NBA Team Win Percentage Prediction

Math 156 Final Project

Chinmay Varshneya, Khang Nguyen, William Sun, Kanzah Jamil

Spring 2024

Contents

1 Introduction

At the end of the day, every team in the National Basketball Association (NBA) is looking to win as many games as possible so they may make the playoffs and preferably do so with a high seeding. Fortunately, the league rigorously documents the statistics of each team, allowing us to understand relationships in statistical trends and ultimately make predictions.

Our goal is to predict the win percentage of a team given their per game stats both regarding their own performance as well as the average performance of their opponents. In order to do this we took historical data then trained and optimized several regression models. Ultimately we selected the best performing one for the test set. Such a model will be useful for teams that are looking to predict their performance mid-season to determine what their record will be which is extremely relevant for a team's playoff implications. As the NBA regular season is very long with each team playing 82 games, this means that relatively early into the season a team will have a decent sample size to calculate their average statistics per game. Using our model, they can forecast what their win percentage will look like for the rest of the season and therefore look to make the needed adjustments to find success.

2 Data

Our dataset was sourced from https://www.basketball-reference.com/, each record representing aggregated game statistics for an individual team across a single season. The dataset spans every season from 1989 to 2024 and can be accessed at the following link: https://stathead.com/tiny/0xXHR.

In total, there were 36 columns/variables. Each row contained the following per game stats: MP (minutes played), FG (field goals scored), FGA (field goals attempted), 2P (2 point field goals scored), 2PA (2-point field goals attempted), 3P (3 point field goals scored), 3PA (3 point field goals attempted), FT (free throws scored), FTA (free throws attempted), ORB (offensive rebounds), DRB (defensive rebounds), TRB (total rebounds), AST (assists), STL (steals), BLK (blocks), TOV (turnovers), PF (personal fouls), PTS (points scored). A row also contained the following statistics, the O indicating opposition, with the same correspondence as above: O_FG, O_FGA, O_2P, O_2PA, O_3P, O_3PA, O_FT, O_FTA, O_ORB, O_DRB, O_TRB, O_AST, O_STL, O_BLK, O_TOV, O_PF, O_PT, providing a comprehensive overview of each team's performance with regards to other teams' performances.

Finally, each row is identified with a team name, their rank, and season column, along with columns for total games played, wins, losses, and win percentage. Since the games played in a season is not constant (for instance, in the 2020 season, the regular season was

truncated due to COVID interruptions), we decided to make the column we sought to target in various models the team's win percentage.

After implementing the described adjustments and considerations, we proceeded under the following assumptions:

- The win percentage is a consistent measure of success across different NBA seasons, despite variation in the number of games played due to external factors.
- The findings from our analysis can be generalized across all NBA teams.
- Each season's data is independent of the others; one season's outcome does not influence another.

	Rk	Season	Team	G	W	L	W/L%	MP	FG	FGA	...	O_FTA	O_ORB	O_DRB	O_TRB	O_AST	O_STL	O_BLK	O_TOV	O_PF	O_PTS
0	1	2023	ATL	82	36	46	0.439	242.1	43.0	92.5	...	21.8	10.6	33.6	44.2	28.2	7.8	5.6	14.1	19.4	120.5
1	2	2023	BOS	82	64	18	0.780	241.8	43.9	90.2	...	17.3	11.1	32.3	43.3	24.9	6.2	3.7	12.0	17.3	109.2
2	3	2023	CHO	82	21	61	0.256	240.6	40.0	87.0	...	20.7	10.6	34.8	45.4	28.7	7.1	4.8	13.6	17.5	116.8
3	4	2023	CHI	82	39	43	0.476	243.7	42.0	89.5	...	21.8	10.1	33.3	43.4	27.9	6.8	4.9	14.0	18.8	113.7
4	5	2023	CLE	82	48	34	0.585	241.5	41.8	87.2	...	21.0	10.0	32.6	42.7	25.3	7.7	5.0	13.6	18.7	110.2

Figure 1: Data head

2.1 Data Cleaning and Preprocessing

We took several measures in preparing the raw dataset into data that is ready to be used in modeling. We began by dropping the following columns from our design matrix as they are either irrelevant, collinear to, or directly affecting the win percentage: team name, their rank, and season column, along with columns for total games played, wins, and losses. Next, we cleaned the data by imputing missing data with the mean of the column. For predictive modeling, we focused on 'Win/Loss Percentage' (`W/L%`) as our target variable, which serves as a direct indicator of a team's success in a season. The remaining columns were used as features after standardizing them using StandardScaler, thereby transforming each feature to have a mean of 0 and a standard deviation of 1. This ensured they all had similar scale which is a necessary step for PCA and an important preprocessing step for using the data to train the models. Finally, we split the data into 80% train and 20% validation for modeling, after having extracted teams' data for the most recent season to serve as our test set.

