert   Center
Solve value for lasso_normalized: 1391.0
```

7.10 - Optimal rosters for teams in 2023 with $200,000 in salary

```
Roster for ridge_normalized:
     playerId         playerName position
56     201572        Brook Lopez   Center
146   1630217       Desmond Bane    Guard
151   1628415      Dillon Brooks    Guard
156   1628378   Donovan Mitchell    Guard
163   1629130    Duncan Robinson  Forward
257   1628991  Jaren Jackson Jr.  Forward
374   1630568         Luka Garza   Center
409    201144         Mike Conley    Guard
485    200752           Rudy Gay    Guard
580   1630533    Ziaire Williams  Forward
Solve value for ridge_normalized: 1200.000000000001
```

```
Roster for elastic_net_normalized:
     playerId       playerName position
56     201572      Brook Lopez   Center
144    201565      Derrick Rose    Guard
146   1630217      Desmond Bane    Guard
163   1629130  Duncan Robinson  Forward
178    201569      Eric Gordon    Guard
183   1630596      Evan Mobley   Center
228   1629630        Ja Morant    Guard
369      2544     LeBron James  Forward
409    201144      Mike Conley    Guard
485    200752         Rudy Gay    Guard
Solve value for elastic_net_normalized: 1189.0
```

```
Roster for lasso_normalized:
     playerId          playerName position
8      201143           Al Horford  Forward
132   1629028        Deandre Ayton   Center
242   1631105          Jalen Duren   Center
313    203944        Julius Randle  Forward
322   1626157  Karl-Anthony Towns  Forward
341    201567           Kevin Love  Forward
423   1629650          Moses Brown   Center
450   1629052       Oshae Brissett    Guard
496   1630567       Scottie Barnes    Guard
566   1631117       Walker Kessler   Center
Solve value for lasso_normalized: 1230.0
```

7.11 - Optimal free agent and trade targets in 2023 for teams with $30,000 in cap space

```
Roster for ridge_normalized:
     playerId       playerName position
146   1630217     Desmond Bane    Guard
163   1629130  Duncan Robinson  Forward
374   1630568       Luka Garza   Center
580   1630533  Ziaire Williams  Forward
Solve value for ridge_normalized: 476.0


Roster for elastic_net_normalized:
     playerId       playerName position
144    201565     Derrick Rose    Guard
183   1630596      Evan Mobley   Center
485    200752         Rudy Gay    Guard
580   1630533  Ziaire Williams  Forward
Solve value for elastic_net_normalized: 468.0


Roster for lasso_normalized:
     playerId      playerName position
341    201567      Kevin Love  Forward
423   1629650     Moses Brown   Center
496   1630567  Scottie Barnes    Guard
566   1631117  Walker Kessler   Center
Solve value for lasso_normalized: 508.0000000000001
```

7.12 - Regression showing that one lag of total wins is statistically significant in predicting total wins

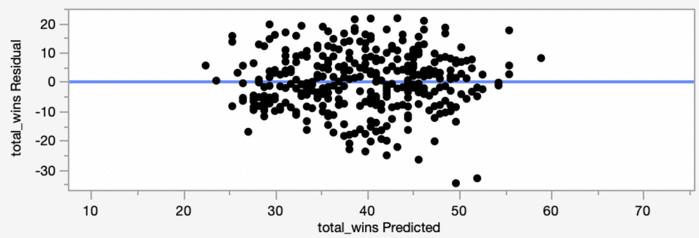
Response total_wins

Effect Summary

Lack Of Fit

Source	DF	Sum of Squares	Mean Square	F Ratio
Lack Of Fit	51	4940.351	96.8696	0.9833
Pure Error	301	29653.798	98.5176	**Prob > F**
Total Error	352	34594.148		0.5110
				Max RSq
				0.4370

Residual by Predicted Plot



Summary of Fit

RSquare	0.343228
RSquare Adj	0.341362
Root Mean Square Error	9.913568
Mean of Response	39.56497
Observations (or Sum Wgts)	354

Analysis of Variance

Source	DF	Sum of Squares	Mean Square	F Ratio
Model	1	18078.857	18078.9	183.9547
Error	352	34594.148	98.3	**Prob > F**
C. Total	353	52673.006		<.0001*

Parameter Estimates

Term	Estimate	Std Error	t Ratio	Prob>|t|	VIF
Intercept	16.665604	1.768678	9.42	<.0001*	.
lag 1	0.5787375	0.04267	13.56	<.0001*	1

Effect Tests

Effect Details