2.2 PCA

In order to further improve efficiency in modeling, we decided to apply Principal Component Analysis (PCA) to our data. Naturally in the data, there would be obvious collinearity

issues (2-point field goals scored and 2-point field goals attempted, offensive rebounds and total rebounds, etc.), so PCA would alleviate these issues. Also, by reducing the amount of dimensions, we would mitigate the Curse of Dimensionality – the notion that the higher the number of dimensions, the less dense the data becomes–to a lesser degree, which would certainly help modeling. In the above graph, we can see the variance as a function of the number of dimensions in the data. Applying the elbow method, we ultimately selected a value of `n = 10`.

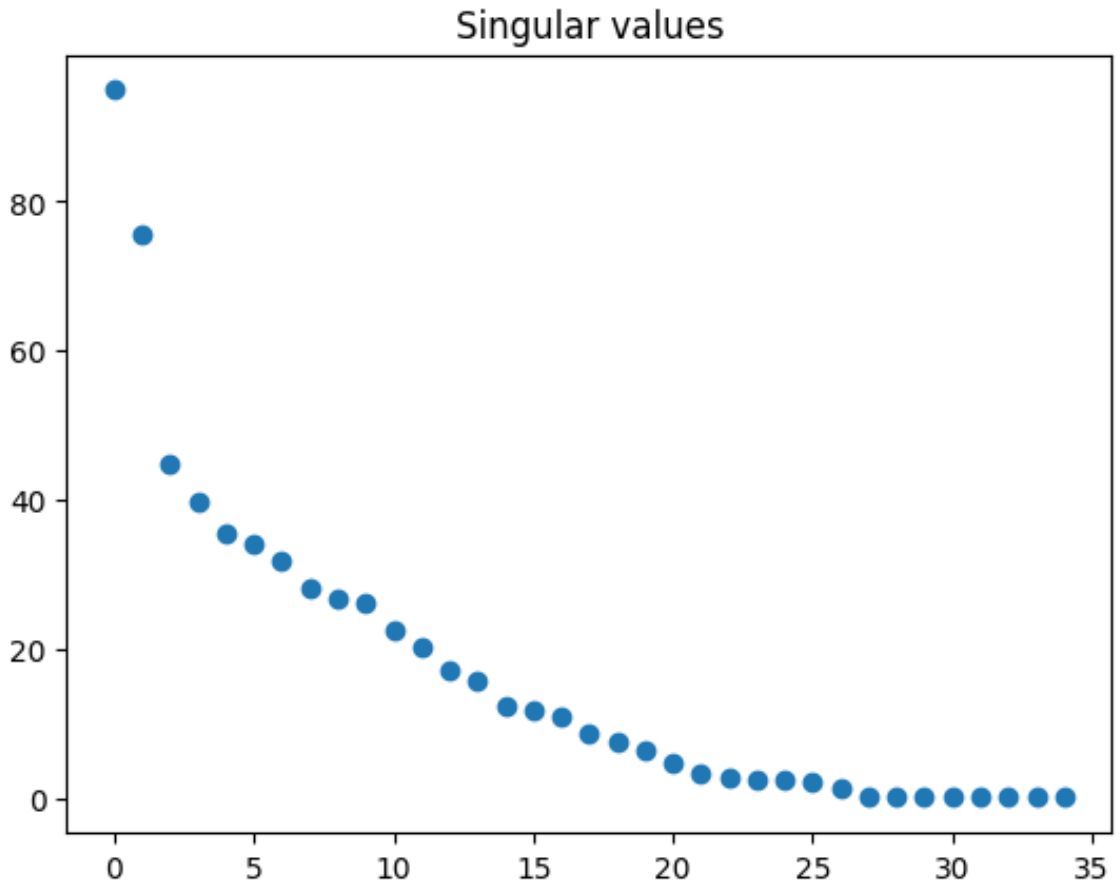

Figure 2: Singular values

Further highlighted is our decision to select `n = 10` as the new number of dimensions. At a relatively lower number than the original number of variables (35), we still retain above 90% of the variance, hence retaining an adequate amount of generalization and meaningful insight of the original data.

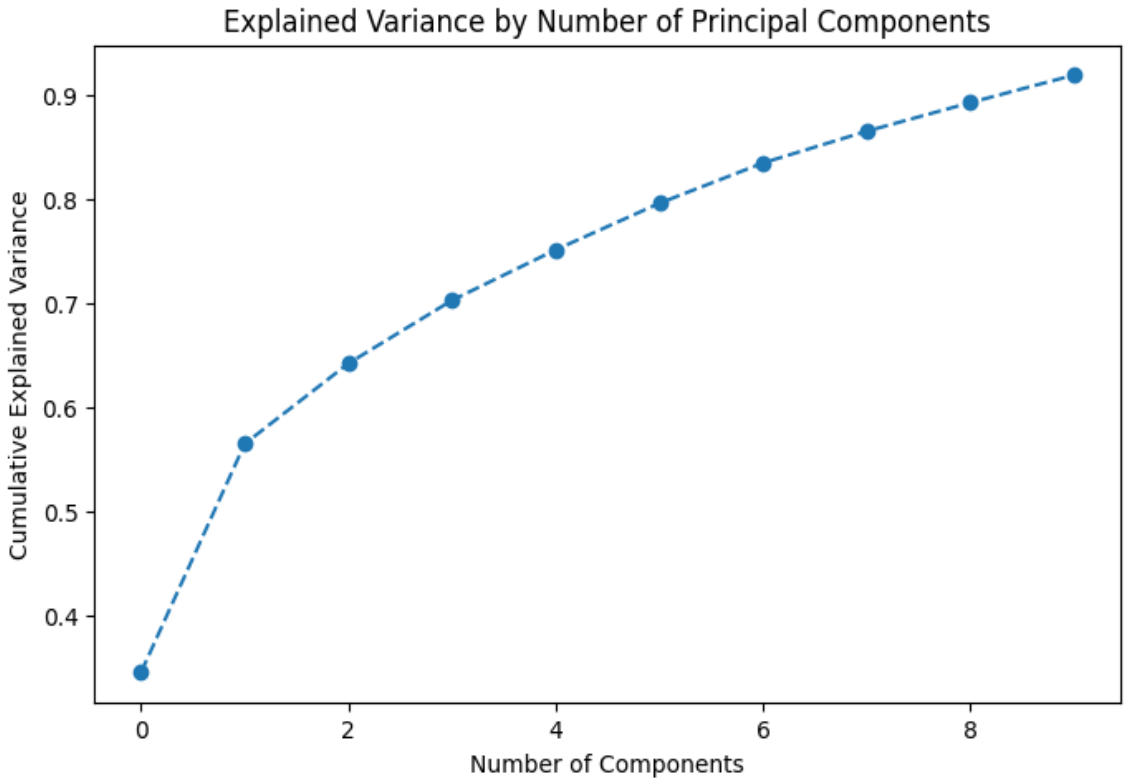

Figure 3: Variance explained

We can further inspect the loadings for the 10 principal components which we chose.

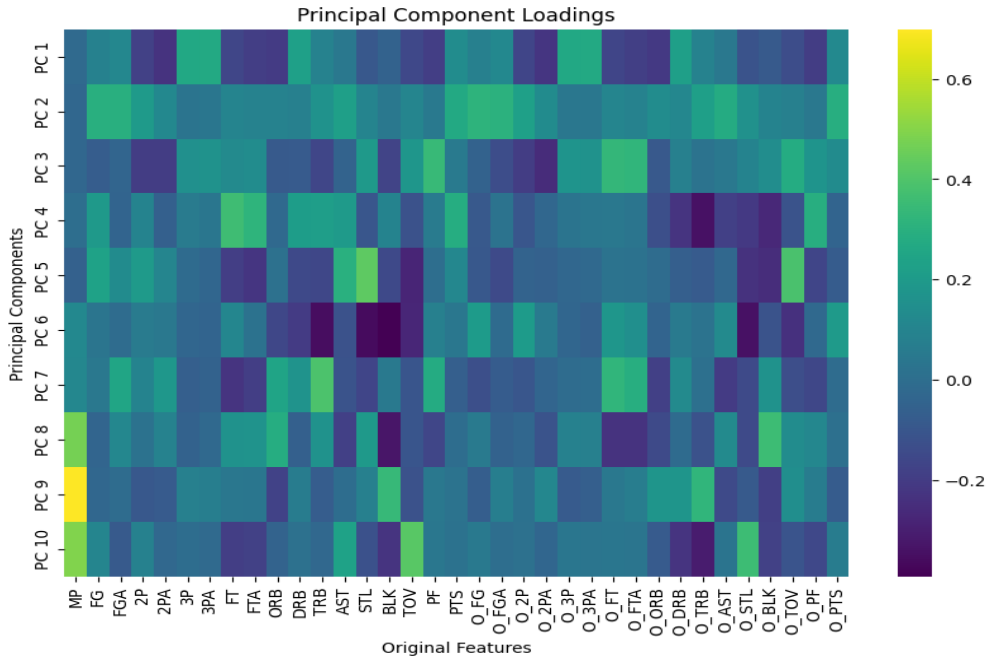

Figure 4: Latent variables

The first principal component at the top of the graph contains the largest amount of variance/information in the data. Inspecting the loading reveals that it, predictably, provides similar weight to 3 pointers made and 3 pointers attempted as well as opponent 3 points made and 3 pointers attempted. This component empirically describes the three point differential between the two teams. The second principal component adds larger loading to the overall number of field goals and rebounds which empirically describes possessions.

3 Models

We implemented a diverse set of regression models to analyze NBA team performances. Each model was chosen to handle specific types of data characteristics and to test different relationships within the data:

- Linear Regression: Assumes a linear relationship between the independent variables and the dependent variable, providing a baseline for performance comparison.
- Ridge Regression: Introduced to mitigate the multicollinearity problem among the basketball statistics by adding a degree of bias through adding a L^2 regularization term.
- Lasso Regression: Incorporates L^1 regularization to get some sparse weights, reducing the number of variables included in the final model, and helps in identifying the most impactful predictors that influence the win percentage.
- Elastic Net: Combines the properties of both L^1 and L^2 regularization to not only select influential predictors but also manage multicollinearity among the variables, providing a robust model.
- Decision Tree Regression: Offers a non-linear approach that can model complex patterns and interactions, without requiring data transformation, which might be missed by linear models.
- Random Forest Regression: Able to handle a large number of features and complex interactions making it suitable for this dataset, which includes a broad range of basketball statistics, without overfitting to the noise within the data.
- Gradient Boosting Regression: Incrementally builds models to minimize predictive errors by focusing on residuals from previous iterations. It employs gradient descent techniques to optimize a loss function, typically Mean Squared Error (MSE) or Mean Absolute Error (MAE).
- Support Vector Regression: Applies a non-linear kernel to find the relationships between the various basketball metrics and the response.

- K-Nearest Neighbors Regression: Looks for the k-nearest labeled data points and averages their values for prediction. This can capture subtle trends in team performance that might be season-specific.

We split the dataset into training, validation, and test sets, using the most recent season's data as the test set to evaluate the models' predictive power in a current context. The validation set was used to determine the performance of the models before any hyperparameter optimization was conducted. We can see which are best-performing models based on their Mean Squared Error (MSE), R-squared, and Adjusted R-squared values. These metrics helped assess the models' accuracy and the proportion of variance in the win percentages they could explain.

Applying the default models above, we obtained the following results:

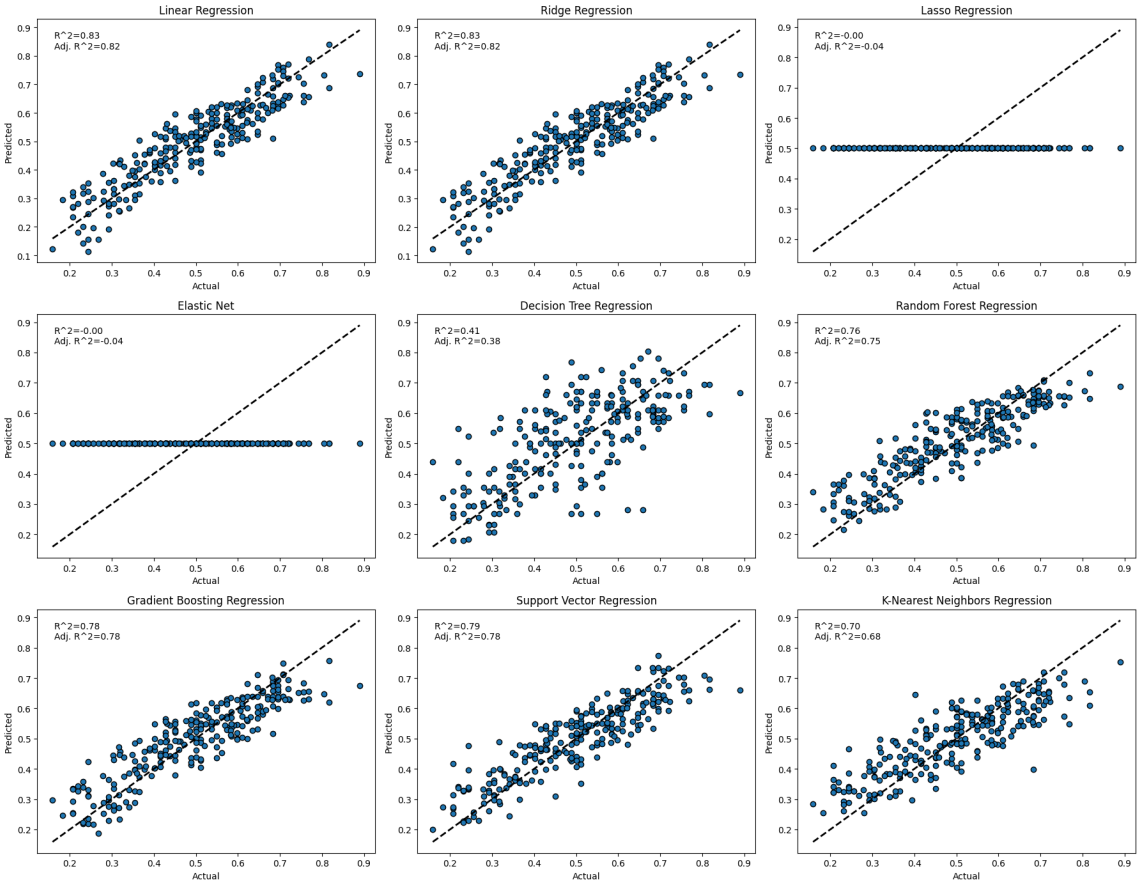

Figure 5: Models Performance

As we can see via the graphs and the quantitative results, the linear regression family of models had the best performance. Gradient boosting is a generally strong model and

also performed well. In order to further improve the models, we sought to optimize the hyperparameters in order to build more robust models.

3.1 Hyperparameter Tuning

We optimized the hyperparameters of each of the models using `GridSearchCV` which tunes them by exploring each combination of the hyperparameters. It uses cross validation by creating folds in the data, then using one for validation and the rest for training as it cycles through the folds.

An important note is that in the `ElasticNetCV` model code, the `l1_ratio` and `alpha` parameters are generarlizations of `LinearRegression` (when `alpha = 0`), `Lasso` (when `l1_ratio = 1`) and `Ridge` (when `l1_ratio = 0`). Hence we will only consider hyperparametrization of `ElasticNetCV` below.

Another important note is that changes to `n_estimators` within `GradientBoostingRegressor` does not effectively improve the MSE, probably because of possible overfitting during cross-validation. Code tests done privately have shown that increasing `n_estimators` to 200 or 400 does not increase the MSE, and increasing it to 1000 decreases the optimal MSE as well as R^2.

We applied the optimized models to the same training and validation data as before, and achieved the following results:

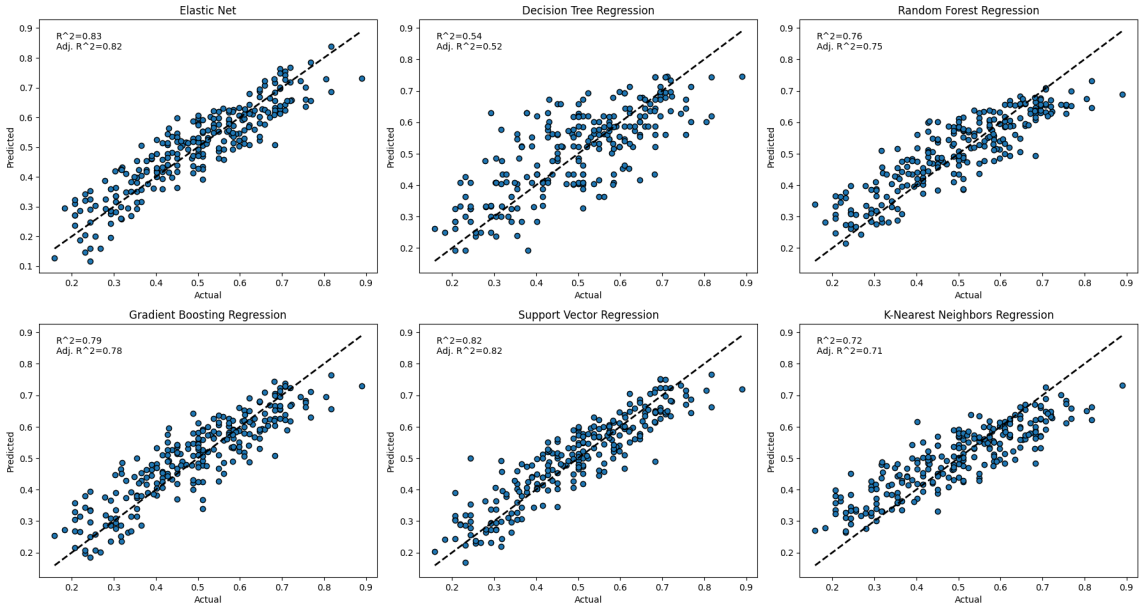

Figure 6: Model Performance after being tuned

With the tuned hyperparameters, in terms of the visualizations of actual vs. predicted win percentages and the MSE, R-squared, and adjusted R-squared values, we naturally see better model performance in all models from before. Again, the elastic net model (remember, essentially a wrapper for Lasso regression) minimized the error between real and predicted values while having the highest R-squared, thus being the "best" model in this case.

4 Model on Test Data

When applied to our test case of the 2023-24 NBA season we were fortunate to have the actual results of the recently completed season available to observe the performance of the final model.

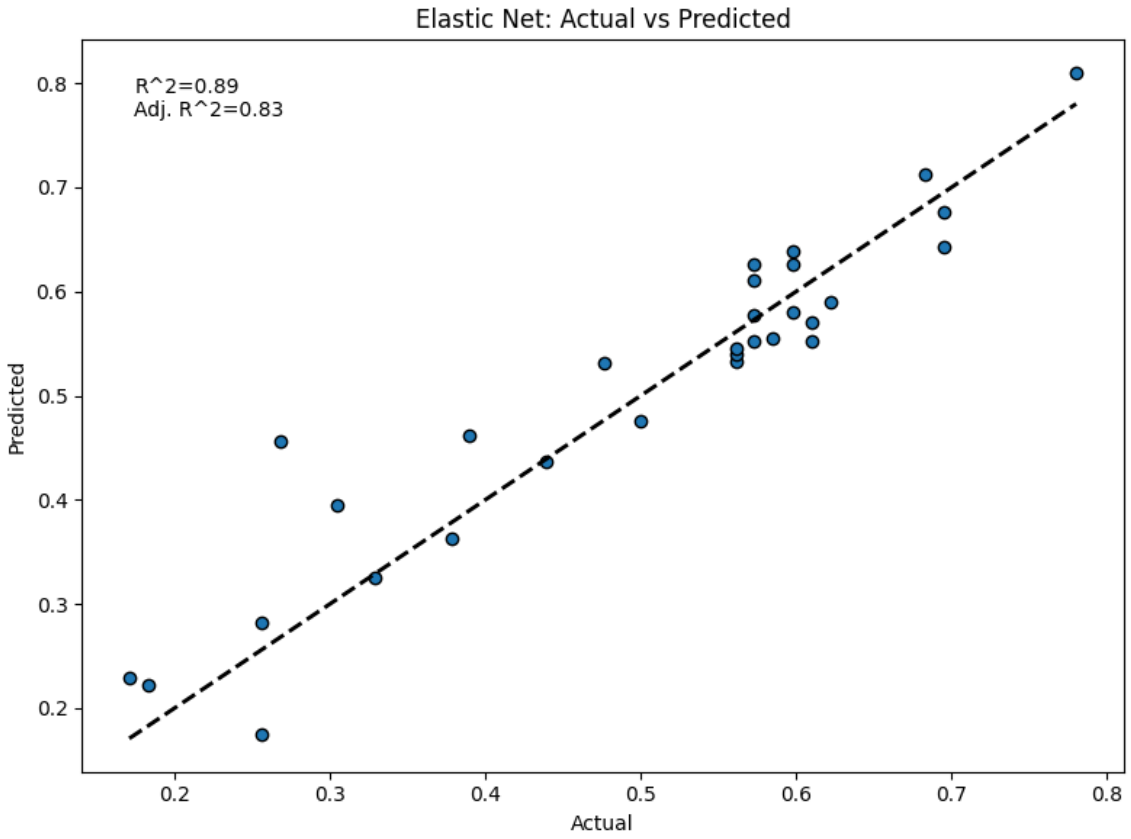

Figure 7: Elastic Net (LASSO) Regression on Test Set

Due to its performance on the validation set and the overall strength of ensemble methods, we also decided to use Gradient Boosting Regression on the test set and received equally encouraging results.

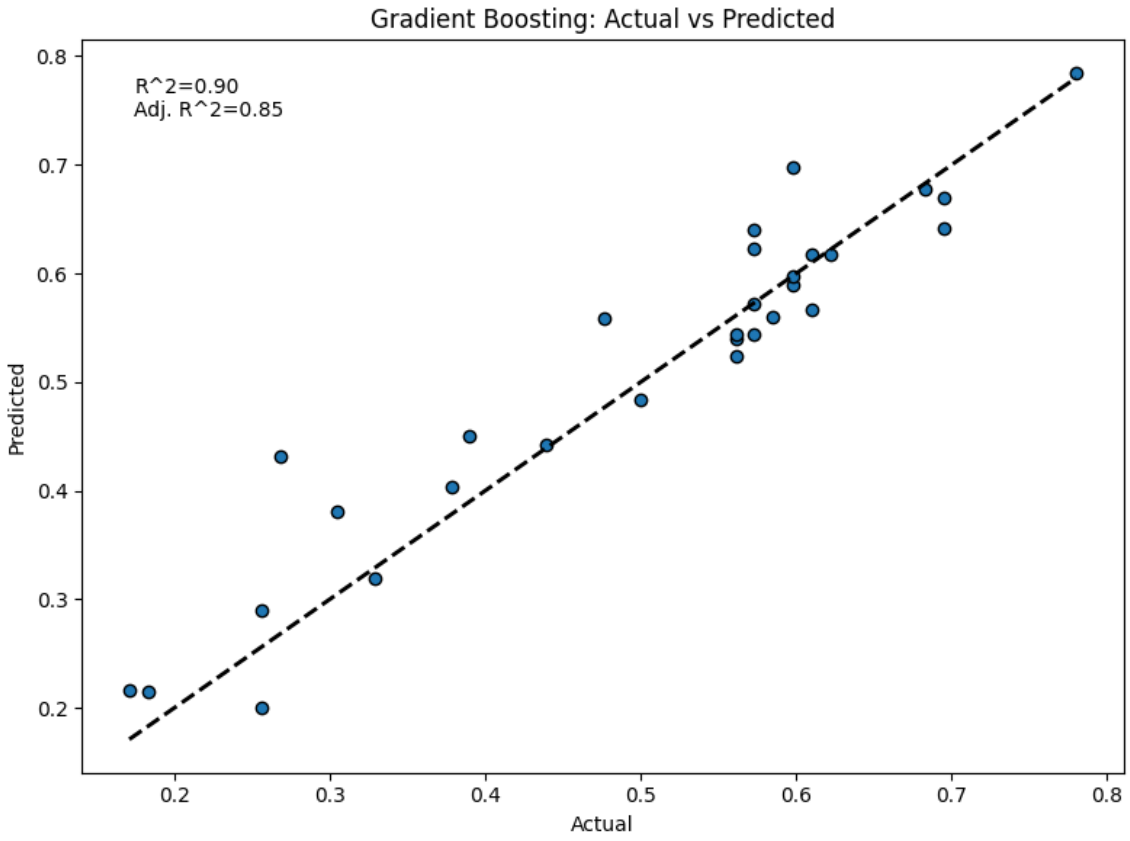

Figure 8: GB Regression on Test Set

4.1 Discussion and Analysis

We will now focus our discussion on the models themselves to explain their performance and their strengths. Because of the application of PCA, which mitigates the multi-collinearity issues in the original data, the Lasso model performs well due to its assumptions. The data was relatively linear due to the strong performance of the optimized models. This makes intuitive sense as an increase in a scoring statistic could lead to a proportional increase in win rate.

However, it is worth noting that the data inherently might not be totally independent. As each row contains a certain team's stats as well as the opposition stats, it is debatable that the data truly is independent. However, given the strong performance of the optimized models, it appears that any possible dependence between observations may not significantly impact the performance of the Lasso model.

Gradient boosting works well here because there is no need to extrapolate the data. Furthermore multi-collinearity, which may still be prevalent despite PCA, does not impact

tree-based models like gradient boosting regression. Since gradient boosting is an ensemble method, as the weak learners iterate, they can begin to learn specific features and relationships in the data.

5 Mathematical Overview of Models

Ultimately we ended up using a Lasso regression model. An elastic net regression model is a linear regression model that includes a regularization term in the error function of the model so as to encourage weight decay. This means the term looks to reduce the magnitudes of the weights of the model so as to combat overfitting. Commonly used regularization terms are the vector p-norms. Elastic net combines both the L^1 and L^2 norms in order to handle complexity better. Ultimately, since we ended up tuning the hyperparameter `l1_ratio` ≈ 1, we are only using the L^1 norm for regularization making it essentially the Lasso model. The Lasso model has its loss function defined as the following:

$$E(w) = \frac{1}{2N}\sum_{i=1}^{N}(y_i - w^T\phi(x_n))^2 + \lambda\|w\|_1,$$

where y_i are the target variables and $w^T\phi$ reflects the weights multiplied by the basis functions applied to the data. However, the more important hyper-parameter is `alpha` which is equivalent to λ. Since $\alpha \approx 0.027$, this means we have a smaller magnitude for the regularization term. Hence we can rewrite

$$E(w) \approx \frac{1}{2N}\sum_{i=1}^{N}(y_i - w^T\phi(x_n))^2$$

This demonstrates the model is not overly prone to overfitting. Our dimensionality reduction likely aided in this regard as we have only 10 latent variables as opposed to the original 35.

The second best model was gradient boosting which also performed well on the validation data. Below we describe the general algorithm for `GradientBoostingRegressor`, likely based on the work of Friedman (2001).

Algorithm for Gradient Boosting

It starts with a base model, typically a one-level decision tree. The initial predictions from this model are used to calculate residuals and address the shortcomings of the current model.

1. **Initialization:** The initial model $F_0(x)$ is usually a shallow decision tree, providing a baseline prediction.

2. **Iterative Improvement:**

   (a) For each iteration $m = 1$ to M, where M is the number of boosting stages:

      i. Compute the residuals $r_{i,m} = y_i - F_{m-1}(x_i)$, where y_i are the true target values, and x_i are the input features.

      ii. Fit a new model $h_m(x)$ to predict these residuals.

      iii. Find the optimal coefficient ρ_m by minimizing the loss function over the residuals:

$$\rho_m = \arg\min_{\rho} \sum_{i=1}^{N} L(y_i, F_{m-1}(x_i) + \rho h_m(x_i))$$

      where L is a loss function, typically the mean squared error (MSE).

      iv. Update the current model by adding the weighted contribution of the new model:

$$F_m(x) = F_{m-1}(x) + \rho_m h_m(x)$$

3. **Final Model:** After M iterations, $F_M(x)$ represents the final model.

6 Limitations

There are some limitations and considerations for the model when looking to generalize. The first is in regard to the training data and changing trends in the NBA. We considered roughly the last 25 years in order to develop the model, however the nature of the game has changed significantly during this period and is reflected in many of the per game stats. Take, for example, the three point differential, which we empirically described as the first principle component of our model. When studying three pointers over the last 25 years we can see a rapid and dramatic year over year increase in such shots, with exceptions for the 1994-1997 seasons.

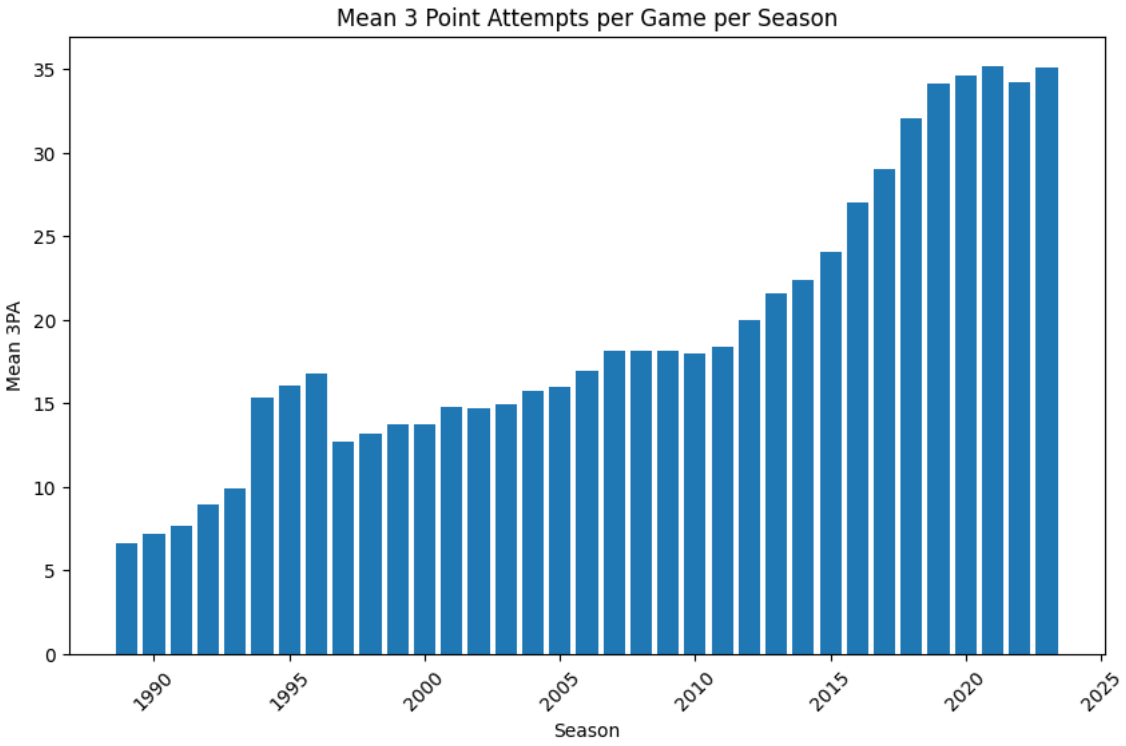

Figure 9: Three point trend

This creates a challenge in terms of finding an appropriate cutoff point for the data such that the training data represents the modern style of NBA basketball while also providing the model with an adequate amount of data to train it. As such, we ran our models with several cutoffs in the data: 2015-2016, 2010-2011, 2000-2001, 1997-1998, and 1990-1991. Ultimately there was no major change in the performance of the models based on such criteria so we ended up using the entire data set. It is highly notable *not* to train the data on any period before 1980, however. This is because the three point shot itself did not exist in the NBA at that point and therefore, the entire nature of the sport was drastically different from both a statistical and strategic point of view. This can also be seen during the exceptional seasons mentioned above, due to the league deciding to move the 3-point line in, which was reverted later. As mentioned above however, the results for the cutoff dataset did not affect much.

Another challenge lies in the complexity of the game of basketball itself. While the collected statistics definitely highly describes teams' tendencies in gameplay and differences in skill levels, as shown by how well we managed to predict teams' win rates, we have yet to be able to dive deeper into each team's performances as well as how a team will perform against another. Further analysis into this issue will require a much deeper analysis into

each player as well as their dynamics together, which is beyond the scope of this paper. However, test runs to model teams' performances over the season have shown that as long as we assume stabilized teams over the summer, we can get a *surprisingly* accurate result of *individual* team performance in the year, which is a fascinating point we would consider revisitingat a later date.

7 Conclusion

Our model allows for teams to leverage their statistical performance in the season to approximate what their win rate will be for their future games. This predictive capability becomes particularly critical as teams approach the playoff cut-off, enabling management to make informed decisions that could enhance their chances of securing a better seed or even qualifying for the playoffs. Furthermore, despite the dynamic nature of NBA gameplay over the years, our robust models assure that the model can adapt to evolving game strategies and player developments.

8 Repository Link

All available data and code can be accessed at https://github.com/ChinmayVar/FinalProject156.

References

[1] Jerome H. Friedman. Greedy function approximation: A gradient boosting machine. In *The Annals of Statistics, volume 29, number 5*, pages 1192–1194. JSTOR, 2001.

[2] Marco Taboga. Gradient boosting. In *Lectures on Machine Learning*. 2021. https://www.statlect.com/machine-learning/gradient-boosting.

[3] Hui Zou and Trevor Hastie. Regularization and variable selection via the elastic net. In *Journal of the Royal Statistical Society Series B: Statistical Methodology*, volume 67, issue 2, pages 301–320. April 2005.